Unmanned Aerial System
Ground Systems
(UAV-GS)

CS-4961 Senior Design
Software Design Document

Prepared by:

Jacky Lu
Tyler Thome
Maria Perez
Claudia Seidel
Juan Hernandez

1 December 2016

CALIFORNIA STATE UNIVERSITY LOS ANGELES

Los Angeles, California
Design Documentation

I. Introduction
A. Outline
   i. Considerations
   ii. Architecture
   iii. Design
B. Description
   i. This document details the design of a server application responsible for providing access
to multiple drones from multiple clients
   ii. We also describe the design of one such client application: a cross-platform user
interface with an easily-portable codebase

II. Design Considerations
A. Dependencies
   i. Wireless LAN 802.11
   ii. Linux (Ubuntu 16.04)
   iii. Robot Operating System (Kinetic Kame)
B. Constraints
   i. Using Robot Operating System (ROS) means we must run on top of Ubuntu Linux,
which weighs on performance when compared to a lower-level or embedded platform
   ii. Users need native performance on multiple devices
C. Goals
   i. Develop a cross-platform application with high performance capacity
   ii. Achieve high throughput of ROS data to user interface, passed through server
   iii. Implement a fluid, easy-to-use interface with easy-to-interpret data displays
   iv. Stream a video feed to user-interface with computer vision features
   v. Create a system which can maintain multiple clients and multiple drones
D. Technologies Used
   i. Qt 5.8
      a. Qt Markup Language (QML) connecting front-end JavaScript logic with back-end
         C++ application code
   ii. OpenCV 3.2.0
   iii. Boost
E. Methods
   i. Create a lightweight C++ server application which encapsulates ROS functionality and
      achieves high speed and scalability using Boost Asio for network interfaces
   ii. Build the user-facing application using Qt for cross-platform application while
      maintaining native responsiveness and performance
   iii. Server maintains a collection of ROS nodes and a collection of client objects containing
      a single TCP socket and multiple UDP sockets
   iv. Integrate OpenCV such that its full suite of APIs can be taken advantage of in future
      iterations

III. Architecture
A. Server
   i. ROSManager: contains encapsulations of NodeHandle, publisher, and subscriber
      instances
ii. ClientManager: keeps reference to instances of TCP and UDP connections to route communication

![Diagram of UAV network interface]

B. Client
   i. CommController
      a. Connect to command event listeners
      b. Connect to server via TCP and UDP sockets
   ii. CVController
      a. Connect to camera control event listeners
      b. Provide access to OpenCV APIs on camera feed
   iii. GUI
      a. Command Controls
         I. Aerial map
         II. Altitude slider
      b. Visualization Widgets
         I. Fuel gauge
         II. Battery gauge
         III. Pressure gauge and graph
         IV. Temperature gauge and graph
      c. Camera View

I. Video output
II. Object detection options
III. Take snapshot button

IV. Design
A. Server
   i. ClientManager – Map of Client IDs to Client objects containing:
      a. Command Listener – Asio TCP socket instantiated on own thread
      b. Data Streamers – POSIX UDP sockets
   ii. ROSManager – Map of UAV IDs to UAV objects containing:
       a. Handle – ros::NodeHandle to keep reference to master process on drone
       b. Publisher(s) – ros::Publishers to explicitly send commands to subscribers on drone
       c. Subscribers – dynamic list of ros::Subscribers corresponding to each sensor and internal component which published useful data
       d. Command interface – a wrapper around a ros::ActionClient responsible for sending navigation goals to the drone
B. Client

i. CommController - QObject that exposes API to front-end for communication with server to accomplish:
   a. Register listeners from UI JavaScript code which connect UDP sensor data stream to widget controllers
   b. Listen for and send three-dimensional coordinate goal commands to server via TCP socket

ii. CVController - QObject to provide special OpenCV functions connected to buttons on GUI and camera view component