SP-13: EVT Autonomous Electric Vehicle

CS 4850 Section 1 Spring 2024

01/14/2024

Team Members

Roles	Name	Major responsibilities	Cell Phone / Alt Email
Team leader	Thomas Fitzgerald	Code Technician	470-535-8954 thomasfitzgerald225@gmail.com
Team members	Ashton Hinton	Project Manager	678-591-7573 ashhin10@gmail.com
Advisor / Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	770-329-3895 Sperry46@kennesaw.view.usg.edu

Team Website: https://sp13evt.github.io/

GitHub Account: https://github.com/SP13EVT

Lines of Code / Comments: Seventy-Two

Number of Components: Two

Contents

Abstract

The KSU Electric Vehicle Team has consistently led the development of autonomous electric go-kart technology. The team has long explored the core technologies, and our project SP-13 EVT (Voltron) built on the latest learnings gained in the discipline, focusing on two aspects. The first one involved improving the autonomous functionality of Voltron, while the latter centered on enhancing the documentation framework to create a solid base for further development. This paper specifically details the stepwise progress made over multiple semesters, demonstrating the use of state-of-the-art technology, Docker, advanced AI, and rigorous documentation to ensure the future potency of developments.

Several senior teams have contributed – several provided enhanced navigation algorithms, while others focused on optimizing software deployment and a dedicated EVT wiki. Such initiatives have significantly helped improve Voltron's performance during nationwide competitions and guarantee sustained knowledge in the EVT. However, the challenges associated with documenting highly complicated software engineering creations remain, yet reading ensures the knowledge continues to be more comprehensive and accessible. This paper sets out the results of this project, which is a well-organized EVT wiki that supports learning, as well as improved autonomous abilities, which give momentum to future developments. The document concludes the transformative experience of EVT and presents future perspectives such as implying more potent AI technologies and continued adjustments of knowledge exchanges to retain knowledge transition robust.

1.0 Introduction

1.1 KSU EVT Team

The KSU Electric Vehicle Team (EVT) is a student led and operated club that builds and maintains an autonomous electric go-kart, called Voltron. The club races Voltron in the EVGrandPrix, a competitive event held by Purdue and features teams from Georgia Tech, University of Michigan, and UC Berkley. Due to the efforts of the EVT Team, Voltron has won two of the national championship races held.

1.1 Voltron's Software

As mentioned before, Voltron is the autonomous vehicle that is developed and maintained by the EVT team. While the club does maintain Voltron's hardware components, this section will focus on discussing its software composition.

Voltron's software is hosted via a Linux operating system and heavily utilizes the Robot Operating System (ROS2) libraries. Voltron's software is composed of various nodes that are initiated and run congruently. Each node is responsible for a specific task and communicates with other nodes via publisher/subscriber or service calls. A class diagram of the nodes and how they communicate is shown in the figure below.

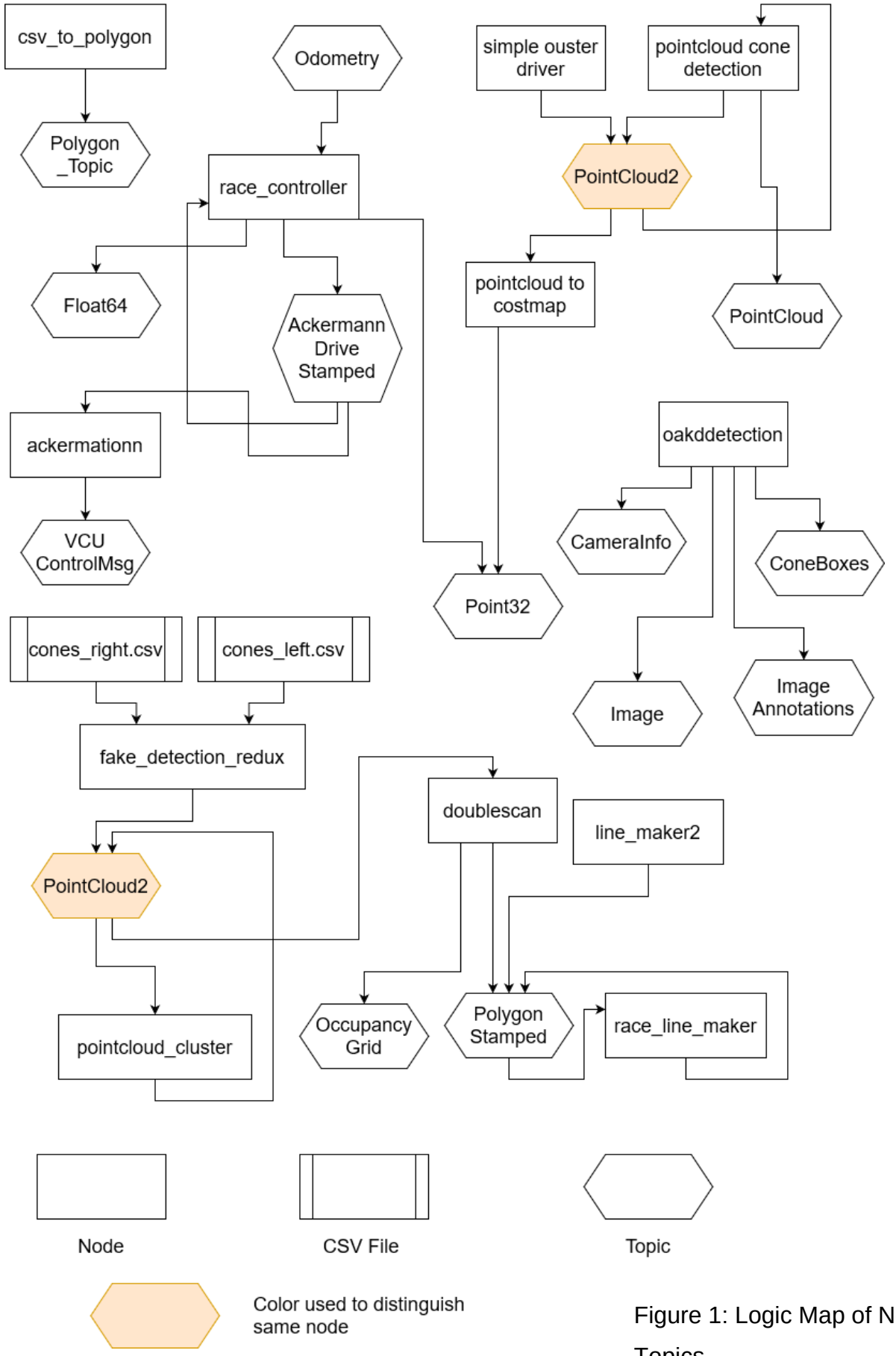

Figure 1: Logic Map of Nodes and Topics

1.2 Fall 2022 EVT Team

In the fall of 2023, James Rivett worked on a senior project for CS4850 in cooperation with the EVT team. Rivett worked with the club's leaders to develop a docker-based workspace. Docker is a specialized platform that allows individuals to build applications. In the case of Rivett's project, he used x11Docker to create a virtual machine of EVT's repository. This allowed individuals to work and develop the software in a private workspace.

The second component of Rivett's project was improving Voltron's communication between its directional input and the physical motor. Before this project, Voltron only estimated the distance to turn without any empirical data. This was remedied by creating a function to fit the motion of capture of Voltron's steering joint. The function was then inverted so that the motor position was output when a directional angle was inputted. This drastically improved the steering communication for Voltron and allowed it to follow its generated pathway closer.

1.3 Spring 2023 EVT Team

In the following semester another group decided to work with the EVT Team for their CS4850 senior project. This team, composed of Jhavon Simon, Kevin Polanski, and Wyatt Whitefield, primarily focused on creating a Development Pipeline. A development pipeline is a structured process for deploying code changes. The team created the pipeline by allowing for the docker system to support Continuous Integration (CI) and Continuous Delivery (CD). CI is the practice of merging a developer's code changes with the central repository, and CD is the practice of delivering code changes to developer's codes.

A second objective that this team focused on was improving the docker system made by the previous senior project. While the previous system improved the EVT team's development process, integrating and building the necessary software was still drastically time consuming. Thus, this group helped to reduce the time needed for integration by incorporating a baseline configuration of the Voltron's software and

building a workspace based on the baseline. This removed the previous practice of downloading the entire workspace contents every time a software test was performed.

1.4 Fall 2023 EVT Team

The next CS4850 Senior Project to work with the EVT Team was composed of the members Austin Klein, Brandon Solon, and Caleb Mcmaster. This group developed two additional components to Voltron to drastically improve its performance. The first of these components was a Race Controller node that managed and switched Voltron's different navigation algorithms. This node worked by defining the starting/finish line of the racetrack. For the first lap, Voltron would be controlled by its reactive controller, but would switch to its model predictive controller once it crossed the defined line into the second lap.

The second component developed for Voltron was a program for optimizing the race line. This program worked by taking the centerline of the track, identified through the dynamic model created by the reactive controller, and generating a path with minimum curvature. This program ensured that the path Voltron followed would be the most efficient path through the course.

1.5 Spring 2024 EVT Team

With the previous senior projects that worked with the EVT team described, it is time to discuss the current project. Our group was initially tasked with developing an addon for the model predictive controller. This addon would be responsible for identifying when a new object was added to the track and switching the active race controller to avoid it. Several months were spent analyzing and learning Voltron's software to little success. Meetings with the EVT Team leader revealed we were missing important files we were unaware of, and that the current implementation of the software was inoperable.

Due to these obstacles, we realized we would not be able to develop a working add-on for Voltron and met with the EVT leader and Professor Perry. It was decided that the goal of our senior project would then switch to developing the documentation for the

EVT Team. This would allow new members to become better acquainted with Voltron's software and have access to resources and support for development.

2.0 Requirements

The requirements for our contribution to Voltron were originally focused on the needs to make an addon for dynamic obstacle recognition and avoidance, and later changed to thoroughly document the work of EVT for future persistence. The latter necessity arose due to the failures in the project's continuity, mainly as various new members were enrolled. Specific requirements were directed at designing a docile applied outgrowing wiki which would become the fundamental basis for future progression and boarding. This section lists the functional and nonfunctional requirements that defined the team's work for both the development of the addon and documentation sections.

2.1 Addon Nonfunctional Requirements

2.1.1 Usability

The user interface for initiating mode switches should be intuitive and easy to use for EVT team members.

2.1.2 Performance

Define performance requirements such as response times for mode switching and processing speeds for real-time data analysis to ensure optimal system performance.

2.1.3 Security

Specify security requirements for data encryption, access control mechanisms, and secure communication channels to protect against unauthorized access.

2.1.4 Safety

Define safety requirements to ensure the system operates reliably under various racing conditions, including fail-safe mechanisms and compliance with safety standards for autonomous vehicles.

2.1.5 Reliability

The system should operate reliably under various racing conditions and track scenarios. to minimize the risk of system failures.

2.2 Addon Functional Requirements

2.2.1 Mode Switching

The add-on must allow for dynamic switching between reactive controller and MPC modes to adapt to changing track conditions and obstacles.

2.2.2 Real-time Object Detection

The system should detect new obstacles in real-time using the onboard LIDAR system for enhanced situational awareness.

2.2.3 Service Calls

Implement service calls within the race controller system to facilitate mode switching seamlessly without disrupting vehicle operation.

2.3 Documentation Nonfunctional Requirements

2.3.1 New Guy Document

The created Wiki must contain a section dedicated to new members of the club. This section must contain useful tutorials and guides to equip new members with a better understanding of Voltron.

2.3.2 Information Accessibility

The created Wiki must allow for information to be more accessible than previously.

2.4 Documentation Functional Requirements

2.4.1 Wiki Creation

A new bookshelf must be created on the EVT's Bookstack to serve as a wiki. This wiki must contain information pertinent to Voltron's programming and any related media.

2.4.2 Voltron Programs Documentation

The existing programs utilized by Voltron must be enhanced with comments that describe the functionality of the program.

3.0 Development

3.1 Documentation Transition

At this point, the development phase was already in full swing, and the writing was on the wall; more effort was needed to divert from improving the software directly and focus on developing indispensable documentation. The primary goal for the team was creating an extensive EVT wiki with detailed guides devoted to Linux and ROS2, the two main components of Voltron's software structure. This goal was motivated by the

necessity to reinforce the intellectual base of the EVT to ensure the in-depth details regarding Voltron's programming were no longer siloed knowledge but a shared asset.

Coding Standards and Commenting: One of the most important tasks realized during the development process was the establishment of critical coding standards and strict comment-based protocols. Each line of code implemented into Voltron's system was explained with descriptive comments, ensuring that anyone, even with minimal prior exposure to the project, could understand and contribute.

3.2 Obstacles

The team faced the difficult task of documenting an already complex and layered software system. The absence of previous comprehensive documentation meant that the team had to reverse-engineer many processes to accurately capture them in the EVT wiki. Creating a system that was both detailed and approachable required difficult efforts and collaboration.

While the documentation added by this project improved the overall understanding of Voltron's program, we were unable to add commentary to every programming file. Some of the files dealt directly with Voltron's hardware and were incredibly difficult to understand. Due to this, no comments were added because we could not be sure the comments would accurately reflect the program's purpose. We tried to overcome this issue by reaching out to the KSU EVT team for help in understanding the file, but due to other pressing concerns they were unable to aid us.

4.0 Analysis

4.1 Wiki Documentation

4.1.1 Enhancing the EVT Wiki WorkSpace

Once our project became tasked with focusing on the documentation, we began with improving their workspace by streamlining their operations and enhancing user

experience. Our goal was to create a central hub where all information– from the EVT's project planning to software and electronics documentation – was not only accessible but also intuitive to navigate.

4.1.2 User Friendly Design

We started off by revamping the homepage, which now will serve as the gateway to all EVT documentation. Our design ensures that both new and existing team members can easily find the necessary resources they need. Categories are represented clearly andd visibly with clickable images that lead to the relevant sections, such as Software, Electronics, Prototype Documentation, and Marketing.

4.1.3 Intuitive Documentation Layout

Each documentation category received a structured layout overhaul:

- Software Documentation: Arranged in a logical flow, it allows members to follow the software development lifecycle from requirements to deployment.

4.1.4 Streamlined Information Access

To address the challenge of navigation and information retrieval, we:

- Implemented a 'Search' function that allows for quick access to specific topics or documents.
- Updated 'Page Navigation' to provide and facilitate a clear understanding of the user's current location within the wiki.

4.1.5 Comprehensive Learning Path

We crafted a detailed 'Learning Path' for new members, guiding them through the EVT system and providing milestones to gauge their progress. This section includes:

- Step-by-step tutorials and links to external resources for foundational skills such as Linux and ROS.

- An overview of setting up the workspace, which details the process of preparing the development environment.

4.1.6 Wiki Components

Introductions (New Members)

This section serves as the entry point for new EVT team members, offering a comprehensive overview of the information and tools necessary to get started with Voltron. We introduced essential sub-sections that serve as a primer to onboard new members effectively:

Guides and Tutorials: A collection of beginner-friendly resources was created, focusing on practical, hands-on guidance for Linux and ROS2. These tutorials help to simplify the software environment and ensure that even members with limited prior experience can quickly come up to speed with experienced members.

Set Up a Workspace: Recognizing the importance of a well-configured development environment, we provided step-by-step instructions to help manage the setup process. This section includes hardware and software requirements, installation guides, and tips for troubleshooting common setup issues.

First Time Launching the Simulation: We dedicated a subsection to guide users through their first simulation with Voltron, acknowledging that initial exposure to such complex systems can be daunting. The instructions were crafted to be clear and accessible, with supplementary resources to support learning.

Glossary

We created a glossary to help new members navigate the technical aspects and acronyms they’re likely to encounter during their time working in the workspace. This resource is subject to growth as their specific project evolves so that it remains a relevant and useful reference tool.

FAQ Section

This section is for anticipated/ common questions or concerns that could possibly come up from new or existing members of the team. We created this section to provide quick and clear answers. This document allows members to both seek out information and contribute their inquiries, fostering a collaborative knowledge base.

Learning Path

The learning path that we suggested includes milestones that help gauge a new member's progress. It is designed to be incremental, ensuring a solid understanding of each phase before advancing, which optimizes the learning experience.

Contact Points

We listed key contact points, including a link to the team's Discord channel, offering an immediate line of communication for collaborative discussion and support. All contacts are to be made within the discord as that is how the team wants it to be and where most of the communication takes place.

Feedback Mechanism

Emphasizing the iterative nature of our project, we implemented a structured feedback mechanism. This includes clear instructions on how to submit feedback and encouragement of ongoing dialogue to continually refine the documentation.

Code Of Conduct & Contribution Guidelines

This section is basic guidelines of what we thought the team would want. It is a respectful and productive work environment, so we clearly outlined the code of conduct. We also explained how new members can actively contribute, ensuring they feel welcomed and valued from the beginning of their time on the team.

Regular Updates

Transparency in documentation is always critical. We indicated the frequency of updates and provided direct access to the latest versions, ensuring that members are always working with the most current information.

Access to Additional Resources

Recognizing the importance of external resources, we provided links to EVT's code repositories and additional learning materials. This open-access approach empowers team members to delve deeper into the project's technical aspects.

4.2 Programming Documentation

The second component of the work done in this project was the enhancement of the existing code for Voltron. The programming code was parsed and analyzed such that comments would be added to accurately explain the section's purpose.

A large majority of this work was explaining how and where a created node would interact with other nodes. This was done through describing the subscribers and publishers and the topics they would interact with, as well as listing the service calls made to other nodes. Figure 2 and Figure 3 show a before and after example of the documentation that was done to the create_line function in the race_line_maker file.

```
# TODO make an action
def create_line(self, _request, response):
    if self.line_msg == None:
       return

    centerline_x = np.empty((0, 1))
    centerline_y = np.empty((0, 1))

    for p in self.line_msg.polygon.points:
        centerline_x = np.append(centerline_x, p.x)
        centerline_y = np.append(centerline_y, p.y)

    distance = np.cumsum(np.sqrt( np.ediff1d(centerline_x, to_begin=0)**2 + np.ediff1d(centerline_y, to_begin=0)**2 ))
    distance = distance/distance[-1]

    fx, fy = interp1d( distance, centerline_x ), interp1d( distance, centerline_y )

    # 200 = the number of points in which we want to represent the track
    alpha = np.linspace(0, 1, 200)

    x_regular, y_regular = fx(alpha), fy(alpha)

    centerline = np.column_stack((x_regular, y_regular))
    centerline_flipped = np.flip(centerline, axis=0)
    for p in centerline_flipped:
        point = Point32()
        point.x = p[0]
        point.y = p[1]
        self.centerline_msg_.polygon.points.append(point)

    self.centerline_msg_.header.frame_id = "map"
    self.centerline_msg_.header.stamp = self.get_clock().now().to_msg()

    self.get_logger().info('created race line')
    return response
```

Figure 2: Before Comments Were Added to race_line_maker.py

```python
# Method that triggers when a service call is made to /race_line_maker/create_line
def create_line(self, _request, response):

    # Checks if needed data is available for the line's creation
    if self.line_msg == None:
       return

    # Creates an empty numpy array
    centerline_x = np.empty((0, 1))
    centerline_y = np.empty((0, 1))

    # Adds each corresponding point to the empty arrays
    for p in self.line_msg.polygon.points:
        centerline_x = np.append(centerline_x, p.x)
        centerline_y = np.append(centerline_y, p.y)

    # Calculating the euclidean distance of each point in the centerline with the starting point
    distance = np.cumsum(np.sqrt( np.ediff1d(centerline_x, to_begin=0)**2 + np.ediff1d(centerline_y, to_begin=0)**2 ))
    distance = distance/distance[-1]

    # Creates a function based on the data points
    fx, fy = interp1d( distance, centerline_x ), interp1d( distance, centerline_y )

    # 200 = the number of points in which we want to represent the track
    alpha = np.linspace(0, 1, 200)

    x_regular, y_regular = fx(alpha), fy(alpha)

    # Adds each of the points of a line to message to upload to PolygonStamped
    centerline = np.column_stack((x_regular, y_regular))
    centerline_flipped = np.flip(centerline, axis=0)
    for p in centerline_flipped:
        point = Point32()
        point.x = p[0]
        point.y = p[1]
        self.centerline_msg_.polygon.points.append(point)

    self.centerline_msg_.header.frame_id = "map"
    self.centerline_msg_.header.stamp = self.get_clock().now().to_msg()

    self.get_logger().info('created race line')
    return response
```

Figure 3: After Comments Were Added to race_line_maker.py

5.0 Results

5.1 Documentation Outcomes

Overall, the outcome of this phase is a structured EVT wiki with an abundance of information including software installation, nodes operation and testing, and problem-solving. This wiki is a great asset that enables simplified mentoring and knowledge transfer. As a result, the intellectual value of EVT becomes available and tangible and only grows through each teams' generation.

5.2 Test Report

When looking back at the efficacy of Tests: Testing was necessary as a convenient way to ensure the documentation is usable. New team members had to create an environment, simulate their devices on running condition or even edit system files using the wiki as their only source of information. The result of these tests reassured the ability and progress made in the writing of documentation.

6.0 Version Control

We created an additional bookshelf on the EVT's Bookstack. In this bookshelf we were able to create and edit documents and guides. The website maintains a changelog and tracks the users and their change commitments. The bookshelf we created was called "Documentation Prototype" but is now labeled as "Voltron Documentation".

In alignment with the documentation efforts, the team embraced a structured version control system, using GitHub to manage software updates. The team also created a new bookshelf on the EVT's Bookstack to document version control practices. This repository included logs of updates, documenting the evolution of Voltron's software and the parallel development of the wiki.

https://wiki.ksuevt.org/shelves

7.0 Conclusion

Reflective Synopsis: For the section, we approached the coding work with the reflection of the transformative journey. Our statement for success with the project is not the many lines of code we wrote but the cognition of a sustainable knowledge-sharing ecosystem that we have left behind. This legacy we created will support future teams to take the brain and technology that is Voltron and EVT even further into the heights of innovation.

Future Directions: Looking ahead, Voltron shall further develop their EVT wiki and face the implementation of bigger, better, and more applied AI for advanced autonomous capabilities, amongst other continual improvements to the efficiency of LKT. The groundwork for the ideal future with this “proof of concept” we completed is very much laid down.

Appendices Documentation Artifacts: The appendices attached hereto are some of the various sources of detailed walkthroughs, code writing, and presentation of diagrams combinations that are artifacts. They have been designed as a lasting educational platform for the development of autonomous electric vehicles.