CyDrone

Team sdmay19-35

Dr. Ali Jannesari, Client & Adviser

Bansho Fukuo, Ian Gottshall, Sammy Sherman, Jianyi Li, Jawad M Rahman & Mehul Shinde.

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

Problem Statement

What does the client want?

➔ A cross platform and open-source application to simulate and control the drone and digitally recreate the real flight environment inside the simulator

Our solution:

➔ CyDrone - an open-source web-application built using ROS, ReactJS, Gazebo, ODM and Blender / Python.

Functional Requirements

Drone Simulation

The web application should allow the user to control a drone in a simulation from their web browser of choice.

Drone Control

The user must be able to control a real drone using their keyboard, onscreen controls, or a flight planner.

Computer Vision Generated Environments

Environments used in the simulator shall be generated by applying computer vision techniques to the video feed captured from the drone’s onboard camera.

Non-Functional Requirements

Safety

When the connection to the drone is lost or interrupted, a collision occurs, a fault occurs with the drone, or any other hazardous event occurs, the situation must be handled safely and the user must be notified.

Scalability

The service must be able to handle a growing number of active users without allowing performance to suffer.

Compatibility

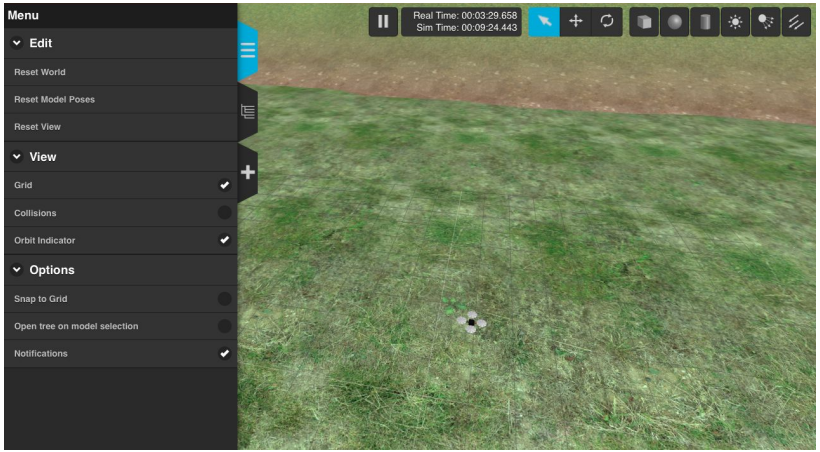
Any modern browser, including those on mobile devices, must be able to access and interact with the web-portal and all of its features.

Deliverables

1. Web application that will be able to simulate a drone in a variety of environments, be able to control the drone and provide necessary data in the process
2. Documentation:
    a. Code
    b. Contribution of the members
3. Manual:
    a. Interaction of the front-end and back-end
    b. Component diagrams and flowcharts

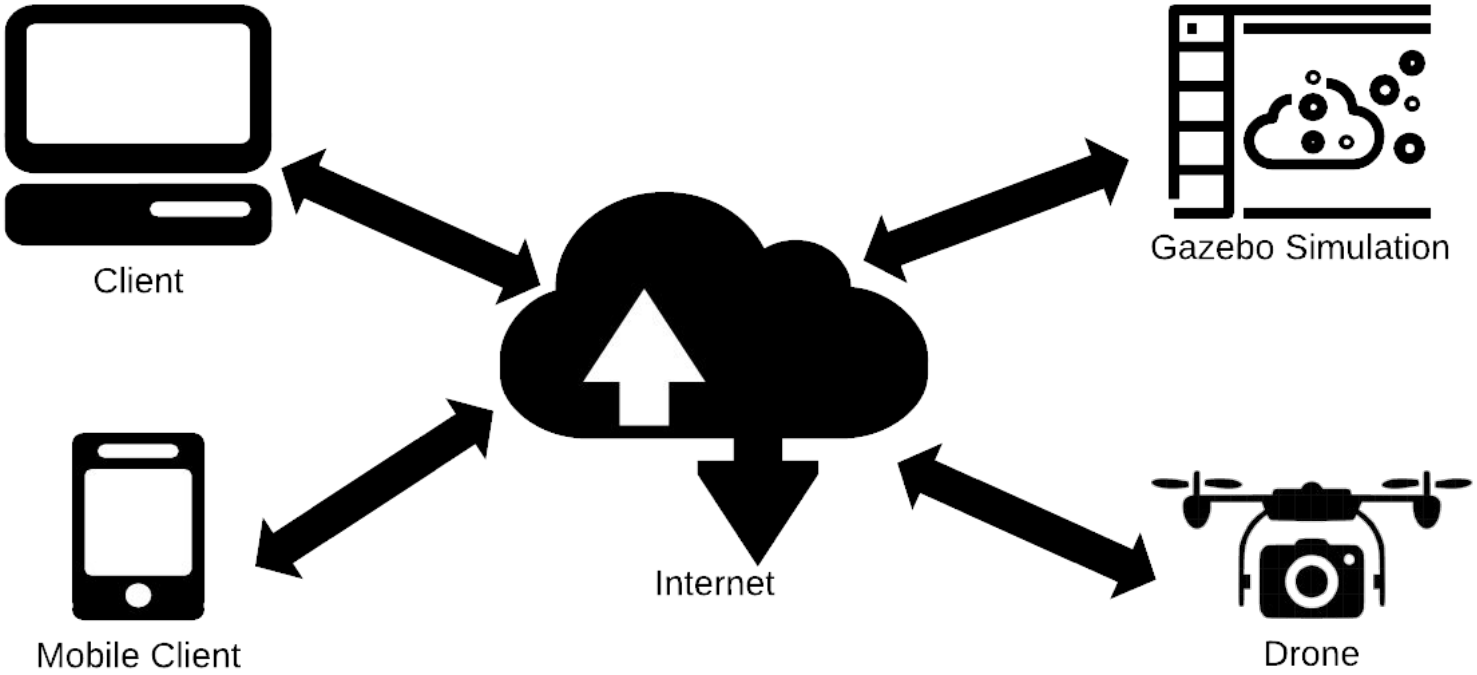
Market and Literature Survey

- Gazebo
  - Open source robot simulator
  - Customizable environment
  - Supports C++ plugins and ROS integration
  - Drawbacks:
    - Resource-heavy
    - Development is not cross-platform
- AirSim
  - Open source autonomous vehicle and drone simulator by Microsoft
  - Built in Unreal Engine
  - Excellent graphics
  - Drawbacks:
    - Need to learn Unreal Engine
    - No ROS integration



Source: “Aerial Informatics and Robotics Platform.” *Microsoft*, https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/.

Conceptual Sketch



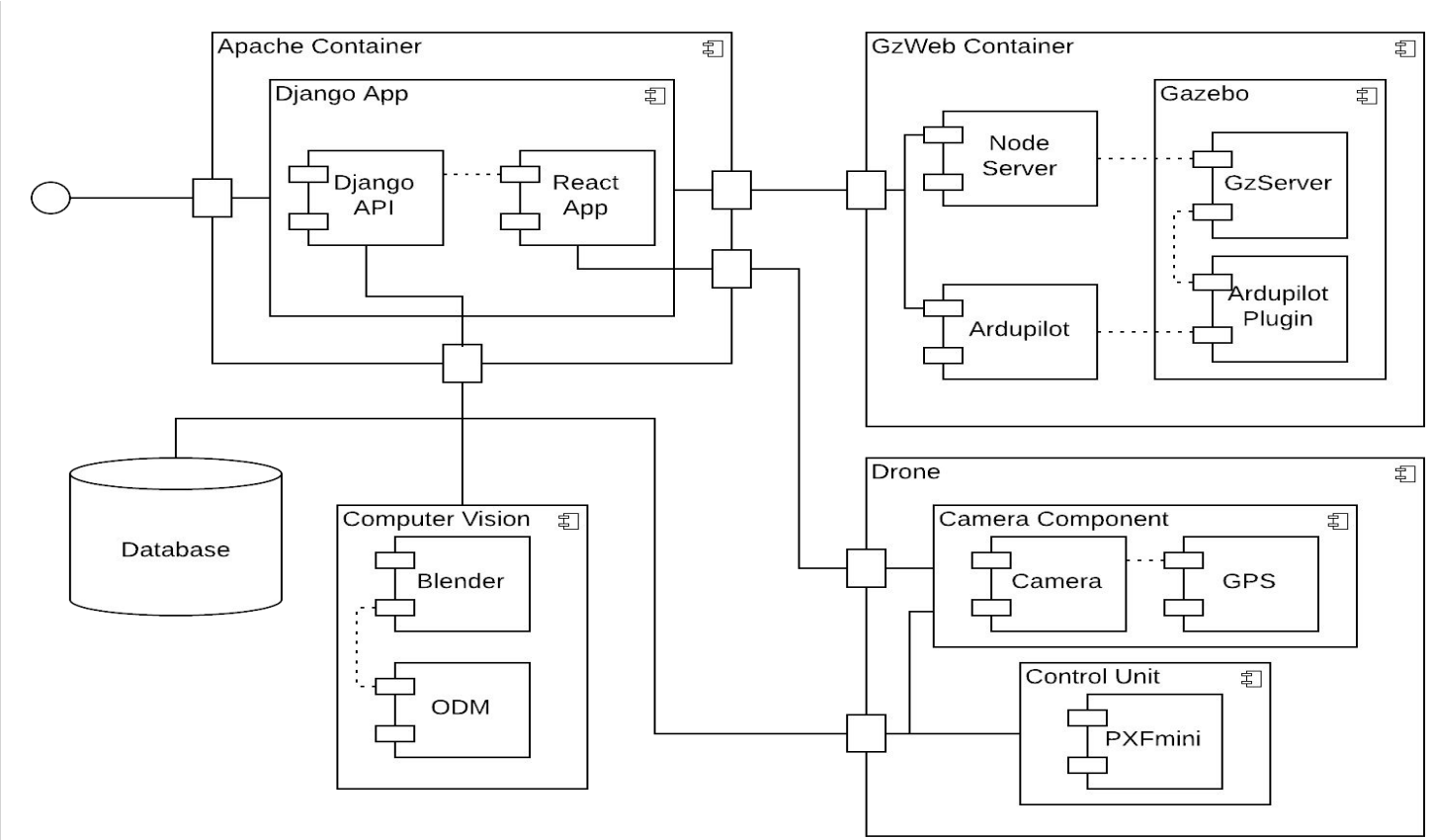
Resource Requirements

- Erle-Copter
- Server running Docker with ample Memory, CPU and GPU
- Camera
- Raspberry Pis
- Project duration - 28 weeks, 9 hours per week per member

Risk & Mitigation

- Risk
  - Gazebo is resource heavy
  - Calibrating the drone with the simulation
  - Incompatible software
  - General unfamiliarity with technology used
- Mitigation
  - Working with the client for equipment
  - Conducting thorough research and testing often

Detailed Design - System Architecture



System Analysis & Details

- Django app (Python, JavaScript)
  - Serves web frontend (React), handles user requests, and starts simulations
  - Communicates with the computer vision app

- GzWeb container (primarily C++)
  - Gazebo runs the simulation itself, with the loaded simulation environment and flight physics
  - GzServer and Node server relay the simulation to the web client
  - ArduPilot handles piloting the drone and plotting flight paths

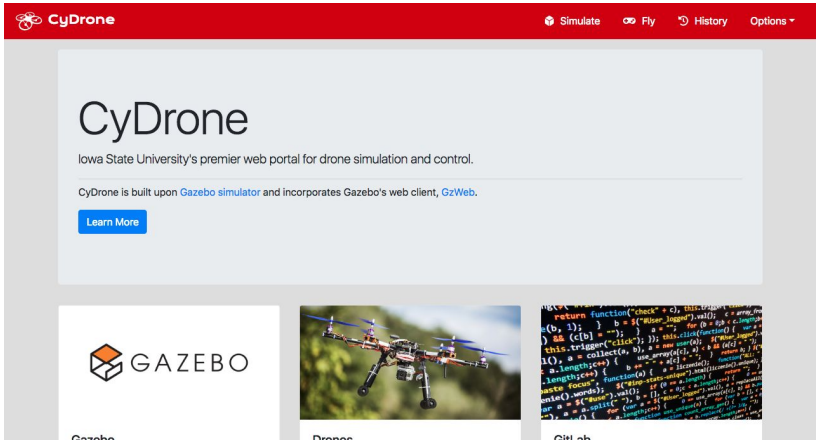
GAZEBO

[Django Logo Source- “Django.” Django, www.djangoproject.com/. Gazebo Logo Source - Osrf. “Why Gazebo?” Gazebo, gazebosim.org/.]

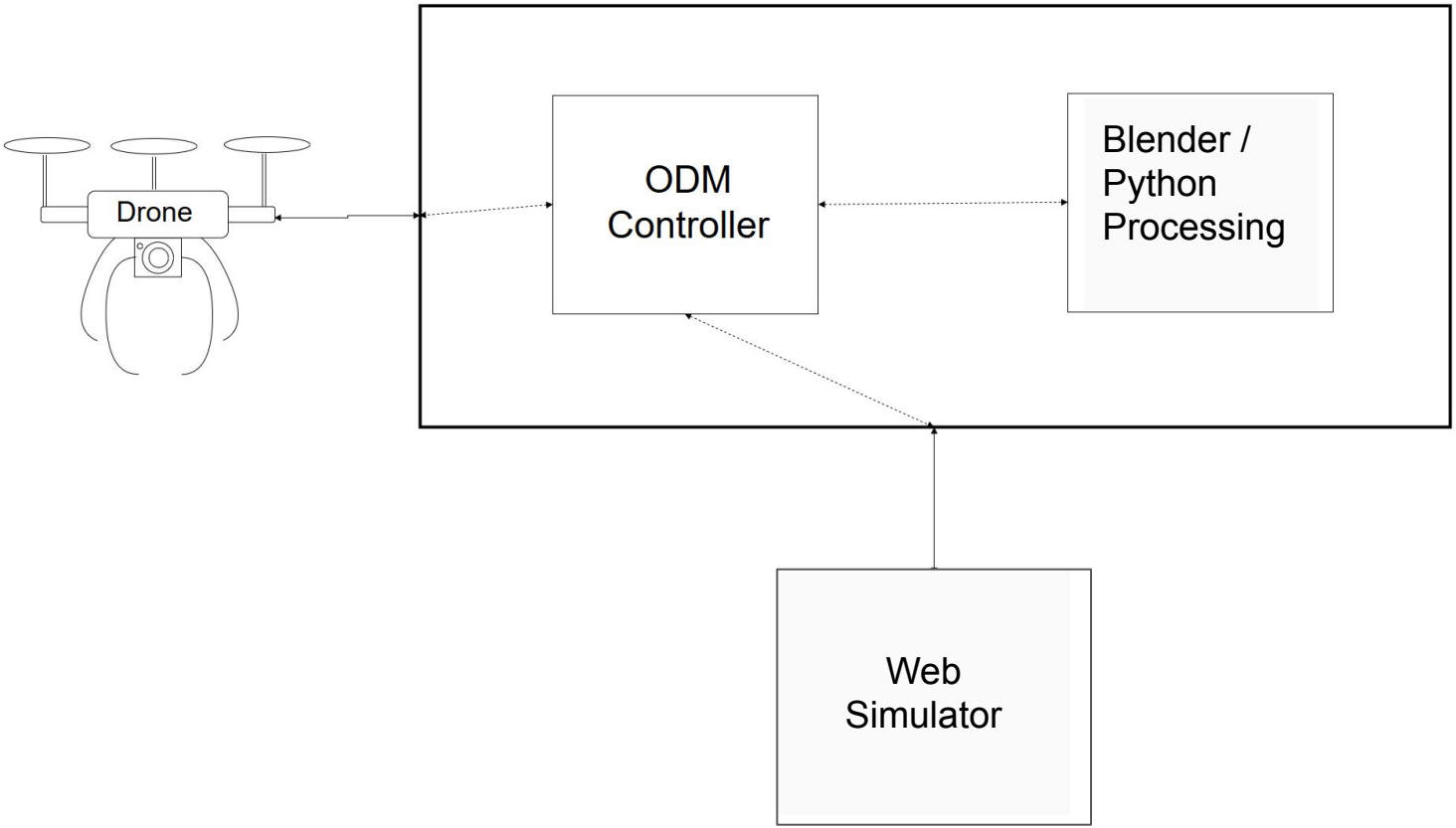
User Interface Description

A ReactJS based web application consisting of 3 main components:

- The homepage displaying news, tutorials and other useful information
- A page for allowing the user to run and control a drone simulation
- A page for launching and controlling a real drone



Detailed Design - System Architecture - CV



System Analysis & Details

- Computer Vision
  - Open Drone Map (ODM) creates 3D model file from drone imagery
  - Blender / Python edits, scales and textures the 3D model file; exports it as a COLLADA file
  - The file is then included in a Simulation Depiction Format (SDF)

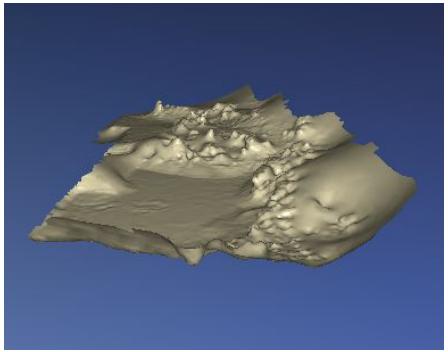
[ODM Logo Source: OpenDroneMap. “OpenDroneMap/ODM.” GitHub, 23 Apr. 2019, github.com/OpenDroneMap/ODM.]

System Analysis & Details

Virtual Environment Generation Process

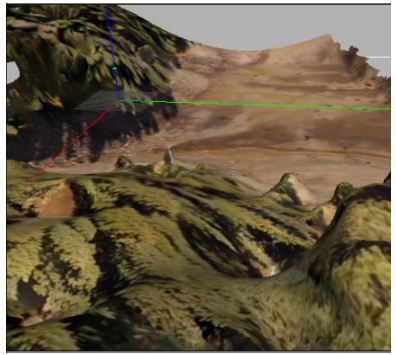
Geotagged JPEG

ODM Script



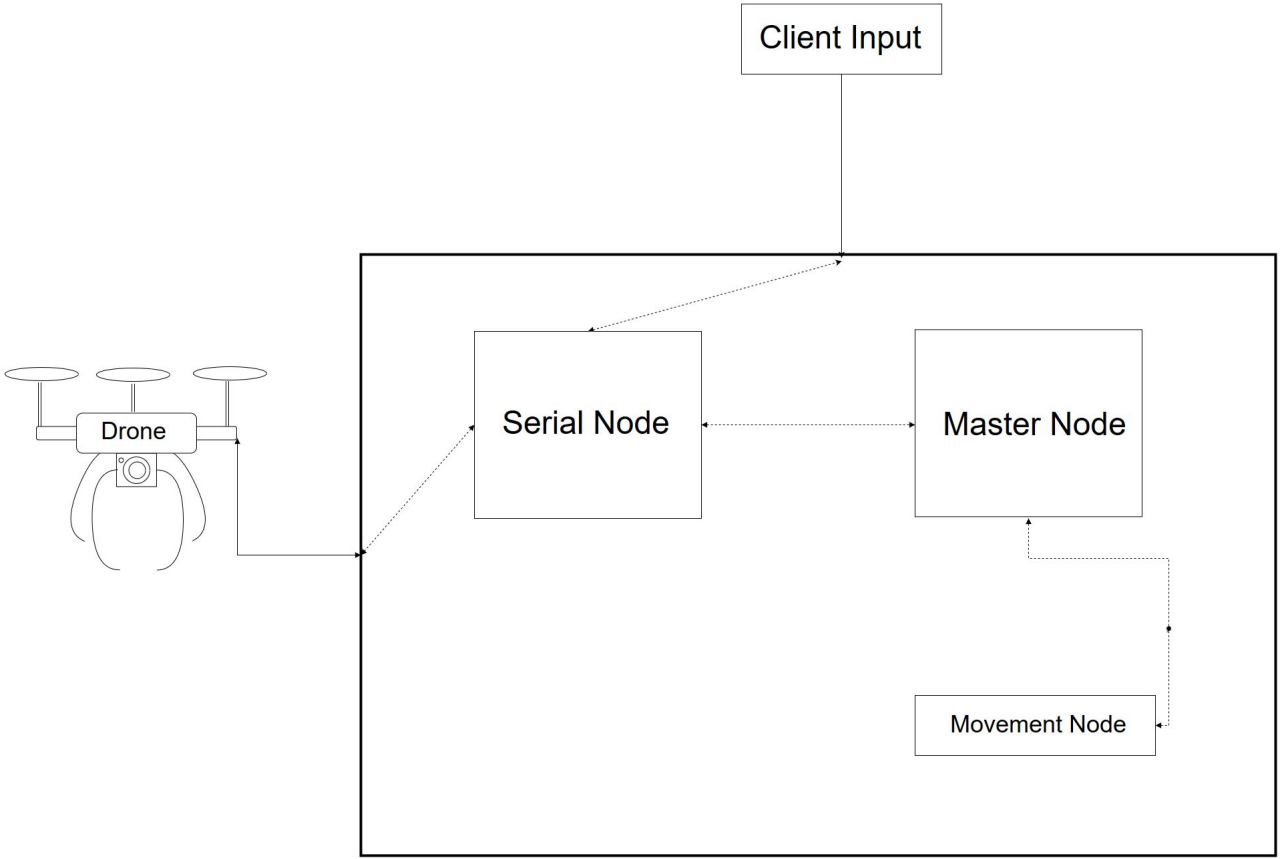
3D Wavefront (.obj)

Blender/ Python Script



COLLADA (.dae) inside an SDF inside a ‘world’

Detailed Design - System Architecture - ROS



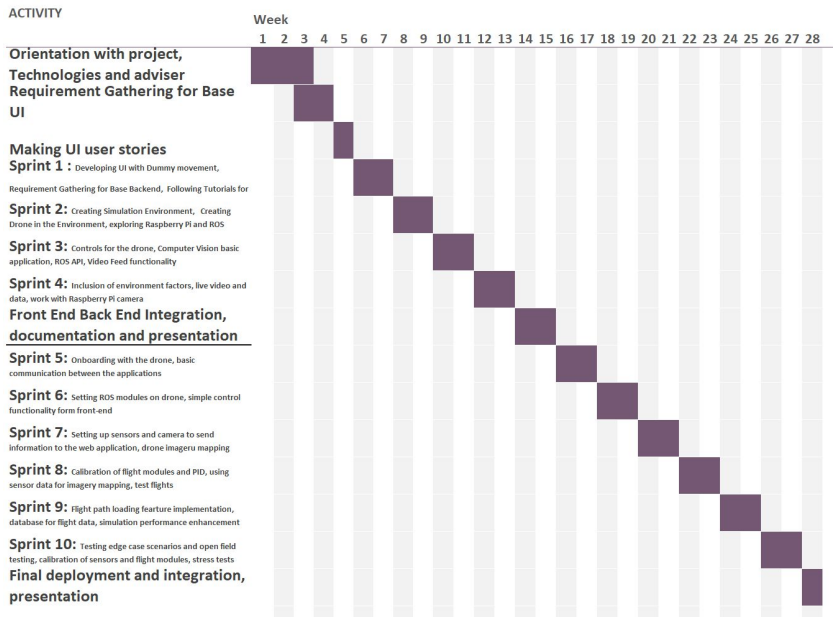
System Analysis & Details

- Hardware
  - Raspberry Pi and PXFmini used for drone control (C++)
    - Runs Robot Operating System (ROS)
  - APM planner is used to calibrate, control and get status updates of the drone
  - Radio Control is also used for operating the drone
  - Another Raspberry Pi used for image processing (Python)
    - Livestreams video from the drone
    - Geo-tags the images taken by it for image stitching purposes

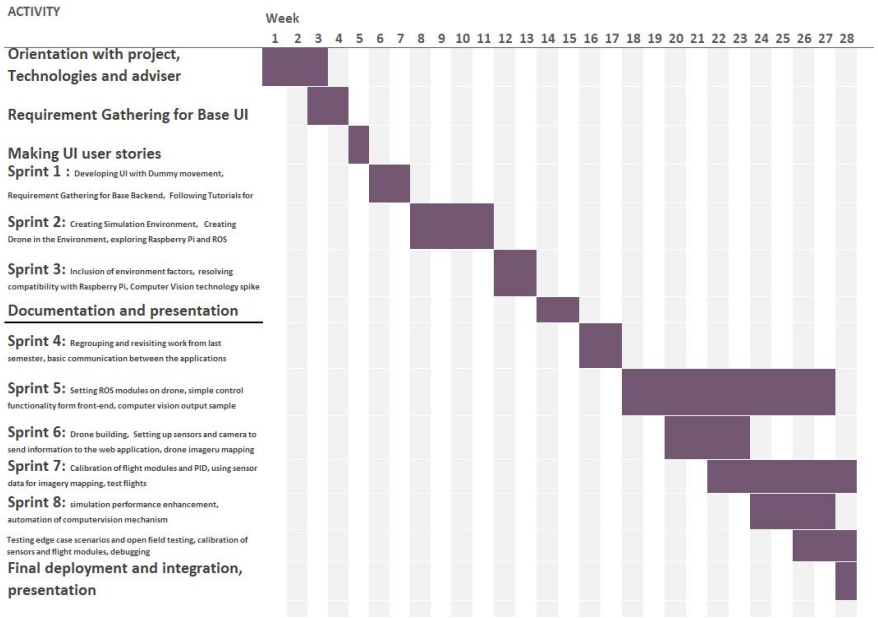
[APM Logo Source: Index of /planner2/_images, ardupilot.org/planner2/_images/. ROS Logo Source:“Press Kit.” ROS.org, www.ros.org/press-kit/.]

Work Plan

Schedule



Proposed schedule



Actual schedule

Challenges

- Compatibility of ROS packages
- Unavailability of system support
- Increasing complexity of the system
- Lack of updated documentation
- Unfamiliar system environment and researching area
- Massive learning curve

Solutions to The Challenges

- Posting on various forums
- Debugging some of the systems ourselves
- Following best practices
- Trial and error approach
- Perseverance

Test Plan: Calibration

- Calibrate the accelerometer, compass and radio control using APM planner.
- Calibrate the ESC according to the methods shown by Erle Robotics.
- Verify that the values displayed right below the Heads-up Display of APM are regular.
- Make sure you have passed all the pre arm checks.

Success Criteria:

APM will not warn the user to calibrate again for the next few flights and show that calibration was successful. The drone also arms using radio control.

Failure Criteria:

Constant warnings from APM to calibrate after the above steps are followed.

[Source:“Why use software for calibration management?” qedge, qedge.sarjen.com/why-use-software-for-calibration-management/.]

Test Plan: Simulation Functionalities

- Load drone simulation environment.
- Controls over the UI control panel.
- Controls over the Keyboard controls.
- Entering valid command 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.

[Source: “Velocidrone FPV Simulator vs. Reality - NCAR Racetrack” William Thielicke, www.youtube.com/watch?v=ewHdnTiNL3M.]

Test Plan: Flight Tests

- Place the drone in an open field.
- Make the drone take off.
- Control the basic movements and rotational controls by using radio control.
- Ensure that the drone is doing exactly as asked.
- Begin moving the drone away from the origin point.
- Make sure that the drone behaves in the same way at different altitudes: 35 feet, 50 feet, and 65 feet.

Success Criteria:

The drone behaves as expected at different altitudes.

Failure Criteria:

Any result other than the success criteria.

Test Plan: Video and Imaging Tests

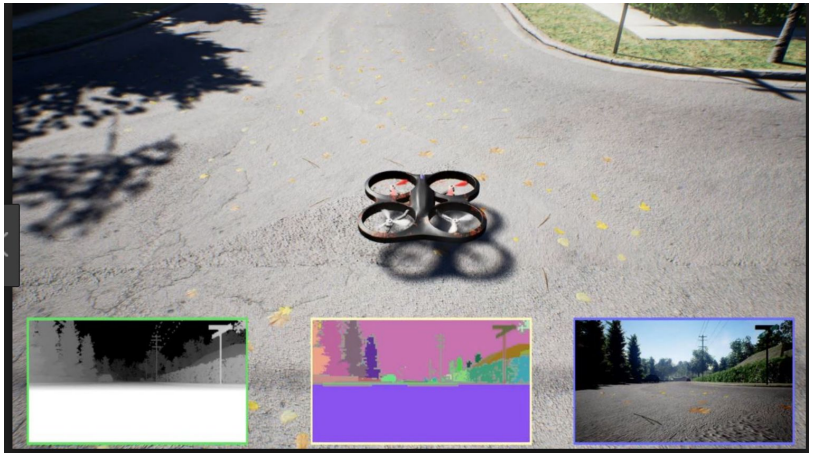
- Place the drone in an open field.
- Open the simulation site in a web browser.
- Control the basic movements and rotational controls by using radio control.
- Ensure that the simulation is receiving video feedback from the drone’s camera.
- Ensure that the quality of video feedback is as expected and there is little to no lag in the video module of the simulation.

Success Criteria:

The drone delivers good quality images and videos.

Failure Criteria:

Any result other than the success criteria.



[Source:”AirSim: A Simulator to Help AI Research for Use in Drones” expouav, www.expouav.com/news/latest/airsim-simulator-help-artificial-intelligence-research-use-drones/]

Test Plan: Generating 3D environment model

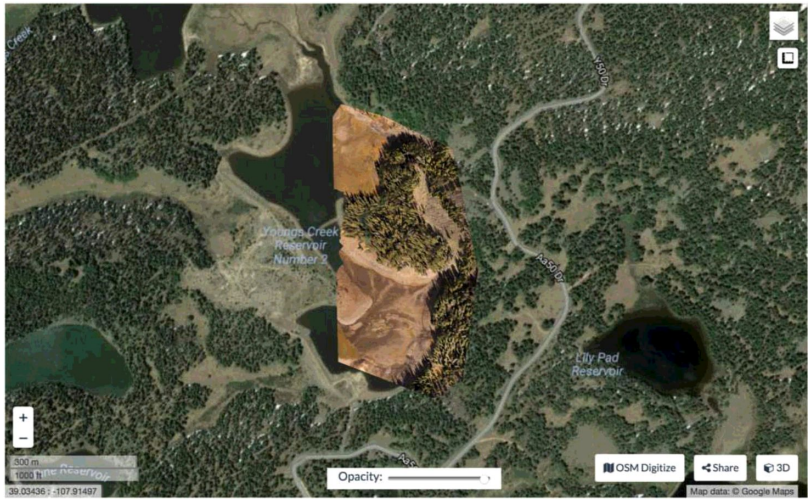
- Login to the simulation site & load a Drone control.
- Take off the Drone to certain attitude.
- Control the basic movements and rotational controls.
- Post-flight, load the images captured by the drone onto the computer vision module.

Success Criteria:

The computer vision module generates a virtual environment almost identical to the real environment.

Failure Criteria:

Any result other than the success criteria.



Trade-offs & Future Work

- Trade-offs:
  - APM in lieu of ROS for controlling the drone from a Computer.
  - Mechanics and aviation knowledge could be applied.
  - Developing our own simulator is very complex and time consuming.
- Future Work:
  - Environment Editor
  - Adding Sensors
  - Machine Learning, object detection
  - Autonomous Flight
  - Flight Path Storage

Q&A

THANK YOU