Envoy Commander

UNIVERSITY OF
CENTRAL FLORIDA

Department of Electrical Engineering and Computer Science

Samuel M. Richie, Dr. Lei Wei
Senior Design 1

Group 32

Yash Gharat	Computer Engineering	yash.gharat@knights.ucf.edu
Andrew Cuevas	Computer Engineering	cuevas.andrew@knights.ucf.edu
Anthony Soffian	Electrical Engineering	antsoffian@knights.ucf.edu

Project Customer/ Advisor/ Contributor

Dr. Chinwendu Enyioha

https://cenyioha.eecs.ucf.edu/
cenyioha@ucf.edu

Table of Contents

Executive Summary 1

Project Description 3

- Project Background 3
- Real-World Application 4
- Motivation 5
- Requirement Specifications 6
  - Hardware Specifications 6
  - Software Specifications 8

Research Related to Project Description 9

- Projects and Literature Review 9
  - A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond 9
  - Expertness Based Cooperative Q-Learning 9
  - Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm 9
  - SPARC: Self-Powered Automated Refuse Cart 10
  - Unsupervised Multi-Agent Exploration of Structured Environments 10
  - A Study of Maze Searching with Multiple Robots System 10
  - High Speed Arduino/Raspberry Pi RC Cars 11
  - The non-stochastic multi-armed bandit problem 11
  - Algorithms for the multi-armed bandit problem 11
  - Dealing with Noisy Data 12
  - Reinforcement Learning for Relation Classification From Noisy Data 12
- Problem Assumptions 13
- Proposed Tasks 13
- Relevant Technologies 15

Q-Learning 15
Communication Implementation 18
Arena Stimuli Consideration 19
Dummy Agent(s) Justification 20
Commander Agent Justification 21
Agent Progress Tracking 23
Project Technical Investigation 27
Comparison of Microcontrollers 27
Commander Construction 29
Dummy Agent Construction 30
Related Standards and Design Constraints 32
Related Standards 32
IEEE/ISO/IEC 29148-2018 32
ISO 17438-1:2016 32
IEEE 802.11-2020 33
Project Constraints 33
Economic and Time Constraints 33
Sustainability and Manufacturability Constraints 34
Environmental, Safety, and Health Constraints 34
Ethical, Social and Political Constraints 34
Project Hardware and Software Design Details 35
Choice of Language 35
Arena and Task 36
Task Detailed 37
Code Organization 38
Commander Agent 38
Dummy Agent 40

Hardware Sensors Discussion 40
KY-037 Sound Detector Module 40
HC-SR04 Ultrasonic Sensor 41
YL-99 Collision Switch 41
LDR Photosensitive Module (LM393) 41
Agent Progress Tracking Detailed 42
Communication 43
Abstract Software Design 45
Project Block Diagram 46
Project Prototype Construction and Coding 47
Prototype Communication Implementation 49
Printed Circuit Board 50
PCB Design 50
PCB Schematics 51
Microcontroller 51
Bi-Directional Logic Level Converter 52
Voltage Boost Converter 52
PCB Bill of Materials 53
Hardware Sensor Calibration and Tuning 54
Dummy Agent Sensors 54
HC-SR04 or TFMini LiDAR Sensor 54
HC-SR04 or TFMini LiDAR sensor Thresholding and Calibration Needs 55
Shopping Cart 55
Shopping Cart Thresholding and Calibration 55
Conventional 2 Wheel Drive 57
Quad Rotational 4 Wheel Drive 58

KY-037 sensor Thresholding and Calibration Needs 60
Sound Sensor Loud Buzzer Accommodations 62
Sound Sensor Soft Buzzer Assumptions 63
YL-99 Switch Calibration Needs 65
Orientation Module Thresholding and Calibration Needs 66
Exploring the TCS 3200 Color Sensitive Photoresistor 67
Calibrating the TCS 3200 Color Sensitive Photoresistor 67
Commander Agent Sensors 69
Potential KY-037 sensor Thresholding and Calibration Needs 69
Prototype Testing 71
Hardware Testing 71
Learning Agent Testing 71
Microcontroller/PCB Testing 72
Sensors Testing 72
Dummy Agent Testing 72
Movement Testing 73
Sensors Testing 73
Arena Testing 73
Software Testing 74
Q-Learning Algorithm Testing 74
Connectivity/Communication Testing 76
Initial Documentation Conclusions and Influences 78
Project Alternative 78
Additional Influence and Inspiration 79
Stepping into Reality 81
Possible Future Exploration and Research 81
Project Part 1 Pre Construction Conclusion 82

Administrative Content 83
House of Quality 83
Project Budget 84
Project Milestones 85
Appendix I-1
Sources I-1
Related Papers I-1
Projects I-2
Q-Learning I-2
Language I-2
Standards I-2

List of Tables

Table 1: Learning Agent Requirements 6
Table 2: Dummy Agent Requirements 7
Table 3: Environment Requirements 7
Table 4: Software Requirements 8
Table 5: Pros/Cons of Game Options 14
Table 6: Communication Map 19
Table 7: Raspberry PI Model 3B+ specifications 27
Table 8: Raspberry PI 4 Model B specifications 27
Table 9: Nvidia Jetson Nano specifications 28
Table 10: PCB Bill of Materials 53
Table 11: Types of RC-Chassi Implementation 55
Table 12: House of Quality 83
Table 13: Project Parts Lists 84
Table 14: Senior Design 1-2 Timeline of Milestones 85

List of Figures

Figure 1: Multi-armed Bandit problem 12

Figure 2: Frozen Lake Environment 16

Figure 3: Bellman Equation 16

Figure 4: Sample Q-table 16

Figure 5: N-Chain Environment Example 17

Figure 6: Pub/sub model with Access Point intermediary 20

Figure 7: Envoy Commander communication process 21

Figure 8: Possible Reward Function 22

Figure 9: Birds-eye camera configuration 23

Figure 10: LED Detection Method 24

Figure 11: Basic Electronic Toll Collection System 25

Figure 12: Visualization of Detachable Commander Agent Base 30

Figure 13: Visualization of Dummy Agent Schematic 31

Figure 14: Traditional Maze 37

Figure 15: Modified Maze 37

Figure 16: Class Diagram 39

Figure 17: Class Diagram 40

Figure 18: Conceptual Block Diagram (With Multiple Field Agents) 46

Figure 19: Design and Responsibility Block Diagram 46

Figure 20: Q-Learning Flowchart 47

Figure 21: Q-Learning Flowchart with Updated Decisions 48

Figure 22: EasyEDA PCB Layout of Commander 50

Figure 23: EasyEDA PCB Schematic of MCU 51

Figure 24: EasyEDA PCB Schematic of BSS138 Design 52

Figure 25: EasyEDA PCB Schematic of TPS6107 Voltage Boost Converter 52

Figure 26: Example of Calibrated Turn Distance 56

Figure 27: Alternative Sensor Design and Placement 57

Figure 28: 3 Sensor Model for Two Wheel Drive Chassi 58

Figure 29: Wheel Based vs. Axle Based Maneuverability 59

Figure 30: Stationary Axle Turn 60

Figure 31: Basic Noise Thresholding Visualization 61

Figure 32: Agent Loud Buzzer Simulation Example 63

Figure 33: (Novel) Critical Noise Level Diagram 64

Figure 34: Simple Reroute Visualization 66

Figure 35: Different Lighting Conditions 68

Figure 36: Adjusted Sound Offset 70

Figure 37: Open AI Gym Environment Setup 75

Figure 38: TCP vs UDP Communication standard 76

Figure 39: Testing Communication response 77

Figure 40: BattleBots arena 78

Figure 41: Texas Instruments Robotics Systems Learning Kit 79

Figure 42: Protograph Taken from UCF ASME Sumo-Bots competition 80

Executive Summary

One of the most classic reinforcement learning problems is that of the multi-armed bandit problem. In this problem, we have some learning agent that supposedly has multiple arms and is situated in front of some number of slot machines. Each slot machine is given its own distribution of success that is unknown to the learning agent. When each lever is pulled on a slot machine, the learning agent is given either a reward of 0 or +1 for failure or success, respectively. The learning agent wants to maximize the rewards it receives from the machines, without knowing exactly when each of the machines will result in a success. There are many solutions to the multi-armed bandit problem, but many of them are just simulations. In the Envoy Commander, we want to investigate solving a more complicated version of this problem using a real-world learning agent and a simple game.

The goal of the Envoy Commander is to create an intelligent learning agent that can seamlessly communicate with one or more dummy field agents to optimize a received reward when completing a simple task or playing a game. The learning agent should be able to add more field agents in as though they are plug-and-play and coordinate all of them to either complete the task better (receiving a better reward) or faster. To do this, it will act as a virtual commander to the field agents, using the observations they make through various sensors such as ultrasonic, photoresistors, and bumper switches to decide on the appropriate next action to take while maximizing its rewards. These sensors, as with any, come with some noise and uncertainty in measurements which are usually left unconsidered by using thresholds that will give a binary response (detected/not detected) and minimize the amount of false-positives or false-negatives. However, our learning agent should learn/estimate these uncertainties and use them as factors in its decision making (i.e. the most accurate sensor is taken as a priority). There might also be some uncertainty in the communication between the learning agent and the dummy agent and it should account for this as well.

To further define the aforementioned “learning agent”, we are expecting a microcontroller with a set of sensors that will be able to understand/track the progress of the field agents but only rely on their sensor observations to take actions from state to state. The knowledge of the progress will be used to determine the reward it receives and to know when it has reached the goal state. The learning agent will also need to wirelessly communicate with the field agents to receive state observations and send actions. With this, comes the problem of concurrency with messages, this will have to be handled efficiently such that the learning process and overall progress towards the goal is not hindered. The learning agent should also have enough computing ability to implement a reinforcement learning algorithm, which will be along the lines of Q-Learning, without too much strain as it will still need to continue to monitor/execute the other functions simultaneously. On the other hand, the dummy field agents only need limited capabilities, depending largely on the chosen task/game. For example, if the game were a maze, the dummy agents would only need to move in four directions and make observations with simple sensors. However, if the game were to sort colored blocks, the agents would need a camera to distinguish the various colors and be able to construct a color mask and send

those values to the commander, making it more complicated. In all situations, the field agents require the ability to wirelessly send/receive messages to their commander.

The learning agent will employ Q-Learning to learn from its environment. Q-Learning is a specific reinforcement learning algorithm in which the learning agent maintains a Q-Table with a value for each state-action pair. The Q-value (Quality) is dependent on the reward received for that state-action pair. In other words, it keeps track of the reward received for a specific action taken from a specific state. When the agent encounters the same state again, it will choose the action that previously gave the highest reward and continue on. However, using hyperparameters (parameters that tune the learning process), we can have some chance that the agent instead takes a random action so that it can explore other possibilities and not always be stuck in the same path. Without this probability, the agent is likely to always take the value it chose first, even if that was a random choice. The more states and episodes the learning agent endures through, the more it learns and optimizes the way it completes the task and plays the game. But since the algorithm is so reward-dependent, it is also important to choose a reward scheme that will push the learning agent towards efficiently and intelligently completing the goal instead of just trying to finish. A good example is the multi-armed bandit problem discussed earlier. Which only gave a reward with a successful lever-pull. Since it was trying to optimize the rewards from the slot machine, just choosing the best machine is useless if it did not give a winning pull. The agent is therefore not rewarded for just guessing the right machine. It has to choose the right machine and get a successful result in order to be rewarded, there is no partial credit.

Our supervisor/customer, Dr. Chinwendu Enyioha also has some suggestions regarding the overall implementation of this project. The primary one to be considered is the notion of a dynamic/stochastic environment. Since a real-world application for a learning agent does not necessarily guarantee a uniform, constant environment, it should not be given one. The learning agent should be able to finish the task with a very general set of constraints such that the design of either itself or its field-agent subordinates need not be changed. For example, just moving around/changing the maze would be ok, but adding a button to open a gate would not, since the field agent would need something to press the button with. The frequency of the environment changing and the extent to which it is changed can also vary.

While the Envoy Commander may not seem to have an immediate application, the techniques used in the solution can come into play in many settings. Package-delivery robots, militaristic strategy, and even simple daily tasks such as vacuuming. The key goal is to have a centralized, intelligent, learning agent that will coordinate other, unintelligent, field agents to perform a task in the most efficient way by using a combination of reinforcement learning, wireless communication, and sensor observations while being able to plug-and-play other field agents constrained only by the computing power of the central learning agent. Through episodic tasks and stochastic environments, the main agent should maximize its rewards and continue to learn without much help, if any at all, from a human agent.

Project Description

Project Background

As the world of machine learning and advanced AI grows, there is a general trend towards cooperative, web-like, systems of AI. Whether they are physical robots, distributed models, or federated models, there is a push towards machines working together to accomplish a given task in some environment. However, there is also a need for more affordable solutions, a complex network of AI can be rather expensive since all of the agents need to have complex processing capabilities, which could range from more expensive hardware or just better sensors for more accurate information. Either way it seems logical to just throw in some really good AI that can not only support themselves, but also intelligently cooperate when completing the task.

There are a few ways to go about this. As mentioned previously, there are distributed machine learning models that help with the usual scalability and efficiency issues that machine learning solutions come with. Instead of one machine taking all of the data to analyze and process, multiple machines work on the same data and model training to take care of the memory issue in a divide-and-conquer approach. Recently though, there has been a concern for data privacy issues so a new algorithm was devised. Known as federated machine learning, only the model parameters are stored by nodes on cloud servers and it is trained on a device, such as a mobile phone, and served to the user without a need for their data to leave their device. A common example is Google's GBoard, the mobile keyboard whose suggestions and swipe inference are cached locally.

Now the problem with distributed learning is that it not only requires many powerful machines to train the large dataset, it is also a solution to the problem of scalability. On the other hand, federated learning is good when there are privacy concerns and when the device the model is being served on can handle the load of training a model. Nowadays, mobile phones come with machine learning dedicated cores so this may not be an issue, But our project seeks to find a solution to a completely different problem, thus needing a different approach.

We want to solve the problem of scalability of machine learning systems while also lowering the cost of that solution. So our proposal follows a centralized machine learning approach. In this algorithm, the scalability problem works a little differently for a few reasons. Instead of having a strict, real-time algorithm, we are implementing a type of reinforcement learning known as Q-Learning. In this case, the models will be pre-trained based on a set reward function and exploring the problem beforehand. The benefit of this method is that the agents performing the task and cooperating need very little computing power, they only need to be able to communicate effectively with a centralized learning agent, known as the Envoy Commander, An added benefit is that this method is scalable as more "dummy agents" can be added to the communication set depending on the problem at hand. This project is more a demonstration of centralized Q-Learning agents applied to a problem that needs scalability. To demonstrate this, we will set up a maze-like game for the agents to escape from and they will use onboard sensors and communication to send information to the centralized Q-learning agent which will return the action for them to take based on the reward function optimization.

Real-World Application

Multi agent learning and similar communication concepts can be applied daily in a variety of walks of life with human interaction to guide many smaller IOT (Internet of Things) based communications. Examples of such are military and police based evolving technologies to deal with hostile situations that use smaller robots such as bomb-defusal and conflict resolution type robots, that previously required a human to work in place of what is taken care of by remote controlled robots. In the modern world, these robots usually come in the form of quad-copters for information gathering, or motor powered ground vehicles that run on wheels or tracks to handle situations that require more finesse or control. These robots come with a variety of functions ranging from information gathering such as assessing the situation of a crime, or more complicated activities such as information distribution and handling hostile human parties. Other immediate IOT activities that can implement a network of simple communication can be a scenario such as smart home securities and systems such as the ring doorbell. For example, many modern security systems automatically alert the authorities should a certain set of requirements that may be detected as breaking and entering are detected for higher income houses. These kinds of security systems can be extended to scenarios such as natural disasters, where something as simple as an advanced thermostat may be able to detect disaster such as fire in certain parts of rooms and assess the situation before having to go through an intermediate operator such as police dispatch, and therefore respond to situations faster. Eventually, technology of this caliber may be able to respond to danger from sets of pre-based protocols, and eventually form their own solutions for scenarios based on massive sets of information from multiple houses.

Expanding on the idea of applying Q-learning to a multivariable system can be simplified to a group of machines working towards a goal, and this idea can always be applied to various levels of integration for almost any aspect of everyday life. The problem in question, however, is the implementation of how we simulate large scale problems such as these on the smaller scale to be expanded upon in the future. In the modern world, these projects are multi-million dollar industry problems that work with millions of people with millions of scenarios to consider. The modern scale deals with systems that play with lives at stake such as automated hospital response systems and bomb-defusal robots, where our scale deals with simple games and situational awareness. Modern robots, while more sophisticated in function, lack the artificial and networked-internet-of-things expanded communications that we are trying to employ on a smaller scale. In the event that we can display successful situational awareness and understanding of simple tasks such as games, then we may expand from the goal of a system being to solve a maze, to finding a human in a burning building with a team of rescue robots. After all, serious problems to humans are games and puzzles to be solved for computer systems.

For real world implementation, modern robots in use today are equipped with sensors that range from being able to sense through materials (infra-red heat detection and other visualization methods) to simple tasks such as sound detection. In our game set, the main focus of implementation will be to successfully work towards, but not necessarily complete, the goal. In the real world, systems of robots aren’t always successful, and this process is part of learning, whether this learning is for humans, or in our case, robots. Start small, and expand.

Motivation

We wish to unionize computer science and engineering to create a machine learning solution for a hardware-constrained problem that uses data from noisy physical sensors to coordinate efforts in a common task.

As a group we have collectively spent time learning about machine learning through various classes offered at UCF such as Robot Vision, Algorithms for Machine learning, and related coursework and wish to proactively apply some of the knowledge that these classes have to offer. Being able to see knowledge being gained through classes manifest itself in real world situations and apply abstract principles in a way that is uncommonly represented is invigorating, and quite frankly part of the challenge. By presenting ourselves with the task of designing something that could in theory be split into multiple projects over a much longer time span, and seeing it come to life at the speed of which it is being designed is invigorating. Understanding that this project is a potential stepping stone into a much larger and relatively unexplored market where very few other engineers have the same kind of specialty and cross understanding, grants our team perspective and drive to dive just a little deeper than the average team, because it may be the difference between who achieves something great and who just rides their degree through.

Another great motivation that pushes our group is the lack of pre-existing implementations of q-learning at this scale. While only exploring fundamental q-learning principles, this type of reinforcement learning is rarely targeted with anything that isn't pure code or theoretical, so proving that it can be practically implemented to some degree can open pathways for more modern implementations of AI that go deeper than numbers and search algorithms. Some modern applications of artificial intelligence include improving shopping suggestions on Amazon or by monitoring websites for common scam red flags. By expanding reward functions and optimizing them for what humans consider desirable, Q-learning implementations are a stepping stone for doing just that. While this project is relatively low budget and simple when compared to the artificial intelligence and big corporation level of money and complexity, it works as a stepping stone and a potential motivator for the wider development of robots. Robots with higher functionalities would eventually be able to begin optimizing tasks such as saving people in burning buildings or different ways to perform surgeries that can deal with complications in different ways.

While these are tasks that are already explored by traditional artificial intelligence and robots, the difference is where the optimization comes from. Q-learning has the unique ability to learn directly from situations that are happening on the fly, and while using traditional methods to initialize a robots learning process are still necessary, improvised learning makes up for some of the shortcomings of robots, and begins to introduce the very human characteristic of creativity to machines.

By presenting creativity and problem solving to machines, while machine surgery exists, intelligence shows up very infrequently in any robot implementation. While not yet the science fiction level of robot intelligence making decisions, being able to make decisions where humans still determine what the machine works towards is a big step towards discovering solutions that humans might not be able to discover on their own.

In summary our group is convinced that this project sinks its fingers into something more than just itself and potentially opens doors for new technologies and possibilities. By allowing ourselves to take a little bit of risk and try something new we are able to rise to

the challenge of this relatively challenging project and area of study, and are excited to see where we can go with it.

Requirement Specifications

The specifications are split into two main categories, the hard and tangible requirements, and the design requirements for the game that is to be played by the learning and field agents in question. Hardware specifications are likewise split into smaller sub-categories based on where they apply in the larger system, and have been simplified from previous versions of this document.

Hardware Specifications

The main hardware categories include the learning agent (which does field interpretation), the field agents (which supply information to the learning agent), and the environment (which is the information to be given to the field agents)

Table 1: Learning Agent Requirements

1.1	The learning agent must explicitly take steps to solving the goal as outlined by the software design, 75% of the time.
1.2	The microcontroller must be able to complete and acknowledge tasks using a maximum of 4 Gigabytes of memory
1.3	The microcontroller must make decisions, defined by using explicit approaches determined by the software design
1.4	The learning agent must be able to intake specific stimuli and quantify the environment as numeric data (light level, distance to objects)
1.5	The learning agent must be able to operate within a maximum range of 50 Ft.
1.6	The learning agent must be able to operate with a minimum range of 5 Feet
1.7	The microcontroller must be able to operate with a 3.3V logic level for a minimum of 20 minutes.
1.8	The Learning agent should be cheap, comprising less than 25% of the project budget.
1.9	The learning agent should be able to consider information from a minimum of two sources at a time.
1.10	The Learning agent must be able to communicate with a smarter machine such as a computer, for the minimum operating time suggested below
1.11	The learning agent must be able to operate for a minimum of 10 minutes at a time
1.10	The learning agent must be able to sustain ranged communication for a

	minimum of 10 minutes
1.11	The learning agent must be able to convert stimuli data into commands for a minimum of two agents.

Table 2: Dummy Agent Requirements

2.1	Field agent sensors must be able to quantify data from the environment as numeric data (sounds, pressure sensors)
2.2	The field agent must support a variety of sensors including but not limited to ultrasonic sensors, bumpers, and photoresistors for a minimum of 10 minutes
2.3	The field agents should be mobile, as to move in a two dimensional plane
2.4	The field agents should be reasonably small, and not exceed a size of one square foot
2.5	The field agents should be able to communicate at a range of at least 5 feet.
2.6	The field agents should be able to operate a 3.3V logic level system for a minimum of 10 minutes
2.7	The field agents should be able to operate a 5V motor system (of wheels or tracks) for a minimum of 10 minutes.
2.7	The collective cost of all the field agents must not exceed more than 50% of the project budget.

Table 3: Environment Requirements

3.1	The arena must be stochastic and provide at least 3 different scenarios
3.2	The arena must scale in weight to the size of the agents, and currently stand < 50 lbs for 5x5 ft^2)
3.3	The environment must be able to provide mechanical stimuli (buzzers, sensors) for a minimum of 10 minutes
3.4	The environment must be able to withstand impacts from moving field agents within reason (impacts measured as the field agents moving at 50% power.
3.5	The environment must be able to implement a game as described in the software design
3.6	The arena must cost within 25% of the final project budget

Software Specifications

Table 4: Software Requirements

4.1	The software must explicitly define two types of agents, learning and field
4.2	The software must interface and communicate data to the master controllers defined as the project managers
4.3	The software must show the project managers that information is being transmitted between the agents whether faulty information or accurate
4.4	The software must respond to the stimuli it interacts with and make decisions whether the information is accurate or not, this will be defined as learning
4.5	The software must understand the information being passed to it, and quantify it as variables.
4.6	The software must define a game and success conditions for the learning agent
4.7	The learning agent must receive information from the field agents
4.8	The learning agent must provide instructions that indicate that a goal is being moved towards
4.9	Success and positive stimuli must be quantified and measured, given their own units and rates
4.10	The software must react to positive stimuli, changing its approach to be non-linear
4.11	The software must be able to make new decisions not previously designed, based on numbers that were not previously determined (AI-Aspect)
4.12	Non-interpretable data must be categorized as tso, and responded to in a similar light

Software specifications are still variable due to the nature of the game being debated and the goals at hand still undefined. These will be more specific as the rules and nature of the game are narrowed down.

Research Related to Project Description

Projects and Literature Review

A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond

While this paper mostly talks about Centralized vs Distributed vs Federated learning and not our project directly, the comparison between these methods, the pros and cons of them, and the applications are great discussions for our paper. Also the discussion of how Federated learning is replacing Centralized learning due to the growing privacy issue and increasing computational power of IOT devices. Federated Learning won't be useful in our case because we want to minimize the costs of the dummy agents to make them scalable. However, we could investigate Distributed learning more because there is no need to assume the dummy agents can't do anything at all. Maybe there is a way to do some processing on the dummy agents to improve the fidelity of communication. The less data we need to transmit, the less errors, also the faster the learning process.

Expertness Based Cooperative Q-Learning

This is somewhat in line with our project. Agreed, there is only one central learning agent for us and this presents a multi-agent problem. But what if we took a different approach to our problem? What if each dummy agent kept track of their own Q-tables and with every new state, they sent the Q-Value to the commander packaged with the sensory observations. The commander then would choose the best agent, assign a new QValue for the (S, a) pair and that's how learning would happen. Obviously there are problems with this that would need to be thought of. But this is again along the lines of Distributed learning vs Centralized learning.

Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm

This paper again presents a multi-agent cooperative Q-learning framework for stochastic games. This is quite similar to our problem except for the fact that we are trying to have a single agent solve this problem without the help of another. But at least this paper introduces the concept of stochastic games, especially static ones, that operate in discrete space, like our maze problem. They prove that when Q-learning is used in this problem, it can converge (under some given circumstances). They discuss issues in Q-learning vs other frameworks such as Markov Processes.

SPARC: Self-Powered Automated Refuse Cart

This project has a main focus outside of the scope of what our plans have been, but there are some elements, mostly involving identification and communication which we were able to extract some information out of to utilize for our project design. One component of our project involves the learning agents having to understand how to optimize their escape, accomplished by opening and closing tunnels, and this was accomplished by reading through a section of SPARC which detailed some options they discussed on positioning when the concept of RFID stakes or identification markers of some form to see where their garbage bin was in relation to their environment. For our use, we can utilize this feature by implementing an RFID tracking system to allow the learning agent to get a reward whenever it passes under this tunnel system as an exploration benefit. Another idea concept they discussed was a pathfinding algorithm called Markov Decision Processes which they implemented to add some autonomy to their project, and under the same umbrella our Q-Learning algorithm will utilize a different set of parameters, quality-based, to accomplish our arena challenge, but this different system allowed for some clarification on other variations of algorithms.

Unsupervised Multi-Agent Exploration of Structured Environments

The authors of this paper discussed a concept regarding the nature of agents carrying out navigational tasks in an unknown environment on a layered level, which is an advanced version of our current design on navigating a one dimensional maze. The concepts that provided benefits to read from was assigning each section of the maze into departments that could potentially be given a role to a specific agent to navigate if we chose to increase our amount of envoys to deploy. These agents would have to identify the sections on their own and be able to develop a sense of their location relative to each other to remove potential “bad” routes for the remaining agents going along their pathways. Further into their algorithm they also created an exploration script that allowed an agents to determine its routes from a certain position in the maze, and utilizing this knowledge they can choose to continue exploring a potential route and communicate with its fellow agents by sending out occasional transmissions that would help avoid repeated pathways and provide knowledge sharing techniques by broadcasting a message of a new route and keeping track of the time it last spoke with a nearby agent to then share the knowledge it learned from the last time they “spoke”.

A Study of Maze Searching with Multiple Robots System

This paper utilized similar methods to the previous, running the maze with a CDAC algorithm instead, but used the concepts of trailer markers and simple communication methods as well as a more specialized approach to labeling the agents, giving them tags that coincided with functionality like wall, stay, and last robots, all following their own statuses to collaborate more effectively with its peers. They chose to use one of the simple robot structures to navigate their maze that contained paralleling features such as touch sensors, photo reflectors, and with them used a very simple 4-bit communication network for one exchange between the agents. The design of their arena we look to take inspiration from as we begin its construction, as we look to label the specific turns, sections, and for our project specifically “secret” tunnels for our envoys to learn how to interact with and reach the end goal at the highest reward state.

High Speed Arduino/Raspberry Pi RC Cars

For both of these projects, one from an Arduino and the other from a Raspberry Pi, they utilized methods to create simple rc cars that would be controllable by their respective MCU which we could take inspiration from to build our learning agent systems. From the start the idea of creating our agents from RC parts was a possibility due to our possession of test subject readily available that would not require additional orders to be processed, and from these two projects, the Arduino one in particular, it gave us some comparison to draw from on how we should plan to design our agent to commander system. Even though the Raspberry Pi design was more prebuilt, the logic still transfers over to the more custom one where some elements are somewhat unnecessary for a design such as ours, the knowledge of the amount of designs available makes the possibility of our design coexisting in a maze with a few adjustments to collect more data more of a realistic goal of making them interact as actual collaborative envoys.

The non-stochastic multi-armed bandit problem

The multi-armed bandit problem is one that is closely related to our collaboration problem. The usual setup is that some gambler has access to many slot machines that independently have some distribution for a payout. This means that the machines won't win at the same time but it is possible, given a large number of trials, to understand when the payout is more or less likely to payout. Q-learning is often a great approach to this problem, as are Markov Decision processes because, we can either make assumptions about the slot machines and use those to approximate the distribution, thus giving us a good way to set up either the reward function or the transition probabilities. This paper, however, takes no assumptions regarding these slot machines but instead takes it a step back. In their problem, they instead assume that each slot machine is initially assigned an arbitrary and unknown sequence of rewards which gives no hints as to the actual distributions of payouts. The reason this paper helps is because, like in our maze, the rewards to be generated are unknown and not able to be estimated by the learning agent. However, they are also not completely deterministic since we want to make the maze stochastically dynamic so that the rewards will change as we change the maze.

Algorithms for the multi-armed bandit problem

Again, with the premise that the multi armed bandit problem is similar to our maze problem, we wanted to examine the different approaches that can be taken to solve this problem, since it is a common one. That is exactly what this paper does, and they not only discuss different algorithms that can be taken, they also measure and compare the performances of these algorithms against each other. They tended to find that various algorithms perform rather similarly until the number of arms and the distribution of rewards changes. So some algorithms work better in some contexts, this reiterates the point that machine learning solutions are very specific to the problem. In our case, we will probably not be exploring other algorithms, since Q-learning will work a little better and we don't have any understanding of transition probabilities and want to stick with the model-free approach. However, this paper did show the importance of laying out necessary assumptions when solving this stochastic problem, so that is something to discuss at a later point.

Figure 1: Multi-armed Bandit problem

Dealing with Noisy Data

Noise is prevalent in all forms of noise and it is important that we not only try are best to limit how much noise affects the data, but also how we deal with it when we make decisions and construct measures from it. Machine learning applications are severely affected by this noisy data since it can cause bias and this can cause points of failure for the solutions. Since no sensor is perfect, our project will also have to deal with noisy data in many forms. Whether it may be communication, sensor measurements, inaccurate motion, or even just data processing. We should identify these points of noise and do our best to fix the issue.

Reinforcement Learning for Relation Classification From Noisy Data

This paper examines relation classification in a bag of sentences but explains how current methods assume that all data provided is meaningful and describes relations. In reality, there is noise in data such that if classification is performed at the “bag level” it will suffer from the noise in the data. So the researchers propose a modular model that first attempts to choose high quality data dn then train the model on that data, thus filtering out the noise. The score from the classifier is passed back as a reward for the reinforcement learning model and the cycle helps improve what data to select. This effectively deals with the noisy data and gives overall better performance. We can learn from this in our own project at a lower level. Instead of using machine learning to filter out our noisy data, we can examine the data manually and determine what is meaningful from the sensors and through various plots, maybe even improve the data from which we are learning. Since this project is to be scalable, it is important to reduce the amount of computation needed by the commander agent. A great way to accomplish this at the root level is to reduce the noise in the raw data, so less processing is needed.

Problem Assumptions

As with any complex AI/ML problem, there is always a set of assumptions that are made in order for the problem to be solvable. There is some information which would be needed in an ideal scenario which is just not possible, given the time and budget

constraints. The first assumption we are to make is that the information from any sensors we use is accurate and only our scaling and thresholding of the data will cause errors. In reality, no matter how complex or expensive they are, sensors produce noisy, imperfect data. So our thresholding and scaling may limit the amount of noise that affects our decisions and masks but it will not be able to give us a perfect measure.

Another assumption we make is that our dummy agents can move at one uniform step at a time. At least in the current iteration of the problem, we update the Q-table at every action taken by the dummy agent and it receives another action to take. There are no doubt some issues here because we are using nonuniform RC cars as our dummy agents that differ in speed, size, and more importantly, turning radius. This means that some agents will need to take more actions to get around turns. There may be ways around this assumption if we control the power associated with the motors in the cars so that they all maintain some constant velocity. However in a real-world application, the company is not likely to use various dummy agents but instead they will use uniform agents. This is why it is safe to assume that the agents will take the same step everytime.

At this current stage in the problem, one more crucial assumption we will be making is that the Raspberry Pi 4 will not hinder the learning process. This is a really big assumption because there is a lot of load it has to undertake between the Q-learning, communication, data scaling, and perhaps arena operation. These are clearly a lot of tasks but given the beefy nature of the Raspberry Pi 4, it should be able to handle this load, although some thread optimization may be needed. The backup, if it is not able to handle all the tasks, is to offload the data scaling and arena operation to a smaller, cheaper, Raspberry Pi Zero. This MCU costs around $5 so it will not affect the budget too much and it can handle these simple tasks but will add another link in the communication chain, something we would like to actively avoid. For now though, until we are able to test it, it may be better to assume that it can handle the load to keep the project moving forward.

Proposed Tasks

There are many different applications for our environment detection project, and such that we can create many different games for our dummy agents and “envoy commander” to process, and here we have found some feasible ideas that could potentially be implemented by using this baseline of questions to choose, such as is this game feasible for our agents to accomplish? Is the complexity enough to be considered as a finalist for our overall project? Should the puzzles be considered accomplished basically on completion or would a partial credit be given to our system if not entirely finished? Do we focus on the ability of the agents or its relation with our tower? Answering these questions will be a rolling task as the majority of our focus is on the environment sensing ability itself rather than the medium which we choose to use this technology, so nailing down a final choice is not in our best interests at this time. Regardless, the options discovered so far is as follows, the concept of a escape room triggered by incentives and most likely switches to be activated, using actual known puzzles such as sudoku, using a trash-collecting concept to utilize the tower ability between our agents and the “tower” to locate and move pieces of “garbage” to a specified area of the environment, counting/ordering different objects similar to actual jigsaw puzzles, and instead of puzzles

actual games that can be represented in a simplified form such as following a line leader, tag, and red light/green light. Each of these possibilities bring forth potential avenues to use our algorithms and how they are handled is discussed further in their more descriptive advantages and disadvantages.

Table 5: Pros/Cons of Game Options

Tradeoffs of Potential Project Tasks/Games		
Potential Games	**Pros**	**Cons**
Escape Room (5)	● Most complex, best test of project requirements ● Explicit boundaries and rules to be followed in step-by-step process ● More freedom of creation	● Potential complexities outside of scope for project ● Additional costs to mechanical/electrical components ● Unusable failures possible
Known Board-Based Puzzles (2)	● Predictions possible ● Simple physical requirements ● Guaranteed a solution ● Simplest use of agents	● Limitations of agents involvement, tower-focused ● Less deviation from game
“Trash” Collector (1)	● Most realistic test of project requirements ● Utilizes agents to full extent ● Random nature ● Goal-oriented as intended	● Require pushing/pulling actions from agents ● Another potentially over-complex concept
Object Identification/Sort (4)	● Simplest method ● Cost benefits ● Focus more on degree of complexity to environment detection algorithm	● Possibly too simple for SD ● Sorting requirement from agents, so more physical actions required ● More objects needed for time
Simple Children Games (3)	● Similar to puzzles, mostly solvable and set rules ● Recognizability of actions ● More involvement from agents ● Open to failure from “players”	● More movement needed from agents to complete ● On a tier below escape room for learning complexity, above puzzles and trash

Relevant Technologies

Q-Learning

Q-Learning is a popular model-free, value-based reinforcement learning algorithm. We can begin by understanding this definition, but first a little background on reinforcement learning. Reinforcement learning is a type of machine learning training that uses a reward function to either reward or punish a learning agent that is able to observe its environment, or state. In general, learning algorithms work in state-action pairs, (*S, A*), such that for each state, an action is taken to transition the agent to a new state, *S'*, where it will make new observations. The agent's choice of action could be policy, or value-based. Policy-based algorithms rely on taking the action that tends to end them in the best result. Given a state, *S*, the agent wants to determine what action, *a*, will allow it to maximize its reward by understanding where it will transition to. The policy, π, is the probability of taking *a* when in *S* given a set of observations. Based on these probabilities, the agent will greedily pick such that it will end up in the best state, *S'*, and then update the policy. However, Q-learning is the latter, value based learning algorithm. It picks the action based on an equation that updates some value for (*S, A*) independently of the agent's prior actions/policy. Q-learning is also model-free, meaning it does not use a "model" for its environment which is usually a probability distribution representing the transitions between states. For example, If I have just finished running, I am most likely to drink water or shower and not nearly as likely to put on a tuxedo (an extreme I know). However, that's not to say that it wouldn't ever happen, just a really low probability. But Q-Learning doesn't need to understand this. All it knows is that when it takes an action from a state, it receives a value representing a reward and it wants to maximize this reward to reach its goal. After all, the reward function should be created such that it rewards progress towards the goal and punishes anything else.

To further expand on Q-Learning, I would like to discuss more on the nature of its value-based table, known as the Q-table. The Q-table contains a value for each (*S, a*) that the agent could visit, so given an state-space size of N and an action-space size of M, the table is of dimensions (N *x* M). A common example used to represent this is OpenAI's Frozen Lake Environment.

> *"Winter is here. You and your friends were tossing around a frisbee at the park when you made a wild throw that left the frisbee out in the middle of the lake. The water is mostly frozen, but there are a few holes where the ice has melted. If you step into one of those holes, you'll fall into the freezing water. At this time, there's an international frisbee shortage, so it's absolutely imperative that you navigate across the lake and retrieve the disc. However, the ice is slippery, so you won't always move in the direction you intend."*

```
SFFF       (S: starting point, safe)
FHFH       (F: frozen surface, safe)
FFFH       (H: hole, fall to your doom)
HFFG       (G: goal, where the frisbee is located)
```

Figure 2: Frozen Lake Environment

So the learning agent can take 4 actions, and visit 16 spaces so the Q-table is of dimension 4x16, or 64 values. In this example, the reward function is also very simple, the agent only receives +1 when the goal, *G*, is reached and otherwise gets a reward of 0. Everytime the agent "falls into a hole", the environment resets and the next "episode" starts. In each episode, the agent maintains knowledge of its previous attempts through the Q-table, but it starts from the beginning in a fresh environment. This is the structure of most reinforcement-learning problems. The value for the Q-table is updated using the Bellman Equation:

$$V(s) = max_a(R(s,a) + \gamma V(s'))$$

Figure 3: Bellman Equation

This equation states that given the reward, *R(s,a)*, a discount factor, γ, and the value of the next state, *s',* we want to update the Q-table with the maximum value we can receive. γ represents the agent's focus on immediate vs long term rewards. This largely depends on the problem at hand, such as the Frozen Lake example where the reward is only received in the goal state. With every time step, the agent takes an action, and then updates the Q-table with the new state and reward it received.

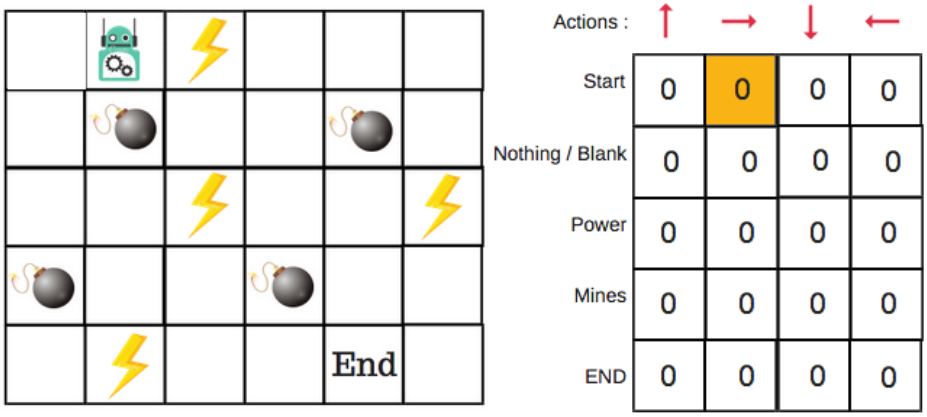


Figure 4: Sample Q-table

Additionally, in the episodic nature of Q-learning there is the concept of exploitation vs exploration. As mentioned previously, reinforcement learning tends to take the greedy approach as a dynamic-programming model. If the agent knows the best action to take in the given state, it will take that action. However, what if it hasn't discovered the action that gives it the best value yet? This is where the ε factor comes in. When deciding what action to take, we can either exploit by taking the action that has the maximum Quality, or we can

explore and take a random action independent of the prior policies. Adjusting ε determines how often we decide to explore. Again, this threshold is determined by the needs of the problem at hand. In some cases, the immediate reward is not the one we are after. One example of this is the N-Chain environment from OpenAI:

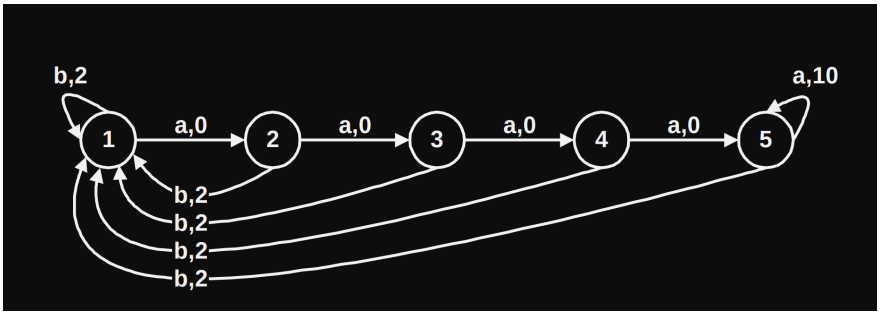


Figure 5: N-Chain Environment Example

In this environment, the agent receives 0 reward for stepping forward but +2 for stepping back in a linear chain of nodes. In this particular example, there are 5 states and 2 actions. Most notable, is at state 5, where the agent receives a reward of +10 for simply looping at that node. So the point of this environment is to train the agent to sniff out the long term reward at the end instead of focusing on the +2 for going back. If ε and γ are not adjusted appropriately, the agent may never escape from the first state, instead choosing to infinitely loop there to receive the +2 reward since stepping forward does not give any reward at all. Eventually, as the disparity in the Q-values increases and the agent is more confident in the best actions to take, this expiration factor should decrease such that it explores less and exploits more, depending on the needs of the problem, of course.

Q-Learning's major benefit is that the agent needs no true prior or current understanding of its environment. In a completely unknown environment, it only needs to be able to make observations of its surroundings, know what actions to take, and receive a gamified reward for taking an action. This has many benefits since the reward function can be designed to meet the needs of the problem and can be tweaked without too much effort. However, on the flip-side, this reward function still determines the learning agent's behavior so it should be created carefully with the problem in mind. There are two general types of reward functions, discrete and continuous. In the examples discussed thus far, the reward functions have been discrete, meaning that the reward is attached to the action taken and the state landed in (+10 at the node 5 or +2 for going back). A continuous reward function may instead use continuous metrics to offer a reward. Some examples could include, time, distance remaining, or even battery left. Since these are not as easily quantified in discrete terms, we would need to relate them in some way to have the agent understand that he needs to maximize it. For example, as the battery percentage dwindles, if we are still not at our goal, we might logarithmically reduce the reward at each time step as a function of the battery percentage. As a response, the agent might exploit more to reach its goal faster. A good reward function will make or break a learning agent.

Communication Implementation

This section will cover the needs that are specific for how the Q-learning can be achieved with the agents and arena that will be constructed. Parts will be discussed briefly at the end of this section, but will be discussed in detail in the Project Prototype and Design Section further into the paper (See Table of Contents)

In order to establish a matrix for Q-learning that implements the nodal dummy to commander agent model that we are trying to achieve, at minimum the commander agent and the dummy agents must be able to send and receive one way communications, meaning that the robots do not necessarily need a synchronous channel to communicate between each other. The additional potential alleys for communication include the stimuli that are to be implemented within the arena. Because these stimuli do manipulate what the dummy agent(s) will send to the commander agent, these are also considered one way communications as well although there is the absence of a microcontroller or relatively intelligent process.

Conceptually, the Q-table is updated whenever an action is taken, and these actions are kept in a table to assess whether the action taken is likely to lead towards a good reward or not. This lookup table, for practical implementation, is updated whenever possible, and this leads to a potential interrupt style communication method. Additionally, the possible methods of communication are endless, but will be truncated into focusing on three generalizations

- Open communication = Communication channel is opened between parties and exchange of information is synchronous and constant
    - Benefits
        - Most stable and most reliable
        - Can establish a waiting queue for communications to deal with one dummy agent or task at a time.
        - Least error handling required
    - Detriments
        - May be more complicated than required and may delay performance
        - Tasks and matrix modifications may take time from the dummy agents that the dummy agents need not wait for.
- Single Response = (TCP style) Messages are sent and replies are generated and received before continuing function
    - Benefits
        - Secure and quick, single tasks and responses guarantees that a queue of communications can interact with multiple dummy agents at a time.
        - Relatively reliable, interrupts for the agents result in less possibility for synchrony related issues.
    - Detriments
        - Requires a party to wait for a response

        - In the event of lost communication, may result in infinite delay
        - Error handling required
- One Way = (UDP style) Messages are sent, and assumed to be received
    - Benefits
        - Quick and leaves the commander with the most computational power to deal with queue (fastest method overall)
        - Reduces synchrony problems related to sending messages
        - Increases synchrony problems with relation to coexisting send requests
        - Least complicated to implement in theory
    - Detriments
        - Most error handling required
        - Least secure method
        - Difficulty in implementation scales the worst.
        -

Table 6: Communication Map

Project Component	**Outbound Communications**	**Inbound Communications**	**Method Considered (sender)**
Arena Stimuli	Dummy Agent(s)	None	One Way
Dummy Agent(s)	Commander Agent	Commander Agent	Single Response
Commander Agent	Dummy Agent(s)	Dummy Agent(s)	One Way Open Communication

Arena Stimuli

Due to the nature of reinforcement learning, the time constraints of this project, and the need to demonstrate collaboration, we decided to have stimuli in the arena that will guide the learning agents either towards the goal or towards a set of shortcuts. This could be accomplished using color sensors and strategically placed LEDs around the maze, RFID tokens, or sound detection via strategically placed buzzers. The importance of these stimuli is to optimize the time the agent will take to learn the maze. If no hints are given at all when navigating the maze, the agent will take too long to learn and issues would arise with running the process overnight. Stimuli, once the agent picks up on them from the set reward function, will act as those guiding hints. For example, everytime the dummy agent passes a RFID checkpoint in some quadrant, it will get a massive reward which will increase based on the sequence of the checkpoints. This way, it will seek out the next tag. Similarly if we were to consider the colored LED approach, everytime it ran into a green LED, the learning agent would receive a larger reward depending on how many it has already found.

Clearly, the RFID checkpoint has its merits over the LED method but if we were to combine the approaches to represent different things, maybe the LEDS will be placed near the aforementioned shortcuts, then they would provide for a good reward structure. Most

of the choice boils down to available pins on the Arduino after the motors are connected via the motor shield and the wifi module is also connected, since those are the necessities in its operation.

Dummy Agent(s)

The Dummy Agents are meant to be affordable, replicable, and scalable, that's why the Envoy Commander is designated as the learning agent to take on the load of making decisions through Q-learning and optimization. The Dummy agent will then have only two functions, it will communicate its state observations to the commander and then take whatever action was designated to it. Thus reliable communication capability is key to the success of this project.

Though there are many ways to implement good, able communication, the one that has seemed to make the most sense in this project is to use WiFi with a TCP implementation on a local wireless access point (WAP). In this implementation, the dummy agent will take some action and land in some state. It will then make some observations using its sensors, and package this data, along with the Q-values associated with its current state and transmit these values to the WAP that can be hosted with an ESP8266 module, perhaps the Nodemcu 1.0, and listen for a request back. The Commander agent ewill be listening on the WAP's port and will process the request and send back the appropriate response. Using this pub/sub model using IPs to allocate the information will allow for efficient information transfer and better space optimization. We only need to send the important information.

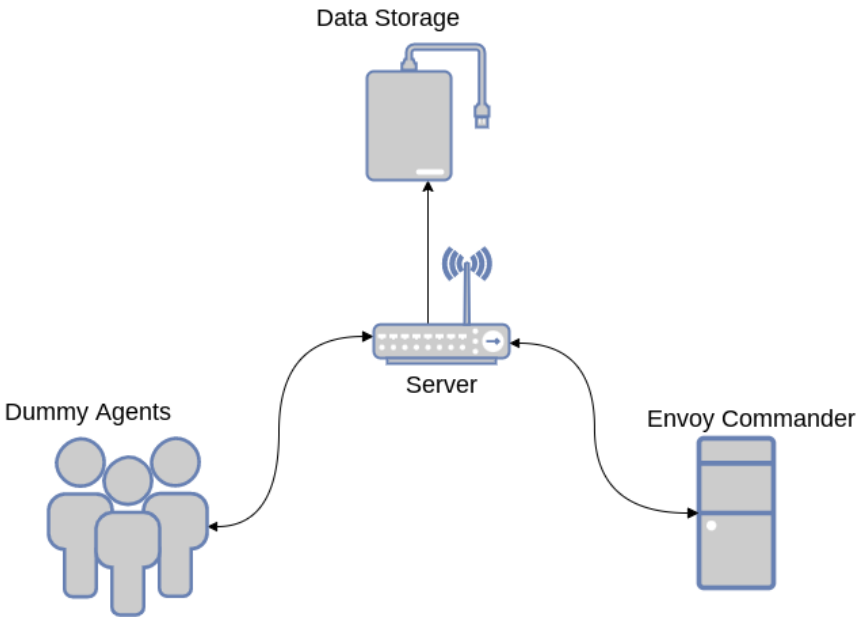


Figure 6: Pub/sub model with Access Point intermediary

Commander Agent

The Envoy Commander will be the highlight of this project since it will be doing all of the heavy lifting. This includes processing the sensory data that is sent from each Dummy

Agent and scaling it such that it will be used as state observations in the Q-Learning process. It will also have to handle multiple responses from various dummy agents and process them accordingly so that they may reach the global goal at the end using local decisions.

To reiterate, the Envoy Commander will be the base of the reinforcement learning. It will receive observations, process and scale the sensory data, choose an action, and return a new Q value along with the chosen action for the appropriate dummy agent. The process will look like the following:

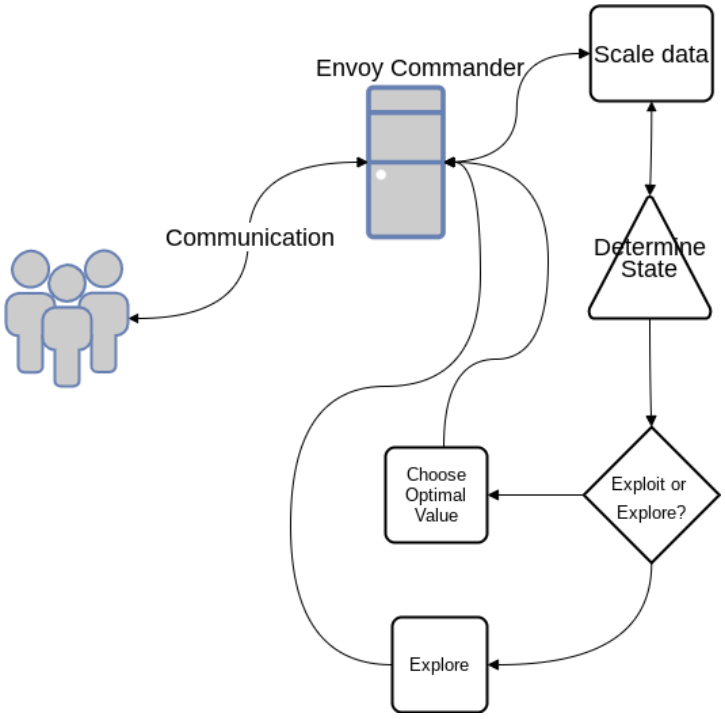


Figure 7: Envoy Commander communication process

To further facilitate this, the learning agent will receive rewards from an abstracted system (the PCB built out). In this method, the PCB will be an omnipresent agent that knows the progress and of the dummy agents in relation to the goal and the possibility of shortcuts nearby but will keep this information hidden from the learning agent, only providing what is needed for learning truthfully. Conceptually there are holes caused in scaling this project due to this, but given the time constraint, this is the best way to go about this. Using either the RFID checkpoints, or some strategically positioned camera to gauge distance, we want to properly track the dummy agents' progress. This secondary sensory information will also be scaled to use for the reward function:

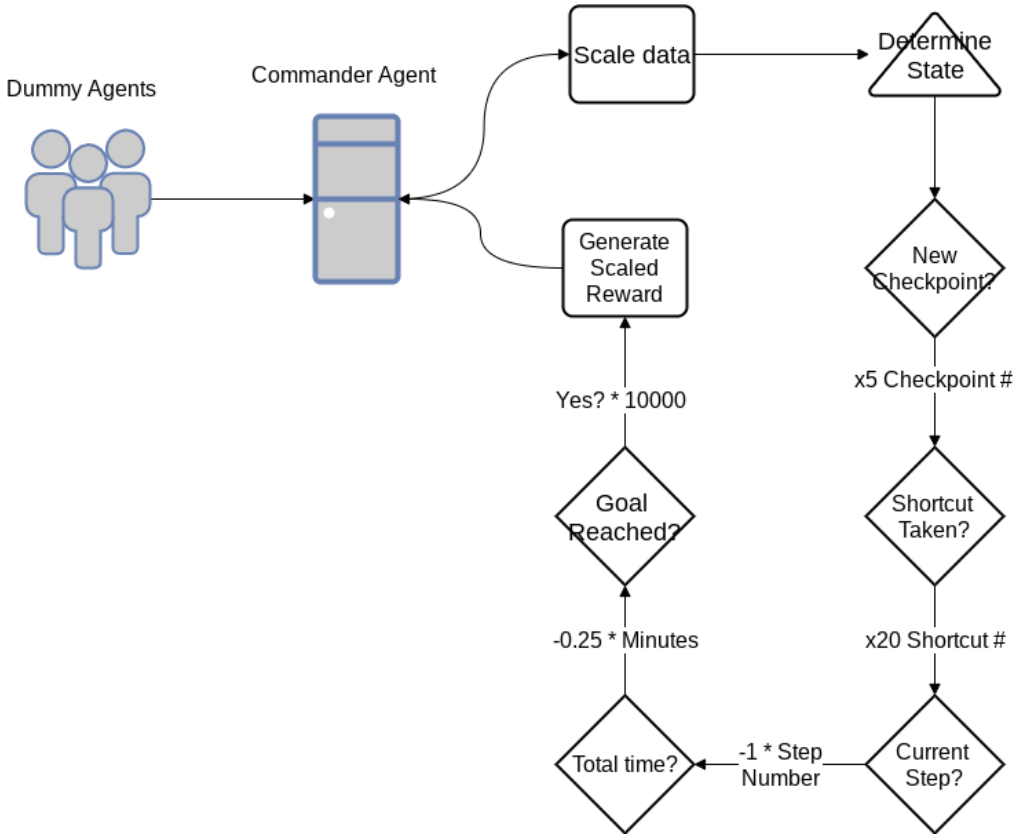


Figure 8: Possible Reward Function

Choosing the appropriate reward function is crucial for this project to work. To simplify the process overall, we have elected to go with a discrete function that will depend on the various sensory data received from the dummy agents as well as the progress tracked by the abstracted PCB. In order to convert continuous sensory data into discrete observations and rewards, we will employ the use of binning the data into set thresholds. It is still difficult to say this for sure because we are not yet sure how the data from the sensors will look. The point of the reward function is to incentivize the appropriate actions. That is why the possible reward function outlined above, rewards the agent every time a positive step is taken towards the goal. The shortcuts are extra incentivized with a huge margin of profit because we want to make sure that when they locate the shortcut, they should optimally take that every time. On the other hand, we want to punish taking unnecessary steps and extra time. Also, when they reach the goal, we want it to have the biggest reward possible because that is where they should end at. After all the rewards are given based on the progress towards that goal, so shortcuts will be valued highly and so will the checkpoints. This is an example of a discrete reward function. A continuous reward function would instead be taking all of these factors as inputs into a function and outputting a continuous value that represents the value of these instead. The downside to this is making sure the scale games have the proper incentives, that is why I have preferred to choose a discrete reward function.

Agent Progress Tracking

Logistically, one of the more difficult aspects of this project, although far less crucial, is the method we will be using for the abstract PCB to track the progress of the dummy agents in the maze, thus also tracking the learning agent's progress towards the goal. This progress will be used to appropriately dole out rewards to the learning agent in order for it to continue learning. To reiterate, the reason that Q-Learning was chosen in this case as a reinforcement learning algorithm to train the agent is because it is model-free. It was better to train the agents and have them learn from the surroundings with no prior knowledge instead of needing training data or an understanding of the transition probabilities in the environment, which are basically even depending on the state-space. But how do we track the moving agent in maze and relay that information to the Raspberry Pi 4 to leverage for reward generation?

The original method that came to our mind, and probably anyone else who takes a first glance at this project is to hook up a camera from the tower, looking down, that would give a birds-eye view of the whole maze and would be able to track where the agents are. In an ideal situation, this would be a great method.

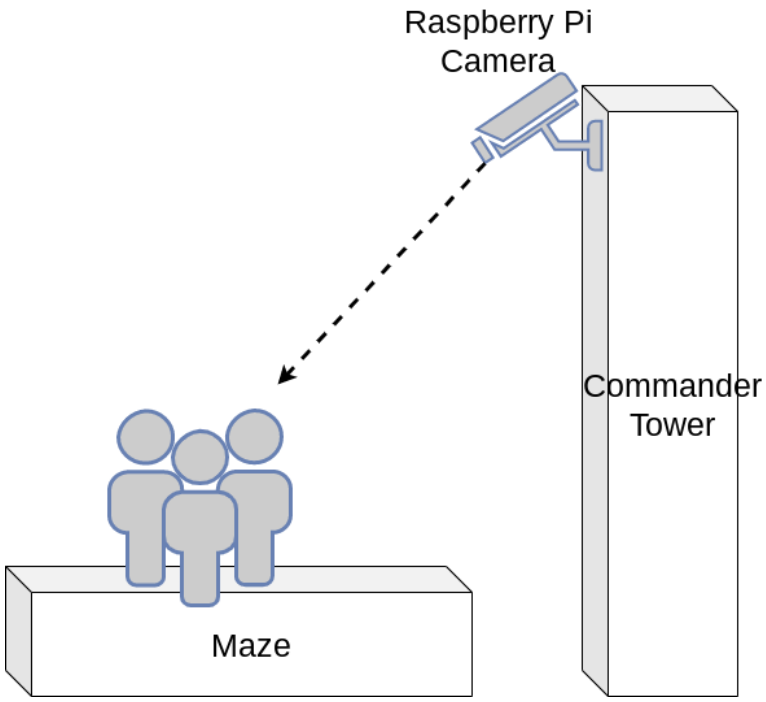


Figure 9: Birds-eye camera configuration

However, as with most Raspberry Pi Camera problems, many issues start to arise when planning out the logistics. First and foremost are typical thresholding issues when dealing with cameras. Popular software such as OpenCV require masking on every frame of the video output it gets and that frame, whether object detection or color detection is based on some set threshold values. If we have too low a threshold however, we may not

properly capture the object as it passes the camera, especially if it moves too fast. On the other hand, if the threshold is too high, we might miss the object as it passes the camera. So through trial and error, a proper value must be found with a good balance of detection and filtering. Another issue with the camera method is the location and configuration of the cameras. Ideally we would want a birds-eye view of the maze so that we can properly capture the moving agents. But how would we different between the agents? Dummy agent 1 and dummy agent 2 would both appear and the collaboration piece would fail. Also, in this location, there is a lower chance of detection for any entity anyways. Improving the object detection would be a project in itself and add at least another month to the timeline to make sure it's perfect. Another, less obvious configuration for the cameras would be to place one at every spot in the maze that could be considered a checkpoint and relay that back. There are so many issues with this configuration as well. At first, there is the price of that many cameras which we clearly don't have the budget for since it would almost triple, if not more, the cost of the project. But even if the cost were not an issue, there is still the problem of power, both computing and electrical. The Raspberry Pi 4, while a significant improvement over other MCUs in its class, is not equipped to be able to handle image and video processing from this array of cameras. This makes this option not scalable and even more so not feasible in the time frame and budget we are given. So we decided to explore other options.

The immediate next option we chose to explore is that of LED color detection using the TCS3200 module for the Arduino on each dummy agent. The clear benefit of this is the price point of this. LED cathodes/anodes are very cheap and easy to wire. So if we were to place these around the arenas for the dummy agent to locate and detect using the color sensor, we could have different colors to represent different paths in the maze and a unique color for the shortcuts. Again, the obvious downsides to this method are very similar to the camera module. For this method to work properly, we would either need a very good color sensor that isn't noisy, a dark room, or be very close to the LED. Getting a good color sensor is somewhat impossible in this scenario because we are on a tight budget and will use the standard arduino TCS3200 module which is particularly noisy but very cheap. So our other options are to get very close to the sensor or work in a very dark room. Both of these are possible actually. Unless we end up using photoresistors to detect some sort of light, which is not likely. A dark room will help reduce the noise related to the color sensor since the LED cathode will be the only emitting light and it will be really easy to fixate on this for the color sensor as it navigates. Since the dummy agent will mostly be relying on ultrasonic sensors to move around in collaboration with the learning agent, it really doesn't need light in the room to move around.

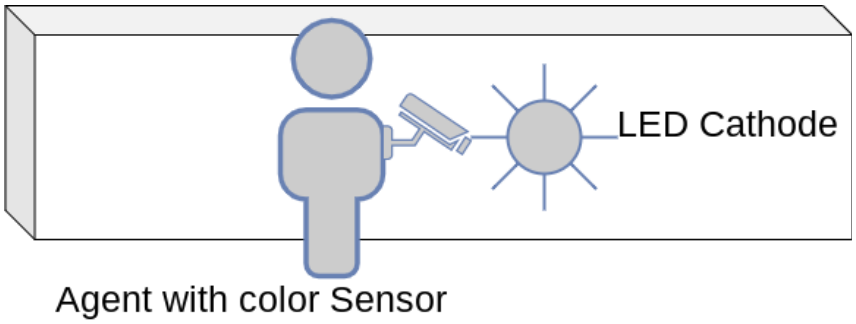


Figure 10: LED Detection Method

Another way we could reduce noise for the color sensor, would be to move the dummy agent closer to the actual LED. This is a bit harder to achieve since it would involve two different options. We could either make the walls of the maze narrower and more fit to the scale of the dummy agents so that the LEDS are basically right next to it or we could also extend the LED out on some contraption to be closer to an aligned color sensor on the dummy agent, assuming it can only move in one orientation as planned. The problem with this method is that the sensor may very well be bumped into by the dummy agent and possibly break the rod to which it is attached. So in this case, simply narrowing the walls and limiting the turning radius of the dummy agent. One more issue that arises in this context is the overall wiring of the LEDs. It is still unclear whether they would all connect to some central Arduino Uno or not and planning this out, depending on the scale of the arenas could cause many problems.

A better and more novel idea we wanted to explore is that of miniature toll booths around the maze as checkpoints. Toll booths such as the *SunPass* and *EasyPass* use an active RFID transponder that communicates with some reader as one drives past and the license plate is grabbed with a picture and read as well. Using the combination of these methods, they are able to identify the car and appropriately charge the account. In some cases, the gate won't even open until the toll is paid by cash or the card in the car. There have been some showings of using the basic RFID reader that ships in many arduino kits to act as a card for the tolls.

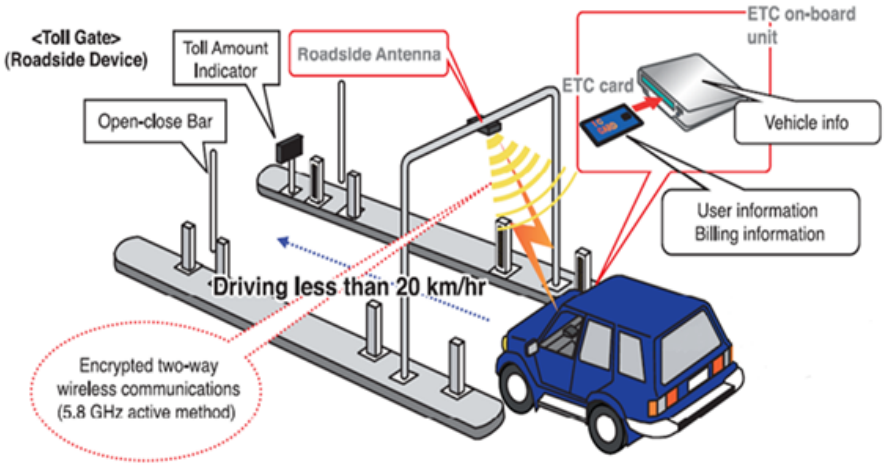


Figure 11: Basic Electronic Toll Collection System

In all of these examples however, the car is first detected with an ultrasonic sensor and then the researcher taps the reader with the RFID tag to pay for the toll. This is because of one crucial issue. Due to the frequency of the RID reader, the tag can only be read from ~6cm away. Basically, the card has to be tapped on the reader or held near it in order for it to appropriately register, if even at all. To get longer range readers, the budget would have to be increased. To read a tag about 50cm away, it adds around $20 and for distances ~1m it increases to the $100s range which we are not willing to allocate in our budget for this project. Initially though our idea was to place an RFID reader on each dummy agent so that when it passes by a token placed on the wall, it could glean

important information from each tag regarding the location. This info could be packaged along with the sensor data that is already being sent and will accurately track the progress of the dummy agents. However, the distance for the reader significantly puts a pin in this idea unless some way could be leveraged so that the RFID tag with high accuracy, perhaps putting it on the ground as it drives over it.

The last idea, and perhaps the most feasible is to place ultrasonic sensors that read basic distances. They could initially be calibrated to face the wall and that could be zero point. If the value changes from that sensor, we could report it to the abstracted PCB to use for the reward function. Since the values would change for only one ultrasonic sensor at a time, we could accurately track the progress of the dummy agents. The downside to this method is that we would not be able to identify which dummy agent is passing through so collaboration would be difficult, but to make this easier on us, we could place the dummy agents in different locations so that it will be easier to tell which agent is passing. However, this idea still has to be researched more and is very dependent on the shape and configuration of the maze since power and wiring will still be an issue.

Project Technical Investigation

Comparison of Microcontrollers

Table 7: Raspberry PI Model 3B+ specifications

Price	$70
CPU	BroadCom BCM3827B0 ARMv8 Cortex-A53 64-bit SOC @1.4GHZ
RAM	1GB LPDDR2 SDRAM
Power Input	5V/2.5A DC
Connectivity	2.4GHz and 5GHz 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2
GPIO	https://www.etechnophiles.com/wp-content/uploads/2020/12/Raspberry-Pi-3-B-Pinout-in-detail.jpg
Dimensions	85.6mm x 56.5mm
Notable Features	4 USB 2.0 Ports Full-size HDMI CSI Camera Port for Raspberry Pi Camera Power-over-Ethernet support with PoE HAT

Table 8: Raspberry PI 4 Model B specifications

Price	$92
CPU	BroadCom BCM2711 **Quad Core** ARMv8 Cortex-A72 64-bit SOC @1.5GHz
RAM	**2GB, 4GB, or 8GB LPDDR4-3200 SDRAM**
Power Input	5V/3A DC **via USB-C** or GPIO header
Connectivity	**2.4GHz and 5GHz IEEE 802.11ac wireless LAN, Bluetooth 5.0, BLE** Gigabit Ethernet
GPIO	https://www.etechnophiles.com/wp-content/uploads/2021/01/R-Pi-4-GPIO-Pinout.jpg
Dimensions	85.6mm × 56.5mm
Notable Features	2x micro-HDMI Ports (4K support) 2-lane MIPI CSI camera Port 2 USB 3.0, 2 USB 2.0 Power-over-Ethernet support with PoE HAT

Table 9: Nvidia Jetson Nano specifications

Price	$99
CPU	**Quad Core** ARMv8 Cortex-A57 64-bit SOC @1.4GHz
RAM	**4GB LPDDR4-1600**
Power Input	5V/2A DC
Connectivity	Gigabit Ethernet M.2 Key E
GPIO	https://www.element14.com/community/servlet/JiveServlet/downloadImage/38-32473-714539/870-687/JetsonNano-expansion-pinout.png
Dimensions	100 mm × 80 mm × 29 mm
Notable Features	**HDMI 2.0,** eDP 1.4 (4K support) MIPI CSI-2 4x USB 3.0, USB 2.0 Micro-B **NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores**

At this time, these are the microcontrollers being considered, more research needs to be done regarding the demands of the project, since the goal of the Envoy Commander has yet to be decided on fully. Currently, the *Raspberry Pi 4 Model B* (Table 5) is the best candidate as it combines the best elements of both worlds. We get Bluetooth and WiFi capabilities on the newest standards and a relatively powerful CPU which has 4 cores. It also gives us the option to have 8GB of RAM if necessary while still maintaining low power over USB-C or GPIO which are very attractive options. On the other hand, the *Nvidia Jetson Nano* (Table 6) comes with a dedicated GPU with NVIDIA CUDA® cores that are ideal for machine learning. However, it's lack of onboard Wifi capability and more importantly, Bluetooth support make it less attractive. This issue can be remedied with a separate USB dongle for both although some care would need to be taken for compatibility. This would also increase the price by at least $50. Lastly, the *Raspberry Pi Model 3B+* (Table 4) is the weakest option here, only added because of its lower price point (sometimes coming down to $35) and wide support. But the single-core processor, 1GB of RAM, and outdated hardware standards make this project more complicated than it should be.

There are also other options that could be explored if more power is needed. These are the most ideal due to their relatively low cost and compact design. However, due to the current proposed design of the project, the learning agent does not have to operate in a small form, since it is not moving around or has to be attached on a wall. The sensors it will be utilizing to gauge the progress of the field agents can still be attached to the arena wall connected to a large machine outside. This is also a good option only because we already have access to high-end processing in personal desktops. However, the price of those would have to be accounted for and we would like to be able to keep the study controlled and on a budget. This is also why smaller computers such as the *Intel® NUC*

Boards were not considered. Although they are very high performance microcontrollers with reputable processors and impressive features, the $600+ price point makes it non-viable, it would be better off to simply purchase a new desktop. The *Raspberry Pi 4 Model B* is not only in our budget range but is the most powerful budget microprocessor readily available. But more research can still be done on this topic.

Commander Construction

As stated previously, the Raspberry Pi 4 is our targeted microprocessor for the project. Because the commander and the decision making code contained within is the most sensitive and a point of access for potentially project breaking errors to arise relative to the rest of the project, we will be printing a circuit board to support this module, and the power supplier included should make this component stand alone compatible. Depending on the final decision for communication protocol between the dummy agent(s) and the Envoy Commander, the PCB will be modularized into three primary components. The most important module connector will be the logic level converter between the power source and the Raspberry Pi 4. These two modules are non-negotiable, as the project objective is designed to run in a standalone environment, regardless of wired connection. Expanding upon this idea, however, is that for the purposes of additional information gathering, there may be a third separable component of the printed circuit board specifically for additional hardware based sensors. These may be constructed of additional arduino modules such as KY-037 sound intensity sensors or light sensors. These have the purpose of possibly triangulating locations for the dummy agents or assisting in tracking dummy agent progress.

Ideally, however, for the purpose of ease of transportation, the Commander Agent will likely be attached to the arena from an elevated point of view. Not only will this provide perspective for the dummy agents to use for triangulation, but to ensure that the wireless connection(s) that the dummy agent(s) will utilize have minimal interference when communicating, as communication faults can greatly impact the decisions make in conjunction with the Q-table and decision making process.

It is important to note that the Commander Agent does not move, and thus reduces the need for an additional power module for mobility. Due to prioritizing functionality, the decision was made to reduce the amount of connections directly going in and out of the Raspberry Pi; however, if need be for special connection and communication purposes, there may be a separate fourth module that will interface an Arduino ESP8266 module directly to the board. Because the PCB will be assumed to be the most stable connection, and this module (if needed) would be one of the most important modules in the project, it makes sense to prioritize this connection separately with respect to the rest of the other hardware modules.

Being poised at a higher point of view and detachable, we had also come to the conclusion that the base should be heavier than the commander itself. This is just a precaution to avoid damaging the most expensive and most involved part of the project upon testing because the dummy agent(s) will be moving. Additionally, padding for the sides of the tower may be considered as well given the possibility that it still may fall due to human error and unpredictable situations.

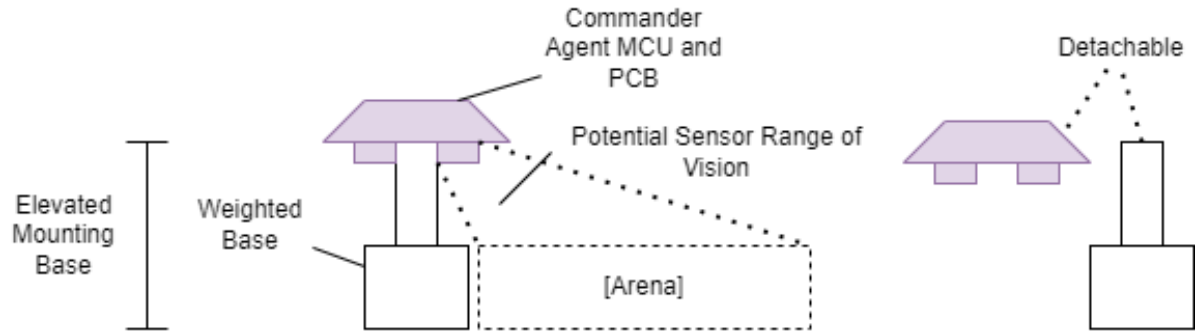


Figure 12: Visualization of Detachable Commander Agent Base

The images included in the diagram are not to scale and the height of the base neck can be adjusted to accommodate the future design of the Commander Agent, but a clear line of communication is vital for proper communication between the commander and the other agents.

Dummy Agent Construction

For our agents, our initial designs began with cost efficiency in mind as these envoys only played a portion of our project so comparably their budget impact should be minimal as well. We discussed potential requirements such as how many we were planning to run at once in the arena, narrowing down to a minimum of two with potential increase in numbers in future implementations if the worry of collisions is negligible. We then looked to see the actual mobility of our robot, performing its movements in a grid pattern or traditional turning radiuses, with benefits to each depending on our ability to design an arena to accommodate their movements as well as design the agents themselves would have to be in order to accomplish either movement strategy. We opted to go with the less complicated route with a standard turning vehicle which would have a range of sensors to communicate with the commander its current environment when decisions are required to be made by attaching a simple Arduino or Raspberry Pi board to hold standard observations and values to a vehicle of our choice. This vessel ended up amounting to the equivalent of an toy remote controller car that could be easily adjusted to include motor shields, sensors, rechargeable batteries, and any additional requirements to successfully power and drive our agents.

We are also considered having the envoys communicate with one another using the environment as a guide with certain tracking and markers in the arena to allow for specialization between each agent to follow a specific task such as one for path optimization based on interaction with the numerous buttons to activate potential faster pathways to relay this information for the commander to make future decisions on. This role system benefits our idea of allowing each learning agent to accomplish certain thresholds on their own time, but still with the central feature of navigating its environment without any previous knowledge of its location. In order to property navigate the maze, the envoys are equipped with a range of sensors that are help the robot navigate against the walls through contact, image processing to determine what portion of the arena it occupies, and potential mapping-related sensor to have the agents create a set of navigational points to follow to then communicate with its peers as well as the command

tower to initiate future decision making processes based on the current iteration of the Q-table.

These RC cars, sizing around 3in/5in/1in will provide the building blocks to have our Arduino's and Pi's attached to in order to properly have a commander-controlled multi-car system, and the goal of this is to reach a minimum of two navigating the arena at once with the potential for upscale as we accomplish the implementation of the initial goal as increasing the amount of agents will lead to an increased chance of collisions as well as allowing the simplification of the project's main goal during the first round of prototyping and testing. Although these envoy's tasks are handled mostly by our control tower that contains a majority of this project's design, the cars are still established as storing temporary data that could aid in its navigation systems, hence the need for a simple MCU that will be decided upon based on availability.

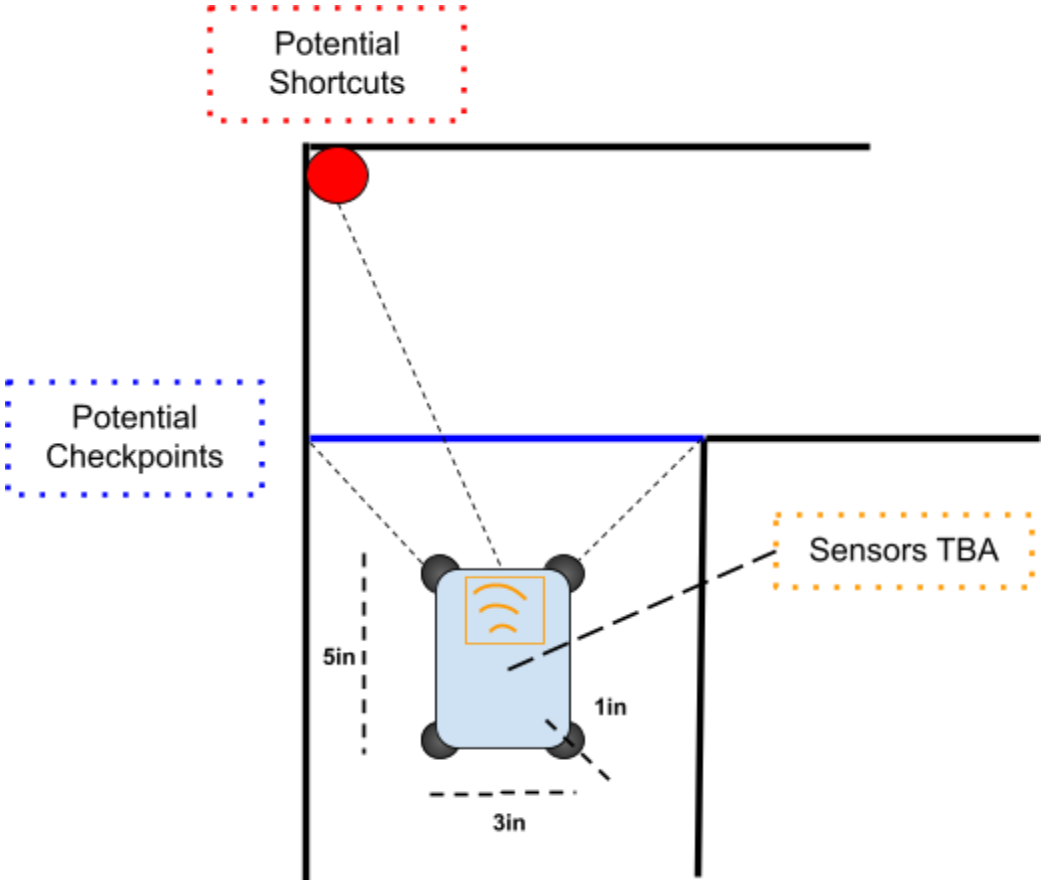


Figure 13: Visualization of Dummy Agent Schematic

Related Standards and Design Constraints

This section details the choices we have made that coincide with potential constraints and standards that we must abide by to properly design our “Envoy Commander” system and will change how we choose to build our project and the effect it has on aiding us in optimizing its electrical and software components on a cost basis but lie within the guidelines set to ensure the safety and well-being of the participants involved.

Related Standards

Using standards related to core features of our project design from IEEE, ISO, and numerous other databases to prevent any issues related to any electrical or software component of our learning agents, control tower, or possible sections of the obstacle course itself. These relevant standards that would aid in our project are listed below deal mainly with the communication field as a majority of the envoys rely on constant communication between them and the commander with some additional standards in place for the few electrical components we design to function correctly.

IEEE/ISO/IEC 29148-2018

This global standard details some basic guides for us to follow as we continue through the software portion of our project to set goals, deadlines, and other project necessities such as the general accomplishment of referencing a defined system, validation against potential world needs, or the ability of its implementation. These sets of requirements aid us in narrowing down the specifications for our project design, but we need additional pathways to transform these needs into developed requirement specifications to be bound by certain constraints, qualified by measurable conditions, and verifiable. Once these requirements are set, we can consider potential prototyping and embedded design to allow our project to be viewed by our peers and professors to not only provide feedback, but to give rise to potential new questions to be answered as we navigate the process of designing our project to lead to a conclusion after addressing our initial requirement specifications as a comparison,

ISO 17438-1:2016

This standard from ISO speaks about the navigational components that we would encounter with our learning agents as they navigate the obstacles and include the arena’s position data using specific reference material either with RFID readers, WIFI/Bluetooth to locate certain portions of the maze to then provide rewards or negative responses to whatever actions they choose to take in to successfully navigate by knowing what section of the arena it is currently in to send to our control tower to then make quality decisions as it continues through whichever part it currently resides.

IEEE 802.11-2020

IEEE established this standard in relation to communication purposes, and in our case we will utilize its understanding of specific network standards to assist in our communication between our learning agents and our command tower to properly send and receive signal data either through Bluetooth or Wireless (PHY/MAC) means to align with the Q-Learning algorithm being developed. This standard describes the functions to operate such a connection point and how they coexist with other WLAN points in whichever location we choose to test and perform our project in.

Project Constraints

The three main constraints we are facing are technological, budget-based, and time, with the latter acting as the most significant in our case as a majority of our project software-oriented requires more detailed effort into its execution as well as the testing of the minor electrical components that we would utilize. Since we are shooting for a fairly low budget with our minimized group this will end on the lower section of constraints, but this also dampens our potential technological uses as we strive to build the simplest robot to carry the bulk of our software elements. Additional factors we consider are listed in the remainder of this section and including potential constraints that could arise as we continuously develop this project.

Economic and Time Constraints

With our current budget as of the latest changes to our project ranging from $300-$500, we are spending a majority of our funds on the electrical components that make up our tower module as that system requires more custom-tailored design options such as the PCB which allows us to utilize the remaining budget into creating the environment of our project into more detail if needed, but for the current iteration we are choosing to create the simplest possible vessel for our algorithms to be run on. For the actual economic implications, the real creatives come from our software elements which have little impact as the maze implementation is mostly relegated to the lands of Senior Design, however the progress we make could bear additional results as we further into our projects completion, as if were to increase our budget we could involve more complex processes to visualize our Q-Learning algorithms. On the time end, our schedule is laid out under the project milestones tab, but after the completion of Senior Design 1’s papers we look to receive the remainder of our ordered parts before the beginning of the spring semester in order to property test and build some basic skeletons of our project parts in a realistic and efficient fashion. WIth time being one of our crucial factors for this particular project, we will strive to develop our software elements to be representative in our physical production without exceeding our own time limitations to prevent future issues.

Sustainability and Manufacturability Constraints

Sustainability constraints are fairly limited to the environment we choose to place our agents in, and in the final showcase of our project it will most likely be indoors to prevent any elemental issues for our learning agents to navigate, placing us in the position to value the constitution of our learning agents to handle basic interactions with our maze setup to then navigate successfully. To counter any problems with the vehicles themselves, we will provide pre-prepared backups to be used in case of general failures on the part of the learning agents, as this section of our project is the only one prone to such problem areas that would require additional maintenance. Speaking on the manufacturability of our project, that falls mainly on our controller tower which will be made up of mostly prefabricated parts aside from our singular PCB to more expensive such as our Raspberry Pi 4 board to allow for our increased demands in processes and memory usage as well as its additional functionalities. However, aside from this section a majority of our project will contain easily manufactured products such as our learning agents consisting of simple sensors, arduinos, and the vehicle itself being simple toy rc cars. We would also build our environment using prebuilt wall sections with simple mechanical button mechanisms to open and close certain areas of the passages to allow for additional pathways for our “envoys” to proceed through.

Environmental, Safety, and Health Constraints

With little to none environmental and health concerns regarding our envoys, possibly using more environmentally-friendly methods to power our project with additional costs made as a justifiable usage, the main safety constraints we will implement will deal with the electrical components both tested to ensure usage as well as when in continuous use they will not become a danger to not just us in the building phase, but if one were to add improvements to our design during its runtime we could do so without potential harm due to faulty wiring or lack of grounding for the control tower. In order to make certain of this, we would seek to order parts that meet a certain compliance level and only utilize them under a specific set of safety standards.

Ethical, Social and Political Constraints

There are various types of Q-learning implementations that we have found during our research, so potential social and political elements could involve the variety in how they were created and the differences between them as well as the eventual algorithm that we would be developing. As most of the physical parts are either pre-built or done relatively from scratch there is little to discuss on that end for their impact, but if we were to expand our budget to formalize the process and utilize our methods in more practical applications they could lead to manufacturing developments such as the type of vehicles being used to run the algorithm or the quality of the control tower which would differ depending on our choice of manufacturer.

Project Hardware and Software Design Details

Choice of Language

Choosing a language for coding the software and providing instructions to the agents is pivotal for both the efficiency and the overall “readability” of this project. For the dummy agent, it is pretty straightforward since the Arduino ecosystem readily supports C with a plethora of available libraries made by either Arduino or other third-party developers and a wide variety of compatible hardware components that are cheap and easy to use. This will allow us to spend less money and time on the Dummy agents whose job is very minimal anyways and will make them easily reproducible, as they should be.

On the other hand, we have the Envoy Commander, the agent that will do the primary heavy-lifting of Q-Learning and concurrency-handling such that the Dummy agents can make progress towards the goal. For this, we have decided to use Python for many reasons. First off, Python is well vetted for machine learning thanks to its collection of in-built and third-party libraries that offer complex, optimized functions with only a slight learning curve when compared to other languages. Not only do the libraries make it easy with straightforward functions and thorough documentation, Python itself, as a multi-paradigm language, makes it easy to pick for all kinds of programmers. Whether your background is object-oriented, functional, or scripted, there is a solution that can be written in Python that suits you. Now whether that is the best way is always debateable, but to just get a solution off the ground, is mostly effortless. It has many features such as mapping functions, list comprehension, dynamic typing and inference, etc. It does this while maintaining readability for all programmers, especially if the solution being built needs general-purpose functionality. Moreover, Python is also easier to integrate and adapt. Thanks to libraries such as scipy, numpy, and tensorflow, it is able to integrate with other softwares such as MATLAB and R which are good leverage in machine learning. Lastly, not only is Python free and open-source, it also pushes high productivity because less time and effort is spent on the semantics of the code and more time is spent on developing the solution.

Now this is not to say that Python doesn’t also have its share of reasons to look elsewhere. It is most known for its slow execution when compared to low-level languages such as C and C++ and that also forces programmers to use libraries which are better optimized than the in-built functions. A good example is the use of arrays. Using numpy’s mapping and array transformations is faster than simply looping through the array and applying your transformation. When dealing with the curse of dimensionality, this computation time is crucial, especially in problems such as ours, where real-time execution is as important as accuracy. Python also tends to have issues with threading since it only supports single-thread execution. Thankfully, we can handle this issue with concurrency of our own and also implement multi-processing instead.

Arena and Task

For our arena, it will feature a complex maze with a simple top-down view from our command tower operating a majority of the processes with few tasks delegated to the learning agents themselves to store and communicate. For the envoys to escape, they would have to first navigate their surroundings and send the initial data they collect to the commander to send out individual tasks to whatever agent, and in the case we choose to have more than three agents, have specialized tasks to accomplish portions of the goal individually. The environment itself will start off as a standard maze with crossing points, general walls, lines in the center to denote a midpoint, and start/end points for the agents to accomplish. Along with this, we will assign navigational markers throughout the maze to understand their relative positioning under the guise of colors which is subject to change depending on the specific sensors we choose for our learning agents. However, the main caveat to this standard escape room scenario is the opportunity for the agents to find more complicated methods to escape, and we introduce the concept of a tunneling system between the walls in the maze.

These “secret” pathways will be able to be activated in some capacity by the envoys, either through a proximity based system or physical buttons to open the new pathway. For our Q-Learning system, we will weigh the decisions to go through pathways if able to incentivize the learning algorithm to choose to act in a specific way to activate these faster routes. The main trouble comes the collisions opportunities between the agents, the wall, and themselves, so we will most likely construct the walls and respective tunneling out of some styrofoam-like material that we could easily cut and move to add to the random nature of the arena and allow for soft impacts if it would occur. And for the actual trapdoor system it will most likely be mechanically oriented to allow for the least amount of potential issues involving an activation switch that is possible to be opened by the agent and then wide enough to navigate and allow time for the robot to continue through fully, with the added benefit of some pathways being closed and some on a time limit upon opening for an additional random element.

The checkpoint system we must establish will have either sound, visual, or an RFID implementation system that would let the commander know where in the maze it exists to provide some assistance and potentially communicate with other envoys. These environmental markers will come in the form of some motion-based sound to allow the agent knowledge of area, a color or shape centered system that catalogues basic identification, or we can copy over the idea of giving RFID boosts during its secret tunnel runs and spread them throughout the course to act as checkpoints. An added feature would also be the reward system in place to acknowledge sections of the maze as “good” as it traverses it, giving higher priority to areas near the end or when selecting the option to go through the agent-activated tunnel system to aid in its quest to gain the highest reward count.

Task Detailed

The maze will be modularly constructed out of simple materials such that it can be easily rearranged. This is because we want the learning agent to be environment-agnostic. If the agent truly knows nothing of his environment and uses Q-learning principles to navigate the maze via the dummy agents, then randomizing the starting location of the agent and the overall configuration of the maze should not affect its performance. This is a better approach because it is more scalable. Instead of learning the route of the maze, we want the agent to learn about the quirks of the task, things like LEDS near the shortcuts or the importance of the checkpoints. We want it to optimize how many steps it takes and a shorter completion time. So to do this, we propose the following method, coupled with a proper reward function (defined earlier).

A traditional maze looks much like this, though this is obviously just a drawing and is more complex than how our maze would be.

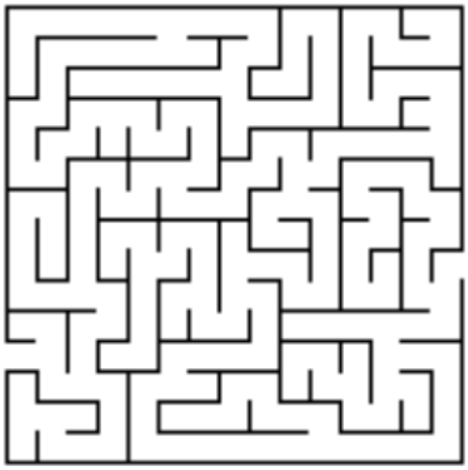


Figure 14: Traditional Maze

However, for our task, we would make modifications to this traditional maze to fit our needs.

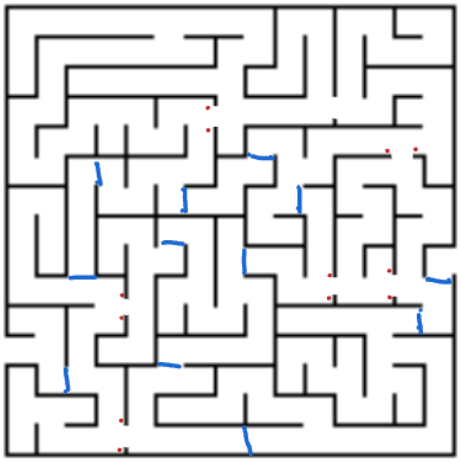


Figure 15: Modified Maze

The red dots represent possible locations for shortcut-defining LEDS whereas the blue lines represent possible locations for checkpoints. Checkpoints were placed at points where it was clear that there was progress towards the end goal. Also it is important to point out there will be only one clear goal at the end. The clear downside of changing the configuration of the maze from above will be that we would need to strategically place the checkpoints and shortcuts in each configuration as well.

The overall testing process for the task is pretty straight forward. After configuring the maze appropriately, we will first place the dummy agents in their starting locations and then ensure that they are able to communicate either directly with the commander or with some intermediary server that will relay the command to the commander. After this, the Q-learning algorithm will take over and the learning process will begin. At the end of each episode, which will be defined by either reaching the goal, or after a certain amount of timesteps, largely dependent on the size of the maze and the scale of the dummy agents. We want to make sure the episodes last long enough for the commander to be able to reach the goal at some point, given the velocity of the agent and the distance with the longest path from start to finish, again dependent on the scale of the maze and the agents. However we don't want the episodes to take hours at a time because of possible power issues. There is a possibility of checkpoints and saving the model in case the learning process does indeed end early unexpectedly. Unfortunately, finding the balance in the episode length and the learning rate will be something that is experimental. My current idea is to build a mock version of this environment virtually so that we can test these factors in a controlled environment with little cost to the actual project. The smaller the scale of the entire project, perhaps the better performance we will be able to achieve with the learning agent.

Code Organization

The code will follow a general structure as with most multi-integrated distributed agents.

Commander Agent

The Commander agent, being the most complicated and focused part of this project, will need to be split out into multiple, modular and testable classes to make debugging easier. In a project where reinforcement learning is so important, it is equally important to know whether the parts of learning are accurate using various metrics and debugging methods. But above all is utilizing the classes in Python’s multi-paradigm nature to break apart the aspects of the learning agent. Overall there will be at least three separate classes for the agent. The first and most developed will naturally be the Q-Learning class. This will be broken into the main methods often used in reinforcement learning. In order of execution, there will be a stream method that will use a thread on the raspberry pi and continuously stream incoming data from the web server located on the custom pcb. This is the interpreted data from the dummy agents that will be scaled and converted into json objects to be transmitted out. Then this data will be stored and first passed to a getObservation method that will take this data and scale it some more to turn it into a state observation to use, setting variables such as “inShortcut”, “time step”, “sensor data”, etc. This object will be saved into a state variable and then passed again to

a “generate reward function. Additionally packaged with the dummy agent’s code will be the progress tracking of the agent. This information will be abstracted away from the rest of the agent but due to the computation-heavy nature of it, we need to be able to generate a discrete reward for the agent. So after the reward is generated and saved for the incoming data, the commander agent will also take the Q table that was packaged with the current request and pass it to the learning function. This function will take in the observation, the reward, and the current step and generate a new action based on the exploit vs. explore concept while updating the provided Q table. These two components will then be published back to the web server to a certain port. The dummy agent, listening to that port will then interpret that data.

Another class that will be created in the learning agent is the pub-sub. This object will have methods that listen to the web server waiting for new requests. It will also interpret these requests to make sure it is organized and linked to the appropriate dummy agent. This should not be too difficult if a UID is set of each dummy agent added to the network. Additionally there will be a publish method that will make a POST request to send this data back to the esp8266 web server to store for when the appropriate dummy agent picks on it.

The final class will be used to scale the observation data. Since we are interpreting data from multiple sensors that all look different, we want to construct just one observation or atleast scaled data from these. The biggest point of this is to make sure that we properly understand noise for this data. This could mean passing the data through something akin to a Kalman filter or if we manually understand the noise bias from the sensor, we could adjust the noise of each sensor manually as well. Maybe this would be scaling the data or just shifting it to fit.

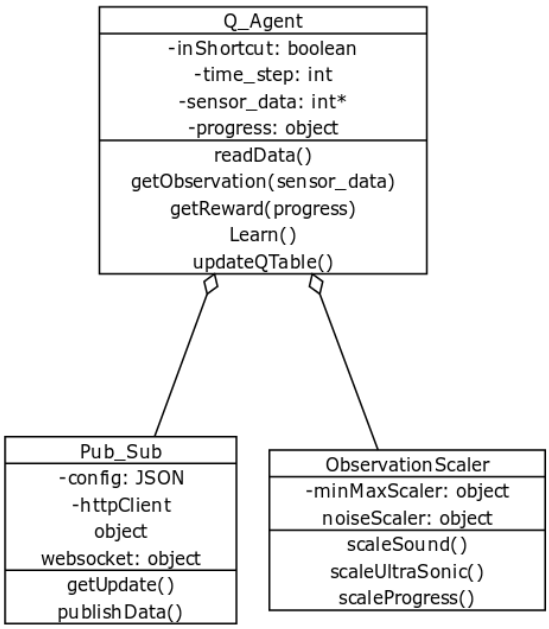


Figure 16: Class Diagram

Dummy Agent

While a relatively simple part of this project, there are still many parts of code this would need. I think the most important part of this code is reading in data from the various sensors. To accommodate this, there will be a class just for that, dubbed SensorData. This will read in the data from the sensors on command, having a function for each sensor. Additionally in this class, the appropriate Q Table will be stored, considered “sensor data” just so a new class isn’t created for this.

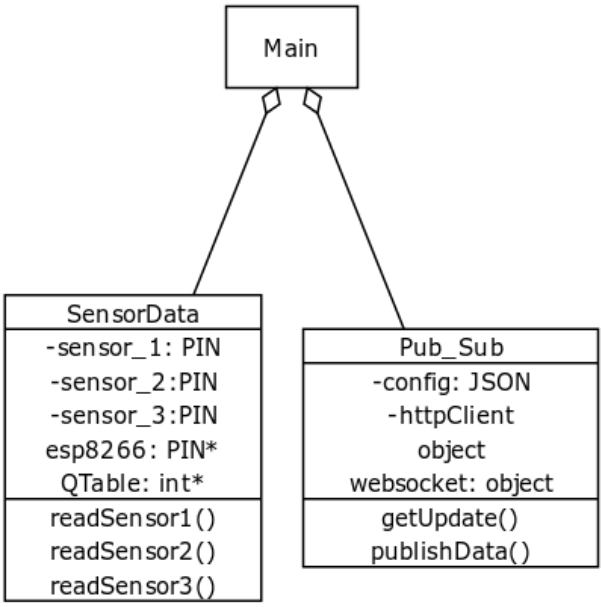


Figure 17: Class Diagram

Hardware Sensors Discussion

For the dummy agent, in order to draw conclusions and provide the commander agent information to work off of for learning and environmental construction, the commander agent must have some sort of metric or understanding of the situation before a reward or Q-table can be made and adjusted. A primary problem associated with this is simplifying the objective. For project purposes, Q-learning would prefer more discrete values for weight calculations, and being able to make real world situations into things that are measurable in discrete values is a major component of ensuring reinforcement learning is possible.

However; the dummy agent(s) may not be able to distinguish successful and good moves and actions solely from the environment. This is where our model falls short of an ideal q-learning robot simulation. The stimuli in the environment is designed to guide the robot towards a goal, but the dummy agent does not know this yet. This illustrates that real scenarios in the real world can lead towards the intended goal, without being exactly what they are in real life. In order to understand these stimuli, however, the dummy agent must

be able to return some kind of information to the commander agent for improved decision making on the commander agent's end.

The following hardwares are the primary considerations in terms of parts that will assist in helping our dummy agent relegate information along. While there may be more or less sensors in the final product, it is important to consider that each arduino board or alternative board per dummy agent will only have a limited amount of pins dedicated to ensuring hardware sensor function, as many will be consumed by the motor shield and motors along with the wireless module that we require.

KY-037 Sound Detector Module

This module requires two power supply pins and two functional pins and can output in digital or analog signals. These can both be interpreted and one or both pins can be used and be useful for the purpose of the project.

This module in particular was chosen because it was lightweight, small, versatile and noted to be reliable in its function. Because there will potentially be buzzers in the arena to indicate proximity to a certain goal, this module is near essential as it provides a great amount of information that is a notch above the basics. Additionally, this module may also be used on the commander agent to gauge or triangulate a position within the arena given the position of the commander agent is fixed. Should this be used, a method for locating the dummy agent in space exists, and this greatly simplifies the problem presented to our group.

HC-SR04 Ultrasonic Sensor

This module requires two power supply pins and two functional pins, and can output to arduino programs as a distance. This module is an essential module and may be interchanged with modules such as the TFMini LiDAR module, but the generic ultrasonic sensor suffices for this project.

This model was chosen because the effective range is advertised to be 400cm, which roughly estimates 13 feet, and far exceeds the size of our constructed arena. By using this module, our dummy agent can prevent collisions, and perhaps draw conclusions about where it stands. The only shortcoming of using this device is the lack of color sensitivity, of which we could interchange, but the main functionality would be orientation and decision making for the actions that are queued. As an essential module only requiring two functional pins, this component seems like a cost efficient and stable option that will most likely be used.

YL-99 Collision Switch

This module was chosen because of the possibility of running into objects. While typically a no-brainer to include collision handling, because of the limited space requiring three whole pins for a tiny module that may be redundant is unlikely. While being considered, this component will only be used if there is extra space and need for collision handling. Additionally, however, in the event that this switch is needed, the environment could implement kinds of mechanical switches or levers that could be activated by the dummy agent. Until these components are decided as final, this module will not be.

LDR Photosensitive Module (LM393)

This module was decided upon because of the existence of lights within the arena that is being made. Because of the possibility of light activated and reliant puzzles or motivators, having a module that can detect light is paramount.This module requires two power pins to function and two functional pins, which matches the size of the majority of the other modules being used for the dummy agent.

This module detects light and outputs an intensity level constantly. However, one of the drawbacks of the photoresistor is the construction and relative weakness jutting out from the module itself. Being a relatively expensive module with respect to the other essential modules, this module may or may not be implemented, but preferably has a spot as one of the essential sensors for our agent as it provides opportunities for learning to be done. Depending on the time allowance, this may be one of the later additions to the code and dummy agent.

Miscellaneous other hardware sensors include color detecting photoresistors (TCS3200) or infrared based rangefinders, but the essential modules include a sound detector, rangefinder, Wi-Fi module (Discussed in section: Communication), and motor shield and power modules for the dummy agent. After these are implemented other sensors may follow, but the sensors will reflect the goals that the arena will set-up and support the development of the q-matrix first and foremost.

Agent Progress Tracking Detailed

From a “big-picture” view of the problem, we have a commander agent that coordinates 1 or more dummy agents. The dummy agents have no real understanding of their overall progress towards the goal, their relation to each other, or any “willpower” to make decisions themselves. All they can do is observe their current state, communicate this to the commander so that they may receive back updated q tables and then take the action communicated back (based on the action-space). The actual learning agent, the commander, takes these state observations, looks them up in the Q-table, returns back the action with new rewards in the updated Q-table. To coordinate the dummy agents, the commander may send back only one action and have the other hold, or send back an action to N dummy agents, whatever is considered optimal.

Now a few issues arise with this. Namely, the problem of the reward function. How do we decide what rewards to give the commander and dummy agent, if they don't actually know the progress towards their goal or the purpose of the environmental stimuli we have placed along the way? One way to go about this is to abstract away some information and use that to train the agents. One example to demonstrate this is to have a camera attached to the commander that can visualize (in an Eagle Eye format) the progress of the dummy agents after each action taken and assign an appropriate reward. In this system, the information received by the camera will only be available to the programmer, the researchers, to leverage towards the reward function. Another idea would be to use low-level RFID gates as checkpoints along the maze. These gates could be numbered and that number will correlate to the reward. The further along in the maze the agent is, the better the reward. With this solution, there are power issues to work out regarding how all the gates will be connected to each other and how those signals will transmit back to the commander tower to be leveraged for the reward function. The first

solution solves this issue but also has a higher margin of error for detection since camera detection has a lot of color thresholding issues. Having the dummy agents with RFID tags just pass through gates is a lot easier to use as tracking progress.

Going back to the original discussion, another problem that arises out of coordinating the dummy agents towards a singular goal is deciding which action to assign to each agent. One current solution we have come up with is having them work on different tasks, for instance one agent finds the fastest way to get to the goal of the maze, but the other wants to focus on better long term rewards. This approach can be achieved through hyperparameter tuning prior to the training, based on how many agents there are. Another approach would be having the dummy agents simply start at different positions in the maze and as they move towards the goal, the information they provide to the commander could be used to collaborate. For instance, line of sight for soldiers in warfare. If one soldier can see enemies behind a hill because he is on a high vantage point atop a tower whereas his peer is at the bottom of the hill and cannot see the enemies on the other side. Soldier 1, through some communication topology, can pass that information such that soldier 2 is either informed or simply instructed to stay put. Notice in this example, Soldier 1 did not directly transmit that information to soldier 2, instead the higher-ups in the communicative ranks were informed so that they made the informed decision, with their infinite wisdom. But the issue with this approach, though easily implementable, is that not only is time lost in communication, but there is also a need to only transmit necessary information to streamline the learning process and minimize power use, since most of the power will already be used in movement and sensors.

Communication

The communication design is what will make or break the multi-agent process desired from the potential fleet of dummies. Because of this, this section may change due to how feasible each future alternative appears to be in the coming months. The alternatives being explored at this time are a combination of different methods used to communicate between generic Arduino boards, and the Raspberry Pi 4 we have acquired for this project. Listed below are some potential alternatives that exist to accomplish our Q-learning objectives. Not all of these alternatives will be explained in full within this document.

- Wi-Fi Interfacing using the nRF24L01 module for arduino boards and the Raspberry Pi VNC Module
    - Benefits
        - Wifi connectivity and potential for intermediate management
        - Built in module for Raspberry Pi requires no additional pins.
        - The Arduino module has the most support and researchable information available.
        - Potential Co-Reception and networking
    - Constraints
        - Requires an intermediary device for setup and connection
        - Requires the most amount of pins to manage for the arduino board.

        - Requires Raspberry Pi pre-boot modification (potential inconsistency)
- Bluetooth Interfacing between the built in Raspberry Pi 4 Module and generic HC-05 Arduino module
    - Benefits
        - Module is built into the Raspberry Pi and doesn't require additional pins to work.
        - Previous knowledge of bluetooth pairing and widely available support
    - Constraints
        - Bluetooth based delay and potentially smallest range implementation
        - Limited data transmissions, requires more software side management within the commander
        - Difficulty multi-interfacing arduino devices
- ESP8266 Arduino Wi-Fi interfacing to intermediary service
    - Benefits
        - Intermediate connection management available
        - Requires less pins from the dummy agent than the nRF24L01 module
        - Potentially widest range given implementation of the module
    - Constraints
        - Intermediate management required
        - Module has less readily available sources for implementation relative to the older module
        - Module would require more software management and troubleshooting

Using the models of communication mentioned above, the primary option being explored in this section and the first to be tested will begin from the top moving downwards. This order was selected due to the ease of access of materials, and the potential drawbacks of the later options in the list. Given the space requirement of the arduino wireless module, it may be preferred that our Arduino board be switched to a board that boasts the ESP8266 wireless connection module without needing external connection. However, the current selected option is preferable to be able to support both 'Open Connection' scheme of communication and 'Single Response' that is preferable between the Dummy and Commander agents.

Simplified, the Raspberry Pi for this communication schematic requires a specific operating system that allows the Pi to connect to Wi-Fi. In order to do this a pre-configuration is required that connects the operating system to the network and enables remote access into its shell. This can alternatively be accomplished by using ethernet connected to an available port, but this will be avoided.

On the arduino end of the communication, after the module is connected to the necessary pins and the frequency is set to the correct connection, information is able to be sent

The communication mode being utilized between the dummy agent and the commander agent is single response (as referenced in section: Communication Implementation). The technology outlined above uses the nRF24L01. The reason this

implementation was decided on to attempt and test first is that it was decided to be the most versatile of the options considered. By being more versatile the avenues for various hardware sensors and possible arena setups are preserved, while simultaneously preventing the need for extreme code based error handling. The largest constraint by choosing this module is that the module is relatively old compared to the other approaches considered. While this does support the evidence that the module is reliable and has documentation and first hand use-cases online, this also demonstrates itself in the way that it takes significantly more resources than opposing Wi-Fi transceivers. Additionally, the signal that is received and sent is raw frequency radio based “Wi-Fi,” and while this doesn’t necessarily make it worse than monitored conventional Wi-Fi that desktops and smart devices have access to, this raw output can have implications that make the sent signals vulnerable to interference when exposed to many other signals surrounding its source. This problem doesn’t present itself as much when only two or three signals are sent as in the case of multiple dummy agents, but if many different technologies in the same room are active at the same time as multiple smart devices on a building’s 2.4 GHz band rate, it may interfere with complete accuracy. Nonetheless, this issue will be monitored closely during development, and addressed promptly.

Abstract Software Design and Interaction

The interaction between the Envoy Commander and the dummy agents contained within the arena are mostly separate, and the purpose of keeping them as such is for ease of adding more dummy and learning agents while also maintaining lightweight and understandable software design. Because each primary component of the project has their own relatively intelligent processing device, the software will be split into the commander agent and the respective dummy agents individually interacting with the commander. While each learning agent interacts with the commander agent with modularity in mind, the actions that are taken by the commanded agents are determined by the information that they end up passing to the commander. Each dummy agent will store their own q-table, of which the commander agent will use to give them orders on what actions to take, which will be encoded within each dummy agent. By hard-coding the actions taken and return values or thresholds that are to be used by the dummy agents (See section titled Hardware Sensor Calibration and Tuning), modular design in abstraction and practicality can be achieved in a way that maintains simplicity and completeness at the same time.

In summary, upon the onset of a simulation, or episode of the project, the agent(s) that are placed within the arena will be in a constant loop of actions being taken and communications being sent to the commander agent that utilizes the information that they accrue to either gain more information about their surroundings, or try their hand towards reaching the goal that they were intended and designed to reach. By finding a balance of either using their knowledge or trying to gain more, they emulate the classic reinforcement learning characteristics of trial and error, and emphasize the principle of delayed reward. These steps towards more human and imperfect styles of learning give the learning agents an edge over perfect approaches that may never explore the less optimized ways of thinking and their advantages. Expansions upon the principles and implementations in practical and idealistic ways are explored in the following sections.

Project Block Diagrams

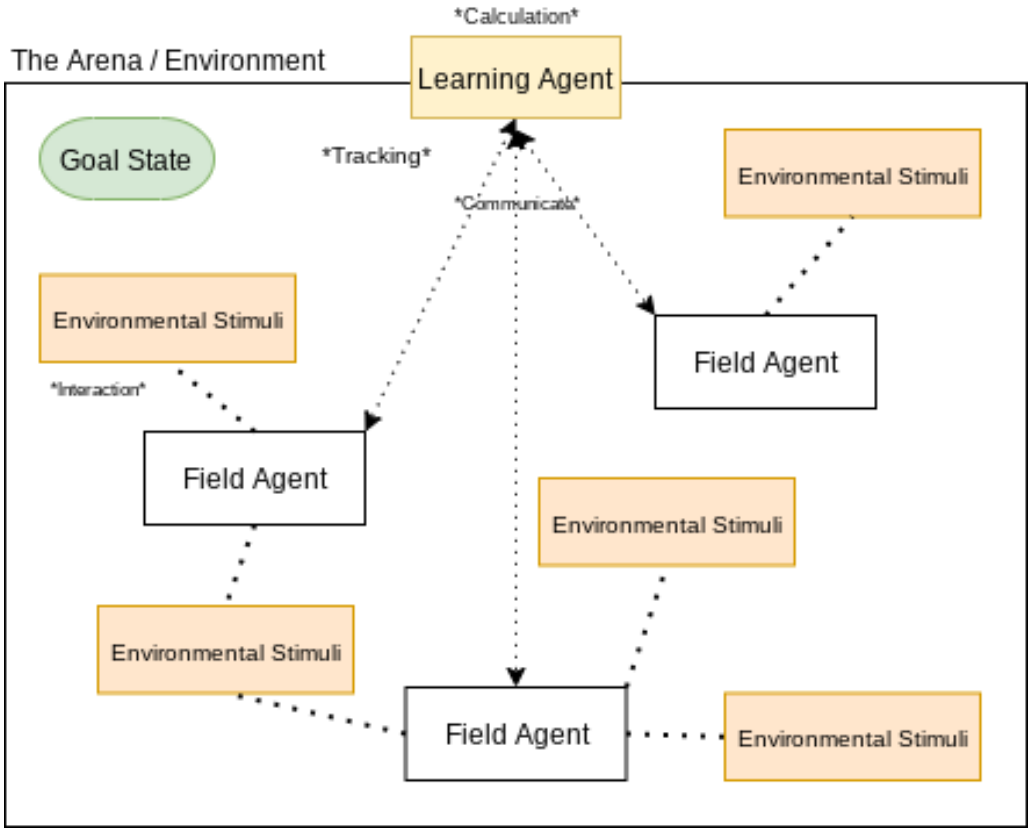


Figure 18: Conceptual Block Diagram (With Multiple Field Agents)

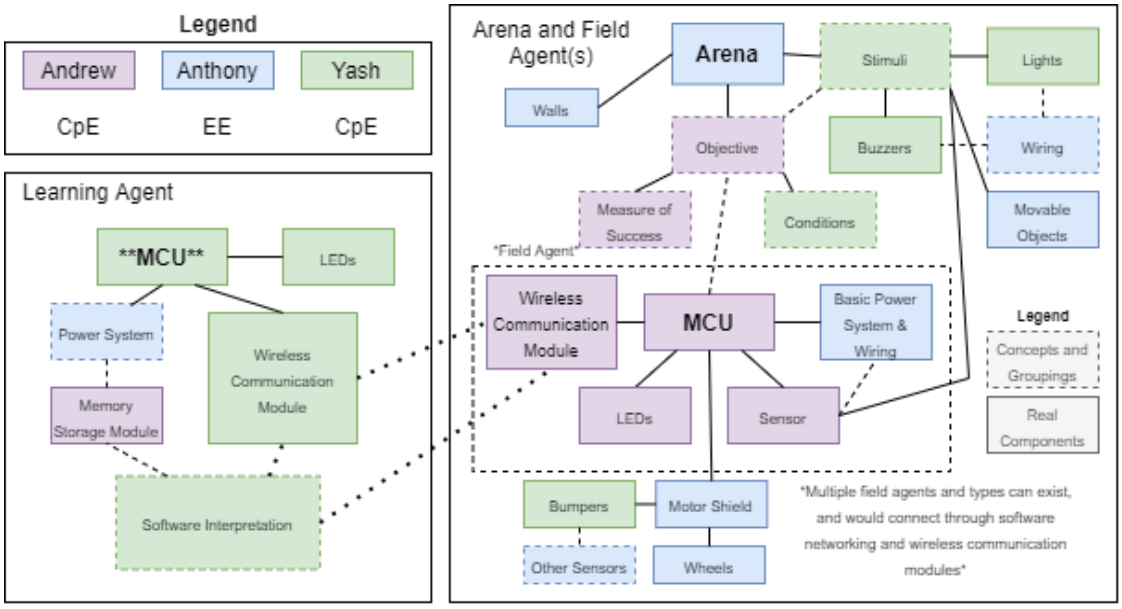


Figure 19: Design and Responsibility Block Diagram

Project Prototype Construction and Coding

The typical flowchart for Q-Learning code looks like the following:

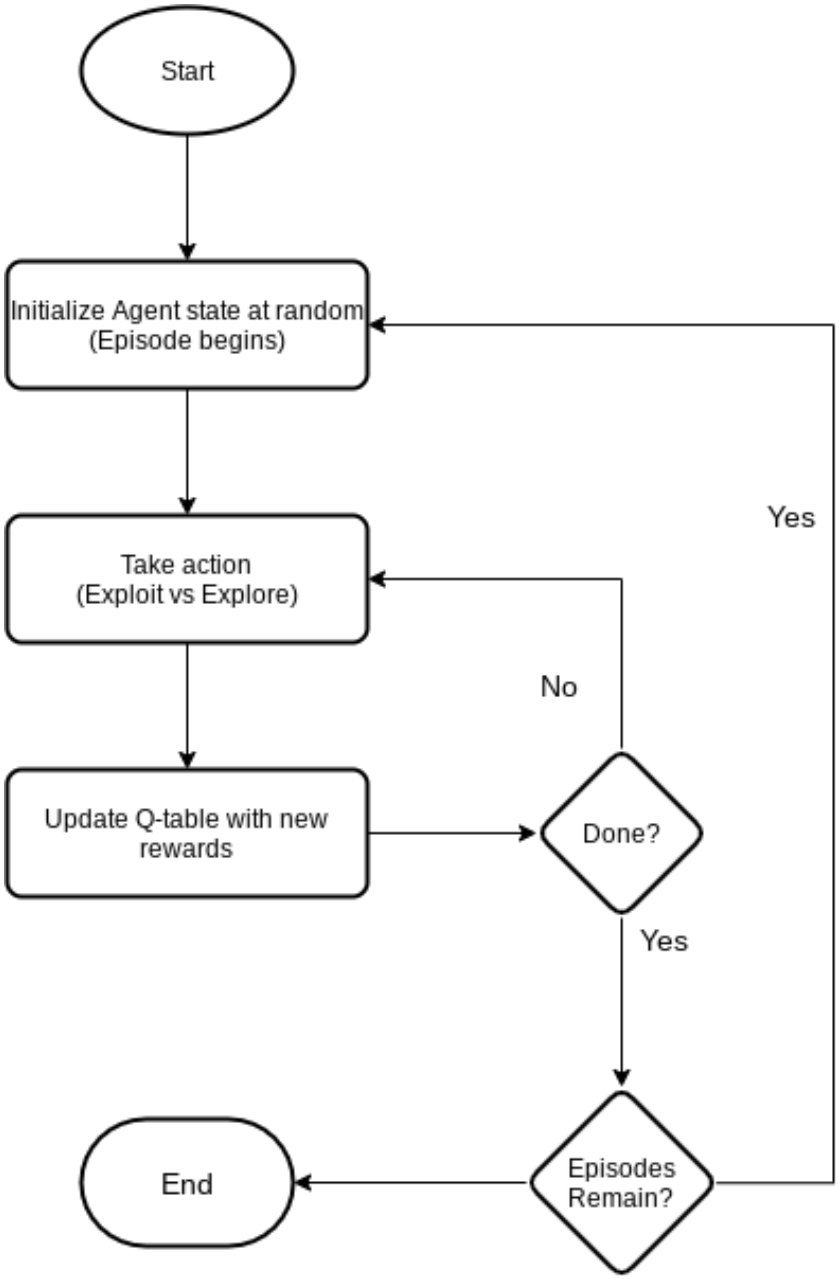


Figure 20: Q-Learning Flowchart

For this project, we want to alter this flow to add in the ability for the commander to coordinate multiple dummy agents. To do this, a few more decision steps need to be added in, to look something like:

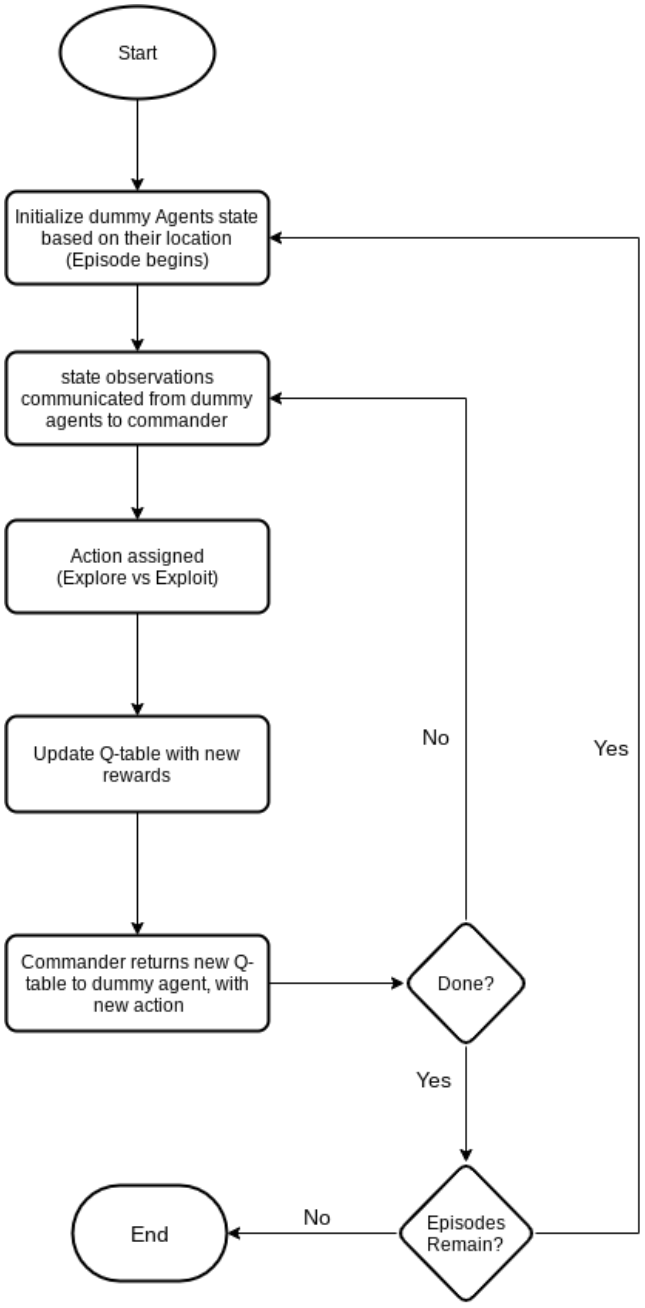


Figure 21: Q-Learning Flowchart with Updated Decisions

In this updated form of the Q learning, a few simple blocks are added, mainly to communicate information between the commander and the dummy agents. This may seem like a trivial step to add in but it allows for collaboration to happen. As the communication is received by the commander, it will have to process them concurrently, though the exact method is still up for research. This concurrent processing or parallel processing (depending on how it may go) is what will drive the information sharing and collaboration efforts. However, this aspect is still in development and research.

Prototype and Finalized Communication Implementation

Based on the section titled Communication Implementation the scheme of communication that was decided on was a locally hosted Raspberry Pi server. There are two methods that are in consideration at the moment, but both use a similar principle that can be applied to either method. Whichever one is chosen, a TCP type “single response” server will be set up for hosting on a Raspberry Pi device. The two approaches differ by the hosting server, where one employs the command agent as the host IP and Port for local communications on the network. Of course, as explored previously, they will need to find a common network to host and set-up the communication, but once this connection is established, using Raspberry Pi console access the commander agent will be able to receive the connections and interpret them, consecutively sending it back as well. This connection being referenced is with respect to all of the agents involved in learning and the current simulation. The alternative implementation is much the same, where the difference lies in who hosts the server. While it would make sense that any Raspberry Pi device would be able to host the server, learning agents could be outfitted with their own Raspberry Pi 0s, or miniature command agent style boards that could handle their own interactions. However, to avoid the added costs and complications associated with networking multiple hosts, having a potential auxiliary learning agent that does not actually do what was previously defined as learning, but simply assists in hosting the network is desirable. As such, the alternative approach is to host the network using an additional Raspberry Pi 0 as a support learning agent to aid in communications. In this way, communications can travel through an intermediary to speed up communications between the Envoy Commander and the learning agents.

Regardless of who is hosting the server, the principle is the same. After the console of the host agent has been setup, all outbound and inbound connections will communicate in a way that utilizes the built in WiFi module that comes with Raspberry Pis, or the wireless module specific to the learning agent’s arduino whether it be the older nRF24L01 module or the newer ESP8266 module interfacing. These wireless modules will be able to co-access an HTML server with an accompanying php script or set of scripts that have the defined software design included within to access the different actions and logic that need to be exchanged between the commander and learning agents.

By using this type of connection and implementing TCP single response communication, chained TCP communications can be used to communicate more complicated chains of information, and the possibilities for use are relatively infinite, as therefore, properly coded scripts can act as and access eachother per device. The next section details some potential information that may transmit using this technology.

Printed Circuit Board

PCB Design

With very little experience in making the PCB, we attempted to use a minimum amount of components to lessen the impact of failure points, and this was accomplished from our previous knowledge of Eagle/EasyEDA of designing components. After going through the process of choosing our parts, which are mostly listed in the sensor discussion unit, aside from a few quality of life additions to make our PCB fit with the project we are implementing. As we are in the beginning stages of design, this layout is prone to change as more research is done into our PCB as the current iteration is only based on material learned from the previous Junior Design course and neglects some basic housekeeping methods to make this board usable. This is seen in our layering methods which are unfinished to this point as they will be added when a more concrete design is found, along with sorting the different voltages and keeping the smaller components closer in proximity to their respective major part to prevent potential problems. Additionally, with a majority of our budget being spent on this RPi4, our MCU is likely to change to lessen costs as the design process continues to fit whatever role this PCB plays in the future, so most likely the layers will be attempted to be kept at a two to three range to not only help our budget, but to also assist in reducing the complexity of the board. The .brd file below is a demonstration of this strategy and we look to improve this layout in the future.

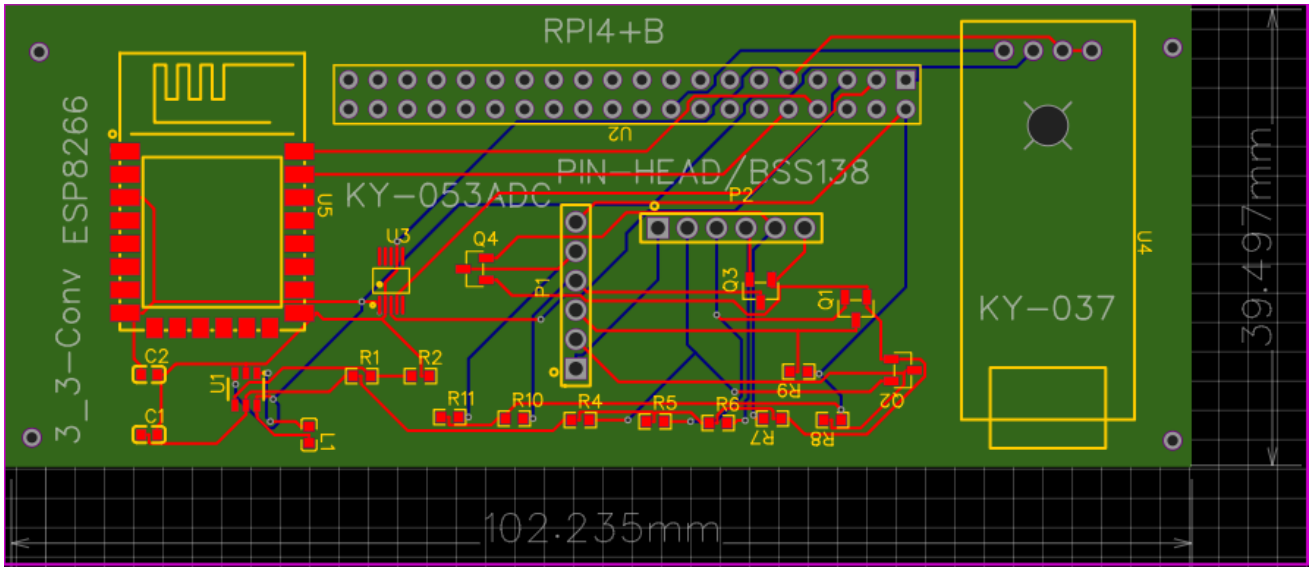


Figure 22: EasyEDA PCB Layout of Commander

Currently, the PCB we have is on the larger end, mostly due to our choice in MCU which could potentially be revised at a later date, with a length of around four inches and a width of one and a half which in the current design is implemented in our control tower. However, this could be changed after some consideration into the specific parts that would be required for its usage, as the commander tower role it currently holds could change upon needing the PCB in a different place in this project to account for a less prestigious role if we choose to connect the components to the control tower design manually.

PCB Schematics

Microcontroller

The microcontroller schematic below consists of a Raspberry Pi Model 4+B, with its corresponding parts of a KY-037/KY0-53 sound sensor module, logic level converters in the BSS138 and Pin-Headers combination, and the ESP8266 which is a communication module to transmit data, along with several basic components such as switches, resistors, LEDS, etc to assist the MCU in its functionality..

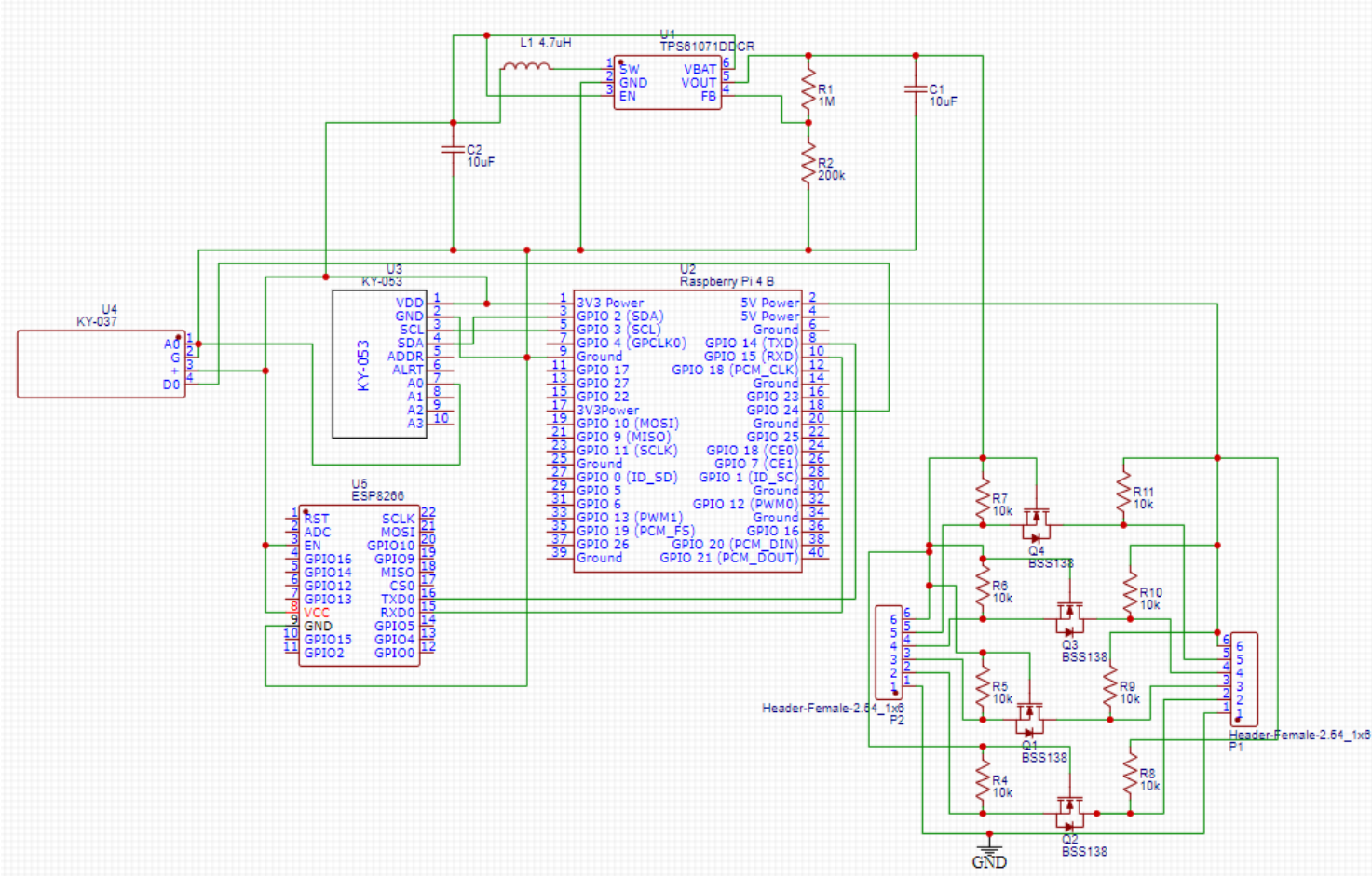


Figure 23: EasyEDA PCB Schematic of MCU

Bi-Directional Logic Level Converter

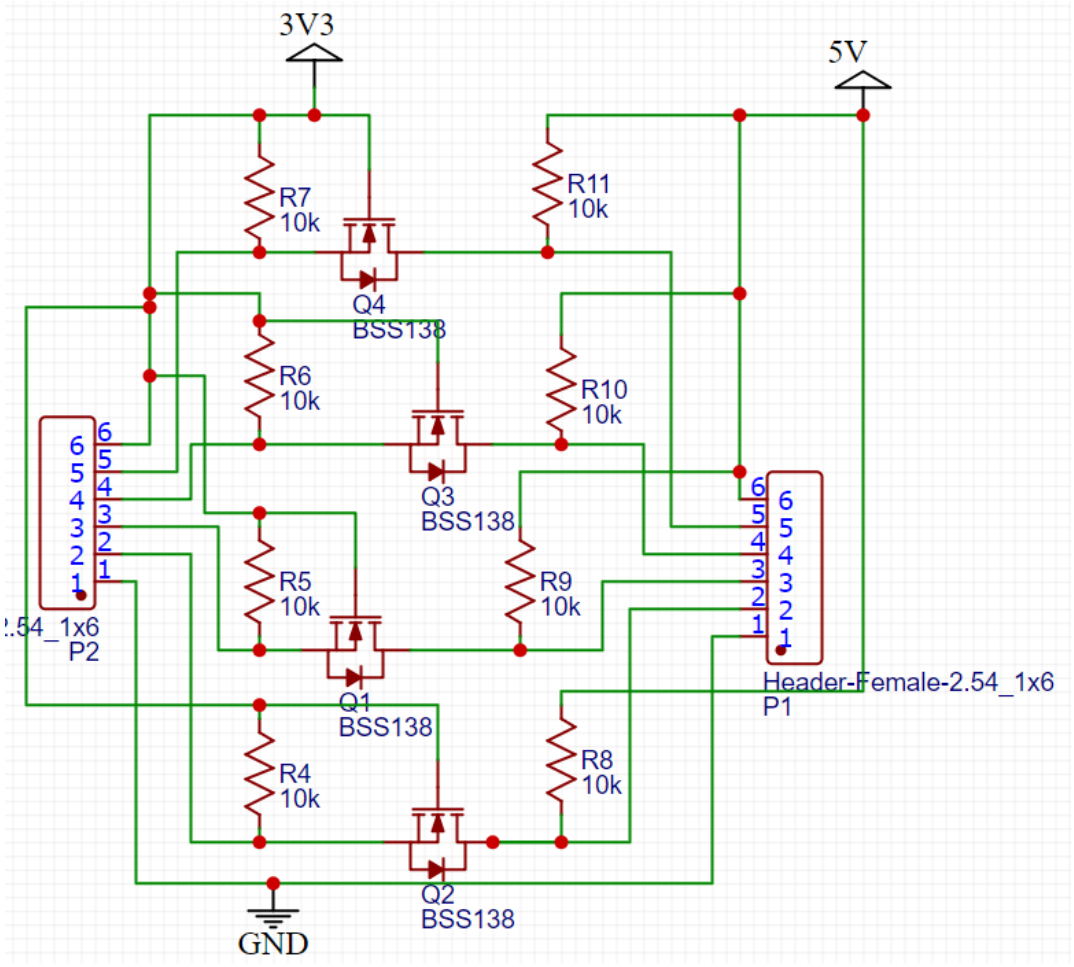


Figure 24: EasyEDA PCB Schematic of BSS138 Design

Voltage Boost Converter

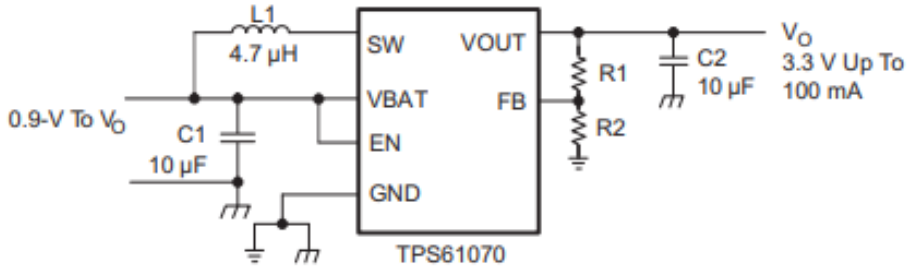


Figure 25: EasyEDA PCB Schematic of TPS6107 Voltage Boost Converter

PCB Bill of Materials

The list of materials in the below table include the current components we would need to acquire from a PCB vendor to request the construction of to later implement into some aspect of our final design. The vendors that we would seek to design this PCB from are undetermined as of yet, and this section will be detailing our vendor decision upon navigating through the potential options.

Table 10: PCB Bill of Materials

Name	Footprint	Quantity	Manufacturer Part	Manufacturer	Price
KY-037	KY-037	1		HUABAN	8
10uF Capacitor	C0603	2	0603X106K100	WTC	0.022
4.7uH Inductor	L0603	1	KPS0603LD4R7KST	Yanchuang	0.05
Header-Female-2.54_1x6	HDR-TH_6P-P2.54-V-F	2	Female header 1*6p	BOOMELE	0.059
BSS138	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR	4	BSS138	CJ	0.027
1M Resistor	R0603	1	AECR0603F1M00K9	ResistorToday	0.003
200k Resistor	R0603	1	AR03BTBX2003	Viking Tech	0.129
10k Resistor	R0603	8	RE0603FRE0710KL	YAGEO	0.005
TPS61071DDCR	SOT-23-6_L2.9-W1.6-P0.95-LS2.8-BR	1	TPS61071DDCR	Texas Instruments	2.197
Raspberry Pi 4 B	HDR-M-2.54_2X20	1		Raspberry Pi	180
KY-053	MSOP-10_L3.0-W3.0-P0.50-LS5.0-BL	1	ADS1115IDGSR	TI	5.1
ESP8266	WIFIM-SMD_ESP-12F-ESP8266MOD	1	ESP-12F(ESP8266 MOD)	Ai-Thinker	7
				Total Cost	202.592

Hardware Sensor Calibration and Tuning

For all the sensors included for each component of the project, a certain amount of calibration will be needed and this varies for each sensor included. All sensors will first be considered by how they function and the values that they return to the code. The code will then be described with respect for how the value will be used for arena-traversal or miscellaneous decision making processes. Each sensor's return value will then be thresholded for each abstract movement function that it pertains to, and possibly how it might interact with other sensors per project component.

Thresholding, in this case will be abstract, and described procedurally as to not having constructed the prototypes in full yet. Values returned by the sensors are also subject to change as similar sensors with different pinouts or return values are switched to. These values will be recorded and finalized in a final section after the first prototype has been completed. Once they have been measured and adjusted to the size of the arena, dummy agent(s), and commander agent, the final measurements will be included in a later section.

The first finalized hardware sensor being utilized will be for navigation, and is the HC-SR04 Ultrasonic Sensor (Distance Sensor) or TFMini LiDAR Sensor (Laser Based Rangefinder)

This section also serves as prototype sensor configuration, as all the sensors present in this section will be present in the designs leading up to the final design. These procedures for testing and sensor calibration are also subject to compatibility issues with the intended code design and arena stimuli.

Dummy Agent Sensors

HC-SR04 or TFMini LiDAR Sensor

For both the ultrasonic sensor and LiDAR sensors being discussed, the procedure will be the same. The differences between the two are primarily range, cost, and function in how they measure distance. The HR-SR04 Ultrasonic Sensor uses a trigger-echo system similar to a bat that uses reflected sound from the surroundings to find the distance in front of the module. The advertised range of this module ranges from 2cm to 400cm. Logically, this module will be used given the size of the largest length of the arena is smaller than 400cm. Because the dummy agent is most likely to be constructed first out of the components of the project, the arena may also be designed around this constraint. The TFMini LiDAR sensor has a larger range of between 30cm and 1200cm. This range, being slightly overkill, may only be utilized if the size of the agent and thus the arena requires a more consistent module between these values, as the module would require a positioning that is almost a half foot behind the front of the robot to accommodate for the (advertised) minimum range of the module. Again, the arena may be re-designed around this constraint, but unless the ultrasonic sensor proves inconsistent, this assumption and accommodation is unlikely.

HC-SR04 or TFMini LiDAR sensor Thresholding and Calibration Needs

The use of this sensor is primarily navigation. The movement of the dummy agent as it is designed relies on wheels and basic RC car motions. Because of the different implementations of movement for different RC car chassi. The different types of RC car movement is discussed in a previous section, and will be expanded upon here for specific reference to how the rangefinder being used will be calibrated.

Regardless of the chassi-style that the dummy agent utilizes, the same two motions stay true and can be modularized into distinct functions. The two functions are pure forward and backward motion, and the need to turn. While going forward and backwards stands the same between all the chassi styles considered, turning will require completely different kinds of calibration and navigation algorithms. These algorithms are discussed briefly here, and will be expanded upon as tests are run in a future section.

Table 11: Types of RC-Chassi Implementation

Ideal # Of Sensors	RC Chassi-Style	Difficulty of Implementation
1	Shopping Cart	Simplest
3	Conventional 2 Wheel Drive	Hardest
1	Quad Rotational 4 Wheel Drive	Difficult

Shopping Cart

This chassi style is characterized by primary control belonging solely to two wheels parallel to each other, which are static and can rotate in opposite directions. Additional wheels and supports are optional, and exist only to balance the agent, as this type of chassi relies on rotation around the axes of the primary wheels.

For this style of movement, implementation is simple because the forward and backward motion is easily modularizable from rotation and turning. In this case, rotation and turning can be mutually exclusive, as the agent in this configuration does not necessarily require forwards and backwards motion to turn. The abstract implementation of this movement is simple and the most flexible because of these reasons, and requires the least amount of calibration and measurement.

Shopping Cart Thresholding and Calibration

The simplest and most likely implementation for this type of chassi is the two function motion and turn separation. With its use explored further in the Prototype Code sections, the forward motion requires the sensors being discussed to ensure that the robot stops far enough from the wall to enable a sound rotation of any degree, and not damage the agent or the arena. Because the wheels can be centered based on the size of the

agent, the only necessary interrupt for the rangefinder to implement is when the bot is about to run into a wall or object. Because the arena will not be running into any complex objects that require complex interactions, a simple measurement from the module to the furthest edge of the bot is the most important measurement to calibrate and threshold. Simply put, based on the turn radius and expected deviation, if the length is shorter than the furthest radius of space the agent utilizes plus the expected deviation, raise an interrupt to turn the bot. Simply represented, below, this is the only measurement required.

For testing, which will also be elaborated on further, is included in a future section. (See Final Design)

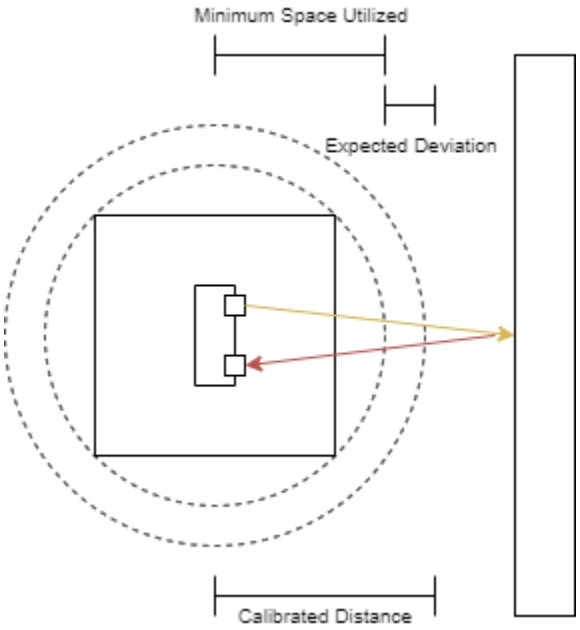


Figure 26: Example of Calibrated Turn Distance

The inner dotted line is the expected space that will be taken if rotating in place around the agent's center, while the expected deviation and outer radius may be caused by uncertainties and randomness in mechanical operation.

Limitations for this design is the possibility that the agent runs into an object that the sensor does not detect due to being centered. A possible fix for this design is a second sensor that works in tandem, placed on the furthest sides of the bot. This would detect the edges potentially colliding with the walls of the arena. This restriction, however, is not algorithm breaking, and will only be implemented if additional active pins are available. It is unlikely that another design would be considered but in the event that speed is desired or a more advanced turning algorithm is required for coordination with other agents, the third alternative of multiple sensors and varying primary wheel speeds can be adjusted to ensure a specific turn radius.

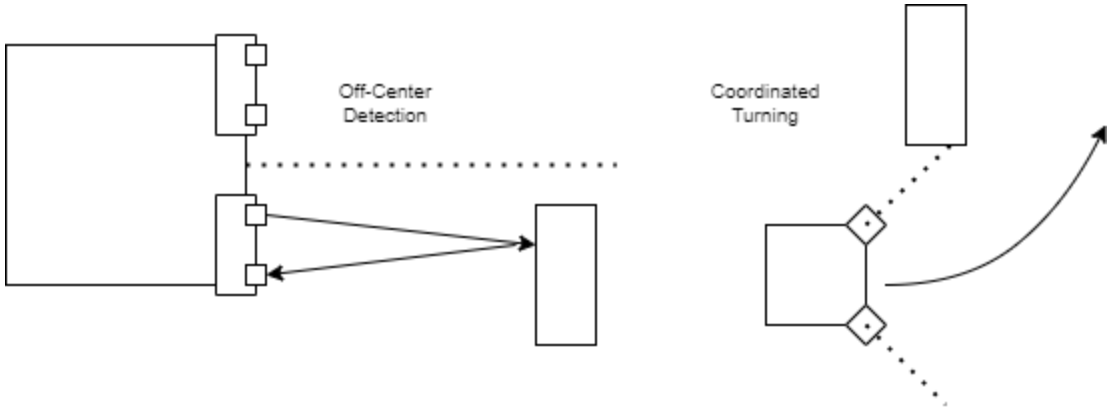


Figure 27: Alternative Sensor Design and Placement

The left design accounts for a relatively wide agent, while the model on the right exemplifies the need for additional non-centric sensors and rangefinders.

Conventional 2 Wheel Drive

This chassi style is characterized by primary control belonging to either the front or back wheels, and rotation based on the motion of the vehicle coordinated with the angle of rotation of the rotating axle (typically the front). Because of the static wheels being either the front or back of the chassi, rotation is difficult because it requires forward and backward motion to make precise movements which is required to navigate a maze or escape chamber as intended. Because of this, the most complicated algorithm of timed motions is required, including the potential need for backwards and an angled directional sensor. Because of these requirements, this chassi is the least desirable and will most likely be avoided. This also implies it will be explored the least as it is only included as a back-up plan in the event of no other chassi being available.

Two movement implementations and algorithms will be explored briefly, but only abstractly, as this model is unlikely to be used. Because of the requirement for multiple sensors, the decision to use this model will likely also require a redesigned sensor load out and arena.

The movement for this kind of chassi is similar to driving a car. As driving real cars require (in most cases) a human driver, it is immediately clear why re-making this process is likely impossible with a single sensor and may require multiple in its place to avoid collisions and aid in agent navigation. Because of the great amount of space per rotation of this chassi, it makes sense that the collective calibrated distance that takes into account expected deviation is massive. This requires learning to be completed at a far distance with respect to object detection, and more complicated processes would be needed to complete tasks and perform any sort of information collection and learning. Simplified, an algorithm with this turning and motion chassi would involve long and involved calibration with operations that mimic maneuvers such as three point turns and reverse detection as well. The added limitations for the other sensors due to needing multiple sensors and complicated code implementation are extreme, and due to the improbability of implementation, will not be explored in depth unless necessary.

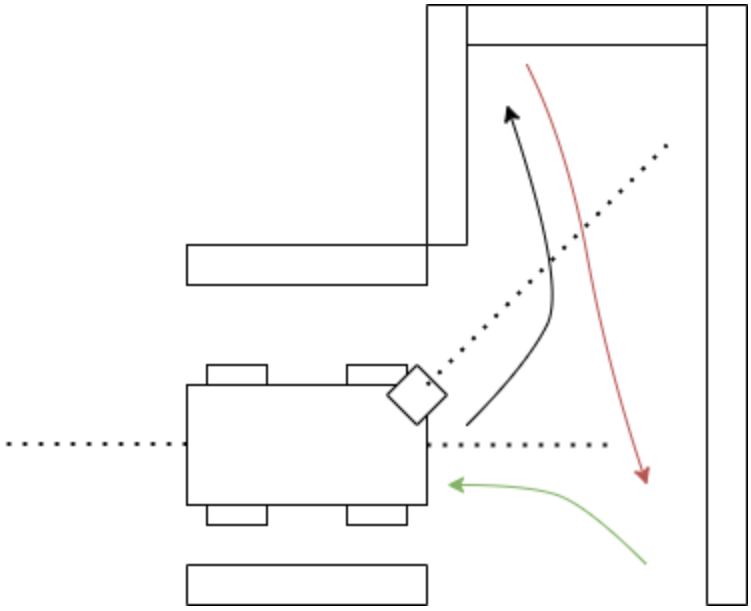


Figure 28: 3 Sensor Model for Two Wheel Drive Chassi

This model demonstrates the use of three rangefinders with respect to a complicated maneuver, which would need much more calibration and adjustment to complete consistently at different angles with respect to the former model.

The second motion algorithm would be based off of a unidirectional kind of motion, and an arena designed for either clockwise or counterclockwise traversal. By utilizing this model, reversing and a backwards facing sensor may not be necessary. Unfortunately for this kind of design, however, the great modifications required for the design of the arena and the additional space and calibration this design would need for modularizing actions and additional sensor information is too great to rely on primarily, and will be an alternative approach if absolutely necessary.

Quad Rotational 4 Wheel Drive

This chassi style is characterized by dual variable axles that increase the angle of rotation for the chassi by keeping the axles opposite turning and non-static. While the most difficult to manage electronically and mechanically, finding a chassi of this style would middle the comparatively harder static based axle 2 wheel drive, and the simple shopping cart style discussed formerly. Because of the speed advantages and possible arena design work arounds, this option will still be considered.

This movement style is closer to the shopping cart style, where a repeated algorithm of movements could simulate a static turn with a much smaller space requirement than the two wheel drive. Because of this, while more complicated, the two function motion design can be utilized by selecting this chassi, and will be explored more in depth as it would be possible with a single sensor.

To match the shopping cart style of motion, the two basic functions that would be implemented would be forward and backward motion, and a procedure to turn “in place.” What separates this chassi design from the double axle, two wheel drive is the need for significantly less space. By being able to turn both wheels at a time, due to the lightweight and powerful nature of chassi with this design, the sudden change in axle angle introduces much more deviation from a calibration standpoint, but greatly increases the mobility of the vehicle constructed. The reason this is important is that the absolute need for multiple sensors is mitigated (although desirable), but leaves the number of functions and complicated navigation routines smaller and more handleable. Calibration, therefore, would be a mixture between the two previously discussed chassi.

By having full control over the size and design of the arena being constructed after the dummy agent has been finalized, the arena would therefore need sufficient space in corridors and horizontal spacing that is enough for multipoint turns to turn around. This space, of course, would be found through extensive testing of a multi-point algorithm that may require the use of collision switches (YL-99 discussed further down in this section). Walls may be required to be more sturdy and, because chassi with this schema are typically smaller, may not require the same kind of real-world vehicle T-shaped dead ends for traditional 3 point turns.

While the forward and backwards calibration is relatively straightforward, the principle of only requiring one rangefinder sensor is based on the multi-point turns that are capable by using an extreme turning radius available to dynamic axis 4 wheel drive vehicles. A simplified diagram is shown below.

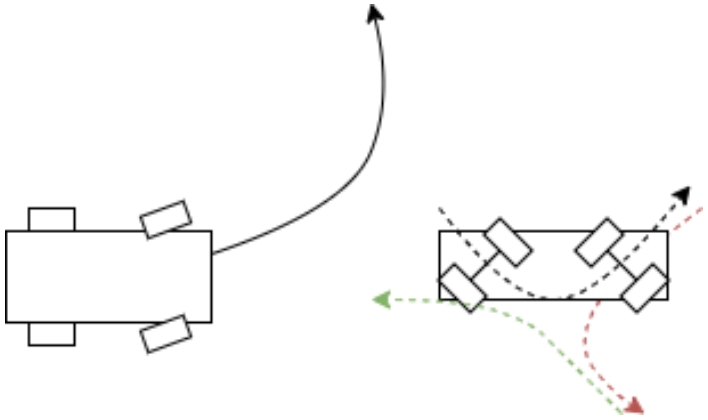


Figure 29: Wheel Based vs. Axle Based Maneuverability

As shown by the difference in axle rotation against traditional wheel rotation, lightweight axle rotation can perform a star-point turn (with increasing number of points) depending on the shrinking size of the corridor or turn that needs to be made. Take note that this only involves complete 180 degree turns, where fewer points or even a less steep angle of chassi rotation would be needed for a turn.

Because of this chassi still requiring forward motion while turning, it is likely that the predetermined wheel and axle movements that are pre-programmed per RC car will stay consistent. Supporting the less extreme turning angles and more stationary motion is the ability to isolate wheel and axle motions. For instance, in the following diagram, at the

peak of each point in a star turn, the axle is able to stay stationary by making the wheels rotate in opposite directions for the same wheels in an axle. In the diagram, to find its maximum axle rotation, the left wheel rotates backwards while the right wheel rotates forwards. By rotating at equal speeds, the axle rotates around its center without moving the actual chassi.

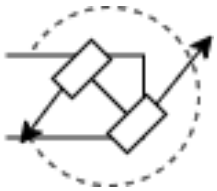


Figure 30: Stationary Axle Turn

Being able to change wheel orientation while keeping the main body static enables the Quad Wheel Dynamic Axle Chassi to perform more advanced maneuvers, which enables a simpler action distinction by using tools and technologies that are readily available, rather than re-designing the chassi.

In order to save space this explanation is truncated slightly. This module would use a similar calibration method to the method used to calibrate the turning distance of the shopping cart style chassi. Because this method is slightly more complicated and introduces more deviation, logically the turning distance from each forward wall would exceed the distance than the module to the end of the chassi, but can be found through finding the area that a full star point turn (which would change specific to each chassis maximum turn radius) takes, and using the radius of the complete 180 degree turn, which simulates the maximum area required. Additionally, however, a rear collision module or a reverse facing sensor may be required, although in theory a tight star based turn would not run the risk of impacting a wall given the correct detection distance was given.

This sensor design may require similar calibration and considerations of the figure titled “Alternative Sensor Design and Placement.” Similarly, this design is subject to change based on arena requirements and sensor needs.

KY-037 sensor Thresholding and Calibration Needs

This sensor is described in detail in the Hardware Sensors Discussion Section above, and for simplicity purposes, alternatives will not be explored. This sensor works by taking an absolute decibel measurement from the module's surroundings and returning a value to the console that can be read as a floating point value. Because of this, regardless of the alternative sensor being used, the same procedure for testing and calibration will be used.

Unfortunately for this sensor, it would be very sensitive to the motor features of the rest of the dummy, and may require some sort of isolation from the sounds of the agent caused by various sensors, but most likely the wheels. Because the function of this sensor is to interact with specifically placed stimuli around the arena, the likely location for the

sensor is slightly above the agent. By elevating the sensor slightly, it may place distance between the module and the motors that power the wheels, and possible collisions that may occur that also make noise.

While the primary justification for having the sensor is to track distance from the goal of the game (which is the exit or potential shortcuts), depending on the volume level of the buzzers in the arena, there may be two separate thresholds to explore. For thresholding, there are two cases that need to be explored, and are based on the intensity of the buzzers being used.

The two cases for the buzzers is a persistent decibel level that can drown out ambient noise, and a quieter decibel level that is only detectable when it is in a certain proximity. However, not unique to either buzzer level is the required accommodation for motors that are being used to move the learning agent around. This threshold, which shall be decided first, is based on the noise that is made when the agent is moving. Logically, if this sound is too great for even the distanced sensor to detect other sounds, the sensor would no longer be isolated, and input from the sensor would only be taken into consideration when the agent is not moving, and thus only environmental sound levels would be considered. The two following use-cases will assume that motion does not overwhelm environmental detection, and will thresh-hold for positive response when the buzzer gets louder, or negative response when the buzzer seems to get quieter or disappears.

Because it is mentioned many times, the procedure for thresholding is as follows. First the ambient return of a quiet environment will be the return value of the sensor, and the baseline to compare any other sounds to. A slight overapproximation will be the level that the sensor listens to before deciding that any other noise is significant. A second level of ambience must be measured, however, for when the agent is moving. These two are the only sound levels that should be considered normal, as any other sounds must be contributed to the environment. Because the sound of the agent may be loud, however, it must be noted that other sounds that fall beneath this threshold must be and are neglected while the agent is moving. This sound will be assumed to be common, as the machinery does not have many other functions that make noise given that the chassi does not change. There therefore must be some kind of synchrony to alert that values beneath this threshold are to be neglected while the agent is moving. This may change in the following cases discussed, but the two assumed thresholds will be named ‘Ambient Sound Threshold,’ and ‘Ambient Moving Threshold.

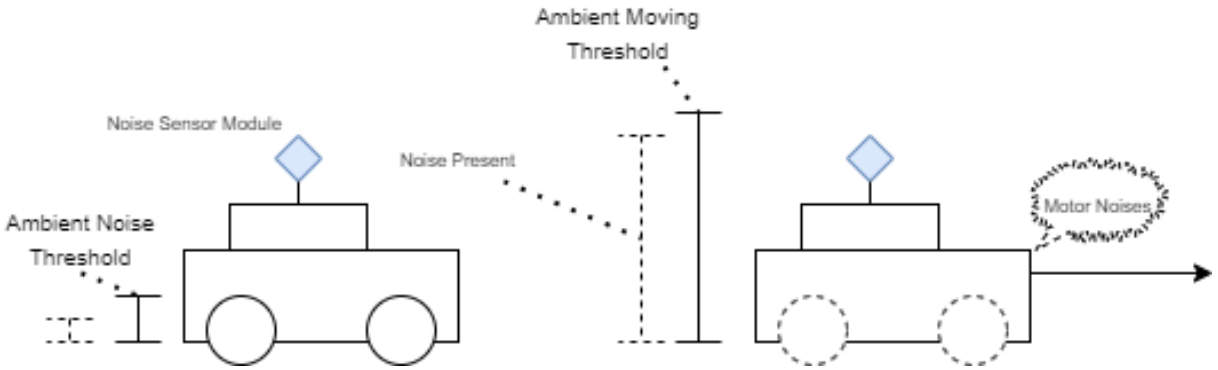


Figure 31: Basic Noise Thresholding Visualization

In order for noise for the sensor to be significant, it must be a level above the threshold that is measured when the agent is either moving or not moving. A similar principle is applied to all sensor thresholds, but is easier to distinguish in this case. Note the elevated noise sensor module farther from the motors of the agent.

Sound Sensor Loud Buzzer Accommodations

In the event that this buzzer case is true, then the sound detector when the agent is not moving would assign a certain reward to its position and pattern of motion if the buzzing is getting louder. This implies that, past the certain ambient noise threshold the sensor would measure when installed, there would always be some kind of return from this sensor from any point in the arena. Nodes for buzzers in the arena may then be placed at certain distances where they do not interfere with each other's specific reward values, or a single very loud buzzer at the exit would constantly signify the need to venture towards that direction.

Limitations of this model are the constant need for the sensor to return information to the commander agent. Because the module would always be active due to always having a loud enough sound source, it may be important to disable sound and sensor collection during trivial motions or movements, as the added sound from motion can also interfere with the desired effect the buzzers have. However, juxtaposition to this assumption is the fact that there will always be some kind of reward and response associated with every action that the dummy agent takes. This is due to always having a value to measure, and because the buzzers rig the arena in such a way that they give a sense of distance, there will always be a conclusion to be drawn. A non-technical limitation is that a very loud buzzer would be very annoying for testers.

Specific thresholds of noise that are relevant to this case, are sounds that fall beneath the ambient moving threshold and the ambient sound threshold. Sounds that must raise an interrupt or are significant for the code and decision making process include these sounds because, if they persist, they must not be random collisions and are likely related to the buzzers placed in the arena. Therefore, some kind of duration sensing algorithm will likely be utilized to signify that a buzzer is being detected by this sensor. Because the buzzers do not change in intensity, if a sound is constant and does not fall within the ambient 'moving' or 'sound' thresholds, if it is a certain intensity it must be due to a buzzer. Specific buzzer thresholds and assumptions will be discussed further below.

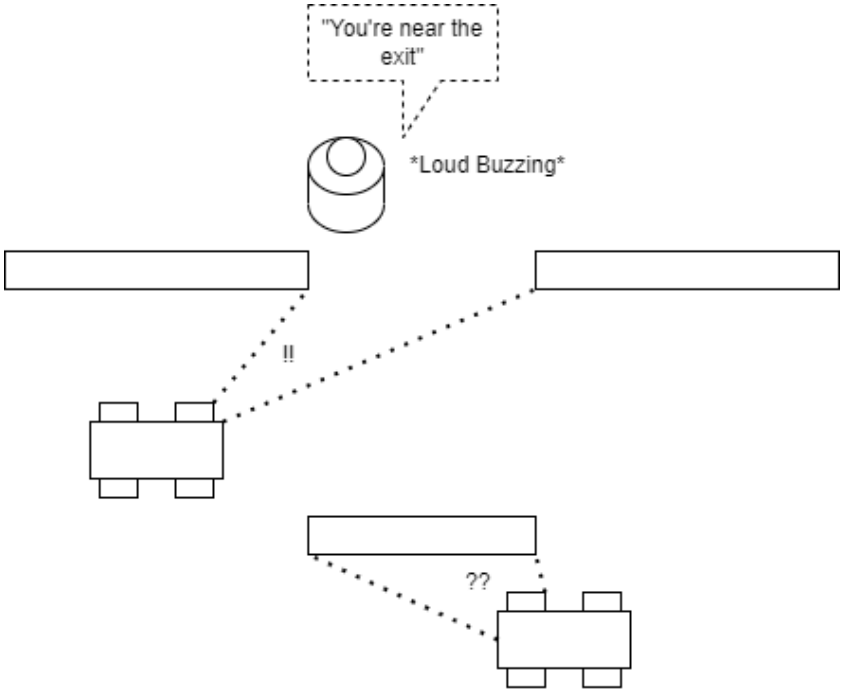


Figure 32: Agent Loud Buzzer Simulation Example

While both agents have a sense of the direction they are going due to the volume of the buzzer, the closer agent raises an interrupt that may interrupt its basic code navigation cycle in order to approach the sound more accurately because the sound it interprets is growing louder as it moves in that direction.

Sound Sensor Soft Buzzer Assumptions

In the event that the buzzer(s) being utilized within the arena are not loud enough to be heard throughout the entire arena, different thresholds based on a different design would be desirable. This difference is due to the sounds that are being picked up having different meanings compared to omnipresent noise from a loud buzzer configuration.

By having soft buzzers, this either implies that a single exit buzzer would require an extreme reward response from being detected (as it would signify that the agent is close to the desired result), or that the proximal route is important in some way and that some kind of modification to the decision making process needs to be raised. Having soft buzzers, however, almost guarantees that the noise from buzzers would be nullified by any noise that the agent makes, meaning that the configuration that supports a successful task would only be detectable when the agent is still and specifically listening for sensor stimuli.

Some Limitations of this model, therefore, include a less constantly present detection of noise, and the fact that thresholds must be more strict. This implies that the sensor is less orientation based leaving the agent more lost in a vast arena, and more novelty based as the sensor would only be utilized in code for very specific moments when soft sounds can be listened for. These novel levels of detection, requiring a certain time of

active listening, would be based on constant level of sound, as varying sound levels could be contributed to ambient sounds that may be other project displays in the same room, or conversation between non-arena entities in a close proximity.

The procedure to calibrate this kind of sensor detection, however, falls in line with repeated testing post construction, similar to the other previously discussed modules. By detecting and potentially graphing the levels of ambient sounds can be averaged to sense when a sound level has been persistent enough at a certain level for an interrupt to be raised. For instance, if sound levels persist on average at the same level for half a second, the average detection time between actions must be at least .6 (or a larger interval). Detection time, however, would be a running timer that only counts when a sound level is persistent at a critical sound level. This timer would need to reset every time a sound level changes, so that ambient noise is ignored and sound levels at non-critical and default thresholds are detected. This principle and algorithm is visualized below without sound durations.

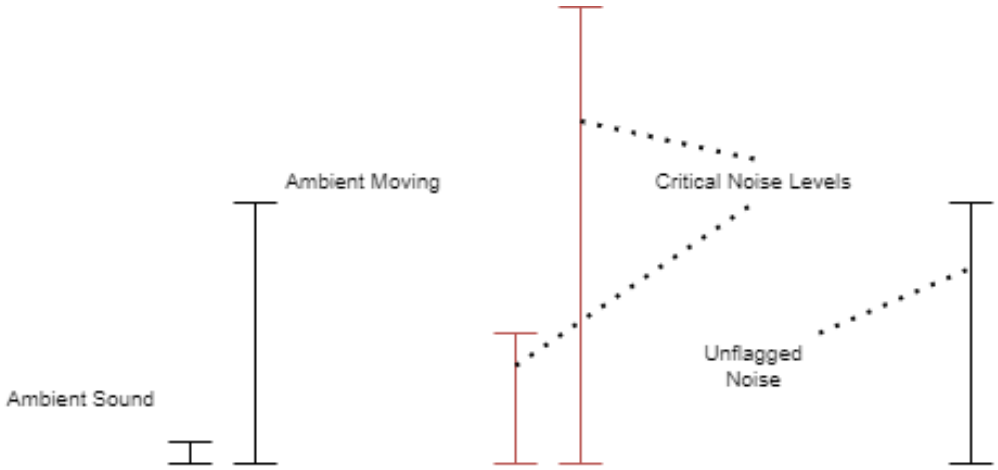


Figure 33: (Novel) Critical Noise Level Diagram

For soft buzzer configurations, the buzzers may not flag any sounds for being too far away from any agent sensors; however, constant sound levels that fall between or above thresholds may throw an interrupt to signify that a significant stimulus is around a learning agent.

Limitations of these sensor designs is that because only sound volume is recognizable, in the event that a loud and continuous stimulus like a buzzer from outside of the simulation area is present, false information may be passed during a run, potentially adulterating the trial run happening. Similar to this, if buzzers or sounds that happen within the arena are ill-placed and are unaccounted for, a similar reaction will happen, and the potential for improper detection and assumptions built into each reward function and q-table may occur. Therefore, the arena buzzers or any other stimuli that are designed to be detected by this module must be carefully constructed and calibrated together.

YL-99 Switch Calibration Needs

This sensor is the simplest and smallest sensor that may or may not be required. Because this module is based on the potential inclusion of a rear-facing collision sensor or not, it is mostly just included to interrupt a navigation algorithm that would make the agent turn around or re-center itself based on a separate sensor. In this case, that separate sensor is the ultrasonic sensor or TFMini LiDAR sensor. Fortunately, this module would not require any thresholding, and is typically functional on connection. By returning a value when the signal is high (when the switch is closed for active high configuration), this switch will simply indicate if the agent runs into something from the rear. This switch may be used for a variety of calibration needs for navigation in general, however. Primarily, however, it is intended as a rear navigation interrupt switch that may interrupt any currently active navigation process.

A collision from the rear will always be coded around if possible, and serves solely to indicate that some kind of navigation algorithm has gone wrong, specifically the rangefinder sensors. Because of this, this sensor would trigger a re-route or learning discard for the current action, as clearly something has gone amiss. Unfortunately, as the dummy agents will be programmed to learn and function at a low and simple level, the switch cannot determine what kind of collision occurred because it will likely either be another agent, or simply an off angled wall. Because of this, unless multiple collision sensors are active (which would preferably be avoided), the only calibration needed is to ensure that interrupting the current function or algorithm is problem free.

Alternative uses for this module and calibration alternatives include switches that only trigger if the front of the agent impacts at the corner of the chassi. This implies that only a slight re-navigation algorithm would need to be employed, and learning can occur from the specific action without needing major re-orientation. For instance, if the right corner of the chassi runs into an object and the central rangefinder does not sense a wall, the agent may only need to reverse slightly with a left wheel orientation that re-aligns forward motion slightly left of the original path, which the agent will assume to still maintain a straight navigation.

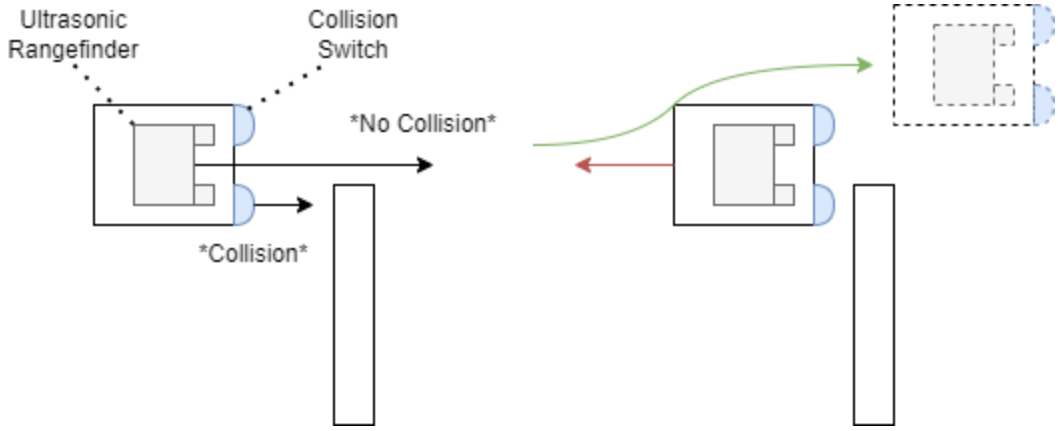


Figure 34: Simple Reroute Visualization

A simple re-routing algorithm makes the bot back up, and turn slightly left, then right again to make space on the right side of the agent. This is an example of solving problems with less complicated sensors than designing a vision algorithm with multiple rangefinders.

Orientation Module Thresholding and Calibration Needs

While an essential part of navigation and learning for the robot, this section has been temporarily truncated while additional research continues for alternatives for this sensor. Initially the most promising stimuli and sensor coordination that was being considered was the RFID toll both schematic explored previously in arena stimuli and construction, and one of the alternatives and likely sensor combinations is painted or colored planes with different LED and color sensors. Another potentially explored alternative is having different LED lights to coordinate with a photoresistor that detects light levels similar to the sound sensor discussed previously. Nonetheless, although all these sensors and their coordinated stimuli are different, the concept and process is the same, to recognize different parts of the arena based on the rig that the arena is meant to impart on the learning algorithm the agents employ. Expanding on this idea, is that the arena may still be redesigned around these markers, and that these sensor thresholds are ultimately dependent on what the sensors absorb.

To reiterate, this sensor, whether the previously discussed LDR Photosensitive Module (LM393) or the (TCS3200) Color to Light Frequency sensor is used, the goal is the same. Each sensor is designed to recognize a planted stimuli in the environment around it that will have some kind of unique signature to it. In the case of RFID booths, this would have been some kind of sensor that reads the signature reflected by a certain RFID tag to indicate where in the arena the dummy agent currently is. This would have been the most straightforward approach, but unfortunately is limited by expenses and sensor quality. These sensors that would have been within our budget range would not perform as needed for the purpose that they serve without spending an unreasonably much larger sum of money, and thus alternatives are being considered.

In the case of basic photoresistors, potentially, different patterns of blinking lights would have to have been applied above the agent and shined directly onto the bot in a

kind of morse code to signify what kind of shortcut or location the agent is currently in. Expanding upon this, an alternative to unique location stimuli would be a different light intensity, which could be accomplished by using a different color LED as each color if sufficiently far apart on the color spectrum would return a different value or range of values. Unfortunately, because of the uncertainty of detection for simple input sensors, these methods are unreliable and another alternative must be explored. In any event, the calibration process for this would be similar to the sound calibration process where a certain threshold would have to be set for ambient and no-light to detect changes in the environment. This, however, would be very difficult to standardize without great consideration of the lighting conditions of the room, or at the very least isolating the arena being constructed. These alternatives are being considered for the construction of the final arena as well.

The most realistic and therefore most explored option moving forward is the use of the TCS 3200 Color to Light Frequency Sensor. This sensor is active as a photoresistor that is also sensitive to reflective sensitivity. This sensor includes white LEDs that shine light on the surface being detected. Unique stimuli that the arena would therefore reflect, however, is simpler in this case, as different colored surfaces can be used.

Exploring the TCS 3200 Color Sensitive Photoresistor

Construction wise, this photoresistor will likely be facing the ground on any side of the agent (likely the front). The placement of the sensor is due to the purposeful avoidance of frontal collisions due to the rangefinder positioned facing forward, and facing downwards would enable detection of colored planes beneath the agent, which can be consistently detected based on the size of the planes. This enables the detection of passing a shortcut marked path or special interrupt code routine. Because of this placement, however, making sure that all known values that the photoresistor returns are important. Without going into too much detail, the returned values from the photoresistor being used are in RGB. Detailing how these values being returned will be segmented into unique IDs will be specified in the following section.

Calibrating the TCS 3200 Color Sensitive Photoresistor

Similarly to the previously explored ‘RFID Toll Booth and Tag’ concept, each shortcut or important sector of the arena will therefore be colored special hues. For instance, in the event of two paths leading to the same exit, a shorter path may be colored green, and a longer path may have a red floor. By traversing a certain path, a photoresistor may have detected that a majority of the path was “non-ideal,” and a certain kind of reward reflecting this can be returned based on the detection of red floors. Because, however, the photoresistor needs to understand that the floor was red, this needs to be unmistakably accurate even in different lighting conditions. To accomplish this, the procedure for calibration would require detection in a room of different colored lights.

Using a variable warmth light, a room that contains the arena will have all the planes that the arena can possibly hoist. Additionally this may be used to maintain the

position of the agent in the arena, and detect when the agent has successfully escaped the arena. This idea will be explored further in a future section, but for all of the valid plane colors, there must be a range of RGB values that encompass each color of the planes that are desirable. In order to test these values, the minimum and maximum of RGB that is detected when the photoresistor is held to these planes will be recorded, and these ranges will be found by changing the warmth of the variable bulb that illuminates the arena and the room the arena is contained in. These variable bulbs are to account for the common types of lights that may be present in any display room, and with various lighting conditions. Although the values may not directly represent the warmth and brightness of a light in a room, the important values must encompass any condition the planes might be detected as, and avoid detecting a different color.

Although simple to type in concept, this will require extensive testing of all of the planes and many recorded values for all red, green, and blue channel detections.

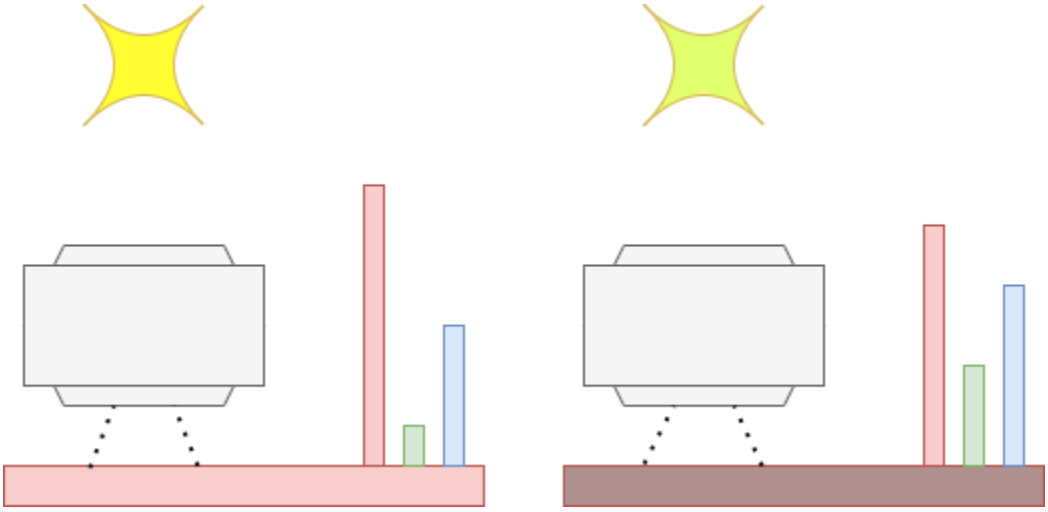


Figure 35: Different Lighting Conditions

The same color sensitive photoresistor may return different values for the same plane of color given different lighting conditions. The plane is the same for both conditions, but the slightly cooler and green-tinted light shining on the plane shows that the green and blue values are closer for the right side of the diagram, even given that the plane is the same

Because of the concept illustrated in the diagram, the upper and lower limit for each R, G, and B, value that is returned for each plane must be recorded, and the planes of different colors must be cross compared with each other to not interfere with each reading, so that locations within the arena are interpreted as intended.

Limitations of this design are the fact that there is room for misinterpretation between the same planes and different dummy agents due to small interferences between connections and the room that the arena is placed in. While these may be calibrated again for different colors, in the event that so many colors are needed that multiple shades of the same color are required, there may be inconsistencies in what planes are interpreted as, which can lead to a sort of disconnect between where the agents are actually and what the commander interprets. Of course, while this may be checked by a certain consistency

check, these may only be so sophisticated with the level of what we're working with, and may not solve all of the inconsistencies that occur. These inconsistencies may only be magnified by possible communication errors, but overall this model does have a more reliable detection method than single RFID tags or light based encoding communications.

Fortunately, the inconsistencies of this model are counterbalanced by the ability to color whole sections of the arena, leaving little to no possibility to miss passing by a 'node' in the arena. This is due to a whole shortcut passage being highlighted green against needing to end up in a specific position that the arena stimuli need to coincidentally detect. This enables a range of arena designs that better simulate the robot-vision of future q-learning detection, which would likewise not need some kind of verification stimuli, but can read the environment on their own. By scaling this type of detection to the agents, more robust arenas may be created.

Commander Agent Sensors

Potential KY-037 sensor Thresholding and Calibration Needs

Expanding upon the previous discussion, the KY-037 sound detector may be positioned in a way to give reference for the learning agents in the arena. By also detecting the same constant stimuli that the other agents in the arena interpret, there may serve a second alternative to ascertain where in the arena the agents stand. By possibly triangulating the distance of the agent to the escape buzzer, there may be ways to further enhance the information that the command agent receives in terms of distance and orientation to the exit. This may be useful for when the arena is changing, and because the commander agent is elevated, this is a way to make the sound modules more consistent across different scenarios.

The procedure to calibrate the sensor itself in terms of ambient noise is the same. (See section titled KY-037 sensor Thresholding and Calibration Needs). Because this process is relatively complex and reliant on arena design, in this case the calibration for the elevated sound sensor is important to adjust to the sound of the buzzer as well. The important distinction is that this sensor does not need to be active during trial runs and data collection. This is because the commander agent does not move, and unless buzzers are to be implemented onto the learning agents as well, this sensor has no active purpose for running.

However, this sensor may have purpose in the event that ambient noise is an issue, and may be a disablable module. By this it is meant that the commander agent may detect the level of noise independent of the buzzers in the room, although this may only be possible if the soft buzzer approach is implemented. By using an external sound sensor that can, in place of a complicated sound analysis function, detect that ambient noise in the room, this sensor may be the robust and flexible calibration that the dummy agents need to properly utilize their "echolocation" abilities.

A large limitation of this sensor being present, however, is that it introduces a sensor that requires interrupt management and conscious management to synchronize it with audio sensors that are located on the dummy agents as well. This adds a great amount of code density and is a secondary feature that simply augments an essential feature, so it may be added simply at first and gradually expanded into its full potential and

intention. Additionally, the addition of a sensor as such on a relatively lightweight pcb may interfere with the weighting of the commander agent mounting, potentially harming the modularity of the module with respect to the arena, requiring more calibration as the location of the sound sensor may matter if not disabled. Due to this, the logical following action is to include an optional flag to enable this sensor, and potentially different modes of operation to make sure that it does not interfere with essential sensor communication.

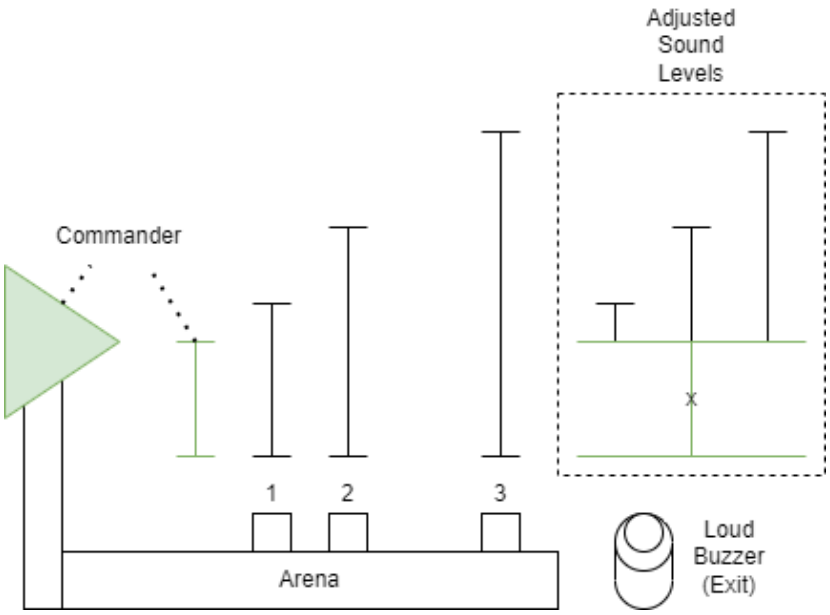


Figure 36: Adjusted Sound Offset

The sound detected relatively external to the stimuli of the arena can serve as an offset for the agents contained within the arena. Given an instant communication this can be used in place or in tandem with the previous approach of time based audio level differentiation.

While this sensor is largely optional given the in-depth testing and implementation being used to ensure that the dummy/learning agent operates as intended, this is an additional feature that may need to be added onto the printed circuit board and detachable commander agent base, and is worth investigating further. This module is largely included because of the expected extra active pins and space that come with the advanced Raspberry Pi 4 board, and additional modules may be explored in the future as well.

An additional module which isn't necessarily external hardware that will be utilized is the WiFi module that is built into the Raspberry Pi boards that will be used. The WiFi hosting capabilities and method was investigated previously in the section titled Prototype Communication Implementation. In short, this board that hosts the server that the dummy and commander agent communication will be tested on will be done in a way that ensures that each of the sensors active can send data that is interpretable by the commander agent with a separate code protocol that runs when simulations or data collection is not active. This protocol, however, will be inactive when simulations are run, and data that is sent thereforth will be assumed to be accurate to the degree of the testing protocol expected.

Following limitations of using the on-board hosting service are the inconsistencies that come with connecting a relatively weak connection onto a multi-device hosting network such as the public wifi in educational institutions. Nonetheless, the html access protocol will enable secure and consistent communication.

Prototype Testing

This section includes a combination of the hardware and software components of our Envoy Commander that would require testing to match their expected behavior when running our eventual prototypes for a majority of Senior Design 2. To note, all of these are projections for how we would go about testing upon finalizing the orders and receiving said parts as we expect to begin this process around the December to January range, so these methods are subject to change upon potential deviation from our current component choices either due to general faults or change of direction for that project section. The current tests that we plan to highlight are our PCB, the software functionalities, and the combined efforts of our project's interactions with one another to form a proper system.

Hardware Testing

Even though the software components of our project will be of a higher importance, we cannot discount the testing of our physical descriptions that allow us to use Q-Learning in a more realistic sense, and this requires us to implement testing procedures for our Learning Agent, its Dummy Agents, and the Arena itself which will be detailed in this section. This will include subsections for each major topic with some general concerns like power or communication being explained in their generic fields. We assume a majority of the parts that we already own or look to order in the future are up to our established standards and in general working order, but regardless we will continue to test them in methods that reflect the respective component, for example power would be ensuring our PCB's, MCU's, their respective sensors, and additional components are inputting the correct and safe amount of power within a certain threshold level that will be based off their datasheets.

Learning Agent Testing

For our learning agent, which will be constructed from our PCB involving our RPi4+B and general logic/power converters, as well as additional computational assistance measures to support our Q-Learning algorithm and its communication of its data to and from our Dummy Agents. We will rely on the software outputs to determine their functionality as a majority of problems would arise from that section as we assume the parts we ordered for our Learning Agent are operational, so to ensure that the proper data is being produced by sending sample data points as well as multimeters/oscilloscopes to ensure that the voltages and signals are aligned with our expectations.

Microcontroller/PCB Testing

For our “Commander”, this MCU will be tested through the pins for basic functionality along with its additional features, as it is our PCB, that contributes to additional testing measures such as the performance of our MCU which will be able to withstand heavy stress as we chose to pool our budget into buying a more than capable component capable of handling all the necessary tasks of the Q-Learning algorithm as well as the general communication system that enables the Dummy Agents to proceed throughout our Arena. Verifying this system of relaying information between the Dummy and Learning Agents can be done without actual movement, just the concept of relaying information to the Commander as a testing measure.

We would also begint to test the power capabilities of our MCU and its various added components by utilizing a Multimeter to match what we would expect from our data sheets, such as a max of 5V at 3A for our RPi4+B and a power consumption range of 2.7W to 6.4W, all that would be detailed when running our intialing demonstrations to ensure the safety of the components we are using.

Sensors Testing

Getting to the specific sensors of our PCB, these will primarily focus on our ESP8266 module which will assist in providing a stable WiFi connection. We would proceed with testing this component by sending additional sample data from our devices to ensure its compatibility with the Learning and Dummy Agents. An additional module that we would potentially add was the KY-037 MIC system that would detect sound up to a certain range and output a digital signal that our PCB system would interface with to determine if a certain threshold is met to warrant sending a signal out to create future commands. To utilize the analog data that we would obtain, we would have to purchase an additional ADC module that would convert the analog input into data that the Pi can interact with. More testing of these specific module features will be detailed in the Project Protype section. Smaller systems such as our power and logic converters are somewhat idle systems as their simplicity renders them simple to test and even more assumed of their functionality upon using them for our project

Dummy Agent Testing

The “Envoys” themselves would be of a rather simplistic design in favor of portioning our budget towards the Learning Agent’s ability to enact our algorithm to the highest degree, so that these Dummy Agents in their crude forms would do the most simple of tasks, and in this case navigating a randomized arena. To properly test their ability to move, we would perform initial movement tests based on whichever movement option we end up deciding is the optimal one, and to see their ability to communicate and sense their environment, we would provide them with a wide array of sensing options to ensure a navigational consistency wherever in the maze it is located

Movement Testing

For our pathfinding system we will encourage optimization findings to escape its current location in the maze, and in order to accomplish this basic step we need to verify the vehicle's ability to turn on a surface to be determined and respond to commands from our Learning Agent to enact this Q-learning algorithm. Since the terrain itself will most likely be of laminated floor or smooth surface of some form, we will not have to worry about accounting for environmental factors outside of the arena it finds itself in. However, we will still need to support the different possible movement styles that we have prototyped and ensure the motors can properly make specific turns and actions without producing inefficient pathways that could result in unnecessary loss in rewards. Since there ideally would be two or more dummy agents, we would also have to account for potential collisions that would affect their navigational abilities between them, so to avoid potential testing dilemmas we will align each agent with a certain task in mind to discourage any overlap. We will still track their testing results as each car will not only functionally work apart, but physically our cars will be of different speeds and sizes, making baseline testing particular to each agent which will be detailed in the prototyping section further above.

Sensors Testing

More details of our numerous sensors can be found in our prototype section, but this field contains the data we would interpret when receiving these parts, more specifically their relative ranges for their power consumption, current figures, and additional testing tables we could establish to get a better comparison picture. When tested, we will first try on its base levels its functionality without a Dummy Agent attached and additionally after its implementation to ensure no issues were made when mounting the respective sensors onto the agent or during the initial phase of testing we missed a potential problem area that would then further be described in the malleable prototyping section of this document.

Arena Testing

Any testing needed for our environment will fall under durability and connections between the potential identification marking methods that we have planned and the Dummy Agent for it to interpret and send to our Learning Agent that would represent checkpoints and potential shortcuts, with these methods being up to experimental change, we will rely on more simulated testing procedures for our arena as the majority of its complexity will start off weak and gradually expand as the project progresses. The components in the maze that we would physically need to test would be any LEDs that assist our Dummy Agents and determine the operational ability of each of them, and if we choose to complicate our arena we would have to install additional power supply locations if we were to continue with our shortcut upscaling as we determine the feasibility of each layer of this navigational system.

Software Testing

The “Envoy Commander” project was founded in the virtual domain, so this section will detail a majority of our testing procedures and requirements that we would encounter during the debugging stage. For scaling purposes, simulated environments are a necessity to any project, as we will additionally utilize this concept to detect any potential errors found and replicate them to some degree in the physical elements of our project. These environment will be module-focused, meaning that we would not be able to interact with the actual components aside from our main applications system as these inputs would have to be created virtually, although a downside from testing potential intractability, this still provides some insight that would not be possible without this simulated process, mostly being tested with arena interactions between the dummy agent and the learning agent to see their reactions from our Q-Learning system.

For these particular modules not included in our main simulation, we would apply basic operational tests that would apply some basic input and have it align with a response ideal to their configurations. This is noted below in the system test tables that describe each of the potential modules and the results we would intend to verify.

Q-Learning Algorithm Testing

Clearly a major aspect of the software engineering in this project is the machine learning foundation it comes from. The multi-armed bandit problem is very commonly referenced in reinforcement problems because of its versatility and has been implemented in many different variations, languages, and use cases. However, every problem has different solutions and it is important to reconstruct and analyze the problem with respect to the clients’ needs. It is more important to dry-run test the solution to make sure it works before committing to hardware. As many professionals often say “measure twice, cut once”. To “measure” twice I will be constructing and implementing this problem in a custom OpenAI Gym environment.

To refresh, OpenAi Gym is a “*toolkit for developing and comparing reinforcement learning algorithms.”* It is commonly used for beginner reinforcement learning algorithms due to their already expansive set of pre-built problems such as SpaceInvaders, Pendulum, and FrozenLake. Fortunately, due to its open source code, we can also construct environments by following a set interface. Each environment essentially needs five things.

```
import gym
from gym import spaces

class CustomEnv(gym.Env):
  """Custom Environment that follows gym interface"""
  metadata = {'render.modes': ['human']}

  def __init__(self, arg1, arg2, ...):
    super(CustomEnv, self).__init__()


    # Define action and observation space
    # They must be gym.spaces objects


    # Example when using discrete actions:
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)


    # Example for using image as input:
    self.observation_space = spaces.Box(low=0, high=255, shape=
                    (HEIGHT, WIDTH, N_CHANNELS), dtype=np.uint8)

  def step(self, action):
    # Execute one time step within the environment
    ...

  def reset(self):
    # Reset the state of the environment to an initial state
    ...


  def render(self, mode='human', close=False):
    # Render the environment to the screen
    ...
```

Figure 37: Open AI Gym Environment Setup

Action and Observation spaces represent what actions the agent can take and how many states are present in the environment. Additionally, the environment needs step, reset, and render functions. The step function is perhaps the most important as this is what will present us with our new state, the reward received from taking the action to land up in this state, and whether we have reached the end. These functions can be supplemented with others as well but these are the cores. So to test our code, I will be creating an environment that will closely resemble our problem. Thai will also make it a bit more modular, allowing us to test out various reward functions, maze shapes, and collaboration techniques while maintaining the same boilerplate code.

Another benefit of testing with this method is that the learning algorithm can be tested without a dependency on the hardware. This is important because apart from portability issues, there is a concern with power issues. The wireless dummy agents, being in the field, will have a need for battery power. Similarly, the arena itself may have battery operated components. Reinforcement learning takes a long process depending on the reward function and learning rate, especially when we factor in sensor detection and movement speed, as well as what we would consider a step, an episode could take up to an hour at a time. Given the amount of communication and sensor detection that takes place, the battery may die before the learning finishes. One way to prevent this from happening would be to establish “learning” checkpoints so that if the learning process were to indeed be interrupted, it would at least be able to resume at a later point. It would either assume it's in the same state as when it ended or it will start a fresh episode but maintain its Q table from before. I propose another way. If we can accurately simulate the environment, to a certain degree, we can complete the learning process virtually, or offline,

and then simply give that Q table to the learning agent so that no power is wasted. This solution is definitely not scalable in any sense but if in an industry-level implementation, more time and resources will need to be spent on learning and QA.

Connectivity/Communication Testing

Connectivity is another crucial part of this project and should be thoroughly tested before deploying this project into a use case. There are many points of failure and even more ways of failure. To name a few points of failure, there is the obvious communication between the commander and the field agents but there is also the possible intermediary communication to a server to relieve the load from the Raspberry Pi commander. This would create two more points of failure. This assumes that we will be pursuing the option of WiFi connectivity instead of Bluetooth. Bluetooth connectivity, while having benefits over WiFi, comes with its own issues as well. The data would have to be serialized and sent over and this could result in bit errors and further synchronization issues. To avoid this, proper testing guidelines will be set.

The first step to testing connectivity and communication is to ensure messages get passed in the right order. TCP/IP message passing calls for handshakes to ensure that messages get passed in the right order, unlike UDP which does not care if the data is received on the other end, TCP doesn't send another message until it ensures the other one was indeed sent. Each has their use cases but for our problem, the messages have to be confirmed since we should not be able to request new states until the change from the last one has been confirmed. So when testing messages, we should choose the communication standard and send a plethora of data over the network and manually test that the messages have been received properly.

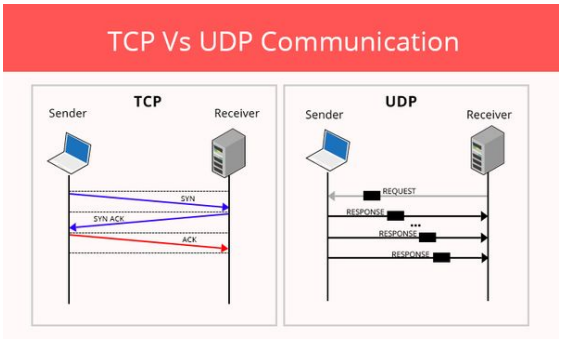


Figure 38: TCP vs UDP Communication standard

The next step in connectivity testing is range. Since the project hinges on the fixed-location commander being able to communicate with the dynamic field agents that are moving towards a goal, it is important to know the agents will be able to communicate from all points in the maze. To test the communication range, we can infinitely send messages for the commander to simply receive and print out. Then we can move the field agents around the maze in a grid format, making sure that the messages are received at every time step, pausing at crucial points to make sure there will be no interference. Then we can have the field agents traverse the maze with manual input from us. At every step it takes in this traversal, we will send out the current location in the maze traversal and

record to make sure that each message got received. Along the same lines, when we add a second field agent to the mix, we should incorporate testing for that agent too. So we would run the same algorithm and send messages again. This would simultaneously test the commander's ability to receive messages from multiple points of contact and maintain the order.

The final step to testing the communication would be to check our communication algorithm for responding to each agent. This is equally important because we want to give the appropriate instructions to the appropriate field agents. In order to increase scalability, we also need to make sure the instructions are sent back in appropriate order. To test this, I propose a simple program similar to a marco polo scheme with multiple players. So when an agent sends out a message "Marco" with a timestamp and unique identifier packaged, the commander will respond back "Polo" packaged with the ID. The logs of these messages and the timestamps will allow us to check if the communication is working.

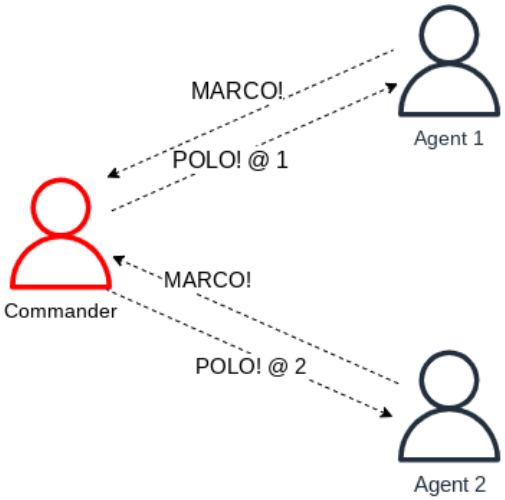


Figure 39: Testing Communication response

Initial Documentation Conclusions and Influences

Project Alternative

Due to the ambitious nature of this project and the obvious supply constraints that come with it, a sort of plan B has been formed to accommodate the inevitable issues that may come with the project. Even though it is a pretty established project, there are issues out of our hands that should be accounted for. For the alternate project, much of it stays the same but the game changes significantly.

It is much like the BattleBots game in which there are two bots placed in an arena and the objective of the game is for one to push the other out of the ring in a sumo-wrestler style battle. In the traditional game, the bots are controlled via a remote control by their respective owners so the strategy is determined in real time by the human and adapted to the play style of the opponent.

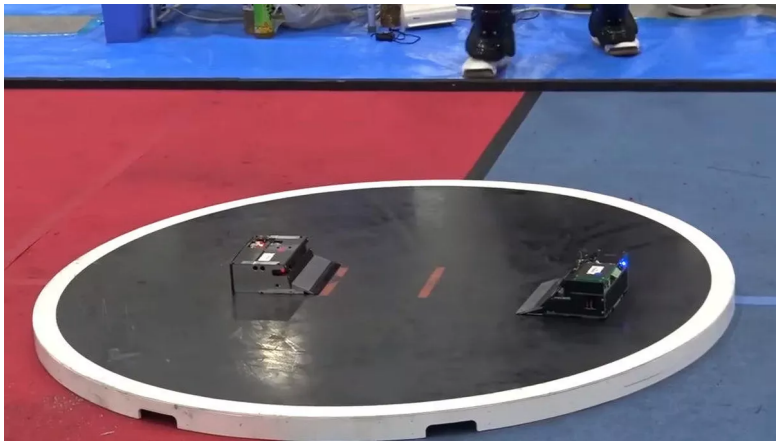


Figure 40: BattleBots arena

In our version of this game however, we would maintain the dummy agents and have two of them. The opponents would be simple objects in this scenario, maybe an object that is large enough to be found by moving around. It would be irresponsible to make it anything small because the dummy agents or the commander agent will not be equipped with cameras or any kind of object detection ability. The commander agent will still issue commands to the dummy agent based off of a model-free reinforcement algorithm and the arena will still issue progress to the commander via a custom pcb that will be integrated into the arena. The obvious downside to this is that while it solves the same problem, the change of the game makes it far less complex. This is good however as it will focus more on the marriage between the hardware and the software instead of the complex design of the game. There are many other issues that this solves with the current project. The major problem this solves is the progress tracking and reward function. Due to the complexity of the maze problem, the progress tracking problem is harder to solve than anticipated and if we take an arena with a strict boundary and goal, it will be far easier to construct a discrete reward structure that can be sent to the learning agent. Basically +10 if they push the opponent outside the arena and maybe a time

multiplier to do it faster. As you can see, the problem becomes simplified and more time can be spent to improve the algorithm and perform analysis.

A bigger problem solved by this is the amount of materials needed and the overall cost of it. Since the materials are pretty simple and can be easily obtained with it being more modular, it will make a better project in the current shipping constraints. This is still to be investigated however, I think a good approach would be to also make an OpenAI Gym environment for this project and test to see how the algorithm fares.

Additional Influence and Inspiration

Figure 41: Texas Instruments Robotics Systems Learning Kit

Inspirations for this project stem from prior experience that we all have had in relation to our previous projects coming from a University of Central Florida student background. One of the introductory courses to the program provided included a study into the basic movement and competitive design aspect of making a simple robot complete maze navigation using a simple algorithm that was for the most part designed on our behalf and constructed with careful directions given to a relatively oblivious student base. Being able to construct a similar project, replacing the simple algorithm with a much more sophisticated artificial intelligence based algorithm and being able to implement personal design experience is not only a liberating way to prove gained knowledge over a four year span, but a comparable parallel and example for the beginnings and uncertainties that a younger group of students may currently face when deciding to enter this program. Having the ability to now create our own instructions and compete in a new sense against our own design is inspiring and in its own way justification for placing additional stress that is greater than average expectations. By testing the limits of how complex a smaller group of students is able to complete in a short time span is a testament to the expectations that a university of this caliber comes to instil in its students.

Additionally, other influences for this project come from our group's student participation in outside extracurriculars such as professor guided research and self study into the general fields of reinforcement learning and robot construction. By having these

prior backgrounds, the task of designing has been expedited and the additional pitfalls that would pass over new designer's heads can be considered before the final design and construction is being made. Specifically relating to Battle-Bots, the designs considered such as the shopping cart movement and design prototype comes directly from prior design experimentation that was completed for one student's participation in the ASME battle bots competition. This prior experience along with personal experience with arduino style components and programming contribute to the reduced amount of time required to learn about and utilize these components in this wider and less restricted design course. Another student's professor-guided research has also inspired the direction taken for reinforcement learning, and being able to reference and represent a more modern and complicated field of study in this project is exhilarating, giving an opportunity to show applications of what is typically thought to be abstract study.

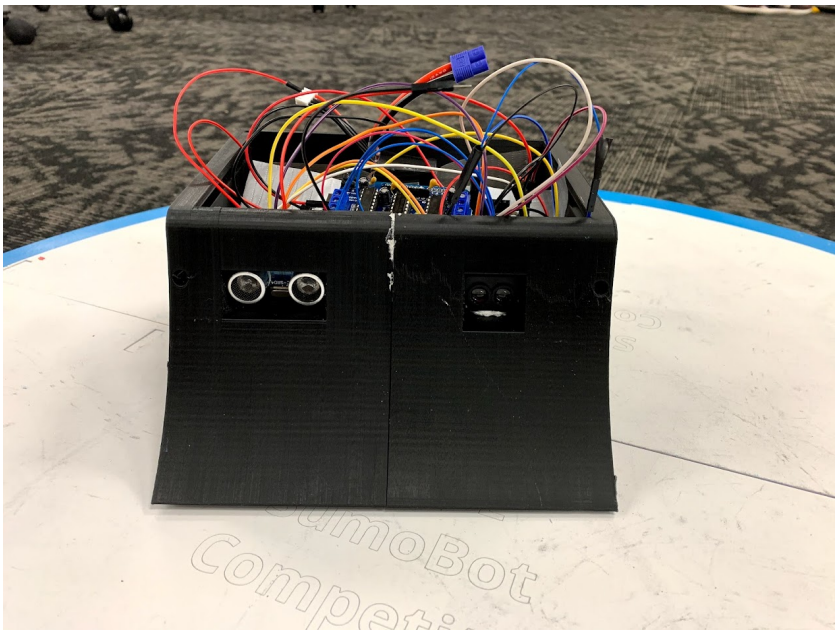


Figure 42: Protograph Taken from UCF ASME Sumo-Bots competition

The above design seen in Figure 31 is a robot constructed missing it's enclosing roof component that was 3D printed and designed by a team one of our members who was also a constituent of for the ASME battle bots competition. This design utilizes a two active and two support wheel system that is considered in the chassi considerations section included previously in this paper under the name of shopping-cart. The benefits of having constructed the above robot (albeit one not having a printed circuit board) include the knowledge of using components such as the HC-SR04 Ultrasonic Sensor (also pictured above) with knowledge relating to calibration and MCU centric robot design. This chassi was also designed to receive impacts within a certain force rating and function within set environmental restrictions, similar to the bots expected to function in our constructed arena.

Stepping into Reality

Being able to translate abstract concepts and emerging technologies into tangible creations and physical representations is very complex, and the goal of this project is to be able to make this happen. Placing large amounts of responsibility and ambitious design concepts into our own hands is what we believe takes researched knowledge to the next level, and is a step towards additional research, or the path into monetizing or utilizing what has already been researched. Taking a different approach to research and forcing uncomfortable decisions in the name of progress is what we believe will make us grow as future designers, engineers, and innovators. By suffering slightly now we have the opportunity to reduce suffering later for those who may be able to use whatever technology may stem from our sacrifices and ideas.

In the same vein, however, we understand that our project is a few steps away from being unrealistic in its current state. The timeline of the project and the requirements that exist are complicated and expedited in a way that may prevent deep exploration with documentation and communication. Because of this and our desire to innovate, project components are flexible and may change depending on the desires and goals that our members wish to accomplish. This potential alternative, however, is a backup plan and is part of understanding the limitations of time constrained research and production. Being held to tight standards is what makes projects fascinating and robust, and having the freedom to change things at will leads to potential for new approaches and designs to be considered as well. Our willingness to take what comes and understanding the need for multiple plans to exist is what will ensure our success in this research based project and further onto the final construction of the robot system arena.

Possible Future Exploration and Research

With the gained understanding from designing and constructing this project, as a group we are convinced that accomplishments such as autonomous robots that can survey a room and retrieve items or interact with the environment to adjust to new situations is fully viable and designable with enough funding. Related tasks include retrieving living beings from a building on fire, or in a lighter sense getting a soda from a fridge in a way that does not knock anything over. By being able to parametrize humanely desired metrics into numbers and things that robots can understand is a huge step into getting machines to accomplish what humans want them to, and to further the quality of life that innovators hope to accomplish. Restraints with current models, and therefore topics to explore, is the sheer amount of data and “experience” that robots may need to undergo specific to the experiences required before success. This means that, under current models, a robot may need to experience hundreds of rescue operations under heavy fire and collapsing buildings, many of which may not succeed, before a life is saved. Because this is not optimal, future research includes ways to train specific robots that emulate real environments, and while this topic is explored lightly in this paper, true research and implementation of this goal is out of our budget and time constraints. Nonetheless, we hope our small steps encourage larger steps from other wealthier parties in the future.

Project Part 1 Pre Construction Conclusion

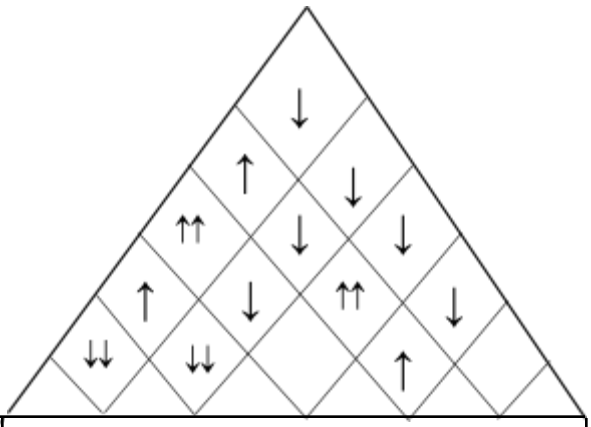
This semester, much like most of this undergraduate career, has been spent in the pandemic and this has resulted in many consequences for the overall state of the project. With the current ongoing supply shortage, there was an increase in prices of materials in order to make deadlines. A traditional Raspberry Pi 4b is around $75 but we ended up paying around $180, more than double because that was the only option that would arrive in time. There is now a concern with other parts as well since we are not buying in bulk, many items we wished for are on backorder and this may cause future delays. In addition to the supply shortage, there is also a significant lack of collaboration that is able to happen with this project due to the pandemic. There are definite health concerns around meeting in person and the project did not move forward as was expected.

However, a lot of good has also come for this project. At this point in time, ignoring supply issues, the project is still very possible in theory and though the scalability won't be proven in this project, it seems that it will be. The raspberry pi 4 is a very capable microcontroller that is both easy to use and quite powerful. Additionally, the implementation of a model-free reinforcement learning algorithm has been done before and is pretty easy to implement. One of the group members, Yash Gharat, has already implemented something similar before and this makes this a lot easier. I think the biggest complication that has been forming is the point of communication between the Arduino-driven dummy agents and the Raspberry Pi-driven commander learning agent. With the web-server pub-sub model, there is an issue of communication lag as each needs to make a request to the server. The server needs to interpret these requests and then store them accordingly and pass them to the appropriate agents in a timely fashion. The current solution to this is to make a router-type web-server that hosts a LAN instead. This will limit communication overhead and will hopefully improve transmission and interpreting times. Another option is to go the bluetooth route and scale down the project such that it is small enough to fit on a table. This will make bluetooth faster and will limit the communication range issue that tends to come with bluetooth while cutting out the middleman and computation. The definite downside to this method that the web-server method tried to solve was the load on the raspberry pi commander agent. Since the dummy agents are meant to not do any computation at all, they will simply send out information but this can result in a single point of failure with the commander agent. This was already a known problem in the project and there does not seem to be a way around this but it can be less-likely if the commander has some help to do functions unrelated to the actual machine learning.

Again, to reiterate, there is always the option of making the commander agent an actual desktop PC instead of a raspberry pi. Since it is a static component in the project, there is no push to make it in a small form factor. However, to lower the overall cost of the project and make it reproducible if other applications arise to use this algorithm, the group members had decided it would be best to use this microcontroller. Additionally, its GPIO may make message passing for progress tracking easier with the custom module that will process the progress of the dummy agents. It would also help with communication to the dummy agents since many parts and third party libraries are already available built specifically for the raspberry pi.

Administrative Content

House of Quality



			Engineering Requirements					
			(1) Memory	(2) Cost	(3) Communications	(4) Dimensions	(5) Sensor Quality	(6) Power Use
			+	-	+	-	+	-
Marketing Requirements	(1) Cost	-	↓↓	↓↓	↓↓	↓	↓	↓
	(2) AI Ability	+	↑↑	↓↓	↑↑	↓	↑↑	↓
	(3) Game	+	↑↑	↓	↑↑	↑	↑	
	(4) Power Eff	+	↑	↓↓	↓			↑
	(5) Integrity	+		↓		↑		
Engineering Requirements Parameters			(1) >3 GB RAM	(2) $300 - $500	(3) <50ft Range	(4) 5x5 ft^2 to Scale		

Project Budget

Table 13: Project Parts Lists

Parts List	
Description	**Price Estimate**
Raspberry Pi Model 4	$35 - $50
Various minor printed circuit board components (communication and power)	$20
Raspberry Pi Compatible Sensors (Ultrasound, Lidar, Photoresistors, Bumpers and Buttons)	$50 - $60
Raspberry Pi Compatible Power Source (Battery casing and motor shield if necessary)	$20 - $30
Wireless Modules	$20 - $60
Storage Modules	$10 - $20
Various Basic Microcontrollers*	$10 - $20 ea.
Various Power Casings and Batteries	$10 - $30
Various LED and circuit response components	$10 - $20
Arena materials (Structural)	$20 - $30
Arena Components (Lights, power supplies, stands, etc.)	$30 - $40
Arena communications (buzzers, lights)*	$20
Estimated Cost	$295 - $505

*Depends on field agent functionality

Project Milestones

Table 14: Senior Design 1-2 Timeline of Milestones

Senior Design 1			
Milestone	**Dates**	**Duration**	**Status**
Project Selection	8/26 - 9/9	2 weeks	Completed
Divide and Conquer	9/9 - 9/16	1 week	Completed
Divide and Conquer 2.0	9/20 - 10/1	2 weeks	Completed
Table of Contents	10/22	3 weeks	Completed
60 Page Draft	10/1-11/5	4 weeks	Completed
100 Page Draft	11/19	6 weeks	Completed
Finalized Document	12/7	9 weeks	In Progress
Order Parts	10/1-1/10	14 weeks	In-Progress
Senior Design 2			
First Prototype	TBD	TBD	Not Started
First Critical Design Review	TBD	TBD	Not Started
Peer-Reviewed Presentation	TBD	TBD	Not Started
Conference Paper	TBD	TBD	Not Started
First Demo	TBD	TBD	Not Started
FInal Demo	TBD	TBD	Not Started
Final Presentation	TBD	TBD	Not Started

Appendix

Sources

Related Papers

S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani, "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond," in IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5476-5497, 1 April1, 2021, doi: 10.1109/JIOT.2020.3030072.

M. N. Ahmadabadi and M. Asadpour, "Expertness based cooperative Q-learning," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 32, no. 1, pp. 66-76, Feb. 2002, doi: 10.1109/3477.979961.

Hu, Junling and Michael P. Wellman. “Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm.” ICML (1998).

S. Yano et al., "A study of maze searching with multiple robots system," MHS'96 Proceedings of the Seventh International Symposium on Micro Machine and Human Science, 1996, pp. 207-211, doi: 10.1109/MHS.1996.563425.
https://ieeexplore.ieee.org/document/563425

Maio, D., & Rizzi, S. (1995). Unsupervised Multi-Agent Exploration of Structured Environments. In ICMAS (pp. 269-275).
https://www.aaai.org/Papers/ICMAS/1995/ICMAS95-036.pdf

Auer, Peter, et al. "The nonstochastic multiarmed bandit problem." SIAM journal on computing 32.1 (2002): 48-77.

Kuleshov, Volodymyr, and Doina Precup. "Algorithms for multi-armed bandit problems." arXiv preprint arXiv:1402.6028 (2014).

Feng, J., M. Huang, L. Zhao, Y. Yang, and X. Zhu. “Reinforcement Learning for Relation Classification From Noisy Data”. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, Apr. 2018, https://ojs.aaai.org/index.php/AAAI/article/view/12063.

Projects

Aquib. (2018, December 21). High Speed Arduino RC Car. Arduino Project Hub. Retrieved from https://create.arduino.cc/projecthub/muhammad-aqib/high-speed-arduino-rc-car-5c2a3d

Tagliabue, J. (2018, October 19). Self-driving (very small) cars - part I. Towards Data Science. Retrieved from https://towardsdatascience.com/self-driving-very-small-cars-part-i-398cca26f930

Drayton, A., Schmitz, W., McGarvey, S., & Vento, C. (n.d.). SPARC. Senior Design Project. Retrieved from https://www.ece.ucf.edu/seniordesign/sp2018su2018/g12/

Q-Learning

https://gym.openai.com/envs/FrozenLake-v0/
https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c
[Q1] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8.3-4 (1992): 279-292.

[Q2] Ms.S.Manju, & Dr.Ms.M.Punithavalli,. (2011). An Analysis of Q-Learning Algorithms with Strategies of Reward Function. International Journal on Computer Science and Engineering. 3.

Language

https://www.zarantech.com/blog/pros-and-cons-of-python-in-machine-learning/

Standards

"IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) , vol., no., pp.1-4379, 26 Feb. 2021, doi: 10.1109/IEEESTD.2021.9363693.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363693&isnumber=9363692

"ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in ISO/IEC/IEEE 29148:2018(E) , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8559686&isnumber=8559685