ELEC 390: Principles of Design and Development

Final Report for Project DMW

Duck Moving Vehicle

Team 2

Team Members: Name (NetId)
Jacob Chisholm (21jc138)

Hendrix Gryspeerdt (21hgg3)

Luke Strickland (21laps1)

Matthew Szalawiga (21mps6)

Date of Submission:

Prepared For:

Dr. Matthew Pan

Dr. Amr Mohamed

Statement of Originality

The members of Team 2, listed on the page above, declare that the work presented in this document is our own. Any information, materials, data or content presented that is not solely ours has been properly acknowledged and cited in accordance with the IEEE format.

The use of AI and large language models (LLMs) has been limited to the acceptable uses as outlined in the ELEC 390 course syllabus.

Signed,

Jacob Chisholm	Luke Strickland
Matthew Szalawiga	Hendrix Gryspeerdt

Contribution Table

Section	**Content**	**Write-Up**	**Editing**
Executive Summary	Hendrix, Luke, Matthew, Jacob	Hendrix	Jacob, Luke, Hendrix, Matthew
Introduction	Luke	Luke	Jacob, Luke, Hendrix, Matthew
System Design	Jacob, Luke, Hendrix	Jacob, Matthew, Luke, Hendrix	Jacob, Luke, Matthew, Hendrix
Methodology	Jacob, Luke, Hendrix	Luke, Hendrix	Jacob, Luke, Hendrix, Matthew
Implementation	Jacob, Hendrix	Hendrix	Jacob, Luke, Hendrix,
Ethical Decision Making and Considerations	Matthew	Matthew, Luke	Luke, Hendrix, Matthew
Results	Jacob	Jacob	Jacob, Luke, Hendrix, Matthew
Discussion	Jacob, Hendrix, Luke	Jacob, Hendrix, Luke	Jacob, Luke, Hendrix, Matthew
Conclusions and Reflections	Jacob, Luke, Hendrix, Matthew	Hendrix, Jacob, Luke	Jacob, Luke, Hendrix, Matthew

Executive Summary

The objective of this project was to develop an autonomous an autonomous taxi using a modified PiCarX [1] capable of operating in the model environment of Quackston [2]. The company name for our team is “The Quaxi Company.” The project was moderately successful in multiple aspects such as obstacle detection and lane following, however due to complications over the course of the 3 months it fell short on some other aspects.

The end goal of the project was to have the autonomous tax compete in the Quackston model environment on March 24-25, 2025. Due to Teaching assistant strikes, the competition did not proceed, however meaningful tests were completed in a testing environment the team had set up.

Despite the complications, the team developed a basic autonomous driving system for the PiCarX. Lane detection and following, as well as obstacle detection and avoidance were accomplished using classical image processing techniques. The passenger cabin and restraint equipment system was accomplished with a custom duck lift. The routing and navigation system was developed, however was not fully integrated into the driving system and the competition Vehicle Positioning and Fare System [3].

As the project developed, requirements and objectives were re-thought and the designs were changed. For example, the requirement of detecting road signs was dropped because lane line and stop line detection was sufficient. The team also learned about the ethical considerations regarding vehicle behaviour in different road conditions and passenger and fare selection. Despite the challenges and difficulties of the project, there multiple learning benefits came from work on this project.

Table of Contents

1) Introduction

1.1 Background Information

1.1.1 Autonomous Driving

Autonomous driving refers to vehicles capable of navigating and operating without any human input. These vehicles rely on advanced technology to perform tasks traditionally done by the driver, such as steering, braking, and decision-making. To achieve this, autonomous vehicles (AVs) utilize a suite of sensors, such as cameras, LIDAR, radar, and ultrasonic sensors, which create a complete view of the car's surroundings. These sensors gather much more information than a human is able to. An onboard computer processes this data in real-time, enabling the vehicle to make decisions and self-drive. The Society of Automotive Engineers (SAE) has established six levels of autonomous driving, ranging from Level 0 to Level 5. Levels 0 to 2 focus on driver assistance features, where the driver must remain fully engaged and supervise the system.

- Level 0: Provides warnings and momentary assistance, such as collision alerts.
- Level 1: Offers either steering or braking/acceleration support such as adaptive cruise control.
- Level 2: Delivers both steering and braking/acceleration support but still requires constant driver oversight.

In Levels 3 to 5, the vehicle assumes greater responsibility, requiring little to no human intervention.

- Level 3: Can autonomously drive under specific conditions but requires the driver to take over when requested.
- Level 4: Operates independently under limited conditions without needing driver intervention.
- Level 5: Achieves full self-driving in all conditions and environments [4].

As of today, the highest level of autonomous driving achieved commercially is Level 4. Companies like Waymo have deployed autonomous taxis that operate under specific conditions without requiring human intervention [5]. Meanwhile, Tesla's vehicles are classified as Level 2, as their Autopilot system still requires drivers to remain ready to take over at any moment.

1.1.2 Significance and Challenges

Autonomous driving has the potential to significantly impact safety, accessibility, and efficiency.

- Safety: AVs reduce the number of accidents caused by human error. Studies show that SAE Level 4 self-driving cars are approximately 36 percent less likely to be involved in moderate-injury crashes and 90 percent less likely to be involved in fatal accidents [6].
- Efficiency: Self driving cars are more fuel/ power efficient. They are able to drive in such a way that is 15-20 percent more efficient, industry experts call this 'eco driving' [7].
- Accessibility: Self-driving cars offer more options to individuals unable to drive, such as the elderly or those with disabilities.

With ongoing technological advancements, autonomous driving has shown immense promise for shaping the future of transportation safety, accessibility, and efficiency.

The field of AVs faces numerous ethical and technical challenges. Public acceptance is a significant one, as many people are hesitant to even consider the idea of AVs. Additionally, predicting the behaviors of other vehicles on the road requires advanced algorithms and advanced decision-making systems. Another major challenge is enabling AVs to operate safely in varying conditions, such as during a snowstorm when sensors may become obstructed. Ethical concerns can also pose a challenge, such as addressing moral dilemmas where AVs might need to prioritize lives in the event of a crash, raising questions about whose life should take precedence, if at all. Also, legal liability is a critical issue, as determining fault in cases where an AV commits a road violation or causes an accident remains an unresolved and contentious matter. Clearly there are a lot of challenges in the field of Avs.

1.1.3 Context

The town of Quackston [2], a rubber duck community, is embracing the future of transportation by adopting autonomous taxis. To support this initiative, Quackston has invited multiple companies to test their AVs on its roads. Our company, The Quaxi Company, is proposing a plan for our AV, WhatsUpDuck, which aims to revolutionize transportation in Quackston. One of our primary goals is to efficiently transport ducks to their destinations while maximizing cash flow and maintaining a strong reputation for safety and reliability. Additionally, we are committed to developing a Level 4 SAE AV tailored for Quackston's specific conditions.

1.2 Problem Statement

The current state of fully AVs in 2025 face challenges in safely navigating complex urban environments while complying with traffic laws and maintaining passenger safety. Furthermore, existing AVs lack accessibility features, which causes barriers for passengers with varying physical abilities and lack efficiency in passenger loading and unloading. This project aims to address these gaps by developing an autonomous vehicle system that integrates technologies, such as computer vision and machine learning, to enable safe urban navigation while incorporating innovative accessibility solutions to promote inclusivity and efficiency.

1.3 Objectives

In the project proposal there were three primary objectives our group set for this project:

1. Develop an autonomous vehicle capable of safely navigating through urban environments while adhering to all road rules.
2. Design a duck lift to accommodate passengers with varying levels of physical ability, thereby enhancing accessibility and significantly reducing loading times.
3. Place in the top 10 in the competition at the end of the semester.

1.4 Scope

The boundaries are being defined for the project to ensure it aligns with the competition guidelines while focusing on the teams' goals for our autonomous vehicle system. The project

will aim to achieve SAE Level 3/4 self-driving capabilities, enabling the AV to navigate urban environments with minimal human intervention while following the road rules. The AV will locate pickup points and perform optimal path planning to that point. Additionally, the AV will detect and respond to lane lines, ducks, other vehicles, and traffic signs, and dynamically update its route based on real-time conditions. These features will ensure that the vehicle meets both competition requirements and the teams own design objectives.

At the same time, there are limits intentionally set on what the project will address to maintain feasibility. The project will not address loading duck passengers onto the lowered lifting platform, it will be done by a team member, as this falls outside the autonomous functionality we aim to achieve. We will not implement a fully end-to-end AI system; instead, a hybrid approach combining AI with hardcoded logic will be used for driving. Finally, optimizing the cost of the vehicle will not be a focus, the teams priority is to develop a fully functional prototype that meets performance requirements and not cost constraints. By defining these boundaries, we can concentrate our efforts on creating a reliable AV within the scope of the competition

1.5 Structure of Report

The report is structured into multiple sections reflecting a systematic approach to engineering design. It includes system design, methodology, implementation, ethical decision making, and results, each addressing individual aspects of the project's development and execution. The discussion section provides critical analysis of the outcomes, challenges encountered, lessons learned, and potential societal and environmental impacts. Finally, the report concludes with reflections, suggestions for future improvements, and a summary of key insights. References, and appendices are included to cover everything needed.

2) System Design

2.1 Performance Requirements

2.1.1 Software Requirements

The original software requirements stated that the software stack must use less than 1GB of RAM due to limitations of the Raspberry Pi. Additionally, the software must be able to recognize and react to non-emergency situations in less than 2 seconds and emergency situations in less than 500 milliseconds.

These requirements were later revised when the hardware used was expanded to include a Jetson Nano and an Intel Real-Sense. With the additional hardware, the software stack could be split between the Pi and Jetson. The Jetson Nano added an additional 4GB of RAM on top of the Pi's existing 4GB effectively doubling the system's memory. The addition of the Jetson also provided another quad core processor and a GPU, greatly lessening the optimization requirements for the software. This did come with the trade-off of additional power consumption which introduced a brown-out issue, which will be addressed in a later section.

2.1.2 Detection Requirements

The original detection requirements were designed assuming an AI vision recognition system. To ensure safety, it was set out that the system must be able to detect objects and road signs with an accuracy greater than 90% and within 250 milliseconds of them appearing.

Due to the introduction of an Intel RealSense camera, lidar enabled object detection was possible. This method of object detection relies on hardcoded algorithms instead of an AI based solution. Thus, the original requirements no longer apply to object detection.

2.1.3 Driving Requirements

After defining the requirements for software and detection, the overall driving performance was considered. To ensure safe operation, the vehicle must slow to 20% or less of its maximum

speed when approaching hazard zones such as stop signs, merge lanes, and intersections. The vehicle must always stay within the borders of the lane, with less than a 20% deviation from the centerline. Additionally, when entering intersections, the vehicle must stop on or before the intersection stop line. When stopping or reducing speed, the brake lights must be illuminated. Finally, when following other cars, a minimum separation distance of one full car length must be maintained.

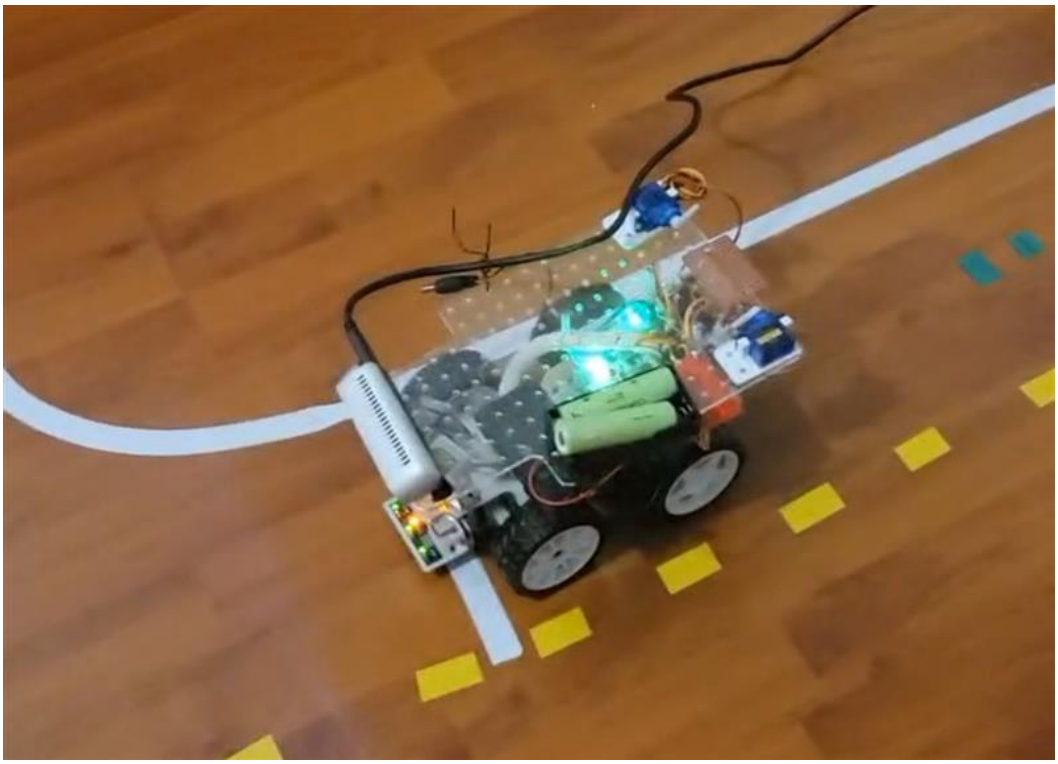


Figure 1: Vehicle stopping on the line at an intersection

2.1.4 Duck Lift Requirements

The duck lift system shall complete its entire operation within the vehicle's 10-second stationary period at either pickup or destination points for senior ducks or ducks with disabilities. The lift timing shall be allocated as follows:

- Duck Placement: 4 seconds shall be allocated for safely and comfortably placing the duck onto the lift platform.
- Platform Movement: A maximum of 6 seconds shall be allocated for lift movement, divided into two sections. The first is 4 seconds for platform movement when carrying a duck

(slower operation for safety considerations) and the second is 2 seconds for the platform to move without a duck (faster operation when duck is not present).

Success will be measured by the lift consistently meeting these timing constraints within 0.5 seconds each stage while ensuring safety standards for the duck passengers.

2.2 Team-Specific Design Criteria

Several key criteria influenced the team's design decisions, primarily performance, ethics, safety, and maintainability.

2.2.1 Simplicity

When considering two design choices, simplicity was considered above all else. If a design is simplistic, it is easy to implement, maintain, and modify. Additionally, simple designs with few inputs are less prone to failure than designs with high complexity.

To meet this design criteria, the code was written using ROS, a robotics ecosystem with a wide variety of prebuilt packages and messages. The use of ROS allows for rapid prototyping and integration through a standardization of program communication.

2.2.2 Ethics

Equally important to simplicity was emphasizing passenger safety and accessibility for senior and disabled ducks. This consideration drove the development of the Duck Lift System. This system was designed explicitly to accommodate passengers with varying abilities, supporting the ethical commitment to inclusivity and fairness. This criterion was fully achieved, enhancing the overall usability and ethical integrity of the solution.

2.2.3 Passenger Experience & Safety

Safety considerations shaped every aspect in the design from the structural stability of the duck lift to the collision avoidance functionality and movement control. Passenger safety was improved through the integration of infrared collision detection and stopping protocols. A smooth steering algorithm was implemented to ensure passenger experience. The final design

ensures robust passenger safety and reliable obstacle avoidance. This safety criterion was consistently achieved and verified through testing.

2.2.4 Modularity & Testability

Modularity and testability were considered as the final design criterion, emphasizing ease of future modification, debugging, and expansion. Using ROS to develop all the system software ensured that code could easily and quickly be tested and verified. Furthermore, ROS also allows Hardware In The Loop (HILT) testing, allowing for code to easily be switched for alternate versions.

2.3 Functional Overview

The vehicle control system design has two computing units: a Raspberry Pi and an NVIDIA Jetson.

The Raspberry Pi acts as the main vehicle controller, managing driving behavior through a state machine. This controller receives commands from the router and transitions between various states, including Driving, Waiting, Blocked, Turn Left, Turn Right, and Go Straight. During the Driving state, the Pi continuously executes line-following routines using real-time image data from a downward-facing camera positioned to the right. The line detection algorithm converts images to grayscale, applies thresholding, employs Canny edge detection, and utilizes the Hough transform to identify two parallel lane lines along the x-direction. The algorithm calculates the center position between these lines, providing reliable lane-tracking data. A PID controller combined with a smoothing filter generates precise and stable steering commands based on this detected line position.

The Raspberry Pi handles collision avoidance via dedicated collision callbacks triggered by ultrasonic sensors and data from the Jetson's infrared object detection system. If an obstacle is detected within a predefined threshold, the system transitions to the Blocked state, stopping the car immediately. Normal driving resumes only after the obstacle clears. Furthermore, the Pi interfaces directly with the PiCar hardware through custom-developed C++ software integrated

as a ROS node. This software node listens for drive power and turn angle commands and publishes relevant sensor data from the PiCar board.

The NVIDIA Jetson unit manages obstacle detection using an infrared camera capable of identifying obstacles on the road surface, with closer objects appearing brighter. The Jetson's image processing pipeline includes a blur filter, thresholding, and Canny edge detection to determine the size and location of obstacles accurately. Detected objects are communicated as vectors via ROS to the Raspberry Pi, informing collision avoidance decisions.

The NVIDIA Jetson unit would have also run the route navigation system which would provide commands to the vehicle when in the Waiting state of whether to Turn Left, Turn Right, or Go Straight, however more work had to be done to complete this integration. The navigation system, implemented in python to interface with the VPFS, stores a map of the road network and uses the Floyd-Warshall Algorithm [8] to determine an optimal route for the car to follow to reach a destination. When vehicle is at an intersection, to determine the command, Turn Left/Right or Go straight, the navigation system would be provided with the current road segment the vehicle is on from the VFPS, and the next road segment to take from the computed route; then the angle between the road segments is calculated to determine the direction to proceed.

Both vehicle control units collaborate through ROS, exchanging real time sensor data, control commands, and obstacle information. This structured division of tasks ensures efficient utilization of resources, maintaining performance standards such as timely obstacle detection and responsive lane tracking, ultimately providing safe and effective system.

2.4 System Architecture

Below, Figure 2 depicts the power distribution for our vehicle. Due to the addition of an NVIDIA Jetson Nano, a custom 3S Li-Ion battery pack was added along with a buck converter module. This improved power system was needed to provide enough current to run our motors harder and to supply the necessary current at 5V to the Jetson.

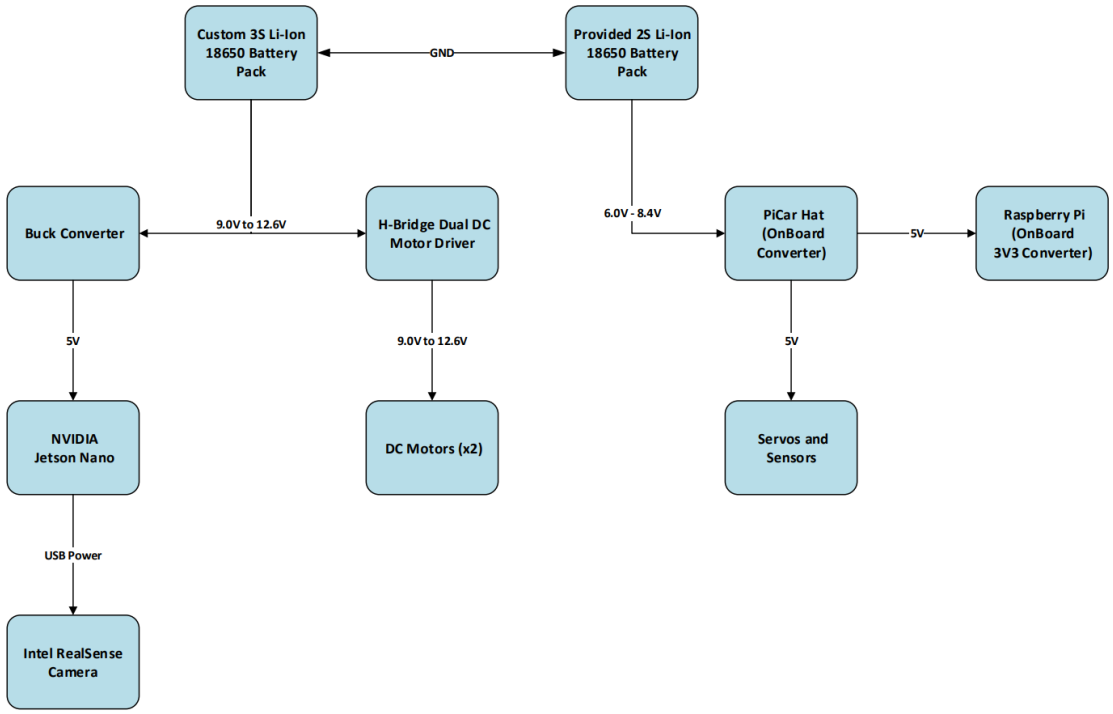


Figure 2: Vehicle Power Distribution Diagram

Below, Figure 3 shows the hardware connections in the vehicle, depicting which hardware modules are directly connected to each other and what protocols each of the connections are using.

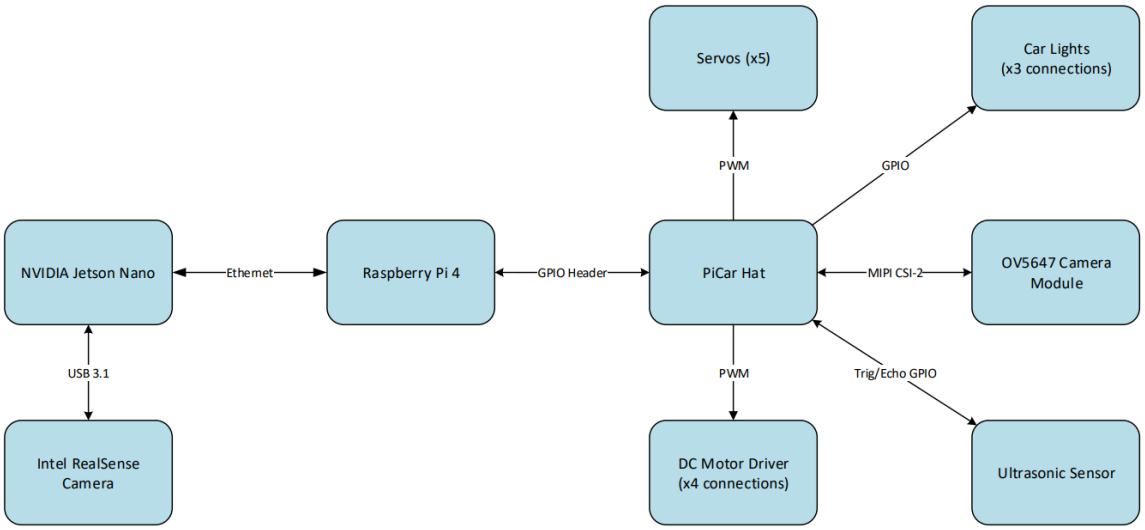


Figure 3: Vehicle Hardware Connection Diagram

Figure 4 depicts the interactions between ROS nodes (different pieces of software) while operating on the car. It is worth noting that planning and routing nodes are not depicted in this diagram due to the issues that arose during the semester that interfered with the in lab testing on the track which prevented routing to be fully integrated with the rest of the software system.

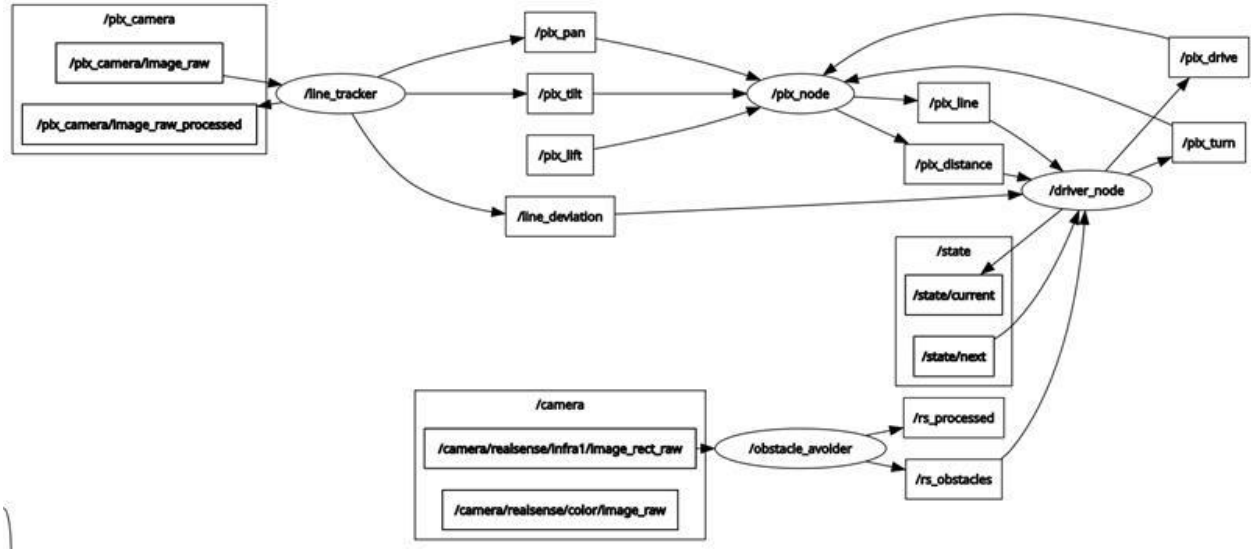


Figure 4: Vehicle ROS node diagram

Despite the lack of testing with routing and planning, the functionality was still developed, however it was not integrated into vehicle control system. The data pipeline for initializing and building the routing system is described by Figure 5 below:

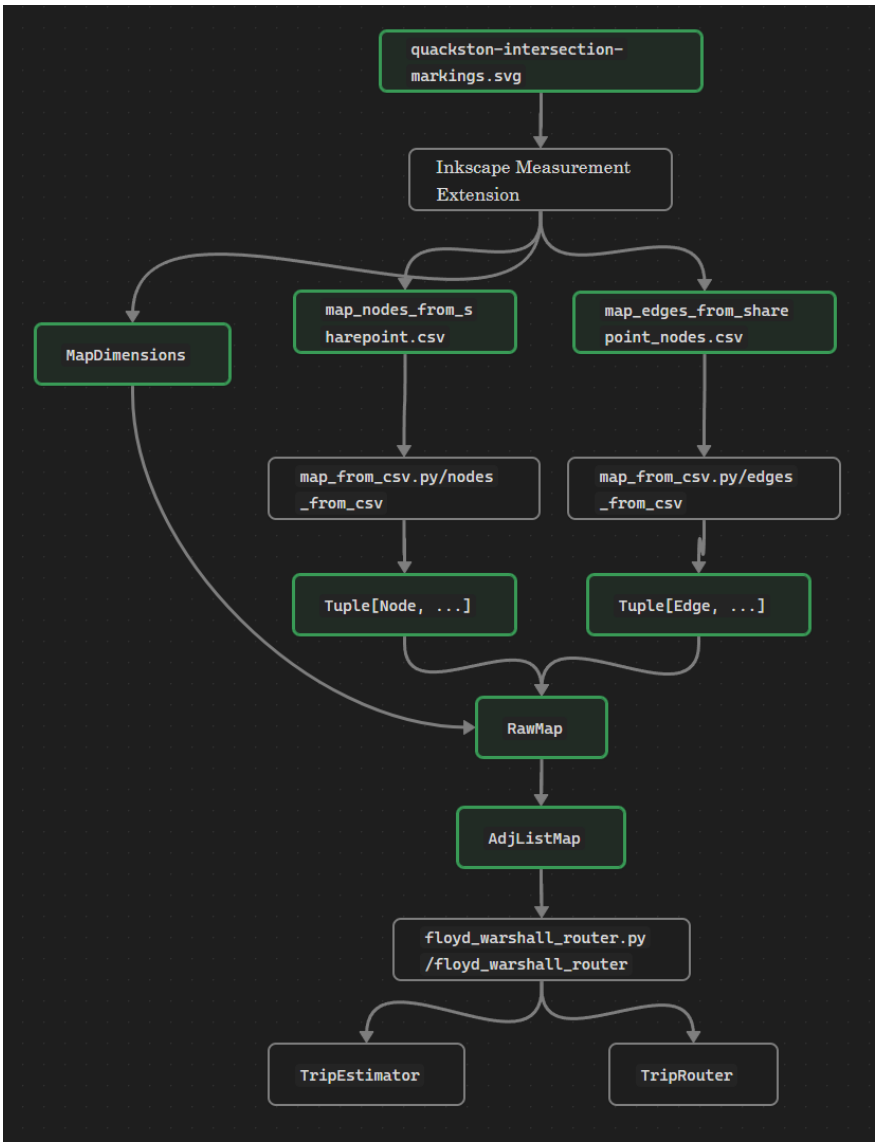


Figure 5: Routing System Build Pipeline. Grey boxes represent functions, green boxes represent data. The core input to this routing system is the svg file "quackston-intersection-markings.svg" which details the road network. That file is analyzed using Inkscape (https://inkscape.org/) to produce road measurements and intersection locations stored in csv files. Those measurements are then analyzed using Python functions to construct the map data structure in a format suitable for applying the Floyd-Warshall algorithm.

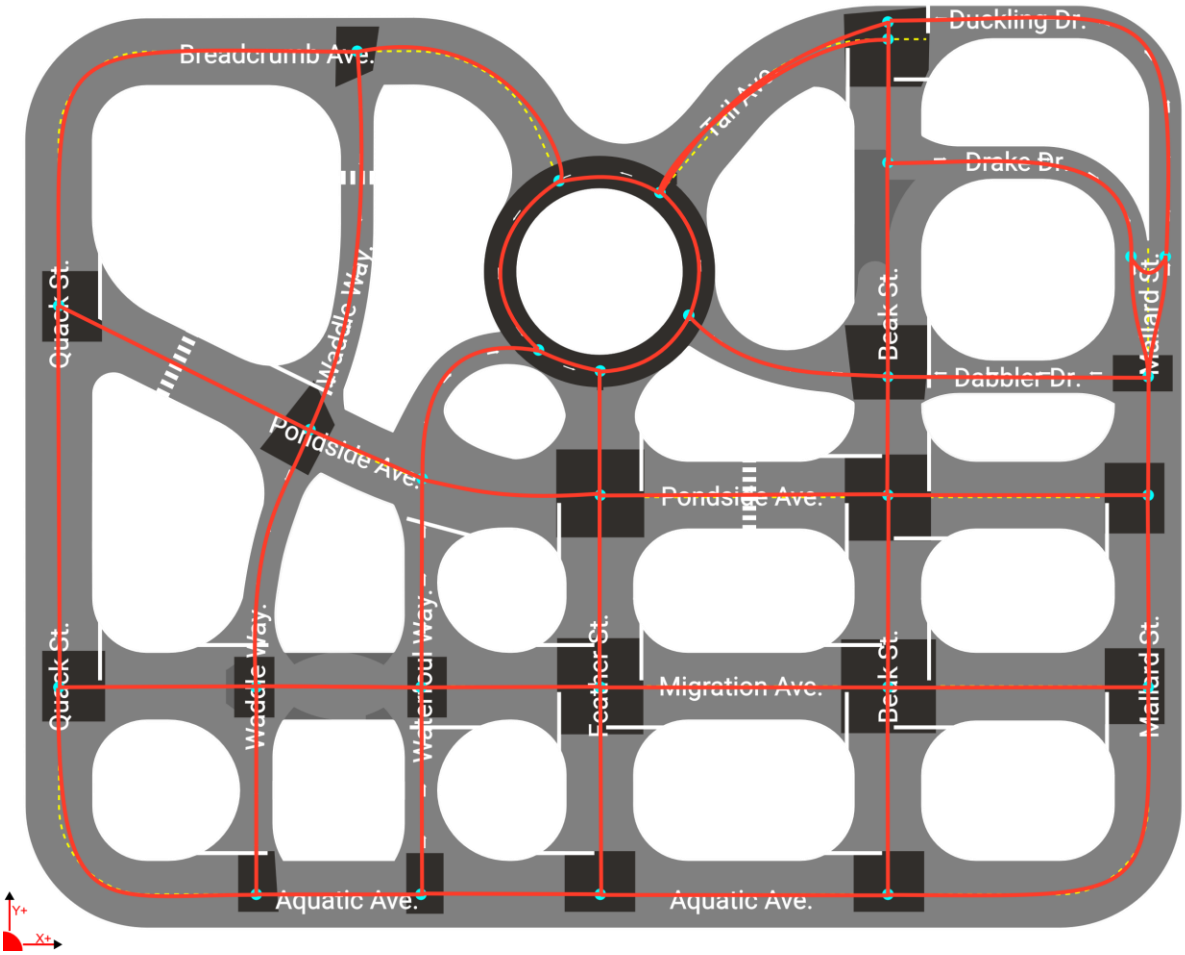


Figure 6: image for the labeled map file "quackson-intersection-markings.svg" with the road paths labeled denoted by red lines and the intersections labeled with turquoise dots. This is the key input to the routing system initialization function depicted in Figure 5.

The key outputs from building the system are the TripEstimator and TripRouter functions. These functions are used as core components in the active routing system in the following two figures. If there were multiple possible trips to choose from, the routing and navigation system would need to provide information about the estimated distance of those trips so that a decision can be made as to which trip to take. Therefore, before the routing system may compute the route for a particular trip, it may need to compute its distance to be compared, this is the use of a TripEstimator (see Figure 7).

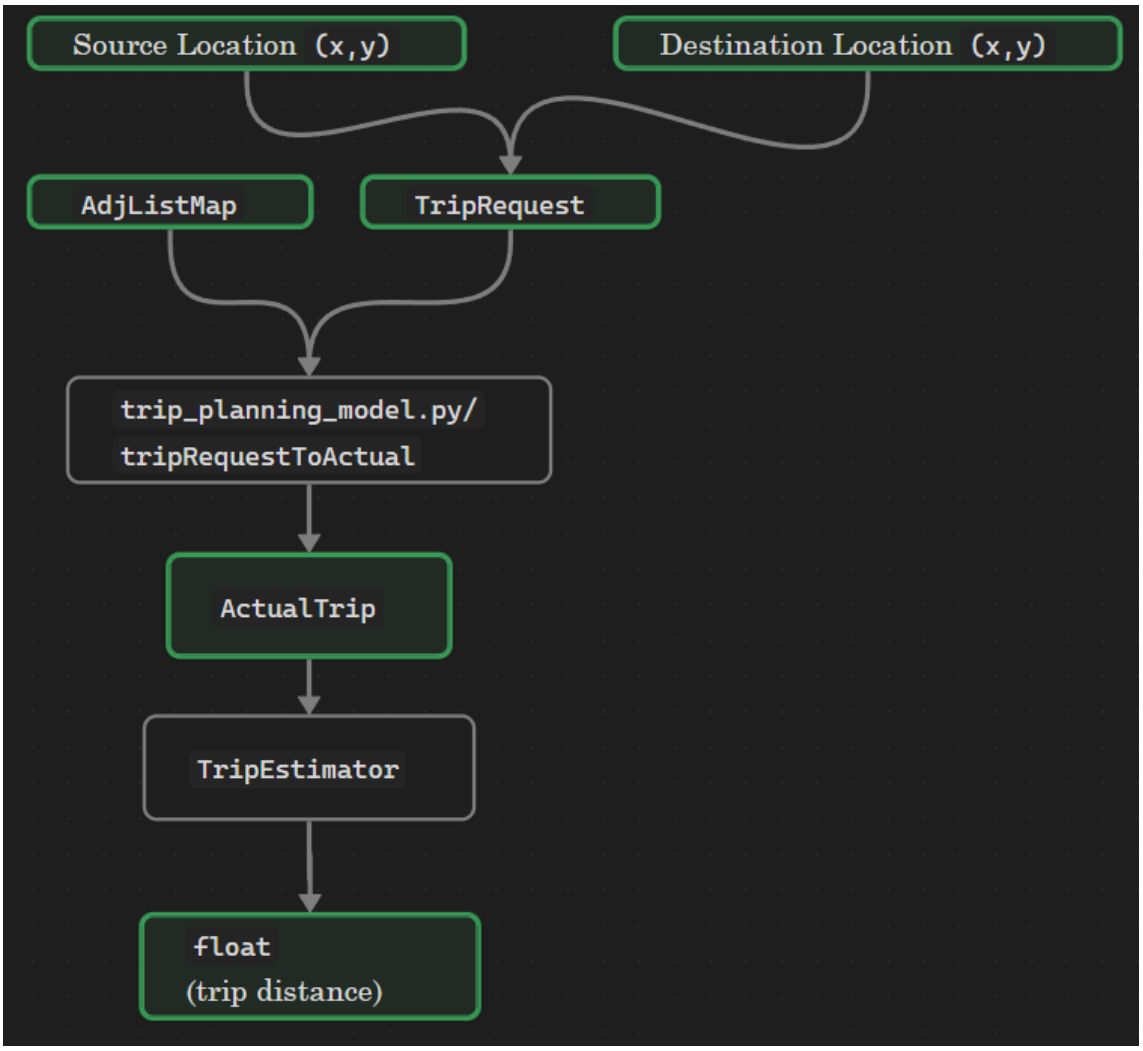


Figure 7: Use of a TripEstimator function to calculate trip distance. The required inputs here are the source and destination locations, the AdjListMap would have been generated during routing system initialization. The float distance for the trip estimate is in the same units of the MapDimensions originally used to specify the AdjListMap. In the case of Quackston, the units are in meters.

Once a particular trip is selected, a TripRouter can be used to calculate the direction commands for the vehicle control system (see Figure 8).

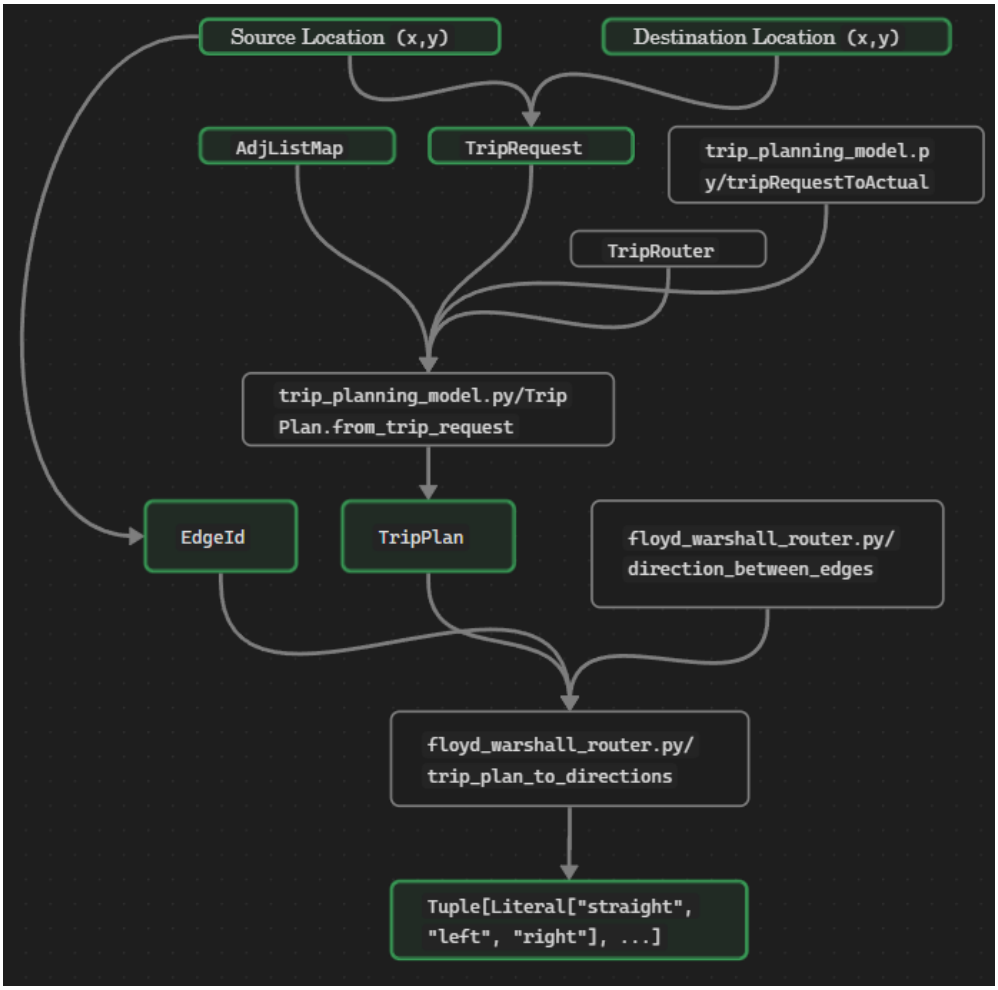


Figure 8: Data flow diagram making use of the TripRouter component of the routing system. The real-time inputs to the system are the source and destination locations which make up the TripRequest. The cached inputs from the Build Pipeline are the TripRouter and AdjListMap. The ultimate output of this stage of the system is the tuple of direction commands seen at the bottom of the figure.

3) Methodology

3.1 Technologies and Tools

Our autonomous driving solution utilized the components detailed in Table 1.

Table 1: Technologies and tools used

Category	**Tool/Technology**	**Description/Use**
Hardware	Raspberry Pi	Provided
	Jetson Nano	Used for object detection

	Intel RealSense D435	Used for object detection
	PiCar-X	Provided
	External Batteries	Used for boosting motors
	Servos	Used for duck lift
	LEDs	Provided
	LPFK S104 PCB	Used to make PCBs
	Creality K1 3D Printer	Used for 3d printing
	L298 Motor driver	Used to drive motors at higher voltage
Software	Python	Programing language
	CPP	Programing language
	Linux I2C headers	Interfacing
	GPIOD/pigpio	Interfacing
	OpenCV	Computer vision library
	KiCAD	Used to design PCB
	Bambu Studio	Used to slice 3D models
	Fusion 360	Used to create 3D models

3.2 Algorithms

Obstacle detection was accomplished using the Intel Real-Sense camera. This camera features an infrared (IR) sensor capable of measuring the intensity of a reflected infrared beam. Closer objects appear brighter in the IR image meaning that bright spots in the image would indicate nearby obstacles. Through the application of a blur filter, thresholding and Canny's edge [9] detection, obstacles on the road can be easily seen as in the below image.

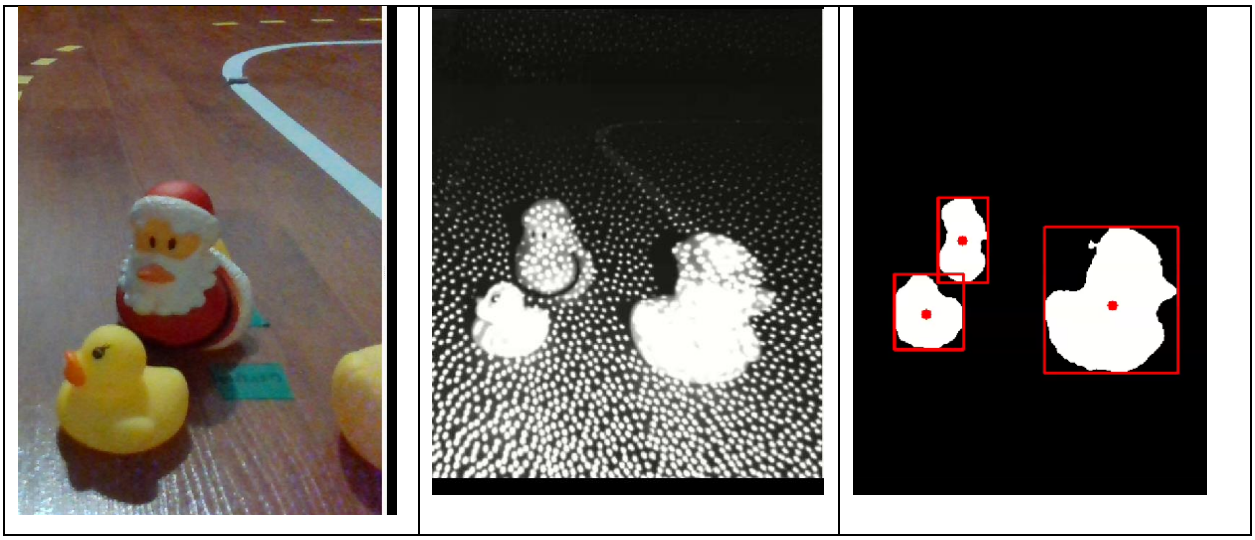


Figure 9: Obstacle detection demonstration. Left: color image that is cropped on the right due to limited field of view. Middle: IR image with wider field of view. Right: Obstacle detection bounding boxes output. permanent link to the code implementation for this algorithm on the team's git repository: https://github.com/hendrixgq/ELEC390/blob/95f626fe72dfd44d0633c10421380504515bb172/ros2_ws/src/vision/duck_avoidance/src/obsticle_avoider.cpp#L105.

Line detection was accomplished using Canny edge detection along with Hough's transform [10] to detect the lane line. See Figure 10, shown below. Using the information from the line detection algorithm, a PID controller was used alongside motion smoothing to calculate and apply a correctional output to the steering.

Figure 10: Lane line detection example. Left: poorly lit color image of the road lane line. Right: processed image extracting the direction of the lane line. Here is a permanent link to the implementation of this algorithm on the team's git repository: https://github.com/hendrixgq/ELEC390/blob/95f626fe72dfd44d0633c10421380504515bb172/ros2_ws/src/vision/roadline_tracker/src/linetracker.cpp#L39.

For the routing and navigation system, the Floyd-Warshall algorithm [8] was chosen for its simplicity and efficiency on the relatively small and static road network. This algorithm along

with path reconstruction were implemented in python in the team's git repository [11] and snippets of the code are show in Figure 1111 and Figure 1212 below.

```
16          # Pre-compute the shortest paths between all pairs of nodes in the map
17          n = len(map.adjs)
18          # dist[i][j] is to be the length of the shortest path from i to j
19          dist = [[float('inf')]*n for _ in range(n)]
20          # pred[i][j] is to be the predecessor edge_id of j on the shortest path from i to j
21          pred = [[None]*n for _ in range(n)]
22          # initialize paths of length 0
23          for i in range(n):
24              dist[i][i] = 0
25          # initialize paths of length 1
26          for src, adj in enumerate(map.adjs):
27              for edge_id in adj:
28                  edge = map.raw.edges[edge_id]
29                  dest = edge.dest
30                  if dist[src][dest] > edge.length:
31                      dist[src][dest] = edge.length
32                      pred[src][dest] = edge_id
33          # compute all other shortest paths
34          for k in range(n):
35              for i in range(n):
36                  for j in range(n):
37                      if dist[i][j] > dist[i][k] + dist[k][j]:
38                          dist[i][j] = dist[i][k] + dist[k][j]
39                          pred[i][j] = pred[k][j]
40          # check for negative cycles
41          for i in range(n):
42              if dist[i][i] < 0:
43                  raise ValueError("map contains a negative cycle")
```

Figure 11: Floyd-Warshall Algorithm distance matrix calculation code. As you can see, the definition of the predecessor matrix here is different in that it stores edges instead of vertices, this was done so that the information about specific edges could be easily accessed during path reconstruction and resolves any ambiguity in the case where there are multiple parallel edges. Permanent link to code on github: https://github.com/hendrixgq/ELEC390/blob/95f626fe72dfd44d0633c10421380504515bb172/Design/mapping/floyd_warshall_router.py#L8.

```
48        def reconstruct_path(pred: Sequence[Sequence[EdgeId]], src: NodeId, dest: NodeId) -> Tuple[EdgeId, ...]:
49            reverse_path = []
50            while src != dest:
51                edge_id = pred[src][dest]
52                if edge_id is None:
53                    raise ValueError(f"No path from {src} to {dest}")
54                reverse_path.append(edge_id)
55                dest = map.raw.edges[edge_id].src
56            return tuple(reversed(reverse_path))
```

Figure 12: Floyd-Warshall Algorithm path reconstruction function (docstring comment removed for clarity). Permanent link to code on team's git repository: https://github.com/hendrixgg/ELEC390/blob/95f626fe72dfd44d0633c10421380504515bb172/Design/mapping/floyd_warshall_router.py#L48

3.3 Training

Our approach did not involve training traditional machine learning models, such as neural networks. Instead, we adopted an infrared-based obstacle detection method using the Intel RealSense D435 camera [12]. By relying on infrared imaging and classical image processing, we significantly improved detection accuracy under diverse lighting conditions without the complexity or uncertainty associated with training datasets. This method allowed us to achieve consistent and reliable obstacle detection performance, effectively eliminating the need for conventional model training, validation, and testing processes.

4) Implementation

4.1 System Integration

Using ROS, C++ and Python programs were easily integrated through the ROS message passing API. All software components were clearly defined early in the design process, further simplifying the system integration through standardization. It is the result of these design choices that the only software challenge was interfacing with the PiCarX Hat.

Early in testing, several problems with the PiCarX sample code were identified including improper constants, arithmetic errors, and control limitations. To overcome these challenges,

the decision was made to reverse engineer the PiCarX Hat. This decision allowed the interface code to be rewritten in C++, providing better responses and controllability.

Hardware integration presented various challenges including improper motor driver sizing and brownouts. To resolve the issues with the onboard motor drivers, an external motor driver was added, allowing the motors to run at full power. To resolve the brownouts caused by the Nvidia Jetson, a secondary power supply and buck converter were added to the design.

4.2 Development Timeline

The original development schedule for the project as submitted for the project proposal is shown in Table 2 below. The “Actual Completion” column is added to the right to list the date of actual completion. See Table 3 which is organized chronologically “Actual Completion” excluding incomplete milestones can be found.

Table 2: Original Project Timeline from project proposal report with actual completion dates.

#	Milestone	Target Date	Actual Completion
1	PiCar can be Remote Controlled and Record Key Metrics	2025-02-01	2025-03-22
2	Basic Autonomous Drive Controller Works	2025-02-02	2025-03-24
3	Data Sets Obtained for Training Object Detection	2025-02-09	Alternative Method Chosen
4	PiCar Can Drive Autonomously Along a Lane	2025-02-09	2025-03-23
5	Road Map Integrated into Planning System	2025-02-19	2025-02-27
6	Route Planning System Done	2025-02-19	2025-02-27
7	Path Planning Based on Route Plan Works	2025-02-19	2025-03-24
8	Object Detection Works	2025-02-19	2025-03-23
9	Each Capability is Integrated as a Cohesive System within the PiCar	2025-02-19	Not Completed
10	PiCar Can Drive Autonomously Through an Intersection	2025-02-15	2025-03-24

11	PiCar Can Drive Autonomously to any Destination in Quackston	2025-03-01	Not Completed
12	Optimized Ride Selection Algorithm Complete	2025-03-08	Not Completed
13	PiCar Duck Lift Cabin and Restraint Equipment System Complete	2025-03-09	2025-03-06
14	Begin Writing Final Report	2025-03-10	2025-03-30
15	Vehicle Tested in Competition Setting	2025-03-16	Not Completed
16	Compete in Final Competition	2025-03-24	Not Completed
17	Final Project Report Complete	2025-04-04	2025-04-06

Table 3: Actual Project Timeline.

#	Milestone	Actual Completion
4	PiCar Can Drive Autonomously Along a Lane	2025-03-23
5	Road Map Integrated into Planning System	2025-02-27
6	Route Planning System Done	2025-02-27
13	PiCar Duck Lift Cabin and Restraint Equipment System Complete	2025-03-06
1	PiCar can be Remote Controlled and Record Key Metrics	2025-03-22
4	PiCar Can Drive Autonomously Along a Lane	2025-03-23
8	Object Detection Works	2025-03-23
7	Path Planning Based on Route Plan Works	2025-03-24
10	PiCar Can Drive Autonomously Through an Intersection	2025-03-24
14	Begin Writing Final Report	2025-03-30
17	Final Project Report Complete	2025-04-06

The main reason for the vehicle not being tested in the competition setting was due to the late completion of the autonomous driving capabilities and the breakdown of the vehicle networking capability with the raspberry pi not being able to connect to local WIFI at the

competition location. Also, not integrating the navigation system and the VPFS into the final system was the cause for milestones 9 and 11 not being completed.

5) Ethical Decision Making and Considerations

The ethical framework guiding decision-making for The Quaxi Company draws from Deontological Ethics, Virtue Ethics, and AI Ethics Principles. Each ethical view played a part in shaping our approach to autonomous decisions such as fare selection and autonomous driving.

Deontology is a branch of philosophy that focuses on the morality of an action. In deontological ethics, the moral worth of an action is determined by how it adheres to these principles and duties, rather than by its consequences [13]. The Quaxi Company used Deontological ethics to define our duties to customers. The company laid out some guidelines for what was wanted to accomplish which were to consider all customers fairly including seniors, disabled and regular riders regardless. A challenge that resulted from this decision was the trade-off between ethical fairness and business efficiency, as for example, providing equal access to all passengers has a reduced potential revenue per trip for seniors and disabled users but this was accepted to uphold fairness and inclusivity.

Virtue ethics also played a role when considering fares encouraging fairness and transparency. The Quaxi Company wanted to be open about our decision making with the aim to build trust with our customers through fairness and inclusivity. The challenge with this was that the complexity of the system made full transparency difficult to explain to all users regardless of technical background.

In autonomous driving, AI Ethics Principles were followed as many ethical challenges were presented. For scenarios such as possible collisions, our company prioritized passenger safety. Moving obstacle avoidance was implemented and tested, the car can slow down and stop immediately upon detection. This fully aligns with ideal AI ethics, which includes active avoidance, however it does not have the ability to give the right of decision to the passenger as the passenger is an inanimate object.

Overall, the decisions made were consistent with our original ethical framework, but challenges were introduced. Due to the time constraint of the project limiting the technical scope the ethical goals originally laid out couldn't all fully be met. The Quaxi Company still believes that our system aligned with our core principles of fairness, transparency, and harm reduction regardless of the trade-offs and challenges faced.

6) Results

There are both qualitative and quantitative results and performance metrics for the outcomes of the project. The quantitative performance metrics were (1) obstacle detection rate %, (2) obstacle detection time required, (3) maximum vehicle deviation from lane center in straight and (4) curved lanes. The qualitative performance metrics were (5) brake lights turn on when stopping, (6) vehicle navigation system produces direction commands, (7) vehicle can drive autonomously to any destination in Quackston, (8) Fares can be completed, (9) minimum reputation score of 80%, (10) rank in the top 5% of teams in the competition, (11) vehicle stops at intersections, (12) vehicle position information communicated via VPFS.

The results are summarized in the following list:

1. Obstacle detection rate 100%
2. Obstacle detection time required 100 milliseconds
3. Vehicle deviation from lane center in straight lanes 4% (goal 20% or less)
4. Vehicle deviation from lane center in curved lanes 28% (goal 20% or less)
5. Brake lights turn on when stopping – yes
6. Vehicle navigation system provides vehicle directions – yes
7. Vehicle can drive autonomously to any destination in Quackston – no
8. Fares can be completed – no
9. Minimum reputation score of 80% - no reputation score achieved
10. Rank in the top 5% of teams in the competition – our vehicle did not function on the Quackston map, so we were at the bottom of the list
11. Vehicle stops at intersections – yes

12. Vehicle position information obtained from VPFS – no
13. Vehicle slows down in hazard zones – no
14. Vehicle maintains a safe following distance to other vehicles – unknown

Further elaboration on the results is detailed in the following paragraphs, with details regarding the Intel Real-Sense camera for obstacle detection, lane line detection and deviation, and the competition results.

The Intel Real-Sense camera provides an infrared image, which when mixed with the use of various filtering and convolutions, allows for object detection rate of 100%. Not waiting for classification allowed us to exceed our initial requirement of 250 milliseconds and allowed for the detection of objects within 100 milliseconds.

With white electrical tape road lines in our testing setup (see Figure 1), the vehicle performed well with having a max deviation of 4% from the center line on straight paths. The vehicle also successfully stops on the line at an intersection as seen in Figure 1 and turns on the brake lights accordingly. For driving around corners the maximum deviation from the center observed was 28% which while still ensures the vehicle stays in its designated lane, is exceeding the requirement of 20% which was originally set.

Due to complications preventing in-lab testing and electronic communication issues at competition, there were multiple original driving requirements that were not met. Specifically, the position related requirements which depended on the use of the VPFS, and other vehicles were not accomplished due to lack of testing and ability to gather position data. This resulted in slowing down in hazard zones, vehicle separation and driving in unfamiliar situations such as populated roads to not be sufficiently tested and implemented.

7) Discussion

7.1 Analysis of Results and Challenges

Based on the above results, many of the goals were met for the fare collection and mapping system while nearly all the goals were met or exceeded by the vehicle control system. In attempting to meet these goals, several successes and failures occurred.

The supplied PiCarX kit and codebase had an abundance of issues. For example, the PWM constants in the datasheet were found to be wrong, the drive power and steering behavior did not exhibit the expected linear behaviour, and finally, the drive speed did not change when the power was between 20 and 100. After extensive trial and error, the given Python codebase was rewritten in C++ using the correct constants and providing the expected linear control.

When testing and implementing steering control for line following, it was found that there is a large delay between a change in output and input. This resulted in jittery steering, which often caused the car to overshoot the road line and drive off the side of the road. To solve this problem, a moving average filter was used over the PID input. Furthermore, an exponential controller was added to the PID output. Solving the problem, the result was smooth turning and a calm ride for passengers.

The final challenge faced in this project was presented at competition. During the project development, the network stack on the Raspberry Pi became corrupted. This resulted in the Raspberry Pi being unable to add new network connections. Consequently, the team was prevented from testing the vehicle at competition. Fortunately, the team was able to test the performance of the car in a different location.

A final limitation of the vehicle is that the two main components, driving and planning, were never integrated. Thus, the vehicle is currently unable to move between intersections without external input from a human. However, both were independently tested and verified, thus system correctness is guaranteed.

7.2 Lessons Learned

This project provided valuable insights across technical development, teamwork, and project management. In terms of technical skills, new platforms and libraries were explored. The importance of modularity and testability for rapid prototyping was explored through the Robotics Operating System (ROS). Furthermore, an insight into multidisciplinary design was gained through CAD programs and 3D printing.

From a teamwork perspective, clear communication and division of responsibilities were essential. Deadlines, an equally important factor, were however overlooked. As a result, the project was either massively ahead or behind the predicted schedule. Confusion in timing led to integration issues and partially met requirements. Going forward, we recognise the need for better synchronization and more consistent communication to keep everyone aligned.

In terms of project management, the above issues caused further difficulties when trying to track project progress. Much of the final project was created within small, short periods rather than spread over a long duration. For the future, the importance of regularly scheduled meetings will not be overlooked.

Finally, the use of a single, shared, physical notebook limited the team's ability to consistently record project documentation. For future projects, the team plans to employ a digital notebook, allowing instantaneous synchronization and digital timestamping.

7.3 Societal and Environmental Impacts

Autonomous vehicles/taxis have the potential to reduce traffic accidents, lower emissions if implemented with electric vehicles (EVs) and the ability to improve accessibility. However, ethical issues and societal implications arise from these vehicles. Autonomous taxis have the concern of job displacement as companies will no longer have to pay people like Uber and taxi drivers leaving people who rely on these positions to make a living without a job.

Environmentally, since these autonomous are very sensor dense it naturally makes sense to implement them as EVs which reduced the emissions due to gas vehicles. Also, once the density

of autonomous vehicles has increased to an extent algorithms can be implemented to efficiently and effectively route trips in a way to minimize time spent and environmental impacts. The traded downside to this routing is that it relies on transmitting customers locations and desired locations which can be prone to malicious intent if not implemented properly.

This project has changed the teams perception on the decisions engineers have to make in the workplace as the ethical, societal, and environmental impacts of these decisions is much greater than what they appear to be at a glance.

8) Conclusions and Reflections

8.1 Summary of Work

8.1.1 Key aspects of the project

The primary object of this project was to develop an autonomous taxi system using a PiCarX that was modified by the team and navigate the town Quackston. The project sought to address challenges associated with autonomous urban navigation, accessibility for passengers with disabilities, and maximizing profit and reputation.

The methodology employed image processing techniques using OpenCV to do lane detection, lane following, and obstacle detection and avoidance, which yielded good results, including 100% obstacle detection accuracy within 100 milliseconds. Despite setbacks, such as hardware integration issues and a corrupted Raspberry Pi network stack, the team successfully developed a functional vehicle capable of autonomously following lanes and stopping correctly at intersections. However, full integration of the routing/navigation system and VPFS was not completed, impacting the ability to autonomously navigate to specified destinations.

8.1.2 Team performance, skills and areas for improvement

Overall, the team struggled to allocate sufficient time to the project during weeks 7-12 due to conflicting commitments from other coursework. Moving forward, there is clear room for improvement through more detailed planning, better delegation of tasks, and the establishment

of internal deadlines. To better prepare for future projects, particularly the upcoming capstone, the team intends to engage in detailed planning very early in the process, proactively assigning roles, responsibilities, and internal deadlines to prevent similar challenges and ensure steady progress with room for error.

There are a few notable individual skills to highlight:

- Jacob Chisholm excelled in line following and obstacle avoidance using IR, which is outside the scope of the course.
- Luke Strickland's 3D modeling skills allowed the creation of a fully functional duck lift system that was the main driver for the teams' accessibility initiative.
- Hendrix Gryspeerdt provided significant contributions to the algorithms implemented, he has a real knack for coding.
- Matthew Szalawiga was the electrical team member bringing his skills in battery management and power electronics to the car.

8.2 Future Work

Below are some suggestions for improving this project in future years:

- **Budget Allocation for Next Year**: Since most of the budget was spent acquiring the kits this year, next year the budget could be used to allocate to each to team to develop a piece of hardware to assist the operation of the vehicle. This hardware design aspect can provide and opportunity to increase the Electrical Engineering involvement and contributions to the project.

- **VPFS Software Functions:** For the VPFS create pre-defined software functions to streamline interaction with the vehicle. The issues that were presented with trying to get the VPFS to work extended too long which inhibited teams from being able to work on and refine their AI's and algorithms that relied on VPFS interaction.

- **PiCarX Software Stack:** The provided software stack features several issued outlined in the above report. Furthermore, many teams experienced integration roadblocks. Using ROS, many of these integration challenges can be overcome with the previously mentioned C++ ROS node for the PiCarX.

8.3 Closing Remarks

In conclusion, the project could be considered a success. Due to external influence, the VPFS system was unavailable. Thus, even if the navigation and driving components were integrated, it would have had no effect on the testing of the vehicle. Considering the only evaluated component, the driving system, consisting of lane following and obstacle avoidance, the car was able to meet all the previously set goals.

This project was an important opportunity to bridge theoretical knowledge with practical application. It allowed exploration of new tools and technologies, the importance of planning and regular meetings, and finally, the ethics involved in autonomous driving.

References

[1] “Robot Resources.” Accessed: Apr. 06, 2025. [Online]. Available: https://queensuca.sharepoint.com/teams/CCT-938446/SitePages/Robot-Resources.aspx

[2] “Welcome to the Town of Quackston.” Accessed: Apr. 06, 2025. [Online]. Available: https://queensuca.sharepoint.com/teams/CCT-938446/SitePages/The-Town.aspx

[3] “Vehicle Positioning and Fare System (VPFS).” Accessed: Apr. 06, 2025. [Online]. Available: https://queensuca.sharepoint.com/teams/CCT-938446/SitePages/GPS-and-Fare-System.aspx

[4] “ELEC 390 Principles of Design & Dev. W25 - Home.” Accessed: Mar. 30, 2025. [Online]. Available: https://queensuca.sharepoint.com/teams/CCT-938446/SitePages/LearningTeamHome.aspx

[5] “Waymo - Self-Driving Cars - Autonomous Vehicles - Ride-Hail.” Accessed: Jan. 25, 2025. [Online]. Available: https://waymo.com/

[6] “Autonomous Vehicles Are Great at Driving Straight - IEEE Spectrum.” Accessed: Jan. 25, 2025. [Online]. Available: https://spectrum.ieee.org/autonomous-vehicles-great-at-straights

[7] M. Igini, “Environmental Pros and Cons of Self-Driving Cars,” Earth.Org. Accessed: Jan. 25, 2025. [Online]. Available: https://earth.org/pros-and-cons-of-self-driving-cars/

[8] “Floyd–Warshall algorithm,” *Wikipedia*. Jan. 15, 2025. Accessed: Apr. 06, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Floyd%E2%80%93Warshall_algorithm&oldid=1269555308

[9] “Canny edge detector,” *Wikipedia*. Mar. 12, 2025. Accessed: Apr. 06, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Canny_edge_detector&oldid=1280078283

[10] “Hough transform,” *Wikipedia*. Mar. 29, 2025. Accessed: Apr. 06, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hough_transform&oldid=1282956703

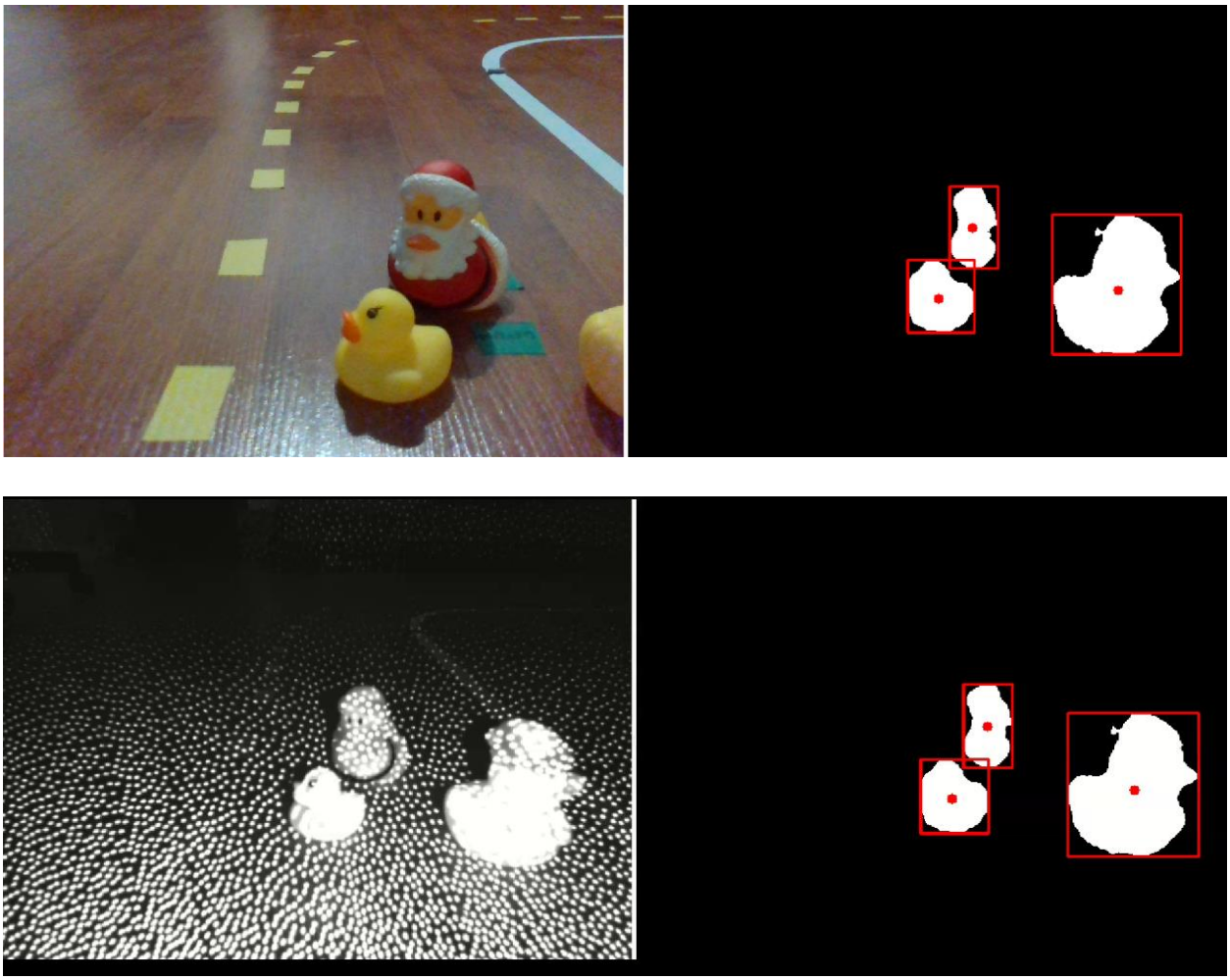
[11] “hendrixgg/ELEC390: Queen’s University 3rd year design project course on a small autonomous vehicle.” Accessed: Apr. 06, 2025. [Online]. Available: https://github.com/hendrixgg/ELEC390/tree/main

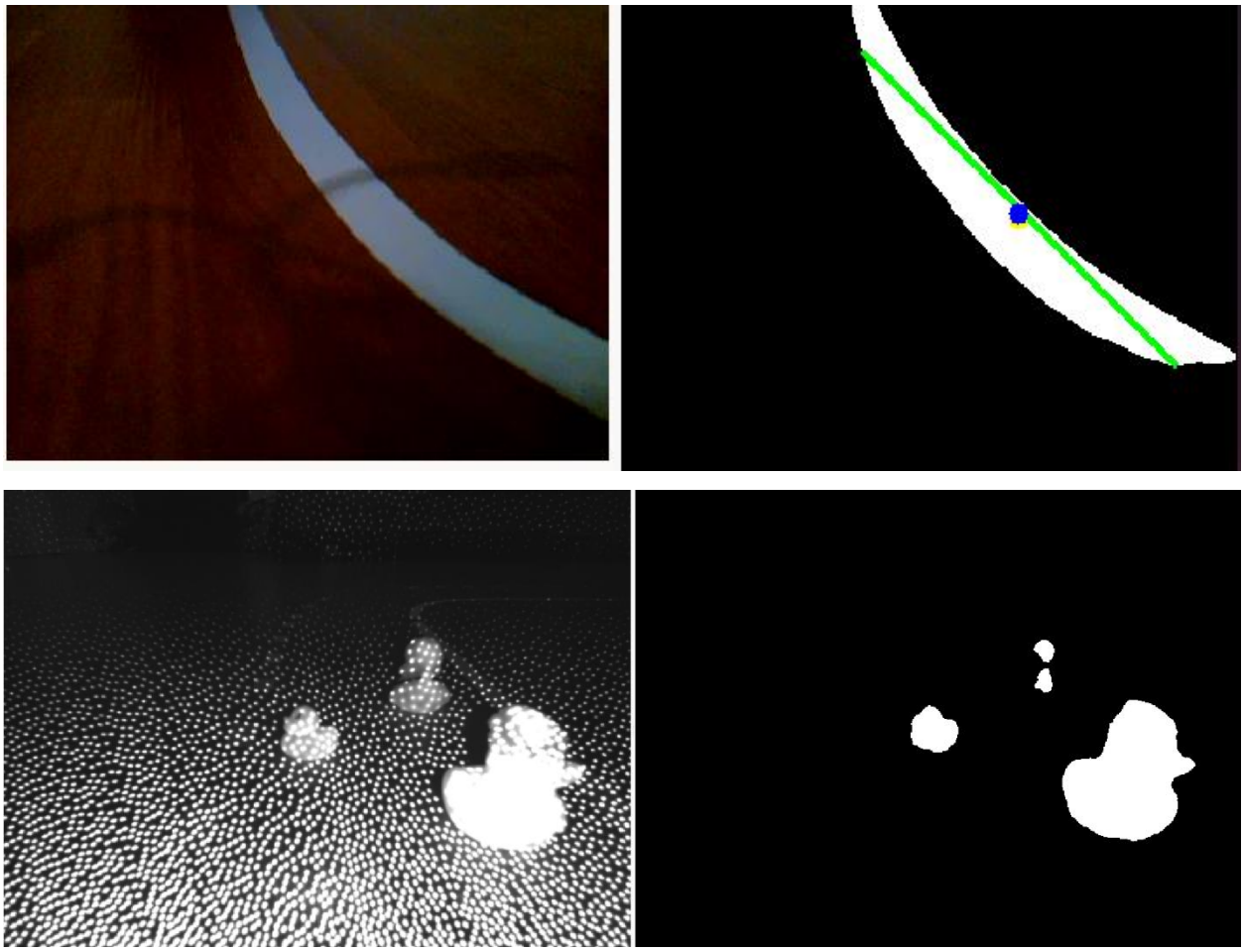
[12] “Depth Camera D435,” Intel® RealSense™ Depth and Tracking Cameras. Accessed: Apr. 07, 2025. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/

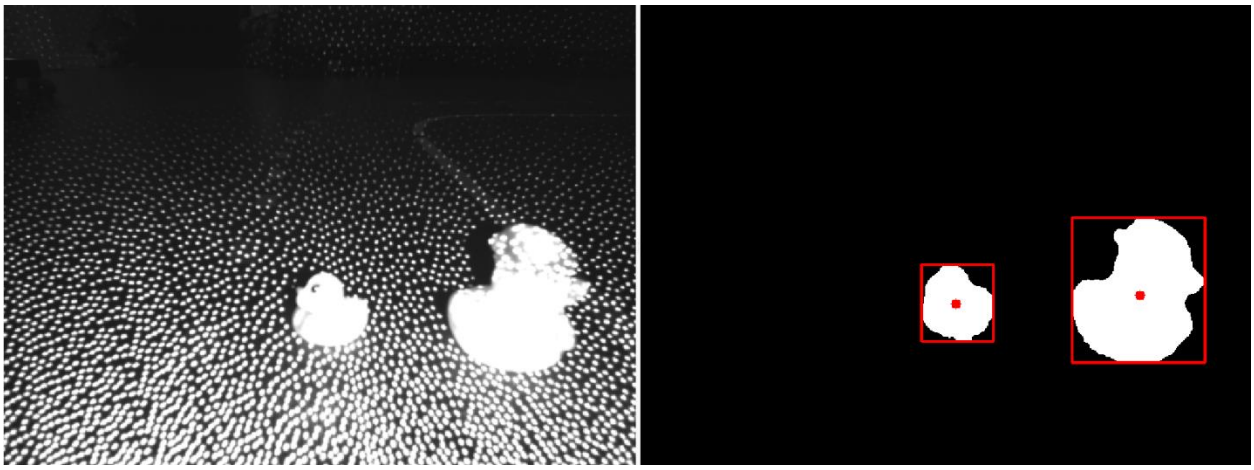
[13] S. Rueter, “Understanding Deontology: Ethics and Principles.” Accessed: Apr. 01, 2025. [Online]. Available: https://www.philosophos.org/ethics-deontology

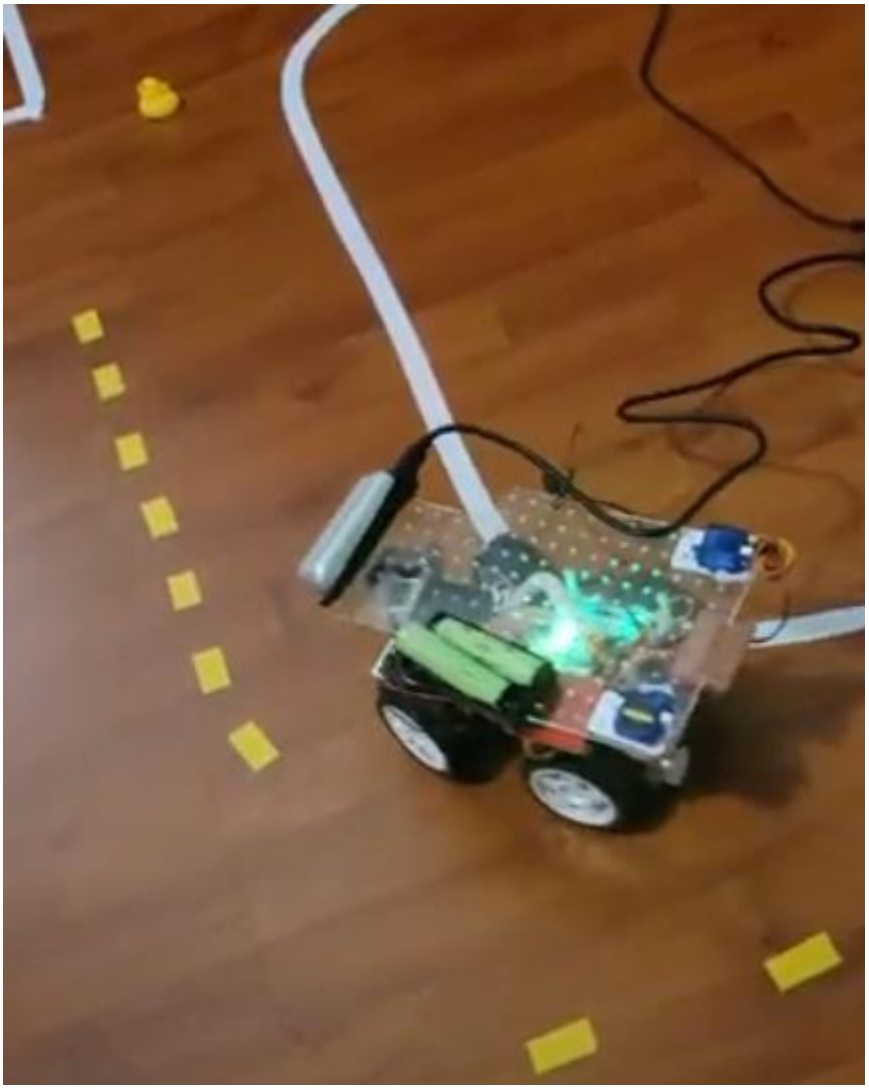
Appendix

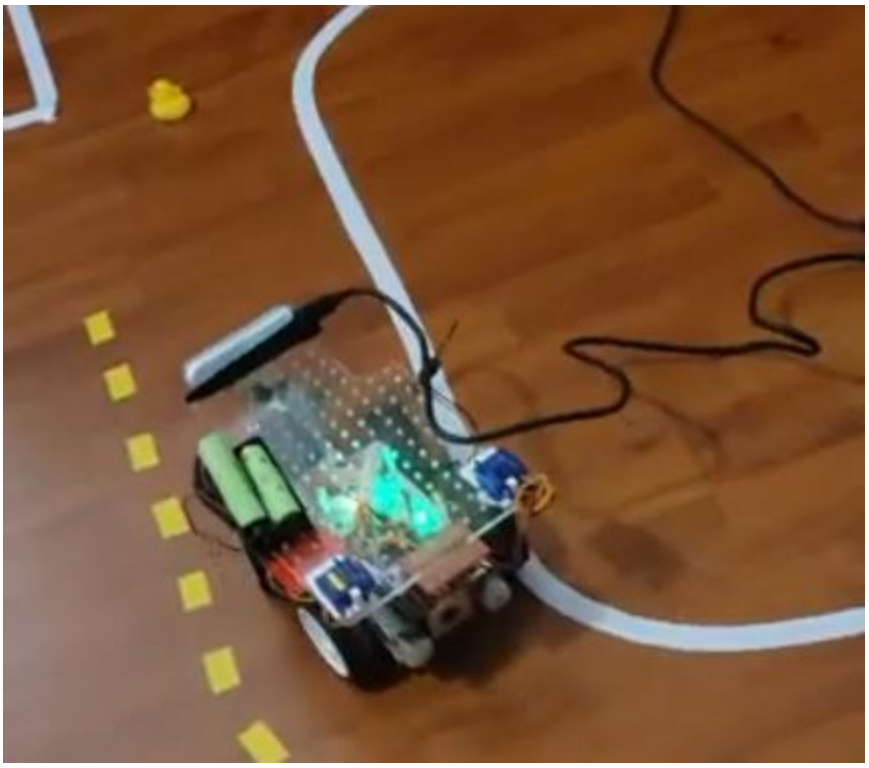
Below are several images testing the car. To see a video of the car working, please visit the following link: https://jchisholm204.github.io/posts/elec390/

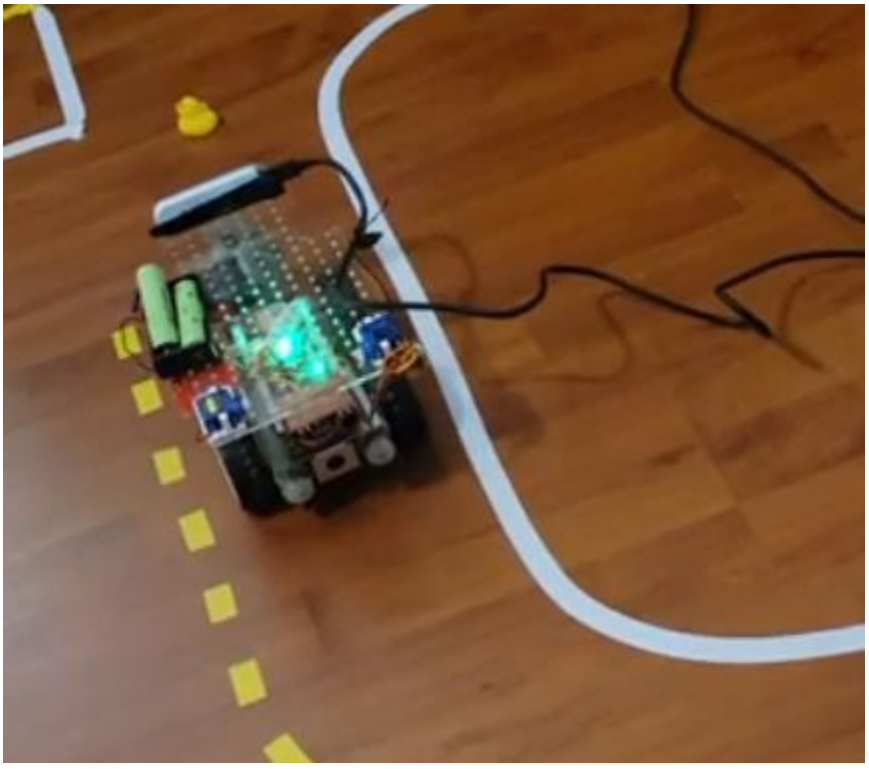


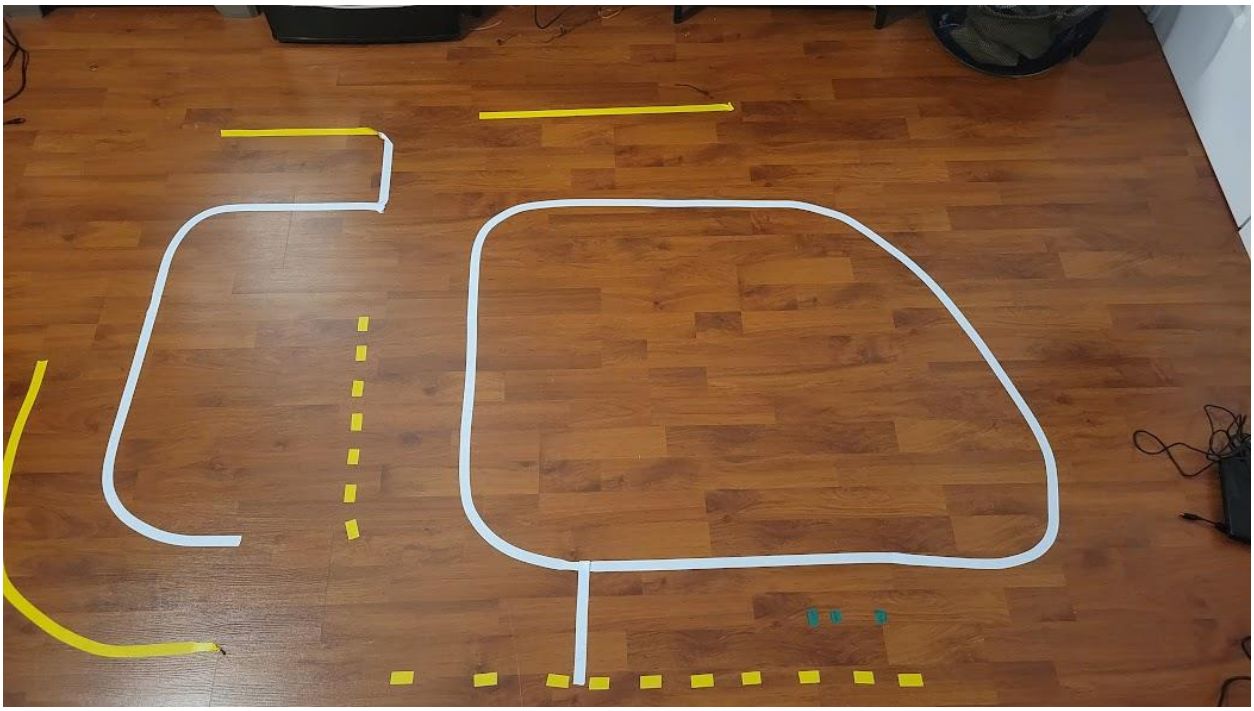


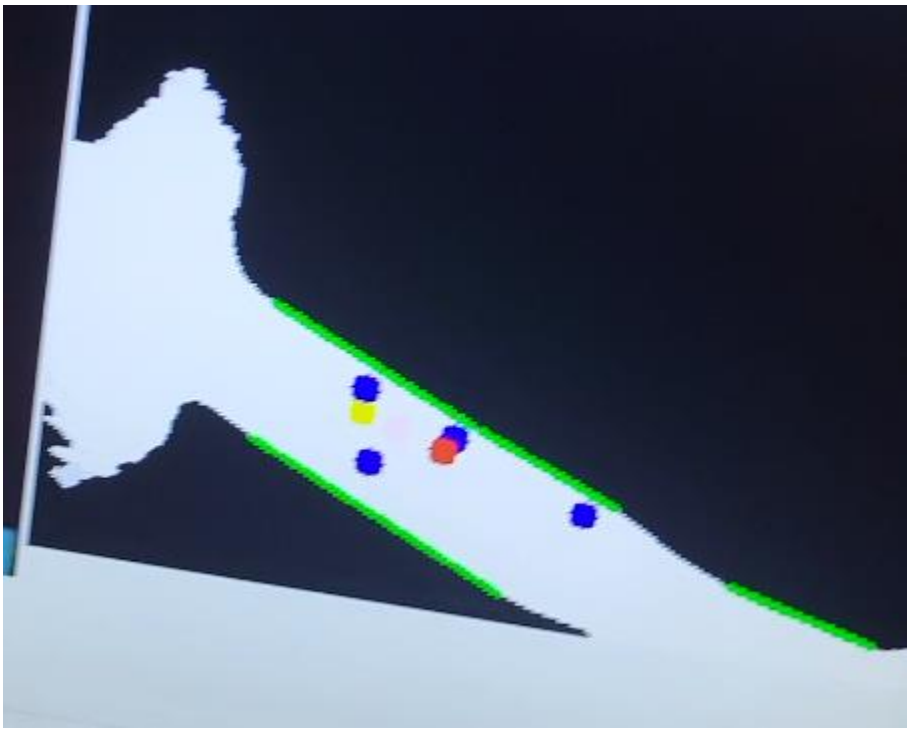


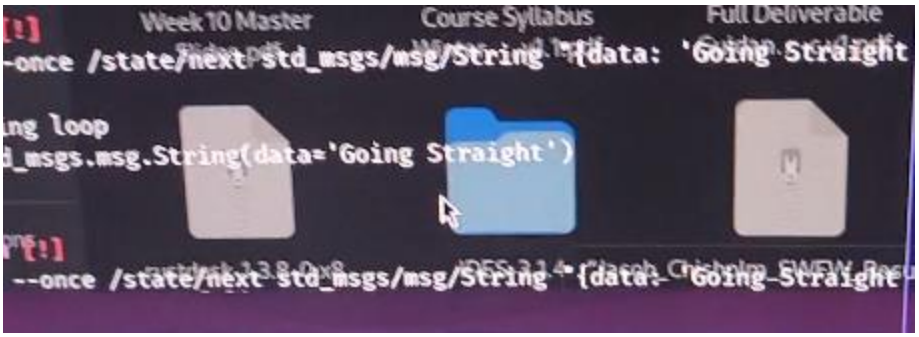











Week 10 Master
Course Syllabus
Full Deliverable
-once /state/next std_msgs/msg/String "{data: 'Going Straight
ng loop
_msgs.msg.String(data='Going Straight')
--once /state/next std_msgs/msg/String "{data: 'Going Straight