(Front Page)

Table of Contents:

1. **Project Vision pp. 00**

1.1. Background pp. 00

1.2. Socio-economic Impact, Business Objectives, and Gap Analysis pp. 00

1.3. Security and Ethical Concerns pp. 00

1.4. Glossary of Key Terms pp. 00

1. **Project Planning and Management pp. 00**

2.1. Term Information pp. 00

2.2. Tools and Technology pp. 00

2.3. Best Standards and Practices / Project Plan pp. 00

2.4. Risk Management pp. 00

1. **System Requirement Analysis pp. 00**

3.1. Functional Requirements pp. 00

3.2. Non-functional Requirements pp. 00

3.3. On-Screen Appearance of Landing and Other Pages Requirements pp. 00

3.4. Wireframe Designs pp. 00

1. **Functional Requirements Specification pp. 00**

4.1. Stakeholders pp. 00

4.2. Actors and Goals pp. 00

4.3. Users Stories, Scenarios and Use Cases pp. 00

4.4. System Sequence / Activity Diagrams pp. 00

1. **User Interface Specifications pp. 00**

5.1. Preliminary Design pp. 00

5.2. User Effort Estimation pp. 00

1. **Static Design pp. 00**

6.1. Class Model pp. 00

6.2. System Operation Contracts pp. 00

6.3. Mathematical Model pp. 00

6.4. Entity Relation pp. 00

1. **Dynamic Design pp. 00**

7.1. Sequence Diagrams pp. 00

7.2. Interface Specification pp. 00

7.3. State Diagrams pp. 00

1. **System Architecture and System Design pp. 00**

8.1. Subsystems / Component / Design Pattern Identification pp. 00

8.2. Mapping Subsystems to Hardware (Deployment Diagram) pp. 00

8.3. Persistent Data Storage pp. 00

8.4. Network Protocol pp. 00

8.5. Global Control Flow pp. 00

8.6. Hardware Requirement pp. 00

1. **Algorithms and Data Structures pp. 00**

9.1. Algorithms pp. 00

9.2. Data Structures pp. 00

1. **User Interface Design and Implementation pp. 00**

10.1. User Interface Design pp. 00

10.2. User Interface Implementation pp. 00

1. **Testing pp. 00**

11.1. Unit Test Architecture and Strategy/Framework pp. 00

11.2. Unit Test Definition / Test Data Selection pp. 00

11.3. System Test Specification pp. 00

11.4. Test Reports per Spring pp. 00

1. **References pp. 00**

1. Project Vision

1.1. Background

On the modern battlefield, the ability to deploy troops on the ground in a fast and efficient manner is important to ensuring your side has the tactical advantage. One critical step in deploying troops is using vehicles to deliver troops to the correct destination. Today’s technology allows for the use of computer vision and robotics to control different vehicles. These pilotless systems must be able to drive to the correct location without hitting each other, as well as handle navigation issues that may arise with loss of sight.

1.2. Socio-economic Impact, Business Objectives, and Gap Analysis

Every vehicle needs at least one pilot controlling the vehicle to make sure it is on the right course and nothing goes wrong. What we want to do is reduce the need for pilots in case there is a mission where there are more vehicles than pilots. By programming a vehicle to autonomously follow another vehicle, there will be less need for pilots. The use of a system like this will save a lot of time and money, because there will be less need for soldiers who know how to operate specific vehicles and training for soldiers without that knowledge. With a system of one leader vehicle and one follower vehicle, there would only need to be a pilot of the leader vehicle, cutting the need for pilots in half. Since the autonomous vehicles can also follow each other, so long as the front-most follower vehicle is following the leader vehicle, the amount of pilots can be specified to whatever is needed for the mission. This can be useful if there is a shortage of pilots. Additionally, the risks of pilot error are significantly reduced because the autonomous vehicles will follow their programming. There will be protocols for just about every scenario, so the soldiers inside the follower vehicles will only need to have minimal knowledge of the vehicle. Therefore, a leader-follower vehicle system will cut the cost of pilots and pilot training as intended.

1.3. Security and Ethical Concerns

Since the autonomous vehicles will be self-piloted, it is important that the programming does not get compromised. If a hacker is able to get control of a vehicle, then that puts everyone inside the vehicle at risk. That is why it’s important to make sure the programming is protected by encryption to add layers of security so that the vehicles are protected against cyber attacks. It is also important that there is a method of decryption so that a soldier may debug or repair the vehicle’s programming if there is something wrong with the code.

1.4. Glossary of Key Terms

Pilot - The individual who utilizes the App Controller to navigate the Leader Vehicle.

App Controller - The client-side application the Pilot uses to control the Leader Vehicle.

Leader Vehicle - The robot that the Pilot is directly controlling with the App Controller.

Follower Vehicle - The robot that autonomously follows the Leader Vehicle using its Camera.

Camera - The sensor attached to a Follower Vehicle that allows it to autonomously follow a Leader Vehicle by sensing the Leader’s location.

Leading Object - The object on the Leader Vehicle that the Follower Vehicle’s Camera senses to follow using its orientation.

Obstruction Object - Any object that hinders the path of a Follower Vehicle.

Learning Module - An autonomous module that the Follower Vehicle uses to learn where to follow the Leader Vehicle.

Obstruction Object - Any object that hinders the path of a Follower Vehicle.

Line of Sight - The state when a Follower Vehicle directly views the Leader Vehicle.

Navigational Log - The data the Lead Vehicle and Follower record to determine relative position.

2. Project Execution and Planning

2.1. Term Information

Our team will build a robotic vehicle system consisting of a leader vehicle and one or more follower vehicles. There must be at least two vehicles in this system (one to lead and one to follow). These vehicles must not collide with each other. The project should also consider incorporating a system that handles the follower vehicle losing line of sight or maneuvering around an obstruction object, the ability to make any vehicle the leader vehicle, and communication of navigational data between the robots.

2.2. Tools and Technology

The robot that our team is using as a vehicle is Sunfounder’s PiCar-V, which is essentially a robot hardware kit made to be built around a Raspberry Pi. The Raspberry Pi is used to control the hardware to move the robot and control a camera. Once the kit is built, there will be four wheels, three servos, two motors, and a camera all attached to the Raspberry Pi. The motors will control the motion of the wheels. One servo will control the turning direction of the wheels and the other two servos will control the position of the camera. The camera connects to the Raspberry Pi via USB to directly provide its video feed. The PiCar-V has its own implementation for controlling the robot but also allows for open source coding, which is how we will implement our system around the PiCar-V kit. Our system will consist of two PiCar-Vs, one as a leader and one as a follower, so we will have two kits and two Raspberry Pis to work with. We will also have a desktop application, with remote controller support, that the pilot will use to control the robots and our own router to make sure the communication between the robots and the application is on a closed network. The Raspberry Pis will run Python code executed in Linux OS, specifically Raspian. The desktop application will run Visual C# code on Windows OS. The desktop client will communicate with the robot servers through a gRPC protocol. Additionally, we will use OpenCV for the camera on the follower vehicle to follow the leading vehicle by reading the orientation of its leading object, which we will use QR codes for.

2.3. Best Standards and Practices / Project Plan

We decided to use the Scrum approach for managing our project. This will allow us to plan our progress through Sprints and manage our progress through team meetings. Each Sprint was roughly 2 weeks, where our team focused to complete specific goals by the end of the Sprint. To split the work as evenly as possible, we decided to split into team roles as such:

Alex Alwardt - Network & Video Streaming Developer

Anton Cataldo - Documentation / Image Recognition Developer

Scott Dudley - Documentation / Vehicle Systems & Network Developer

Christian Nickolaou - Project Lead / Vehicle Systems & Image Recognition Developer

Eric Ramocki - Desktop Application Developer

Sean Ramocki - Desktop Application Developer

We decided for the full team to meet twice a week to discuss progress for individual team members to achieve the team goals. These meetings also allowed the team to work together to bridge the connection between the progress of different roles. Our full Sprint schedule is provided below:

Sprint 1: 9/6/2018 - 9/13/2018

* Project Plan
* Requirements
* Tool Chain Setup
* Project Scope Capture

Sprint 2: 9/13/2018 - 9/25/2018

* Basic Diagrams
* User Stories / Detailed Use Cases
* Hardware Research
* Start Network Messaging Protocol

Sprint 3: 9/25/2018 - 10/11/2018

* Desktop Application UI Skeleton
* Receive/Build Vehicles
* Finish Network Server
* Finish Network Client
* Registering the Vehicles

Sprint 4: 10/11/2018 - 10/23/2018

* Configuration of Motors/Servos
* Making Application Asynchronous
* Full Movement of Leader Vehicle
* Start Video Streaming
* Start Image Recognition

Sprint 5: 10/23/2018 - 11/6/2018

* Finish Video Streaming on Vehicle
* Finish Video Streaming on Application
* Direct IP Connection in Application
* Finish Desktop Application Implementation
* Finish Installing/Integrating Dependencies
* Image Recognition for Follower Vehicle

Sprint 6: 11/6/2018 - 11/27/2018

* Image Recognition between Leader and Follower Vehicles
* Collision Detection for Follower Vehicle
* Recording/Setting Distance Between Vehicles
* Camera Debugging
* Finalize Diagrams
* Finish Documentation

Sprint 7: 11/27/2018 - 12/4/2018

* Testing / Debugging
* Dry Run

Sprint 8: 12/4/2018 - 12/11/2018

* Testing / Debugging
* Final Competition

2.4. Risk Management

Discord

When progress was halted for a certain aspect of the project, we scheduled additional meetings the necessary roles to fix the problems

3. System Requirement Analysis

**Requirements:**

3.1. Functional Requirements

1. The application shall allow the leader vehicle to move forward by remote control.
2. The application shall allow the leader vehicle to move backwards by remote control.
3. The application shall allow the leader vehicle to turn left by remote control.
4. The application shall allow the leader vehicle to turn right by remote control.
5. The application shall be able to transmit position changes to the follower vehicle.
6. The application shall allow the follower vehicle to interpret position changes into left position motion.
7. The application shall allow the follower vehicle to interpret position changes into right position motion.
8. The application shall allow the follower vehicle to interpret position changes into forward position motion.
9. The application shall allow the follower vehicle to interpret position changes into backwards position motion.
10. The application shall allow the follower vehicle to follow directly behind the leader vehicle.
11. The application shall allow a follower vehicle to follow directly behind another follower vehicle, as long as there is a leader vehicle in the front of the vehicle train.
12. The application shall prevent the follower from colliding with the leader vehicle.
13. The application shall stop motion on all out of range remote controlled vehicles.
14. The system shall only support one wireless controller.
15. The system shall only support one concurrent pilot at a time.
16. The application shall be able to log diagnostic information about each drive.
17. The application shall be able to export diagnostic information about each drive.
18. The application shall be able to playback previous diagnostic information from earlier drives.
19. The application shall be able to specify the wireless band of the remote controller.
20. The application shall allow the remote control to specify the channel of the remote control.
21. The application shall allow the remote control to control the forward speed of the leader vehicle.
22. The application shall allow the remote control to control the forward acceleration of the leader vehicle.
23. The application shall allow the remote control to control the backwards speed of the leader vehicle.
24. The application shall allow the remote control to control the backwards acceleration of the leader vehicle.
25. The application shall display a warning if the leader vehicle is unable to change its current position.
26. The application shall display a warning if the follower vehicle is unable to change its current position.
27. The application shall be able to display the current position of the leader vehicle upon being stuck.
28. The system shall be compatible with all major operating systems.
29. The application shall encrypt the positional changes of the leader vehicle.
30. The application shall encrypt the positional changes of the follower vehicle.
31. The application shall be able to decrypt the information transmitted from the vehicles.
32. The leader vehicle shall encrypt the signal sent to the follower vehicle(s).
33. The follower vehicles shall be able to decrypt the signal sent from the leader vehicle.
34. The application shall allow the leader vehicle to receive input from a remote control.



1. The application shall be able to look for the leader vehicle upon loss of sight.
2. The application shall be able to designate any vehicle in the system as the leader vehicle.
3. The application shall be able to designate any vehicle in the system as the follower vehicle.
4. The application shall be able to use image pattern recognition to determine the location of the lead vehicle.
5. The application shall be able to navigate around obstructions in the following vehicles path.
6. The application shall be able to determine the distance between the follower and leader vehicle.
7. The application shall be able to determine the angular position between the follower and leader vehicle.
8. The application shall only have one leader vehicle at a time.
9. The application shall allow one or more following vehicles.
10. The application shall be able to determine orientation of following vehicle relative to lead vehicle.
11. The application should be able to register vehicles in the system.
12. The application shall notify the lead vehicle that the following vehicle(s) have lost sight.

3.2. Non-functional Requirements

1. The application shall prevent the follower vehicle from being within (DISTANCE) around the vehicle.
2. All users actions shall not have a response time greater than 250ms.
3. The system shall have zero severity level 1 defects.
4. Meantime between failures shall be at least 30 days.
5. The system shall support a remote control range of 20 meters.

3.3. On-Screen Appearance of Landing and Other Pages Requirements

3.4. Wireframe Designs

4. Functional Requirements Specification

4.1. Stakeholders

4.2. Actors and Goals

Pilot

App Controller

Leader Vehicle

Follower Vehicle

Camera

Leading Object

4.3. Users Stories, Scenarios and Use Cases

**User Stories:**

1. As a Pilot, I can register the Leader Vehicle.
2. As a Pilot, I can register the Follower Vehicle(s).
3. As the Leader Vehicle, I can lead the Follower Vehicle(s).
4. As a Follower Vehicle, I can follow the Leader Vehicle.
5. As the Remote Controlling Device, I can control the movement of the Leader Vehicle.
6. As a Pilot, I can use the Remote Controlling Device.
7. As a Follower Vehicle, I may lose Line of Sight to the Leader Vehicle.
8. As a Follower Vehicle, I can recognize an Obstruction Object.
9. As a Follower Vehicle, I can navigate around obstructions.
10. As a Follower Vehicle, I can notify the Leader Vehicle if they are no longer in Line of Sight.
11. As a Follower Vehicle, I can notify the pilot when there is loss of line-of-sight.
12. As a Follower Vehicle, I can reestablish Line of Sight with the Leader Vehicle.
13. As a Follower Vehicle, I can notify the Pilot when Line of Sight is regained.
14. As a Follower Vehicle, I can respond when the Leader Vehicle moves forward.
15. As a Follower Vehicle, I can respond when the Leader Vehicle moves backward.
16. As a Follower Vehicle, I can respond when the Leader Vehicle turns left.
17. As a Follower Vehicle, I can respond when the Leader Vehicle turns right.
18. As a Follower Vehicle, I can respond when the Leader Vehicle stops.
19. As the Leader Vehicle, I may lose connection with the Remote Controlling Device.
20. As the Leader Vehicle, I can notify the Pilot if I lose connection with the Remote Controlling Device.
21. As a Leader Vehicle, I keep track of the Home Location.
22. As a Leader Vehicle, I can automatically navigate to the Home Location.
23. As a Follower vehicle, I can keep track of the Home Location.
24. As a Follower vehicle, I can automatically navigate to the Home Location.
25. Jeff, a Pilot controls the Lead Vehicle to sharply turn down a hallway that causes the Follower Vehicle to lose Line of Sight with the Lead Vehicle. The Lead vehicle notifies the pilot that Line of Sight has been broken. The Follow Vehicle enacts its protocol to relocate the Lead Vehicle. When Line of Sight is reestablished, the Lead Vehicle notifies the pilot.
26. Jeff is controlling the Lead Vehicle with the Following Vehicle behind when an obstructions falls between the vehicles. There is no line-of-sight loss, however, the following vehicle enacts a protocol that allows for the vehicle to search for a way around the obstruction and continue to follow the lead vehicle.
27. Jeff is piloting the lead vehicle to a destination, but remembers that he forgot something at the starting point. He presses a button that automatically returns the lead vehicle and any following vehicles back to a home position.
28. Sally is a soldier on the battlefield and she’s currently pinned down and unable to find a way to safety. She knows there’s a trap outside her position and has no way to avoid it without triggering it herself. She setups her remote controlled rc car and uses it to safely trigger the trap, allowing her to escape in one piece.
29. Jeff wants to pilot the Lead Vehicle. Before he begins, he sets up the Lead Vehicle and Follower Vehicles one behind the other. He then enacts the Register Vehicle protocol to make sure all the Vehicles are connected and the Vehicle that is the Lead properly registers as the Lead, and the Followers are all properly registered to follow.
30. Jeff wants to control the Lead Vehicle and have a Follower Vehicle trail it. Jeff has already properly registered the vehicles. Jeff takes the Remote Controlling Device and begins to navigate the Lead Vehicle as he wishes. The Follower Vehicle properly trails the Lead Vehicle.

**Detailed Use Cases:**

|  |  |
| --- | --- |
| Name | 1. Register Vehicle |
| Pre/Entry Condition | The Vehicle is on and broadcasting. |
| Trigger | The Pilot clicks “Register Vehicle” button. |
| Post/Exit-condition | A message of success is displayed in a pop-up and the vehicle information is added to the list and registered in the system. |
| Main flow of events -- identify all data elements | --The Pilot clicks the “Register Vehicle” button.  --A window is displayed listing broadcast IDs of vehicles.  --The Pilot selects the vehicle and clicks the “Add” button.  --The Pilot enters vehicle nickname.  --The Pilot chooses whether vehicle is a Leader Vehicle or a Follower Vehicle.  --The Pilot clicks “Accept”. |
| Exceptions and alternate actions | The vehicle nickname is already in use, then the Pilot is advised that they need to choose a different name. |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 2. Pilot Controls the Leader Vehicle |
| Pre/Entry Condition | The Leader Vehicle is registered. |
| Trigger | The Pilot inputs a command into the Remote Controlling Device. |
| Post/Exit-condition | The Leader Vehicle responds to the command. |
| Main flow of events -- identify all data elements | -- The Pilot inputs a command into the Remote Controlling Device.  -- The Remote Controlling Device sends a signal to the Leader Vehicle.  -- The Leader Vehicle responds to the command. |
| Exceptions and alternate actions |  |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 3. Follower Vehicle is Obstructed |
| Pre/Entry Condition | The Follower Vehicle is registered and has line of sight on Leader Vehicle |
| Trigger | The Follower Vehicle detects an Obstruction Object in its path. |
| Post/Exit-condition | The Follower Vehicle has navigated around the Obstruction Object. |
| Main flow of events -- identify all data elements | -- The Follower Vehicle is properly following behind the Leader Vehicle.  -- The Follower Vehicle identifies an Obstruction Object in its designated path.  -- The Follower Vehicle navigates around the Obstruction Object and continues on its path. |
| Exceptions and alternate actions | If the Leader Vehicle is out of the Line of Sight, the Follower Vehicle shall try to regain the Line of Sight with the Leader Vehicle. |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 4. Follower Vehicle follows Leader |
| Pre/Entry Condition | The Leader Vehicle and the Follower Vehicles are both registered. |
| Trigger | The Leader Vehicle moves away from the Follower Vehicle. |
| Post/Exit-condition | The Follower Vehicle is moving towards the Leader Vehicle. |
| Main flow of events -- identify all data elements | -- The Follower Vehicle detects that the Leader Vehicle is moving away from it.  -- The Follower Vehicle determines the direction that it must move in order to move towards the Leader Vehicle.  -- The Follower Vehicle moves towards the Leader Vehicle. |
| Exceptions and alternate actions |  |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 5. Follower Vehicle Loses Line of Sight |
| Pre/Entry Condition | The Follower Vehicle is following the Leader Vehicle. |
| Trigger | An Obstruction Object appears and the Follower Vehicle loses the Line of Sight of the Leader Vehicle. |
| Post/Exit-condition | The Follower Vehicle relocates the Leader Vehicle. |
| Main flow of events -- identify all data elements | -- The Follower Vehicle stops in place.  -- The Follower Vehicle shall try to regain line of sight with the Leader Vehicle.  -- The Follower Vehicle regains the Line of Sight with the Leader Vehicle. |
| Exceptions and alternate actions |  |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 6. Collision Prevention |
| Pre/Entry Condition | The Follower Vehicle approaches a certain proximity to the Leader Vehicle. |
| Trigger | The Follower Vehicle becomes too close to the Leader Vehicle. |
| Post/Exit-condition | The Follower Vehicle is no longer within a certain proximity to the Leader Vehicle. |
| Main flow of events -- identify all data elements | -- The Follower Vehicle becomes too close to the Leader Vehicle.  -- The Follower Vehicle stops in place.  -- The Follower Vehicle waits in place for the Leader Vehicle to move away.  -- The Follower Vehicle is no longer within a certain proximity to the Leader Vehicle. |
| Exceptions and alternate actions | -- The Follower Vehicle reverses if the Leader Vehicle advances towards the Follower Vehicle. |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 7. Connection Lost |
| Pre/Entry Condition | The Application is connected with the Leader Vehicle. |
| Trigger | The Application loses connection with the Leader Vehicle. |
| Post/Exit-condition | A Notification is sent to the Pilot that the connection was lost. |
| Main flow of events -- identify all data elements | -- The Application loses connection with the Leader Vehicle.  -- The Leader Vehicle stops in place.  -- A Notification is sent to the Pilot that connection to the Leader Vehicle is lost. |
| Exceptions and alternate actions |  |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 8. Search Mode |
| Pre/Entry Condition | The Application has lost connection to a Leader Vehicle or a Follower Vehicle. |
| Trigger | Search mode is triggered. |
| Post/Exit-condition | The Vehicle is reconnected. |
| Main flow of events -- identify all data elements | -- The Application has lost connection to a Leader Vehicle or a Follower Vehicle.  -- The Vehicle regains signal connection.  -- A Notification is sent to the Pilot. |
| Exceptions and alternate actions | -- The Application is unable to locate any vehicles. |
| Special requirements -- if any |  |

|  |  |
| --- | --- |
| Name | 9. Return Home |
| Pre/Entry Condition | A Follower Vehicle is connected. |
| Trigger | A Return Home command is initiated. |
| Post/Exit-condition | The Follower Vehicle returns to its initial location known as “Home”. |
| Main flow of events -- identify all data elements | -- A Return Home command is initiated.  -- The Follower Vehicle retraces its previous steps and returns to the point of origin.  -- The Follower Vehicle returns to its initial location known as “Home”.  -- The Pilot is notified that the Follower Vehicle has returned home. |
| Exceptions and alternate actions |  |
| Special requirements -- if any |  |

4.4. System Sequence / Activity Diagrams

5. User Interface Specifications

5.1. Preliminary Design

5.2. User Effort Estimation

6. Static Design

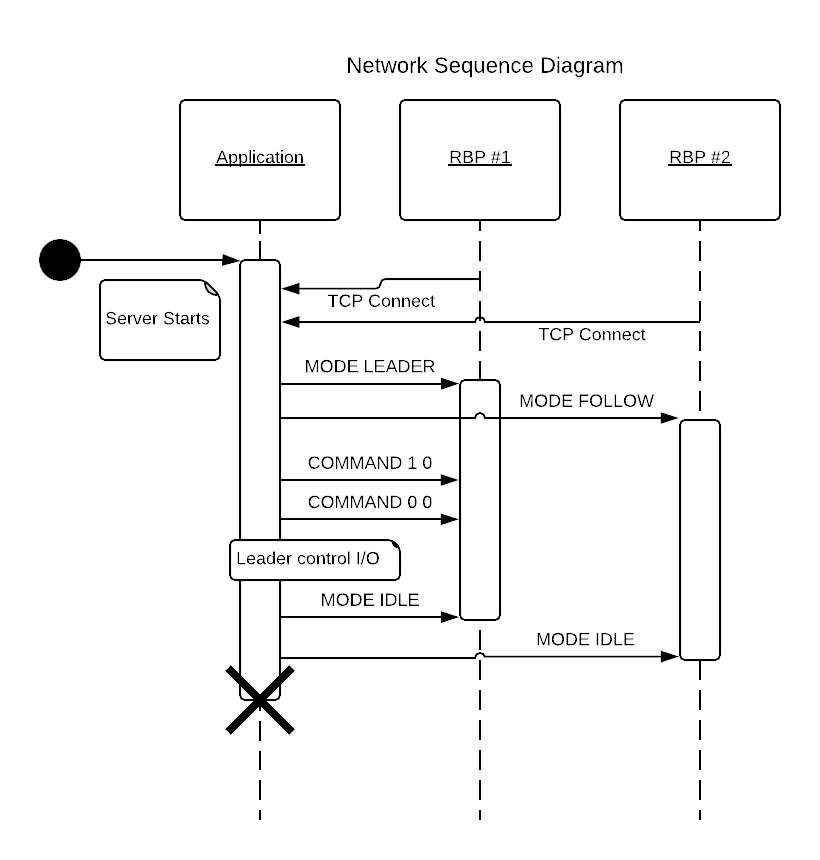
6.1. Class Model

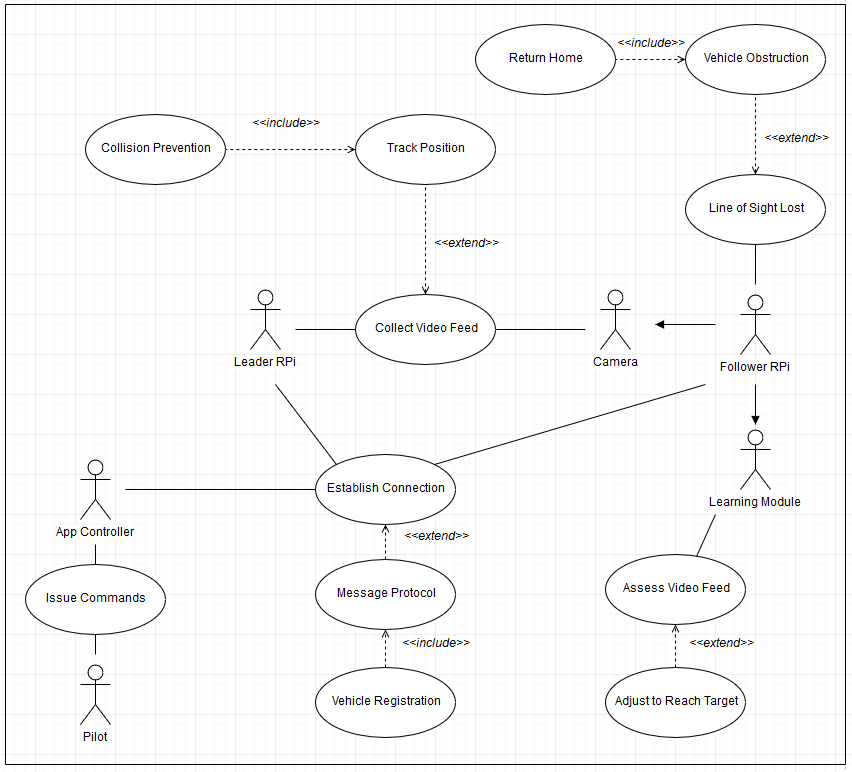
6.2. System Operation Contracts

6.3. Mathematical Model

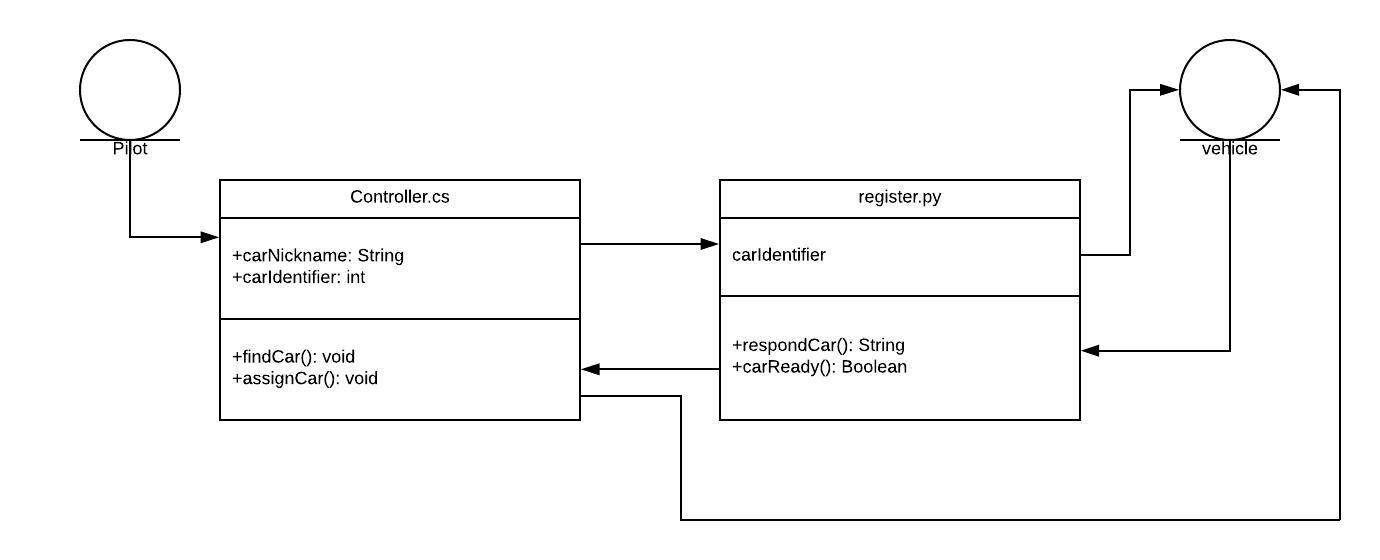
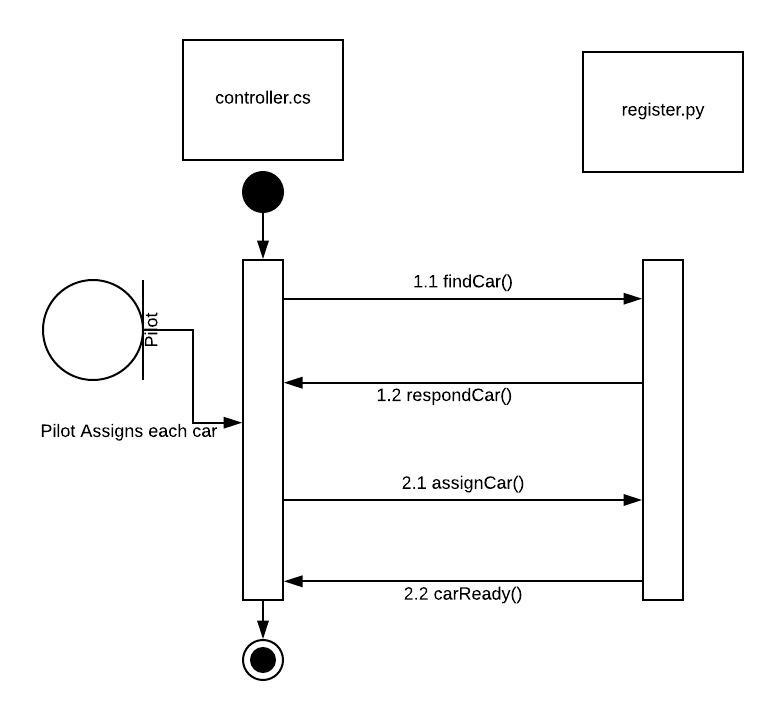
6.4. Entity Relation

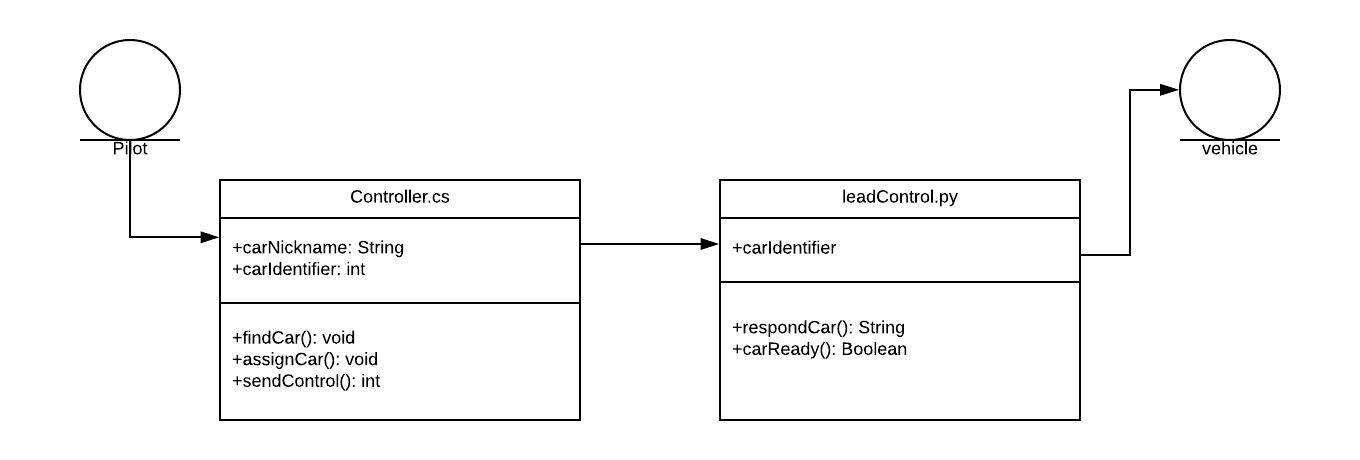
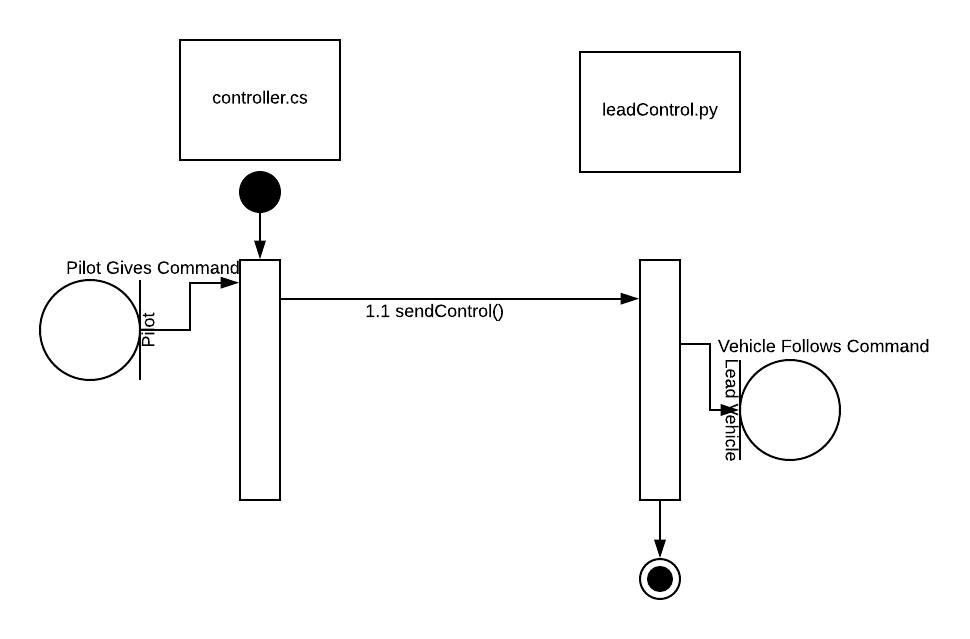
7. Dynamic Design

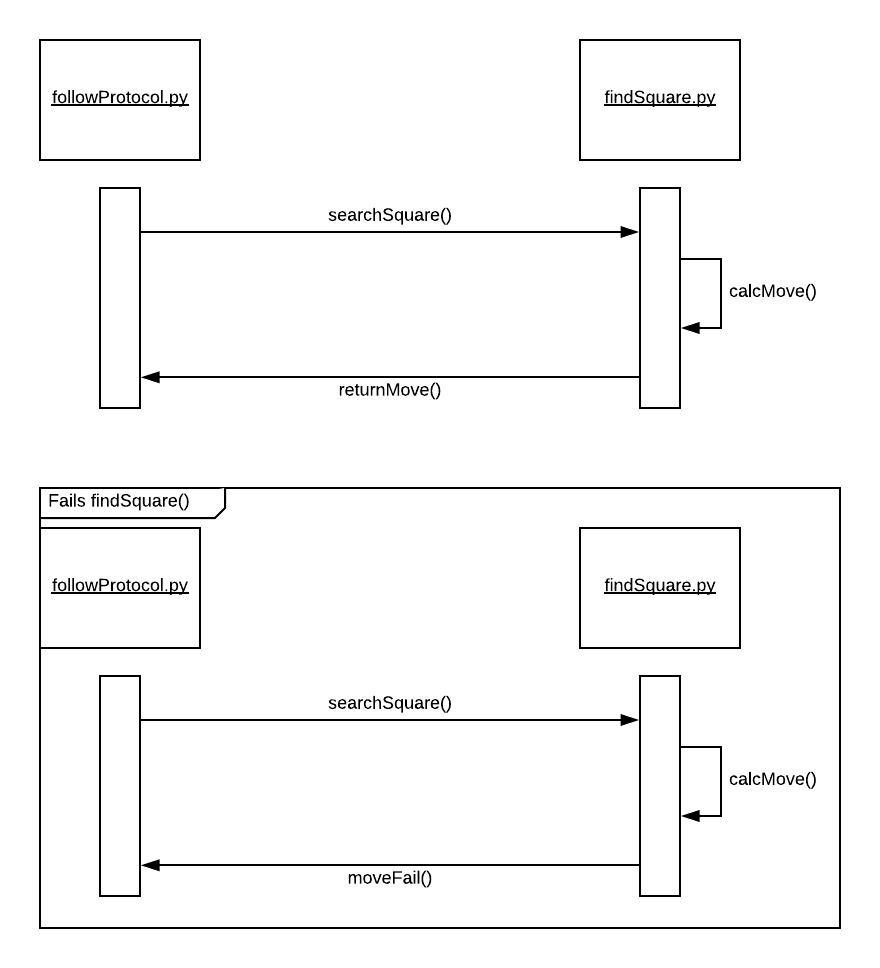
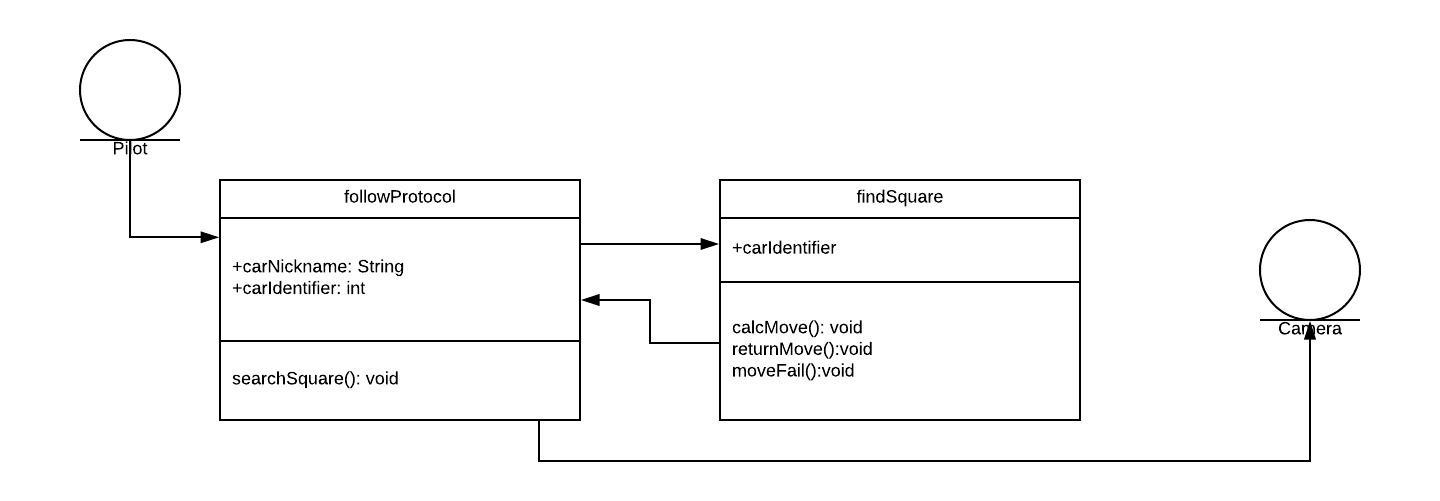


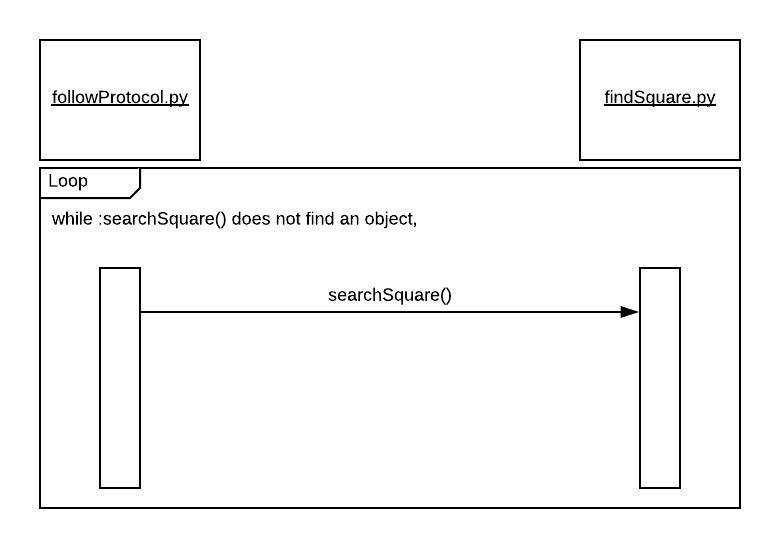
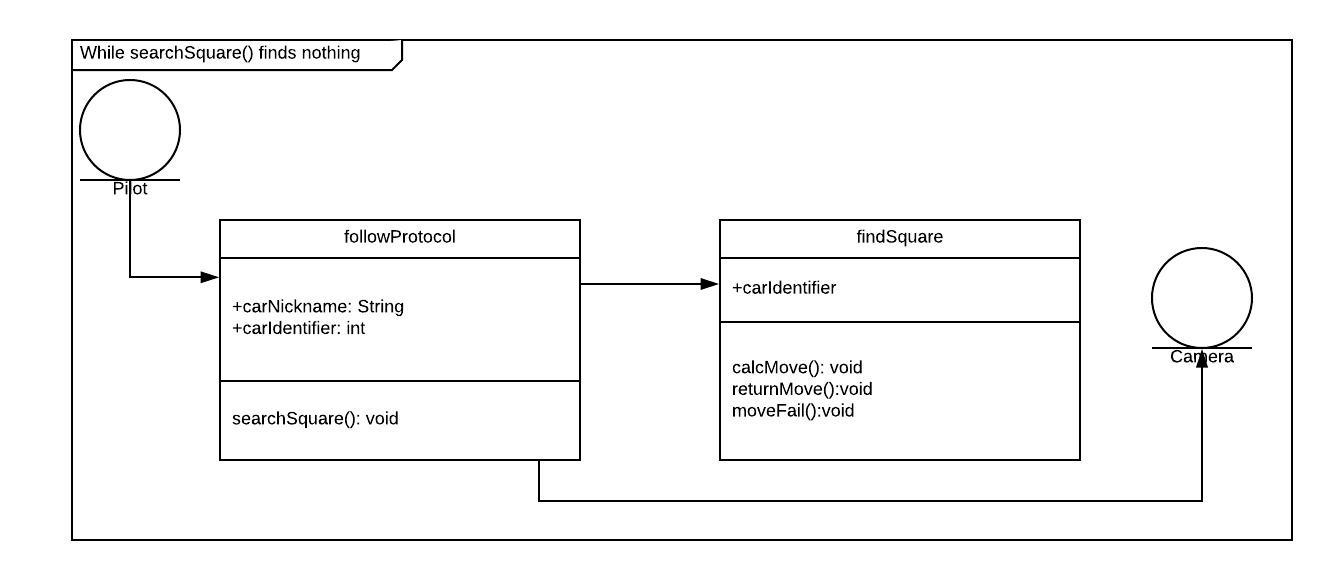


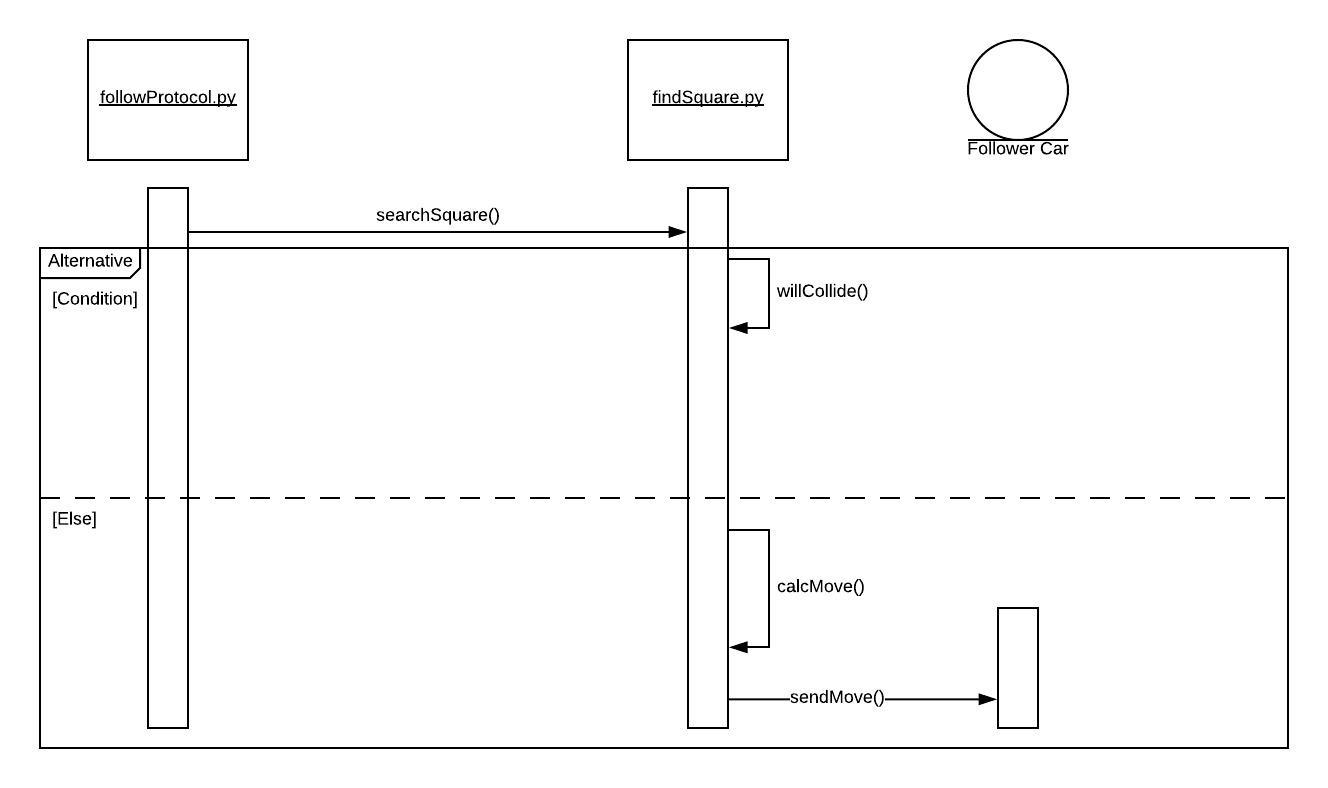
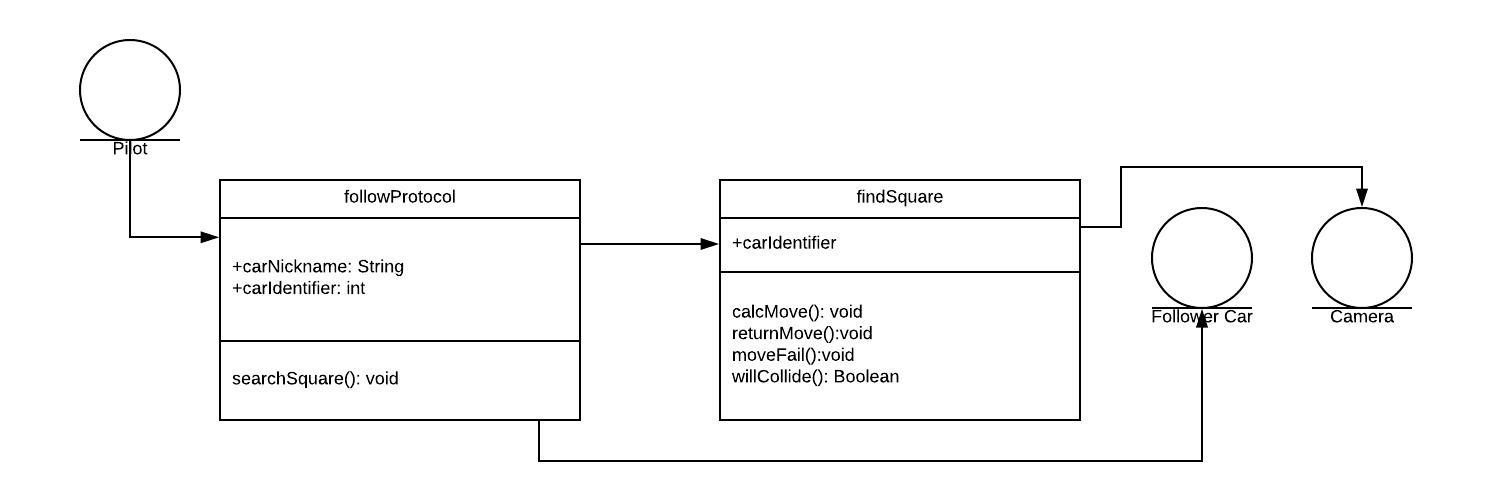
1. Registration



2 Leader Control.

3. Follow the Leader  
  


4. Line of sight Failure   
  


5. Collision Prevention  
  


7.1. Sequence Diagrams

7.2. Interface Specification

7.3. State Diagrams

8. System Architecture and System Design

8.1. Subsystems / Component / Design Pattern Identification

8.2. Mapping Subsystems to Hardware (Deployment Diagram)

8.3. Persistent Data Storage

8.4. Network Protocol

GRPC

8.5. Global Control Flow

8.6. Hardware Requirement

Raspberry Pi 3 + components / PiCar-V Kit

9. Algorithms and Data Structures

9.1. Algorithms

9.2. Data Structures

10. User Interface Design and Implementation

10.1. User Interface Design

10.2. User Interface Implementation

11. Testing

11.1. Unit Test Architecture and Strategy/Framework

11.2. Unit Test Definition / Test Data Selection

11.3. System Test Specification

11.4. Test Reports per Spring

12. References