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.

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 substantial 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 2017.

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 our team has chosen to use a CMOS imaging sensor. The CMOS imaging sensor will be used to identify obstacles in the satellites path, and we will estimate distance based on location of obstacles relative to the bottom of the frame. Since the camera is fixed, objects on flat ground will appear in approximately the same location in any given frame. There will be an accelerometer onboard so pitch can be taken into consideration when determining distances on an incline. I believe this will be an effective and accurate method of detecting static obstacles in Black Rock Desert, Nevada. The camera sensor is also cheap, effective for the climate, and can be implemented to minimize 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 Zero: The Raspberry Pi Zero is a well documented pre-built board that could be easily adapted to our project. It is small enough to easily fit within our CanSat while also leaving space for other electrical components.[12] Another advantage of the Pi is the built in camera port, which will be needed for the obstacle avoidance system.
c) Custom Board: A custom built board would be the best option if done correctly. The dimensions could be developed with the CanSat 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 Raspberry Pi Zero 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 Arduino, while an excellent board for embedded systems, may not provided the needed processing power for our system.

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/