AR Enabled Ground Station

Final Report

Sdmay18-33

Client: Radek Kornicki
Advisor: Diane Rover

Team Members

Ryan Decker: Front End Developer
Nick Pelland: Deliverables Manager | Back End Developer
Jarrett Betke: Communications Lead | Scribe | Back End Developer
Ethan Sabado: Front End Developer
Nick Behrens: Gitlab Master | Back End Developer
Ridwan Faniyi: Front End Developer

Email: sdmay18-33@iastate.edu
Website: https://sdmay18-33.sd.ece.iastate.edu/

1 Introduction

1.1 Acknowledgement

We would like to acknowledge our client, Radek Kornicki, who has provided us with immeasurable support in the forms of technical expertise, funding, equipment, and professional advice has been a key component in our project. We would also like to acknowledge our faculty advisor Dr. Diane Rover, who has helped keep our team functioning smoothly throughout the project.

1.2 Problem and Project Statement

The increasing popularity of drones has led to an massive increase in business and consumer applications. Many of these applications either require or would benefit from better control of the drone. However, in the current market there is a lack of consumer, and professional equipment that provides the pilot with a better operating experience, especially utilizing Augmented Reality (AR) technology.

Our project seeks to integrate the heads up AR display of Google Glass with a custom drone flight controller produced by UAVX. Specifically, our AR display will be able to show alerts, critical data, and general drone status in a efficient and timely manner. This will allow for better control of the drones through increased awareness and ease of use. The pilot will find it significantly easier to find and use the relevant data for their applications, resulting in better piloting and decreased risk of damage to drone or other other surrounding obstacles.

1.3 operational Environment

We expect our product to generally be used outside in clear conditions. However, we require that our product must work in any conditions that a drone pilot would be able to fly their drone in. These conditions are generally not very harsh, as drones generally do not fly in inclement weather due to low power output. Our product is not to be used in heavy rain, extreme winds, thunderstorms, tornadoes, tsunamis, volcanic eruptions, and a variety of other natural disasters.

1.4 Intended Users and uses

We intend our product to be used by drone pilots with a wide range of skills, experience, and applications. This has led us to stick to a general platform in regards to UI design, and not focus on too specific functions and features. However, we do expect our primary consumers to be business operating drones for a business need.

1.5 Assumptions and Limitations

Assumptions:

- The product will be designed for rules and regulations pertaining to the United States
- The Google Glass will only interface with the UAVX ground controller

Limitations:

- Will not connect to more than one controller at any single time
- Will be kept within normal operating temperatures (-20 C to 40 C)
- The product will be within short range of the controller due to bluetooth capability (~15 meters)

1.6 Major Milestones and Deliverables

The deliverables expected at the end of our project are:

1. A fully functioning Google Glass app that features a clean and responsive UI that provides the user with relevant and timely information from the drone
2. An android app that is used to send the commands from the ground control software over to the Google Glass for processing
3. The ground station will communicate with the drone over a wireless protocol

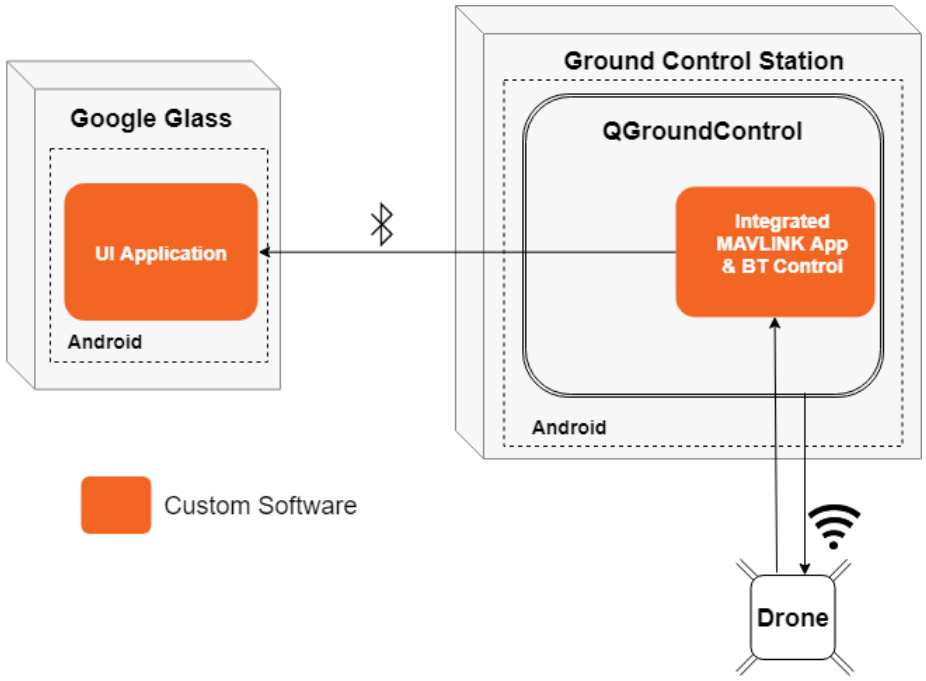
1.7 Previous Work and Literature

In the drone industry, augmented reality and virtual reality are increasingly adopted. Virtual reality equipment is mainly used for drone racing and military applications. In these races, drones have cameras and the pilot has goggles in order to fly the drone in places that are not directly visible to the pilot. However, VR equipment cannot be used for this project because it requires the drones to have cameras, it also does not allow pilots to visually see the drone when piloting it. In military applications, drone simulations using VR are used for pilots to practice in various conditions that mimic reality. All in all our augmented reality glasses will only be showing drone alerts like speed, altitude, and battery like VR equipment for drone racing, however without the visualization and camera feed from the drone.

2 Design

2.1 System Level Design

Figure 2.1.



The systems level design displayed in Fig. 2.1. has three main components: the Ground Station, the Google Glass, and the Drone. The Ground Station is the controller which houses the hardware supporting ground station software QGroundControl. QGroundControl is the open source ground station software that provides full flight control and mission planning for any MAVLink enabled drone. MAVLink, or Micro Air Vehicle Link, is a communications protocol used from small unmanned aerial vehicles, or drones. The drone is the unmanned aerial vehicle (UAV) that is remotely piloted, however for testing purposes, the drone was simulated by ArduPilot. ArduPilot is an open source platform aimed for trustworthy UAV simulation, based on the Arduino platform. Google Glass is the brand of optical head-mounted display in the form of eyeglasses worn by the pilot.

2.2 Ground Station

Under development by UAVX is the Ground Station (see figure 2.2), the hand-held controller powered by a Boundary Devices single board computer with an i.MX.6 processor. The controller will support the modified ground station software QGroundControl, and provide the pilot control of piloting the drone with a geographically mapped visual display of the surrounding topography.

Compatibility with the specified processor is the main requirement for the operating system of this device. The i.MX6 is a 32 bit processor that supports multiple operating systems. The only viable operating system that works with both this processor and the ground control software (discussed in the next section) is Android, which is the operating system used for development.

FIGURE 2.2.

2.3 GROUND STATION SOFTWARE

The main application that allows for in-flight drone communications is QGroundControl (QGC), developed by a group of open source app developers, and modified by our senior design group to support our design requirements.

This application was recommended to us by our client as it a popular ground control software that is compatible with many operating systems. Of the compatible operating systems, only 64-bit linux is supports not 32 bit linux. Since we have a 32 bit processor, we had to build on the Android version of the application as it is the only operating system that is supported by both QGC and the i.MX.6 32 bit processor.

2.4 INTEGRATED MAVLINK APP & BT CONTROL IMPLEMENTATION

The modifications to QGroundControl were developed on QT, which is a cross-platform framework. All of the MAVLINK messages are handled through the implemented C++ *MAVLinkExporter* object. The object has functions to filter MAVLink messages and format them into our specified JSON format. The bluetooth control is created using the JNI framework in order to access the native android bluetooth functions. We use this JNI framework to create static methods that are then accessed by our C++ *BluetoothInterface* object. This interface is what is accessed to start/stop bluetooth connections and send bluetooth messages. The current version of our code has a hard coded socket ID that is constant between this and the Google Glass application. The application currently connects to any device over bluetooth that is paired to the ground station with "glass" (not case sensitive) in the name.

In the QGroundControl settings, a UAVX settings category was created and connected to a UAVX QML file. This UI contains a checkbox that is connected to the global setting *connectGG*. When this variable is true (i.e. the checkbox is checked) *MAVLinkExporter* will attempt to connect to a bluetooth device, specified above, every five seconds if it is not currently connected.

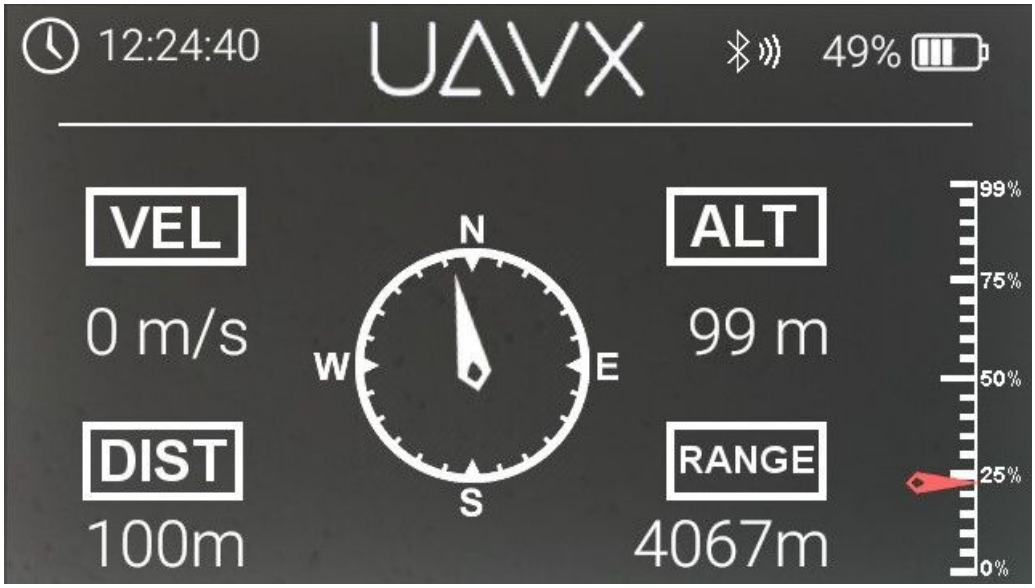
2.5 AUGMENTED REALITY DEVICE

The Google Glass was chosen to display critical flight data to the pilot in real time. The Glass features eyeglasses with a peripheral display projected within the eye glass lens. This was recommended by our client at UAVX due to the ease of use of the devices and available documentation for development.

2.6 Google Glass User Interface

The Google Glass UI (see figure 2.6) was designed to follow the styling choices and standards of previously developed UIs that are/have been used for military jets, rockets, and spacecraft. As a result one standard that we tried to adhere to is the NASA Integrated Display and Graphics Standard (IDAGS).

Figure 2.6.

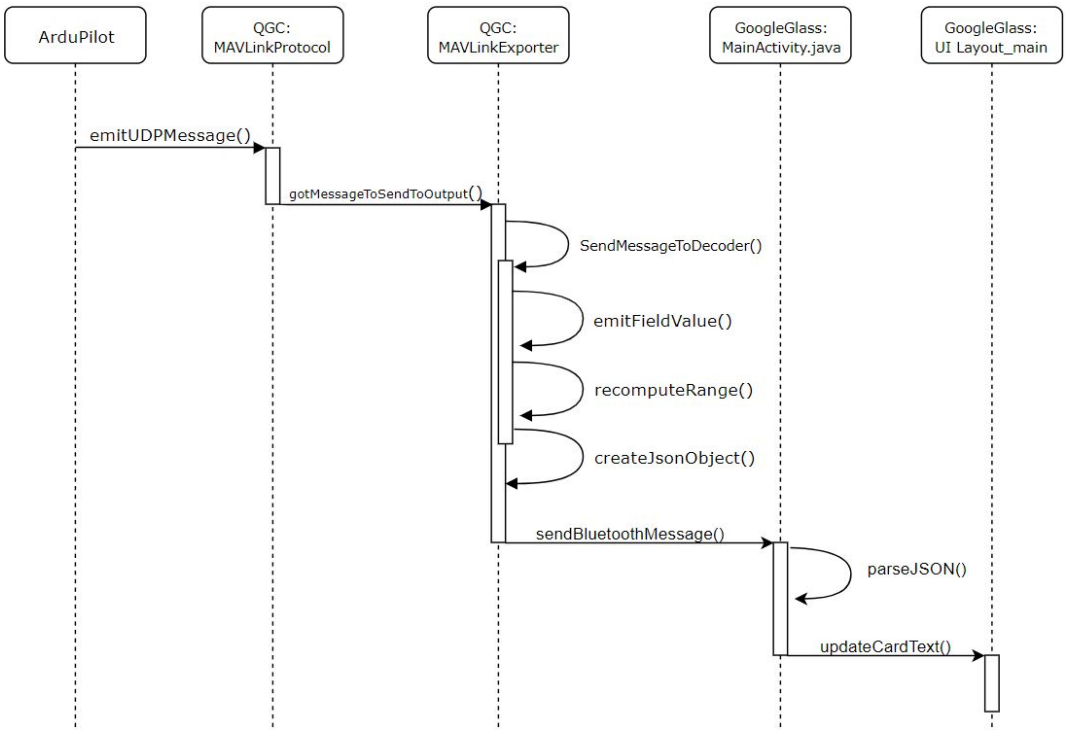


The Google Glass presents an interesting challenge when designing UI, because the screen occupies a small portion of the user's vision, meaning that we must be selective when displaying information, or else it will become too dense. Careful consideration must be given when choosing colors due to an uncertain background. Our testing found that the color white was clearly visible in most environments.

The display initially hides the information, but not the labels. Once connected over bluetooth to a ground station, the wireless icon beside the bluetooth icon will appear, indicating that the Google Glass is correctly connected to a device. Once the Google Glass is successfully parsing the bluetooth messages and receiving telemetry, the text fields underneath the labels will populate and show the most recent telemetry data. The parser works by reading in bluetooth messages through the Google Glasses built in bluetooth framework, and then parsing the information using a JSON parsing library. Our philosophy was to perform almost all the calculations for the telemetry on the ground station side, sending only the values to the Google Glass due to the limited computing power and battery life of the Google Glass.

2.7 MAVLINK Message Code Flow

Figure 2.7.



Above is a code flow for the objects and functions that collect, parse, compute, and send the MAVLINK messages that are received from the drone.

3 Testing and Implementation

3.1 Interface Specifications

There are two main interfaces involved with testing the project, namely the Drone Simulation-Ground Station face and the Ground Station-Google Glass face. Isolated tests were performed on the a Linux machine before interfacing with a Ground Station. Likewise, the Google Glass application was independently built and tested before it was interfaced with the Ground Station. In order to best understand the scope of testing and implementation within the project, each component will be discussed independently according to the progress of the project.

Drone Simulation

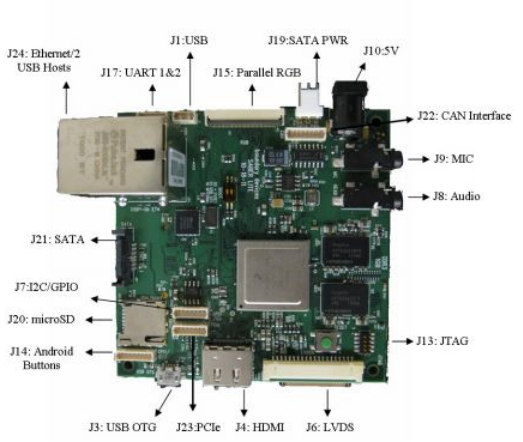
Since there are complexities with directly testing a drone, we opted to simulate a drone for all of our testing. ArduPilot is a common and reliable open source program used for drone simulation. This program has the ability to mimic drones using the MAVLINK protocol and simulate both a fixed wing and copter style drones. All of our testing and designing was done with a copter style drone simulation (ArduCopter) as per recommendations by our client. This program was executed on a Linux machine running Ubuntu and connected to a network using IP assignment.

Ground Station

The project's goal is to implement our ground control software on the custom UAVX ground station controller. These controllers were not available to us during our project, as they are still undergoing production. In order to effectively test our software, we were given the BD-SL-i.MX6 development board which uses the same i.MX6 processor as the UAVX ground station. We were able to successfully run our software on this board to confirm compatibility.

The BD-SL-i.MX6 board (is a low-cost development board built by Boundary Devices. It is equipped with the i.MX6 6Quad application processor that enables development for Linux OS (or Android) multimedia applications. This single-board computer is the functional equivalent of the Ground Station controller developed by UAVX. Displayed in Fig 3.1 is top view of the development board. More hardware-specific information on the development board can be found in the reference material listed in the appendix sec. A.1.

FIGURE 3.1.



This board only allowed for limited testing as there is no onboard bluetooth or wifi chipsets. There are also inherent inconveniences with using this board, such as no on board display or input peripherals. To avoid these difficulties we installed our application onto an android phone after confirming compatibility with the i.MX6 processor running android. Using an android phone allowed us to connect over a network to a linux machine running ArduPilot as well as connect to the Google Glass over bluetooth.

Google Glass

One main feature of this project is the marriage of the UAVX developed ground station and the Google Glass. Google Glass released its first version in 2013 as the explorer edition. A new version has been released very recently, namely the Enterprise edition. Both versions were used for testing and development. Information regarding current hardware/software support and editions can be found in a reference in the appendix sec A.1.

During the composite development of our Google Glass application, we divided up each part to allow the team to develop independently. One of the main features of our application was bluetooth functionality between the Google Glass and the ground station. Ensuring this feature works properly, an android device was used to simulate the ground station and sent bluetooth messages with the RFCOMM protocol. Another part of the Google Glass application is its JSON parser. The end product consists of the Google Glass reading and interpreting JSONs. To test this functionality, we hard coded JSON files into the application and parsed them to make sure that we have the correct output.

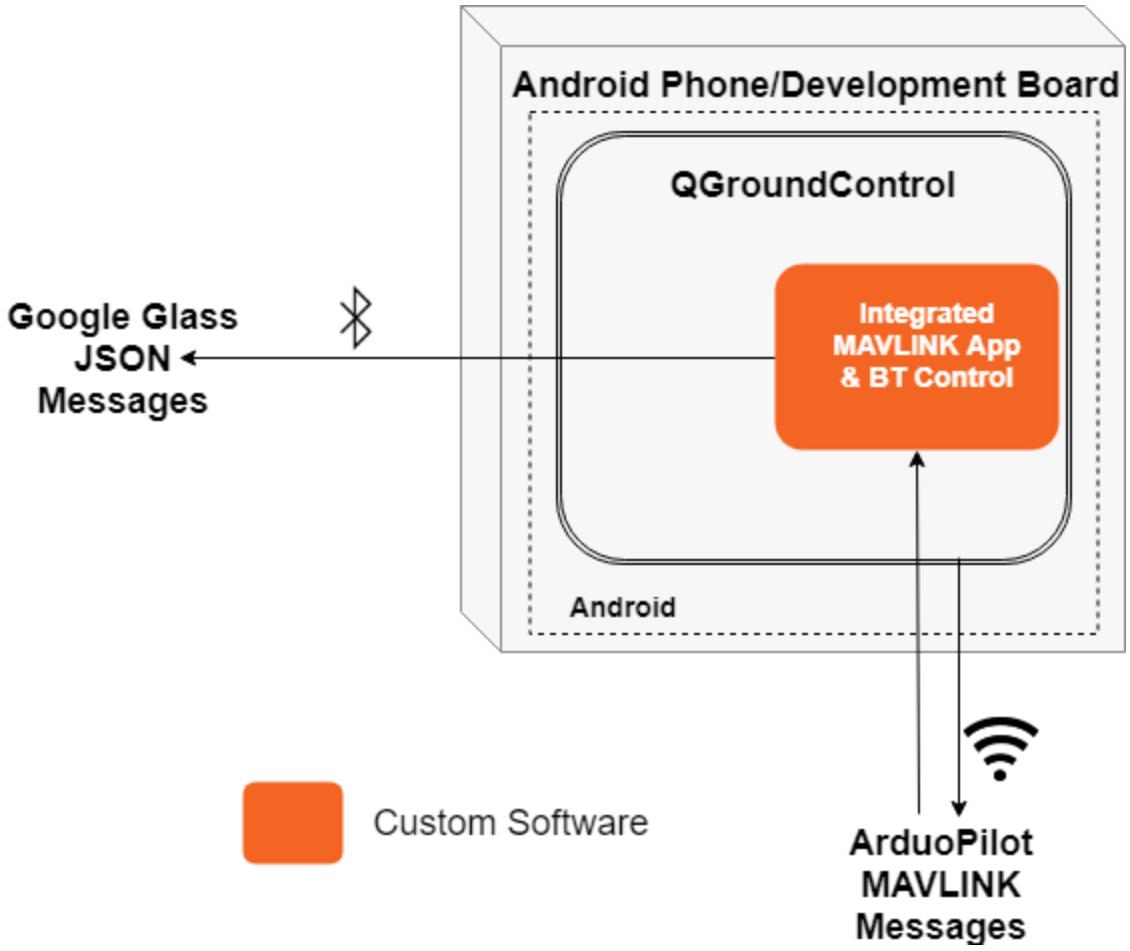
3.2 Hardware and software

In section 3.1, the core testing procedure between the three main components was introduced; the connection between a drone and the ground station, the ground station software, the connection between the ground station and Google Glass, and the google glass application.

As an integrated full system test, we used a Linux laptop running ArduPilot and created a network between the Linux laptop and phone/development board. The phone [development board] was connected to the Google Glass using bluetooth such that the phone [development board] acts as the client sending data to the Google Glass acting as the server.

3.3 Process

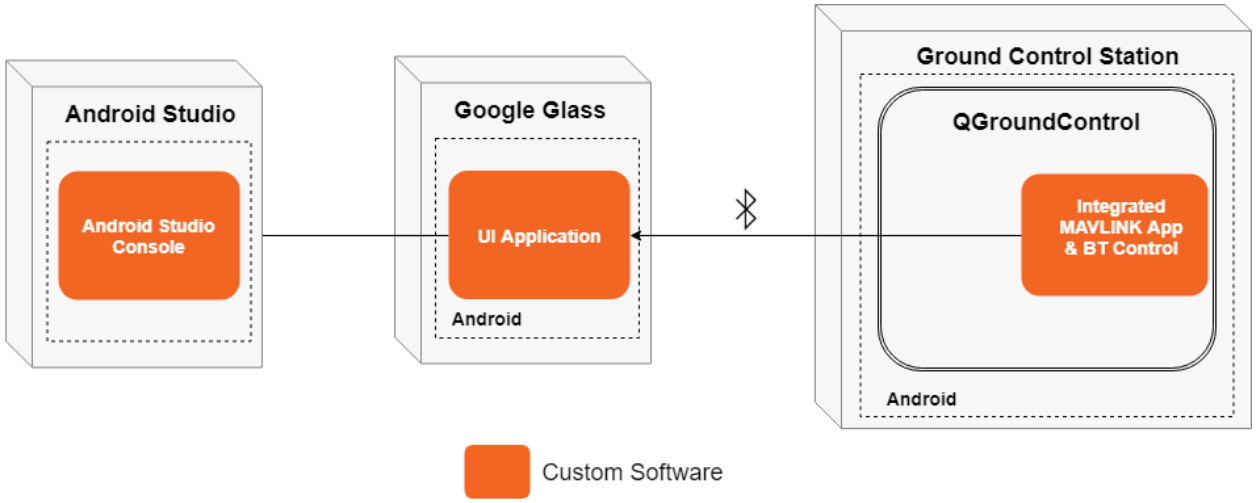
Figure 3.3.1.



Since testing with a drone is dangerous, expensive, and time consuming, we used the drone simulator software ArduPilot (ArduCopter). This software outputs the same MAVLink commands that would be outputted from a compatible drone. ArduCopter is used on a separate laptop or mobile device and sends MAVLink commands over WiFi to the phone [development board]. Our ground control software then interprets most of these commands then filters and processes relevant information. These completed messages are compressed into our specified JSON message format and are sent over a bluetooth socket to

the Google Glass. This JSON data is interpreted and used to update the corresponding graphics on the Google Glass UI. The message accuracy was verified by viewing messages using the Android Studio console.

Figure 3.3.2. Google Glass Test Flow



3.4 Results

In our application testing, the QGC software runs the simulation and generates JSON messages. The Google Glass is able to interpret these messages and update the UI display with velocity, throttle, flight time, drone battery life, distance from launch, throttle, and nautical direction.

4 Closing Material

4.1 Conclusion/Achievements

In relation to our goal for this project, we have been able able to accomplish the required use case functionality for this project, communication between the drone simulation, the ground-station, and the Google Glass. We were able to accomplish this by initially separating the different parts of the projects and assigning them to different members of the group according to their technical interest and strength. After the completion of these different parts, we proceeded to the integration of the project as a whole. The ground-station is successfully communicating with the Google Glass via bluetooth and the drone simulator also sends messages successfully to the ground-station app.

4.2 What could have been improved

Although, the requirements for this project are fulfilled, there are still prospective additions to maximize the functionalities of this project. For example, in order to maximally utilize the AR capabilities of the google glass, calibrating the gyroscope sensors on the google glass such that the pilot can be oriented towards the aerial location of the drone at any point the drone is in the air. This would fulfill the AR description of the project as the project title denotes. Another feature to be improved upon is the connection quality between the individual components of the Google Glass and the Ground Station to make the arrival of the messages on the Google Glass faster. Lastly, we also thought about improving the UI elements on the google glass to diversify features and improve usability.

4.3 Educational Fruits of Labor

Upon reflection of the year-long project, there was a wide range of information, tools, and skills obtained to design and accomplish the intended use case for this project. Regarding the back-end development, members were exposed to the technological background of how aerial vehicles used MAVLink drone protocol library for communications. The Ground Station software QGroundControl was written in C++, and so members were familiarized with the language and gained a technical understanding of the software itself. The Bluetooth functionality required a combination of experimentation and technical work to create a module that operates within QGroundControl. In parallel to the the back-end, the front end development with Google Glass, Android Studio, and communication protocols were required to be integrated for each version of the design, testing iterations, and implementations. There was also a fair amount of graphic design work for the UI displayed icons.

Overall, this project broadened the team members' undergraduate experience to solving problems with hardware and software components, and moreover, gain the confidence that it is possible to individually and collectively solve difficult technological problems.

Appendices

A.1 Reference Material

1. Google Glass Development - https://developers.google.com/glass/
2. Ardupilot Software - http://ardupilot.org/plane/index.html
3. QGroundControl Software - https://donlakeflyer.gitbooks.io/qgroundcontrol-user-guide/content/en/
4. Development Board Manual - https://boundarydevices.com/wp-content/uploads/2014/11/SABRE_Lite_Hardware_Manual_rev11.pdf
5. Development Board Schematics - https://boundarydevices.com/wp-content/uploads/2014/11/sabre_lite-revD.pdf
6. MAVLINK reference - http://mavlink.org/messages/common
7. Google Glass Landing Page - https://x.company/glass/

A.2 Operation Manual

The Operational Manual can be found on our group's GitLab Wiki page:
https://git.ece.iastate.edu/sd/sdmay18-33/wikis/Home

Or hosted on publicly hosted on Github at:

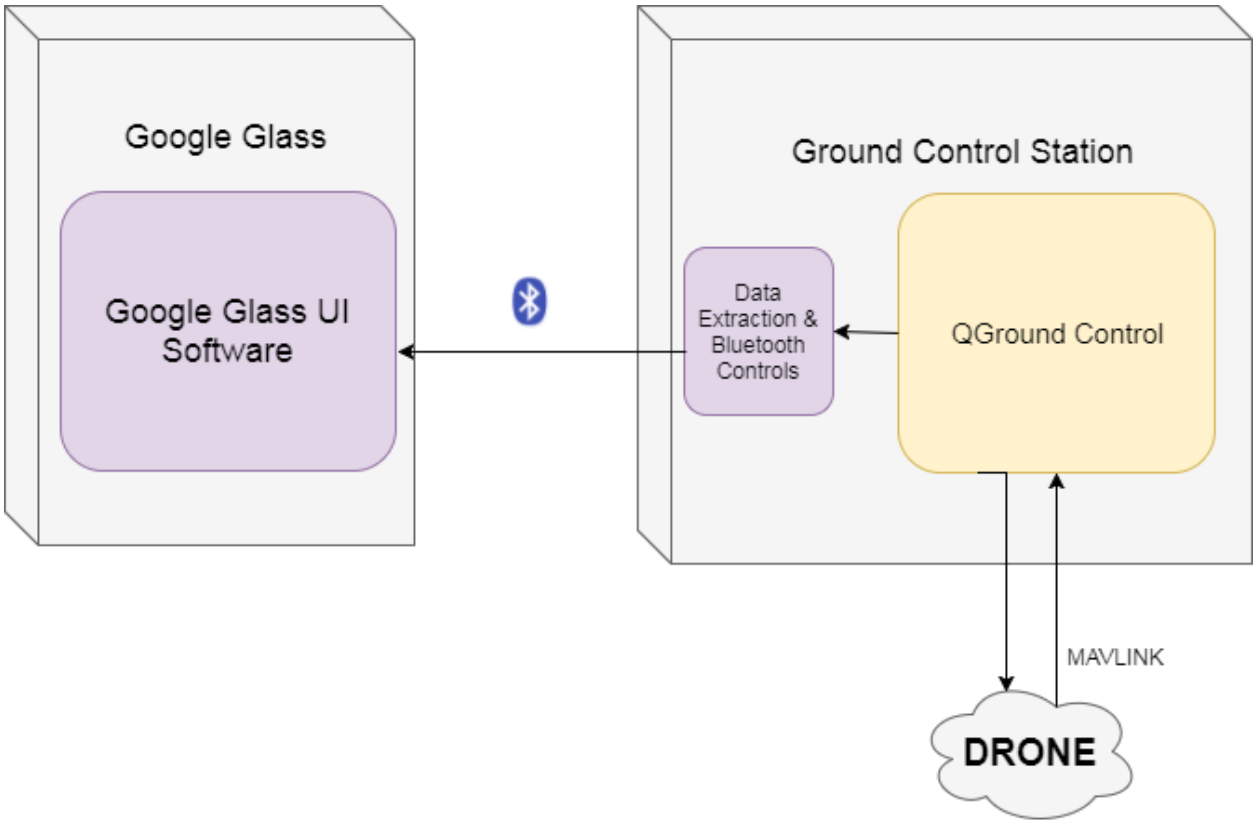
https://github.com/nab0310/sdmay18-33/wiki

A.3 Project Design Versions

During the design process, there were many roadblocks that caused us to make major changes to our project design. The first major change we had to make switching the operating system of the ground station from Linux to Android. This was so because there is not native support for 32-bit Linux with QGroundControl. We looked into other ground control software, but QGroundControl still seemed to be the most widely used and available, so changing the operating system to Android (which natively handles both 32 and 64 bit apps) was the rational solution.

The second major change we made was integrating the data extraction and bluetooth controls directly into QGroundControl, exemplified in figure A.3. This was done because we were unable to listen to the UDP port that the MAVLink messages were being sent over at the same time that QGroundControl was listening to the port. Ideally, the data extraction would be separate from the ground control software, but this was not feasible and would require more processor time. Integrating this aspect of our project into the QGroundControl software also allowed us to create more UI functionalities within the app allowing a simpler connection process to google glass for the user.

Figure A.3



A.4 Code

https://git.ece.iastate.edu/sd/sdmay18-33