AutoVot: A Vehicle to Vehicle Communication System for Autonomous Driving

Authors: Joel Anyanti, Fausto Leyva, Jeffrey Tsaw
Electrical and Computer Engineering, Carnegie Mellon University

Abstract—**We cannot deny that the future of transportation looks to be dominated by autonomous vehicles. Given this trend we look to solutions in this area that may enhance such technologies. Vehicle to Vehicle communication (v2v) may provide additional safety and increased network efficiency in autonomous vehicles systems. In this project we intend to demonstrate value behind v2v by delivering a convoy system cable of navigating a course while avoiding all obstacles.**

Index Terms—**Autonomous Driving, Object Detection, Vehicle to Vehicle Communication, Jetson Nano, Convoy, AprilTags.**

1 INTRODUCTION

The transportation industry has recently turned to autonomous driving solutions to perform transportation tasks previously exclusive to human drivers. Indeed as the technology that enables self-driving becomes more reliable and commercially available, an autonomous future looks to be inevitable. Autonomous driving technology in its current state relies on individual vehicles to make decisions based on self provided sensory input and perception. As impressive as the technology may become, we recognize that coordination between vehicles would provide additional safety benefits and increase the overall efficiency of transportation networks. To demonstrate this claim, we aim to develop an autonomous driving system which leverages vehicle to vehicle communication to perform a coordinated task.

For our development we have chosen to focus on implementing a convoy system between a lead and follow vehicle. The 'lead' vehicle will be fully equipped with image sensing to allow for object detection and path planning. By contrast, the 'trail' vehicle will be blind and rely solely on information from the 'lead' to navigate the course. The two vehicles will be tasked with driving from one end of a track to another while avoiding any obstacles in their path. With this task we intend to highlight the benefits of communication between vehicles and ultimately hope to achieve zero collisions in our verification runs.

2 DESIGN REQUIREMENTS

In order to best accommodate our circumstances we considered the physical attributes of the vehicles to parameterize our testing course. We recognize that at the scale we are developing under, attention to these details are critical to defining thoughtful requirements. In particular we relied on speed estimates for the vehicles to determine the length of the straight away for our course. To estimate the speed of our vehicles we needed to factor in both the payload weight and the RPM of the motors.

Standard DC motors for Arduino based RC vehicles are rated at $200 \pm 10\%$ RPM @ 6v. The wheel diameter for the corresponding motors are 65mm. With the following equations we are able to estimate V_{max} for the vehicles:

$$\omega = 2\pi f \tag{1}$$

$$v = r\omega \tag{2}$$

The resulting velocity we obtain from these equation is $0.68m/s \pm 10\%$ (See Appendix A) but we must also factor in the added weight of the vehicle components into the overall speed of the vehicle. The required components (camera, sensors, motor shield, Arduino, Jetson Nano, power banks) for the fully equipped lead vehicle are estimated to be around 600g (5.88N) of weight. We project that this will cost around 15% slowdown from the theoretical maximum of the motors. This leaves us with the following: $V_{max} = 0.58m/s \pm 10\%$.

Given these calculations we assume the top speed of the vehicles to be a generous 0.5m/s with the lead car. This projection allows us reasonable margin of error as the weight projects are overestimates and the 'trail' vehicles will have lighter payloads.

In determining the final length of the course, we wanted to maintain a reasonable length for adequate testing. A length that is too long would likely lead to reproducibility errors and difficulty in collecting data. By contrast, A test too short may be unimpressive and less able to illustrate the value proposition. Acknowledging this, we targeted a test run of 1 minute total time for completion of the course. Given that a vehicle moving at 0.5m/s will reach about 30m maximum distance forgoing acceleration we opted for a 20m straight away course with a width of 1.22m. These values are based on the standard dimension for a track and field race track. A diagram of this track can be seen below in Figure 1.

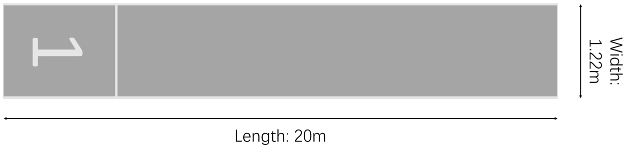


Figure 1: Diagram of task verification track

This length would allow for about 20 seconds of margin time for avoidance and navigation given our expected vehicle velocity. We consider this to be a reasonable amount of time given the requirement of successfully navigating two vehicles to the end of the track. The decision to use a straight away course with no turns is motivated by the fact that such a system would rely much less on localization/-global positioning which would cause greater difficulty in implementation.

Critically we must also acknowledge the vehicle avoidance portion of the development as a driving factor for our requirements. In essence we need to know what the minimum object detection distance is our solution needs to achieve to successfully avoid obstacles in its path. For this we consider the following three sub-distances: latency distance, stopping distance and buffer distance. Latency distance is the distance a vehicle travels over the time required to make one pass through the software pipeline. This includes object detection, path planning and communication. Stopping distance is the distance a vehicle will travel after a stop command before reaching a complete stop. Buffer distance is the distance left between the obstacle and the vehicle after the vehicle has stopped. This breakdown is clearly illustrated in Figure 2 below.

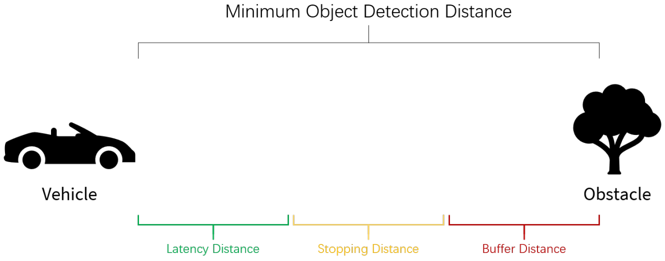


Figure 2: Diagram of object detection distance breakdown

In order to obtain the stopping distance we estimated the distance the vehicle travels over from its max speed to a full stop. Using the following physics equations in relation the our vehicle parameters,

$$v^2 = v_0^2 + 2a\Delta x \quad (3)$$

$$F = ma \quad (4)$$

The results from this calculation (See Appendix A) result in a stopping distance of 0.03m as we observe in Table 1.We note that since the buffer distance is a fixed parameter set by the group, there is no corresponding calculation. The numbers provided in Table 1 for the compute task latencies are rough estimates based on research into our methods of choice. We consider these estimates to be generous as they include slack for anticipated worst case conditions. With this we are able to calculate the latency distance using the following equation

$$x_{lat} = v_{max}(t_{det} + t_{plan} + t_{comm}) \quad (5)$$

This leaves us with a latency distance of 0.125m. Thus the overall minimum object detection distance we must target is 0.205m considering all of the sub-distances.

Vehicle Maximum Speed	0.5m/s
Object Detection Latency	100ms
Path Planning Latency	10ms
Communication Latency	100 ±40ms
Latency Distance	0.125m
Stopping Distance	0.03m
Buffer Distance	0.05m
Minimum Object Detection Distance	0.205m
Vehicle Length	0.2794m
Vehicle Width	0.1778m
Track Length	20m
Track Width	1.22m

Table 1: Summary of required metrics

For the autonomous navigation aspect of our project, we need he lead care to be able to detect static obstacles in the path and make planning decisions to successfully maneuver through the obstacles. To accomplish this, we must require that the convoy system maintains 0 collisions through the track, as no crash mitigation will be implemented. Given the track distance and the buffer, stopping, and latency distance, we calculated an object detection latency of 100ms. More specifically, this involves having the on-board stereo camera relay image information to the compute node, which applies an object detection algorithm once every 100ms. This is equivalent to developing an object detection algorithm that is capable of performing at 10fps, assuming negligible latency (i.e a camera latency much less than 100ms) between the stereo camera and the compute node.

Regarding the nature of the object detection, rather than implementing an object classification algorithm, we aim to overlay the image from the stereo camera with bounding boxes surrounding the object. This would give a smooth estimate as to where the obstacles in the path are, and thus make planning decisions that produced the least risk of crashing into obstacles. Given the combined stopping and buffer distances is 0.08m, we require any obstacle to be detected within 0.2m in order to maximize the avoidance probability without imposing too strict requirements on the vehicle hardware. Assuming a frequency of object detection performed at 10fps, and a vehicle speed of 0.5m/s, this would give an average 2.4 frames before the vehicle reaches the buffer and stopping distance and jeopardizes colliding with the obstacle. For this reason, we require our object detection algorithm to be able to perform at 90% precision and 95% recall, in order to minimize the probability of missed detections and subsequent collisions. With 95% recall, the probability of not detecting the object before the vehicle reaches the danger area reduces to 0.25%. Notice that we prioritize recall, since we want to minimize false negatives more than false positives. In the worst case, this would translate into having our object detection detect phantom objects on top of actual obstacles, which would not be a problem assuming our planning algorithm is sufficient.

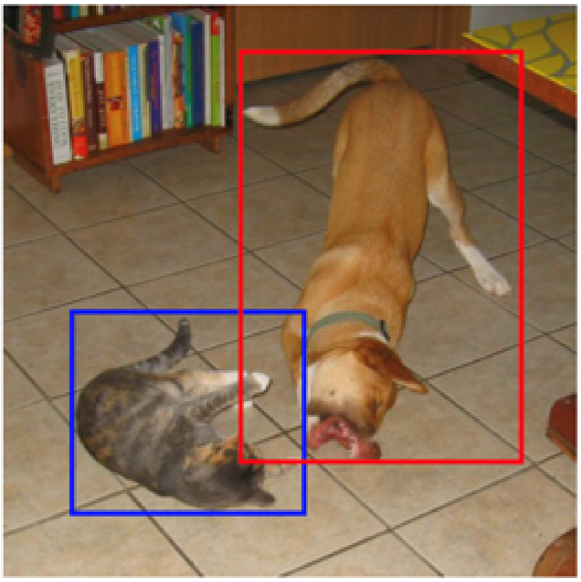


Figure 3: Example bounding box from object detection algorithm

For planning, we simply need an algorithm capable of making navigational decisions based on the object detection algorithm's output and the current camera feed. For this to be successful, we require a latency of 10ms, to allow for the vehicle to dynamically adjust and make smooth turns to navigate around obstacles. The planning stack will be located on the onboard computation node, and communication between the compute node and the motors will be done via an onboard Arduino for simplicity. Using an Arduino as a motor controller would allow the planning stack to output target angles for the wheels to turn rather than PWM signals. This gives greater flexibility to the development of the planning algorithm.

Since we have a lead car equipped with the sensory equipment (camera, distance-tracking sensors) to help it maneuver through the track, we need the lead car to be able to effectively and accurately communicate with the rest of the convoy. We are going to be using Bluetooth as our mode of communication across vehicles which must be reliable and relatively low-latency.

One important assumption we are making is that the Bluetooth conditions are ideal for communication; meaning our track will be free from huge obstructions between the transmission of Bluetooth signals. We are not expecting the latency of Bluetooth messages to exceed 100ms. Upon our research we found that the ideal Bluetooth latency is around 34ms, but can range up to 100-300ms, so we are taking the lower bound since our cars will be in close proximity with a clear path between them. Communication will consist of relaying localization information (map updates) perceived by the lead car, out to the trailing cars. We are expecting the Bluetooth connection to be reliable enough for the lead vehicle to use a broadcasting protocol to accurately send updates on the position of the other vehicles; since we will be sending messages frequently, we expect that a lost packet will not have a significant effect on the vehicles.

3 ARCHITECTURE OVERVIEW

The architecture of the Autovot system consists of 4 major subsystems, vehicle mechanics, object detection, localization/path planning and communication. These subsystems are clearly broken down in the system diagram which can be found illustrated in Figure 10 and Figure 11 in the Appendix (see appendix B). Worth noting regarding the architecture is the two different classes of vehicles that we intend to implement. The key difference between the two designs is that the lead vehicle will be equipped with a slightly more powerful Jetson Nano unit as well as a camera for perception. The trail vehicles will run identical setup with the exception of the scaled down 2GB version of the Jetson Nano as well as not having any camera unit at all. The reason we opted for the less powerful Nano for the trail cars is due to the fact that there will be no image processing performed by these vehicles.

3.1 Vehicle Mechanics

The diagram outlining the overall system for Autovot is based on the lead vehicle to fully cover the extent of the development. For simplicity we can assume that the 'follow' vehicles are identical in regards to the core mechanical systems. The core driver for the mechanical system is the Arduino Uno R3 board which will house the logic for controlling the motor systems. To power the Arduino we will leverage the USB connection between the Arduino and Jetson to supply the required 5v. This connection will also be used to relay commands to steer the vehicle. Our design requires the use of 12 GPIO pins (4 PWM enabled) to establish connections between the LN298 H Bridge Motor Controllers. Each controller drives two DC motors to enable the steering of the vehicle. Notably we specify the use of a 9 V battery to deliver power to the controllers which in turn power the drivers. A simple diagram of this architecture can be seen in Figure 7 below. The LN298 modules experience a 2 V voltage drop which leads to about 7 V terminal voltage to power the motors. As the DC motors are rated for 4-6 V this will satisfy the voltage requirement.

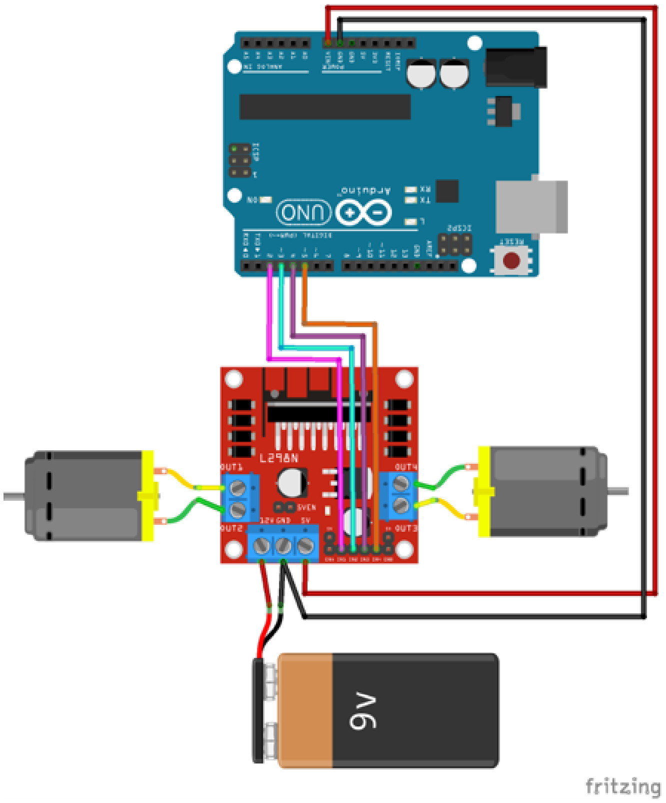


Figure 4: Diagram of the connection between the L298N H Driver and Arduino Board.

3.2 Object Detection

Aboard the lead car will be an Intel RealSense Depth camera that will relay RGB-D information to the Jetson Nano also located on the lead car. The camera will be positioned facing forward such that it can see the obstacles that are on the track. The raw image input will be processed using OpenCV to extract the depth information using the stereo function in OpenCV. This image will then be fed through SSD, or MobileNet V2 to extract the bounding boxes around any objects in the image. The advantage of these networks is the high frequency that they can be performed at, in particular, MobileNet is a single shot detector designed to be applied to embedded computer vision applications such as ours. From here, the resulting image is overlaid with the depth image from the camera and fed to the planning stack. Furthermore, to account for processing issues given the input image size, we propose downsampling the image to increase processing speed. With the requirement of object detection at 0.2m, downsampling would not reduce the resolution of the image as to prevent obstacle detection. Rather than retraining MobileNet to work for our obstacles, if out-of-the-box MobileNet fails to perform well on our obstacles, we will place images from common objects in the COCO dataset, of which MobileNet was trained on, onto our obstacles to boost performance.

3.3 Path Planning

During the planning stage, an angle at which to turn the wheels is chosen in order to avoid the obstacles. Based on the depth information and the edge of the bounding box, calculate an angle to turn the car. Finally we will limit the angles to values between 0 and 80 degrees to ensure the car remains moving forward. 2 angles are calculated based on the left and right edge of the largest bounding box, and the smaller of the two angles is chosen as the angle to turn the wheels to. In order to prevent making unnecessarily wide turns around the obstacles, we limit initial angle planning decisions to the object detection range (0.2m) to the buffer and stopping distance (0.08m), and only allow for small adjustments (0-15 degrees) after the initial angle at which to turn the wheels is determined.

3.4 Localization

In order to allow the convoy to navigate together, localization for the cars will be done using an IMU that tracks accelerometer and gyroscope information that can be passed between the lead and following cars in order to determine relative position. However, we also noticed that IMUs are typically inaccurate after more than 10m. To accommodate for the fact that our track will be 20m long, we plan to implement a system that utilizes April Tagging for localization. The idea is to leverage the fact that we know the parameters of the vehicle track beforehand to provide the vehicles with a software based map that is tethered to the physical implementation of the track. This will enable us to discretize the 2D software map by breaking up the track into a grid formation. Using AprilTags, we can mark checkpoints on the physical track which have direct correspondence to features that exist in the software map. The lead vehicle will be able to compare the distance from a detected checkpoint to itself. In doing so we create relative positioning landmarks to aid in localizing the lead vehicle. This strategy combined with data from the IMU should allow for a reasonable solution for localizing the lead vehicle. Critically the locations of the obstacles will not be present in the software map provided to the vehicles at the beginning of the course. Therefore, it is the job of the lead vehicle to broadcast the location of detected obstacles as communicated map updates to allow the following vehicles to adequately navigate their physical environment. Admittedly, we anticipate difficulty in getting highly accurate localization in the following vehicles. However, we note that for the IMU tracing we expect ideal conditions for the track (no tire slipping due to inadequate friction). Furthermore, we will attempt to enhance the localization system with open loop odometry on the wheels of the vehicle. In essence, we will gather empirical data from vehicle tests to allow for estimation of total displacement as a function of the vehicle commands over time. Accordingly we will pay special attention to this portion of the development as we believe this will be the most challenging portion of the project.

3.5 Vehicle-to-Vehicle Communication

For the vehicle to vehicle (v2v) communication, we will have the lead car act as the master node within a piconet. In the piconet architecture, the lead node can maintain a stable connection with up to 7 other trailing nodes (maximum of 2 trailing nodes required for our application). With our piconet network setup, we plan on directing the convoy by frequently sending messages regarding new obstacles that the lead car detects. Our target message frequency is between 2-4Hz for this application. We chose this range because the car will travel a maximum of 0.25m given its maximum speed. We believe that any more than this distance and we may have serious issues avoiding obstacles. With our compute latency estimated at 250ms we anticipate that this frequency range will be achievable.

The core idea for the communication is to send messages that include map updates to the trailing vehicles. The initial software map will mirror the physically implemented track; however, will not contain obstacle locations. Since the software map will be implemented within a grid formation, the broadcasted messages will be locations within the grid to mark as obstacle locations. This will allow the trailing vehicles to perform local decisions about how to navigate the course. Furthermore, since these updates will be of the form of a coordinate and some signaling parameters, we anticipate that the message size will be much smaller than 100Kb. Therefore, with a transmission speed of 1Mb/s we are confident that we will be able to meet the 150ms communication latency requirements.

4 DESIGN TRADE STUDIES

4.1 Vehicle Mechanics

4.1.1 RC Vehicle

One of the most critical components of this development is the vehicle platform on which to build on. Perhaps the more obvious solution would be to retrofit an existing RC car to fit the requirements of this project; However, as we found out this would prove to be a more challenging solution. An off the shelf RC vehicle would provide us with a reliable platform but we must also acknowledge the setbacks of this option. Namely, it would be difficult to integrate such a system easily with the rest of our components and this would also eat up the lion's share of the budget. Traditional RC car systems rely on radio technologies (typically on a 2.4 Ghz band). This means that we would have to reconfigure the radio system to talk to the compute systems in order to allow for navigation of the vehicle. Furthermore, it would be difficult to mount various components such as sensors and cameras on a traditional RC vehicle system. For this reason we opted for an in house solution to provide this requirement. Developing our system from the ground up meant we had control of various parameters such as vehicle speed, size, power draw, weight and interoperability. Indeed, the ability to directly define how the vehicle would communicate with the rest of the vehicle systems was the most convincing advantage that a custom solution provided. Nevertheless, we cannot ignore some of the disadvantages to a custom solution. Most notably the custom solution leads to a loss of development time due to design and fabrication of the vehicles. Above all we recognized the need for flexibility in such a project and this was a solution that would provide us with that.

4.1.2 RC Controls

Given the choice of a custom vehicle solution we would be required to develop a method for controlling the RC vehicle through the compute systems. As the main compute board for this project is the Nvidia Jetson Nano, this was a natural candidate for developing the control system on. To accommodate the requirements of the project, the Jetson would need to be able to drive 4 total PWM signals from its GPIO pins. This is the required number of pins for driving 2 LN298 H motor controllers. Since only 3 PWM pins are available by default, the solution would require re-configuring the firmware of the board to activate another pin. This domain was mostly unfamiliar to the group and would be difficult to trust without having a Jetson on hand to validate the solution. Another solution to this issue would be to purchase a specialized PCB to handle the PWM signals. Similarly, this solution was less than optimal since the information we found regarding the boards left us unsure about getting the desired result. For this reason, we opted to go with the more familiar Arduino system to serve as the control system for the vehicle. We felt much more comfortable in this decision since RC Arduino cars are quite common projects and the group has had previous experience working in that domain. Admittedly the Arduino solution costs us in increased latency from command to execution; However, we are confident that this latency should not pose a large issue in the progression of the project. Another disadvantage is the additional albeit small weight and power draw from this solution. This setback was offset by the ease of development provided by the Arduino platform. In addition, the Arduino solution did present itself with the additional benefit of modularity. In particular, separating the control system from the main compute system alleviates potential issues that may arise from integration and synchronization of the control interface with the rest of the compute laden tasks. Ultimately, the decision favored the solution which we were most familiar with and could develop the most seamlessly.

4.1.3 Steering System

Another artifact of developing a custom vehicle is the implementation of the steering system. Steering systems play an important role in the navigation of the vehicle since turn radius determines the distance at which an object can be avoided at. The most commonly used system for steering is known as the Ackerman Steering System. This approach utilizes a steering pivot to direct the foremost wheel around a central turning radius. The back wheels remain in a stationary straight direction at all times. The main advantage with this system is the ability to independently control the speed and direction of the vehicle. Since the steering follows a specified angle, this solution boasts an elegant way to pre-calcultate the trajectory of the vehicle. This mechanism also additionally reduces the effects of tire slipping while driving. It is worth noting however the complexity of such a design in a project of this scope. Indeed, using this method would require an axle structure that allows for pivoting the front wheel. Given this fact, we opted to go with an approach that better suited the timeline of our project. Our solution of choice was to use what is known as Differential Steering to guide the direction of our vehicle. This design relies on the relative difference between wheel speeds to produce a turn. The vehicle will experience a turn in the direction of the wheels which have a lower angular velocity. This solution presents a relatively more simple approach to developing the steering system. As our vehicles will use 4 DC motors in a 4WD system, we just have to tune the motor speeds to produce the turns that we desire. Admittedly, the Differential Steering technique produces a more complex relationship between wheel speed and direction of the vehicle. In fact, this relationship has been explored in greater depths [**6**] to allow for more accurate calculations For this reason we may have to introduce adjustments motivated by empirical data to allow for better calculation of vehicle trajectory with this approach. In all, we trade-off simplicity of implementation for ease of trajectory calculation with this decision.

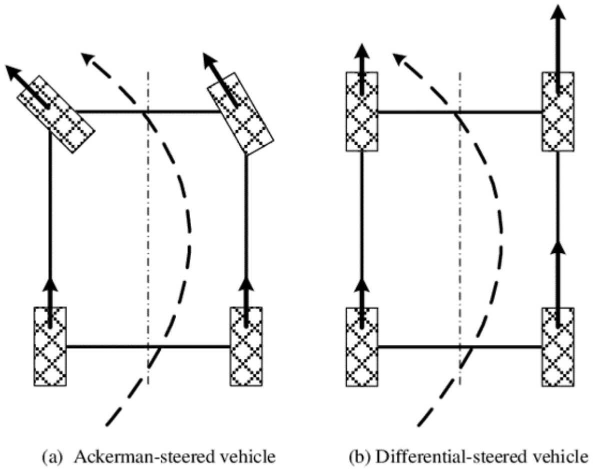

(a) Ackerman-steered vehicle
(b) Differential-steered vehicle

Figure 5: Diagram of Ackerman vs. Differential Steering Designs.

Trade studies of sub-systems can also be included in this section. You should use sections with the `subsection` command to split up this section either in

4.1.4 Vehicle Track

The ultimate goal for this project is to develop a vehicle convoy system that is capable of completing a set course without any collisions. It is therefore crucial for the vehicle track to best reflect the core of this project. Clearly road structures can be much more complex than straight lines and clean angles; However, the focus of the project was less about vehicle mobility and more about communication. In general, a course with curvature would have required a more robust localization and navigation system. We acknowledge that a global positioning system (GPS) would be useful for such a case; However, at the scale that the project operates such technologies would be too inaccurate (5m tolerance) to pose as viable solutions. This difficulty is further exposed by the less than optimal steering system which the vehicles use to navigate. Indeed, sharp turns and wide banks would be much more challenging to navigate given these mechanics. Convervely, straight tracks relax these requirements, allowing the group to focus on the communication portion of the project. A straight course is much more predictable and lends to a more simple implementation overall. It is also worth mentioning that a straight course also requires much less effort for the group to develop as we anticipate being able to use the university track for our testing. A more complex solution with turns and other complex road structures would certainly require only more time to develop and test. Above all, the decision was made with the idea of best accommodating the core motivation of the project.

4.2 Localization

For localization purposes, we proposed using only IMUs to localize the lead and following vehicles, and using the accelerometer information to infer distance travelled and distance from the lead car, as well as the relative orientations to allow the convoy to navigate the course successfully. However, after more research on IMUs, we found that they become inaccurate after 10m. This presents a problem as following vehicle planning decisions will be made off of inaccurate odometry calculations. Since our track will be 20m long, this solution cannot guarantee that the convoy will be able to travel together properly after 10m, so we decided to use alternative measures on top of IMUs.

An alternative to using IMUs for localization purposes was to use a global camera that would be able to determine the relative locations of all the vehicles and obstacles. However, this would require physically mounting a camera high enough to view the entire track, while still being able to identify the vehicles accurately. This introduces a challenge as to how to mount the camera, which might not be physically feasible in our testing location. Furthermore, given the field of view necessary for the camera, it will be hard to find one that has good enough resolution to determine the locations of the vehicles to the necessary accuracy. Even if such a camera exists, it would likely be unfeasible given our current budget. For these reasons, we decided to move away from a global camera system and use an integrated approach with IMUs and April Tagging.

4.3 Imaging

For imaging, we considered the following approaches.

One approach we considered was using LiDAR for object detection. This has the advantage of producing point cloud data for the surrounding environment, which can then be passed through an object detection algorithm to exactly outline the shape of the obstacles. LiDAR also has the added advantage of producing information that can easily be converted to 3D maps of the environment and could potentially make object detection and planning easier than with traditional RGB cameras. However, after further consideration, we found that most high quality LiDAR sensors are expensive and beyond the realm of possibility given our current budget. There are small LiDAR sensors within our budget, however, the point cloud information that these sensors provide are not rich enough for object detection algorithms such as VoxelNet to run successfully. Therefore, for these reasons, we decided against using LiDAR for sensing and opted to use a more traditional RGB camera.

We also considered using a regular RGB camera, such as the Logitech HD Pro Webcam. These cameras produce high quality RGB images that are suitable for a wide variety of object detection algorithms such as YOLO V3, VGG16, etc. These cameras are also traditionally cheaper than depth cameras and easier to replace should something go wrong. However, standard RGB cameras don't provide depth information, and it is difficult to infer depth from these images. This would make planning difficult, as there would be no way to infer how far the obstacles are from the car, and hence, how extreme of a turn to make to avoid the detected obstacle. This would complicate the planning process, and make it hard to achieve our requirement of 0 collisions. For this reason, we decided to pursue a stereo camera approach, which not only produces RGB images, but can also use the stereo nature of the camera to determine depth of objects in an image.

4.4 Object Detection

4.4.1 April Tagging

We initially considered using April Tagging for object detection. This would involve placing April Tags on the objects, and using the camera and OpenCV to automatically detect the tags. These April Tags are similar in nature to QR codes, however, they encode less information for faster processing. This approach would simplify object detection, since OpenCV would make it easy to create a bounding box around the detected tag. Furthermore, these tags also have the advantage of being unique, and can help with localization, assuming the order of which the tags should appear in the course is known. However, this approach is less realistic and versatile. Since our goal was to simulate true autonomous behaviour as best as possible, we opted to use a more general approach that relies on neural networks.

4.4.2 VGG16 / YOLO v3

In order to achieve the best possible object detection results, we considered using a variety of algorithms. The first class of algorithms we considered were traditional object classification algorithms such as VGG16, and YOLO v3. While these algorithms are very robust and perform well on relevant datasets such as COCO, we were afraid the onboard Jetson Nano would not have the computation power to process images at a fast enough rate. At the current latency outlined above, an object would have 2 opportunities to be detected otherwise the RC vehicle would likely collide with it. If these algorithms were used, this window would likely reduce to 1 opportunity at best for each obstacle to be detected. Assuming an accuracy of 95%, the probability of collision would be around 5%, compared to 0.25% if a similarly accurate algorithm, but faster algorithm were used. For these reasons, we decided to move to MobileNet v2, a more lightweight algorithm that still provides similar object detection accuracy at a much lower latency.

4.4.3 Faster R-CNN

We also considered using object detection algorithms that are able to factor in depth information. One such algorithm that we initially considered was Faster R-CNN. This algorithm relies on using a region proposal network (RPN), in order to develop candidate regions of which objects can be located. This allows Faster R-CNN to generate accurate 3D bounding boxes around objects. However, Faster R-CNN and other algorithms that rely on RPNs are slower than algorithms based on Single Shot Detectors (SSD) such as MobileNet. Furthermore, the main advantage of Faster R-CNN was that it was able to output 3D bounding boxes around any detected objects. However, for our project, we realized that the depth parameter of the bounding box is not critical to our planning algorithm, and hence, traditional 2D bounding boxes would suffice.

4.5 Communication

4.5.1 Communication Technologies

We utilize bluetooth as our mode of communication, but another potential option to communicate between vehicles is WiFi Direct. One clear advantage of WiFi is that it has a long range. In a real world scenario with real cars, WiFi's long range makes it realistic for vehicles to communicate with each other.. WiFi also has desired ad hoc behaviour by seamlessly allowing new nodes to send messages in a network (without having to explicitly pair). The main tradeoffs between the two forms of communication were ease of programiamlity and practicality. Compared to WiFi direct, bluetooth is more simple to program and setup (especially since WiFi direct is a more proprietary technology). Although bluetooth has less range and lower speeds, it is a reliable form of communication and for the vehicle track, the range of bluetooth will be more than enough (our vehicles will be less than 5m away at any given time and current bluetooth technology can easily range up to 50m). Since range is not an issue and our course will be free from large structural obstacles (such as walls or metal objects), we found that bluetooth was the practical solution for our form of communication.

4.5.2 Communication Protocols

Our vehicles will communicate within a piconet which consists of a master node and up to 7 trailing nodes. This form of networking works well for our purposes since it allows the lead vehicle to easily transmit information out to the trailing cars.

Some other existing messaging protocols such as MQTT were considered for the communication architecture as well. Notably, this publish and subscribe framework is used in many IoT edge applications. We decide to move against this option for our solution as the added functionality was not worth the extra implementation cost. We thought about using a transmission protocol which uses handshakes (like TCP) but acknowledgements will add unnecessary overhead to our communication (increasing latency). Since the Bluetooth connection between the vehicles will be stable throughout the course, we do not need a super reliable protocol and are prioritizing low-latency. Therefore we will be using a broadcasting protocol where the lead car sends messages out to the trailing cars and the trailing cars listen for messages. Our project does not require the trailing cars to send messages to the lead car so this protocol works well.

5 SYSTEM DESCRIPTION

5.1 Vehicle Mechanics

To develop our vehicles we opted for a custom designed chassis over the likes of those offered by some third party vendors. This was motivated by the desire for flexibility and exactness to better fit our overall requirements. We needed the vehicle to be able to hold all of our required components, most notably, the Nvidia Jetson and the Stereo camera. This space requirement ultimately drew us away from pre-existing solutions as space would be a premium with such options. In order to accommodate the required components for our vehicle, we designed a full CAD model of the anticipated design for the car. The vehicle chassis will be an 11 x 7in design with customized screw holes for each of the required components. This is done to ensure that there are no issues with components falling off the vehicle in the event of collisions. This CAD model is clearly depicted below as Figure 6:

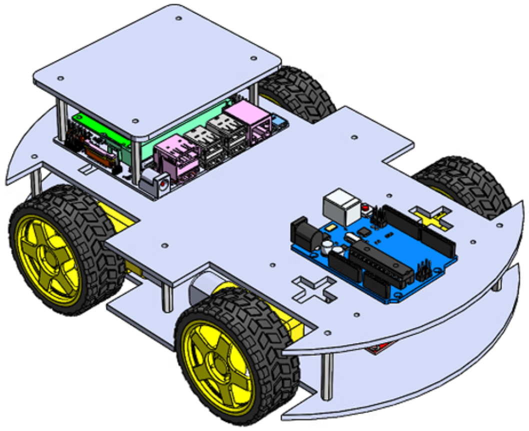


Figure 6: CAD design model for the anticipated vehicle systems

As is visible from the CAD model, the vehicle will be driven by 4 DC motors. These motors will be controlled by using the L298N H Motor Shield and powered by a 9V battery. The connections between these components will be tied together with a mini breadboard to manage cables. The Arduino unit will serve as the controller for the vehicle steering system. For our implementation we decided to receive commands serially from the Jetson Nano via the Arduino USB connection. The Jeston is tasked with providing an approach angle to the Arduino which in turn is used to determine the heading of the vehicle. This is done by defining a discrete space over the unit circle in the interval of 0 to 127 (1 Byte angle space). As depicted in the Figure 5 below, each discrete angle around the circle corresponds to the desired direction that the vehicle should turn.

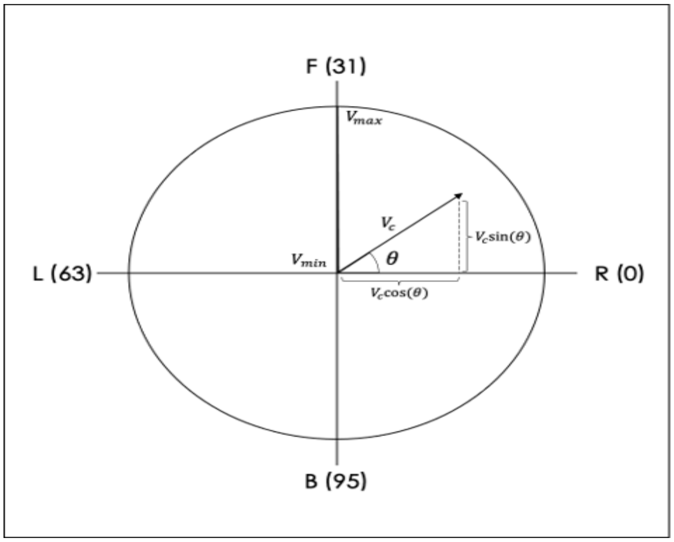


Figure 7: Diagram of the steering system design

5.2 Camera

After researching potential stereo cameras that would suit our requirements, we decided to use an Intel RealSense Camera that we already owned. Not only would this relieve pressure on our budget, but it is also easily integrated with ROS, Python, and OpenCV. The camera also has a horizontal field of view of 86 degrees and a frame rate of up to 90fps, both of which exceed the necessary capabilities for this project. This camera also has the added advantage of being familiar to us, as most of us have a bit of experience integrating the camera with Python in the past. We also engineered the body of our RC vehicles to ensure the camera would fit on the car. Interfacing between the Intel RealSense and the Jetson is well documented and fairly straightforward to implement.

5.3 Object Detection

Obstacle detection will be handled by implementing MobileNet v2 on the Jetson Nano using Python and ROS. Keras has an existing implementation of MobileNet v2 with pre-trained weights that will be used as a baseline due to its baseline success on the COCO dataset. We will then tune the network to achieve better precision and recall by training parts of the network on printouts of images from COCO at various distances and orientations to ensure that the bounding boxes are still properly determined. For the purposes of this project we will not retrain MobileNet on a custom dataset, as we feel that this would take too long to compile a dataset of suitable size, and retraining might not guarantee an increase in performance.

5.4 Path Planning

Outputted images from MobileNet v2 should appear like they do in figure 2. We can combine this image with the depth map from the depth capabilities of the Intel RealSense. This depth map, as seen in Figure 6, along with the bounding box will give the Jetson a reasonable estimate for the distance L from the vehicle's current location to the obstacle. From this point, we can determine the horizontal distance of the object from the depth map, by finding the pixel distance, $D1$, $D2$ from the right and left edge of the bounding box to the center of the image respectively, as in Figure 7. Now, this pixel distance will be scaled heuristically by a factor of k, determined experimentally, to generate an estimate of the real distance to the right edge of the object, $kD1$, and left edge of the object $kD2$. In order to ensure safe navigation around the object, we add a small distance to the above, which will become the target location for the car to navigate to. This gives us a distance of $kD1+\varepsilon$ to the right edge, and $kD2+\varepsilon$ to the left edge. From here, we can calculate the angle to turn the wheels assuming a right angled triangle between the depth of the object and the horizontal distance. This gives the relationship tan $\tan(\theta_1) = \frac{kD1+\varepsilon}{L}$, and $\tan(\theta_2) = \frac{kD2+\varepsilon}{L}$, where θ_1 is the angle to avoid the right edge of the box, and θ_2 is the same angle but for the left edge. From here, we pick the smaller of the two angles to turn the vehicle.

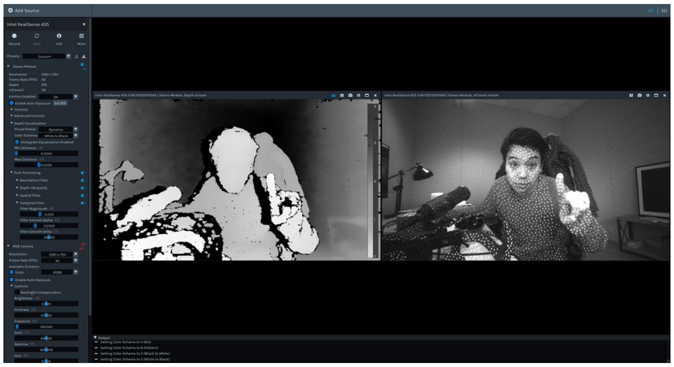


Figure 8: Sample depth image from Intel RealSense camera

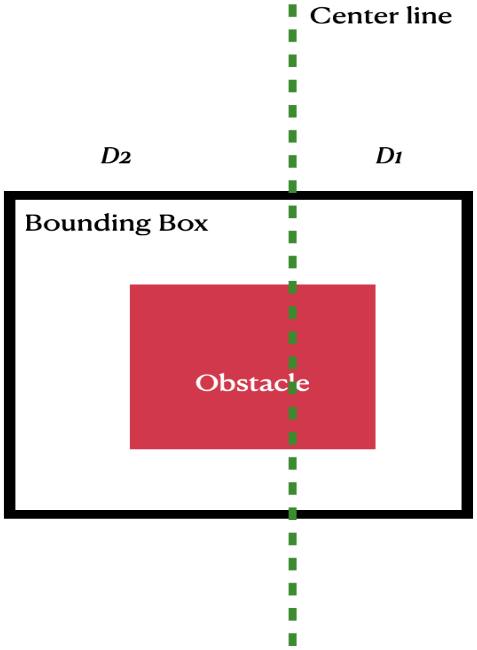


Figure 9: Inputs for path planning: $D1$ is the distance to right edge of bounding box, $D2$ is distance to left edge

Since our requirements dictate that each obstacle has, on average, 2.4 opportunities to be detected, we angle determinations to when a new object is detected. This would ensure that the RC car won't over avoid the obstacles and take too long to finish the course.

5.5 Localization

Localization for our vehicles will be done in two parts. One part will be using April Tags inherent to the course, and the other will be done using IMU data. We decided to use the Adafruit 9-DOF IMU Fusion for the purposes of this project. This IMU is capable of detecting up to 9-DOF, which is more than needed for the purposes of this project. Communication between this IMU and the Jetson will be done using I2C, and the IMU is capable of updating its accelerometer and gyroscope information at a frequency of 100Hz, which is ideal since this information will need to be communicated between vehicles at the same frequency.

5.6 Communication

The Nvidia Jetson 4GB variant does not come native with wireless capabilities so we have to rely on a Wifi + Bluetooth NIC module to provide this feature. Conversely, the Jetson Nano 2GB comes with a WiFi + Bluetooth adapter so no extra hardware is required to enable wireless communication. For the software implementation of the network we intend to use the PyBluez library which is the default open source library for bluetooth development in python. Our communication system must also coordinate with the ROS system so we will be sure to take note of the interface between the two.

6 PROJECT MANAGEMENT

6.1 Schedule

We had to modify our schedule after some deeper consideration between hardware components we were using (such as switching to a stereo camera instead of a LIDAR sensor). The reflections of the new schedule can be seen in the updated Gantt chart (See Appendix C)

6.2 Team Member Responsibilities

Our project can naturally be segmented into three parts. RC vehicle mechanics, Object Detection and Path Planning, and Communication Protocol.

Joel will be taking the lead on RC Vehicle manufacturing, including designing and manufacturing the body, as well as creating an interface to allow the vehicles to turn properly simply by inputting an angle. As soon as the RC cars are working to the specified requirements, Joel will assist Fausto and Jeffrey on the communication and object detection/planning parts of the project.

Jeffrey is in charge of the object detection and path planning aspect of the project. His responsibilities include connecting the Intel RealSense camera to the Jetson and extracting the RGB image and depth map, developing and tuning the MobileNet v2 object detection model to ensure obstacles are detected successfully, implementing, and modifying, if necessary, the planning algorithm to meet our requirements. Jeffrey will also work with Joel to test this aspect of the project on the physical RC cars.

Fausto will be working on establishing Bluetooth connection between RC cars and creating message structures for ease of message parsing and decoding. His tasks involve developing a suitable communication protocol to allow the following vehicles to maintain the convoy formation, as well as determining how to incorporate the communicated information into planning for the following vehicles.

graphicx

6.3 Budget

Part	Price	Qty	Subtotal
Wheel + DC Motor (x4)	$16.99	3	$50.97
9v Power Supply Holder	$8.98	1	$8.98
5v Power Supply	$15.95	3	$47.85
Jeston Nano (2Gb)	$59.00	2	$118.00
Jetson Nano WiFi + Bluetooth Card	$26.99	1	$26.99
Micro SD Card 32 GB (x3)	$15.99	1	$15.99
USB to Arduino Cable (x5)	$13.27	1	$13.27
9-DOF Absolute Orientation IMU Fusion Breakout	$38.95	3	$116.85
Metric Steel Pan Head Screws M3 (x100)	$2.96	1	$2.96
Steel Hex Nut M3 (x100)	$0.88	1	$0.88

Table 2: Purchased items list

The table above summarizes the purchased parts for this project. The total cost for these items comes to $426.14. Since we already had some of the components before, we did not have to purchase them. These such items are listed in the table below.

Part	Price	Qty	Subtotal
Arduino Uno R3	$21.40	3	$64.20
Nvidia Jetson Nano (4GB)	$99.00	1	$99.00
Wireless NIC Module for Jetson Nano	$26.99	1	$26.99
Intel RealSense D415	$160.00	1	$160.00

Table 3: Borrowed items list

6.4 Risk Management

In order to mitigate the budget risk, we decided to CAD our own RC cars and avoided excess spending on better components to make these cars. We also simplified the steering mechanics and used an Arduino as a motor controller for simplicity, since we are more familiar with this method of motor control, rather than having the Jetson communicate with the motors directly. If the steering control of the RC vehicles fails to perform at the necessary level to avoid objects, we have planned to reduce the size of the objects and increase the distance at which objects are to be detected, allowing planning decisions to be made easier without putting as much stress on the physical RC vehicle.

The object detection aspect of the project is dependent upon MobileNet performing at a suitable level, as well as the algorithm having a low enough latency to meet our requirements. We realize that in practice there are a lot of variables and we have developed an alternative approach should we determine our current plan is unfeasible. This would involve putting April Tags on all our objects, and using the Intel RealSense to detect the April Tags and use the associated bounding box for planning decisions. Alternatively, if the computation latency is too expensive to meet our requirements, we also considered slowing the car down such that the window to detect objects becomes larger, and false negatives do not become critical.

Finally, we realized that our solution approach is very reliant on communication between lead and following vehicles, as this is what defines the convoy. We initially wanted to develop our own complex protocol to ensure robustness, however, due to the complexity of this task, we decided to simplify the communication to just broadcasting instructions between lead and following vehicle. If we fail to develop an accurate communication protocol between the lead and following vehicles, our alternative approach is to physically tie the vehicles together in order to maintain a strict bound on how far the vehicles are, which would make determining what instructions to tell the following vehicle much easier.

7 RELATED WORK

There are a lot of projects online that deal with autonomous navigation of RC vehicles. However, most of these projects inovlve a single car navigating through a course, rather than a convoy. One notable project was one produced by students at WPI also created a similar project that involved autonomous navigation of RC vehicles. However, unlike our project they relied on LiDAR and other forms of odometry using EKF to perfect localization. Our project not only performs object detection differently, but also attempts to navigate a convoy of vehicle, rather than an isolated RC vehicle. Other projects have also used similar object detection approaches, but all rely on more complex path planning approaches, rather than a simple bounding box approach we decided to employ.

8 SUMMARY

From our preliminary solution approach, our project has undergone several design changes to meet our requirements. However, despite several iterations of design, our solution still has limits on performance. RC Vehicle performance is limited by fluctuations in voltages, and the assumption that all the motors will behave similarly. Our object detection algorithm and planning assume ideal lighting conditions that might not always be possible. Finally, communication is limited by the behaviour of the Bluetooth chips on the Jetson. To improve our performance and make our project success, we have considered several alternatives to the solution presented above. With a larger budget, we would likely have bought more expensive RC parts, such as Donkey cars, more powerful computation nodes, which would guarantee better consistency in performance.

References

[1] X. Wu, M. Xu, and L. Wang. "Differential speed steering control for four-wheel independent driving electric vehicle". In: (2013), pp. 1–6. DOI: 10.1109/ISIE.2013.6563667.

9 Appendix

9.1 Appendix A

Calculation for Estimated Vehicle Maximum Velocity

$$\omega = 2\pi f$$
$$v = r\omega$$

RPM @6V (frequency)	200RPM ± 10%
Wheel Diameter	65mm

Table 4: Vehicle Parameters

$$\begin{aligned} f = 200rpm10\% = 200/60s = 3.33Hz10\% \\ r = D/2 \\ v = \frac{\omega D}{2} = \frac{2D\pi f}{2} = \pi D f \\ v = \pi(65 \cdot 10^{-3})(3.33Hz) \pm 10\% = 0.68m/s \pm 10\% \end{aligned}$$

9.2 Appendix B

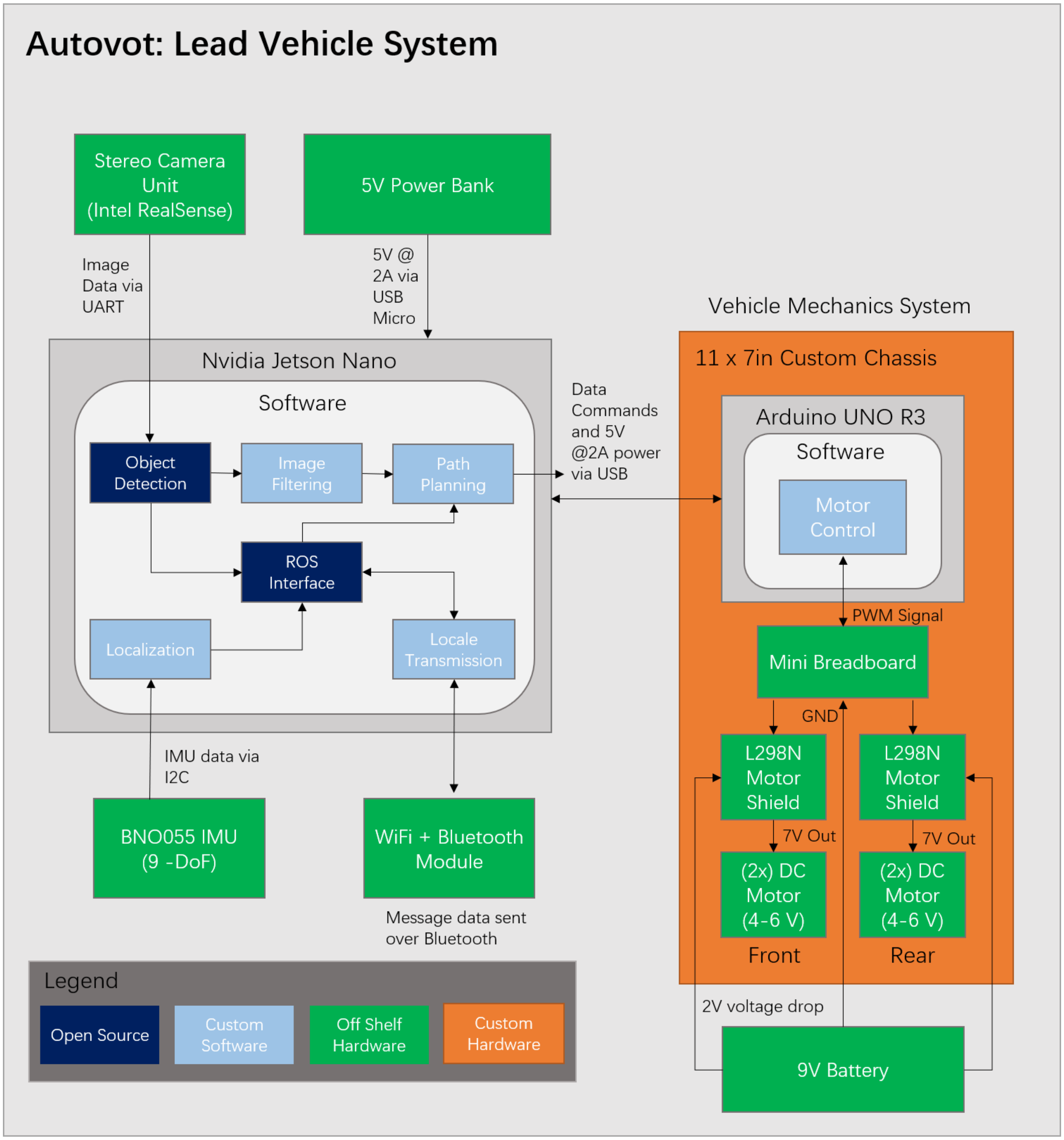


Figure 10: Architecture Diagram for the lead vehicles

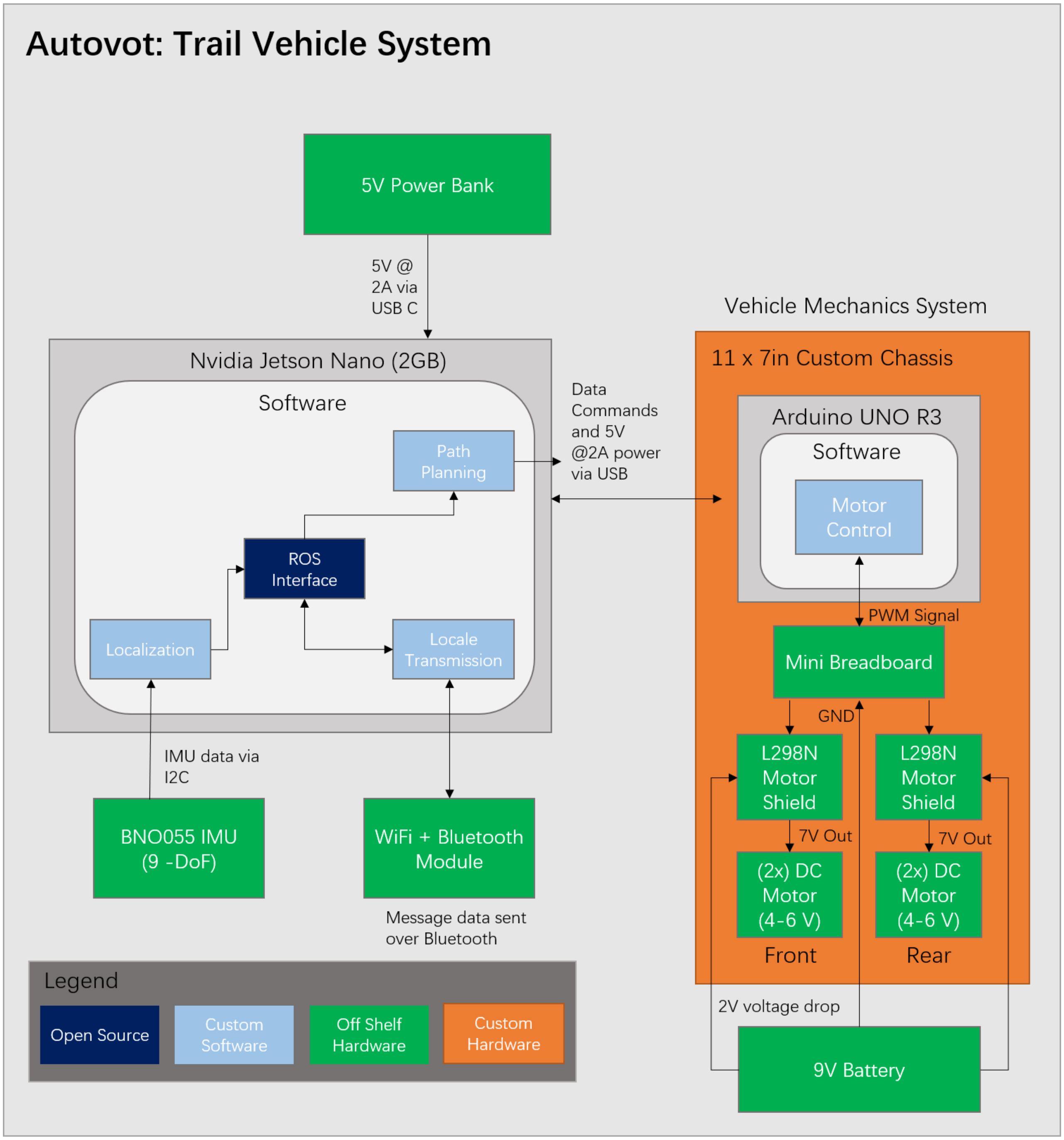
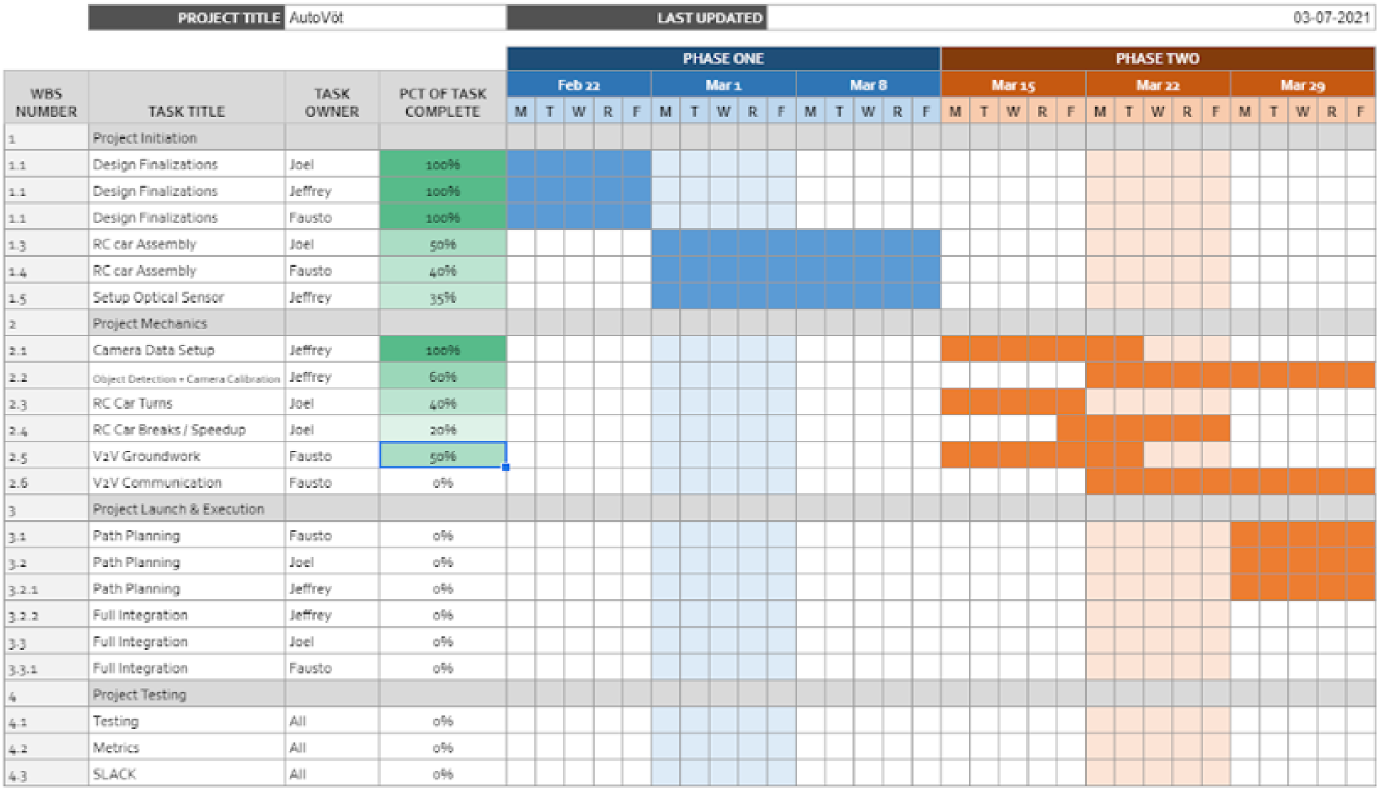


Figure 11: Architecture Diagram for the trail vehicles

Appendix C



PROJECT TITLE	AutoVöt	LAST UPDATED	03-07-2021

				PHASE ONE															PHASE TWO														
				Feb 22					Mar 1					Mar 8					Mar 15					Mar 22					Mar 29				
WBS NUMBER	TASK TITLE	TASK OWNER	PCT OF TASK COMPLETE	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	Project Initiation																																
1.1	Design Finalizations	Joel	100%	■	■	■	■	■																									
1.1	Design Finalizations	Jeffrey	100%	■	■	■	■	■																									
1.1	Design Finalizations	Fausto	100%	■	■	■	■	■																									
1.3	RC car Assembly	Joel	50%						■	■	■	■	■	■	■	■	■	■															
1.4	RC car Assembly	Fausto	40%						■	■	■	■	■	■	■	■	■	■															
1.5	Setup Optical Sensor	Jeffrey	35%						■	■	■	■	■	■	■	■	■	■															
2	Project Mechanics																																
2.1	Camera Data Setup	Jeffrey	100%																■	■	■	■	■	■	■								
2.2	Object Detection + Camera Calibration	Jeffrey	60%																					■	■	■	■	■	■	■	■	■	■
2.3	RC Car Turns	Joel	40%																■	■	■	■	■										
2.4	RC Car Breaks / Speedup	Joel	20%																				■	■	■	■	■	■					
2.5	V2V Groundwork	Fausto	50%																■	■	■	■	■	■	■								
2.6	V2V Communication	Fausto	0%																					■	■	■	■	■	■	■	■	■	■
3	Project Launch & Execution																																
3.1	Path Planning	Fausto	0%																										■	■	■	■	■
3.2	Path Planning	Joel	0%																										■	■	■	■	■
3.2.1	Path Planning	Jeffrey	0%																										■	■	■	■	■
3.2.2	Full Integration	Jeffrey	0%																														
3.3	Full Integration	Joel	0%																														
3.3.1	Full Integration	Fausto	0%																														
4	Project Testing																																
4.1	Testing	All	0%																														
4.2	Metrics	All	0%																														
4.3	SLACK	All	0%																														

Figure 12: Gantt Chart Part 1

Appendix D

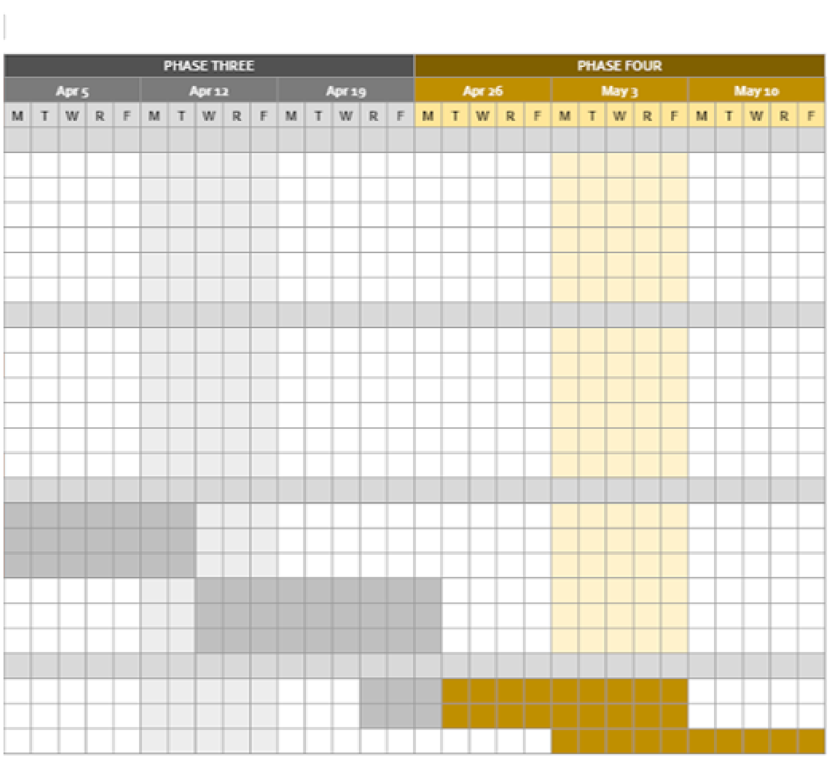


Figure 13: Gantt Chart Part 2