CyDrone

DESIGN DOCUMENT

Team sdmay19-35

Client & Adviser
Dr. Ali Jannesari

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: 10/11/18

Table of Contents

List of Figures

List of Tables

List of Symbols

1 Introduction

1.1 Acknowledgement

Team 35's client: Dr. Ali Jannesari

Team 35's advisor: Dr. Ali Jannesari

1.2 Problem and Project 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 ROS as module serving as the simulation environments for the project.

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

1.3 Operational 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 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 Deliverables

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

- 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.
- Fast, robust and engaging.
- It will be accessible on the World Wide Web and will run on any major web browser.
- It will be a cross-platform application, that is, it can be used both form desktop and mobile.
- It has the ability to controlling a real drone and loading flight paths on to it.
- It can save flight paths in the database for later simulation or use.

Deliverables

- 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.
- 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 along with a complete version of this design document by 05/03/2019.

2. Specifications and Analysis

2.1 Proposed Design

We have decided to develop our web-portal by utilizing ReactJS, a JavaScript library for creating user interfaces. React is efficient, easy to implement, and promotes maintainability and usability. Our team is designing and implementing our own simulation environment using Three.js, a cross-browser JavaScript library for rendering 3D graphics, and Cannon.js, an open-source JavaScript physics engine. Three.js was chosen for its ease of use, plethora of features, and vast compatibility with web browsers. Furthermore, we will be taking advantage of the React component for Three.js called Three-full in order to smoothly integrate with our React app. In order to simulate realistic physics, we selected Cannon.js due to its relatively small file size, variety of features, and easy integration with Three.js.

Utilizing JavaScript libraries allows our server to operate with minimal overhead by moving resource heavy tasks to the client side. This allows us to design our server with more effective client management techniques to prevent and mitigate any interruptions, data loss, and all hazardous or otherwise undesirable events. In order to allow users to join simulations in-progress, we will utilize Socket.io, a JavaScript library for real-time communication over web sockets. Whenever a user desires to join another user's simulation, a request is sent to the user running the simulation asking for approval. If approved, a WebSocket will be established between the clients and all updates to the simulation will be broadcasted to the observer(s).

Our current prototype renders a simple, rudimentary drone object in a basic environment consisting of a light and a floor. The user is able to move the drone up, down, left, right, forward and back as well as rotate left and right. There is a separate view in the top right that displays the drone's perspective. Physics have been implemented and the drone reacts to collisions and gravity in semi-realistic ways.

Non-functional Requirements

Requirement	Description
Safety	The system must notify the user if a connection is lost, a collision was detected, or any other potentially hazardous event occurs.
Reliability	All data transmitted must reach its intended target.
Scalability	The system must be able to handle an growing number of simultaneous users.
Availability	The system should be available to interact with 99% of the time.
Maintainability	Current and future developers should easily be able to maintain the system.
Usability	The web-portal must be easy to understand and use.
Compatibility	The web-portal must be accessible from all modern browsers and mobile devices.
Response Time	The system must operate in real-time.

Table 1: Non-functional Requirements

Functional Requirements

Requirement	Description
Video Feed	The drone must broadcast real-time video captured from its on-board camera.
Customizable Environments	Simulation environments must be completely customizable to the user who created it.
Persistent Data	Custom environments and other user data should persist for future use.
Alerts	A notification is sent to alert the user of the occurrence of a hazardous, or otherwise important, event.
Connectivity	The system must implement 4/5G, Wi-Fi, RF, Bluetooth, and GPRS in order to ensure a connection in almost any scenario.
Statistics	The web-portal must display accurate statistics about the drone and environment.
Sockets	Client should connect to other clients via socket if they are viewing their simulation.
Server	Must have a simple server for serving static assets and interacting with the database.
Database	Must implement a database to store persistent data.
Authentication	Access should be restricted to only verified or permitted users.

/Authorization	
NFPA 2400 Compliance	The user and the capabilities provided by our web-portal must be compliant with NFPA 2400, Standard for Small Unmanned Aircraft Systems.
ISO/IEC 12207 Compliance	Our software must follow the software lifecycle process defined by ISO/IEC 12207 standards.

Table 2: Functional Requirements

2.2 Design Analysis

The team has worked towards researching optimal solutions in terms of technologies being used along with developing a prototype of the application.

Our client wanted a cross platform software solution to simulate and control a drone and hence we decided to develop a web application which will be available for users from different platforms. One of the first design decision that the team had to make was to select a framework/library to develop the web application with. ReactJS was the team's first consideration since the client recommended the library. We researched on it and found out that it has a big online community. Couple of the team members who were going to work on the front-end development had prior experience with ReactJS. Hence, we chose ReactJS over AngularJS or any other framework/library available for development of web application such as ours.

One of the goals of this project is to control a real drone using the web application and loading an already simulated flight path on to the drone. This is however part of phase 2 of the design project and will be worked on in the second semester. However, to implement drone controls on the web application we are using a back-end service to translate front-end control input into Robot Operating System (ROS) instructions for the drone. The drone that our client has runs ROS and hence can follow ROS instructions.

In the web application, when the user wants to run a simulation, the client sends an https request to the server which then sends a token that turns the client into a listen-server which now can host other users who just want to observe the ongoing simulation. The listen-server capabilities to edit and select environment for their simulation. The environments are build using THREE.js library and cannon.js physics engine. These technologies were selected because of the resources available online to learn and use them. The technologies were included in the project to give a close-to-real simulation experience that factored physical aspects in a drone flight.

The database stores user data, flight-path data, simulation environments and the data collected by the drone. We are using MySQL connection to database majorly because of the team's prior experience in using the service and because of large online user community of the technology. The server is the only communicating node to the database and any database related operations must go through the server to maintain the uniformity and security of the system.

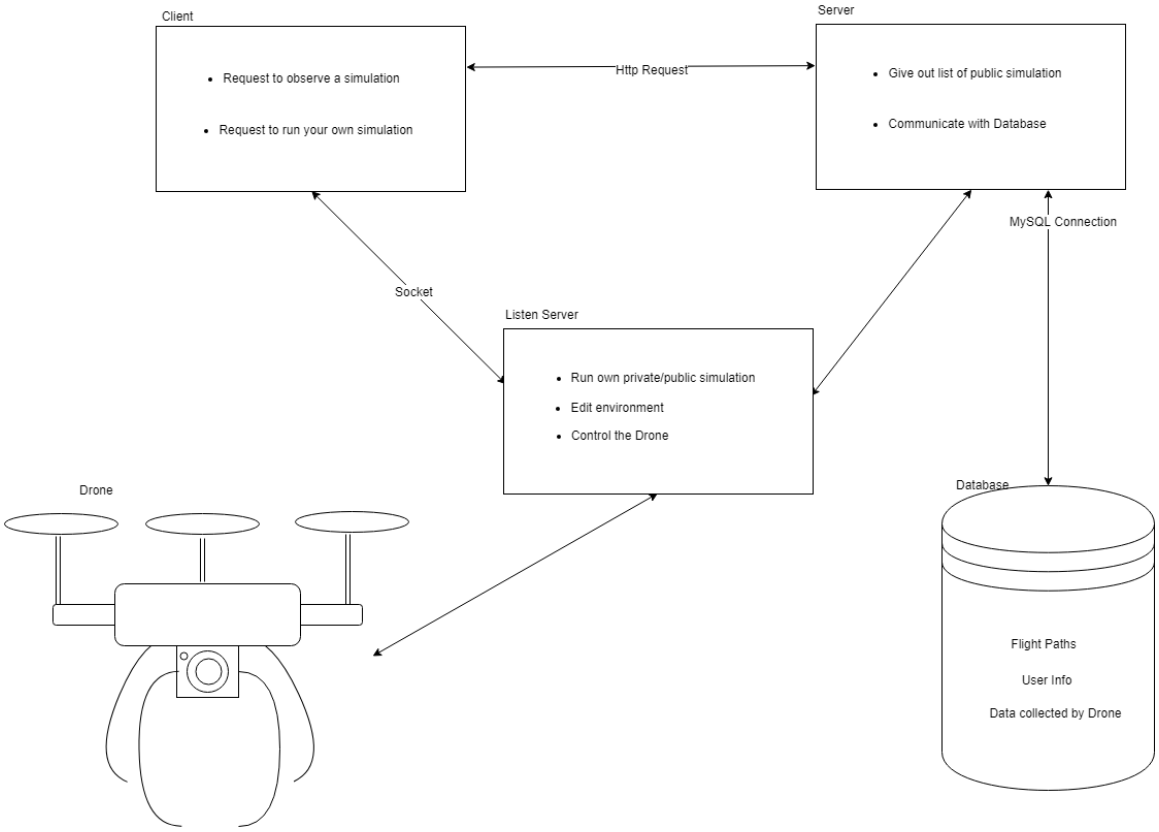

Figure 1: Block Diagram of the project

Because the drone control is part of phase 2 most of the project implementation in phase 1 is drone simulation only. Hence, we have not tried the RC controlling on the drone using the application which will be implemented in phase 2.

Since this project is open source, future developers will have to set up their own database connection because the database used for this project is limited to senior design use. Another limitation of this project is the type of drone that can be used with the web application. The system will be configured for the drone of our client and other drone users might have to make necessary changes to the code to be able to use the application with their drone.

The entire application is open-source and is well documented for developers to work on it further even past the final submission of this project. The web application makes the project platform-independent and hence can be used from a wide array of platforms and devices.

3 Testing and Implementation

3.1 Interface Specifications

The simulation & drone controller web app will communicate with the server if the user are using drone simulation. by control the physical drone, web app will communicate directly with drone, and drone will send the collected data to client side. client will send collected data to server to saving the path data that drone flight to the database.

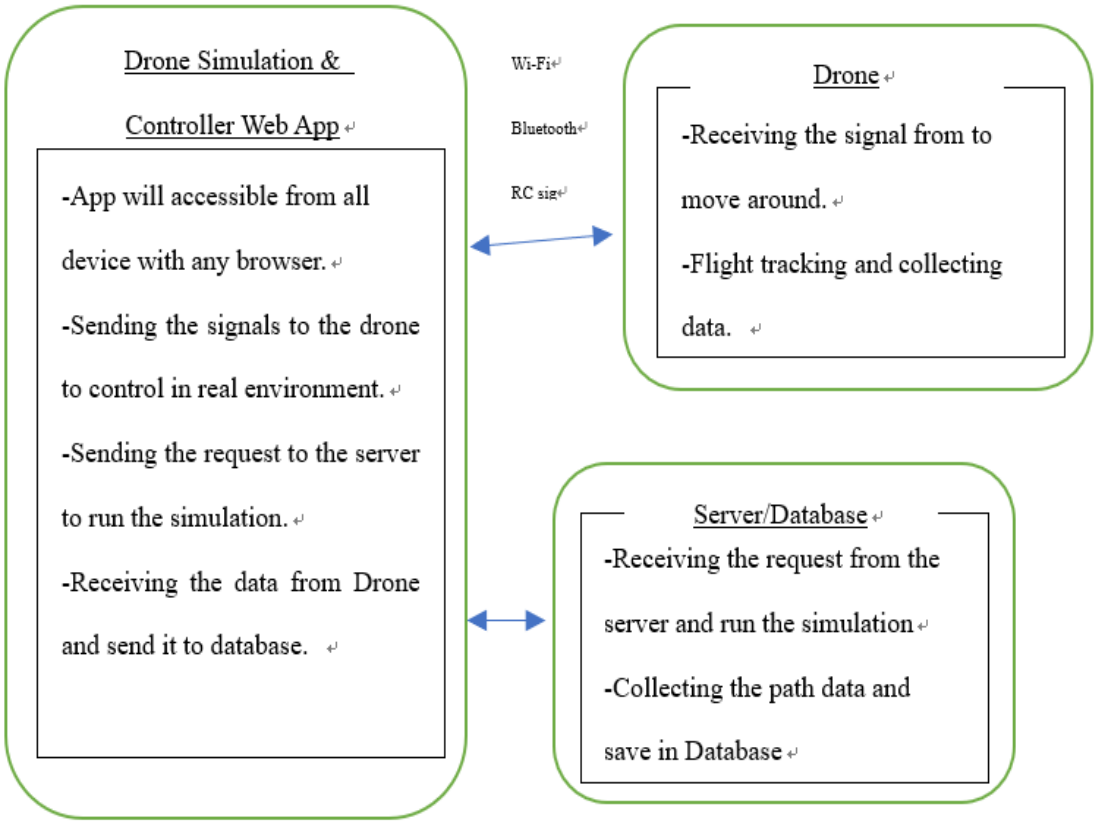

Figure 2: Interface Diagram

3.2 HARDWARE AND SOFTWARE

Since our project involve two part of section: Simulation and Physical drone, for testing the simulation, we only require the testing software which will be Junit and Jest. For testing the physical drone's ability to fly, we need to ensure that the drone meets our controller requirement. That hardware is provided by the project client. The software for testing the server and programing will be Postman and Junit. Those resources can be found on the internet. For the testing the connection and function for the all platform devices, project client will be provided some of phone devices.

Hardware used for test:

- Drone
- Smart Phone

Software used for test:

- Postman
    - To test if the server is always being responsive, we test the server and backend communications using an API called Postman. It provides an interface for sending HTTP requests and checks if the server is sending the correct responses to requests.
- Jest
    - Jest is used for unit testing of our JavaScript code.

Hardware Description

Drone

Figure 2: Drone

(*Source*:thefastmode.com)

Drone equipment with Camera, it's able to connect with the web app (client) by Wi-Fi, Bluetooth, RC signals. It will also have radar, thermal sensor, and so on.

3.3 Functional Testing, Non-Functional Testing

Shown below is a summary of the test plans we have written and implemented.

Functional and Non-Functional Testing	Description
System Testing	• Simulation Functionality • Environment Editor Functionalities • Window Scaling • Browser Compatibility
Safety Testing	• Safety Alerts in Simulation • Low Battery Alert • Weak Signal Strength • Login Security • Safety Alerts for Real Control
Performance Testing	• Server Responsiveness • Calibration

3.4 Process

Section I: Frontend. In addition to automated unit tests using Jest, manual tests of the frontend must be conducted to ensure a high-quality user experience. This section describes tests for the web frontend.

1. Simulation Functionalities
    a. Open the simulation site in Chrome, log in with a test user account, and load a simulation environment.
    b. Try using each of the keyboard controls.
    c. Try using each of the controls on the UI control panel.
    d. Try entering each of the valid commands into the terminal.

    Success Criteria: Each command responds in less than 0.25 seconds and performs the correct action.
    Failure Criteria: Any result other than the success criteria.

2. Environment Editor Functionalities
    a. Open the simulation site in Chrome, log in with a test user account, and load a simulation environment for editing.
    b. Try placing an object.
    c. Try saving and reloading the environment.

    Success Criteria: Each command responds in less than 0.25 seconds and performs the correct action.
    Failure Criteria: Any result other than the success criteria.

3. Window Scaling
    a. Open the simulation site in Chrome, log in with a test user account, and load a simulation environment.

b. Resize the window to a quarter of the size of the screen.
c. Verify that the simulation view resized accordingly.
d. Repeat steps a - c with the environment editor.

Success Criteria: The window resizes properly, and all UI elements are visible and usable.
Failure Criteria: Any result other than the success criteria.

4. Safety Alerts in Simulation
    a. Open the simulation site in Chrome, log in with a test user account, and load a simulation environment.
    b. Try to crash the drone into a nearby obstacle.
    c. Verify that a warning is displayed to the user at least 3 seconds before impact.
    d. Reload the simulation and bring the battery level down to 20%.
    e. Verify that a warning is displayed to the user.
    f. Reload the simulation and begin moving the drone away from the origin point.
    g. When the drone reaches a distance of 2,000 feet from the origin point, verify that a warning is displayed.

    Success Criteria: All warnings are displayed at the proper time.
    Failure Criteria: Any result other than the success criteria.

5. Browser Compatibility
    a. Repeat tests 1-4 using Firefox.
    b. Repeat tests 1-4 using Safari.
    c. Repeat tests 1-4 using Edge.

    Success Criteria: Tests 1-4 pass on the different browsers.
    Failure Criteria: Any result other than the success criteria.

Section II: Backend. This section describes test plans for the backend server. Our backend server is tested using Postman as discussed above.

6. Server Responsiveness
    a. Ensure that the server is running.
    b. From a different machine, load and run each Postman test.

    Success Criteria: All of the Postman tests pass.
    Failure Criteria: Any result other than the success criteria.

7. Login Security
    a. Try to log into the server with a valid username but an invalid password.
    b. Try to log into the server with an invalid username.

    Success Criteria: The user cannot access the system.
    Failure Criteria: Any result other than the success criteria.

Section III: Hardware. This section describes test plans for the drone's performance when being controlled by the simulator.

8. Calibration
    a. Place the drone in an open field.

b. Open the simulation site in a web browser, log in with a test user account, and load a simulation environment.
c. Synchronize the simulation to the drone.
d. Control the using each of the basic movement and rotation controls and verify that the positional data match after each trial.
e. Repeat step d 2 times for accuracy.

Success Criteria: The change in position/rotation observed differs from the simulation by less than a 0.1% margin of error.
Failure Criteria: Any result other than the success criteria.

9. Safety Alerts for Real Control
    a. Open the simulation site in a web browser, log in with a test user account, and load a simulation environment.
    b. Synchronize the simulation to the drone.
    c. Move the drone towards a nearby obstacle, being careful not to actually crash it.
    d. Verify that a warning is displayed to the user at least 3 seconds before predicted impact.
    e. Bring the battery level down to 20%.
    f. Verify that a warning is displayed to the user.
    g. Recharge the battery enough to complete the next steps.
    h. Begin moving the drone away from the origin point.
    i. When the drone reaches a distance of 2,000 feet from the origin point, verify that a warning is displayed.

    Success Criteria: All warnings are displayed at the proper time.
    Failure Criteria: Any result other than the success criteria.

3.5 Results

This project is a two-semester project and still under way. While we have not yet conclusively proven the results of most of our tests, we have begun to implement Test Driven Development by adding Jest to our React project and generating some unit tests. The results of our current unit tests have been used to ensure minor functionality such as the drone's position upon calling a movement function.

4 Closing Material

4.1 Conclusion

This project will meet client's goal of developing an open source simulation and drone controller that will serve as Iowa State University's premier drone control and simulation software. 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. In addition, this open-source project will serve as a useful tool for people from various walks of life and for professionals in need of such a solution. The project will adhere to all the standards mentioned in the requirements and will be worked on under supervision of the adviser. Furthermore, the team will follow Agile development methodology to achieve the mentioned goals in stipulated time. Proper documentation of the project as well as the open-source API will be uploaded and maintained on the website throughout the development process.