Race of D.O.O.M.

Driving Obediently On the Motorway

DESIGN DOCUMENT

sdmay25-08

Client: Dr Bigelow

Adviser: Dr Bigelow

Team Members:

Alex Crandall – Component Designer

Wesley Jansen – Team Representative

Elizabeth Schmitt – Test Specialist

Ben Towle – Client Interaction Specialist

Lalith Vattyam – Team Manager

Email: sdmay25-08@iastate.edu

Website: sdmay25-08.sd.ece.iastate.edu

Date: 12/7/2024

Version: 10

Executive Summary

Our project "Race of Doom" is a fun way to bring a strong level of understanding in the autonomous driving space. Technology increases at a rapid rate, so it is important to have a wholistic understanding of software and hardware in an area where not only efficiency is a concern but also the safety of the public and the major ethical concerns that are consistently at play. Race of Doom is meant to mimic a real-world scenario of driving where we will be using an RC car to go around a track. It can navigate around real-world obstacles that drivers see daily in a safe and efficient manner. Another major goal of ours is to have a solid hardware and software design that can be passed on to future groups. This involves not only quality work, but also a lot of documentation, so the team after us can pick up where we left off.

The track that we are using is two lanes wide, and we have obstacles such as left and right turns, people crossing, construction area, obstacles in the road, and stop signs. These are common obstacles that we see every day. Developing efficient, reliable, safe, and protected software allowed us to utilize our knowledge from our curriculum to come up with a robust solution. We are using HC-SRO4 Ultrasonic Sensors and a RealSense camera to get the optimal data for precise autonomous driving.

We have done a lot of testing on our RC car and can see that it is moving through the track correctly. The design efficiently identifies objects, whether they be signs or obstacles in the road. After these are identified the car moves around obstacles or do what the signs indicate seamlessly. The car moves in a safe manner so that everyone "within" and outside of the car is safe from crashes and jerking movements. We have also made sure to create and update documentation as we go. This includes the readme, the many figures shown in this design document, and the design document itself. Documentation is important since this is a legacy project. We want to leave this project in a good state for future groups.

Our design creates a great base for an autonomous vehicle and all the components and logic that is required of it. Steps to be taken next for this project include integrating the RP-Lidar sensor into the design and finding a way to decrease the speed of the car through the motor. The Lidar sensor works well with very accurate data. The next team will need to find a way to supply the Raspberry Pi 5 with more power so that it can run properly. They will also need to find a way to slow the car down so that it is not just moving in small increments. Right now, the design has a work around for the quickness of the motor. The next team will be able to work on a new motor that does not need a software bandage fix.

Learning Summary

Development Standards & Practices Used

- IEEE Standard Glossary of Computer Hardware Terminology (610.10)
- ISO/IE/IEEE International Standard – Software and Systems Engineering
- IEEE 2846-2022 – IEEE Standard for Assumptions in Safety-Related models for Automated Driving Systems

Summary of Requirements

- The RC car runs fully autonomously
- The RC car does not run into walls
- The RC car does not run into obstacles
- The RC car stays in the correct lane
- The RC car changes lanes when necessary
- The RC car stops at stop signs
- The RC car stops and waits for people to cross at crosswalk
- The RC car turns right and left when sign indicates

Applicable Courses from Iowa State University Curriculum

CPR E 281

EE 201

EE 230

CPR E 288

COM S 252

CPRE 308

CPR E 310

New Skills/Knowledge acquired that was not taught in courses

We have been able to practice a lot of new techniques and skills while working on this project. First off, we have had much more practice designing hardware for an embedded system. We have never had so much freedom when it comes to design. We have learned how to create a scope for a project from nothing. This has been a big learning curve. We found that we had a lot of ideas and learned that choosing one path can be difficult yet necessary.

The team has also learned how to use a Raspberry Pi. None of us have used a Raspberry Pi in the way we must for this project. We have learned how to power and SSH on this microcomputer. We have found how to connect the pins to all the other sensors to it. Working with these sensors has also taught us a lot about all the different hardware components and how to work with them. Hardware is not something we work with much, since a majority of our team is in Computer Engineering.

Table of Contents

List of figures

1. Introduction

1.1. Problem Statement

For our project, “Race of Doom”, we are creating a small-scale model of an autonomous vehicle. The idea is for this model to be used for actual full-sized vehicles. We will be optimizing software and hardware components for the vehicle to be able to move completely autonomously throughout a track with real-world obstacles. This product will help to prevent distracted and unsafe driving. Distracted, tired, or drunk driving is very unsafe. Human errors when driving cause many crashes and deaths. The vehicle will move in a way that will be very predictable and safe on the roads with many safety features. Leading to less crashes and human fatality.

1.2. Intended Users

We have a large, intended audience. Anyone who wants an autonomous vehicle would be able to use the final full-scale product. Users may want it because they know they get distracted while driving, they get nervous while driving, or they might want to sleep on a long car ride. Everyone on the road will be affected by our design if it is put into full-sized vehicles. Even those who do not drive these cars will have to drive on the same road as those who do.

One potential user is the “Tech Guru”. This user wants the newest, most cutting-edge thing. They also prioritize efficiency and processing power of the design. This person needs the design to be very modern and up to date. They need the best sensors, with many software updates being pushed up to keep everything modern. This user can benefit from an autonomous vehicle because it is the most intelligent, cutting-edge thing in cars. This connects to our overarching goal of getting these vehicles that are less dangerous on the road.

A second potential user is the “Worried Driver”. This user has safety and reliability as their number one concern. We need to make sure that our design has very little room for error. This person is a very important user to us. We need to consider the safety needs above all else, making sure our design can adapt to unexpected behavior on the road. The worriers can benefit from our car design because they may be nervous about driving on their own. A nervous driver can act unpredictably on the road. This driver could use the autonomous vehicle to help them drive more safely.

A third potential user is the “Incompetent Driver”. This is someone who cannot make precise decisions under pressure while driving. This could be a user who simply does not enjoy driving, someone who is tired, a new driver, or someone who cannot drive well in their current state. The common thread between these users is that they need to get someone but cannot do it safely on their own. They need a car that can drive for them. They will benefit from an autonomous vehicle. This car will get them to their destination without help from the user. The incompetent driver connects to our overall goal of getting people around in a timely and safe manner.

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

We have several vehicle requirements and constraints that fit into various categories. These requirements and constraints are listed below in different categories.

Functional Requirements:

- The vehicle can move in every direction.
- The vehicle moves completely autonomously.
- If vehicle gets stuck on an object or wall, it must be able to find its way out
- The vehicle must be able to maintain the center of its lane
- The vehicle must obey the unique road signs it encounters with the appropriate action
  - Stop Sign: Upon encountering a stop sign, the vehicle must come to a complete stop, wait for at least one second, and finally look left and right before continuing
    - To simulate "looking left and right", the car will simply turn its servo (wheels) left and right
  - Construction Sign: Once the car detects that there is a construction sign it proceeds with caution.
  - Construction Cone: Construction cones are placed in a construction zone, meaning they only happen right after a construction sign. The car will know that it needs to look for cones here and move with caution around them.
  - Crosswalk sign: If there is a "person" standing at the cross walk it will wait until they are no longer there. If there is no one it just continues to coast by.
  - Generic object or Random pedestrian: The car must simply steer around the object or pedestrian and then re-align itself to the correct lane
  - Turn Indicators: The car must make a 90 degree turn to the left or right while not going outside of its lane in the process (must "hug" the right lane on a right turn and vice versa
- The vehicle must reach an indicated finish point and exit the client program once it reaches this point

Team Requirements:

- The team writes all the code themselves.

- The team implements all the hardware themselves.

User Experience Requirements:

- The vehicle slows to a stop instead of stopping very suddenly.
- The vehicle stays within the track.
- The vehicle moves about the track without hitting any walls, “pedestrians”, or obstacles on the road.
- The vehicle stops when it sees a stop sign.
- The vehicle slows down at a crosswalk.
- The vehicle changes lanes when needed.
- The vehicle turns when the turn sign is indicated.

Monetary Constraint:

- The budget is $500 for all sensors and hardware used.

2.2. Engineering Standards

It might be a big surprise to some people, but engineering standards are an important part of everyday living. Technology is such a huge part of every person's day-to-day life. The IEEE standards make sure that people have positive experiences with the technology that is so engrained in their everyday lives. Without these standards many items would not be quality enough for users to want to interact with them. An example is the quality of speakers. People are constantly listening to music, whether through tiny or big speakers, within the IEEE standards these speakers could be very staticky, creating a negative experience for many people.

These engineering standards are also very important for our race of doom projects. Below are three IEEE standards that are important to keep in mind as we continue the development of our project.

IEEE Standard Glossary of Computer Hardware Terminology (610.10)

This IEEE standard provides us with constraints on what we refer to different signals as. These naming conventions are mostly constrained for their respective purposes. Therefore, this IEEE Standard helps us to keep terminology consistent and easy to understand by a team of individuals.

This standard holds a great deal of relevance to our project. It will mostly impact on the way we document our design. It will be most helpful for future groups to look at our naming conventions and that will result in less confusion overall.

ISO/IE/IEEE International Standard – Software and systems engineering – Software testing – Part 2: Test processed

This IEEE Standard helps developers understand the test process specifications of a software system. These test processes include unit, integration, and regression testing. Overall, this standard gives software developers an idea for what the lifecycle of the software testing process is. This will be especially helpful for us, because our design will demand rigorous testing and trials to ensure that our vehicle's autonomous design is reliable and efficient.

This standard has major relevance to our team's testing process. We have tested using the rigorous demands of this standard. It is very important for our autonomous vehicle to be reliable and work 100% of the time. People's lives could be in our hands and one mistake made in an autonomous vehicle could be fatal.

IEEE 2846-2022 - IEEE Standard for Assumptions in Safety-Related Models for Automated Driving Systems

The purpose of this IEEE standard is to specify the minimum assumptions needed to ensure the safety and reliability of autonomous vehicles. Such include what it means for an autonomous car to "Drive Safely". It is important for us to be on the same page as a team by making sure we follow this standard.

This standard is more applied to actual autonomous cars used on real roads. However, it has some relevance to us since our R.C car simulates a safe driving environment. This is because a lot of our objectives are to stop at stop signs, yield to pedestrians, and avoid hazardous objects.

As a team, we also considered a few other IEEE standards. One of these was IEEE P3412: Standard for Autonomous Driving Architecture. This standard covers the full lifecycle of an autonomous vehicle. It gives important standards for the architecture of autonomous driving. This can be important for us to look at to make architectural decisions in the future. It is not one of our main standards since it is talking more about the whole life of a full-sized autonomous vehicle, while we are working on a small prototype.

Another standard that was found was IEEE 7001-2021: IEEE Standard for Transparency of Autonomous Systems. This standard talks about being transparent and testing autonomous systems in an objective way. This will be important for us to do. We need to have objective tests to grade each of the team's vehicles fairly, since it is a race. We also must consider the transparency bit. We need to make sure to be transparent in our documentation so that the team next year will be able to understand where we left off. This is a good standard to consider, but we did not think it was as important as those listed above.

In our project we have implemented the IEEE standards by following a standardized naming convention for the different signals which will allow our software to be acceptable, readable, and understandable by teams.

We also adhere to the second and third IEEE standard listed above by having a well-thought-out testing process that will be written on our documents that will not only give our peers but possible future users the scale, scope, and risk of the project. By having this standardized testing process, we get to show the reliability of our software and how the testing that is done gives credibility to the safety of autonomous vehicles as that's a primary concern many people have.

3 Project Plan

3.1 Project Management/Tracking Procedures

For Race of Doom our team has adopted the agile project management style. Agile is an adaptive workflow style. This works well for our team because we did not know everything upfront. For example, we were not sure if the processing power of a Raspberry Pi was enough. We knew that using Bluetooth to connect the Raspberry Pi to a computer to do data processing could be a possibility. This is something we did not know until the hardware was all put together so that we could test and analyze the speed. This is just one example of something that we did not know right away, making an agile workflow advantageous. Our goals for the project were not completely clear. We were even unsure of what the racetrack would look like due to location and other constraints. Agile is great for an ever-changing project like Race of Doom.

Our team is using Discord to communicate with one another. This is an easy communication tool to keep us connected. Git is used to keep track of all our code. We are keeping our documentation in a CyBox. CyBox makes collaborating and sharing documents easy.

3.2 Task Decomposition

For the Race of Doom project, we have a timeline to complete large tasks that can be split up into smaller/more manageable tasks. For example, one large task we must complete is assembling the hardware design for the R.C. car. This task can be divided into several smaller tasks that can be completed in the span of a few days/weeks. For example, setting up the HC-SRO4 Ultrasonic sensors with the Raspberry Pi processor board is a smaller task that adds to the milestone of completing the task of R.C car assembly. The following tasks can be provided as follows...

I. Disassembly of previous R.C car design - Completed on 10/23

II. Assemble the new HW Design of R.C car - Completed on 03/31

a. Test the connection and data visualization of RealSense D455 Webcam to determine the best place to mount it onto the Car

b. Repeat task a for the HC-SRO4 Ultrasonic sensors

c. Mount the needed sensors onto the car after determining the most optimal placements of each sensor

d. Connect each sensor to the appropriate Raspberry Pi 5 pins using a breadboard and jumper wires

III. Retrieve the needed components to implement our Racecourse obstacles – Completed on 4/18

a. Design our obstacle course

b. Figure out what components are needed for each obstacle

c. Find a location to have the obstacle course

d. Retrieve stop signs, turn signs, pedestrian walk signs, and construction signs.

IV. Autonomous movement of R.C car - Completed on 04/25

a. Install the necessary libraries and SDKs for Raspberry Pi, Camera, and RPLidar Sensor

b. Construct a rough software design plan to facilitate communication and connections between our central processing unit, and external components/sensors.

c. Implement code for the camera to detect different signs.

d. Implement code for the vehicle to not hit things using ultrasonic sensors.

e. Integrate the camera and movement code together to create one cohesive test.

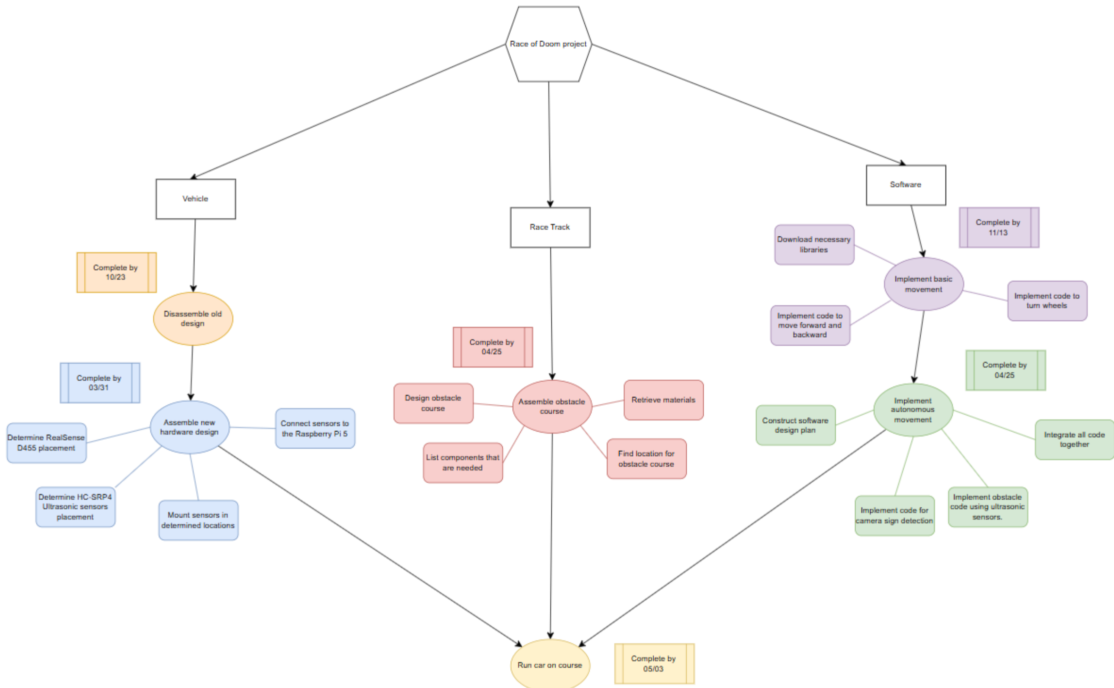


Figure 1: Task Decomposition

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The key milestones for this semester for the Race of Doom project are deconstruction, construction, retrieving components for obstacles, and basic movements. These can change because of the Agile development process.

The first milestone is deconstruction, this milestone involves deconstructing the car from last year so that it can be built back up again. The first task of deconstruction involves testing the old team's hardware to see if the car can move using it. The metric to know if we got it done is if we turned it on and got it to move at all. This metric basically tells us whether the hardware still works to move the car in any way. We can base some of our design off their hardware based off of this. The second task involved in this milestone is taking the car apart. The metric for this is that only the main shell of the car remains. There are no wires, circuit boards, or batteries that did not come with the car. Taking the car apart is important to get to the bones of the car and see how everything works.

The second milestone is construction, which involves a lot of planning and assembly. The first task for this milestone is research. We need to research and compare the different processors and sensors to use for our design. The measurable output for this task is a list of at least 2 potential processors and 3 potential sensors. The next task is to create a hardware design diagram based on the research that was done. The diagram must include the sensors and processors we chose and how they are all connected. The diagram is the measurable progress for this task. The next task is to mount the processor. The Raspberry Pi must be mounted onto the car for this step. The metric to determine this is done is having the Raspberry Pi mounted to the middle of the car. Then we must test the connection and data visualization of RealSense D455 Webcam to determine the best place to mount it onto the Car. The output for this task is the RealSense D455 to be mounted onto the car in the optimal location. This task is repeated for the HC-SRO4 sensor. The last task is to connect each sensor to the appropriate Raspberry Pi 5 input.

The Third key milestone for our project is to retrieve components for obstacles. The first task for this is to decide what obstacles would be used for the course. Once we have made a final decision on all our obstacles, we will create a document to write down all our obstacles with visuals to help understanding. This will be how we ensure that we are finished with the first task. The second task is to design our obstacle course. The delivery for this task will be a diagram of the course. After this design is created the last task is to find the components needed to create the obstacle. The components needed must be found and ordered if needed. All the components must be either obtained or ordered for this milestone to be considered finished.

The last key milestone for Race of Doom this semester is to autonomous movement for the vehicle. The first task for this milestone is to install the necessary libraries and SDKs for Raspberry Pi and Camera. All the libraries and SDKs are needed to be able to code on Raspberry Pi5. These must be installed on all our computers for the task to be considered done. The final task is to construct a software design plan to facilitate communication and connections between our central processing unit, and external components/sensors. The code that is created must make the car move autonomously without hitting anything and while following the instructions of the signs. If it cannot move by itself without hitting any obstacles and while following all the signs this milestone will not be considered done.

3.4 Project Timeline/Schedule

Task name	Start date	End date	Assigned	Status	09.02.2025	16.02.2025	23.02.2025	02.03.2025	09.03.2025	15.03.2025	16.03.2025	23.03.2025	###	06.04.2025	13.04.2025	20.04.2025	27.04.2025	04.05.2025	11.05.2025
PCB Circuit Design	02.01.2025	09.03.2025	Alex	Done															
Get the R.C Car to run again	07.02.2025	13.02.2025	Ben	Done															
Integrate RPLidar Framework	13.02.2025	24.02.2025	Ben/Lalith	Done															
Integrate Webcam SDK	13.02.2025	24.02.2025	Lizzy/Wes	Done															
Software Architecture Design	23.02.2025	02.03.2025	All except Alex	Done															
Track Obstacle #1 - Stop Sign	09.03.2025	19.03.2025	Alex	Done															
Track Obstacle #2 - Pop Up Pedestrians	30.03.2025	06.04.2025	Alex	Done															
Track Obstacle #3 - Construction Zone	08.04.2025	28.04.2025	Alex	Done															
Assemble Track and Test with R.C. Car	30.04.2025	11.05.2025	All	Done															
Development Teams - Sprint 1	24.02.2025	10.03.2025	All except Alex	Done															
Development Teams - Sprint 2	10.03.2025	24.03.2025	All except Alex	Done															
Development Teams - Sprint 3	24.03.2025	07.04.2025	All except Alex	Done															
Development Teams - Spring 4	07.04.2025	21.04.2025	All except Alex	Done															
Development Teams - Sprint 5	21.04.2025	05.05.2025	All except Alex	Done															

Figure 2: Second Semester Gantt Chart

3.5 Risks and Risk Management/Mitigation

All the tasks in the first milestone of Race of Doom, disassembly of the car, have very similar risk involved in each of them. This milestone involves handling a lot of the hardware components. There is a risk of breaking components for these tasks. Some of the pieces, such as the wires, are fragile. Because of this the risk of breaking components while disassembling the car is high. The probably of breaking something is probably around 15%. The good news is the parts that are more fragile are also easier to replace, like the wires. These components are cheap, and we have a lot of them to use if needed. The most important thing for these tasks is to be very gentile when taking everything apart. It is important not to yank things and to think before just pulling. Being mindful of our actions will reduce the risk of broken components by a lot.

For the second milestone while constructing the car there are many risks involved. The most prominent thing is the risk of short circuits. We must make sure before turning anything on that we know what it is doing and that it will work. If we do have a short circuit, it could just simply ruin a part or fry our battery forcing us into a large setback. Also, while constructing the car there are connectivity risks. We are not sure exactly how different components will work together so trying to force things could be a waste of time or a huge headache. Finally, the physical building of the car and the layout of the sensors comes with the risk of things not being secure. Once the car starts rolling, we do not want things to fly off the car, so this is a definite risk. We also want to have sensors facing every way that we want. The overall risk for this milestone is high and I would assume running into some sort of setback is almost guaranteed at about 90%. This is because no project goes perfectly, and you learn most from just trying things out. It is very important to try to mitigate this risk. Firstly, it is most important to try to prevent short circuiting. Carefully designing the hardware using diagrams and checking these designs thoroughly is vital. When assembling these designs, we must follow the diagrams carefully, making sure everything it turned off before connecting any components. Triply checking the hardware matches the diagram exactly before turning anything on is the last step. Once everything is working properly, they must be attached to the vehicle very securely. Testing movements need to be slow at first, to make sure none of the components fall off or break.

Retrieving parts of the obstacle design is milestone 3. The main risk for the first task of this milestone is being overly optimistic or pessimistic about what our vehicles will be able to do. Since we are not exactly sure how the hardware and software will work yet for our design, the risk of not having a perfect understanding of things that the car does is high. The risk of designing obstacles that are too simple or too difficult is high, probably about 83% chance that this will happen. Knowing this, we need to design obstacles that are adaptable. The components that we order should be able to be used for more than one thing. This way the challenges can either increase or decrease in difficulty depending on how things are going. The second task involves ordering the determined parts from ETG. The risk here is possibly ordering the wrong parts. This can be prevented by double checking the order before placing it. With this check the probability of this happening is low at about 18% chance. If this happens and we need some different components, we can try to look at local stores to prevent more miscommunications. When we have the parts the last task of the course design can happen. Creating the whole course without having the whole picture of what the vehicles will be like will be tricky. Like for the first task, the probability of us estimating our own abilities wrong is high. The chance is at about 76% that this will happen. The design we make for the truck will be subject to change. It would also be good for us to design multiple courses, some easier than others. This will make it easier for us later.

The fourth milestone of implementing basic autonomous movements on the vehicle is done in software. This makes it less risky in general. Mistakes on software are typically much easier to fix than on hardware. The first task of installing different libraries and SDKs may be frustrating. It is often more difficult to install software than it feels like it should be. The risk of installing things incorrectly is high at about 58% chance. Installing something wrong is easy to fix. This part of the project may take some trial and error, but the risk of things going wrong in a way that is unfixable is very low. The second task is to research. The only risk involved in research is spending too much time doing it. The risk of spending too much time researching how to program onto a Raspberry Pi 5 is about 60%. I think it is important to investigate this and make sure it is not to go over that time. It is okay to learn by doing it in this instant because the probability of software being so messed up that it cannot be undone is very small. That ties in well to the last task of developing a basic interface for the movement of the car. The risk of messing up software being too bad is very low, probably about a 14% chance. This part of the project will not be too risky.

For the first milestone we had to test the previous group's design and see how well it worked and what we could reuse. There was a risk there of the code and design not working together and us having to restart. This risk came to pass, and we basically had to start our design over from scratch. This was very frustrating, since we thought we had a good hardware base, meaning we could do a lot of cool things using software, but that did not happen. We have been able to create our own new hardware design, but it has taken up a lot of time that we expected to be programming. This has been a bummer.

Another risk was that the raspberry pi would not have enough power for us to be able to use all of the sensors that we had planned to use. This ended up happening and we had to get rid of the RPLidar sensor. The Raspberry Pi5 could not give it enough power. We had to weigh which sensor was more important to us, the Lidar or the camera. We decided on the camera but are losing some really accurate distance data that we could have had with the Lidar sensor. This was something we were able to talk through to make a final decision on.

3.6 Personnel Effort Requirements

Task	Alex	Wesley	Lizzy	Ben	Lalith
Disassembly	6	1	1	1	6
Basic Movement	0	13	13	13	13
Assembly	23	14	14	15	10
Autonomous Movement	0	20	19	28	26
Obstacle Course	10	6	6	6	5

Figure 3: Expected effort (hours)

Task	Alex	Lizzy	Wesley	Lalith	Ben
Disassembly	6 hours	1 hour	1 hour	6 hours	1 hour
Assembly	15 hours	15 hours	13 hours	12 hours	14 hours
Gather Parts for Obstacles	4 hours	4 hours	6 hours	3 hours	5 hours
Basic Autonomous Motion	10 hours	15 hours	15 hours	14 hours	15 hours

Task	Alex	Wesley	Lizzy	Ben	Lalith
Disassembly	6	1	1	1	6
Basic Movement	0	12	11	18	11
Assembly	21	14	14	25	10
Autonomous Movement	0	21	19	32	26
Obstacle Course	9	7	7	9	5

Figure 4: Actual effort (hours)

Overall, we stood with our plan for personal effort. Alex has a more special role, as can be seen above, because he is the only Electrical Engineering major. He was very hardware focused. The others were less specialized and worked on a little of everything. Everyone hit there hours, and Ben went in extra to keep the team on our feet. We are all very thankful for all of Ben's contributions.

3.7 Other Resource Requirements

Hardware Resources

1. RealSense D455 Depth Camera
2. Raspberry Pi 5
3. HC-SRO4 Ultrasonic Sensors
4. Breadboard
5. Electronic Speed Controller (ESC)
6. Servo
7. Motor
8. 8.4 V Battery

Software Resources

1. PyQT – For processing and running scripts compatible with Raspberry Pi
2. PyRealSense– For gathering and interpreting Realsense Camera data in Python

4 Design

4.1 Design Context

4.1.1 Broader Context

It is important that we think of the broader context when designing this autonomous vehicle. Autonomous vehicles can drastically change the way people all over the world are able to live their lives. We are designing for communities that have the infrastructure for autonomous vehicles. Larger cities are a good start, but these cars affect everyone. From workers that manufacture them to those that will own them. Below is a chart that addresses some important concerns and considerations about a large-scale version of the car that we are designing.

Area	Description	Examples
Public health, safety, and welfare	Drivers and passengers will benefit from advanced safety features of autonomous vehicles. The car's ability to yield at yield signs, stop at stops, and avoid obstacles.	Reducing safety risks Preventing distracted and unsafe driving Promoting safe driving practices
Global, cultural, and social	Communities value their members' safety and our project will reduce risk of accidents. Adaptability is important in cultures and this project adapts to an ever-advancing society.	Community safety Ethical development Encourage technological advancements
Environmental	We must use energy efficiently with the Pi, which is a low-power component. The vehicle will reduce emissions by being programmed to drive efficiently.	Sustainable materials Low-power component: Raspberry Pi Energy consumption
Economic	The final product must be affordable with affordable components. The product will create more jobs.	Reduction in accident cost Job opportunities Cost-effective solutions

4.1.2 Prior Work/Solutions

Many groups and companies have been working on autonomous vehicles for some time now. From lane assist to taxis that full on drive themselves. There are varying levels of autonomy. *Supeed Pasricha* has done research on optimal design for autonomous vehicles. This research shows that it is important to have sensors all around the car for things like parking assist, blind spot detection, and street sign recognition [5]. This is like our design in the fact that we will have sensors in the front, back, and on the sides of the car. We think that it is very important to collect all the data that we can to get a accurate read of the car's surroundings.

A research paper was written on the different levels of autonomy and what sensors are used for these levels. This paper talks about a combination of LiDAR, Radar, and camera sensors on the vehicle. The camera is needed in front, overlapping with other sensors that can process things faster. Other sensors that are quicker are used in other areas around the car as well [3]. This research makes an excellent point about using many different types of sensors that serve different purposes. Our design includes camera and lidar sensors. We also include one other type, the HC-SRO4 Ultrasonic Sensors. This ping sensor is able to sense and process objects very efficiently. The values from it can also be cross referenced with other sensors to lower risk. We feel that this sensor is a great addition to just making our design stand out even more.

This is not the first year that a senior design group has done the race of doom project. In 2024 there was another group that had the same goal in mind. We were given their hardware and software to be able to rework their design. They used the RC car with an Arduino for processing. They used ping sensors to race through obstacles [3]. We chose to keep their RC car as an amazing base. The Ultrasonic Sensors are also being used for our design as quick processing information. One way we choose to differ from them is by having our computers process the information. This way we can store much more data. That way we can have the car do more things without being restricted by the processing power of the Arduino. We found that a Raspberry Pi has more Bluetooth connectivity than Arduino, so we chose to replace the processing unit. Since we have more storage to work with now, we are also able to use more sensors.

4.1.3 Technical Complexity

Our Race of Doom project consists of many complex components. The Raspberry Pi 5 is the brain of everything that our vehicle does. We have programmed this Pi through SSH, while writing all our own code to it. This is a little but powerful machine with many libraries that we have learned to understand. We have also connected this processor to the motor and sensors. We have connected the Pi in a way that allows the car to move in the desired direction given these sensors outputs. The code on it needs to understand all the sensor information, process it, and adjust the car's positioning accordingly. The other components are sensors. The HC-SRO4 Ultrasonic Sensors have a certain set of outputs. We will need to find equations for each of these outputs that indicate the data in a readable format. The camera will require photo processing. This includes being able to detect different colors and shapes. A very complex undertaking that was new to all of us.

The problem scope includes many different challenges on the racetrack. The industry standard for autonomous vehicles right now includes blind spot detection and backup cameras. Our track will have obstacles including street sign detection. This is photo processing, finding and avoiding walls, yielding, and changing lanes. This far exceeds what the average vehicle can do autonomously right now.

4.2 Design Exploration

4.2.1 Design Decisions

1. Placement of the sensors

    a. We made a key design decision by determining where each sensor should be placed. We decided to place 3 ultrasonic sensors on the ~~peripheral~~ front of the car with one on each side. The RealSense Webcam will be placed at the front for image processing. We decided to have sensors in these locations because each has a different job. The ultrasonic sensors on the front detect immediate danger of things that may be hit. They are angled so each one gets a slightly different view so that we can get an entire picture of what is in the front of the car. The Camera was also placed at the front so that it can detect the signs so that the vehicle is able to abide by the laws of the road. The sensors on the sides of the car determine what lane the car should be in, and when it needs to lane correct because it is too far to one side or the other of the lane.

2. GPIO pin functionality

   a. When creating our hardware design, a lot of important decisions to make involve which GPIO pins on the Raspberry Pi should connect to which sensor. The decisions made were based on where each sensor needs power distributed to it, as well as which pins are easiest to read/write from. We looked at a lot of Raspberry Pi5 documentation to determine which of these pins should be used.

3. Communication between P.C and Raspberry Pi

   a. There are different ways to establish communication between where we write our code (PC), and the Raspberry Pi. Some options include communication through Bluetooth, WiFi, and SSH. We decided to use SSH to communicate between the Pi and the PC because that is the most secure and efficient way to communicate.

.

4.2.2 Ideation

For the decision about communication between the P.C and Raspberry Pi, we had a lot of options to consider for connecting our program onto the Raspberry Pi. The thought process that went into this design decision involves how secure the connection is, how performance and speed will be affected, and how convenient or easy it is for us developers to implement the connection. Here are 5 options that we considered...

1. SSH - (Secure Shell)
2. Wireless Connection over bluetooth
3. File Transfer using SCP (Secure Copy Protocol)
4. Wired connection with USB or HDMI cable
5. Remote desktop connection (VNC)

4.2.3 Decision-Making and Trade-Off

We created Pros and Cons lists for each of the ideas for the connection between the P.C. and the Raspberry Pi 5. Below you can see the charts we made to make this decision. We decided that the scripting and automation abilities as well as remote access were 2 aspects of the connection that were high priority for our project. We also have reliable WiFi here at Iowa State and having a PI GUI was not that important to us. These considerations made SSH an obvious choice in the end. We sacrificed some speed and user-friendly aspects for the sake of remote access. We can access the Pi information on our induvial laptops. This is a huge plus. SSH became an obvious choice after these charts were made.

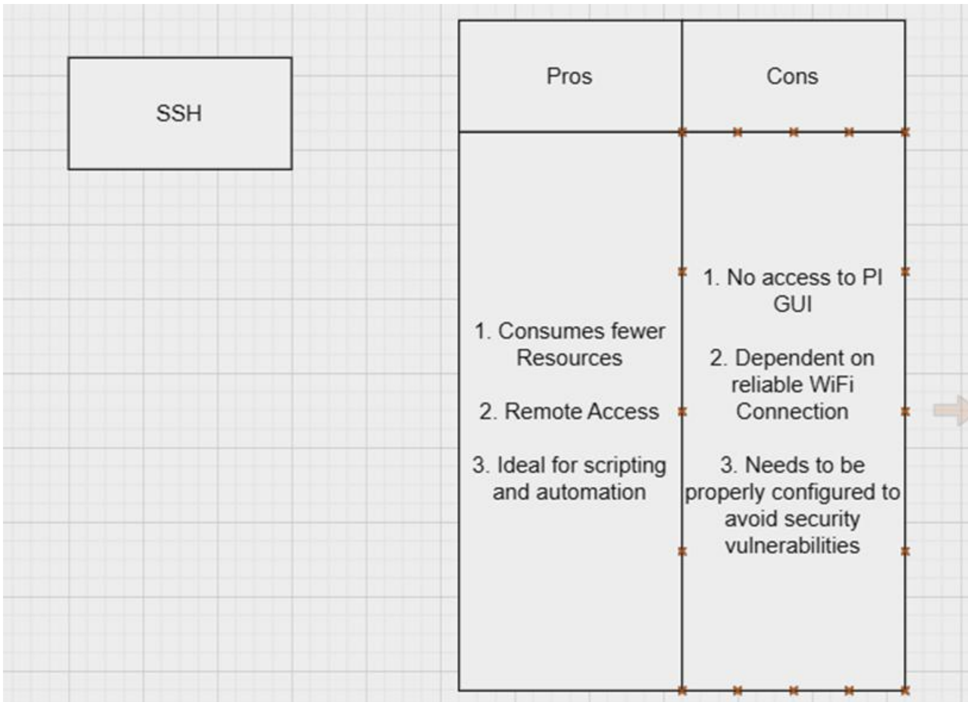


Figure 5: SSH

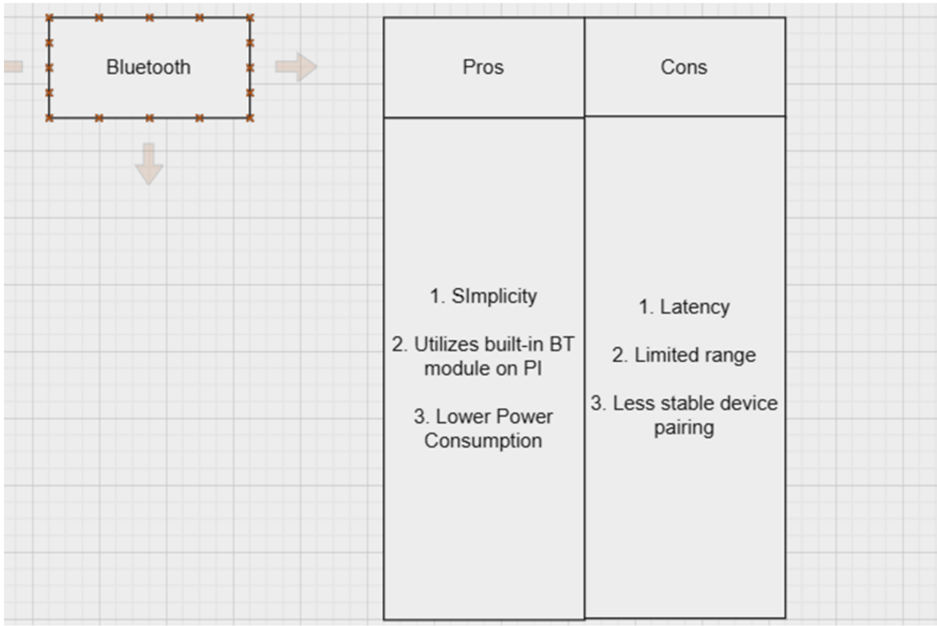


Figure 4: Bluetooth

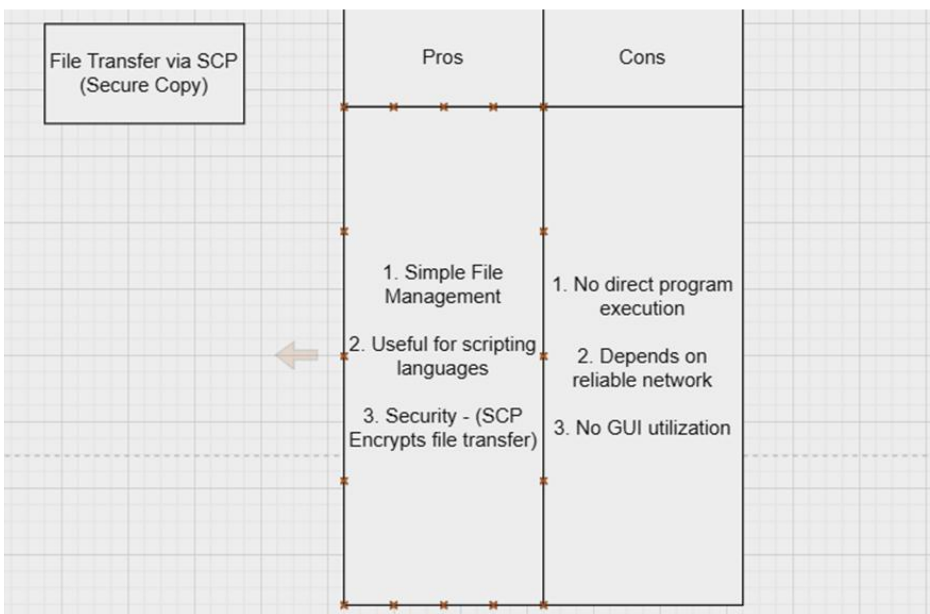


Figure 7: File Transfer via SCP

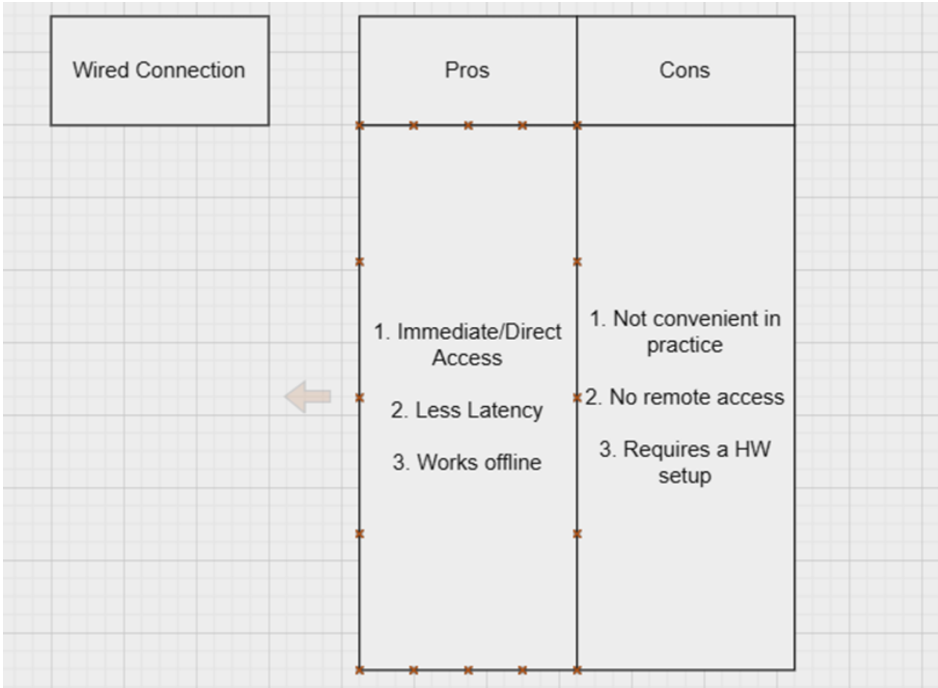


Figure 8: Wired Connection

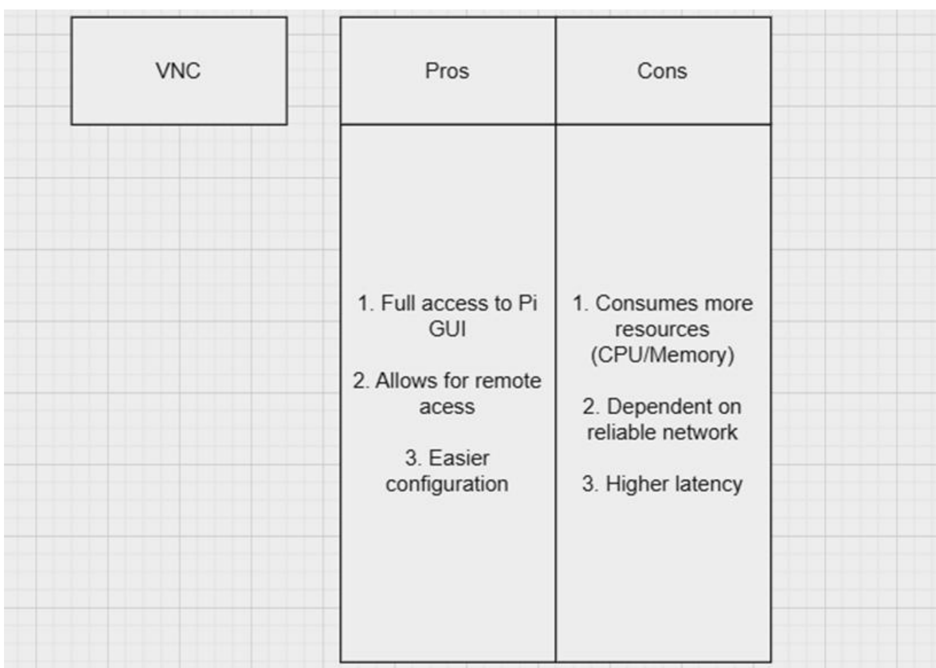


Figure 9: VNC

4.3 Final Design

4.3.1 Overview

The design of our R.C car begins with our Hardware design layer and mechanical components of the Traxxas Remote Controlled Car. Since for this project, we have no need for the remote, we instead use a series of small, programmable computers integrated onto one single board called a “Raspberry Pi” This will allow our software layer (i.e. the code responsible for autonomous movement) to control the car and steer it away from hazardous obstacles.

In addition to the Raspberry Pi, our system design includes the utilization of sensors and cameras to help the car know where it is, and what objects stand in the way. These sensors are connected to the Raspberry Pi so that our software knows what driving decision to make based on the input of the sensors.

4.3.2 Detailed Design and Visuals

Hardware

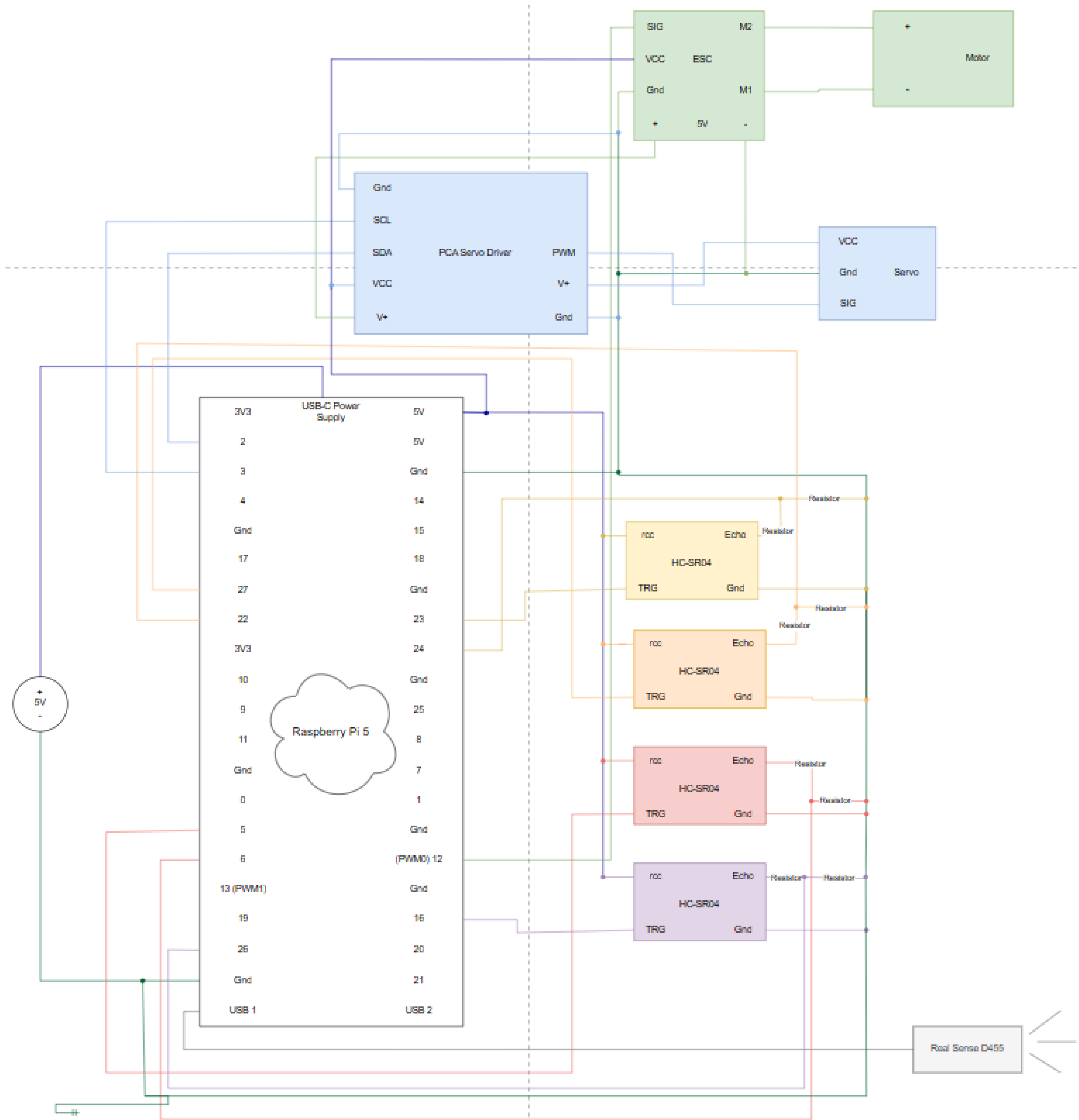


Figure 10: Hardware Design

This schematic Diagram Highlights our hardware implementation for the system. The central operation center of our system is the Raspberry Pi 5 Processing board. This is meant to direct incoming and outgoing signals from our sensors and distribute power to devices like our servo, motor, and webcam. The Raspberry Pi also comes with USB input and output ports to help us gather sensor information from the camera and Ultrasound sensors. The Ultrasonic sensors take in information about the distance of the objects around it. The Webcam also takes information about the colors and signs in front of the car, and it sends data to see if a road sign is detected. Information from both these sensors is received and processed by the Pi, which then makes decisions based on what is found. The Servo turns the wheels, and the ESC controls the wheels. These are both connected to the Pi through the PCA Servo Driver, which makes both works more precisely. The Raspberry Pi determines when and how each of these needs to mauver. There are three Ultrasonic sensors mounted to the front of the car with one on the right side. The ones in the front are for detecting objects, while the one on the right side is used to determine which lane the car is in.

Software

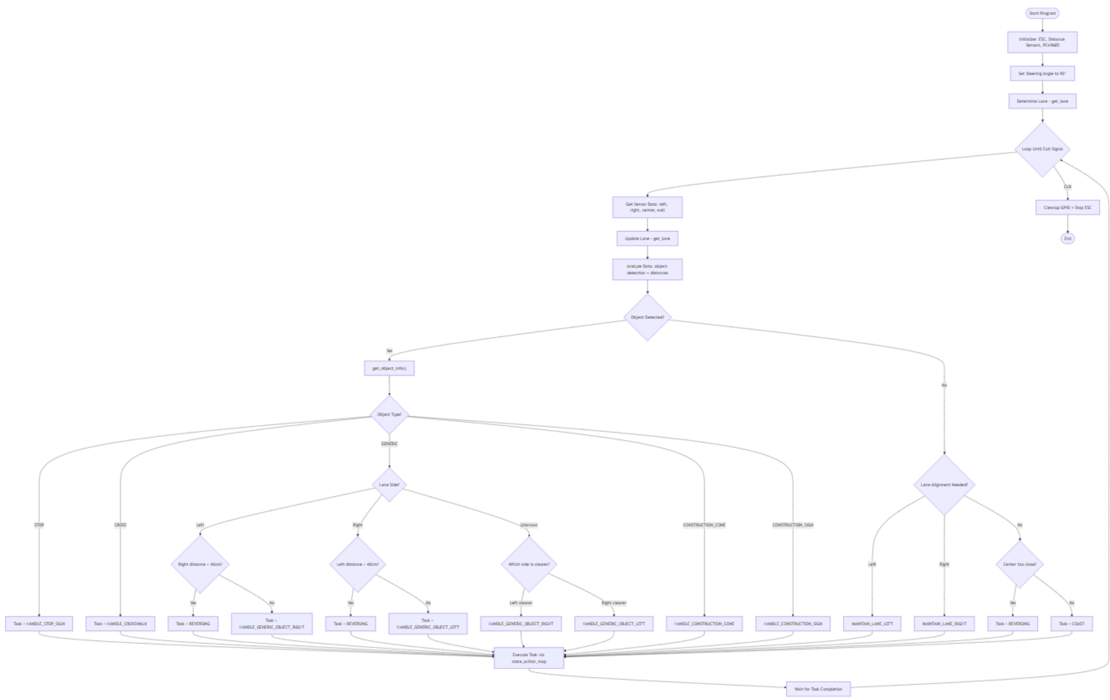


Figure 11: Software Design

Above is our high-level software design for the project. The basic idea is that the car sends data to our computer, for the RealSense webcam, and the Raspberry Pi, for the other sensors. The processor retrieves the data from the car and the sensors mounted on the car. It then directs the data to where it will be processed in the main logic. The sensors convert the raw data they produce and turn it into readable and understandable numbers. All the data we retrieve goes to the main logic where the decisions are made. The decisions are made through logical statements trying to decide on the next best course of action. The decision is sent to the commands, ultimately telling the car what to do next. This tells the car to scan, drive or whatever the main logic decides needs to happen.

One of the most difficult things in software design was training an AI model to work with the RealSense webcam to determine signs. The code can see signs, determine what they are with a certain level of confidence. A few photos are included below to show the results of our trained model.

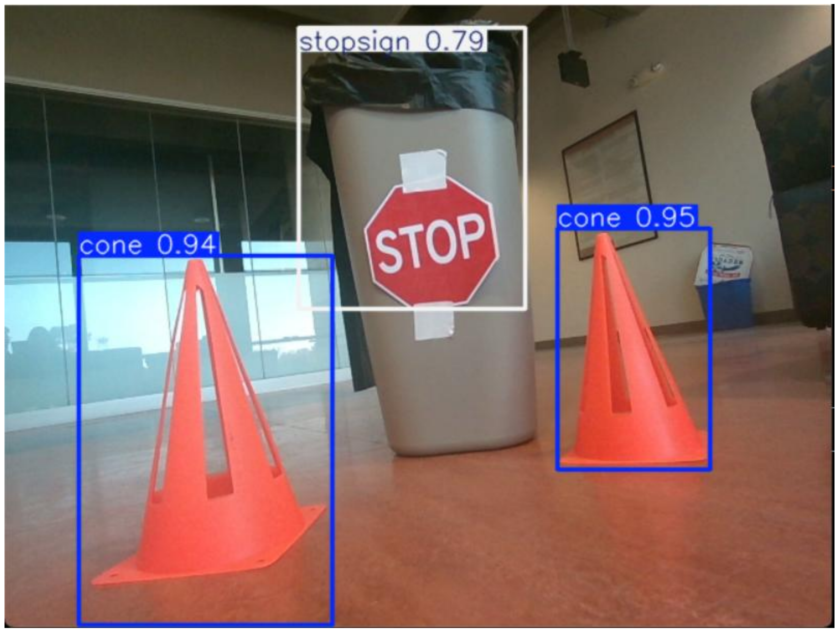


Figure 12: RealSense output cones and stop sign

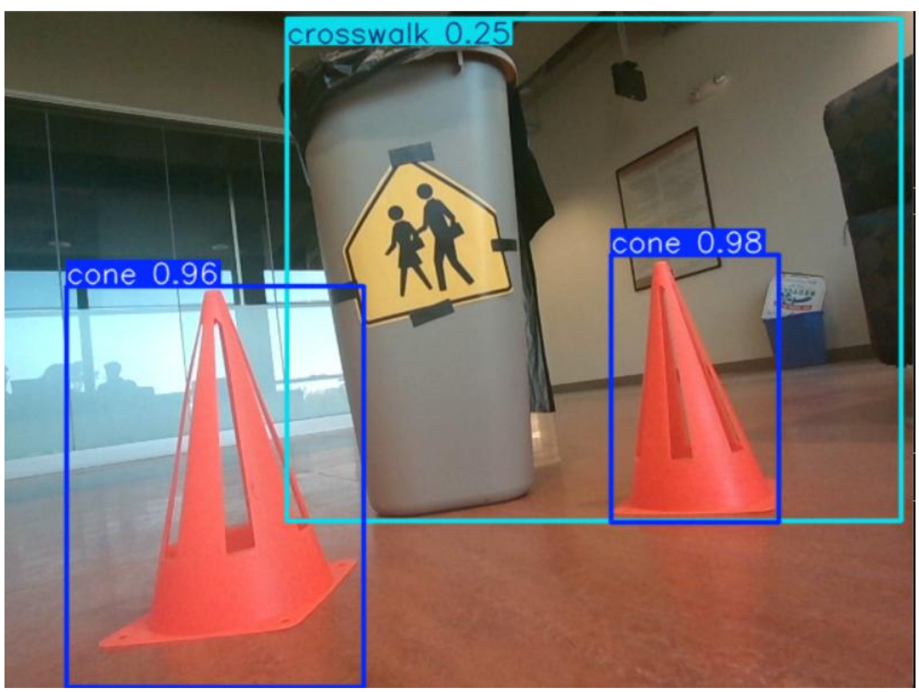


Figure 13: RealSense output cone and crosswalk

4.3.3 Functionality

Since the actual scope of our project does not have a lot of use in the real-world (unless people race R.C cars for a living), the applications of autonomous vehicle software is revolutionizing modern-day technology. If we investigate the scope of real-world autonomous cars, it is clear to see that these applications are intended to aid a user's driving experience. As we know, not all people in the real world are great drivers, so designs like these are intended to reduce the risk of human error on the road. The sensors and components on an autonomous vehicle can help prevent collisions either with other cars, or objects/pedestrians on the road. Our Race of Doom project aims to simulate the real-world driving experience on a smaller scale so that our design can be applicable to users who feel less comfortable driving.

4.3.4 Areas of Concern and Development

It has been difficult to work through this project since it feels that the scope was ever changing. Unexpected challenges seemed to arise every day that needed to be delt with. This includes not enough power given to the Raspberry Pi, the Pi not having enough power for the different components, and the speed of the car. It has been difficult to get the Pi enough power over a long period of time. It seemed to drain rather fast, but we found a battery pack that connects directly to the computer. We have gotten around then by plugging the batteries in whenever we are not testing the movement of the car.

To solve the issue of the Raspberry Pi5 not producing enough power we had to lose some parts of our design. The RPLidar sensor can no longer be used. It requires too much power so it cannot run just from the Pi. This is a good sensor that we wanted to use, and we have had to adjust our design considerably because of it.

We have tried many different solutions to get the car to drive more slowly. We tried to get new gear to make the wheels turn more slowly. This gear did not fit properly onto the vehicle, so the idea had to be scrapped. The next thing we tried was replacing our current motor with a brushless motor. We thought by doing this we could control the speed more easily through the software, but that did not work. Slight changes in the speed of the software have big results. Our final solution was to just have the car move in pulses. The car moves in a jerkier manner because of this, but this was the only way to get it to actual detect objects and make decisions without going too fast and running into things before information could be processed. It is a compromise that is not ideal.

4.4 Technology Considerations

Hardware

- HC-SRO4 Ultrasonic sensor
    - Strengths – High Accuracy, Cost Efficient, Lots of Documentation

- Weaknesses – Lower Response time, Affected by environmental factors

- Raspberry Pi
  - Strengths – Very Popular, Lots of Documentation, High Storage
  - Weaknesses – Limited Processing Power, Unideal for resource intensive software
- RealSense D455 Depth Camera
  - Strengths – Capable of detecting color, can identify certain images
  - Weaknesses – High power consumption, requires lots of processing power

5 Testing

We have put strong emphasis on testing early and often for our project. It is important that we test each component of our design separately before putting things together. Making sure everything works individually is important and will help us to debug when things are put together. Testing the entire project together will go very smoothly because of our early testing. Below we have laid out our plan at every level.

5.1 Unit Testing

The units tested are going to be movement controls, actuators, sensors, and nerf gun mechanism. We will have to write scripts that test command movements and verify responses to moving left, right, backward, forward, and stop. For the HC-SR04 Ultrasonic Sensors, and Depth camera we also need to simulate and test the different obstacles orientations and distances. By sending commands to the Servo, Motor, and ESC we can verify position and physical response. We can test the nerf gun by activating the mechanism to fire it. The tools we can use for this testing are unit testing framework PyRealSense and PyQT.

5.2 Interface Testing

Some of the interfaces for our design are sensor to processor, processor to actuators, and processor to nerf gun mechanism. We can use mocking tools, data loggers, integration scripts, and PyQT to do the interface tests. To test the sensor to processor connections we will inject some sample data from sensors then verify that the data is processed correctly by Raspberry Pi. For the processor to actuator connection, we will send commands for movement through the Raspberry Pi and then check the actuator responses. For the nerf gun we can send firing signals, checking the response after.

5.3 Integration Testing

Some critical integration paths for Race of Doom are movement command integration, sensor data integration, and safety features integration. All these paths ensure navigation is autonomous and safe. Sensor data integration consists of combining data from all the different sensors in a coherent way. We will have to test our data fusion algorithms, ensuring accurate representations of the environment. Movement commands must be coordinated and processed correctly through the Raspberry Pi. We can simulate this functionality through a track with obstacles while we monitor behavior. For safety we must ensure that the vehicle slows in yield areas, stops at stop signs, and avoids all obstacles. We can set up a track to simulate these things. The tools include a data fusion algorithm, physical track setup, and simulation environments.

5.4 System Testing

Our strategy for system level testing is to combine unit, interface, and integration tests to validate functionality of the system overall. The tests will include autonomous mode, full track navigation, and emergency stop. We must ensure that the vehicle can navigate autonomously. We must run the car on the entire track to test the movements overall, obstacle avoidance, and safety features. We must also test the vehicles' response to an emergency stop command. We will need simulation tools, test tracks, and data loggers.

5.5 Regression Testing

Our strategies for regression testing include re-running tests that were previously passed when new features or code are added. We will also maintain our regression tests that cover all critical features. The critical features are safety features, sensor data accuracy, and autonomous movement. The tools for these tests are automated test scripts, version control systems, and continuous integration tools.

5.6 Acceptance Testing

One strategy for acceptance testing is to create test scenarios for all non-functional and functional requirements. We will conduct demonstrations live on the test track. We will gather feedback from our stakeholder and adjust as needed. We will include our client using live demos, acceptance criteria, and feedback sessions.

5.7 Security Testing

We will not need security testing since there is no sensitive data involved on our project.

5.7 Results

One of the tests we have done is the unit test for the ultrasonic sensors. With these tests, we got to see how accurate the sensors distance measurements are. When I put an object in front of the sensor and measured the distance out with a meter stick, I found that the sensor read distances accurately with a tolerance of 0.5-1 centimeter. For our purposes, this will work very well considering that these are not the only sensors used. However, some readings can count as outliers which may cause some confusion for our autonomous machine later down the road. For objects that were very close to the sensor, occasionally the sensor will read that the distance is more than 2 meters away (outside of its intended range). For these cases, we have been very careful in our software design to ignore outliers like these and identify when a ping may be inaccurate.

The RealSense Camera has been difficult due to training the AI model. It has come with a lot of testing on whether the model we trained truly worked for different distances and for different sign locations. It was new for all of us so many tests were needed.

Integration testing was the most important thing for this project. Getting all the parts to work together and talk to each other is one of the most difficult things about our project. We worked hard and added basically one component at a time to our project, making sure each works well before moving to the next thing. The result was a car that moves with all of the parts working together.

6 Implementation

We have implemented a working, small-scale autonomous vehicle. This vehicle consists of an R.C. Car, HC-SR04 Ultrasonic Sensors, RealSense camera, and Raspberry Pi5. The Raspberry Pi 5 is a computer used as the brain of the whole machine; everything controls it. It is attached to the body of the car with a battery on it so that it does not die. The R.C. Car no longer requires a remote control, but we use the body, ESC, servo, and motor. The ESC is used to control speed, the servo makes the wheels turn left and right, and the motor makes it go. All these components are connected to Raspberry Pi 5, which controls when and how these actions are done. The Ultrasonic sensors are connected to the front and sides of the vehicle. There are three in the front which are used to detect objects that are in the way of the vehicle and the sensors on each side are used to make sure that the car is in the correct lane. The lanes are determined by how far the car is from the wall. The Camera is attached to the front of the vehicle above the Ultrasonic sensors. This camera is used to determine if there are any signs on the road that require the car to do a certain action.

These hardware components are all controlled or read by software on the Pi, the software then determines how to use this information and what action to do next. The code is written in Python. It is written in a way that the car naturally moves forward a bit, then sees if anything needs attention. It determines if anything needs to be done by checking the sensors. If The Ultrasonic sensors in the front determine that there is an obstacle in the way, the car will move until it is out of the way of the object. If the Ultrasonic sensors on the side determine that the car is too far left or right for the lane it should be in, this will be corrected at this time. If the RealSense Camera finds a sign that requires a certain action, that action will be done. If none of these things happen, the car will move forward and do the whole process over again. This is all done very quickly on the Pi.

The RealSense Camera uses AI to determine what is in its path. We have been training an AI to detect stop, crosswalk, left turn, right turn, and construction signs. When the camera determines that there is a stop sign in range the car comes to a stop for a time, then goes again as a real car on the road would. The crosswalk sign causes the car to stop until “pedestrians” have safely crossed the path. This is determined by the Ultrasonic Sensor. The left and right turn signs make the car turn one way or the other with enough leeway to stay in the correct lane. The construction signs instruct the vehicle to move to the other lane in the road due to the construction. This is all programmed in the Python coding language.

There were a few goals that we had initially that were unable to be met. These were having 2 teams compete and using the RPLidar sensor. We were not able to have 2 different teams to compete in the Race of Doom for a few different reasons. The first is that we had a very small team of only 5 people in total this year. In previous years this project had 15 or so people. The lack of other teammates made it hard for us to split up with the workload. We also did not expect the work needed on the hardware to be so extensive. We expected to be able to use the same vehicle as was used last year, but we had to change a lot of things up. There were a lot of roadblocks with the hardware, such as with speed control. Because of all of this we decided to combine into a single group that does all the coding together. We want to make a good work car for any team next year. We also did not end up using the RPLidar sensor. We spent a lot of time trying to get it to function and output data. In the end we found that it needed more power than the Pi can provide at this moment. There is a component that can be added to increase the power of the Pi, but we have decided to perfect what we have instead of trying to add a new component.

6.1 Design Analysis

Overall, our implemented design does all the functionality that we were hoping for. It is able to navigate around a course while following traffic laws and without running into anything. The vehicle moves autonomously in a predicable way. It can properly get from one location to another while staying in its lane, avoiding obstacles, and obeying all traffic laws. It does not make sudden movements that would give a potential passenger too much trouble. The Ultrasonic Sensors detect objects accurately and the Raspberry Pi5 can make decisions based on data from these sensors. The RealSense Camera detects all signs, and the Pi can make decisions based on what sign is seen. We hit our goal of the functionality that the vehicle was supposed to do.

There were a couple of things that did not go as planned. Because of the difficulty with speed control the vehicle goes for a little bit then stops, it inches forward instead of giving one smooth ride. We decided this was better than making the vehicle move uncontrollably fast. We could have addressed this issue of speed earlier and maybe we would have had the time to get the speed issue fixed. We found this software patch was the best thing to do with the time we had. This will be great for next year’s team to figure out and implement better speed control.

7 Ethics and Professional Responsibility

Since last semester, nothing has changed in this area because the way our vehicle works and its requirements have not changed. The only thing that has changed from the initial design is us becoming one team instead of 2 and us taking away the RPLidar Sensor.

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of responsibility	Definition	IEEE Code of Ethics

Work competence	Completing tasks efficiently and with quality	“7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.” This relates to us providing proper criticism for our work throughout the project.
Financial Responsibility	Providing a solution to our listed problem while maintaining a reasonable budget	“4. To reject bribery in all forms.” Staying honest to our budget constraints as a group and remaining responsible.
Communication Honesty	The ability to communicate with the team while maintaining integrity	“3. To be honest or realistic in stating claims or estimates based on available data.” This relates to our group providing valid sample data and not misrepresenting the values.
Health, Safety, Well-Being	Focusing on the mental state of others and preventing the overworking of one group member	“8. To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin.” This ethical statement aligns with the well-being of our team and its members.
Property Ownership	Respecting the materials we have been provided, along with honoring the components	“8. To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression.” This ethic also applies here to allow for little judgment with property and ideas.
Sustainability	Reduce the wear and tear of the project and respect the usage of materials	“5. To improve the understanding of technology, its appropriate application, and potential consequences.”

		Respecting material usage reflects understanding how to apply technology responsibly while considering its long-term consequences.
Social Responsibility	Produce easy-to-use products that are user-friendly and provide guidelines	"10. to support colleagues and coworkers in their professional development and to support them in following this code of ethics." Being communicative and speaking with members of the team ensures social responsibility.

Our group excels in communication, as we are closely connected and effectively coordinate meeting times, task assignments, and weekly responsibilities. However, one area for improvement is ensuring alignment on the project's scope. At times, there is some uncertainty about the target audience for our product. Establishing a shared understanding of this focus could enhance our teamwork and overall effectiveness.

7.2 Four Principles

	Beneficence (Promoting Good)	Nonmaleficence (Avoiding Harm)	Respect for Autonomy	Justice
Public Health & Welfare	Improves safety guidelines in following traffic laws	Reduces operator risk by enhancing accurate safety protocols in risk diversion	Enables users to override autonomous driving and have full control of the car	Ensure equitable access to AV technology Addressing potential job displacement in the transportation sector
Global/Culture	Enhance global mobility and connectivity. Promote international collaboration on AVs	Addressing cultural differences in driving norms to ensure AVs can adapt to these changes.	Provide local communities with the opportunity to determine how to integrate AVs	Promoting fair distribution of AV benefits and addressing potential negative social impacts.
Environmental	Adapts to different environmental conditions for safe driving	Minimizes environmental impact with lower green-house gas emissions	Empower customers to choose AVs that align with environmental values. Provide transparency about the environmental impact	Ensure environmental benefits are shared equitably including underserved communities.
Economic	Reduces overall energy consumption which reduces run cost of the car	Standardizes economic value of energy consumption for transportation	Allows customers to not have to worry about significant price changes in run cost of their vehicles.	Ensures economic standardization allowing for customers and the economy to not have significant marginal changes in operational costs for their vehicle

Figure 54: Four Principles

The nonmaleficence and public health & welfare pair is important to our team. Our design is focused on the safety of the users. One thing we will be doing is creating functionality that will stop and accelerate gently. This is just one example of how to keep the users of this car from getting injured. Our team will improve in this area by testing often and making sure that the car moves safely every time. The margin of error must be very small, especially in situations that may be unexpected corner cases.

7.3 Virtues

We have determined three virtues that are most important to our team. These virtues are patience, honesty, timeliness, and ambition. We have listed these virtues below along with our definitions and how we will support them.

- Patience: The ability to tolerate difficulty in design, programming, and collaborating with other teammates.
    - We will support this virtue by allowing a lot of time for design and encouraging taking a break when things feel frustrating. We will also be able to understand when problems may arise between team members.
- Honesty: The ability to bring a problem up to the team when any one of us is having a problem while working on the project or disagrees with another team member in a matter related to the project.
    - We will support this by encouraging everyone's opinions before making decisions. We will also strive to create an environment that is not judgmental.
- Timeliness: Striving to go to every team meeting and to get to them on time.
    - We will support this by attempting to set up meetings times that work for everyone.
- Ambition: The desire to achieve success in our project.
    - We will support this by encouraging everyone to share their ideas and what they personally want to project to look like.

Below, we all have reflected on our own personal strengths and weaknesses when it comes to these virtues.

Alex Crandell:

A virtue that I feel I have done well this semester is ambition. Most of our core work involves hardware and connecting sensors to the R car. While I am new to the electrical engineering field, the motivation to collaborate with teammates and complete the components remained throughout the semester. This is important to me since I usually would be nervous about completing tasks and letting the team down but newly found confidence and helpful collaboration has allowed me to succeed.

I would say my timeliness was my hardest virtue for the semester since we spent a decent amount of time gathering components and starting our core model of the RC car. Being more efficient and collaborating could help me become more focused and timelier. This is important to me for efficiency reasons as I do not like to procrastinate tasks and dwell on them. I plan on being present as much as possible in the next chapter of the course to maximize time efficiency.

Wesley Jansen:

Throughout this project, I have demonstrated both patience and ambition well. For me, this project has been a major challenge. It is challenging to have something so open-ended and no instructions on how to use anything; figure it out. With this being said, however, I am very interested in our project and bringing our ideas to fruition. My interest fuels my ambition and desire to produce a high-level product I am happy with. I have also run into a lot of issues with how the different components work together and at times was a struggle to not get upset and quit but I stuck with it and kept patient knowing I would have to get it done at some point. These two virtues have been important to me because, without them, I would have simply quit. I could ride the coattails of my teammates and get what I get, but since I have been patient with my abilities and have this ambition for our project, I have been able to stick with it and give helpful additions.

Something I have struggled with more this semester has been my timeliness. Since it has been a lot of new concepts or hardware, I have been working with things that have taken me much longer than I expected. I would tell my teammates I was going to get a certain part done during a week and set time aside to do it. A certain part I was going to do, however, would take a lot more time to complete than anticipated, leading to poor timeliness. For next semester I will make sure to be more reasonable with my abilities and give myself plenty of time for senior design to get my parts done on time and high quality.

Elizabeth Schmitt:

In this project I feel that I have demonstrated the virtues of patience and honesty. I do not have much experience with hardware implementation and design, so it has been quite a challenge for me. I have been very honest with my teammates about these feelings of confusion and my own shortcomings. With that, I have also exhibited patience. I have been patient with myself since I know I am still learning, and I will not be perfect right away. I have grown a lot in patience this semester through this class. I have learned not to be so hard on myself. I know that I can do this, it might just take a little extra time.

One virtue that I have struggled with while working on this project is ambition. Most of what we have been working on this semester is hardware. I do not have a lot of experience with hardware so all the stuff we have done has been very difficult for me so far. This has made it difficult to strive for success instead of just the bare minimum. I think I can work on this virtue more by learning the hardware better. This will make me more excited about the project. That excitement will ultimately lead to ambition.

Ben Towle:

One virtue that I feel I have demonstrated so far during senior design is the virtue of ambition. I find that this project is fun and interesting to work on. Therefore, I have noticed that I’ve taken a proactive role in our project implementation. I think it is important to find passion and drive towards projects like these so that during moments of struggle and hardship, you remember why you are going through it.

One virtue that I have been falling short on is patience with myself. A lot of times, I get frustrated with myself for not understanding things very quickly and that can lead to a lot more turmoil during my time working on a project. My goal for the next semester is to not overthink and panic when errors and moments of being stuck occur. It is important for me to remember that those moments are a part of the process and can help prepare you for the real world ahead.

Lalith Vattyam:

One virtue which I have demonstrated this semester is organization and participation. This project is a new experience for me as here was a lot of ambiguity at the start. Our group had to work together to come up with ways to complete all tasks and have our final project as a base for future projects to take further. During this phase I enjoyed planning and coordinating a lot as it would keep all of us on task to reach deadlines. We split our work into 2-week sprints which allowed us to reach the stage where we are now.

The project was also a new experience due to the fact that I didn't have much experience in raspberry pi or python and I wasn't very good at circuits. However, this project helped me a lot because it allowed me to learn from my peers and I'm now confident in understanding our solution and have a general idea for participating in discussions. Although it is very important to me, one virtue I find that is important to me but haven't demonstrated is patience. This virtue is very important to me as I hope to climb the corporate ladder in the near future. For this reason it is important that I always practice patience. During this next semester I plan on taking extra time to spend on the project and use my resources better to get my questions answered.

8 Conclusions

8.1 Summary of Progress

The results of our project have ultimately been a success. We have addressed our teams overall goal while only making minor changes to our initial design. We were able to get all the hardware to work together even with a team that has more of an expertise in software. We were able to power the Pi without having to plug it into the wall. We have a vehicle that moves autonomously through real world obstacles. Overall, we are extremely proud of the work we have done to make this project happen, despite all of the setbacks with hardware.

8.2 Value Provided

We set out on this project to address the problem of distracted and reckless driving on the road, we have done this by creating a small-scale vehicle that can be created into a large-scale car. It can get the user from one point to the other safely without getting into a crash. The one thing that is a drawback is pulsing forward instead of giving the user a completely smooth ride. In the broader context this vehicle can help everyone to be safer on the road. This design provides value by using cutting edge sensors and computer technology to address the real-world problem of bad driving.

8.3 Next Steps

This project has some future work that can be done. The next steps for the Race of Doom project include creating a new motor and implementing the RPLidar sensor into the design. A new motor needs to be created that can give the designer more ability to control the speed of the car. Right now, the motor is either off, or it is making the vehicle go very fast. This must be addressed before any more progress is made on the car to provide for a smoother ride. The RPLidar sensor should also be integrated into the design. The team will have to find a way to get more power to it from the Raspberry Pi5. This sensor will be so useful because it provides extremely accurate data about the objects that are all around the vehicle. We also already have the sensor; it should not go to waste.

9 References

[1] "14. API - Input Devices — gpiozero 2.0.1 Documentation," *Readthedocs.io*, 2015. https://gpiozero.readthedocs.io/en/stable/api_input.html#distancesensor-hc-sr04

[2] A. Nguyen, A. Gienger, B. Dubin, B. Carlson, and C. Tow, "sdmay24-06 • Race of Doom," Iastate.edu, 2024. https://sdmay24-06.sd.ece.iastate.edu/#weeklyreports (accessed Dec. 07, 2024).

[3] D. Coffin, S. Oliver, and J. VerWey, "Building Vehicle Autonomy: Sensors, Semiconductors, Software and U.S. Competitiveness," SSRN Electronic Journal, 2019, doi: https://doi.org/10.2139/ssrn.3492778.

[4] "Raspberry Pi GPIO Pinout," *pinout.xyz*. https://pinout.xyz/

[5] S. Pasricha, "Autonomous Vehicle Design – Sudeep Pasricha," *Colostate.edu*, 2023. https://www.engr.colostate.edu/~sudeep/autonomous-vehicle-design/

[6] Traxxas, "XL-5 Electronic Speed Control," Accessed: Dec. 7, 2024. [Online].

10 Appendices

Appendix 1 – Operation Manual

Installations

Before running anything on the Raspberry Pi, install the following:

1. sudo apt update
2. sudo apt install python3 python3-pip
3. pip3 install flask
4. sudo apt install python3-async.io (anyother library too) To use the ping sensors install the following:
5. pip3 install gpiozero

Hardware startup

If errors occur go to "Troubleshooting Hardware Section"

1. Connect ESC to battery with the big wires (ESC is blue thing near back of car.)
2. Turn on pi with the batteries attached on the bottom of the pi (button on side) - 4 green lights by battery should turn on, 1 green light on pi turns on.
3. Press button on ESC - make sure it is blinking green
4. Go to "Running Code on Pi" section

Troubleshooting Hardware Section

1. If ESC shows red light/does not turn on - charge big lipo battery with tool inside locker
2. If pi ssh connection goes off and pi lights red and turns off you must charge pi batteries with USBC cord.
3. Ensure all wires connected properly. Refer hardware design in design document.

Connecting to Pi

1. Plug HDMI into monitor and raspberry pi
2. Power Raspberry Pi
3. open terminal on monitor for raspberry pi
4. type hostname -I on monitor for raspberry pi

5. Locate IP address ex. 10.26.42.175 2610:130:110:1506:84F2:6B35:11AE:6484 ==== The IP address is 10.26.42.175

6. On separate computer ssh to raspberry pi ex. ssh sdmay25@10.26.42.175

7. Ask Team member for password.

Starting Flask Server

Navigate to the Flask server code directory:
cd /path/to/flask/server (should be wherever the project is) Start the server:
python3 camera_server.py

Running Code on Pi

1. Connect to Pi (explained above)

2. Once on pi navigate to code you would like to run

3. run "python3 <example.py>"

Changing Code on Pi

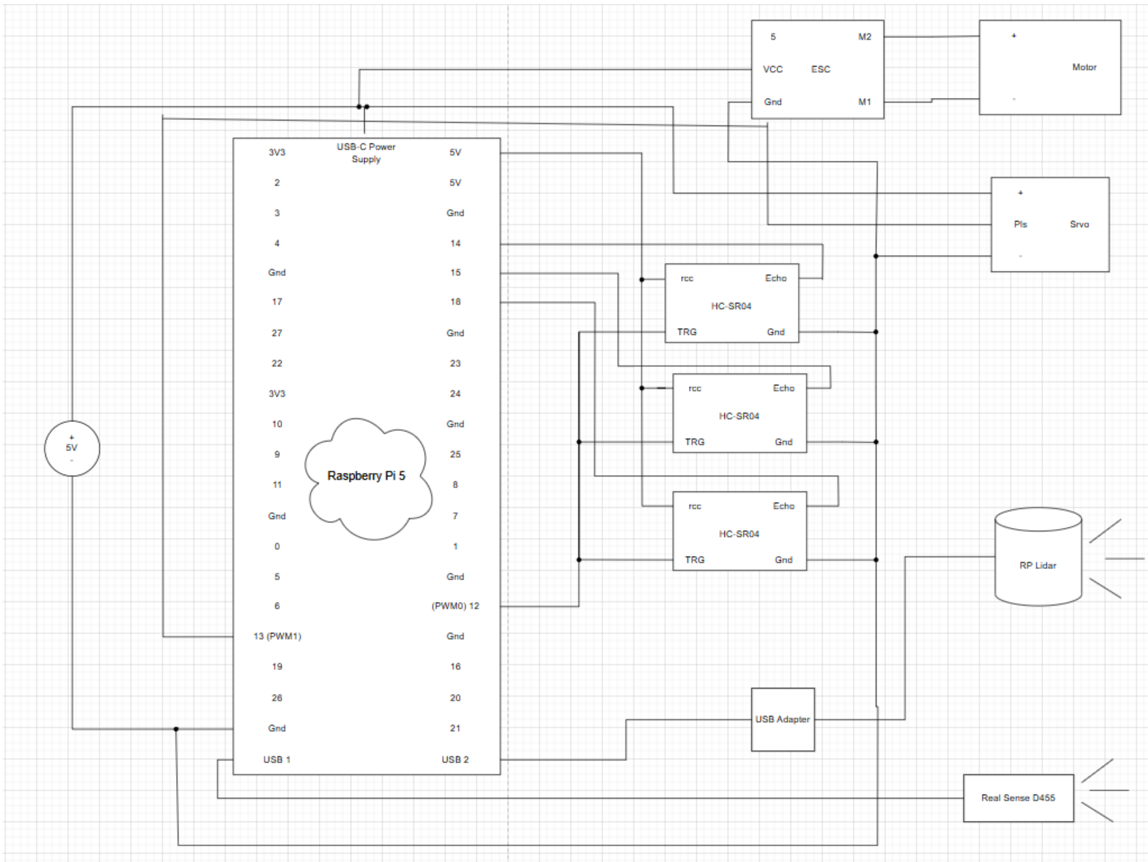
1. run "nano <example.py>"

Uploading New Code to Pi

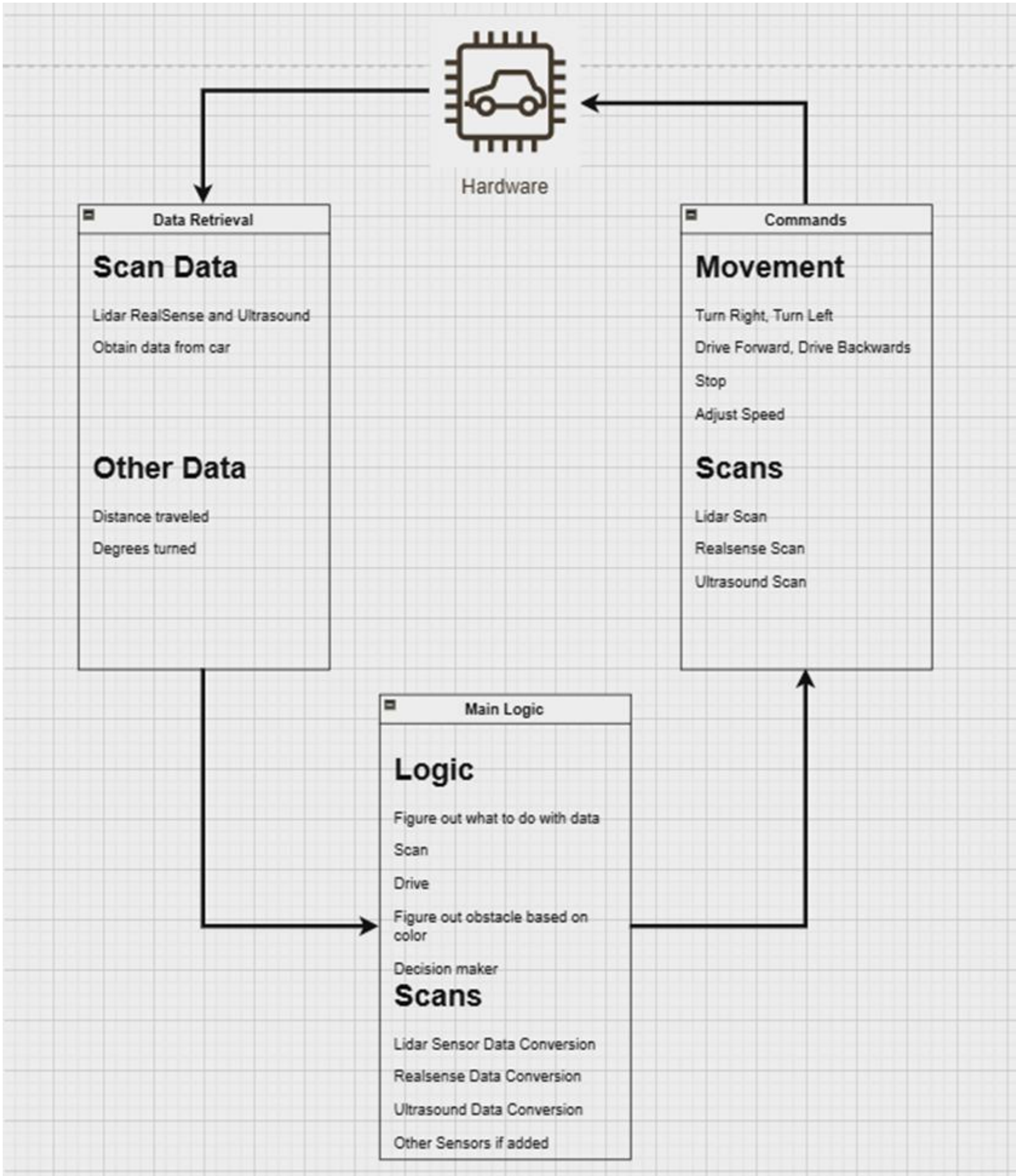
You have two easy options:

Option 1: SCP (recommended) From your local computer:
scp localfile.py sdmay25@10.26.42.175:/path/on/pi/
Option 2: Git Pull (if code is in GitLab) On the Pi:
cd /path/to/project
git pull origin main

APPENDIX 2 – ALTERNATIVE/ INITIAL VERSION OF DESIGN



Above is the original hardware design for our autonomous vehicle. This design has changed in a few notable ways. We used to have an RPLidar sensor but found that it requires too much power to the Raspberry Pi and it did not make sense for us to have it anymore. We also added an ultrasonic sensor to our vehicle. This addition is on the side of the car. It is used to determine what lane the car is in, and it can lane correct if needed based on the data received from this sensor. We also added the PCA Servo driver, which gives us more precises control of the ESC and Servo, which has proven extremely useful when it comes to turning. These additions were extremely necessary for the success of our car.



Above you can see the original software design. This design was created when we were not very far along in the design process. It was very broad and needed to be edited to have more detail. Overall, however, this design is good it just needs more detail on specific functions, like Figure 11 that is listed on page 27.

Another change that we made was in our team composition. Initially, we had 2 separate teams that would have different code and race each other. We decided that with the hardware issues we had and the nature of our project it made more sense to work together to have one quality product that moved safely through the course.

APPENDIX 3 – CODE

https://git.ece.iastate.edu/sd/sdmay25-08

There is a lot of code used for the movements of this car. A few snippets of this code is included below to give any idea of how everything works. Each of these programs are located in the SRC folder.

main.py:

```
import asyncio
import signal
from gpiozero import PWMOutputDevice, DistanceSensor, Servo
from functools import partial

from enums.Speeds import Speeds
from enums.Distances import Distances
from enums.Tasks import Tasks
from enums.Angles import Angles

from util.select_random_object import select_random_object
from util.lane_management.get_lane import get_lane
from util.lane_management.get_lane_status import get_lane_status
from util.sensor_print_format import get_and_output_sensor_data

from tasks.set_steering_angle import set_steering_angle
from tasks.forward import forward
from tasks.handle_generic_object import handle_generic_object_right, handle_generic_object_left
from tasks.back_up import back_up
from tasks.ping_object import get_average_distance
from tasks.handle_construction_cone import handle_construction_cone
from tasks.handle_construction_sign import handle_construction_sign
from tasks.handle_stop_sign import handle_stop_sign
from tasks.handle_crosswalk import handle_crosswalk
from tasks.handle_turn_right import handle_turn_right
from tasks.handle_turn_left import handle_turn_left
from tasks.stop import stop_for_1_second
from tasks.lane_correction import maintain_lane_align_right, maintain_lane_align_left
from tasks.back_out_of_wall import back_out_of_wall

from controller.webcam_request import get_object_info

# For servo control
from adafruit_pca9685 import PCA9685
import board
import busio

# Initialize Peripherals
esc = PWMOutputDevice(12, frequency=100)

left_sensor = DistanceSensor(trigger=23, echo=24)
right_sensor = DistanceSensor(trigger=17, echo=27)
center_sensor = DistanceSensor(trigger=5, echo=6)
wall_sensor = DistanceSensor(trigger=16, echo=26)

```

```python
i2c = busio.I2C(board.SCL, board.SDA)
pca = PCA9685(i2c)
pca.frequency = 50

flare_servo = Servo(18)

# Exit flag for signal handling
should_exit = False

# Car should start in an idle position
current_state = Tasks.STOPPED.value

# Tracks the lane that the car is supposed to be in
# True = left lane, False = right lane
assigned_lane = False

count = 0

# "Oh Shit" flag - if sensor values stay within a certain range for a certain amount of time, we're likely stuck
MAX_BUMP_COUNT_TOLERABLE = 2
bump_count = 0

# Map each state to its associated action
# I know I could have done this in the enum, but just get off my back alright
state_action_map = {
    Tasks.STOPPED.value: lambda: stop_for_1_second(),
    Tasks.COAST.value: lambda: forward(esc=esc, distance_cm=Distances.STANDARD_PULSE_DISTANCE.value),   #Can be plain forward as well, we'll see with testing
    Tasks.MAINTAIN_LANE_RIGHT.value: lambda: maintain_lane_align_right(esc=esc, pca=pca, wall_sensor=wall_sensor, assigned_lane=assigned_lane, left_sensor=left_sensor, right_sensor=right_sensor,
    Tasks.MAINTAIN_LANE_LEFT.value: lambda: maintain_lane_align_left(esc=esc, pca=pca, wall_sensor=wall_sensor, assigned_lane=assigned_lane, left_sensor=left_sensor, right_sensor=right_sensor, ce
    Tasks.HANDLE_STOP_SIGN.value: lambda: handle_stop_sign(esc, pca, left_sensor, right_sensor, center_sensor, wall_sensor),
    Tasks.HANDLE_CROSSWALK.value: lambda: handle_crosswalk(esc, pca, left_sensor, right_sensor, center_sensor, wall_sensor),
    Tasks.HANDLE_CONSTRUCTION_CONE.value: lambda: handle_construction_cone(esc, pca, wall_sensor),
    Tasks.HANDLE_CONSTRUCTION_SIGN.value: lambda: handle_construction_sign(),
    Tasks.HANDLE_GENERIC_OBJECT_RIGHT.value: lambda: handle_generic_object_right(esc=esc, pca=pca),
    Tasks.HANDLE_GENERIC_OBJECT_LEFT.value: lambda: handle_generic_object_left(esc=esc, pca=pca),
    Tasks.REVERSING.value: lambda: back_up(esc=esc, distance_cm=Distances.BACK_UP_DISTANCE.value),
    Tasks.HANDLE_TURN_RIGHT.value: lambda: handle_turn_right(esc=esc, pca=pca),
    Tasks.HANDLE_TURN_LEFT.value: lambda: handle_turn_left(),
    Tasks.BACK_OUT_OF_WALL.value: lambda: back_out_of_wall(esc, pca),
}
```

```python
def handle_exit(signum, frame):
    """
    Exit logic to terminate the program gracefully after a ctrl-c input
      -Updates the global loop flag to exit the main loop
      -Waits for the current task to finish up before cleanup

    """
    global should_exit
    should_exit = True
    print("\nSignal received. Preparing to shut down...")

# Register signal handlers to point to exit code
signal.signal(signal.SIGINT, handle_exit)
signal.signal(signal.SIGTERM, handle_exit)


async def cleanup():
    """Cleans up the GPIO pins and set periphreals to neutral values before exiting"""
    esc.value = Speeds.NEUTRAL.value
    await set_steering_angle(pca, Angles.STRAIGHT_WHEELS_DEGREES.value)
    await asyncio.sleep(1)
    esc.close()



async def execute_task(interrupt_callback):
    """Carry out the instance of current_task be calling its mapped function for key: State"""
    if interrupt_callback is not None:
        await interrupt_callback()
    else:
        print("Warning: No interrupt callback found for the current state.")
    await asyncio.sleep(0.1)


async def get_current_state_from_data(data, current_lane):
    """
        This function will take in the data recieved from ping sensors and decide which state to transition to
    """
    global bump_count
    global count

    try:
        # If we ran into wall twice, we know what to do...
        if bump_count == MAX_BUMP_COUNT_TOLERABLE:
            bump_count = 0
            return Tasks.REVERSING
```

```python

        # track how we are doing on maintaining our lane
        lane_status = get_lane_status(data["wall_distance"], assigned_lane)

        if data["object_detected"]:

            obj_distance = min(
                await get_average_distance(left_sensor),
                await get_average_distance(right_sensor),
                await get_average_distance(center_sensor)
            )

            if obj_distance < 25.0:
                print(f"Object too close,{obj_distance}cm reversing")
                back_out_needed = (data["center_distance"] > Distances.SAFE_DISTANCE_THRESHOLD.value
                                    and data["left_distance"] > Distances.SAFE_DISTANCE_THRESHOLD.value
                                    and data["right_distance"] < 40.0)

                if back_out_needed:
                    print("Backing out of wall")
                    return Tasks.BACK_OUT_OF_WALL.value
                else:
                    return Tasks.REVERSING.value


            print("Count ", count)
            object_type = await get_object_info()


            if object_type == "GENERIC":
                print("Handling Generic")
                if current_lane == -1: #In left Lane
                    if data["right_distance"] < 40.0:
                        return Tasks.REVERSING.value
                    else:
                        return Tasks.HANDLE_GENERIC_OBJECT_RIGHT.value #Turn right around the object
                elif current_lane == 1: #In  right Lane
                    if data["left_distance"] < 40.0:
                        return Tasks.REVERSING.value
                    else:
                        return Tasks.HANDLE_GENERIC_OBJECT_LEFT.value #Turn left around the object
                else:
                    if data["left_distance"] < data["right_distance"]:
                        return Tasks.HANDLE_GENERIC_OBJECT_RIGHT.value
                    else:
                        return Tasks.HANDLE_GENERIC_OBJECT_LEFT.value

```

```python
            elif object_type == "CROSS":

                print("Handling Cross")
                return Tasks.HANDLE_CROSSWALK.value

            elif object_type == "STOP":
                print("Handling Stop")
                return Tasks.HANDLE_STOP_SIGN.value

            elif object_type  == "CONSTRUCTION_CONE":
                print("Handling Construction Cone")
                return Tasks.HANDLE_CONSTRUCTION_CONE.value

            # elif object_type  == "RIGHT_TURN":
            #     print("Handling Turn Right")
            #     return Tasks.HANDLE_TURN_RIGHT.value

            # elif object_type  == "LEFT_TURN":
            #     print("Handling Turn Left")
            #     return Tasks.HANDLE_TURN_LEFT.value

        elif lane_status == 1:
            print("Maintaining lane by aligning left")
            return Tasks.MAINTAIN_LANE_LEFT.value

        elif lane_status == -1:
            print("Maintaining lane by aligning right")
            return Tasks.MAINTAIN_LANE_RIGHT.value

        elif data["center_distance"] < 20.0:
            print("Backing Out")
            return Tasks.REVERSING.value

        else:
            print("Coasting")
            return Tasks.COAST.value
    except Exception as e:
        print(f"An error occurred: {e}")
        return Tasks.STOPPED
```

```python
async def set_dynamic_parameters_to_function(function, data):
    global state_action_map
    distance_to_travel = min(
        data["left_distance"],
        data["right_distance"],
        data["center_distance"],
    )

    if function == handle_generic_object_right:
        print("Avoiding obstacle traveling this far to the right: ", distance_to_travel)
        state_action_map[current_state] = partial(
                                        handle_generic_object_right,
                                        esc=esc,
                                        pca=pca,
                                        distance_cm=distance_to_travel + 3.0
                                    )
    elif function == handle_generic_object_left:
        print("Avoiding obstacle traveling this far to the left: ", distance_to_travel)
        state_action_map[current_state] = partial(
                                        handle_generic_object_left,
                                        esc=esc,
                                        pca=pca,
                                        distance_cm=distance_to_travel + 3.0
                                    )


async def main():
    global should_exit
    global state_action_map
    global current_state
    global assigned_lane
    global esc
    global wall_sensor
    global bump_count
    current_task = None

    try:
        await set_steering_angle(pca, 90.0)
        await asyncio.sleep(1)
        current_lane = get_lane(wall_sensor=wall_sensor)

        while not should_exit:
            # Get Data
            data = await get_and_output_sensor_data(left_sensor, right_sensor, center_sensor, wall_sensor)

            # Update our lane tracker
            current_lane = get_lane(wall_sensor=wall_sensor)

            # Interpret the data and update the current state
            current_state = await get_current_state_from_data(data, current_lane)

            # Special case: dynamically define the callback's parameters based on data readings
            # Use case: if the object is on the left, we would like to turn right around it
            # That way, we don't risk running into the object
            # Likely More to be added later so... TODO
            if current_state == Tasks.HANDLE_GENERIC_OBJECT_RIGHT.value:

                await set_dynamic_parameters_to_function(handle_generic_object_right, data)
            elif current_state == Tasks.HANDLE_GENERIC_OBJECT_LEFT.value:
                await set_dynamic_parameters_to_function(handle_generic_object_left, data)


            # Assign the asynchronous task to that of the current state
            if not current_task or current_task.done():
                current_task = asyncio.create_task(execute_task(interrupt_callback=state_action_map.get(current_state)))
                await asyncio.sleep(0.1)


             # Await task if it's running
            if current_task and not current_task.done():
                try:
                    await current_task
                except asyncio.CancelledError:
                    pass

            await asyncio.sleep(0.2)

    except Exception as e:
        print(f"Unhandled exception: {e}")

    finally:
        # Cancel any running tasks during cleanup
        if current_task:
            current_task.cancel()
            try:
                await current_task
```

```python
    finally:
        # Cancel any running tasks during cleanup
        if current_task:
            current_task.cancel()
            try:
                await current_task
            except asyncio.CancelledError:
                pass

        await cleanup()


if __name__ == "__main__":
    asyncio.run(main())
```

actions/ping_object.py:

```python
import asyncio

async def get_average_distance(sensor, num_samples=5, delay=0.05):
    readings = []
    for _ in range(num_samples):
        readings.append(sensor.distance * 100)
        await asyncio.sleep(delay)
    return sum(readings) / len(readings)


async def scan_for_object(left_sensor, right_sensor, center_sensor, threshold):  # threshold in cm
    # left_distance = await get_average_distance(sensor=left_sensor)
    # right_distance = await get_average_distance(sensor=right_sensor)
    # center_distance = await get_average_distance(sensor=center_sensor)

    left_distance = left_sensor.distance * 100
    right_distance = right_sensor.distance * 100
    center_distance = center_sensor.distance * 100

    print(f"Left: {left_distance:.2f} cm, Right: {right_distance:.2f} cm, Center: {center_distance:.2f} cm")

    object_detected = (
        left_distance < threshold or
        right_distance < threshold or
        center_distance < threshold
    )

    print("Object?", object_detected)

    object_data = {
        "left_distance": left_distance,
        "right_distance": right_distance,
        "center_distance": center_distance,
        "object_detected": object_detected
    }
    return object_data
```

controller/get_ping_data.py:

```python
import os, sys
sys.path.append(os.path.abspath(".."))
from tasks.ping_object import scan_for_object

async def get_data(left_sensor, right_sensor, center_sensor, wall_sensor):
    """
        takes in the data read from ping sensors all at once. Once the data is read in main,
        we can decide which state to transition to
    """
    # Get the distance data from the 3 front sensors
    data_recieved = await scan_for_object(left_sensor, right_sensor, center_sensor)

    # Create a dictionary to hold the data
    wall_distance = wall_sensor.distance * 100.0

    # Append side sensor distances to entire data dictionary
    data_recieved["wall_distance"] = wall_distance

    return data_recieved
```

enums/Distance.py

```python
from enum import Enum

class Distances(Enum):
    """
        Distances used for vehicle navigation and obstacle avoidance.
    """
    SAFE_DISTANCE_THRESHOLD = 70.0 # Random objects on the road should be detected within this distance in cm
    STANDARD_PULSE_DISTANCE = 25.0  # The distance that the esc should go when it sees nothing
    BACK_UP_DISTANCE = 25.0
    LANE_CORRECTION_DISTANCE = 25.0

    # Define the appropriate distances between right side sensor --> Wall for each lane
    RIGHT_LANE_MIN = 7.0
    RIGHT_LANE_MAX = 45.0

    LEFT_LANE_MIN = 45.00001
    LEFT_LANE_MAX = 78.0

    #Minimum distance that the left and right ping sensors can see before object is out of its view
    MIN_PERIPHREAL = 35.0
```

tasks/back_up.py

```python
import asyncio
import os, sys
sys.path.append(os.path.abspath(".."))
from enums.Speeds import Speeds

async def back_up(esc, distance_cm):
    """
        Back up the car for a specified distance
    """
    print("Backing up for distance:", distance_cm)
    delay = distance_cm / 100.0
    esc.value = Speeds.BACKWARD.value
    await asyncio.sleep(delay)
    esc.value = Speeds.NEUTRAL.value
    await asyncio.sleep(0.15)
```

APPENDIX 4 – TEAM CONTRACT

Team Members

Alex Crandell, Wesley Jansen, Elizabeth Schmitt, Ben Towle, Lalith Vattyam

Required Skill Sets for Your Project

1. PCB Design - Alex
2. Python – Wesley, Lizzy, Ben, Lalith
3. Knowledge of Servo Motors, ESC, and Ultrasonic Sensors – Alex/Lalith
4. Raspberry Pi Applications - All
5. Version Control Software (Git) - All
6. Software Design, Development, and Testing – Wesley, Lizzy, Ben, Lalith

Project Management Style Adopted by the team

Our Team will be adopting the Agile Project Management Styles for this project.

Individual Project Management Roles

Alex – Hardware Designer/Tester

Wesley – SCRUM Master

Lizzy – Documentation Expert

Ben – Software Expert

Lalith – Project Manager

Team Contract

Team Members:

1) Alex Crandell

2) Wesley Jansen

3) Elizabeth Schmitt

4) Ben Towle

5) Lalitha Vattyam

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

   Team meetings are every Monday at 5:00 in Coover.

2. Preferred method of communication updates, reminders, issues, and scheduling

   Primarily face-to-face meetings, primary form of communication will be snapchat and emails

3. Decision-making policy (e.g., consensus, majority vote):

   Majority Vote after a discussion

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

   Ben will be the official timekeeper for the group. We will share the total and weekly minutes in our weekly reports.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

   All members are expected to be at every meeting with full preparation and participation. If a member is not able to make it, he/she is expected to communicate at least 1 day prior to the meeting.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

   Each member is expected to complete their assignment in their given timelines, if there are any concerns that are disallowing the member to complete the given task, he/she must make sure to communicate as soon as possible.

3. Expected level of communication with other team members:

   We expect the highest level of communication from our team members. As we are seniors it is expected that we make sure to always keep everyone in the loop.

4. Expected level of commitment to team decisions and tasks:

   Everyone's voice matters, all team decisions will be taken by a majority vote after a group discussion. After the vote, if a person is still not too happy with the decision we will open up for one more discussion after the vote which will lead to a second vote which will be the final deciding vote.

Leadership

1. Leadership roles for each team member:
   a. Ben – Client Interaction
   b. Lizzy – Testing
   c. Alex – Component Design
   d. Lalith – Team Organization
   e. Wesley - Team Representative
2. Strategies for supporting and guiding the work of all team members:
   a. Consistent Communication
   b. Openness to asking questions and listening to others
   c. Being open to go above and beyond to not only do your task but learn or help another person with theirs
3. Strategies for recognizing the contributions of all team members:

a. A pat on the back and a “job well done”

b. Team outing

c. Specific recognition in reports and public speaking events

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

    a. Lalith – Overall team organization and leadership skills along with understanding complex logic, and a professional background with cloud computing techniques and Azure solutions.

    b. Alex – EE expertise, with CPR E background knowledge giving a wholistic understanding of the project

    c. Ben – Complex logic expertise with high work ethic to finish task on hand. Has background knowledge of UI design and development as well as database administration.

    d. Wesley – Very high understanding of CPR E concepts with a commendable internship background in software packaging

    e. Lizzy – Solid CPR E understanding with significant knowledge in operating systems and kernel threads. Also has professional knowledge of C++ which is what most of the reused code is programmed in.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

    a. Whiteboarding

    b. Always reemphasize that there are no wrong answers

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

    a. Our team is solid already in terms of getting along and there is that comfort level among us to let each other know when the team environment would be obstructing their opportunity

    b. We are fully in agreement that communication should be easy and flowing amongst us and that the lack of such communication that someone may feel which would obstruct their opportunity would be something to be brought up to the professors.

Goal-Setting, Planning, and Execution

1. Team goals for this semester: Come up with a functional prototype (i.e. and R.C car with some type of autonomous movement)
2. Strategies for planning and assigning individual and team work:
    a. Our group will be split up into two sub-teams of 2 working on their own individual cars. Lalith and Ben will be on one team while Lizzy and Wesley will be on the other. Alex will be working alongside both teams by helping with hardware solutions for their respective R.C cars
3. Strategies for keeping on task:
    a. We will be using an excel sheet with a time and date log. This will have all our deadlines along with what parts and segments need to be done by what date to make sure that we finish our project on time. We will also be using 2-week sprints to keep a consistent flow of tasks getting done.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. We will have a meeting with the team member as a group and explain that further infractions will have to be reported to the professor and advisors
2. What will your team do if the infractions continue?
    a. We will contact the professors and advisors and ask about next steps

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the*

consequences as stated in this contract.

1) ________Lalith Vattyam__________________________________ DATE _________9/19/2024___________

2) ________Benjamin Towle_________________________________DATE _________9/19/2024___________

3) ________Alex Crandall___________________________________DATE _________9/19/2024___________

4) ________Wesley Jansen___________________________________ DATE _________9/19/2024___________

5) ________Elizabeth Schmitt_______________________________DATE _________9/19/2024___________