Brigham Young University

AUVSI Capstone Team (Team 45)

UAS Subsystem Interface Definition

ID	Rev.	Date	Description	Author	Checked By
SS-001	0.1	10-25-2018	initial draft	Andrew Torgesen	Jake Johnson & John Akagi
SS-001	0.2	10-30-2018	adjusted wording	Andrew Torgesen	Kameron Eves
SS-001	1.0	10-30-2018	adjusted diagram	Andrew Torgesen	Brady Moon
SS-001	1.1	11-05-2018	added introduction and fixed typos	Andrew Torgesen	Brady Moon

1 Introduction

At its heart, the AUVSI competition is a systems engineering competition, testing how well a team can bring together a complex amalgamation of software and hardware components to accomplish sophisticated tasks in autonomy and aviation. While no key success measure directly measures this integration, all of the key success measures are acheived through adequate system integration. Thus, as part of the Concept Development process for the UAS, proper interface protocols must be defined so that inter-component testing can commence as soon as possible. Upon identifying the most critical subsystem interfaces, tests may be designed to evaluate the effectiveness of our chosen means of communicating between subsystems.

2 Subsystem Interfaces

Figure 1 gives a top-level description of the major hardware and software subsystems, as well as how they interface in the fully-functioning UAS. Table 1 lists descriptions of the functions of each software component listed in the figure.

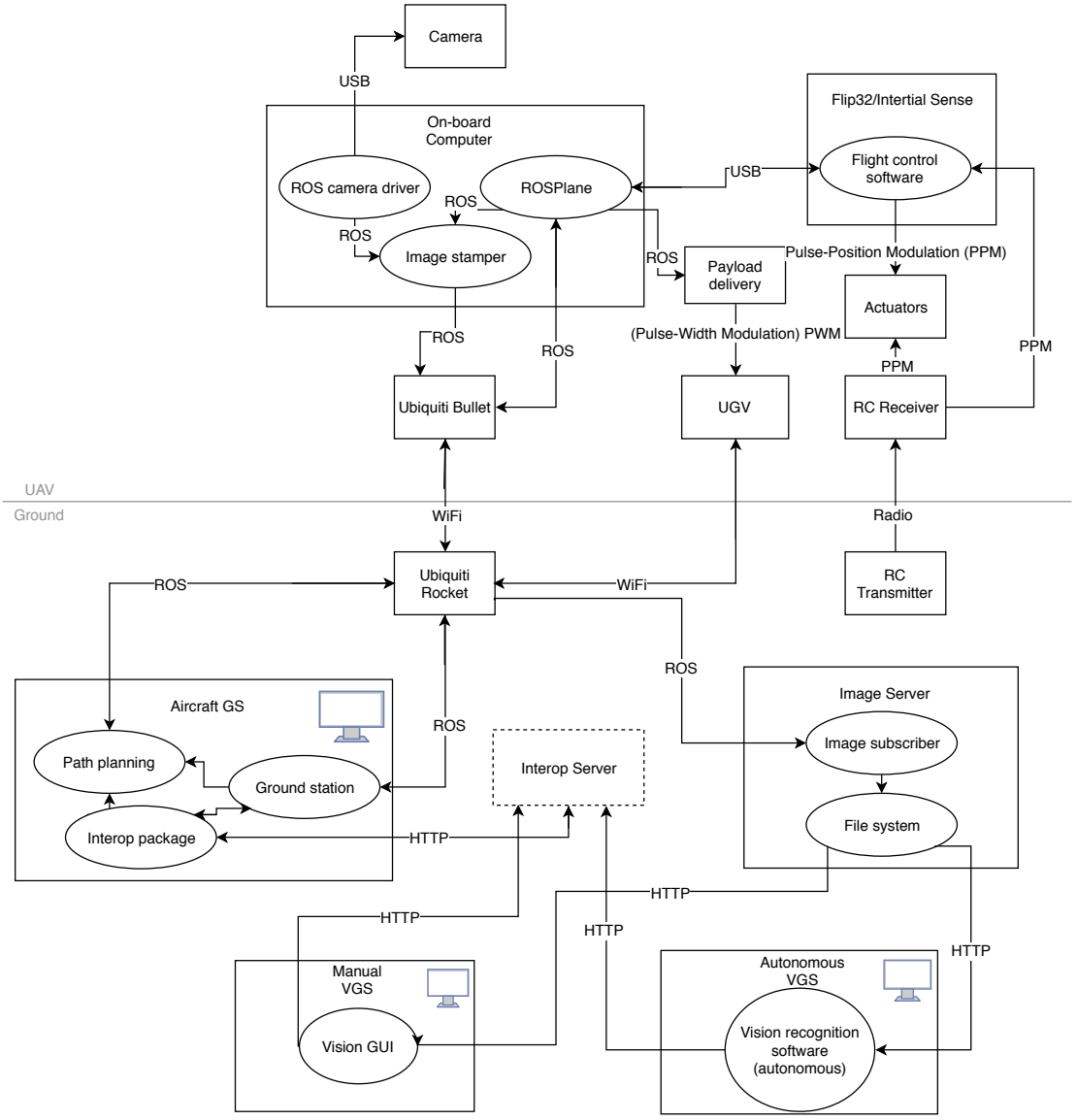

Figure 1: System-wide interface diagram for the UAS. Hardware is denoted by a box, and software is denoted by an oval.

Table 1: Descriptions of the functions of the software components listed in Figure 1.

Software Component	**Description**
ROS camera driver	Reads the serial input from the camera and streams it as ROS messages so other ROS programs have access to the camera images in real time.
ROSPlane	Top-level autopilot. Takes a set of waypoints and converts them into low-level commands to be interpreted by the flight control software. Also constructs a state vector containing all of the dynamic states of the UAS.
Image stamper	Takes streamed camera images and stamps them with time and UAS state data. This facilitates subsequent geolocation of objects found in each image.
Flight control software	Converts low-level autopilot commands into actuation commands and reads in sensor data. Consists of: • ROSFlight: handles autopilot commands, reads in airspeed and barometer data • Inertial Sense: reads in GPS and inertial sensor data
Path planning	Given the details of the competition (including obstacle and flight area data), plans a series of waypoints for the UAS.
ground station	Allows for the visualization of the UAS and provides an interface for sending waypoint, loiter, and return-to-home commands.
Interop package	Communicates with the judges' interop server, and serves up competition details over the ROS network. Also reports UAS data back to the judges' server.
Image subscriber	Captures streamed camera images from the ROS network.
File system	Stores images from Image subscriber on the computer's file system for direct HTTP access by ground station computers.
Vision GUI	Provides an interface for the manual classification of targets in images, as well as reporting the classification data to the judges' server.
Vision recognition software (autonomous)	Runs computer vision software that autonomously classifies targets in images and reports the results to the judges' server.

3 Conclusion

As can be seen from Figure 1, both radio and WiFi will be used to facilitate connection between the subsystems on the ground and in the air. The Ubiquiti data link allows for communication between the ground and the aircraft over a WiFi network. A 2.4 GHz radio link (independent) between the radio transmitter and receiver allows for manual control and arming/disarming of the aircraft.

The Robot Operating System (ROS) is what facilitates the majority of inter-component communication over the WiFi network. ROS is a Linux middle-ware and development protocol for creating modular programs for robotics. ROS allows for real-time communication between machines running individual nodes, or executables, over a WiFi network. In our system, all subsystems communicating via ROS either are or will be developed as ROS nodes to be run on a machine with Linux installed. For more information about ROS nodes and how they communicate over a network, see http://www.ros.org/.