Drone Simulator

PROJECT PLAN

Team sdmay19-35

Dr. Ali Jannesari, Client & Adviser

Bansho Fukuo, Test Engineer & Back-End Developer
Ian Gottshall, Scrum Master & Front-End Developer
Jawad M Rahman, Meeting Manager & Back-End Developer
Jianyi Li, Test Engineer & Back-End Developer
Sammy Sherman, Report Manager & Front-End Developer
Mehul Shinde, Team Lead & Back-End Developer

Email: sdmay19-35@iastate.edu
Website: https://sdmay19-35.sd.ece.iastate.edu

Revised: 9/28/2018 (Version 1)

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

ROS: Robot Operating System

1 Introductory Material

1.1 Acknowledgement

Team 35's client: Dr. Ali Jannesari

Team 35's advisor: Dr. Ali Jannesari

1.2 Problem Statement

The client currently has a drone which uses Nvidia GPU and a camera. The drone has a hotspot built into it and can be connected remotely via command prompt interfaces such as ssh. Our task is to create a web portal system which can visually depict the simulation and controls of the drone.

Consists of two components, each separated and clearly identified:

-General problem statement – a drone simulation and control web-application is needed to operate the drone and eventually autonomously fly it using the simulation data.

-General solution approach – the web application will use ReactJS to run the controls and simulation along with Gazebo and ROS as modules serving as the simulation environments for the project.

-The user wants to have ISU's own open source drone simulator which could be used to simulate as well as control a drone.

1.3 Operating Environment

The operating environment for this project is a web browser front-end, connected to a server back-end running as a desktop app on any operating system.

1.4 Intended Users and Intended Uses

There are multiple uses of this product. It can be used for educational, research and recreational purposes. The simulation will imitate real world flight physics, providing the users with an interactive experience. That being said, the drone will be used in schools, among the students of all age and experience. Students will be able to understand the laws and concepts of physics better by using the simulator.

The product will also be used by researchers, especially those interested in the use of sensors. Recreational users would also use this to understand how a drone works.

1.5 Assumptions and Limitations.

Assumptions

- Hardware and operating system environment are provided through ISU, and those resources will sufficient for the developing the simulation software
- Number of the user access to the simulation server are limited.
- The end user can manipulate the simulator without specific instructions.
- Product will be open-sources.

Limitations

- Server performance limitation - our projects are powered by the GPU, if the provided server machines are insufficient, our team need to find or arrange for different resources.

1.6 Expected End Product and Other Deliverables

A fully operating simulator will meet the users'/ client's needs by providing them the following features:

1. An environment of their choice. For example, the user will be able to select if they want to fly their drone on urban, rural or in a forest environment
2. Fast, robust and engaging
3. It will be accessible on the World Wide Web and will run on any major web browser
4. It will be a cross-platform application, that is, it can be used both form desktop and mobile

Deliverables

1. The client will receive documentation of the code, which will include a report of what each member of the team did and the hours that they have worked - to be delivered by 05/03/2019.
2. The client will also receive a manual which will provide a high level description of what the project does and how the front-end and back-end works and how they interact. The manual will also include component diagrams and flow charts - to be delivered by 05/03/2019.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The task is to create a web application, which is a drone simulation software using Gazebo, which will interact with ROS. After creating the simulation software, the next step will be to control the physical drone with that web application. The end products will be:

– The final or main product of the project is a drone simulator

– The simulator will be customizable, fast and robust and must use ROS
– It will take in commands from the user and make the drone perform those commands on different environments - forest, urban or countryside

– Once these requirements are met, the simulator would be connected to a physical drone and it will perform according to user needs

2.2 Functional Requirements.

The functional and technical requirements of the project are:

- The application should be web based
    - The application will be open source and be accessible from the World Wide Web, so that anyone from anywhere can use this to fit their needs.
- Ability to select an environment - forest, urban, rural etc.
    - The user should be able to select which environment they want to fly their drone in, they will have multiple options.
- Should be open to all platforms
    - This application should be accessible from both mobile and desktop environments. It will run on all the major browsers.
- Multiple sessions should be allowed
    - Multiple users should be able to access the application and simulate in an environment of their choice.
- Must be able to interact with ROS
    - This is one of the most important requirements. The simulator will likely be used for research purposes later on, and the client will conduct their research using ROS. So this application must be controlled by ROS.
- The application should be fast and robust
    - High performance and efficiency is one of the requirements of our client. This is why the front-end will be written using React and the back-end will be written in C/C++.

2.3 Constraints Considerations

The non-functional requirements for the project are as follows:

- Performance: The physics simulation must perform without server-side lag, and the latency between client and server must be less than 1 second at all times.
- Compatibility: The simulator must be compatible with ROS (Robot Operating System).
- Portability: The simulator must work on any operating system.
- Scalability: The simulator server must be able to handle many clients simultaneously.
- Documentation: The project code must be well-documented, and user guides must be sufficient to guide new users.
- Usability: The simulator's user interface must be intuitive and simple to use.
- Robustness: The simulator must crash gracefully upon encountering errors.

We have not yet established standard protocols for our project.

2.4 Previous Work And Literature

Many similar drone simulators exist. We investigated two simulators during the first weeks of the project: AirSim and Gazebo. Both are insufficient for the client's needs in their unmodified state.

AirSim

AirSim is an open-source drone simulator created by Microsoft. It also has support for simulating autonomous vehicles. It uses Unreal Engine for its physics and graphics. For input, it supports drone remote controls, some popular flight controllers, keyboard controls, and programmatic controls.

AirSim has several advantages over other simulators. AirSim's graphics are excellent compared to most others, and the software is distributed under the MIT license, giving us freedom to use and modify it if we choose to. However, Unreal Engine is complex, and using this project would likely mean learning the engine, which would add many hours to our workload and introduce risks in the form of incomplete knowledge of the platform.

Gazebo

Gazebo is a popular open-source robot simulator. It allows the user to set up a virtual physics-simulated environment by dragging and dropping elements into a 3D space. The simulation elements can be controlled either by C++ modules, called plugins, that are compiled alongside Gazebo and loaded in the same runtime; or by interfacing with ROS (Robot Operating System) and receiving ROS commands from the terminal or another program.

In comparison to our planned project, Gazebo is very resource-heavy, requiring an Nvidia GPU to run. Additionally, it is not cross-platform, since it only works in Ubuntu. These shortcomings prevent us from using it without modification. However, it does boast myriad features and customizability options.

Gazebo has an official JavaScript web client called GzWeb. GzWeb is merely a web-based user interface for Gazebo; it still requires a running Gazebo simulation to function, and thus it is not any more portable than Gazebo itself.

2.5 Proposed Design

For the simulation, we have a few choices for design: utilize Gazebo, utilize AirSim, or create our own software from scratch. The simplest solution would be to incorporate as much of Gazebo's pre-existing software, such as its web client GzWeb, and adapt it to our requirements. Another fairly simple solution would be to implement AirSim and create our own plugins to interface between the web client and AirSim simulator. The most difficult solution is to design our own simulator from scratch, taking inspiration from Gazebo and AirSim. Although very challenging, designing our own simulator would provide for the most portable and best-performing solution.

Our front-end will be created using ReactJS and will unify the simulator and drone control in a user-friendly environment. In order to actually control the simulation or drone, the front-end will send requests to the server which will be interpreted and converted into appropriate ROS commands.

2.6 Technology Considerations

Section 2.4 discusses existing code bases that could be modified for our purposes.

2.7 Safety Considerations

Since our team will using the personal computer and drone that client provided. Our team concerned about those hardware would be secure by the during the testing stage. one other concern on the drone that provided should be well simulated before the real testing, to reduce the risk of the mechanical damage to the drone.

2.8 Task Approach

Our project can be broken into 2 main components, each of which may have contain various sub-components. The first main component is the server which will be running the Gazebo simulator, a websocket server, a HTTP server, and ROS. The other main component is a ReactJS based web-client which will display the simulation from the server, as well as the real drone control view, and allow interaction with it. As the client interacts with the simulation or drone, the server will be receiving requests that will be interpreted to ROS commands and passed to Gazebo or the drone controller.

Figure 1. *GzWeb, a web client provided by Gazebo.*

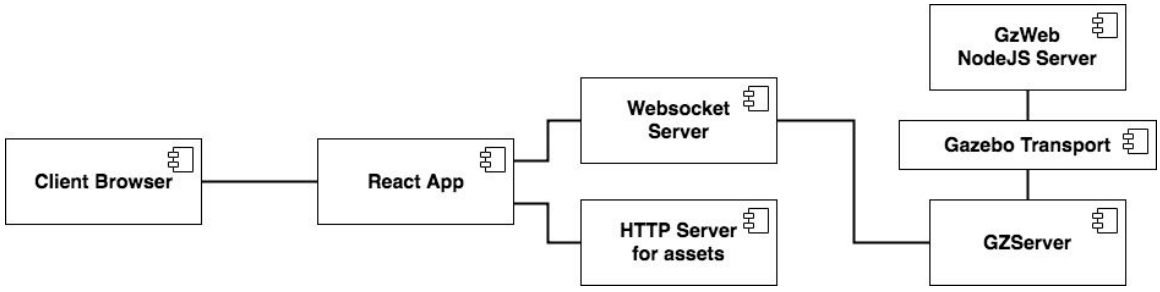
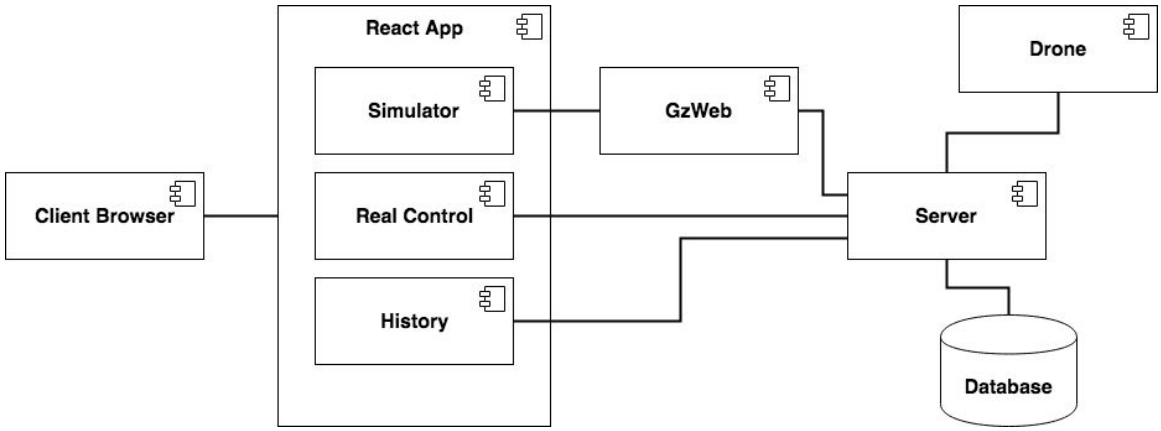


Figure 2. *Overview of the system.*



2.9 Possible Risks And Risk Management

As we proceed, we will likely uncover new, unforeseeable issues. However, some issues that may pose a problem to our current design are: lack of suitable equipment, incompatible software, a general unfamiliarity with the technology being used, or insufficient client-supplied requirements.

In order to mitigate some of these risks, we will work with our client to ensure that we obtain the equipment necessary to deliver their product. However, this may also require that we work directly with the equipment supplier. In an attempt to familiarize ourselves with any and all technologies we require, we will conduct thorough research and test often as we develop. We conduct weekly meetings with our client which we will use to ensure that all requirements are clearly described and understood before attempt to plan our solution.

2.10 Project Proposed Milestones and Evaluation Criteria

The two main milestones are the abilities to simulate a drone from the web-client and control a real drone from the web-client. As the project progresses, more functionality will be added. With the additional functionality, milestones will also be added. Our evaluation

criteria will be: performance, accessibility, portability, and safety. Also, our simulation should be comparable to existing simulators such as Gazebo or AirSim.

2.11 Project Tracking Procedures

Our group will utilize tools such as Gitlab and Trello to keep track of progress. Trello will be used to maintain and organize general, overall project progress. We will use Gitlab to store our code and create issues that refer to specific segments of code. This way we can keep the overall flow separate from our code specific issues and tasks.

2.12 Expected Results and Validation

The result of the project will be a web-based drone simulator and controller. The user will be able to log into their account, view their created environments and models, select/edit/delete environments and models, and utilize them to simulate a drone. The simulation will be quick, smooth, and realistic.

2.13 Test Plan

- Should be open to all platforms
    - Test steps
        i. Using the every device/operating system to access to the server to make sure that all platforms are allow to access.
        ii. Recording the all access to make sure that all platform are able to connect to the server without any errors.
- Multiple sessions should be allowed
    - Test steps
        i. Connecting multiple the devices with server to make sure o starts the sessions to the each devices.
        ii. Making a test, and making sure different devices with independent views with same address server.
- Must be able to interact with ROS
    - Test steps
        i. Using the ROS to interact with the simulation (gazebo) or the drone.
        ii. Using the different situation to testing the performance and the efficiency to make sure that simulator/controller is meet to the requirement

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

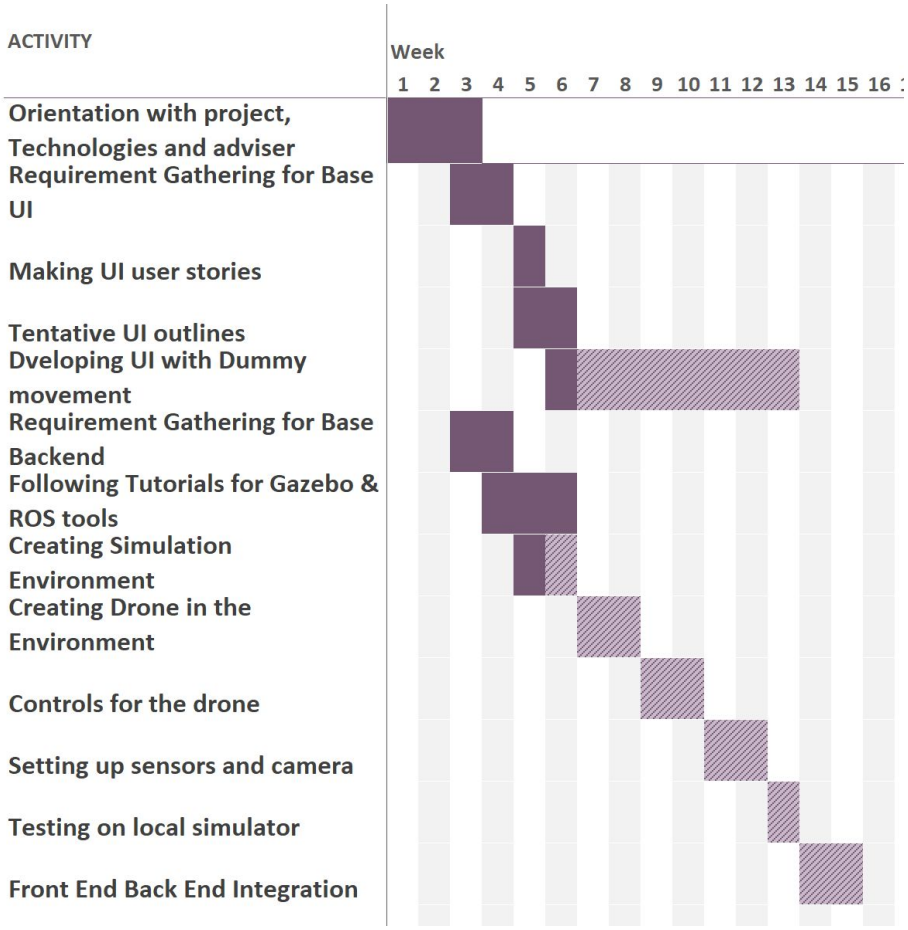
• This project will be worked on by the team, complying with Agile model of development. This would include regular grooming sessions in well defined sprints.

• The grooming session will take place at the beginning of every 2-week-sprint when there will be requirement gathering for the functionality being delivered. Story cards will be generated and each card will be assigned to a team member.

• Each sprint will comprise of two weeks and on Friday of each sprint there will be a sprint-demo in presence of the adviser/client. The sprint-demo will reflect on functionalities ready to ship.

• A detailed schedule in form of a Gantt chart for the project is provided below:

Figure 3. *Detailed project schedule.*



– The schedule laid out in form of the Gantt chart is subjected to changes as and when required.

• Agile process of software development enables a flexibility in the project schedule and since the requirements might change, the schedule is subjected to change. There will however be well defined sprints comprised of two weeks.

The Gantt chart above lays out the schedule for phase-1 of the project which is delivering a web based simulator. The plan for phase-2 i.e. semester two of senior design project is to implement the simulation interface on an actual drone.

The proposed timeline is based on the team's current knowledge of the project. The team plans to collaboratively deliver on the first few sprints and then work individually on well defined story cards groomed in sessions before the start of the sprint.

The team also plans to meet on a weekly basis to update everyone with the status of respective story cards. The team will also separately meet with Dr. Ali Jannesari, the adviser and client of the project, on a weekly basis having sprint demo every other week.

3.2 Feasibility Assessment

We believe it is possible to have simulation of drone in the simulator. We could have simulator-gazebo that could allow us to create a drone simulation, and control the drone. All the functions now we have to have are: 1. It should be open to all platforms; 2. Multiple sessions should be allowed; 3. It must be able to interact with ROS.

3.3 Personnel Effort Requirements

Table 1. *Personnel effort requirements.*

Names	**Roles**	**Expected workload (hours)**
Mehul Shinde	Back-end developer, lead manager	136
Ian Gottshall	Front-end developer, scrum master	136
Bansho Fukuo	Back-end developer, testing engineer	136
Jianyi Li	Back-end developer, testing engineer	136
Sammy Sherman	Front-end developer, report manager	136
Jawad M Rahman	Back-end developer, meeting manager	136

3.4 Other Resource Requirements

Our simulator will utilize pre-existing simulation software called Gazebo which requires a machine running Ubuntu with at least 30GB of storage and ample RAM to ensure satisfiable performance.

For controlling a real drone, of course we will require a drone. Ideally, our platform will be able to select from multiple drones, so we will potentially require numerous drones.

The server will also need to be running ROS in order to communicate with both the simulation and real drone. In addition, we will require a database in order to store the appropriate users data.

3.5 Financial Requirements

No financial requirements have been revealed thus far as our client has informed us that he will be able to provide us with everything that is required.

4 Closure Materials

4.1 Conclusion

This project will meet client's goal of developing an open source simulation system in house at Iowa State University. As part of the team's senior design project, the simulator will serve as a platform for the team to learn and implement various market technologies as well as development practices and get an exposure in real-world software development.

– This open-source project will also serve as a tool useful for people from various walks of life and for professionals in need of such a solution.

– The project will adhere to all the senior design standards and will be worked on under supervision of the adviser.

– The team will follow Agile development methodology to achieve the mentioned goals in stipulated time.

– A proper documentation of the project as well as the open-source API will be uploaded on the website on completion of the project.