The ARLISS Project
Final Report
Senior Capstone

Steven Silvers, Paul Minner, Zhaolong Wu, Zachary DeVita
Capstone Group 27
2016-17 Academic Year

Abstract

This document details our progress while developing software for the ARLISS competition over the duration of the academic year. This includes all original documentation as well as all revisions made to each of the documents, all weekly blog posts from each member of the team, the finalized version of our team poster, and a personalized essay detailing what each member of the team has learned from this project.

Table of Contents

I. The ARLISS Project

A. Introduction

For our capstone project, our team of computer scientists have spent the course of three terms developing software to autonomously navigate a rover to a specified set of coordinates. This rover is intended to compete in the ARLISS competition held in September. ARLISS, standing for "A Rocket Launch for International Student Satellites," is an international competition where teams of engineers from around the world will bring their rovers to compete. To compete in this competition the rover must be small enough to fit into a soda can and must have a weight of less than or equal to that of a soda. This ARLISS capstone project is one of the 5 rocketry projects that involves multidisciplinary teams across students in MIME and EECS department, these projects are all proposed by Nancy Squires, who is our client.

In this competition, our team's rover will be launched by a rocket into the air, and will be ejected from the rocket at approximately 12,000' AGL where it will use a parachute to fall safely to the surface of the Earth. Once the rover reaches the ground it must detach from the parachute and circumvent any potential restraint created by the canopy or suspension lines of the parachute. When the rover is free from the parachute and begins its expedition, the rover will rely on GPS to direct it to the coordinates while using the onboard camera to detect obstacles in its path. GPS is only accurate for approximately 7.8 meters so, when the rover is near the end of the expedition and GPS becomes no longer accurate, the rover will begin to search for the pole that marks the end of the trek. At the base of the pole is an orange traffic cone which is used for the image detection algorithm. When the cone has been detected the rover will drive towards it until contact has been made.

Our capstone team has four group members, each of us is responsible for certain tasks. Zach is our visual expert, he is in charge of converting images to binary data to determine where objects are within the image, both for obstacle avoidance and detection. Steven is our obstacle avoidance expert. He is responsible for the obstacle avoidance module. Paul is the integration expert, he is in charge of integrating all individual software modules together as well as touch the finish pole, parachute deployment and unstuck from obstacle module. Zhaolong is our navigation expert. He is responsible for our GPS navigation algorithm, which directs the rover towards the finish, and makes periodic course corrections. The team worked together flawlessly throughout the entire project. Throughout the development of the ARLISS project, our client Nancy Squires gave us total freedom on making design choices and implementation protocols, she was mainly supervising the project.

As the computer science team, our job is strictly developing the software for the rover. There have been five primary components devised for the implementation of the software.

- Parachute Deployment
- GPS Navigation
- Obstacle Avoidance
- Getting Unstuck from Obstacles
- Locating and Coming in Contact with the Pole

B. Goals

The primary objective for this project is to compete in the ARLISS competition, and to have our rover finish its autonomous expedition by bumping into the traffic cone at the base of the pole which marks the finish. No team in the ARLISS competition history has yet completed this objective in the specific competition we are attempting. What distinguishes this competition from the other ARLISS competitions is the soda can restriction on the size and weight of the rover. Because of this, and the fact that we must rely on the performance of two other engineering teams, we have an alternate metric of success. If the rover is able to safely land on the ground, escape from the parachute, and successfully traverse the terrain for some period of time while navigating around obstacles, then we would consider our goals met. Battery life not lasting for the duration of the rovers trek is beyond our control, as is mechanical drawbacks or limitations. As computer scientists, our measure of success is based solely on the success of the five components listed above.

II. Requirements

A. Original Document

The following document is the original version of our software requirements for our team's project. The Requirements document provides a detailed outline of the project, the intended use and functionality of the software, and a list of all non-functional requirements.

SOFTWARE REQUIREMENTS SPECIFICATION

for

The ARLISS Project

Capstone Group 27
Version 1.0

Steven Silvers
Paul Minner
Zhaolong Wu
Zachary DeVita

Oregon State University
November 2, 2016

Abstract

This document details our plans to create software for a autonomously driven satellite, which will be launched from a rocket at 12,000 feet into the air. Upon safely landing, it must drive itself to a specific GPS location. Our plans include details about specific software modules which need to be implemented, as well as the environment our software must operate in.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to provide a thorough and verbose description of our teams software which we will be developing for use in the ARLISS International Competition; the acronym ARLISS referring to 'A Rocket Launch for International Student Satellites'. The document will explain the purpose and features of the system, as well as, objective of the system, and the constraints under which it must operate. This document is intended for any stakeholders in the project, i.e. clients, instructors, and engineering collaborators, as well as, for the software developers who will be creating the system.

1.2 Document Conventions

This document follows IEEE Format. Bold-faced text has been used to emphasize section and sub-section headings. Italicized text is used to label and recognize diagrams. Requirements will be prioritized into two categories; High-level requirements are those which are mandatory and will be specified throughout the document, while "stretch-goals" will be denoted with a heading of 'Secondary Requirements'.

1.3 Intended Audience & Reading Suggestions

This document is to be read by the software development team, the extended team of engineers who will be collaborating on this project, course instructors, and the client for the project. The document will be publicly accessible as well, and may be read by anyone viewing our exhibit. The document can, and will, be read by our peers in software development, as well as, engineers of all disciplines.

Overall Description – The general public, as well as, the media may find the overall description to be most pertinent, and to give an adequate and effective overview of the ARLISS Project.

System Features – Developers, as well as, the extended team of engineers who are collaborating on this project will require a comprehensive understanding of the system features in order to integrate their design, and, additionally, for testing purposes.

Nonfunctional/Functional Requirements – The computer scientists developing the software, and the electrical and mechanical engineers developing the hardware will need to utilize, and adhere to, the requirements.

1.4 Project Scope

Our team of computer scientists will be developing software to operate an autonomously navigated satellite to a specified set of coordinates during the international competition. Our team's satellite will be ejected from a rocket at approximately 12,000' AGL, and fall safely to the surface of Earth with the aid of a parachute. During the estimated 15 minute "hang time" that the rocket spends in the air, there will be barometric reading taken every 1000ft by the satellite. Once the satellite reaches the ground it must detach from the parachute and circumvent any potential restraint created by the canopy or suspension lines of the parachute. Once moving, the autonomous satellite must utilize its system of cameras and radar to detect obstacles so that it may navigate around them. GPS will be necessary for the satellite to find the coordinates.

The most imperative objective of the software is to safely guide the satellite to its intended destination. This is a requisite, but the secondary objective of taking barometric readings is an additional goal which our group is confident in completing.

2. OVERALL DESCRIPTION

2.1 Product Perspective

This is a new and entirely self contained product. The software is composed of multiple independent systems which will be used at different steps in the satellite's mission. For example, once the satellite has been deployed from the rocket, the parachute deployment system will be used. These systems will be deployed in a specific order, and the timing of the deployment will be based on various sensor readings determining if the time is right to deploy the next system. The only user interface would be the graphing software used to display barometric readings obtained by the satellite during it's descent. The software would interface with many hardware components, including cameras, thermometers, servos, and various sensors. These components will serve as the software's eyes and ears. Memory and power will be constrained, so the software must be as memory and power efficient as possible.

2.2 Product Functions

The software has four main functions, which it performs at each stage of the mission. Each function is described in greater detail in section 3. Parachute deployment, change to drive mode, sensor and motor array initialization, and obstacle avoidance.

2.3 User Classes & Characteristics

The software as a whole is only useful in completing the specific challenges our specific satellite has to complete, although the individual parts of the software may be useful for other applications. Autonomous driving in particular is an upcoming field, so users who wish to create autonomous vehicles may find use from our obstacle avoidance system. Unfortunately, our obstacle avoidance system is designed specifically for navigating dry lake beds, so it's general use is limited.

2.4 Operating Environment

Since the mission is being performed only once in a specific location, the software will be optimized to work best in those conditions. The mission takes place in Black Rock Nevada, in a dry lake bed. This means the navigation software needs to deal with rocks, cracks, and tire tracks, but not much elevation change. Therefore, the navigation software will only work properly in locations similar to Black Rock Nevada. The rest of the software functions could work correctly in other conditions, nonetheless.

2.5 Design & Implementation Constraints

Space is a precious commodity on this satellite, therefore, the least amount of hardware possible will be fitted to it. This limits resources such as power, computation speed, and memory. This means the software will need to be implemented as simply and efficiently as possible while still being adequate to complete it's task. In addition, the autonomous navigation software must act quickly. If not, the satellite will get stuck, or hit an obstacle before the software can correct course to avoid it.

2.6 User Documentation

There will be a manual included with the satellite in PDF format, which details what each function of the software does, as well as it's limitations. For example, in the Obstacle Avoidance section, the manual will list the types of obstacles the software recognizes and avoids. In addition, the code itself will be well commented, so that future developers can understand what it is doing.

2.7 Assumptions & Dependencies

Since the satellite is being designed at the same time as the software which runs it, hardware specifics aren't known. We have a general idea of what the satellite should be when it's finished, but specifics such as the placement of sensors, or even which sensors will be included, isn't currently known. As the project progresses, more details will be revealed to us, and we should be able to adjust the software accordingly.

3. EXTERNAL INTERFACE REQUIREMENTS

3.1 User Interfaces

Because of the CanSat operates autonomously, the team has made the decision to not include a user-interface for the CanSat for the time being, but it is possible a user interface will be created in order to more easily test the satellite. The team evaluated all possible solutions based on the project goals, such as being cost-effective, efficient and weight/space constraints to decide a user-interface wasn't necessary.
A user-interface would, however, be useful during the testing stage. It could display the comprehensive live data that allows engineers to do quick analyze and have the full control of the CanSat.
A user-interface would also be a plus if we can display live data from various sensors during the final competition. Although since the CanSat operates autonomously, it is unnecessary to have a user-interface and will add on another communication layer. Due to the weight/space constraints, the team will evaluate the possibility of having a user-interface upon the completion date of the project.

3.2 Hardware Interfaces

Due to the fact that the mechanical and electrical teams are still under their early modeling and developing stages, we have not yet gotten any hardware specs. It is fairly straightforward to the team that there will be microprocessor(s), memory, buses, micro-controller(s), sensor(s), I/O device(s) and motor(s) involved, we just don't know the specifics. The hardware interfaces run in parallel with several electrical connections carrying parts of the data simultaneously.

3.3 Software Interfaces

This CanSat will be implemented in an object-oriented language. The team may have to design a custom operating system for the CanSat. Different classes will be created for each system of the software.

3.4 Communications Interfaces

A circuit board will perform most of the communication between the software and hardware. Communications interfaces will carry high level of intelligence and pre-defined protocols to achieve the hardware and software parallelism, to provide a secured platform for the CanSat to ensure its functionalities.

4. SYSTEM FEATURES

4.1 Parachute Deployment

A. Description & Priority

This system will detect when the satellite has been deployed from the rocket and activate the parachute module to carry the CanSat safely to the ground.

B. Stimulus/Response Sequences

This subsystem will respond to being released from the rocket, after-which it will deploy the parachute module.

C. Functional Requirements

This system just needs to activate the on-board parachute.

4.2 Transfer Control to Drive Mode

A. Description & Priority

This system will need to get the satellite ready to drive once it has landed on the ground. This involves activating various servos within the CanSat to expand the satellite from the payload container. In addition, the can shaped fairing will be shed from the satellite.

B. Stimulus/Response Sequences

Once the CanSat's sensors have determined that it has landed on the ground, the software will then activate this module.

C. Functional Requirements

This system needs to get the satellite out of the soda can container and expand the satellite's treads to the ground so that it is ready to start moving.

4.3 Sensor & Motor Array Initialization

A. Description & Priority

The driving sensors and motors need first time initialization before it can begin driving. This would include locking on to the target GPS coordinates, and preparing the sensors needed for the obstacle avoidance system.

B. Stimulus/Response Sequences

This process will begin the moment that the previous drive mode system completes its task.

C. Functional Requirements

The GPS location needs to be determined, and the sensors which will be used for the obstacle avoidance system need to be initialized.

4.4 Obstacle Avoidance System

A. Description & Priority

Once the satellite is locked on to the target GPS location, it will need to autonomously drive to the target location. This could be anywhere from less than a kilometer to fourteen kilometers depending on how the parachute is carried. There will be obstacles between where the satellite lands and the target location such as rocks and tire tracks that will need to be avoided by our satellite. This system will take input from the satellite's on-board sensors and use that data to control motor function so that the obstacles will be avoided.

B. Stimulus/Response Sequences

As soon as the sensor and motor array initialization completes, this system will begin. The system will be driving the satellite towards the target GPS location, unless it it is responding to a sensor detecting an obstacle. These obstacles include rocks, tire tracks, and other impassable terrain. The software will respond to the obstacle by finding an alternate route avoiding that obstacle.

4.5 Functional Requirements

Drive the satellite to the target GPS coordinates while avoiding any obstacles that impede the satellite's progress.

4.6 Performance Requirements

The speed at which this software reacts is of crucial importance for the obstacle avoidance system. If the software doesn't react to it's surroundings fast enough, the satellite will hit obstacles before it has a chance to avoid them. Therefore, the software must be able to react to obstacles faster than the satellite can move.

4.7 Safety Requirements

This software is entirely safe. The satellite is unable to do any kind of damage to any person or object in it's vicinity.

4.8 Security Requirements

Security is not a concern for this software. The satellite on which this software is running will only be deployed once, and the software itself is very specific for a single task.

4.9 Software Quality Attributes

This software must be thoroughly tested to ensure it will be able to complete it's task on the first try. Therefore, the software will be tested both by simulating the environment it will be running in, and using the actual satellite in environments like where it will be deployed. The simulations will be used to ensure the software behaves as inteded when faced with obstacles such as rocks and tire tracks, while the satellite tests will ensure the software can detect real obstacles using the provided sensors on the satellite.

4.10 Business Rules

Each system in the software acts independently from the other systems. The parachute deployment system must activate first, followed by the drive mode, sensor and motor array initialization, and obstacle avoidance system. Each system has a specific task to perform, then it ends and a new system begins.

5. SECONDARY REQUIREMENTS

5.1 Data Visualization

A stretch goal for this project would be to develop a system to visualize data sent back from the satellite to a remote control station. This would include data from the satellite's flight, as well as its trip across the desert.

5.2 Appendix A: Glossary

AGL: Stands for ”above ground level”

ARLISS: A collaborative effort between multiple institutions to build, launch, test and recover prototype satellites miniaturized to fit inside a soft drink can.

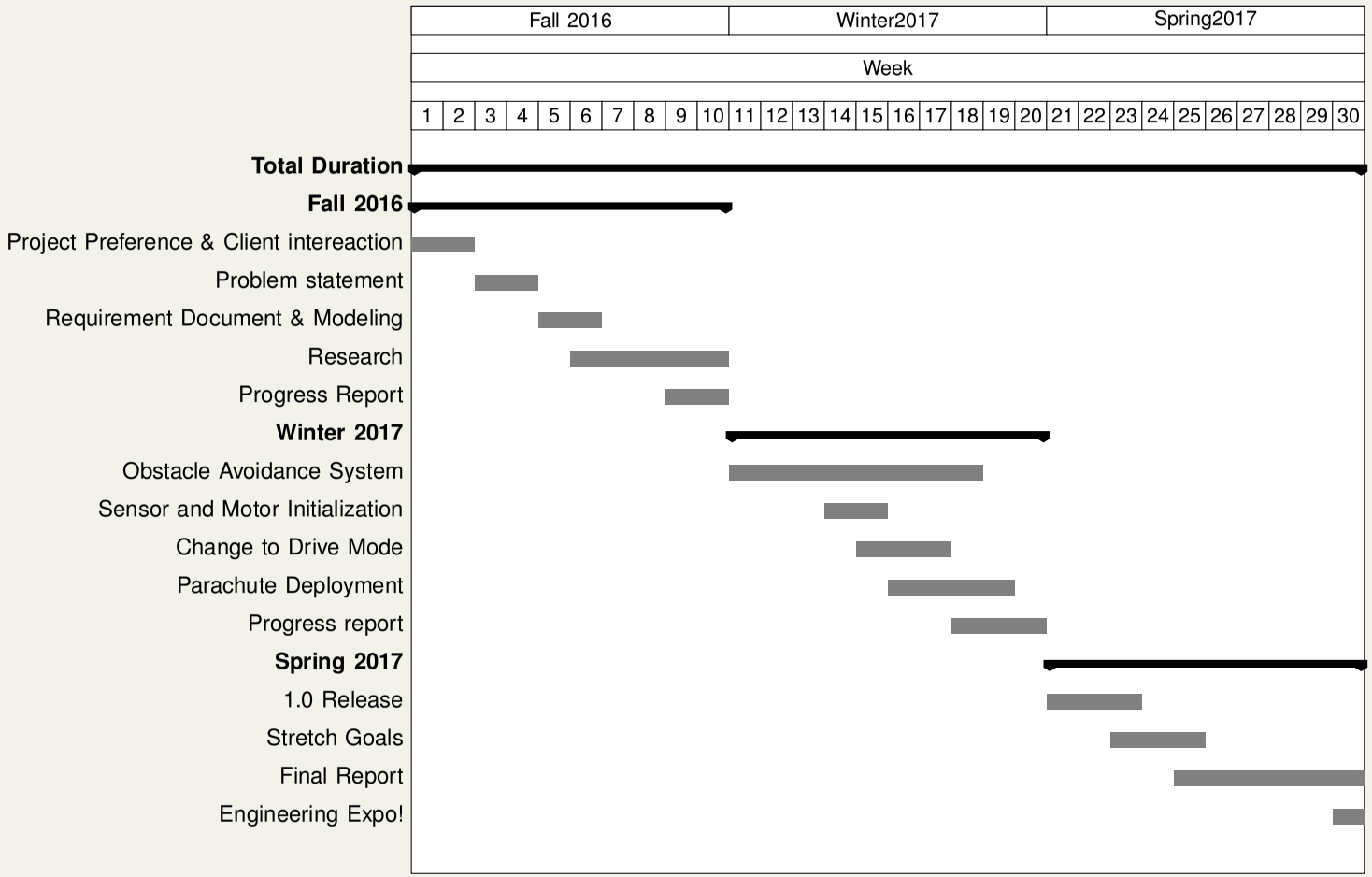
Autonomous: Used to refer to the self-driving satellite. The satellite must navigate to a GPS destination without any human intervention.

CanSat: Name given to the prototype satellites used in the ARLISS competition.

Destination: The location specified by the event organizers which the satellite must navigate to after being launched from the rocket.

Obstacle: Any object which may impede the satellite’s progress to it’s destination. This includes rocks, tire tracks, plants, and uneven terrain.

Satellite: Referred to also as ”CanSat” this is the physical device that our software system will be loaded on.


Fall 2016
Winter2017
Spring2017
Week
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Total Duration
Fall 2016
Project Preference & Client intereaction
Problem statement
Requirement Document & Modeling
Research
Progress Report
Winter 2017
Obstacle Avoidance System
Sensor and Motor Initialization
Change to Drive Mode
Parachute Deployment
Progress report
Spring 2017
1.0 Release
Stretch Goals
Final Report
Engineering Expo!

B. *Revision History*

III. Design Document

A. *Original Document*

The following document is the original version of our team's Design Document. The Design Document describes each software component in detail. This includes the purpose of each component, how it is being implemented, and how the components interacts with one another.

The ARLISS Project
Design Document
CS 461

Steven Silvers, Paul Minner, Zhaolong Wu, Zachary DeVita
Capstone Group 27, Fall 2016

Abstract

This document defines the plan for how the various pieces of the ARLISS Project will be developed and implemented. Each piece that was discussed in the previous technology review will be gone through in detail in its own section.

CONTENTS

I. Introduction

In this document we have divided our project into twelve components, and a framework for each one of these components has been designed by the corresponding member from the list below. Each section will detail how a component will operate, and how it will interact with other related components. This document is intended to provide a structure, and the architectural layout for the entire project.

Project Framework
Zachary DeVita
CMOS Image Sensing
Zachary DeVita
Ultrasonic Radar Sensor
Zachary DeVita
Obstacle Avoidance
Steven Silvers
Control Board
Steven Silvers
Motorized Tracks
Steven Silvers
Parachute Deployment
Paul Minner
Getting Unstuck from Obstacles
Paul Minner
Finding and Touching the Finish Pole
Paul Minner
Navigation System and Algorithm(s)
Zhaolong Wu
Getting Rover Unstuck When It on Its Side
Zhaolong Wu
Payload Fairing
Zhaolong Wu

II. Framework

For our overall project design and framework, our team has decided to utilize the C++ programming language. We will be writing our individual portions of code using a C compiler in order to reduce the overall overhead and energy cost of the program, but we will eventually combine the portions into a C++ framework where we will be able to utilize classes and access modifiers. The team feels strongly that utilizing classes and access modifiers for a project of this magnitude would be a huge benefit, and, since C is a subset of C++, we can get the best of both programming languages. Using C++ compilers and C++ specific tools will also reduce time spent debugging and testing [**?**].

As far as the framework for the project, we will need a series of classes to pass control of the satellite between. This is due to the fact that our satellite's journey will be comprised of four primary stages where the controls for the satellite are vastly different from each other. All of these classes governing stages of control should have a private modifier so they can't be accessed by other classes.

The first stage of the satellite's expedition will be when it is encapsulated in its can-shaped housing. This stage will take place during the period of time that the rocket is on the ground prior to launch, and it will last until some period of time after the satellite has landed safely to the ground via the parachute. Though the satellite will remain in a static state during this period, it must be turned on and ready to receive instructions.

There will be a class designated for the control of the satellite for this period of time. The satellite will simply be listening for interrupts from other classes. There will be an interrupt that tells this class when to deploy its parachute. This interrupt will be provided by another class, and it will be based on some form of a timer. This event should occur shortly after the satellite has been ejected from the rocket. Another interrupt will tell the satellite to pass control to the class responsible for the next control stage of the expedition. Both of these classes with the interrupts should have a protected modifier so they are only visible to the class which has control of the satellite. This event should occur shortly after the satellite has been ejected from the rocket. This event should occur shortly after the satellite has landed safely on Earth's surface.

The second stage of the satellite's expedition will encompass the majority of the satellite's expedition, and there will be a class written specifically for the segment. This class will be responsible for the period of time that the rocket navigates from its landing zone until it has reached approximately 8-9 meters from the pole. The pole marks the final destination of the satellite, and the satellite must physically come in contact with the pole to finish the competition. The reason for the 8-9 meters is due to the fact that GPS is only accurate at 7.8 meters with a 95% confidence interval [**?**].

During this period there will be many other classes interacting with the class retaining control of the satellite. There should be a class which tells this control class if there is obstacles in the way and what direction to move to avoid them. That class should be a protected class as well, and that class should have a parent/child relationship with each class governing the sensors being used to identify obstacles.

There should also be a class which governs the GPS sensor. This class should help direct the satellite on a macro level, not taking into consideration any obstacles at the local level. This protected class should have its navigation instructions be overridden in the case of an obstacle in its path. This class will also send an interrupt to the control class when the satellite has reached 8-9 meters of the pole which will tell the control class to pass off control to the final governing the satellite's navigation.

The third stage of control will be reached only in the case where the satellite becomes stuck or on its side. The satellite design is being made specifically to preventing this type of situation from occurring, but there is still a chance of it happening. There will be a specific routine devised for this circumstance, and control of the satellite will be passed to this stage in this type of event. This class will have access to the CMOD, ultrasonic sensors via inheriting from these protected classes. When the satellite has recovered from the event, and some routine to avert the obstacle has been completed, then the control will be returned to the previous control stage.

The final stage of control will cover the period of time when the satellite reaches the 8-9 meter range from the pole, and it will last until contact with the pole has been made. The class responsible for the satellite's control over this period will no longer take input from the GPS. The system used previously for the obstacle avoidance system, i.e. the CMOS imaging sensor and the ultrasonic sensor, will be used here for navigation. Even if the ultrasonic sensor is operating at 40kHz, considering the target has a spherical shape, the sensor will be only accurate for up to 20.2 feet so the CMOS imaging sensor will have to suffice until the satellite is close to the pole [**?**]. The CMOS imaging sensor will need to switch modes in this stage to look for a pole shaped object and direct the satellite towards it. A separate protected class should be governing the sensor in this stage of navigation. Instead of looking for obstacles the sensor should be searching for an object which is similar to some stored image, allowing for an appropriate level of error.

III. CMOS Image Sensing

For the ARLISS project we are using a combination of a CMOS imaging sensor and at least one ultrasonic sensor in order to detect and avoid obstacles. The path of the satellite, on a macro level, is determined by the GPS sensor, but modifications to the path will need to be set locally using this system.

The idea behind the imaging sensor is that, when there is an abrupt change in color in an image, especially when referring to nature, it often shows that there is an abrupt change in the depth being viewed. For our purposes we want to use this observation to detect objects in the satellite's path, as well as, changes in terrain like a steep incline or a steep decline.

Manipulating raw camera frames can require a relatively large amount of space, as well as consume a large amount of energy. Energy is the most valuable resource for our project because we have a strict limit to the size and weight of the satellite so we have a very limited space for a battery to fit. There are several methods which we must use to minimize this cost, and there is one critical library in particular which we will use to help conserve this limited battery power.

As far as libraries go, we will be utilizing the cv.h library provided by OpenCV. This open-source library provides the capability of image and video I/O, image processing for individual video frames, as well as, built-in object recognition functionality [**?**]. All foreseeable video manipulating should be able to be done with this single library.

The plan for reducing energy consumption is simple; first we will take a frame of raw footage as input from the CMOS device, then we will convert it to grayscale, and from grayscale we can convert it to binary data. Manipulating this binary data will consume far less energy than performing complex algorithms on the original, raw video footage [**?**].

Using the OpenCv library we will be able to capture video and directly load each video frame from the sensor to an IplImage object. OpenCv also has a built-in function called cvCvtColor() which will convert each image to grayscale. There will then need to be an algorithm which will create a digital map of the image using a two-dimensional array, and which will assign values of 1 or 0 depending on the difference in color change between pixels [5]. When objects or abrupt changes in elevation are detected we can combine data from the ultrasonic sensor to determine the distance of the object, and modify the satellite's route to avoid the obstacle.

This functionality of importing the live video feed, converting it to frames, and applying functions to the data to convert it to binary will need to exist in its own class. This class will be responsible for the flow of the input video data from the sensor, and outputting the data as an array of binary values to a separate class. The output from this class will combined with the output from the class responsible for the ultrasonic sensor data, and this combination of data will be used elsewhere to help determine what sequence of events needs to occur in order to safely navigate the terrain.

IV. Ultrasonic Radar Sensor

For the obstacle detection system, data will be created using a complex algorithm which combines data from the CMOS imaging sensor as well as at least one ultrasonic sensor. As mentioned previously, the path of the satellite will be determined based on the terrain encountered and any obstacles detected in its path. The system will switch from the satellite's GPS sensor to the obstacle avoidance system when an obstacle is sensed, and then will temporarily divert from the direct path in order to drive navigate around it. The GPS will be used to determine the path on a macro level, but it can be modified locally using this system.

The reason for the combination of the CMOS sensor and the ultrasonic sensor is simple. Though a camera sensor, like the CMOS, is the absolute best at interpreting texture, shape, and is the natural choice for scene interpretation based on its accuracy, it has no way to tell the distance of an object [**?**]. Without a distance measuring sensor the satellite would not know the difference between an obstacle directly in front of it, and the horizon which is miles away.

The hardware being used will be a ping-response ultrasonic rangefinder. These sensors have two transducers, a speaker which sends out the burst of ultrasonic sound, and a microphone which listens for the sound as it is reflected off objects in its path. This hardware has two connections to the microcontroller, one for the echo pulse output and the other for the trigger pulse input which must be connected to digital I/O ports [**?**].

There will be one primary library for setting up the class for the ultrasonic radar sensor. The library NewPing.h already has built-in functions to read in input from the sensor. This library will allow us to specify the input pins where the sensor will be connected to the microcontroller, as well as set up a maximum distance of interest using a NewPing object. There is also a timer for the ping which allows us to set the frequency that each ping is sent out in milliseconds. For each ping that is sent out we must have a function which listens for the echo, and converts the time between the ping and echo into a distance [**?**].

There will be many distances recorded for each ping so they must be stored in a two-dimensional array which will be used as a map of the terrain. When a second echo is recorded then the two can be compared for accuracy. If the object in question does not show up in subsequent readings then the algorithm is reset. If the object continues to show up in subsequent readings, than this map of the terrain will be output to another class which will combine the map created by the CMOS sensor, and will use the data to determine if the satellite needs to divert from its current path.

Ultrasonic sensors provide not only accurate distance measurements, but they can also utilize this functionality to provide a method of determining angles of objects, or, more importantly, the slope of the ground. This will prove to be important when calculating the slope of the ground that is in the satellite's path. The ground should always show up in the readings. If, for instance, the ground was not being detected, it would mean that the robot was positioned in front of a steep decline. There will need to be some sort of range of acceptable slopes, and these constraints will need to be determined through testing.

V. Continuous Tracks

For the method of how we would travel along the ground, the team members of the ARLISS Project decided to go with the continuous track, or tank tread, method of traveling. The implementation of these tracks will be carried out by the Mechanical Engineering sub-team of the ARLISS Project.

Continuous tracks are widely popular for their ability to traverse rough terrain much better than traditional wheels, however they sacrifice speed in doing so. Our continuous track system will require at least four separate motors, front left, front right, rear left and rear right. Four motors are required by this design because using only two motors leads to a couple different problems, mainly that either the satellite will not be able to turn or its tracks will be significantly underpowered.

Tracks of various styles and sizes will be tested to see which one best suits our needs. We will do our best to simulate the texture of the Black Rock Desert in Nevada where the competition will be held to get the best and most useful test results possible. Tracks will be compared based on ease of traveling across flat ground, how the tracks handle changes in elevation and if there is any noticeable effect on power consumption between tracks. We will also look at various stress tests for the different tracks, such as how easy are they to break and how easy it is to get the track to fall off the satellite. These tests will help us ensure that the satellite will perform without fail at the competition.These tests will range from being as simple as trying to pry the tracks off of the satellite with our hands to advanced as setting up a course of very rough terrain that will try to either break or remove the track. determining the proper length of tread on the track is very important for the satellite. if the tread is too long it won't travel along the hard, flat desert ground as efficiently as it could be. Make the tread too short, and the satellite might not be able to climb out of tire ruts or other divots in its path.

While the satellite is traveling on the rocket and during its journey back down to the ground, it will need to be packed away in a container the size of a standard soda can. In order to achieve this, the continuous track system will need to be able to compact itself in order to conserve space used inside of the of the soda can container. the way that this will be accomplished is by having small servo motors than can contract the Continuous track motor systems and tread into the confined space. Once the satellite has landed the servo motors will then expand the motors and the treads to their standard position for the expedition across the desert. This will be controlled by the software class that handles getting the satellite out of the soda can container, as described in the Framework section of this document.

VI. Control Board

The control board acts as the physical central hub of our system, connecting together and managing the various I/O devices in our satellite system. The control board will be custom built by the Electrical and Computer Engineering sub-team of the ARLISS Project. Having a custom board built in house benefits us in a couple ways, it can make for the most efficient use of limited space and doesn't waste power or weight by not including I/O devices that we will not be using for our project, which would not be true if we used a pre-made third party board.

The current requirements for this control board include being able to handle I/O for our CMOS imaging sensor, sending it messages for when it needs to capture an image and the be able to move that data back for processing. The control board also needs to be able to support motor control functionality, individually telling our motors how much power to use and whether to turn forward or backward. The control board will also need to be able to interface with our parachute module, telling it when to deploy and hearing back from the module if it deployed properly. Lastly the control board will need to be able to interface with our on board GPS sensor. This is essential because the GPS sensor will handle deploying the parachute module as well as be the chief tool of navigation while on the ground trying to reach the final goal.

Our custom built control board will make use of the Atmel ATxmega128A4U chip. This chip is a low power eight bit AVR microcontroller featuring one hundred twenty eight kilobytes flash memory, with thirty four I/O pins and support for thirty four external interrupts[**?**]. The selling point of this chip is it has low power consumption, has enough I/O pins to suite our needs while also not going overboard. Finally, this chip is part of a family of chips, the mega128 family, that many people in our ARLISS project team are already familiar with as a similar chip from this family was used in the Electrical Engineering course Computer Organization and Assembly Language.

The detailed design and implementation is being handled by the Electrical Engineering team of the ARLISS Project. If designing and building our own board falls through and is no longer an option, the backup plan is to use the Raspberry PI Zero as our control board. The PI Zero will meet our basic needs for this project, but will still have some of the extra I/O devices and ports mentioned earlier that will not be needed for the ARLISS Project.

VII. Obstacle Avoidance

As detailed in the CMOS Image Sensing section, the data that we will be working with to detect and avoid obstacles will be a 2D array of binary values. A value of '1' will represent a change in the gray scale image, which will ultimately outline the terrain and give an easy to traverse dataset with which to avoid obstacles with. A value of '0' will represent no major change in shade of gray between neighboring pixels of the image, meaning that this will either be the interior of an object bounded by '1' values, or empty space in front of the camera.

The biggest concern and challenge facing our satellite project is the conservation of power and limiting power usage. Due to the space constraints of the competition our batteries will be smaller than desired for this kind of journey so we must conserve where we can. One way that we are doing this within the obstacle avoidance system is by not continuously taking in video feed from the camera. The setting where our system will be tested is for the most part very flat, meaning we can realistically take a new image to analyze after a short time interval without having to worry too much about a new obstacle appearing before we can avoid it. The exact time interval is not yet known, as it will take physically testing the satellite system to determine the correct value to use.

The way that the 2D array will be used to avoid obstacles is by examining the values by column from left to right to detect if an obstacle has appeared in our path. We expect there to almost always be a row of solid '1' values along the bottom of the 2D array that represents the horizon, where the ground meets the sky in the cameras eye. The algorithm for avoiding an object is quite simple, if a column or close to a column of '1' values appear towards the right side of the array, we stop forward motion, and rotate counterclockwise while periodically stopping to take a new picture to see if we have angled around the obstacle. once we have a picture showing no obstacles, we travel forward in this direction for approximately five seconds before going back to driving towards the target GPS coordinates. If the obstacle appears on the left side of the 2D array, we do the same thing except rotate clockwise.

Once we have the satellite built and the CMOS imaging implemented with the gray scale to 2D array conversion working, we will be able to test our satellite in various conditions and with different obstacles to see what kinds of obstacles we can drive over and ignore, and what obstacles must be driven around. During this testing period we will also tweak the timing at which how often a new picture will be taken and analyzed. The target for this testing is to find the time cycle at which we can take the fewest pictures while still avoiding all of the obstacles that we need to, and thus saving battery usage by the camera.

VIII. Parachute Deployment

For the parachute deployment protocol, we have decided to use GPS to determine when to deploy the parachute. This method is perfect for us because we already need a GPS unit for other uses, and the GPS unit will be more accurate than just deploying the parachute based on a timer, and take less time in the air than deploying the parachute immediately.

The parachute deployment system will be the first part of our software to run, so we will start the system once the satellite separates from the rocket. This will be accomplished by tripping a mechanical switch which sends a signal to the satellite. Once started, the system will take periodic altitude measurements every second until the altitude drops below 1,000 feet. Once below 1,000 feet for two measurements in a row, the software will send a signal to deploy the parachute. We chose 1,000 feet as the desired height because the average reserve parachute for a skydiver deploys in less than 400 feet [**?**]. Since our satellite only weights as much as a can of soda, 400 feet is a conservative estimate of the distance the parachute will take to deploy. This should ensure we dont waste too much time in the air, but also gives us a large margin of error so we dont accidentally hit the ground before our parachute deploys. The worst case accuracy for civilian GPS coordinates is 25 meters (75 feet) [**?**]. This accuracy is much worse than the navigation specification because of the high speed the satellite is traveling. In addition, altitude error is actually about 1.5 times the amount of horizontal error, which can be a

significant amount. In addition, if the GPS doesnt have an unobstructed view of the sky, the data cannot be trusted at all [**?**]. It is highly unlikely that our view will be obstructed, but it is possible a plane could fly overhead, or the rocket could briefly block our view of the sky. This is the reason we take two measurements before deploying the parachute. All of these factors effect the accuracy of our parachute deployment height, which is why we have been conservative. An extra 600 feet of distance to deploy the parachute is plenty of space, but still minimizes our time spent in the air. Once the software sends the signal to deploy the parachute, the parachute deployment system will end, and a new system will begin to check when the satellite is on the ground.

The system will be implemented within its own class which is responsible for deploying the parachute. An interrupt will be sent to the first stage class once the satellite is ejected from the rocket. Then, the parachute deployment class will launch its own function, which will loop continuously, checking the altitude from the GPS until the value of the altitude is less than 1,000 feet. Once this happens, the loop will execute one more time to check if the altitude is still below 1,000 feet. If it is, end the loop, send a signal to the first stage class to deploy the parachute, and hand control back to the first stage class. If the altitude isnt still below 1,000 feet, start the process over again. The first stage class will be responsible for actually deploying the parachute.

IX. Getting Unstuck From Obstacles

The system for getting unstuck from obstacles has two components. First, determining when the satellite is stuck, and second, getting the satellite unstuck. To determine when the satellite is stuck, the satellites speed will be monitored, and if the speed drops below a certain threshold for a specific amount of time, the system to get the satellite unstuck will be enabled [**?**]. To get the satellite unstuck, the system will attempt to move the satellite in different directions until the satellite frees itself.

The first component which checks if the satellite is stuck will be running constantly in the background while the satellite is navigating itself to its destination. It works by checking the coordinates from the GPS sensor every five seconds. After taking a new reading, the algorithm will determine the average speed the satellite has been moving in those five seconds. If the average speed is below the minimum threshold of 0.2 m/s, the satellite will switch from its normal navigation mode to getting unstuck mode [**?**]. This mode will work by attempting to move the satellite a different direction every five seconds, then checking the GPS coordinates like before to see if the average speed is greater than the minimum threshold of 0.2 m/s. If it is, then the satellite must have moved and the system can switch back to its normal navigation mode. The first direction the rover will try is directly backwards, and each time the rover fails to get unstuck, the algorithm will attempt to drive 45 degrees clockwise of the previous direction driven. If the algorithm travels 360 degrees, then it will drive 45 degrees counterclockwise. This cycle will continue until the satellite gets unstuck, or the satellite runs out of battery.

This system will be implemented in two separate pieces. The first piece, which detects if the satellite is stuck, will be running concurrently with the navigation system. The algorithm will be a continuous loop, which checks the coordinates from the GPS, calculates the average speed from the current coordinate and the previous coordinates, checks if the speed is less than the minimum threshold, and sleeps for five seconds. If the speed is less than the minimum threshold, an interrupt will be sent, transferring control from the navigation system to the getting unstuck system. This piece will be located in the class which contains the second stage of control, which is navigation. This second piece will be located in the class which contains the third stage of control, which deals with the satellite becoming stuck, or on its side. The algorithm will be a continuous loop which increments the direction by 45 degrees, attempts to move for five seconds, and checks if the satellite has moved by calculating its average velocity and comparing it to the minimum threshold. Once the satellite is determined to be unstuck, control will be transferred back to the navigation system.

X. Finding and Touching the Finish Pole

For the protocol to find and touch the finish pole, we have decided to use the existing CMOS imaging sensor to find the pole. Once the pole has been located, the satellite just needs to drive in the direction of the pole until it makes contact with it. The most complicated portion of this is the object detection algorithm. This algorithm will use the OpenCV library, due to its object recognition capabilities [**?**]. The object detection algorithm, however, will work differently than the obstacle avoidance algorithm.

Control of the satellite will be switched to this system once the satellite is within the error range for the GPS coordinates of the finish. Once in this mode, this system must do two things, locate the pole, and move towards the pole. To locate the pole, the satellite will slowly rotate, scanning its surroundings with the CMOS imaging sensor. An image will be taken every second, and each image will be scanned for a specific shape. That shape is the shape of the finish pole, which would resemble a long, skinny rectangle on a 2D image. This shape scanning ability is achievable using the OpenCV library [**?**].

Once the shape has been located, the algorithm will instruct the satellite to move forward. Images will still be taken every second, and the direction the satellite is driving will be adjusted based on how far the shape is from the center of the image. If the shape is on the right of the image, the satellite will direct itself more to the right, and if the shape is on the left, the satellite will direct itself more to the left. Once the satellite is very close to the target the shape may not be detectable, so the satellite will continue to move forward in the same direction. Once the satellite hits the pole, it may get deflected and drive in another direction, but we will have completed the competition, so that isnt important. The system can turn itself off after a certain amount of time has passed.

This system will be implemented as its own class. It is the final stage class, and is started once the satellite is within the error margin of the finish pole. There will be two functions in this class. The first will search for the pole, and the second will move towards the pole. The first function will be implemented as a continuous loop, which rotates the satellite ten degrees, gets an image from the CMOS sensor, and checks that image for the shape of the pole using OpenCV. Once the pole is located, the next function will be called. This function will also be implemented as a loop. Each iteration, it will move the satellite forward, get an image from the CMOS sensor, use OpenCV to find the location of the pole in the image, and determine how far to the left or right to navigate the satellite based on the location of the pole in the image. Eventually, the satellite should hit the finish pole, and we will have completed our mission.

XI. Navigation system design

Navigation system is the one of the most crucial parts of any autonomous vehicles. The team is planning to implementing the rover's navigating system with two major goals: to get the rover drive towards the target pole without getting stuck; Design or use some existing algorithms to let the rover remotely updates the best driving path, in order to achieve the optimized battery life. The team divided the navigation system implementation into hardware and software two pieces, in order to help us research more comprehensively.

On the hardware side, by far we have designed two layers to ensure that the CanSat to meet the functional requirements. The team first collaborated with the electrical engineering team and made the agreement on that we will use the FGPMMOPA6C MTK3339 GPS chipset. The MTK3339 ultimate GPS module has an excellent high-sensitivity receiver and a built in antenna. It is capable to get up to -165 dBm sensitivity, track 22 satellites on 66 channels with a 10 Hz update rate, with its ultra low power usage, this GPS module is unbeatable considering its the under 30 dollars price and ultra compact size(16mm * 16mm *5 mm, 4 grams). A GPS module essentially generates a set of way-points(path) based on the given target pole coordinates, in order to make the CanSat to have more precision control and a better motion awareness, we decided to add a 3 axis accelerometer onto the system. At this time being, we have not yet finalized the selection on which specific accelerometer we are going to use on the CanSat.

Software design is the most important piece of the Navigation system, as mentioned above, the CanSat will navigate itself to the target pole based on the given pole coordinates. In order to enable the autonomous driving feature, we have made a basic design which is based on an existing function called distance and angle calculation method.[**?**] With this method, the system essentially compares its current location with the target pole location, Calculate the shortest path between these two points, then travels through the shortest path. If the system senses any obstacles in the path, it will calculate the new path based on the new current location. This simple process is continued until the CanSat gets close to the target pole(roughly within 8 meters), the target pole mode will be turned on in this case. Lastly, if the CanSat has successfully touched the target pole, it gets stopped, and the whole system will be shut down, see **Fig. 1**. This distance and angle calculation method can be expressed in a set of mathematical equations, please see below:

$$A = Long_{pole} - long_{current}$$
$$B = Lat_{pole} - lat_{current}$$
$$\theta_{pole} = tan^{-}1(B/A)$$
$$Angle(\theta) = Pole_{angle} - Current_{angle}$$

Where, A is distance to be travelled in latitude and B is distance to be travelled in longitude

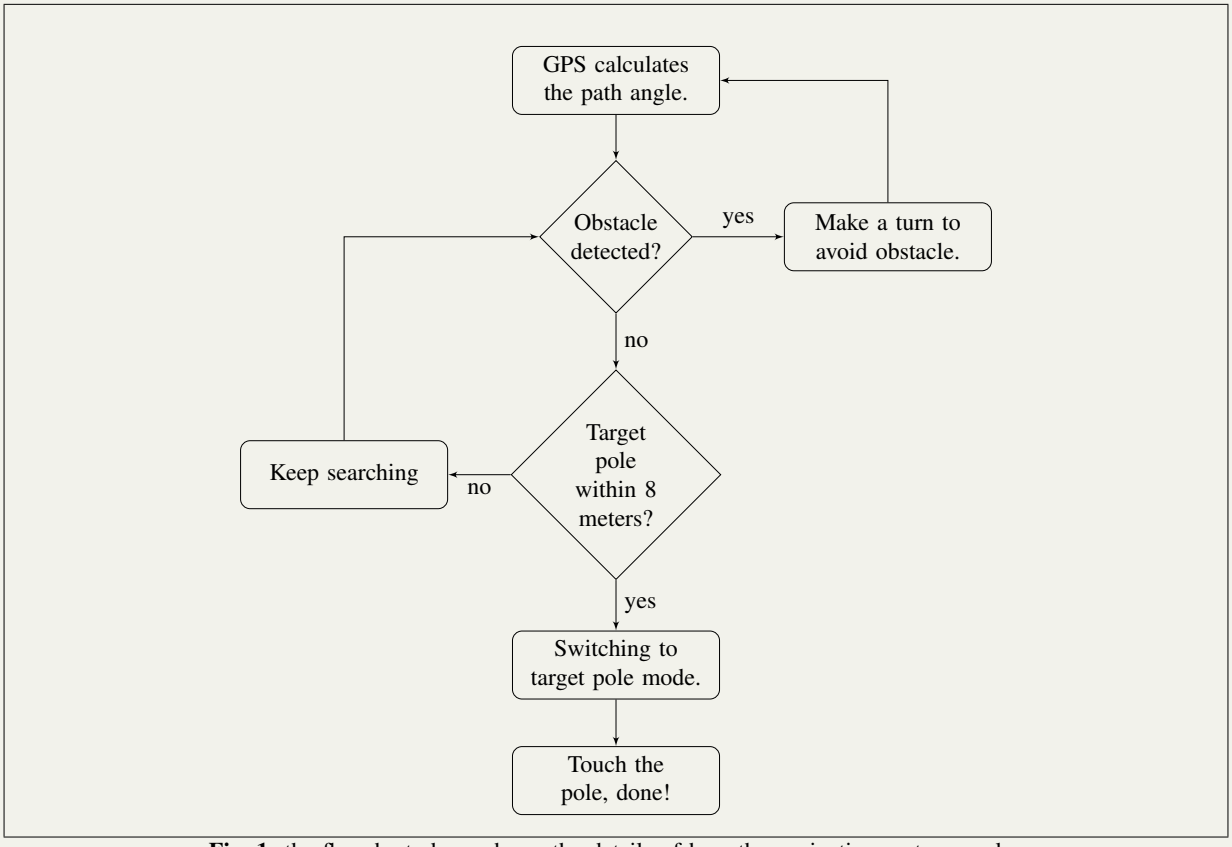


Fig. 1: the flowchart above shows the details of how the navigation system works.

Furthermore, for details about the obstacle avoidance algorithms, as we discussed in the section No.VII, we are essentially using the CMOS image sensor to constantly generating pictures of the front of the path, convert those pictures into a 2D array of binary values, where 1 means object and 0 means open space. [**?**] This 2D array contains a set of binary values can be expressed as this imaginary figure below, the center of this array has a bunch of 1 values which represent an object, more specifically, is an obstacle.

0000000000000000000000000000000000
0000000000000111110000000000000000
0000000000111111111110000000000000
0000000011111111111111000000000000
0000000000111111111100000000000000
0000000000011111111000000000000000
0000000000000111110000000000000000
0000000000000000000000000000000000

XII. Getting rover unstuck when it on its side

The main goal of this task is to get the rover recovered from falling sideways. We have designed a very efficient protocol that ensures the rover is able get unstuck under any kind of scenarios. This protocol can be simply broken down to three layers. The first layer is to determine whether the rover has fell sideways, second layer is called self-rescue program, and the third layer is to place a servo driven arm on the rover body. These three layers has to be executed in this sequence of order in order to make the protocol operates in a cost effective fashion.

To determine whether the rover is operating under normal the conditions or running into problems, we simply utilize its moving speed and heading data from comparing the current waypoint with the last waypoint. If suddenly, the accelerometer

captured an unexpected motion and the rover is constantly sitting on a same waypoint, or moving under a minimum threshold of 0.2m/s, if it is, the system will make the decision on that the rover fell sideways, send an instruction to let the system enable the self-rescue layer.

The self-rescue program layer is a script we are going to implement in the third stage of control. After the system has confirmed that the rover is on its sides, the microprocessor will send an instruction to the motor controller, triggering the motor generating the maximum torque to drive the treads moving opposite direction of its heading, three times. Normally, the motor we are going to use is able to generate torque up to 120 Nm, this type of sudden force is big enough to get the rover recovered consider the rover weighs 300 grams. If the program failed on the rover recovery, then the system will enable the third layer of the protocol.

The third layer is an add-on, though the team believed that the self-rescue program has enough power to recover the rover itself from land sideways. Mechanical engineering team has still made the decision on adding an extra servo on the rover in case for any need. This mechanical arm will be used in the fairing process as well as the third layer of the recovery protocol. If the self-rescue program failed to recover the rover, then the program calls that servo to move mechanical arm push perpendicularly the against the ground to make the recovery.

XIII. Payload fairing

The main purpose of the research done into this subsystem was to help determine the ways the payload can stay safe during its descent and landing. Fairing is essentially the first task the CanSat has to pass when it landed on the ground. The major goal of the fairing protocol is get the rover out then move away from the fairing, and parachute quickly, safely, and without stuck, jam, or having any residues.

Per competition requirement, it is known that the fairing will be built out of a soda can and will probably contain cushioning on the inside to protect the payload. So this limits the options that can be used to separate the fairing. We have developed a solution that is inspired by the SpaceX Falcon Heavy and Falcon 9 payload fairing design, or two half-shells fairing design.[**?**] We broke down this fairing protocol into two stages. First determine whether the CanSat is landed or not, if so then move onto the fairing separation stage so that the rover is able to move away as quick as possible.

To determine whether the CanSat is landed or not, we use the accelerometer to track its ground hitting motion. Speaking the motion, the CanSat create a big deceleration when it goes from descent into a complete stop, any accelerometer in the market is capable to capture this motion. As soon as the accelerometer captured this motion, a signal will be sent to the microprocessor, triggers the fairing separates into two halves. The separation of the fairing is the part that we still stuck on at this moment, we have came up with two feasible options. First, a small explosive or a compressed gas will have to be placed in the fairing in order to cause the separation. Second, as mentioned above in section **XII**, there will be an mechanical arm placed on the rover, this servo controlled arm is primarily used to get the rover unstuck when it falls sideways, we can also write a small script to let this arm to make a series of motions to cause the separation.

Additionally, we have to ensure that the rover moves away from the fairing the opposite direction, compare to its moving direction before landing. This action will ensure that the rover won't get jammed with the parachute and able to move onto the next stage as soon as possible. We simply write a script that let the GPS and accelerometer to determine it's landing direction, and set the rover's initial heading the opposite direction of its descent heading. Many teams from previous years competition have stuck on this stage, specifically their CanSat were jammed by the parachute, made theirs rovers aren't able to go anywhere.

B. *Revision History*

IV. Tech Review

A. *Original Document*

The following document is the original version of our team's Tech Review. The Tech Review was a research-driven document where we divided up the entire project into twelve software components and analyzed them. Each member of the team analyzed three components of the project by comparing and contrasting three alternate methods for implementing each of the compenents. At the end of each analysis, and based on the research, one of the three methods for implementing the component was chosen.

The ARLISS Project Technology Review CS 461

Steven Silvers, Paul Minner, Zhaolong Wu, Zachary DeVita
Capstone Group 27, Fall 2016

Abstract

This document reviews and evaluates potential pieces of technology that could be used to complete our project. Arguments are made for and against each piece of technology before finally settling on which technologies will be used when implementing the project.

CONTENTS

I. Introduction

In this document there are twelve components of our team's software which have been researched by the corresponding member from the list below. Each component compares and contrasts three alternate methods of accomplishing the same goal, and, at the end of each technology review, the method which best fits our teams needs has been chosen.

A Comparison of Languages
Zachary DeVita
Methods of Object Recognition
Zachary DeVita
Switching Modes to Locate the Pole
Zachary DeVita
Control Board
Steven Silvers
Avoiding Obstacles
Steven Silvers
Mode of Transportation
Steven Silvers
Parachute Deployment
Paul Minner
Getting Unstuck from Obstacles
Paul Minner
Find and Touch the Finish Pole
Paul Minner
Navigating algorithms selection
Zhaolong Wu
How to get unstuck if the rover fell or landed sideways
Zhaolong Wu
Payload fairing
Zhaolong Wu

II. A Comparison of Languages

For this tech review I will be comparing and analyzing the differences, and benefits of Assembly Language, C and C++ in regard to microcontrollers. These are the three most prominent languages used for programming single integrated circuit controllers, and each one has its own unique pros and cons.

A. Options

1) Assembly Language: Assembly language has some huge benefits that are virtually incomparable to C, or C++ for that matter. There are several functionalities which are possible in assembly which cannot be accomplished by higher level languages as well. In terms of speed and memory cost, assembly language blows both C and C++ out of the water; there is no comparison or debate on its superiority. The programmer retains ultimate control over each bit of memory when using assembly, but it is debatable whether this is a benefit or an encumbrance. With this control comes more potential for bugs, issues with readability, portability, testing, and a far more laborious implementation of the code. Assembly utilizes preprocessing directives and is used at the earliest stages of the bootloader which is not possible in higher level languages.
2) C: Though not nearly as compressed as assembly language, C utilizes a syntax which is much more compressed than most of higher level languages, which allows it to provide similar functionality at a lower cost of memory. In comparison to assembly language, C has a much larger library, and, thus, a larger instruction set and op-code associated with its commands. This can drastically slow performance in smaller processors which only have an 8 or 16-bit BUS because additional cycles are necessary to carry out each of the operations. This would only be a legitimate issue with small microcontrollers though. The benefits of C over assembly are considerably greater in this day n age where the amount of memory and speed are not nearly as concerning as in previous decades. C is far, far, far easier to program in. This allows C programs to be written faster, and with fewer errors. The programs are easier to read, debug, manage and maintain as well. An interesting fact, you can also write assembly language in a C program, but you cannot write a C program in assembly [1]. Im not sure why you would ever do such a thing but it is possible.

3) C++: It is a common assumption that C++ is unsuitable for use in small embedded systems. This is partially due to the fact that 8-bit and 16-bit processors lack a C++ compiler, but now there are 32-bit microcontrollers available which are supported by C++ compilers. The majority of C++ features have no impact on code size or on CPU speed when being compared to C [2]. C++ has all the same benefits over assembly language that C has, which are very important considerations when choosing the language. The most focal comparison here would be between C and C++. One of the original intentions of C++ was to prevent more mistakes at compile time, and, in comparison to C, it has achieved its goal. It is cleaner than C, easier to debug, easier to read and the code is more robust. C++ has added functionality, i.e. function name overloading, and classes which can be declared using different access modifiers.

B. Goals for Use in Design

The programming language is hugely important for the success of the project. Different languages offer different sets of functionalities, some which might be necessary to meet our goals and some which might just add unnecessary overhead. This decision might be the most important decision made at this stage in the project.

C. Criteria Being Evaluated

The most important aspects to consider when selecting a programming language for our specific project are going to be energy consumption, and ease of use. We are very tight on energy due to our satellite having size and weight constraints, and due to the fact that we have no idea how far our satellite might need to travel to reach its destination. Obviously ease of use would be important because this project is very time sensitive. Additional criteria being evaluated are the speed at which the code is executed, the memory cost of executing the same block of code, and the amount of extra functionality being provided by one language over another. This functionality may be gained from libraries or by tools that are made available by using that language.

Another important consideration is that you only pay for what you use. This is in regards to the cost of memory which seems to always be a concern when considering C++. What this means is that, because C++ is almost exactly a superset of C, there is no principally caused in a program that only uses C features [3]. A block of code written in C can be run through a C++ compiler, and the compiler will treat it the same as if it was ran through a C compiler; it will produce the same machine code.

The big negative with C++ is with the additional overhead which is common in higher level languages. In our specific circumstance, this isnt really a memory or bus size issue as much as a power issue. Because our satellite is so small, and will only weigh a maximum of 150 ounces, the overhead issue starts to become a legitimate issue for whether or not we can power this satellite long enough to reach its destination. The team that won last year had their satellite travel approximately 7 miles to its destination after being blown way off course. Having a program which uses twice as many micro-operations or cycles to complete the same task will absolutely require more energy to operate.

Visual Comparison of Languages

	Energy Consumption	**Ease of Use**	**Functionality**	**Memory Cost**	**Speed**
Assembly	1st	3rd	3rd	1st	1st
C	2nd	2nd	2nd	2nd	2nd
C++	3rd	1st	1st	3rd	3rd

D. Selection

Based on the research, Im a little torn in my decision. I thought coming into this assignment that C was the obvious choice because of lower cost of memory and higher speeds than other higher level languages, and its monstrous benefit over assembly with ease of use, debugging and maintainability. Assembly is out of the question primarily due to there being a considerable chance we would never finish the project. Assembly is just unrealistic for our purposes, and both memory and speed shouldnt be too much of an issue in our project since we will be building this satellite in 2016.

There are some important benefits for using C++, particularly the use of classes and access modifiers, and the fact that it is easier to debug and better at catching bugs at compile-time. Considering the fact that almost all C code can be cut and pasted into a C++ compiler and return the same machine code, our team will use C and utilize the benefits of slightly higher speeds and a lower memory cost, but we will reserve the option to port over to C++ in the case that we decide to implement classes with access modifiers.

III. Methods of Object Recognition

There are many sensors used for object recognition, each with its own differences and benefits. For our project we are looking for an object recognition system which will be accurate, has a relatively high resolution at a short range, is light on energy consumption, and which will operate properly in potentially 100+°F temperatures. These are the primary concerns for choosing this hardware, and this still leaves many options to choose from. The two types of sensors used for this type of functionality are proximity sensors and image sensors. Proximity sensors include the photoelectric, light sensors, as well as the ultrasonic, a.k.a. radar, sensors. These two sensors, as well as, imaging sensors will be analyzed and compared in this technical review.

A. Options

1) Photoelectric Sensors: As far as photoelectric sensors go, I did some research on LIDAR and Sharp IR sensors as they are the most applicable for our teams project. I found a lot of interesting articles about these sensors; they shine a small light at a surface and measure the time it takes for the light to be reflected back. These types of sensors can determine the distance of objects in near-instantaneous time because light moves so fast. They have proven to be very accurate in the right settings as well. IR sensors are very cheap as well and use little energy to be powered. LIDAR sensors, on the other hand, are much less energy efficient and seem to cost a considerable amount of money for a reliable one [4]. The worst part about photoelectric sensors is that sunlight causes a tremendous amount of noise. These sensors will be completely unreliable and ineffective for our purposes, as our satellite will be navigating through the Nevada desert. Photoelectric sensors are absolutely not an option for our teams object recognition system.

2) Ultrasonic Sensors: Ultrasonic sensors, on the other hand, are not compromised by direct sunlight. These sensors are cheap to purchase, light weight, use little energy to be powered, and are accurate in most cases. They rely on the speed of sound opposed to light to determine distances of objects so the response is not nearly as immediate as with photoelectric sensors, but the difference is negligible in the scope of our project. These sensors have a range of more than 6 meters which is more than the range necessary for our requirements. These sensors provide accurate detection of even small objects, and are reliable in critical ambient conditions where there is lots of dirt and dust [5]. They are not easily weatherproofed for a humid climate, but that should not be a problem where the competition is being held.

3) Image Sensing: Image sensing is a third option for object detection. Image sensing is primarily done using a CMOS (complementary metal oxide semiconductor) or a CCD (charge-coupled device) sensor which are commonly used in digital cameras. Both of these systems convert light into electrons, but the methods each of them uses have very distinct differences. Of these two sensors I have chosen CMOS for the comparison based on the fact that they are less sensitive to light, extremely inexpensive, and more importantly, consume far less power than their counterpart [6].

CMOS sensors create a moderate amount of noise due to the process of converting light into electrons. This has a potential to affect the accuracy of a reading. Because these sensors have no way to determine distance, they often require the additional support of photoelectric or ultrasonic sensors. Other relevant perks to using image sensors is that they are able to prioritize obstacles or dangers, and they can detect specific shapes. This is important for the final leg of the satellites expedition, where the satellite must recognize the shape of the pole which it must make contact with to finish the competition. Using an image sensor will definitely require far more complicated programming to detect objects but it seems like a valid consideration for the project [7].

B. Goals for Use in Design

The goal for the sensor, if not clear, is that we need some kind of sensor to send data to the micro-controller so that the data may be analyzed and obstacles can be detected. If the obstacles can be detected using some system of sensors, than the obstacles can also be avoided. The goal is to avoid all obstacles which are determined to be a threat to our satellite reaching its destination.

C. Criteria Being Evaluated

The most important criteria that these sensors are being evaluated on would be the accuracy of the sensor and the energy consumption by the sensor. Again, energy consumption is important because we have tight constraints on space and weight for the satellite. This means that we are limited to a relatively small battery to provide power to the satellite for the entire duration of its expedition. Accuracy is important because we don't want the satellite to miss an obstacle or waste energy thinking there are obstacles when there are none. We need a sensor which creates minimal noise while operating. Also important criteria can be the range of a sensor, the cost of a sensor, and any environmental conditions which may cause adverse affects to the precision of the sensor.

A Comparison of Object Recognition Sensors

	Accuracy	Range	Energy Consumption	Cost	Environmental Conditions
Photoelectric	High	0.8m	Low	Low	Won't work in direct sunlight
Ultrasonic	High	6m	Low	Low	Problems at high pressure or with humidity
CMOS	High	N/A	Low	Low	Problems with humidity

D. Selection

After an analysis and comparison of the various sensors I have chosen to use the combination of an ultrasonic radar sensor and a CMOS imaging sensor. The CMOS imaging sensor will be used to identify obstacles in the satellites path while the ultrasonic radar sensor will be used to find the distances of these objects. Changes in light can help to identify obstacles or abrupt changes in pitch of the terrain, and then the radar can be used to determine the precise distance of the obstacle. I believe this combination will be the most effective, and provide the highest accuracy of the possible sensors. These sensors are also cheap, effective for the climate, and they both have a low energy consumption.

IV. Switching Modes to Locate the Pole

At the satellites final destination there will be a metal pole which the satellite must bump in order to successfully finish the objective. The problem created by this is that the GPS, global positioning satellite, is only accurate to within 7.8 meters with a 95% confidence interval, according to the U.S. government [8]. This means that between 8-9 meters of the pole that there must be a switch to a different interface to guide the satellite. The pole is metal so we have the option of using capacitive or inductive sensors. The other option, which seems much more difficult but may be necessary due to the range, would be utilizing the CMOS image sensing technology that we are already using for the object recognition.

A. Options

1) Capacitive Sensor: A capacitive sensor is a proximity sensor that can detect nearby object by their effect on the electrical field that is sent out by the sensor. Capacitive sensors are cheap, light and relatively durable. Like any electrical component, they have problems in humidity, but they would be much easier to weatherproof than, for example, an ultrasonic sensor. Either way, humidity should not be an issue in the tail end of summer, and in the Nevada desert. This type of sensor will detect anything with a positive or negative charge which could be a potential problem, but probably not at the location of the competition [9]. The bad news about this approach is that capacitive sensors only have a range of approximately 1cm or less. This is not an option for our project.

2) Inductive Sensor: Inductive sensors are proximity sensors as well, but they can only detect metal, whereas the capacitive sensor can detect anything which affects the electric field. Again, these are cheap to buy, light, durable, easy to weatherproof for our purposes, etc. Unless there will be other metal objects in the satellites path for the last 9 meters, I would feel safe to say that this would be accurate as well. But again the range makes these sensors, similarly, not a viable solution for the problem. These sensors have a range of up to 2 inches [10].

3) Image Sensing: After analyzing GPS, capacitive and inductive sensors, all there is left is using the CMOS image sensing technology that will already be in place. The advantage of using CMOS is that the system will already be in place for the obstacle recognition component of the project. This means that there is no additional weight or space used. This also means that there will not be any additional energy consumption, cost or weatherproofing associated with the selection. There will be much more programming needed to implement this design though.

B. Goals for Use in Design

The goal for this component of the software is to determine what method is being when switching modes in the final stage of the satellite's expedition. The sensor being used to locate the pole which the satellite must make contact with must be established. It is based off this that we are able to design the software for making the transition.

C. Criteria being evaluated

The most important criterias for this component are obviously the accuracy of the sensor, and also the range of the sensor. The range is important because GPS is only viable until approximately eight meters from the target so the system must begin working at around eight meters. Other factors being considered are cost, energy consumption and environmental effects on the sensor.

A Comparison of Methods to Locate the Pole

	Accuracy	Range	Energy Consumption	Cost	Environmental Conditions
Capacitive	Moderate	1.0cm	Low	Low	False positives near electrically charged objects.
Inductive	High	2 in	Low	Low	Problemswith humidity
CMOS	High	N/A	*None	*None	Problems with humidity

D. Selection

After analyzing GPS, capacitive and inductive sensors, all there is left is using the CMOS image sensing technology that will already be in place. The way this will have to work is that when the satellite is within 8-9 meters of the pole, the program will need to alternate to a different interface. In this interface, the satellite will be guided by the object recognition system, and it will quit reading data from the GPS. The object recognition software will be designed to pick out abrupt changes in light on the edges of objects which signifies a change in depth. When these objects are interpreted, they will be compared to an object that is a digital representation of the pole. When the satellite finds an object which has the same shape, obviously using some level of error in the computation, it will self-drive itself to the object. This seems like a difficult chunk of code to produce, but it is possible, and it has been done before. Considering there will likely not be any objects, i.e. rocks, which have a similar shape to the pole, I think this is an excellent way to solve this problem.

V. Control Board

A. Options

The options being considered for the control board in our project are the Arduino Uno, Rasperry Pi Model A+ and a custom board developed by a team of ECE capstone students.

B. Goals for Use

The control board is what brings all the hardware together to create a single, functioning system. Our software system will be loaded directly onto this board to control input and output to the various system peripherals, such as the motors and GPS module.

C. Evaluation Criteria

These board options will be evaluated based on their ability to run the embedded system that we develop, ability to interface with various required hardware sensors of the system, if they meet the space constraints of the CanSat design, and how well they manage limited resources such as power.

D. Discussion

a) Arduino Uno: The Arduino Uno is the base level Arduino board and is based upon the ATmega328P board with fourteen digital I/O pins, six of which support pulse with modulation[11]. Arduino supports its own language based on Java, C and C++ as well as its own IDE making it easy to work with. The Uno is 68.6mm long by 53.4mm wide, which means it will fit in a soda can with dimensions 66.3mm by 115.2mm however, we do not know how much of that space will be taken up by other hardware. Battery wise the Uno only requires a maintained five volts from a battery to operate. The Uno is a traditional microcontroller, once it is given power it immediately begins running code which may be ideal for our project.
b) Raspberry Pi Model A+: The model A+ is what is recommended by Raspberry Pi to use for embedded projects due to its lower power consumption. The Raspberry Pi measures 85.60mm by 56mm by 21mm which may make it a very tight fit based on the CanSat space constraints[12]. Another point to mention is that Raspberry Pis aren't traditional microcontrollers, but are instead full fledged computers. The Pi also has a few extra I/O devices prepackaged that aren't necessary for our project, such as an HDMI port.
c) Custom Board: A custom built board would be the best option if done correctly. The dimensions could be developed with the Can Sat restrictions in mind and could only include the necessary hardware. On the downside it would be a untested board, with no way of knowing how much power it would consume or if it would be able to run our system until after it is designed, built and tested.

E. Selection

With these pros and cons in mind, I would select the Arduino Uno as the controller for our project. It is a popular, well documented board that we know for sure works. My second choice would be the custom board, only because with the given time line of the project if it didn't work we would be in a very bad place. The Pi comes in third due mostly to size, not being an actual microcontroller and extra devices that are not needed.

VI. Avoiding Obstacles

A. Options

Options for obstacle avoidance include detecting obstacles and then driving out of the way to get around it, develop an algorithm to determine if it can be driven over or not, or drive straight through the obstacle.

B. Goals for Use

The goal of this system is to make sure our CanSat can reach the finish destination without getting caught or stuck on anything in the CanSat's way.

C. Evaluation Criteria

The technology will be evaluated on how easy it will be to develop, and how effective it will be at accomplishing the goal of reaching the destination.

D. Discussion

a) Drive Around: This method of avoiding obstacles while seeming like the smart choice has some problems. Driving around an obstacle would be smart if the obstacle were say a large rock, but if it were something like a long ditch created by a car tire it could go on for miles, and our CanSat would never be able to reach the destination. It could be fairly easy to develop, as soon as it detected an obstacle it could run a routine that makes the CanSat backup, turn and move forward a bit before trying to proceed to the goal.
b) No Avoidance: This method while sounding awful could actually work. Our CanSat will be tested in the Black Rock Desert of Nevada, a very flat, open space. It would be very easy to develop this as it would mean no programming, which would also save memory on the board. However, if any obstacles at all popped it could be game over for our CanSat.
c) Algorithm Avoidance: This would be the most difficult of the options to implement and would involve heavy testing as well as a great knowledge of automated driving systems. On the up side, knowing when the CanSat can and cannot get over or past an obstacle would be extremely helpful, cutting down drive time and therefore saving battery.

E. Selection

With the evaluation criteria in mind, developing an algorithm to determine whether or not the CanSat should try to avoid an obstacle is the best option for accomplishing our goal. While the other two options would be much easier, they also carry much higher risk of the CanSat failing to reach the finish point, making the algorithm based avoidance the responsible choice.

VII. Mode of Transportation

A. Options

The options considered for mode of transportation are a traditional two wheel setup with a wheel at the top and the bottom of the CanSat and the CanSat being oriented so that the can body would by horizontal. The next option would be using caterpillar tracks with the can oriented vertically. The final option would be a four wheel setup with the can oriented horizontally.

B. Goals for Use

The goal for this part of the system is the actual mode of transportation for getting the CanSat from where it landed to its final destination.

C. Evaluation Criteria

This technology will be evaluated on its ability to efficiently transport the CanSat, ability to handle rough terrain, and ability to conserve space within the delivery CanSat.

D. Discussion

a) Two Wheels: The two wheel setup would most likely use the least amount of power out of the three options, as there would be two total motors. Since they would be oriented horizontally, you could get the biggest wheels possible with the size constraints of the can. Bigger wheels would be better for getting over certain obstacles as opposed to small wheels. With the horizontal orientation, there would not be very much ground clearance between the CanSat body and the ground, which could raise a problem with very small obstacles. Because it is oriented on only two wheels, getting flipped on becomes a not very big concerned because the wheels would always be touching the ground.

b) Four wheels: The four wheel option would provide a stable base for driving the CanSat, but also opens up the CanSat for possibly getting flipped by an obstacle and stuck. This option could be done with two or four motors, but with size limitations already being pushed with four wheels, it would most likely be limited to two motors.
c) Caterpillar Tracks: This option is interesting, as the tracks would definitely be the best at traversing rough terrain and getting over obstacles. On the downside they would take up a lot more room in the CanSat than traditional wheels and would also require a bit more power.

E. Selection

Based on the above criteria and the pros and cons of each, I would use the caterpillar tracks. What they give up in size consumption and power consumption, they more than make up for in ability to get over rough terrain. It does not matter how much space or power is saved for other devices if our CanSat gets stuck on an obstacle while driving to the goal.

VIII. Parachute Deployment

A. Options

The first option being considered for deploying the parachute is deploying at a specific altitude based on an altitude sensor. This option would require software to be written to read the values sent from an altitude sensor. The second option is deploying the parachute after a certain amount of time. This would require a timer to implement. The last, and simplest solution is to deploy the parachute immediately after the satellite separates from the rocket. This option could be entirely mechanical, and therefore not require any software to implement.

B. Goals for Use

In order for the parachute deployment to be considered a success, it must activate before hitting the ground. Ideally, the satellite would reach the ground as quickly as safely possible in order to maximize the battery life of the satellite.

C. Evaluation Criteria

These options will be evaluated based on simplicity, time taken to hit the ground, accuracy, power draw. Simplicity is important to see if a slightly better solution is actually worth the effort to implement. The time taken to hit the ground has an effect on how much power is used before the satellite starts moving towards its target and is therefore important. Accuracy is important since we dont get a second chance if the satellite is destroyed. Finally, power draw is important since power is a limited resource.

Comparison of Evaluation Criteria for Parachute Deployment

	Simplicity	**Time**	**Accuracy**	**Power**
Altitude Sensor	Least	Fastest	High	High
Timer	Less	Fast	Lower	High
Immediate Deployment	Most	Slow	High	Low

D. Discussion

The altitude sensor solution will most likely be the most accurate, and take the least amount of time to hit the ground, but may draw more power than other solutions, and is the least simplistic. This is because software has to be written to check the value of a sensor incrementally, which is the most complicated solution, and constantly checking the value of a sensor will drain power faster than if there were no software. The timer based solution will be less accurate, and draw more power than the final solution, but will take less time to hit the ground, and is simpler than the first solution. There is still the same problem about power, which is that a timer has to be checked constantly and therefore drains power. The final solution to deploy the parachute immediately is the simplest solution, since it requires more software, but it will take much longer for the satellite to reach the ground than the other two solutions. The accuracy of this solution should be excellent as well.

E. Selection

We have selected the timer based solution. This is because we need the satellite to reach the ground relatively quickly to save power. We believe the power draw from the software to check the timer would be less than the power lost from being stuck in the air for much longer if we had picked the immediate deployment method. The problem with the altitude sensor option is that we currently arent planning to add an altitude sensor, and space is a premium, since the rover must fit in a soda can.

IX. Getting Unstuck from Obstacles

A. Options

Our three options were considering involve either adding another hardware component, using existing servos, or just using the treads. The first option is to add a mechanical arm which deploys to get the satellite unstuck from whatever it hit. The second option is to attempt moving different directions with the treads while moving the treads up and down with the servos they are attached to. The third option is to simply use the treads on their own and attempt moving different directions until the satellite can move.

B. Goals for Use

The goal of this system is to recover the satellite if the obstacle avoidance system fails and the satellite hits an obstacle. This is a difficult goal since we dont expect the obstacle avoidance system to fail, and if it does, the rover has encountered something we didnt anticipate. Hopefully, this system is never actually used.

C. Evaluation Criteria

These options will be evaluated on simplicity, likelihood of success, and space constraints. Simplicity is important since more effort may not be worth a slight improvement over another option. Likelihood of success is difficult to guess, but obviously it is very important since the solution needs to work. Space constraints are important since space is limited on the satellite.

Comparision of Evaluation Criteria for Getting Unstuck From Obstacles

	Simplicity	**Success**	**Space**
Mechanical Arm	Least	Most Likely	Uses Space
Moving Tread Servos	Less	Likely	Doesn't Use Space
Moving Different Directions	Most	Least Likely	Doesn't Use Space

D. Discussion

The mechanical arm option is the most likely to succeed since it could implement everything the other options implement as well, but it is by far the most complicated and uses the most space. Determining the best placement of the arm and how to use it also isnt obvious. The arm would be most useful when the treads are completely stuck, because the other options wouldnt be able to deal with this. The option which raises and lowers the treads would be the next most complicated, but it may be more likely to succeed than the last option. Without testing, however, its difficult to determine how helpful changing the height of the treads would actually be in getting unstuck. This option doesnt require any extra space, either. The last option to simply move the treads in different directions to get unstuck is the simplest solution, and requires no space. This option is also the least likely to work, since all other options implement this as well as something else.

E. Selection

We will use the simplest solution, which just attempts to move in different directions until the satellite can move. We picked this solution because it is the simplest to implement, and we werent sure how well the other solutions would actually work. The mechanical arm option would take up too much space, so its not an option, but the option which raises and lowers the treads would have worked. We just dont know if changing the height of the treads would actually help get the satellite unstuck, so for now we arent including it. If, in testing, we find that our simple solution doesnt work well, we may implement the ability to raise and lower the treads as well.

X. Find and Touch the Finish Pole

A. Options

Our options are based on either brute force, or sensors. The first option is to drive in a square pattern, like a lawnmower, in the general area of the pole until we hit it. The second option is to drive in a circular pattern outward until we hit the pole. The final option is to use sensors to detect where the pole is, and drive straight into it.

B. Goals for Use

Our goal is to hit the pole as quickly as possible. We must hit the pole to finish the competition, so this system needs to work correctly, otherwise we lose the competition. Speed is important to preserve battery life. The satellite will likely be low on power at this point, so finishing before it runs out of power is important.

C. Evaluation Criteria

We will evaluate these options based on speed, simplicity and accuracy. Speed is important to make sure we dont run out of power. Simplicity is important to save us time. Finally, accuracy is important because if we fail to hit the pole, we lose the competition.

Comparison of Evaluation Criteria for Finding and Touching the Finish Pole

	Speed	**Simplicity**	**Accuracy**
Square Pattern	Slow	Simple	More Accurate
Circular Pattern	Slow	Simple	More Accurate
Detect using Sensors	Fast	Complicated	Less Accurate

D. Discussion

A separate mode of finding the finish pole is needed because GPS isnt accurate enough. GPS is accurate to within a few meters, but the pole will likely only be a couple inches wide. The first option solves this problem by getting to where the GPS says the pole is an searching in a square pattern, like you would run a lawn mower. This is very simple to implement, and should be accurate. Unfortunately, it may take a long time to find the pole. The second option is just like the first, except the satellite searches in a circular pattern, instead of a square pattern. This has the same pros and cons as the first option. The last option is to search for the pole using the sensors used for the obstacle avoidance system, then driving straight into the pole. This option is the most complicated, and may be less accurate than the other solutions, but could be much faster.

E. Selection

We chose to go with the circular search pattern. We chose this solution because a circular pattern may take slightly less time than a square pattern, since the satellite is checking the area closest to the starting destination, rather than checking far away corners using a square pattern. The option to use sensors doesnt seem accurate enough to us. We may change to the sensor approach later on if when we test the sensors, they seem very accurate.

XI. Navigating algorithms selection

A. Options

- Shortest path method: rover drives to the target pole directly based on the navigation module generated shortest path, without using any obstacle avoidance.
- Decision making method: rover drives to the target pole with obstacle detections. We set a boolean numbers in the program where 1 represents obstacle and 0 represents clear path. When there's an obstacle on the rover's path the obstacle avoidance module will be enabled. [**?**]
- Shortest path + Decision: this method is the combination of these two methods above. The navigation module keeps updating the new shortest paths after the rover went around the obstacle.[**?**]

B. Goals for Use

Navigation system is the one of the most crucial parts of any autonomous vehicles. The main goal for this part of the project is to get the rover drive towards it's target without any stuck,

C. Evaluation Criteria

We evaluated these options based on easiness of implementation, resource needed, Path length, and chances of getting stuck.

Comparison of Evaluation Criteria Navigating Algorithm Selection

	Simplicity	**Resources needed**	**Path length**	**Chances of getting stuck**
Shortest path	Most	Low	Short	High
Decision making	Less	Medium	High	very low
Shortest path + decision making	least	High	Medium	low

D. Discussion

The shortest path method is the easiest implementation for this task, but it has very high chances to hit an obstacle, may result being stuck. The decision making method is less simple, it takes reasonable amount of resources, and having a very low rate to get stuck, the only problem for this method is it might result the rover to travel a very long path, which takes a lot of battery. The combined method is the hardest one to implement but will give us the best outcomes.

E. Selection

The teams selection is we implement and test the decision making method first, if by the end we have extra time we will add the shortest path method on to it. Ultimately the combined method would be a cool feature and hopefully will make the impression.

XII. How to get unstuck if the rover landed sideways

A. Options

- Two wheel rover solution: Two wheel design is a great solution in order to prevent the rover falls sideways, because of its unique shape and weight distribution, the rover has a very low center gravity, that can guarantee itself won't fall sideways.
- Programming approach: Design an algorithm that makes the rover to move the tread opposite of its moving direction when it encounters this situation.
- Mechanical arm solution: By adding a servo driven arm onto the rover, it will get the rover unstuck when the rover landed sideways.

B. Goals for Use

The main goal of this task is to get the rover recovers itself in normal position from landed or fell sideways.

C. Evaluation Criteria

We evaluated these options based on feasibility, chances of success, and space requirement. Feasibility is the first thing to be considered, it would be a waste of time if we found the method we have chosen is less possible to archive after the implementation has started. Speaking on the chances of success, we want to ensure that the method we choose turns out has the best success rate. Per competition rules & requirements, payload's space is extremely limited, space requirement is also an important criterion.

Comparison of Evaluation Criteria for Getting Unstuck if the Rover Landed Sideways

	Feasibility	**Chances of success**	**Space taken**
Two wheel rover	low	High	Optimized
Program the Treads move clockwise	High	Medium	No space needed
Mechanical arm	Low	Medium	Takes space

D. Discussion

Two wheel rover is a great method, it has unbeatable success rate and it utilizes the most of space in a soda can, but definitely beyond our knowledge to implement at this time being. Programming solution is the most feasible method for us computer science major student, the success rate is relatively high and it doesn't require any additional component so saves up some spaces. Mechanical arm method is also great but beyond our knowledge to implement, its success rate is only marginally higher than the programming method.

E. Selection

We chose the programming approach. Recalling our goal is to get the rover recovers itself in normal position from landed or fell sideways, this is the most feasible method under our current knowledge base.

XIII. Payload fairing

A. Options

- Two half-shells solution[**?**] - This method is inspired by the SpaceX Falcon Heavy and Falcon 9 payload fairing design. We cut the soda can into two halves and put the rover into it, the can breaks into halves when it landed on the ground, in order to let the rover moves away freely.
- Wrapping solution - We cut the top and bottom of the soda can off, then cut the rest of it in a flat sheet, then wrap the rover in it and tape them together. Ultimately, this fairing method works the best with the two wheel rover design.
- Catapult solution - In this method, we first cut the top of the soda can off, put a spring in the can then press the rover into it, when the can hit the ground it triggers to release the spring that throws the rover out of the soda can.

B. Goals for Use

The major goal for this part of the system is get the rover out and move away from the fairing quickly, safely and without stuck or having any residues. This is one of the must done tasks in this project.

C. Evaluation Criteria

We evaluated these options based on feasibility, rover's safety, and probability to get stuck or having residues. Feasibility is the first thing to be considered since we want to get everything working in cost-effective ways. Rover's safety is a crucial point, if the rover is broken or damaged during the fairing stage then the whole competition is over. Probability of getting stuck or having residues is also important since the rover might also get damaged if something is jammed into its wheels or body.

Comparison of Evaluation Criteria for Payload Fairing

	Feasibility	**Rover's Safety**	**Probability of getting stuck or having residues**
Two half-shells	High	High	Medium
Wrapping	High	Medium	High
Catapult	Low	Low	Low

D. Discussion

Two half-shells option is a feasible, and relatively safe method, the only problem is it's possible to get the rover stuck in the can. Wrapping option is also a relatively feasible method, but because we break the can into too many pieces that we can't guarantee the rover's safety during its launching and landing stages, those pieces may also end up stuck in the rover. The catapult method has the least feasibility, it puts a lot of stress on the rover, also throw the rover out of the can might damage it. The only advantage of this method is it guarantees the rover can get out of the soda can, and without having any residues.

E. Selection

Upon the discussions, we made the final decision that we will implement the payload fairing by using the two half-shells method. It was an easy decision for us since we made it clear in the discussion that this method is the most feasible and will give the team the best outcomes.

XIV. Conclusion

This document detailed our plans for different pieces of technology which could be used to complete our project. Finding multiple solutions for each technology allowed us to think more deeply about which solution would really be best for each technology. We believe the choices we have picked in this document will give us the best outcome for our project, although some of these decisions may change over the coming weeks, as we better understand what we are building.

XV. References

References

[1] jain.pk, "Using inline assembly in c/c," Oct 2006. [Online]. Available: http://www.codeproject.com/articles/15971/using-inline-assembly-in-c-c

[2] D. Herity, "Modern c in embedded systems part 1: Myth and reality," Feb 2015. [Online]. Available: http://www.embedded.com/design/programming-languages-and-tools/4438660/modern-c--in-embedded-systems---part-1--myth-and-reality

[3] "C vs c++." [Online]. Available: http://www.mikrocontroller.net/articles/C_vs_C%2B%2B

[4] "How does lidar work? the science behind the technology," 2016. [Online]. Available: http://www.lidar-uk.com/how-lidar-works/

[5] B. GmbH, "Ultrasonic sensors," 2016. [Online]. Available: http://www.balluff.com/balluff/mde/en/products/overview-ultrasonic-sensors.jsp

[6] "What is the difference between ccd and cmos image sensors in a digital camera?" Apr 2000. [Online]. Available: http://electronics.howstuffworks.com/cameras-photography/digital/question362.htm

[7] "Types of sensors for target detection and tracking," Nov 2013. [Online]. Available: https://www.intorobotics.com/types-sensors-target-detection-tracking/

[8] "Gps accuracy," Oct 2016. [Online]. Available: http://www.gps.gov/systems/gps/performance/accuracy/

[9] R. MacLachlan, "Capacitive sensor introduction." [Online]. Available: http://www.cs.cmu.edu/~ram/capsense/intro.html

[10] "Inductive proximity sensors." [Online]. Available: http://www.balluff.com/balluff/mus/en/products/inductive-sensors.jsp

[11] "Arduino." [Online]. Available: https://www.arduino.cc/

[12] "Raspberry pi." [Online]. Available: https://www.raspberrypi.org/

B. *Revision History*

V. Weekly Blog Posts

For the duration of this project each team member kept a weekly blog which was updated with individual progress, problems which were encountered, plans for the following week, etc. This section of the document will contain all weekly blog posts for each team member during the project. The blog entries will be organized by term, then by week, and lastly by the team member.

A. *Fall Term*

Week 3

Zachary DeVita -

We just met with the entire team of engineers in our group Tuesday evening for the first time. This essentially just validated what we thought we were doing for our project already. We are working with a sub-team of mechanical engineers as well as a sub-team of electrical engineers to design a soda-can sized ”satellite” which will eject from a rocket at 12,000’ AGL and land safely on the ground using a parachute. This satellite, payload, robot, or whatever you want to call it, will then drive autonomously to a specific set of coordinates through some piece of the Nevada desert.

Progress? Nope, no progress. We just met with our entire team for the first time, and our team of computer scientists have yet to even meet with out client. We have requested a couple extra days to wrap up our first assignment for the course, the ’Problem Statement’. Most of the assignment is finished, it just needs to be revised once to make sure the material is cohesive, and then put into a Tex based format to be submitted.

Problems encountered would be that our project is so undecided upon, ambiguous, that it makes it absurd to write a problem statement at this point. Aside from that, we haven’t been able to set up any kind of group meeting with our client. Our client is completely leaving all decisions made for the project up to us which is great for the most part, and makes communications between us less necessary, but it seems like there might be some important information that would be pertinent to the problem statement.

Paul Minner -

This week, we had our first meeting with every member of the ARLISS group. ARLISS group meetings will be Tuesdays from 6:30 - 7:30 PM from now on. At the meeting, we learned that our can satellite will be a rover which has to maneuver to a specific destination after landing. Until next meeting, we need to think about how this can be done, such as what kind of obstacles the rover needs to be able to go around and what sensors we will use for the robot’s ”senses”.

Steven Silvers -

This week we met for the first time as a group and made contact with our client Dr. Squires. We are currently working on the problem statement document as well as getting our github group setup. A problem that we currently face is that our problem is very open ended, and also depends on the work of two other capstone groups at OSU.

Zhaolong Wu -

This Tuesday, ARLISS Micro Satellite Project group had the first weekly meeting, we have 10 people on board, from ME, ECE and CS departments. In order to fit everyone’s schedule, we proposed to have every Tuesday 6:30-8:00 our official meeting time throughout this fall term. The ME team presented us the main idea of this project. The project will have two options to develop with, MISSION or Rover, it appeals to me more that we will go with the Rover option because it seemed more fun and challenging, until next meeting we will need some ideas in the Computer Science perspective that how conquer it.

Week 4

Zachary DeVita -
We just met with the entire team of engineers in our group Tuesday evening for the first time. This essentially just validated what we thought we were doing for our project already. We are working with a sub-team of mechanical engineers as well as a sub-team of electrical engineers to design a soda-can sized ”satellite” which will eject from a rocket at 12,000’ AGL and land safely on the ground using a parachute. This satellite, payload, robot, or whatever you want to call it, will then drive autonomously to a specific set of coordinates through some piece of the Nevada desert.

Progress? Nope, no progress. We just met with our entire team for the first time, and our team of computer scientists have yet to even meet with out client. We have requested a couple extra days to wrap up our first assignment for the course, the ’Problem Statement’. Most of the assignment is finished, it just needs to be revised once to make sure the material is cohesive, and then put into a Tex based format to be submitted.

Problems encountered would be that our project is so undecided upon, ambiguous, that it makes it absurd to write a problem statement at this point. Aside from that, we haven’t been able to set up any kind of group meeting with our client. Our client is completely leaving all decisions made for the project up to us which is great for the most part, and makes communications between us less necessary, but it seems like there might be some important information that would be pertinent to the problem statement.

Paul Minner -
This week, we met with the whole group and discussed the specifics of the rover. Next meeting, we should know exactly what the rover is going to be. We also got our Problem Statement document finished and turned in. Next week, we plan to meet the whole team to finalize our rover design, update our problem statement document, and write our requirements document.

Steven Silvers -
This week we turned in our initial draft of our problem statement and met with the ME and ECE teammates who are also working on the project. We are looking to figure out a definite timeline for completion of the hardware for the project so that we can plan out the software development.

Zhaolong Wu -
The team gathered together on Tuesday during the weekly meeting, we have narrowed down some possible approaches for this autonomous target finder vehicle, an air drone/glider UAB approach or a rover. We analyzed the pros and cons for both approaches. Air approach will involve power supply issues, as well as the software complex level. Ground approach seems is the best way for now due to the possibility of getting it working, but there are still some issues of it, for instance how to get over oversized bumps or so. Speaking on the software aspect, we still don’t have too much ideas of how we are going to do it. All I can think now is we might need to use some types of sensors, and use some existing libraries and algorithm such as OpenCV sor so.

Next week All team together we will make the final decision on which direction of approach we will use.

Week 5

Zachary DeVita -
This week our team wrapped up the Problem Statement assignment, and worked on our rough draft of the Requirements Document. Communication in the group seems to be adequate enough to get our tasks finished on time so far, which is good. We have met with our extended team this week as well. The only real challenge that we encountered is in the description of our assignments. Some of the assignment descriptions seem vague and incomplete which has created a continuous need for clarification. That, and the fact that we can’t possibly have a legitamate set of requirements because the hardware and design for our project hasn’t been created yet.

There isn’t much to describe about our project at this point; everything is in the air. There is no way for us to be certain of the hardware we will be working with, or how the design of the ROV will look. Plans for the coming week are going to include finishing the final draft of the Requirements Document, or at least a ”final-of-the-first-revision.” This document is a evolving document, but a final draft of the initial revision will be completed this week.

Paul Minner -
This week, we revised our Problem Statement, and wrote our rough drafts for our Requirements Document. We also met with the entire group Tuesday night. We have now decided the rover is going to be a ground based tread design, as opposed to wheels or a flying design. We still don't know the specifics of the sensors, though. This means our requirements documents will be vague, and need to be revised when the sensor specifications are decided. Next week, we will meet will the whole team again to discuss more specifics of the rover, and write a final draft of our Requirements Document, as well as create a Gantt Chart for the rest of the year.

Steven Silvers -
This week we met with the ME and ECE teams and determined what approach we wanted to take with the rover design. As a CS team we wrote a rough draft of our requirements document to be turned in Friday. Our plan for next week is to revise the requirements document, as well as meet with the ME and ECE teams again.

Zhaolong Wu -
This week we were mostly working on the requirement document. During the weekly meeting we voted to develop a ground based vehicle. So now we have a general direction to go with, as the CS team we provide data communication between ECE and ME, so we are now waiting on the ECE team to finalize their selection of electrical parts then we will hopefully in next two weeks get a basic software structure.

Week 6

Zachary DeVita -
This week was working primarily on the requirements document. I modified the previous latex template to include a table of contents. Finishing the formatting on this was a small victory. I had to manually change many of the defaults that were used in the 'IEEETran' document class which turned out to be a nightmare when you are simultaneously using the 'babel' package. There were very obscure and specific workarounds that were necessary for this combination of styling which made in extraordinarily difficult to change the numbering scheme on the table of contents and the section titles. When the document was done it looked amazing though. Everyone on the team pitched in, did their part, and the document came together and was turned in on time. Our extended team, including the mechanical and electrical engineers, did not meet this week. It seemed unnecessary to meet at this time, and many people had midterms this week. We met with our mentor this week, on Monday, to discuss the project and upcoming projects as well. As a team, we briefly discussed next weeks project. Next week we will be working on the technical review. This may be the most influential assignment we complete this term because it should help us determine the final design and framework of our software.

Paul Minner -
This week, we finished the Requirements Document. The entire ARLISS team did not meet today, but the CS group did meet to discuss changes which needed to be made to the Requirements Document. We met with our advisor on Monday, who told us our Requirements Document looked good so far. We each then completed a portion of the document over the course of the week, and on Friday, met to make minor changes and get the paper turned in. We met with our mentor to get the document signed as well.

Steven Silvers -
The plan for next week is to work on the technical review as well as meet with the ME and ECE teams to learn about what progress they have made. Since last week we have finished our requirements document and have began researching autonomous driving systems. One problem I have personally encountered is that I have been sick on and off since the beginning of week four, which makes finding time for capstone plus two other courses difficult.

Zhaolong Wu -
During this week we spent the most of the time on finalizing the requirement document, setup some early goals. The Tuesday meeting is cancelled so we didn't get the chance to talk with ECE and ME team. Next week the ME team will start developing the project physically so we will have more specs on the hardware hopefully start thinking about the actual implementation.

Week 7

Zachary DeVita -
This week our team worked on the technical review for the class. Our team brainstormed 12 different topics to be covered in the technical review, and then split them up so each person was assigned three to write about. I personally reviewed the three most common languages for programming micro-controllers. This was an interesting review, and I learned a lot. This review put C++ on the map as a possibility for a language, and it may even help to determine some of the hardware components we request for the project, i.e. a 32-bit controller. I also compared and contrasted different sensors and systems used for object recognition. From the research I have thrown out photoelectric sensing as an option, and I was able to determine that ultrasonic radar in combination with CMOS imaging sensors would create the best system for the environment we will be operating in. The last topic of mine is comparing methods of recognizing the pole that must be bumped at the end of the expedition in order to finish. We did not meet with our extended team this week either. They had nothing new to share with us. We did meet with our mentor this week to discuss the technical review. Issues we are having would be that we still have not the slightest clue what hardware we will be working for, so every assignment we do in this class in comprised of educated guesses. I guess it is still better to think about these things than not, and maybe this experience will help us to make better decisions on what hardware our team does decide to go with. Next week is the design document. We haven't really talked about this assignment as a group yet, but I'm sure the time will come tomorrow. Hopefully I can get a good start on the poster next week as well.

Paul Minner -
This week, we just worked on the Technical Review Document. There was no meeting with the entire group this week, because the other groups didn't have any new information to talk about. Our CS Team did meet to discuss which pieces each of us would write about for the technical review. On Thursday, we finalized what everybody was going to write about. Next week, we will have a meeting with the entire group, and hopefully we will get more information about the types of sensors the ECE team would like to use. We will also start work on the Design Document. Some problems we're having right now is being sure the information we include on the Technical Review will be correct, because some pieces are also up to other teams.

Steven Silvers -
This week we developed our Technical Review document. We did not meet with the other subteams for the project because they had nothing to discuss with the group as a whole. We met as a group on a couple occasions to go over the plan for the technical review. Our plan is to look towards the next document for the class. I would also like to start discussing tasks for winter break so that we are ready to start at the beginning of winter term. The biggest Problem is that it appears we have conflicting timelines with the ECE and ME teams. Our work is dependent on what they develop, however we cannot begin some of the larger pieces of our software system without knowing what we will be working with.

Zhaolong Wu -
This week, we mainly worked on breakdown this project into many(12) pieces, and each of the team member has picked 3 pieces to evaluate the possible implementation methods, pros and cons. To build up the tech reviews document. Next week, we are going to start the first part of the implementation, the design document. It should be fun.

Week 8

Zachary DeVita -
This week our team focused on finalizing our technology review, and managed to get that turned in early in the week and on time. Aside from finishing the technology review, our team has begun working on the design document assignment. I personally have not begun writing any of the document, but I have started doing some preliminary research for the assignment. I believe the assignment should go relatively smoothly for myself. I do need to figure out what one of my topics could be; in my tech review I compared the 3 most commonly used languages for programming microprocessors, but I don't think this is an appropriate topic for a design document. That is something I'll need to speak with Dr. Kevin McGrath about. We met with our mentor this week, as usual, and he made it sound like we are on track and doing well. We also met with our extended team on Tuesday evening. They were able to give me a prototype CAD image of the satellite which will come in handy for the poster our team is making. They didn't have anything that was too pertinent to discuss, but they are speaking with the point of contact for the competition in hopes to get the precise specs for the satellite. That seems like an important step before we start building the satellite. Next week I, and the rest of the team, will be finishing up research for the design document and writing the final version. More meetings next week, and finishing the design document.

Paul Minner -
This week, we finished our tech review, met with the entire ARLISS group, and started plans on the design document. After meeting with the group, we now have a better idea of what sensors will be used, although we still aren't sure if a camera or sonar will be used for the obstacle navigation system, which is a problem for the design document's accuracy. Additionally, it was decided the GPS will be used to calculate altitude for the parachute deployment, which will now need to be changed in the tech review later. For the design document, we created a simple outline, as well as some ideas for what to include for some sections. Next week, we plan to focus on the design document. There should be another meeting with the entire ARLISS group next week where we will nail down what sensors will be used, and we will talk about how the different technologies will fit together for the design document.

Steven Silvers -
This week we worked on planning out the Design document and also submitted our Tech Review. The plan for the following week is to work on writing the Design document as well as begin planning the progress report and poster. The biggest problem I see at the moment is Thanksgiving break cutting into work time.

Zhaolong Wu -
During this week, we, three teams finally gathered together and discussed the detailed design, the ME team has the basic dimensions specs done for the ECE team, so that they have an idea of what components to use in order to fit into the CanSat, we settled we are going to have a 9V 1200 1500 mA battery for the main power supply, two servos, two motors(brushless or brush), GPS modules for the CanSat. I'm still doing researches on whether we are going to have a camera/SONAR/RADAR for the obstacle detection, as well as if a accelerometer or a compass is needed to enable the autonomous feature.

Next week, we will mainly focus on the design document and the end of term presentation.

Week 9

Zachary DeVita -
This week was spent working on the design document. I managed to finish up my sections of the design document early and am currently done. I formatted the latex for the document using the requirements for the assignment as a guideline and posted it on github for the other members to add their sections. Our team as a whole should be wrapping up all their sections this weekend as well so that we are able to get a running start on the progress report assignment. Our team met with our mentor, as well as, the extended team of engineers this week. Not too much new information to speak of. A note for next week, I need to speak with the extended team and get some much needed information. It is time that our team knows the exact requirements for the competition and what sensors we are using. The extended team needs to have these defined in order for us to effectively accomplish our assignments, and, so far, it has not been that way. The assignments are getting incredibly difficult without this information now that we are getting into the specifics of design. Next week we will be working on the progress report, and hopefully the presentation as well. This is an absurd amount of assignments to all be due in such close proximity of time. Hopefully our planning ahead will pay off in this regard though.

Paul Minner -
This week included Thanksgiving, so our group didn't meet much this week. There was a meeting on Tuesday with the whole group, but I was unable to attend this week. We did manage to finish the majority of the Design Document this week, though. Each group member worked on the same pieces as we did for the Tech Review. Next week, we plan to finish the Design Document, and finish the majority of the final progress report. Some issues we have had is being sure if we will actually stick to the design document, because some decisions are difficult to make without any testing to see how well it works. For example, the algorithm to get the rover unstuck from obstacles may change if we find that the algorithm we thought of doesn't work very well.

Steven Silvers -
This week we finished our design document. We decided to get it done early giving us more time to plan and work on the Progress Report as it seems that the Progress Report will require a but more work. The plan for this week is to layout the progress report design and also to decide what we will do for the video presentation portion. The only problem I foresee is being able to manage multiple projects over multiple courses all being due around the exact same time.

Zhaolong Wu -
This week, I mainly worked on the final design document, I went to the meeting on Tuesday and it was great to hear that the ME team have finalized the basic design motor, motor selection, the ECE team are ready to purchase some sensors. We decided that we will use the Raspberry Pi zero and make the microprocessor. Mean while, we still need to figure out how to make the tread.

Next week will be the last week of this quarter, we will wrap everything up and getting ready for the next phase of this fun project!

Week 10

Zachary DeVita -
Week 10 was a busy week indeed. Our team managed to finish the majority of our Design Document over the weekend so there was not too much time spent working on it. Everyone did there part and researched their topics, wrote up their portions of the document, and we managed to get it turned in on time. We also made sure to get a considerable head start on our progress report. I finished up my sections and the template for the progress report by Thursday or Friday so that I could focus on final projects in other classes, and the presentation for this class. So far everything has gone well, or a least worked itself out. We met with the rest of the team on Tuesday to discuss hardware in more depth than usual. Nothing too concrete has been chosen as far as hardware, but some ideas were thrown out. The electrical engineers apparently want to use an AtMega128 processor. I suspect that it is only because that is what they are accustomed to and not because it is the best tool for the job. Ideas for sensors were also mentioned. Hopefully they consider the CMOS and the ultrasonic sensors that I researched and suggested to them. We finally got the specifications for the competition. At least it is before we start implementing. Our team plans on putting forth some effort and working on this project over break which would be nice. I've requested some supplies from Dr. McGrath so that I can work on some of the coding over the break. Next week is finals, and I still go to finish up that presentation, but, other than that, I'm done.

Paul Minner -
This week we finished up our Design Document, and started working on our Progress Report and presentation. For the progress report, we decided to split the weeks up and each write a couple weeks. We still need to write the rest of the Progress Report. The Design Document we got signed by our mentor and turned in on time. We had a group meeting this week, where we discussed more specifics of how the different components of the rover would work. Over break, we will do more research how to implement our components, and begin working on them. Problems we face is we have to start writing code before we have the hardware which will run the code, so some code may change.

Steven Silvers -
This week we finished our Design Document and got it signed and submitted. We've begun work on our progress report, with the plan to begin recording the presentation over the weekend. At this weeks group meeting we had further discussion about the actual design and implementation of our rover, as well as brainstormed ways to best get out of the can once we have landed. The plan for break is to continue researching methods of implementation and various technologies, as well as starting to code framework for the project. This will be difficult as the ECE team still has not locked down what they will be working with hardware wise. Until the hardware has been figured out, it will be difficult to make any significant progress on implementing the software.

Zhaolong Wu -
It's the final week of this term, we had not to much in beside finishing up the design document and making the progress report and final presentation. We had a team meeting on Tuesday, mainly discussed some protocols we are going to use on the rover, as well as some test cases. Today the Mechanical Team got an email from the ARLISS competition committee, they sent us a detailed CanSat competition specs, we will start working on the actual implementation through the break, and hopefully will have a version 1.0 done by the midterm winter term.

B. Winter Term

Week 1

Zachary DeVita -
Week 1 of winter term. I actually, and sincerely, worked on the project over break. Nothing too impressive, but I have a working program which converts a colored 24-bit .bmp image into a two-dimensional array of binary values. Essentially zeros in the array indicate that the change in grade of the terrain is safe to traverse, and that there are no obstacles present. This program needs some serious testing though, and I'm not certain how to create images that would accurately represent the terrain in Black Rock Desert. I have worked out about all I can without a proper way to test, but I think it is a good start to the most difficult part of the programming for our project. It seems that our electrical engineers have decided to choose an 8-bit AtMega128 processor for the project. This means we would be programming in only C so I spent some time researching how to mimic some of the C++ features that our team was hoping to utilize. Aside from this, it's the first week and our team hasn't even had a meeting so there's not much to write about.

Paul Minner -
This week, we started getting back into the swing of things. We've scheduled our weekly meetings with our TA for 11:30 AM Tuesday, and are currently getting our weekly meetings with the entire group figured out. Over break I researched about the components needed for the parts of the project I'm responsible for, such as GPS and cameras. Soon, we should begin actually implementing our design. Our only implementation problem is that the hardware isn't available yet, and therefore we will have to simulate it.

Steven Silvers -
Start of winter term, over break I studied various autonomous driving methods since that is the main part of the project I am responsible for. I also looked at different hardware implementations and how they would possible effect our programming decisions. We are communicating with the rest of the team to find a weekly meeting time, and have set our weekly TA meeting for Tuesdays at 1:30pm. Hopefully our whole team meeting time will be set soon so the ECE and ME teams can update us on their progress.

Zhaolong Wu -
The winter term kicks off this week, we had one class meeting and instructor McGrath officially announced that we are in the implementation phase of the capstone project, we setup the mentor meeting time on every Tuesday 11:30pm to fit every team member's schedule. We are still waiting on the new weekly team meeting time with the ECE and ME team, and we are going to talk with the client to let her come to the first team meeting to give us a general direction.

During the winter break I spent handful amount of time on researching the autonomous driving vehicle especially on GPS works in it, hopefully we can get our first prototype done soon.

Week 2

Zachary DeVita -
Not much to report for this week. We met with our mentor for the first time this term, and he gave us some due date info for the project. We are supposed to have a prototype for the project, a fully working program, written by week 5 which is comical. We still haven't met with our entire team which includes the other engineers participating in the project. We do have a time to meet with them next week so hopefully we will have something to work with at that point. Much of our programming requires interacting with hardware that is being used on the "satellite", but that hardware is being built by the electrical engineers. Their due date for finishing the hardware, which we must have in order to finish our programming and begin our testing, is approximately the same due date as we have to finish our programming. There seems to be a major flaw in how these multi-disciplinary projects are organized. I'm no rocket surgeon, but it would make infinitely more sense to have the senior design projects, which are cross-disciplinary, have their implementation terms staggered so that the hardware is being completed at the end of one term, and then the programming could begin in the subsequent term. Mind-blowing, isn't it.

Paul Minner -
This week, we haven't accomplished much. Mostly, we have started planning what parts need to be completed for the alpha build due at the middle of the term. The entire ARLISS team is working on setting up a time for group meetings. Currently, Wednesdays at 6pm - 7pm is thought to be the time.

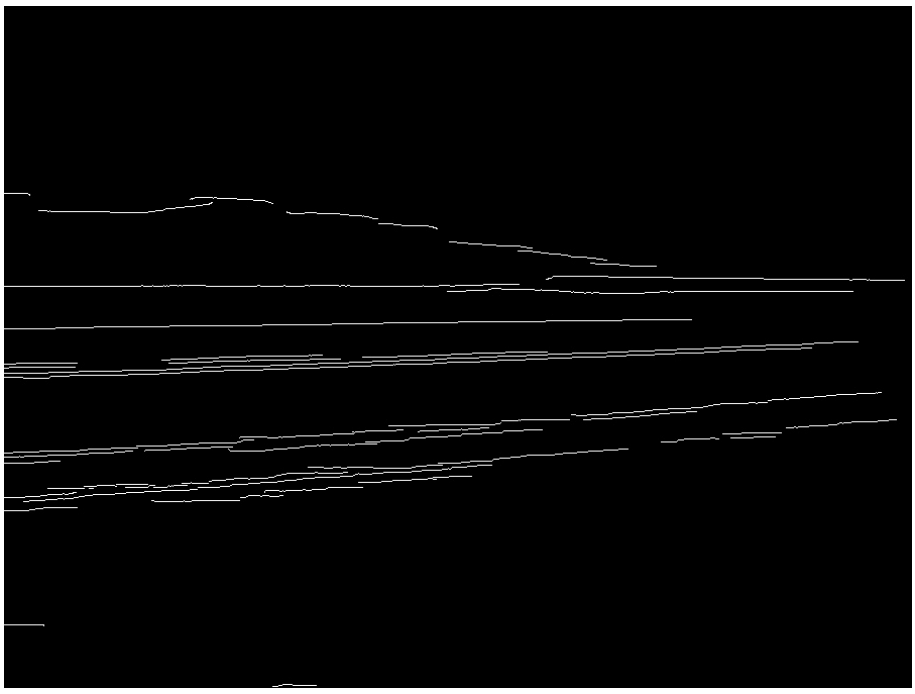
Steven Silvers -
This week we met with our TA Franks for the first time this term, and discussed what is expected of the group as far as implementation of our project for this term. We as a group decided what a successful alpha version of our project would look like. We decided since the other two teams on our project would mostly likely not be finished with the hardware implementation until the end of the term that for our alpha release we would write simulators for our individual modules to demonstrate that they function properly. The goal for the Beta release is to have to code implemented on hardware, but that is entirely dependent on the progress made by the ME and ECE groups.

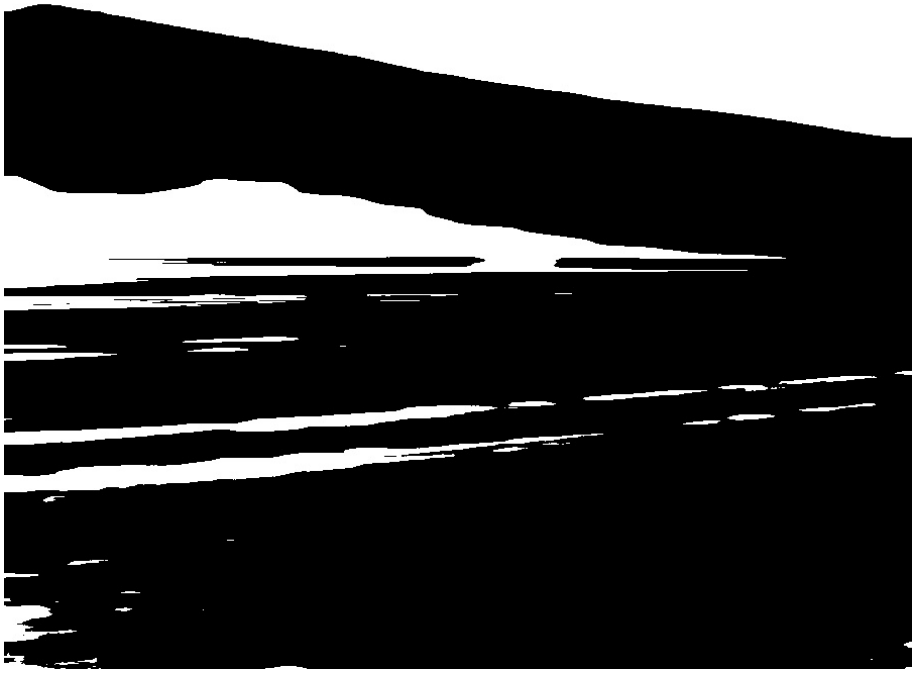
Zhaolong Wu -
During this week, we met our TA for the first time this term. We set the weekly meeting with him on Tuesday 1:45-2:00 pm, he mentioned during the meeting this week that the basic structure of this term in term of our project, we have two check offs, alpha version which will happen in midterm and beta version that I believe would be the final. So we suppose to have a functional physical prototype done by the midterm despite this is way beyond of our control because we have to get the ME team and ECE team to finish the actual rover then we can start to program it. So we have settled down that we will have something simulations done by the midterm, and each person in the group will be in charge with topics that we wrote in the tech-review from last term. We will have to send an email to the TA to specify though.

Other than that, we didn't have to much, I'm still doing researches on my part which I believe that's what every team member is doing at this moment, we still have not met, or heard any people in the ECE and ME team yet, we hope we will do next week.

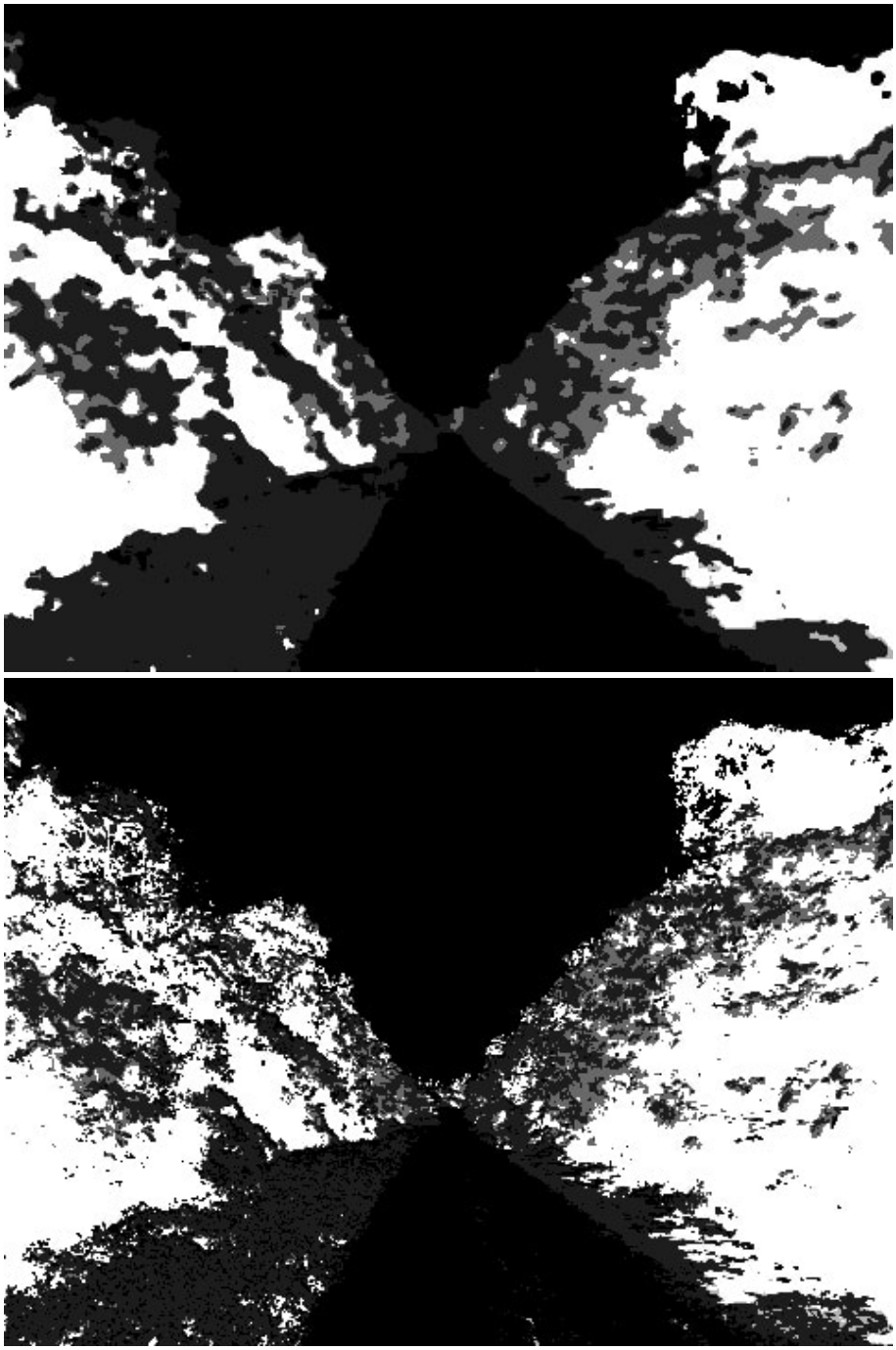
Week 3

Zachary DeVita -
This week our team has decided on a microcontroller which means we know our processing speed, and which means we know which language we can program in. This came as great news as we can now begin our programming for the project. Our team will be writing the code for our project in C, and I have begun writing a program which manipulates images and detects the edges in the images. This utilizes the opencv library which is an amazing, powerful library. So far I have determined a method of converting images which eliminates the noise in the image, and that seems to be the most effective configuration of function variables for our purposes. I will attach images showing the progression. These include an original photo taken in Blackrock Desert and two different methods of eliminating noise which have shown success. There are many different methods of image manipulation which can be utilized for eliminating noise in the image. For both of the methods which I have found, I first convert image to grayscale, then I blur the image along the horizontal axis. For one image I then use the Canny filter to finish the process. On the other image I run it through a binary inverted threshold. Both these methods seem to accomplish what I am looking for so we might rely on testing to determine the best. Now I need to convert these images to a double binary array, or do I? The next step is kinda in the air. The original plan was to convert to a binary array, but I'm not sure if that is the most efficient method.





Here's some other configurations of filters that might be considered as well. Neither of these use the Canny filter; one is blurred, and the other is not blurred.



Paul Minner -

This week, we finally met with the entire ARLISS group. At the meeting, we discussed where each group was, and upcoming due dates. We also finally know exactly what hardware we will be implementing our software on, which is a Raspberry Pi Zero. Each member of our group has been working on implementing their portions of the project, and will continue to do so next week. There should also be another large group meeting next Tuesday to discuss progress with the other teams. One issue we are running into is simulating our software. Some components are difficult to simulate, which means we won't know exactly how the software will react until the hardware is finished and we can test the rover itself. Other than that, simulating most modules of our software for the alpha build seems doable.

Steven Silvers -

This week we held a team wide meeting with the ECE and ME groups to better figure out timelines and what everyone is working on. It sounds like the ECE team has finally settled on using a Raspberry Pi zero as the main

board for the rover, giving us a better idea of how we need to implement our code. Progress has continued to be made on developing and simulating our individual code pieces in preparation for the week 6 alpha release.

Zhaolong Wu -
We finally had the first all major groups together meeting this week, we set the weekly meeting time on Tuesday from 6-7pm. The meeting went very well and it was surely very informative for our computer science sub-team, the mechanical team told us the specs of the rover design and what there schedule looks like for the winter term, just like us, they also need to have a working version of prototype to be done by the week 6, the working version means the rover has to be ready to be tested, under basic functional requirements. For example, being able to drive from given waypoint A to B, driving around a circle, etc.

We split work for now to the alpha version release, Zach is going to continuing working on the camera, Zhaolong is going to work on how to use GPS data, Paul is going to work on the parachute deployment and Steven is going to deal with the board.

Week 4

Zachary DeVita -
Our team met with our mentor and our client this week. Our client showed up at our team meeting which was very beneficial. We were able to get some of our questions answered by here. We also were able to decide on some important stuff at the meeting regarding hardware we will be working with. We now know all the hardware which we will be interacting with! We had a class lecture this week as well. We were given a description of our mid term progress report and some details of the assignment. As far as progress on the project, I have finally been able to get in contact with Bill Smart who is one of the leading researcher in robotics and machine learning here at OSU. He has put me in contact with some of his researchers so that I can get some assistance working on the object detection algorithm. I have yet to hear back from them, but gaining his input and assistance is a huge step in implementing our object detection. The alpha version requirements of our capstone projects has been defined by Kevin McGrath. His requirements make it seem far more plausible to have completed by week 6.

Paul Minner -
This week, we've continued to work on the alpha release of our project. This consists of creating the modules for each section of the journey, and simulating the input from the sensors to make sure they work correctly. We also got our OneNote page set up. We still need to revise our documents to reflect recent changes in our plans, and we need to write our progress report.

Steven Silvers -
This week was mostly business as usual, we continued working on the alpha version of our project, as well as had an all group meeting that included our client Dr. Squires. My development module has been changed in that it will be getting a .mat file as input instead of a 2d binary array. This change will need to be reflected in the requirements document.

Zhaolong Wu -
This week we had 2 major parts going on, during the all group weekly meeting, the ME team leader Kyle laid out the projected expense, as well as how much for now to make one single rover, which is roughly about 299 dollars. As the CS team for now we just need 2 Pi Zeros and one camera. Our client Dr.Squires showed up as well. We discussed some details of the CanSat competition, like what the target pole looks like in order to program the vision system. Per competition rule the rover need to have the GPS coordinates recorded in a time interval, which is fairly easy task, we are planing to put a little script in our program that let it create a log file automatically at the end.

During Thursday's class meeting, both instructors went through things we need to do in the future of the term, I'm aware that first we need a team leader, due to the over crowded classroom and the time conflicts with another class, I wasn't able to find the team members and had to leave class early, I will send out an email to ask.

Week 5

Zachary DeVita -
This week our team met with the other engineers for our project and with our client on Tuesday. We had some new assignments come up for the class this. One of the assignments is the progress report. Our team needs to update the original document with what each of us has been doing for the last five weeks of the term. We also need to make another presentation for our project which will be approximately 30 minutes long. We are also updating our documents from last term with any necessary changes and publishing them on a OneNote notebook. This week I created the OneNote notebook for our project and I've been working with some of Bill Smart's graduate students to develop a machine learning algorithm for locating the pole at the end of our satellite's trek. These types of algorithms are pretty foreign to me so there will be no way that it gets finished before our alpha release but hopefully for our beta release. I've been doing a lot of research on machine learning object recognition and it is extraordinarily interesting to me. I hope I am able to get this chunk of code working.

Paul Minner -
This week, we made significant progress on our modules in preparation for the upcoming progress report and demo. I finished the Parachute Deployment Module, as well as the Getting Unstuck From Obstacles module, with testing programs for each module. Our current plan for the alpha release is to run each module as a separate program being tested with simulated input from the GPS and Camera. Our main problem is some pieces, like converting the image to binary, is difficult to know if it will work correctly until we can use images that are closer to the images the camera will take during the competition. For the alpha release, we aren't worrying about this, though, and are just attempting to have each module work to some degree. Next week, we plan to write up our progress report and demo our alpha release.

Steven Silvers -
This week was spent doing work on the alpha version of the modules I am responsible for, mostly the obstacle avoidance system. Because of this modules complexity it was decided that the alpha would be a lower level "prof of concept" and full functionality would be added by the 1.0 release. I edited both the tech review and the design document to reflect changes that had been made to the project for the sections I am responsible for. The biggest edit was for the control board, as the ECE group finally settled on what board they would be using and it was not what the document said would be used.

Zhaolong Wu -
This week we were mainly focus on revising the design document and get the mid-term assessment ready. I'm still trying to test the gps functions like feed dummy data in and trying to see the systems response. Also I did some researches only and wrote a obstacle detection pseudo code. Next week we will finish up our progress report and get more testing done.

Week 6

Zachary DeVita -
This was a busy week. Our team spent the entire week wrapping up our presentations, middle of the term progress reports and updating our documentation. I believe the only documents our team needed to update were the Tech Review and the Design Document. I had to change pieces in both of these documents; anything referring to us using the ultrasonic sensor had to be removed. My updates were completed at the beginning of the week, as well as, most of the formatting for our team's OneNote. This allowed me to focus on the progress report and the presentation. My section of the presentation ended up being just over 7 minutes which was the expected length for each of our team member's parts. Everything got done just in time to turn in on Friday. Steven concatenated each of the presentation pieces, and I uploaded the finalized documents to the team's OneNote.

Paul Minner -
This week, I finished up the find and touch the finish pole module, so all programming I need to complete for the alpha release is done. I also worked on writing the progress report and presentation. Everything is going smoothly, and we should be able to finish our alpha release by the due date.

Steven Silvers -
Week six was focused entirely on the upcoming alpha release and progress report. We decided that each team member would write their pieces individually and then combine them into one document Thursday, giving us plenty of time to submit the report on Friday. We got together with the ECE and ME teams for our weekly meeting, which ended early because all three groups were busy with progress reports and had nothing of significance to share. The ME team is uncertain of their capstone number, which hopefully they will figure out soon so that we can register as a group for Expo.

Zhaolong Wu -
During week 6, I spent most of the time to do more testing on the navigation system, the two output images above shows that the algorithm in the navigation program for getting the shortest path and initial heading, the output is matching the online GIS calculator. I'm currently waiting on to get the GPS, pcb board, and motor controller, etc. from the ECE team so I can do the integration tests. For the current testing method I used the location coordinates from the competition site Black Rock, NV, I hard coded two sets of coordinates as start point and finish pole.

Week 7

Zachary DeVita -
This was a slow week for progress. Our team met with our mentor as usual, and our extended team had its meeting too. Much of this week was just discussion and planning for the last few weeks of the term. There was some discussion at our team meeting about the mechanical engineering requirements. It seems like they may need some basic driving functionality as one of their end of the term deliverables. I handed off the raspberry pi and other components to Paul so he could install the operating system on it and hopefully get the camera working. Paul also discussed merging some of the programming pieces together in the next couple weeks. We still don't have a finished ROV or the rest of the sensors and components to work with so we can only do so much. I asked the mechanical and electrical engineers for their team numbers so that I could register our team for expo. Hopefully they will know their numbers by next week. The current plan for the coming weeks is to start combining our individual programs together into a single program. My personal task is to finish the machine learning algorithm for locating the goal at the end of the ROVs expedition. I have been researching this, and working with Bill Smart's grad students to learn what I need to do this.

Paul Minner -
This week, we didn't get much new work done, but we have planned out our work for the rest of the term. At the weekly ARLISS meeting, we learned that the ME team will need basic drive from point to point functionality in the next two weeks. This means we will have to get a portion of our program installed on the raspberry pi and working. I now have the raspberry pi, and am planning to install an operating system on it this weekend so we can run our program on it. After getting basic functionality working, we will have to combine all the pieces of our program together and test the program as a whole. Still, our problem is we can now test on the pi, but we don't have access to the sensors and motors that the completed rover will have, so testing on the pi will really just show us if our program is too computationally expensive or not.

Steven Silvers -
Week seven I took a little bit of time to catch my breath and recover from being sick after a stressful week six. We are still waiting on the mechanical and electrical teams to figure out their capstone group numbers so that we can register for expo. The plan for week eight is to begin integrating our individual modules into a single system and to begin testing our system on the Raspberry Pi Zero hardware.

Zhaolong Wu -
During week 6, I spent most of the time to do more testing on the navigation system, the two output images above shows that the algorithm in the navigation program for getting the shortest path and initial heading, the output is matching the online GIS calculator. I'm currently waiting on to get the GPS, pcb board, and motor controller, etc. from the ECE team so I can do the integration tests. For the current testing method I used the location coordinates from the competition site Black Rock, NV, I hard coded two sets of coordinates as start point and finish pole.

Week 8

Zachary DeVita -
Not a lot going on this week. I handed over my computer vision software for the obstacle detection to Steven so he can merge the two softwares into one program. The ECE team is still waiting on some parts for the rover. The mechanical team has an acrylic frame, but hasn't got any progress getting any of the parts onto it yet.

Paul Minner -
This week, I focused on getting the raspberry pi ready for testing with our software. I made sure the operating system was installed, and I started loading my pieces of the program onto it. The most important part to test is the camera related pieces to test how fast they are, but those aren't finished yet. Next week, all the pieces should

be finished so we can test individual pieces and combine all pieces into a single program. Our only problem now is the fact that the camera is so low to the ground. We won't be able to detect objects and obstacles from very far away, so we are urging the mechanical engineering team to allow the camera to spring up so that it is further from the ground.

Steven Silvers -
I spent most of this week attempting to merge my obstacle avoidance module with Zach's imaging module, with little to no success. I think what could be causing the problem is Zach and I may have opencv configured in different ways on our workstations, which is preventing me from properly compiling his source without making major changes. I will be sure to discuss with him in the coming days to see what we can do to fix this problem. Parts are still coming in for the ECE team, at the last group meeting the servos had finally arrived that will be used for unpacking the rover from the payload. The ME team has a frame cut from acrylic, but is not yet assembled and last I heard from them they had not started on the wheel system yet. The plan currently is to have all of our modules working in a single system by the end of the term, and whether or not that gets implemented on hardware is reliant on the progress made by the ECE and ME teams in the coming weeks.

Zhaolong Wu -
This week we had a class meeting to do practice expo pitch and tone. Other than that, I worked on how to update the new shortest route if an obstacle is found and we have to avoid, as well as an servo arm function if the rover falls sideways. I'm still fussy on how the mechanical team is going to release the parachute, I mean what kinda of protocol and is there high level programming involved? Also I'm a bit fussy on the ECE teams controller setup, if they are making the pcb how we are going to call the motor and gps to enable those functions?

Week 9

Paul Minner -
This week, I worked on getting the camera working on the Raspberry Pi. Unfortunately, Nobody was sure how to connect the camera to the Pi, so the camera connector got damaged. I've ordered a new Pi which should arrive Tuesday, but that set back testing on the camera. Next week, I plan to integrate all the individual pieces we have into a single program, although it won't yet include all pieces because some modules aren't yet finished. We are worried that once we test our program on the actual hardware, the camera will be too low and we won't be able to detect obstacles very far in front of us. In that case, we will either have to raise the angle of the camera, or lower the speed which the rover travels. Hopefully, this doesn't end up being an issue.

Steven Silvers -
I made quite a bit of progress this week, I managed to successfully install Opencv on Linux, port Zach's imaging module so that it would run on Linux and not just from Visual Studio and added functionality so that it outputs a .csv file that I can easily open and use with my obstacle avoidance module. The .csv file contains values of zero where there is empty space(safe to drive) and nonzero and sometimes non integer values where there are edges. I just need to make a few edits to my module that allows it to adapt to the size of the .csv file, read any nonzero value as ”1” and leave the zeros as they are, and it should be ready to check against real images.

Zhaolong Wu -
This week, I did making a little program that using opencv library to identify the center of a shape, it is still buggy and I'm still trying to make it work, once it works I will upload the file to the github.

Week 10

Paul Minner -
This week, I finished up my merged program of everybody's modules which I'm going to demo for the Winter Progress Report. It had a few bugs which needed to be fixed, and I updated a couple modules with newer versions. Unfortunately, the ECE team didn't finish their PCB board, so we will not be able to run our software on the completed rover by the end of the term. The rover should be done by Friday of finals week, though, so we should be ready to implement everything as soon as next term starts and everybody is back. Next term, we will have to do a lot of testing to make sure our software is ready for the competition.

Steven Silvers -
This week I finished combining the imaging and obstacle avoidance modules and began running tests of various images to make sure that the avoidance algorithm was reacting appropriately. Testing for this system will continue in spring term, as well as working on optimization as it currently takes a minute to run on a Raspberry Pi Zero which is too long for what we need. I know for certain there are a couple ways the code can be improved, there are a couple image filters being generated from testing that go unused, and a couple functions with multiple nested for loops can be merged into a single loop. I also laid the framework for the poster, as well as added information on my module to the poster. The only things still needed for the poster are a team photo, updated photos of the rover and of our image filters as well as a paragraph on Zach's imaging module.

Zhaolong Wu -
This week we finally heard from the ECE team, and they said they will have the PCB ready on Thursday of finals week, which makes our hope of having a moving rover complete impossible, we will hopefully have it next term so we can make the rover move!!!

C. Spring Term

Week 1

Zachary DeVita -
This is the first week of the term so our team has yet to meet, and we haven't yet met with our client or mentor. We have decided on a meeting time for both our team meetings and meetings with our mentor this week. I know, over the break, Paul worked on optimizing the obstacle avoidance system so that it would be more efficient. I personally spent my break working on the traffic cone detection software. I developed a backup program to detect the traffic cone, in case the machine learning method fell threw. This method uses a series of image filters to locate the traffic cone based on both its features and its orange color. The problem with this method is primarily that each time the frame goes through a filter every pixel in the image must be iterated over, and this process requires many filters. While I spent the break writing this complex, backup method for identifying the target, I also continued to develop the optimal machine learning classifier. I have developed the most effective machine learning classifier, based on my vector file, using the LBP (Local Binary Patterns) method. The LBP method of training a classifier only takes a few hours where the Haar method takes more than a week, so the idea is that I would find the optimal settings for training using the LBP method and then use them for the the training using the Haar method. Right now I am training a Haar classifier based off the optimal settings from my testing and should have this completed and tested by next week!

Paul Minner -
This week, we set up the meeting with the TA, as well as the rest of the group. The group meetings will now be held Wednesdays at 11:00 AM. Some progress was made over spring break on the project. I worked on speeding up the obstacle avoidance system.

Steven Silvers -
This week we set up our weekly meeting times with our TA, as well as with the ECE and ME teams. We recapped where we currently are in the project, what we have left to do and how we plan to get it done. Paul informed me that he made the code optimizations to the obstacle avoidance module over break, and now it runs on the Pi Zero well within the needed time frame. The plan is to see where the ECE team is with their hardware at next week's meeting, so that we can best project when we can begin system testing.

Zhaolong Wu -
This is the first week of spring term, we had set up our TA meeting time and group meeting time, we are excited to finish up our project.

Week 2

Zachary DeVita -
We met with the entire ARLISS team this Wednesday for the first time this term. No good news from them; the hardware is still not operating correctly. There are several mechanical issues with the prototype. The frame seems to be too tight against the wheels for them to turn freely and one of the motors does not work. This has been a tremendous setback for our team. Without the mechanical parts working, neither the electrical team or ourselves are able to do any testing on the rover, and we are beyond the point where we should have already been testing.

As far as myself, I have finished training my haar, machine learning classifier. It took nine days, but I got a working, accurate classifier for identifying a traffic cone! I am able to determine the pixel location for the center of the cone is as well, which is necessary for Paul's algorithm to work correctly. I did run into some issues with false positives, which is almost impossible to avoid, but I used a feature from my previous algorithm to help verify my results. The classifier is a feature based classifier which uses geometry and shape patterns to recognize an object. When the classifier is made, all images which are used in training are converted to grayscale so the color of the object is not taken into consideration. Since the bright orange color uniquely identifies this object and is an atypical color to be observed in nature, I decided to verify results by checking that the color of the object is orange. I take a few points from the result, convert them to their HSV equivalent, and then ensure they are in the appropriate range of values observed on traffic cones. This method seems to work pretty well!

Paul Minner -
This week, the whole ARLISS team met for the first time. There, we learned hardware components of the rover still aren't working correctly. The frame is too tight against the motor so it won't turn, and the ECE team hasn't finished their PCB board yet, so the rover is still in pieces. This means we are still unable to do the testing we wanted to get done. I did get the rover though, and some of the motors are working, so I will start working on implementing the actual motor control functions into our program. Other than that, we have to wait until the rover is finished before we can do more testing.

Steven Silvers -
This week we got back on track with our meetings, and received a a prototype board from the ECE team so we could begin testing our our system on hardware. the ECE team is still a week or so out from having a finished final product which should give us enough time to implement before expo. There is also discussion of writing a separate demo script to run on the rover during expo. A draft of our poster is due Monday, we still need to take a group photo before the final submission.

Zhaolong Wu -
This week we met with the ECE and ME team on Wednesday, good news is Zach got the cone detection part working and showed off to everyone during the meeting! Bad news is ECE team had to redesign the PCB because the old one is too big and won't fit in the rover, they are going to have it ready by next week, also as the implementation goes they found that the motor has enough torque but not enough speed, which led the rover goes very slow, so we are going to replace some high speed lower torque rover, aside, one motor was jammed so we can't perform any testing on the rover. Expo has been signed up already.

Week 3

Zachary DeVita -
The rover is still not fixed. The electrical team has still not been able to test all of their code. We still do not have a rover to implement and test our code on.

I have improved my algorithm for detecting a traffic cone. I have made it so that when there are multiple results found, instead of throwing out the frame, I now check both results for orange and only discard the results if neither are orange. I also tested several additional methods of verifying results. I tried to also implement a check for the white band, but it seemed that it was severely cutting down the rate that any results were returned, so far more false negatives. I am considering a method were I use image filters and convex hulls to verify the data, but this method would use considerable more resources to run.

I made a little improvement on the obstacle detection algorithm. I added a morphological opening to the series of filters which has helped to remove additional noise from the image.

Paul Minner -
This week, we've still been waiting on the other teams to finish the hardware of the rover before we can do more testing. I am currently working on adding the ECE team's motor control functions to our code so we are ready to start testing as soon as the other teams are finished. Unfortunately, we won't get to do nearly as much testing as we had hoped because the rover was supposed to be finished at the end of last term, but still doesn't work. Hopefully, next Wednesday at our group meeting we will have a fully functional rover to start testing.

Steven Silvers -
This week we met with Kirsten to get feedback on our poster, and worked with the rest of the ARLISS groups to figure out a timeline for finishing the project. It sounds like there won't be much time at all to test our software, as new motors had to be ordered and the ECE team is still waiting on a pcb to come in. It will be cutting it close for sure, but if all three groups stay on top of their tasks we should have a finished product by Expo.

Zhaolong Wu -
This week we are trying to finalizing our code, and still waiting on the ece team to give us a functional rover.

Week 4

Zachary DeVita -
Nothing substantial to mention this week. Our team worked on the poster and got it finished. Our code was finalized this week as well. I know Paul spent some time combining some of the electrical engineering code with our code, but I didn't have anything to do other than working on the poster. I turned in my waiver for expo. We met with our mentor and the rest of the team this week. We still do not have a mechanically working ROV to work on. We still cannot implement our code or do any testing.

Paul Minner -
This week, we've worked on getting the repo ready for the code freeze and submitted our poster. I integrated some of the ECE team's code with our own so our code will theoretically work correctly on the actual hardware, although the only testing I have been able to do is to check if the motors move since the rover still isn't finished. The rover was supposed to be finished this week, but the GPS unit got fried, so now the ECE team is waiting for another one. Hopefully, we will get a finished rover soon so we can complete some testing before expo.

Steven Silvers -
This week we began organizing our repo to be ready for the code freeze on May 1st and also got our poster approved and submitted for printing. ECE and ME continue to have setbacks, the most recent being the ECE team fried the GPS sensor so now we need to wait for a new one. I don't believe the rover will be finished and fully tested by Expo, so we have started making plans for alternative ways to demo our software.

Zhaolong Wu -
This week we finalized our code and poster, we got the poster approved by both the client and instructors, and the poster is sent to the library to be printed for the expo! For the code part, mostly working but still no testing due to the ECE team's in-completion. The deadline delayed several times now and this will making us has a chance to not finish the whole project by the expo, I mean the actual rover, all the other code is done. The ece team still haven't written the servo function though!

Week 5

Zachary DeVita -
Week 5... still waiting on the hardware. We met with our mentor and our extended team this week. The mechanical portion of the project still needs a bit of work. They still need to get a new frame made for the rover. Based on what the electrical team said at the meeting, they should have got the replacement GPS module in the mail by now, and it should have been installed on the rover by now. There is a motor that the mechanical team was waiting on as well.

As far as my contribution, nothing substantial to report. I have been working with the algorithm for detecting the traffic cone, and I think I found a way to eliminate more of the false positives which we occasionally still get. We seem to occasionally get a false positive detected at points in an images where light is reflected directly into the camera. I have been working on increasing the size, i.e. number of pixels, of the minimum accepted result. These false positives in the image are so small that I can not make out any kind of cone shape or an orange coloration, so I believe enlarging the minimum accepted result will completely eliminate this issue.

Paul Minner -
The beginning of this week was spent getting the repo ready for the code freeze. I made a few small changes to the code and integrated the ECE team's device controller code into our own. I am unable to test if everything works correctly though, because we still don't have a completed rover to test with. Everything unrelated to the physical components of the rover we know works correctly, though. I also reorganized the repo so it is easier to tell what is our final product, and what is extra material. Other than that, we haven't been able to do much this week because all we're waiting on is testing, which can only be completed with a working rover. Supposedly, we should get a working rover next week, which won't leave much time for testing before expo.

Steven Silvers -
Still waiting on hardware to be finished by the ECE and ME teams, hopefully we can begin testing week 6 but at this point there doesn't seem to be enough time to implement and test the full system on hardware before EXPO. We have the code freeze and midterm report coming up, both of which shouldn't be problems.

Zhaolong Wu -
This week we had a meeting with the ME and ECE team, I'm aware that they still have not finished the rover, so we can't perform any testing yet. Within 10 days till the expo, we are just trying to get our parts work and do our best on the expo.

Week 6

Zachary DeVita -
Still no rover. Spent this week completing the middle of the term progress report and presentation. We met with out mentor. We met with our entire team. We keep being told that we should get the rover in the next couple of days, but I don't know how much testing we would even be able to complete at this point. I am currently preparing for expo; at least we can still display some of our programming features at expo.

Steven Silvers -
We are having trouble getting in touch with the ECE team to see how the rover is coming. It looks like we should be getting the rover on the Wednesday before Expo, however we won't have enough time to integrate and test our software in time to demo it at Expo. I believe the plan is to have the ECE team write a small demo routine so that the rover will move around a bit at expo.

Zhaolong Wu -
One week from EXPO, our software is finished and ready to be tested, ECE AND ME team still haven't managed to put the rover together, at this point we are just going to show off our program on a laptop during the EXPO.

Week 7

Zachary DeVita -
We finally got the rover this Wednesday. It was hot glued together and missing the camera so testing was still on hold. After a feeble attempt to get the camera installed on this hot glued mess, an sd card got fried, a wire became loose and the rover ended up back in pieces. Just before expo, on Friday morning, the rover had been re-assembled with a new sd card and the wiring fixed. There was still no camera installed, but at least the mechanical team had something to show. Expo went well; we had five or six people stop and talk to us. We had no industry people come anywhere near our presentation. I think this is due to the fact that we have red tags and were separated from the CS projects.

Paul Minner -
This week, we finally got a working rover on Wednesday, excluding a camera. We worked with the ECE team to try to get the camera working, but unfortunately, there wasn't enough room in the casing for the camera ribbon to fit. In the process of trying to get the camera working, the SD card on the pi was corrupted, so Simon and I worked with the ECE team trying to get our other SD card set up to run. We ended up not getting everything to work, and were unable to put our code on the pi. There was nothing we could do about this, though, because the only software we could demo at expo used the camera, and the ECE team couldn't mount the camera.

Steven Silvers -
The week of Expo, the ECE team delivered the rover to our team Wednesday night with the idea that we would mount the camera and begin testing. Upon receiving the rover, it was found that the camera port on the raspberry PI was completely blocked off, and there was no room in the electronics box for the camera ribbon. We returned the rover to the ECE team, who spent thursday working with the ME team trying to get something setup for the Expo demo. We already had our plan in place to demo our object detection with a laptop and a webcam so we are set for Expo.

Zhaolong Wu -
EXPO Week! Me and Paul worked with the ECE and ME team to trying to put the rover together, the rover was working on Wednesday but when we tried to mount the camera on we fried the SD card. It took us 2 days to make it barely working again, but somehow the rover stopped working again on the EXPO day, bummer. Now after the EXPO we are just trying to finishing this up and get ready for the competition.

Week 8

Zachary DeVita -

If you were to redo the project from Fall term, what would you tell yourself?

Zack... quit writing scripts for your presentations. It will make your presentation preparation take exponentially longer and it will sound exactly like you are reading from a script. Zack... more importantly, make those mechies get to work, and don't let them spend six weeks discussing the design. Also, try to get your hands on some kind of remote control car so that you can get some of the navigation and obstacle detection algorithms tested.

What's the biggest skill you've learned?

Computer Vision/Machine Learning/OpenCV. I can't decide; they're all so related and are such amazing skills. I have never been so inspired or felt so accomplished as I did when I got the cone detection algorithm working. The whole process of researching, designing, implementing, and testing, and without any direction or instruction made the experience truly incredible when I was able to succeed.

What skills do you see yourself using in the future?

I will absolutely be using computer vision, machine learning in my future; likely just for fun since the career opportunities related to computer vision seem to be rare. I also learned some teamwork skills. I have never had a group project anywhere near the size or duration of this so there was definitely some experience and some improvements gained in the area of teamwork and in my communication skills.

What did you like about the project, and what did you not?

I liked the idea of the project. Every aspect of the project, every individual component of the project sounds so interesting and fun. I loved the opportunity for inventiveness in the project, the freedom and creativity that came with the project. This project allowed for creativity and excitement which is something I have not been allowed to have in my other computer science courses. I love that I experienced all of the stages of software development. I like the feeling that I succeeded, on a personal level.

I dislike that our project failed as a whole. I dislike that two of the teams deadlines were the same; I think ours was actually a week before the electrical engineering teams. I did not like the lack of expertise for our project. I was hoping a project like this would get me noticed and help me land that dream job, but now I just have a good response to the question, "What was a time you've dealt with failure." I also dislike confusing, incomplete and dynamic instructions for my assignments which seem to be the norm for these classes.

What did you learn from your teammates?

I learned that people may surprise you with their accountability. I have had few and far between positive experiences working with teams of students in my college experience, but I learned that, with good communication and clear expectations, people can and will get their work done. I guess I learned that communication might be the most critical factor in the success of a team.

If you were the client for this project, would you be satisfied with the work done?

This is a complicated answer. I don't think that I would be especially thrilled that the project never got completed. I would hope that I would recognize that there was no way that the computer science team could have implemented their components for the project when they never had a working rover to work on and test with. I think I would be able to recognize that the success of the project was completely out of the CS team's hands.

If your project were to be continued next year, what do you think needs to be working on?

I think the rover should be redesigned. Instead of having tracks it should utilize the traditional ARLISS competition rover design, with each end of the cansat being a wheel. After landing on the ground and right before it begins moving, it should have retractable spikes which shoot out of the tires. This would give the diameter of the wheel an extra inch which would eliminate the clearance problem. I think the computer science team should also get a chance to implement software on the rover next year. This would make it more likely to work, and give them a chance at the competition.

Paul Minner -

If you were to redo the project from Fall term, what would you tell yourself?

I would try to establish better communication with the other teams. We only met the the ME and ECE team once a week, and that was usually the only time we would communicate about our progress. We also had a group text, but that wasn't used very often, and would often go ignored or unnoticed. This wasn't an issue early on in the project, but as deadlines approached, it was frustrating being unable to contact other teams to check on their progress, or to just hear vague answers. Possibly we could have set up more than one meeting a week later on in the project so teams would have more accountability.

What's the biggest skill you've learned?

I learned a lot about managing projects. There was no team leader on the CS team, but I was the person responsible for integrating each person's tasks into one cohesive program. I realize now there are much easier ways to do this than how we did it, and will do better in the future.

What skills do you see yourself using in the future?

Working with custom hardware is a common practice in many companies, so I could see some hardware expertise coming in handy. Additionally, familiarity with Linux is a valuable skill to have. On the less technical side, experience working in teams with different skill sets and backgrounds is important as well.

What did you like about the project, and what did you not?

I liked that this project tackled a problem in a relatively new field. Self driving vehicles are a fascinating new invention, and getting some insight into how they actually work was very interesting. I didn't like that this project included multiple majors. I think it could have worked better if we were treated as a single team instead of three separate teams. Every team had their own requirements and deadlines to meet, which didn't always work for the other teams. There was no accountability between teams.

What did you learn from your teammates?

I learned to have more faith in my team to complete the work they said they would. While the ME and ECE team didn't complete their part of the project, everybody on the CS team finished what they said they would, which is a welcome change from many group projects.

If you were the client for this project, would you be satisfied with the work done?

I would be disappointed that there is no completed product yet, but I would be satisfied with what the CS team has done. While our code is untested on the rover itself, we have implemented every feature that we said we would, and tested it on our own machines. Completion of the rest of the project is out of our hands.

If your project were to be continued next year, what do you think needs to be working on?

Obviously, the rover itself still needs to be finished. Code wise, all that remains is testing. There will likely be quite a few tweaks that need to be made, considering how many assumptions we needed to make in order to write the code. Other than that, the code is complete.

Steven Silvers -

If you were to redo the project from Fall term, what would you tell yourself?

To better plan out space in the electronics box, we didn't plan a layout and just assumed it would all fit, which resulted in problems.

What's the biggest skill you've learned?

The ability to work with multiple teams from multiple disciplines, and how to stay busy when others are missing deadlines.

What skills do you see yourself using in the future?

Planning requirements for a project and then meeting those requirements, being able to coordinate with large teams.

What did you like about the project, and what did you not?

It was in the aerospace sector, so it was easy to explain to friends and family. Our client didn't really seem to care about our project, which made it difficult for me to care.

What did you learn from your teammates?

There are multiple methods to complete the same task, each with their own trade-offs.

If you were the client for this project, would you be satisfied with the work done?

Taking into consideration that the project is due in September and there is still all summer to work on, yes at this point as a client I would be satisfied.

If your project were to be continued next year, what do you think needs to be working on?

Maybe redesign the rover, look at using sonar or lidar instead of imaging for obstacle avoidance.

Zhaolong Wu -

If you were to redo the project from Fall term, what would you tell yourself?

Finish the physical rover as soon as possible, having some feasible ideas? Then make them happen, not wait everyone to vote on them and spend time for nothing, build a non functional thing and break it is better than just waiting. Project management and workflow is the most critical task.

What's the biggest skill you've learned?

How to collaborate with students from different disciplines. How to work with a project that involves with multiple deadlines. How to communicate with students that have different skills and disciplines.

What skills do you see yourself using in the future?

I think communication between different disciplines with different skill sets is definitely going to be used in the future, just like how product managers do.

What did you like about the project, and what did you not?

It's an interesting project that involves some aerospace and autonomous driving features, which are two most interesting things now. While the project has a pretty fancy name, the process of the project wasn't very ideal, at least for our CS team.

What did you learn from your teammates?

Their approach to different tasks, some of them have really good Linux and git skills which is what I need to learn. Also simply just how good they are.

If you were the client for this project, would you be satisfied with the work done?

I guess so?

If your project were to be continued next year, what do you think needs to be working on?

Getting a fully functional rover finished before CS team get on, most ME people don't understand that an autonomous rover's most critical and time demanding component is the software.

VI. Poster

COLLEGE OF ENGINEERING

Electrical Engineering & Computer Science

THE ARLISS PROJECT

Autonomously navigate to a target destination while using obstacle avoidance.

ARLISS A Rocket Launch for International Student Satellites

The Goal

Working alongside Electrical and Mechanical Engineering capstone teams, our task was to design and build a small rover that fits inside a standard 12 oz. soda can. This rover will be dropped from a rocket at 12,000' AGL, land safely on the ground and drive itself to a predetermined set of GPS coordinates.

As the Computer Science team, our job was to develop the software package for the rover. This included being able to lock on to and drive towards the GPS coordinates, as well as, an obstacle avoidance system that uses a camera to detect and avoid rough terrain or objects such as rocks that are in the path of our rover.

Implementation

We have implemented the following tasks:

- Parachute Deployment
- GPS Navigation
- Obstacle Avoidance
- Getting Unstuck From Obstacles
- Hitting the Finish Pole

In addition, many tasks were implemented utilizing the onboard camera, which we had to create obstacle detection software for

Parachute Deployment

Once the rover is dropped from the rocket, we had to determine at what point to deploy the parachute. We accomplished this by tracking the GPS coordinates height, and deploying the parachute once we dropped below a certain altitude. This way, the wind won't carry us as far as if we deployed the parachute immediately.

GPS Navigation

For the rover's GPS Navigation functions, we are using an algorithm that determines the shortest path between two given GPS coordinates. The GPS will also keep updating the new best route per request from the obstacle avoidance and unstuck from obstacles modules. This means that the GPS function has to work flawlessly with both of these modules to ensure the rover's safety and efficiency. How the rover behaves during it's driving is also critical, so the GPS function will check if the rover is off-course every few seconds and give route compensation if needed.

Obstacle Avoidance

The obstacle avoidance system ensures that our rover is not impeded on its way to the destination. Taking in filtered images from the obstacle detection software, this system does edge detection on the image to find objects in the rovers path, and then decides how to best get around the object. This is done by treating the filtered black and white image as a matrix of pixels, and summing the number of edges to the left, right or in front of the rover and adjusting the direction of the rover to travel where the fewest edges are found.

Getting Unstuck From Obstacles

In case the obstacle avoidance fails, and we end up hitting an obstacle, we've developed an algorithm to help us get unstuck from what we hit. It works by first attempting to back up the rover, and if the rover doesn't move, back up in different directions until it does move. It detects if the rover has moved by checking the GPS coordinates. This algorithm works best if just the rover's path forward is blocked, but it can still easily move backward.

Find and Touch the Finish Pole

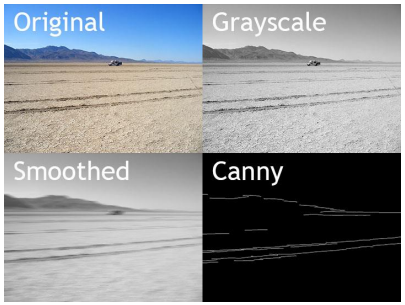
Once the rover get's within the GPS' error range of the finish coordinates, we have to search for the finish pole. This algorithm works by first searching for the finish pole by rotating in place and taking pictures. These pictures are used to detect a traffic cone by our imaging system. Once the cone is detected, the rover is oriented in the direction of the cone, and moves forward, making periodic course corrections along the way.

Computer Vision

Computer vision is used throughout the duration of the rover's expedition, but there are two separate and distinct functions which are being performed. During the stage of the expedition where the rover relies on GPS to direct it towards the target set of coordinates, computer vision will be utilized in the obstacle avoidance system. When the rover is approximately 8 meters away from the target, and the objective of the rover has switched from being directed by GPS to searching for the pole, computer vision will be used to locate the traffic cone at the base of the pole.

To detect obstacles in the path of the rover, each frame is run through a series of filters which are primarily intended to eliminate noise and to detect the edges of the obstacles.

1. Convert Original image to Grayscale
2. Smooth (e.g. blur) Grayscale image
3. Morphological Opening of image
4. Canny Edge Detection



To detect the pole at the end of the rover's expedition, our team has developed a complex system which utilizes machine learning and image processing. For the machine learning, our team trained a Haar Classifier which can be used to detect a traffic cone based on its features and geometry. Though effective at only detecting a traffic cone the majority of the time, machine learning classifiers are prone to returning false positives. The way we have avoided this is by verifying our results by checking the color.

Several points from the result, i.e. pixels, are converted to their HSV (Hue, Saturation & Value) equivalent, and are compared to values which would be typically observed on an orange traffic cone. Without the HSV values being confirmed, the image is rejected. This results in a higher rate of false negatives, but ensures accurate results.

Meet the Team

Team Members

- Zachary DeVita devitaz@oregonstate.edu
- Zhaolong Wu wuzha@oregonstate.edu
- Paul Minner minnerp@oregonstate.edu
- Steven Silvers silverss@oregonstate.edu

Our Client

Dr. Nancy Squires

Senior Instructor of Mechanical Engineering at Oregon State University

squires@engr.orst.edu

Sponsorship

This project was made possible through funding provided by Oregon State University AIAA. To find out more about AIAA, follow the QR code or visit http://groups.engr.oregonstate.edu/aiaa/home

Oregon State UNIVERSITY

VII. Project Documentation

A. *Structure*

We structured our software by dividing it up into its main tasks, which were Parachute Deployment, GPS Navigation, Obstacle Avoidance, Getting Unstuck From Obstacles, and Hitting the Finish Pole. Additionally, we included placeholders for Motor Control Functions and GPS Location Functions, which were implemented by the ECE team. Each task was implemented as its own class, which would be given control over the rover's sensors and motors also implemented as classes. Therefore, we could merge functionality with the ECE team's motor and sensor control functions when they were finished. Each task would run when it was it's time to run specified in the main function.

B. *Implementation*

1) (**Parachute Deployment**) Once the rover is dropped from the rocket, we had to determine at what point to deploy the parachute. We accomplished this by tracking the GPS coordinates height, and deploying the parachute once we dropped below a certain altitude. This way, the wind wont carry us as far as if we deployed the parachute immediately.

2) (**GPS Navigation**) For the rovers GPS Navigation functions, we are using an algorithm that determines the shortest path between two given GPS coordinates. We take the rover's current position and the finish coordinates, and make a vector. Then, we move forward and take our new coordinates so we have another vector between our old coordinates and current coordinates. From the two vectors, we compute the angle we need to turn to be on course towards the finish.

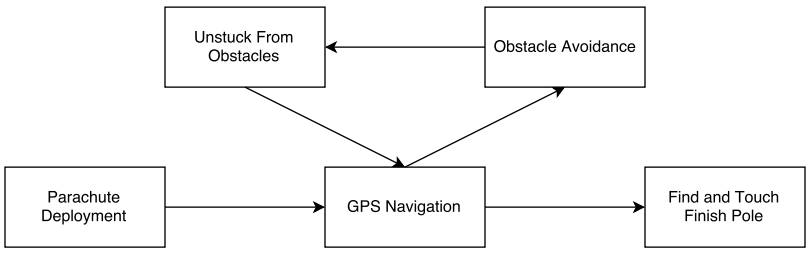
3) (**Obstacle Avoidance**) The obstacle avoidance system ensures that our rover is not impeded on its way to the destination. Taking in filtered images from the obstacle detection software, this system does edge detection on the image to find objects in the rovers path, and then decides how to best get around the object. This is done by treating the filtered black and white image as a matrix of pixels, and summing the number of edges to the left, right or in front of the rover and adjusting the direction of the rover to travel where the fewest edges are found.

4) (**Getting Unstuck From Obstacles**) This system runs if it has detected the rover is stuck. We determine the rover is stuck by checking it's GPS coordinates, so if it hasn't moved after moving for a certain amount of time, this system is run. It works by first attempting to back up the rover, and if the rover doesnt move, back up in different directions until it does move. It detects if the rover has moved by checking the GPS coordinates.

5) (**Find and Touch the Finish Pole**) Once the rover gets within the GPS error range of the finish coordinates, we have to search for the finish pole. This algorithm works by first searching for the finish pole by rotating in place and taking pictures. These pictures are used to detect a traffic cone by our imaging system. Once the cone is detected, the rover is oriented in the direction of the cone, and moves forward, making periodic course corrections along the way.

6) (**Putting it all Together**) Each task is executed at least once in the main program. Many tasks are executed multiple times. This diagram demonstrates the flow of our program.

Fig. 1: A flow diagram of our program's main tasks.



The first thing the program starts is the parachute deployment module. Once on the ground, the GPS Navigation Module starts, which begins moving the rover in the direction of the finish. Next, the Obstacle Avoidance Module

runs to navigate around any obstacles. If the rover detects that it has gotten stuck, the Unstuck from Obstacles module will run next. Then, the GPS Navigation module will run again to correct course. These three modules will continue to run in a loop until the rover is within the GPS error range of the finish coordinates, the control will switch to the Find and Touch the Finish Pole module, which will complete the rover's task and end the program.

C. *Hardware Requirements*

This software is meant to run on a Raspberry Pi Zero running Linux fitted with a custom PCB board to house the GPS sensor and control the various motors and servos attached to it. The only hardware this software will run on correctly is the custom built Rover designed by the ECE and ME teams. A version of the software is available which doesn't include the software to control the motors and sensors, though, which can be run on any Linux system with a camera.

D. *Installation*

1) ***Prerequisites*** In order to install the testing version of this software, which doesn't include code to obtain data from the GPS or move the various motors and servos attached to the rover, a Linux system with OpenCV 2.4.9 is required. Additionally, to run on the actual hardware, wiringPi must be installed as well, which is a PIN based GPIO access library for the Raspberry Pi.

2) ***Compile and Run*** To compile, simply run the makefile located in the final directory. To run the test version on any machine type make test with the DEBUG variable set to 1 in main.h. To run on the actual satellite hardware type make final with the DEBUG variable set to 0 (by default set to test version with DEBUG = 1). When run on the actual hardware, there will be no way to access the operating system to run the program, so use Crontab to run the program automatically after booting.

VIII. Resources

A. *Machine Learning*

Machine learning was a difficult was a difficult aspect of this project. I would not have been able to accomplish what I did without the assistance of Professor Bill Smart and his graduate students, as well as, without a few notable online resources. Bill Smart put me in contact with several of his graduate students who were able to point me in the right direction with various resources, and they were able to find a flaw in my settings which allowed me to eventually complete the training of my Haar classifier.

Bill Smart has his PhD in Computer Science and a Masters in Intelligent Robotics. His research is focused in the areas of robotics and machine learning so he one of the most valuable resources in the world on the particular topic.

Bill Smart
Associate Professor of Mechanical Engineering
Office: 316 Graf Hall
Phone: (541)905-2553 Email: bill.smart@oregonstate.edu

If you are new to machine learning, or new to training a Haar classifier then you should surely download the haartraining toolkit from the following website. This kit comes with some instructions on training your haar classifier, but I found it most useful for the opencv-haar-cascade-positive-image-builder tool. If you only have time or the resources to come up with a limited number of images which have the object you are trying to detect in them, then this tool will help create additional images. For a strong classifier you will need hundreds, maybe thousands of images which have the object in them. What this tool will do is it will turn each image into many images by rotating it along different axis and changing the lighting.

http://www.tectute.com/2011/06/opencv-haartraining.html

Here are a couple websites which have useful tutorials:

http://memememememememe.me/post/training-haar-cascades/
http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html

This next link is to an amazing video tutorial which I followed almost directly:

https://www.youtube.com/watch?v=KFC6jxBKtBQ

All computer vision software which I wrote uses the OpenCV library. OpenCV stands for Open Source Computer Vision. The OpenCV project has amazing documentation for everything you can do with their library. Below is the link to the OpenCV Cascade Classifier page which has the program which I used as my driver to test my Haar classifer. There are many resources here when working on computer vision.

http://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html#cascade-classifier

IX. What We Learned

A. ***Zachary DeVita***

B. ***Paul Minner***

C. ***Steven Silvers***

D. ***Zhaolong Wu***

X. Appendix I: Essential Code

Computer vision provides an integral component in the success of the rover on its expedition. There are two separate and distinct functions which are performed using computer vision. While the rover is being directed by the GPS, computer vision will be used to detect obstacles in the path of the rover. This is accomplished by using a series of filters which are primarily intended to eliminate noise and to detect the edges of the obstacles. The other function of the computer vision is to identify the orange traffic cone. Our team has developed a complex system which utilizes machine learning and image processing to carry out this task.

A. *Obstacle Avoidance*

The obstacle detection algorithm our team has implemented requires that each frame recorded by the camera is run through a series of filters which are intended to eliminate noise in the image and to detect the edges of the obstacles in the rover's path. Below is the algorithm which has been developed.

```
/** @function checkObstacle */
void Obstacle::checkObstacle()
{
  Mat frame, blurred, gray, canny;

  if (!DEBUG) {
    //Capture frame from camera
    VideoCapture capture(0);
    capture >> frame;
  }
  cvtColor(frame, gray, CV_BGR2GRAY);
  blur(gray, blurred, Size(40, 4));

  //  Morphological opening (remove small objects from the foreground)
  erode(blurred, blurred, getStructuringElement(MORPH_RECT, Size(5, 5)));
  dilate(blurred, blurred, getStructuringElement(MORPH_RECT, Size(5, 5)));
  dilate(blurred, blurred, getStructuringElement(MORPH_RECT, Size(5, 5)));

  Canny(blurred, canny, 50, 120, 3);

  int Image_Array[Width][Height];
  for(int i=0; i<canny.rows; i++)
    {
      for(int j=0; j<canny.cols; j++)
        {
          Vec3b color = canny.at<Vec3b>(Point(j,i));
          if(color.val[0] >= 25 && color.val[1] >= 25 && color.val[2] >= 25) {
            Image_Array[j][i] = 1;
          }
          else {
            Image_Array[j][i] = 0;
          }
        }
    }
  //start of obstacle avoidance section
  Analyze(Image_Array, Width, Height);
}
```

As for an explanation, the VideoCapture constructor opens the video capture device and records a single frame from the camera. This data gets stored in a Mat (i.e. matrix) object each time this function is called. The cvtColor() function converts the frame to its equivalent in grayscale, and this gets stored in a separate Mat object. The blur() function does exactly what it sounds like; it blurs the frame using a specified parameter which has been determined through testing. Next, the now blurred, grayscale frame is put through a morphological transformation. The final filter used is the Canny Edge detection algorithm. The Canny algorithm is an extremely complex algorithm that I could write a book explaining so I will explain what it does opposed to how it works. The Canny algorithm is used to filter out any noise left in the image. The algorithm finds all edges in the image and uses the smallest value between thresholds for edge linking. The largest values are used to find initial segments of strong edges [**?**]. The set of nested loops is used to store the results from the Canny Edge detection algorithm into a double array containing 1s and 0s depending on if the individual pixel represents an edge of an object in the frame or not.

B. Traffic Cone Detection

The algorithm implemented for detecting the traffic cone at the base of the pole is a complex system which utilizes machine learning, as well as, some image processing filters. For the machine learning, our team has trained a Haar Classifier which can be used to detect a traffic cone based on its features and geometry. Though effective at only detecting a traffic cone the majority of the time, machine learning classifiers are prone to returning false positives. The way we have avoided this is by verifying the results by confirming that the color of the result is orange.

The way the color is checked is that several points from the potential result, i.e. pixels, are converted to their HSV (Hue, Saturation & Value) equivalent, and are then compared with values which would be typically observed on an orange traffic cone. If the HSV values cannot be confirmed, then the image is rejected. This results in a higher rate of false negatives, but it also ensures that the results are accurate. Below is the algorithm which has been developed.

```
/** @function detectAndDisplay */
int Finish::detectAndDisplay(Mat frame) {
  std::vector<Rect> cones;
  Mat gray;
  int quintant = 0;

  cvtColor(frame, gray, COLOR_BGR2GRAY);
  equalizeHist(gray, gray);

  //-- Detect cones
  cone_cascade.detectMultiScale(gray, cones, 1.1, 2, 0 | CASCADE_SCALE_IMAGE, Size(30, 30));

  if (cones.size() != 1)
    return 0;

  if (!isOrange(frame, cones[0])) {
    return 0;
  }

  rectangle(frame, Point(cones[0].x, cones[0].y), Point(cones[0].x + cones[0].width, cones
    [0].y + cones[0].height), Scalar(255, 255, 255), 2, 8);
  Point target(cones[0].x + cones[0].width / 2, cones[0].y + cones[0].height * 0.8);
  circle(frame, target, 3, Scalar(101, 255, 0), -1, 8);

  imshow("FRAME", frame);

  return selectQuintant(frame.size().width, target.x);
}
```

As for an explanation, the above function takes a video frame from the camera sensor and converts it to grayscale. The grayscale frame has its histogram equalized which essentially means that the brightness of the image is normalized and the contrast of the image is increased. The image is then scanned using the machine learning algorithm for traffic cones. If traffic cones are found, two catty-corner points are stored as a rectangle object in a vector. Then the object is sent to the isOrange() function to confirm the color is orange. Lines 20-24 in the function are purely used for displaying the result. This is useful for testing and showing the algorithm, but these lines will not be implemented in the rover. The final line of code simply calls a function which determines which of the five segments the center of the cone is located in.

```
/** @function isOrange */
bool Finish::isOrange(Mat original, Rect cone) {
  Mat HSV;
  int xVal = cone.x + cone.width / 2;
  int yVal[3] = {
    cone.y + cone.height*0.8,
    cone.y + cone.height*0.2,
    cone.y + cone.height*0.5
  };

  for (int i = 0; i < sizeof(yVal)/sizeof(*yVal); ++i) {
    Mat RGB = original(Rect(xVal, yVal[i], 1, 1));
    cvtColor(RGB, HSV, CV_BGR2HSV);
    Vec3b hsv = HSV.at<Vec3b>(0, 0);
    int H = hsv.val[0];   //  Hue
    int S = hsv.val[1];   //  Saturation
    int V = hsv.val[2];   //  Value

    if (H < 180 && S > 39 && V > 234)
      return true;
  }
  return false;
}
```

The above algorithm takes three points from the potential traffic cone result in the image. The three points are taken from the horizontal center of the rectangle object, and with y-values from near the top, another near the bottom, and another from the center of the object. The HSV values are taken from each point, and, if any one of the points turns out to be orange, then the function returns true.

XI. **Appendix II:**

XII. References