sdmay19-35: Implementing a Web Portal System for Drone Simulation and Control

Biweekly Report 1

January 29th - February 12th

Client: Ali Jannesari

Faculty Advisor: Ali Jannesari

Team Members

Bansho — *Test Engineer. Sensors Hardware Developer.*

Ian — *Scrum Master. Full Stack Developer.*

Li — *Test Engineer. Back-end Developer.*

Jawad — *Meeting Manager. Embedded Systems Developer.*

Mehul — *Project Lead. Computer Vision Developer.*

Sammy — *Report Manager. Lead Front-end Developer.*

Summary of Progress this Report

- Replaced our GzWeb docker image with an official one from osrf's (Open Source Robotics Foundation) docker hub repository.
    - Created a docker-compose script to automate building and running.
    - Reduced file structure to only include the gzweb files we actually modify.
    - Update dockerfile to build upon osrf's image by adding our custom plugin, worlds, and gzweb modifications.
    - Improved movement (still very lacking), added auto-leveling (about the x and z-axis) and added rotation about the y-axis
- Built an image upon mysql:5.5 which loads a dump of the initial state of our development database.
    - Pushed to the hub as cydrone/mysql
- Removed our node server and moved the react project files to their own directory
    - Improved the dockerfile for react app and broke the build into 2 stages (production and development)
    - Added running the tests as a step in the build process so that the image can't be built if errors exist
    - Created a docker-compose script to automate building and running locally
    - Docker-compose script uses cydrone/gzweb image to start an instance of the gzweb container for the simulation page
- Created a Django API for our backend and created a docker image and docker-compose script for production and development.
    - Dockerfile is multistage with 1 stage for production and another stage for development

    - Pulls the latest cydrone/react image
    - Development stage exposes the port and starts the development server
    - Add docker-py package and spawn gzweb containers when simulation/start is visited
    - Use sessions to keep track of which container each user should access
    - Docker-compose script uses the cydrone/mysql image as a mysql service to allow easily running the Django backend on any machine
    - Uses a volume to allow the container to communicate with the host's docker.sock and spawn containers
- Made an Apache directory and created a dockerfile for our server
    - Built upon httpd:2.4.38
    - Uses mod_wsgi to serve our Django (python) API
    - Custom httpd.conf also exists in directory to load the mod_wsgi module and add other custom configurations
    - Adds daemon to docker group and sets the group of the /run directory (where the volume for the docker.sock exists) to docker
    - Docker-compose script uses the cydrone/mysql image as a mysql service to allow easily running the Apache server on any machine
    - Uses a volume to allow the container to communicate with the host's docker.sock and spawn containers
- Set-up our server and deployed our web application
    - Installed Ubuntu 14.04 (per recommendation)
    - Created sdmay19-35 user and added to groups: root, docker.
    - Installed docker, pulled images and set-up docker-compose and deploy scripts for automating the process of updating the images.
    - Gave 'others' nearly full access to /run/docker.sock (not good).
    - Added application 'ctop' to see the containers running and the resources they use.
- Setup the drone and implemented the ROS system
    - Assembled the drone
    - Connected all the pins to the pxf mini following the provided documentation
    - Installed all the necessary packages in the raspberry pi and configured it
    - Created the ROS system on the raspberry pi and created and currently developing a package for drone takeoff, using C++

Pending Issues

- Solve permission issue with containers using /run/docker.sock. Currently the only solution that has worked is giving 'others' nearly full permission, which is really bad practice. The solution should be adding the right user to the right group (not sure which).
- Integrate computer vision environment generation service with the environment on the simulator. The computer vision module is functioning as a separate entity and needs integration into the simulator. Though the computer vision library provides numerous types of file outputs, these outputs need to be converted to an SDF file that can be integrated in the simulator.

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Bansho	Worked on Video feeds,configuration on Raspberry Pi and operating system	21	72
Ian	Researched and worked on integrating the ErleCopter simulation with our Gazebo installation	20	75
Jawad	Worked on setting up the drone and setting up ROS packages/nodes on the drone	21	77
Li	Researched on the Erle Copter's setting up document, and setted it up. Checking the manual control, and researched about the Gps	22	73
Mehul	Set up server. Set up Computer Vision service on the server machine	23	75
Sammy	Improved/fixed existing docker builds. Added new docker builds. Set up server. Deployed web application.	23	84

Plans for Upcoming Reporting Period

- Set up continuous deployment for our server. Shouldn't have to ssh into the server and run the script every time it's updated. We want it to update automatically once the image in the registry has been updated.
- Create and improve tests for the react app. There are a couple in place but we should have one for every page. Since there isn't a lot of logic, these will mostly be testing that the appropriate elements are rendered on each page.
- Research how to properly and most effectively test a django api. Once tests are created, the dockerfile should be modified to also run the tests.
- Finish developing the takeoff package and start testing it. Work on drone movements and landing.
- Implement the environment integration with computer vision and test it.
- Research on making the ROS system efficient.