DRONE FOLLOWER

Kyle Henry

Joshua Regresado

Nabilah Ahmed

FINAL REPORT

12/04/19

Table of Contents

DRONE FOLLOWER

Nabilah Ahmed
Joshua Regresado
Kyle Henry

Concept of Operations

REVISION 3

05 December 2019

Concept of Operations

for

Drone Follower

Team <49>

Approved by:

_____

Project Leader Date

_____

John Lusher II, P.E. Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	9/16/19	All Members		First Pass
2	9/30/2019	All Members		Second Pass
3	12/5/19	All Members		Final

Table of Contents

List of Figures

1. Executive Summary

The Drone Follower system will allow a drone to track an object of differing size and speed. The tracking will be done using computer vision, which will then be translated into telemetry for the drone's flight controller via software in an on-board Raspberry Pi. The tracking system will utilize text detection to identify the desired license plate number, which can then be autonomously trailed by the drone. This video feed and telemetry is then fed to the user. This drone follower system has a variety of uses but will be designed with law enforcement and the military in mind for tracking of vehicles.

The basic capabilities of this system include:

- Tracking of an object such as a person or vehicle from altitude
- Camera gimbaling to keep the target in-view and produce stabilize feed to the user
- Close-quarters tracking of an object through a closed space

Notable drawbacks of this system include:

- Speed at which the drone can move to track high-speed targets

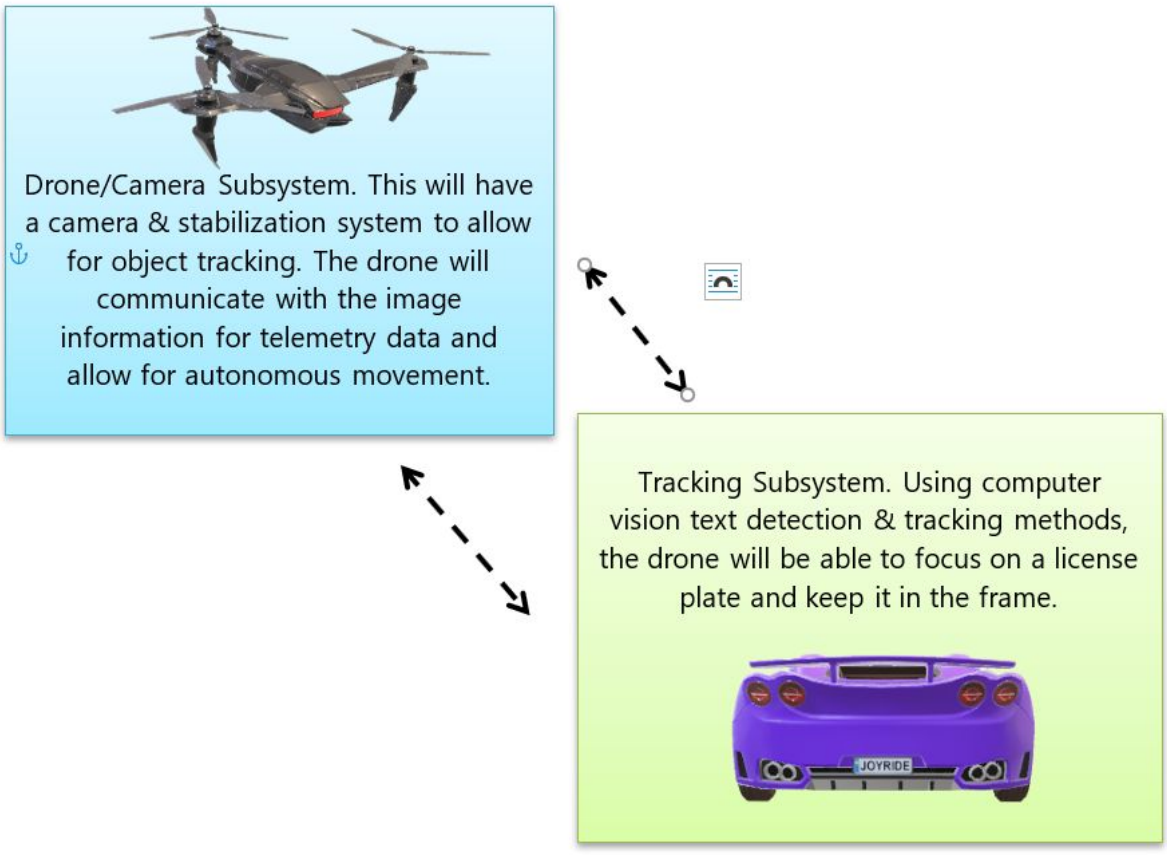

Figure 1: General Concept of Drone Follower

2. Introduction

The tracking of personnel is a widely used technique across both law enforcement and the military but can sometimes prove difficult in situations where line-of-sight to the target is lost in densely populated environments or there is no air support to describe the location of the target. The Drone Follower system hopes to provide an alternative and easy-to-use answer to this issue through an autonomous drone tracking system.Background

In law enforcement, it is estimated that around 35% of suspects get away during a police chase (1). Additionally, many police departments have a no-chase rule because of the danger to the general public, and some high-profile accidents resulting in death and lawsuits due to innocent citizens being hit by either the fleeing suspect or the officer.

In the military, recent events involving existing drones have led to political friction due to the loss of innocent life, and sometimes people cannot always be tracked through a crowd or dense urban environments (2). The Drone Follower sther party more options when deciding if a subject should be chased or tracked.

2.2 Overview

The Drone Follower system is a drone tracking system in which an autonomous drone can track a subject with license plate detection methods using Computer Vision. This system will be capable of tracking subjects in unpopulated and populated areas, inside vehicles, and confined spaces, which increases the safety of the general public as well as its users.

2.3 Referenced Documents and Standards

(1) Police Vehicle Pursuits, Dr. Brian A Reaves (2017) Bureau of Justice Statistics
(2) FAA Small Unmanned Aircraft Regulations (Part 107), FAA (2019)

3. Operating Concept

3.1 Scope

The drone is intended to operate independently of its user. It is expected to find an individual target and to follow it until the battery runs too low and the drone but return to launch. The drone will provide a live camera feed of the target and its precise GPS location. The drone will be operated outdoors during the day, in fair weather conditions that allow the drone to fly.

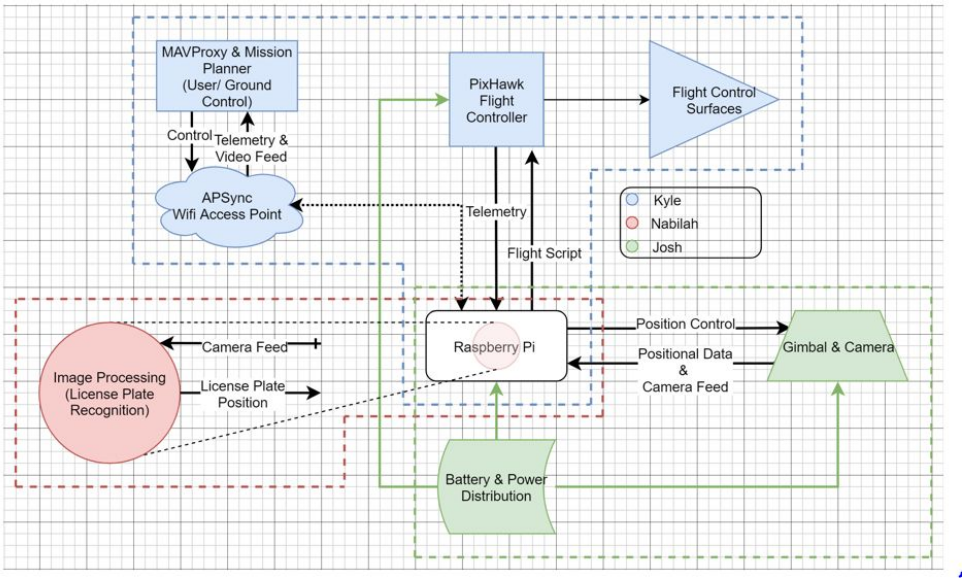

Figure 2: Drone Follower Overview

3.2 Operational Description and Constraints

The drone is intended to be able to track a target from the air. The user will set up the drone and give it the command to fly off and find the target and start tracking. The Computer Vision (CV) subsystem is responsible for providing the drone with image processing and a reliable target to track. Video will be taken directly from the CV program and processed in the Raspberry Pi that is attached to the drone to be converted into telemetry data. The Raspberry Pie and software subsystem will be responsible for maintaining visibility of the vehicle, as well as transforming the object's location in the frame into telemetry data for the Pixhawk flight controller. This telemetry data will be used to maintain a chasing flight pattern behind the tagged subject.

The drone can search and follow the target for as long as the battery can provide stable power to the drone and its subsystems. The drone needs to keep track of its location to keep it

from flying into restricted areas such as an airport. The weather conditions need to be fair for the drone to fly, such as low wind, no rail, no lightning, etc. In order to detect our target, we need fair lighting. Because of this, our system will work best during daylight, but further testing will be done during the night in low light conditions to further improve the image processing from our code (secondary goal). The vehicle must be traveling less than or equal to the drone's max speed, which for our test drone is around 64-80 kmh or otherwise risk losing the target vehicle. The drone must also follow the target from a respectable distance away, a proposed 15m, to allow for rotational corrections in the drone's movement to keep the license plate in view of the camera.

3.3 System Description

The proposed project will be divided into four sub-systems:

1 - Drone Flight Control System

This subsystem will manage the drone's autonomous flight system. We will be using a Raspberry Pi to provide an interface between custom control software and the Pixhawk flight controller in the provided drone subsystem. The Raspberry Pi 3 will be best suited for this application due to the existing documentation and API's provided for both on-board computers. The path of the drone will be controlled by GPS and Python script (DroneKit) in the on-board Raspberry Pi.

Figure 3: An image of the drone that we will be utilizing for this project.

2 - Camera Gimbal Stabilizer

There will be a digital camera mounted on a gimbal stabilizer to keep the target within view of the camera. The gimbal stabilizer is a two-axis system. There will be two brushless motors to control the pitch and the roll. The Yaw can be controlled by the drone flight system. So full three-axis control can be achieved by these two subsystems working together. The frame will need to be built lightweight to keep the work of the gimbal motors at a minimum. When the camera is mounted on the gimbal, it needs to be balanced to keep the work of the motors at a minimum. The gimbal will be controlled by a gimbal controller card.

3- Image/Frame Processing for Text detection and Tracking

Using OpenCV software, this subsystem will be able to both detect and track a vehicle using its license plate. Through Python, a real-time video feed will be the input the code. Once every 15 frames, the code will search the frame image for any text detection using a trained neural network. If text is found, it will then check to see it is a match with the text that the user desires. After finding the text desired, this text will be tracked using a bounding box in future frames. It's location in the frame will be fed to the Raspberry Pi's telemetry code for the Pixhawk for camera and drone movement.

3.4 Modes of Operations

Searching:

In the searching mode, the drone is released into the air where it will scan its surroundings using the Raspberry Pi camera. The drone will then either be given a command make a wide circle to find the target, or given a set location to find and locate the target. The user will input a license plate number, and the drone will track for partial and full matches.

The searching mode may also allow for manual control from the user at a ground location using traditional means, but this must be later tested to ensure correct mode switching from manual to guided during flight.

Follower:

In the follower mode, the drone has found its target and tries to keep the target within view of the camera. The drone stays above a set height and sticks as close to the target as possible, which allows for best contrast and quality in the image.

Standby/Debug:

In the standby mode, the drone will not be searching for anyone, but merely use its camera capabilities to surveil the area. This could allow for debugging processes and recalibration of the tracking system.

3.5 Users

The system is intended to be used in a law enforcement or military situations. The drone is to be used in the field by police officers and soldiers to find and locate targets tagged by the image processor. The first iteration limits this subsystem to vehicles, but may be further developed for the recognition of other targets.

3.6 Support

A user manual will be provided to provide information on how to use the system, so that the user knows how to operate the drone. A specification sheet detailing items such as mass, dimensions, max operating window, etc. A parts list detailing the objects used in our drone follower, such as motors, cameras, frames, etc. so that if parts were to break or malfunction, the user can replace said part.

Tech support will also be provided to the user. There are several modes of communication including telephone number, email, and postal. There will be people on call to help diagnose any problems and provide a solution. Users will be able to send in their malfunctioning drone and our we will restore the drone by replacing broken parts, updating firmware, and verifying full operation of drone sub-systems.

4. Scenario(s)

4.1 Tracking Individual in a Vehicle

If the police are in a committed vehicle chase against the person of interest, they can launch the drone in the air. The hand-held interface will connect to the drone’s on-board imaging system to view the surrounding area and track the vehicle in question via its license plate. With GPS and Computer Vision, the police will have an accurate location descriptor and visual lock-on with the target.

5. Analysis

5.1 Summary of Proposed Improvements

Our drone follower system offers a new perspective on vehicle tracking on a day-to-day basis. In comparison to autonomous drones that costs upwards of $8,000 with limited tracking capabilities, our drone will be significantly cheaper due to our use of a Raspberry Pi camera and a computer vision code.

Other GPS-tracking systems used by the police are large and do not offer a live-feed for the police to view the target. These systems offer only GPS transmittance with a significant time delay. This is a problem for many forces, as the GPS interface does not offer any visual on the target. Without this, the police have a higher chance of getting into collisions with civilians in a high-pressure chase since they do not have enough time to quickly meander through unforeseen traffic. The output video to the user will update live as well as track the object in question, allowing the police to view surrounding areas in comparison to the target. This can prevent potential unwanted collisions and provides enough time to gather as much information as needed on the vehicle in question.

5.2 Disadvantages and Limitations

(1) Our image processor is designed to track objects in the *frame*; due to this, if the target were to exit the frame, our tracker will need to recalibrate and search for text once again after the drone has moved towards the target's predicted location. The time in-between detection will be minimized to ensure that our drone can catch the target quickly, but this time may be precious in high-pressure scenarios.

(2) Our tracker will take in the object's location by analyzing the elements inside the bounding box iteratively. If another object were to *completely* block the target, this may cause issues for our tracker and our system will need to recalibrate. To ameliorate for this, the user will have to manually 'recalibrate', which will allow for a live change in the bounding box to track the target.

(3) The drone's max speed is approximately 64-80kmh which inhibits this system from tracking on high-speed highways. A more powerful drone may be used in the future to accommodate this drawback, but was not feasible at the time of project creation with the given budget.

(4) Rotational speed of the tracked vehicle must also be within a reasonable bounds. As mentioned previously, the license plate must be kept in the frame, and as such the drone must rotate behind the tracked vehicle when the vehicle turns. If this rotation is

done too quickly, the drone will lose sight of the license plate. These “reasonable bounds” will be defined later in project testing.

(5) Distance to target is calculated from the angle of the gimbal to parallel and the current height of the drone. Given that the following distance is proposed to be 50ft (15m), this angle may vary by up to 5 degrees. The height of the drone is also expected to be accurate to within 1-2m. Accordingly, the calculated distance will be accurate within ±2.5m, which may slightly impact the velocity calculation of the drone.

(6) The weather conditions must be favorable, including but not limited to winds not in excess of 15mph, no rain, sunlight, minimal oxidizing agents in the air to prevent long-term corrosion, and visibility must be greater than 15m.

(7) The distance of the drone to the ground station must be less than .8km, and therefore limits the accessibility of the drone for long-range missions. This can be subverted if a range extending antenna were connected to the raspberry pi, but will likely not be done for this project.

(8) The Raspberry Pi Camera has a wide and flat cable that is inflexible and delicate. In order for the gimbal to properly move around in 2-axises, a longer cable needs to be used. The cable is not meant to be repeatedly inserted and removed to the Pi camera, so the connection becomes very fragile if tampered with repeatedly.

5.3 Alternatives

There is currently only one alternative to our proposed drone follower that is in use by the law enforcement. The company StarChase developed a product that allows a GPS tracker to attach to a car during a high-speed pursuit. The tracker is shot through a system that is installed in the front bumper. From this, the officers can track the car to its location and arrest the criminal. However, there are some drawbacks to this system. Firstly, the person in question can easily hear the tracker attaching to the car and he or she can simply remove it without any difficulty. Second, there is no video feed for the user to see the criminal and there is a significant time delay. Our proposed system will have a dart that is small enough to be undetected by the driver. The Drone Follower also offers a live feed of the car and its surrounding traffic conditions; this notifies the police on how to approach the oncoming traffic and reduce the risk of unwanted accidents.

DRONE FOLLOWER

Nabilah Ahmed
Joshua Regresado
Kyle Henry

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION 2

04 December 2019

FUNCTIONAL SYSTEM REQUIREMENTS

for

Drone Follower

Team <49>

Approved by:

_____

Project Leader Date

_____

John Lusher II, P.E. Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	9/16/19	All Members		1st Pass
2	12/4/19	All Members		2nd Pass (Final)

Table of Contents

List of Figures

LIST OF TABLES

1. Introduction

1.1 Purpose and Scope

This specification defines the technical requirements for our Drone Follower. Figure 1 shows a representative integration of the project in the proposed CONOPS. The verification requirements for the project are contained in a separate Verification and Validation Plan.

The Drone Follower system shall allow a drone to track an object of differing size and speed. The tracking will be done using computer vision, which will then be translated into telemetry for the drone's flight controller via software in an on-board Raspberry Pi. The tracking system will utilize text detection to identify the desired license plate number, which can then be autonomously trailed by the drone. This drone follower system has a variety of uses but will be designed with law enforcement and the military in mind for tracking of vehicles.

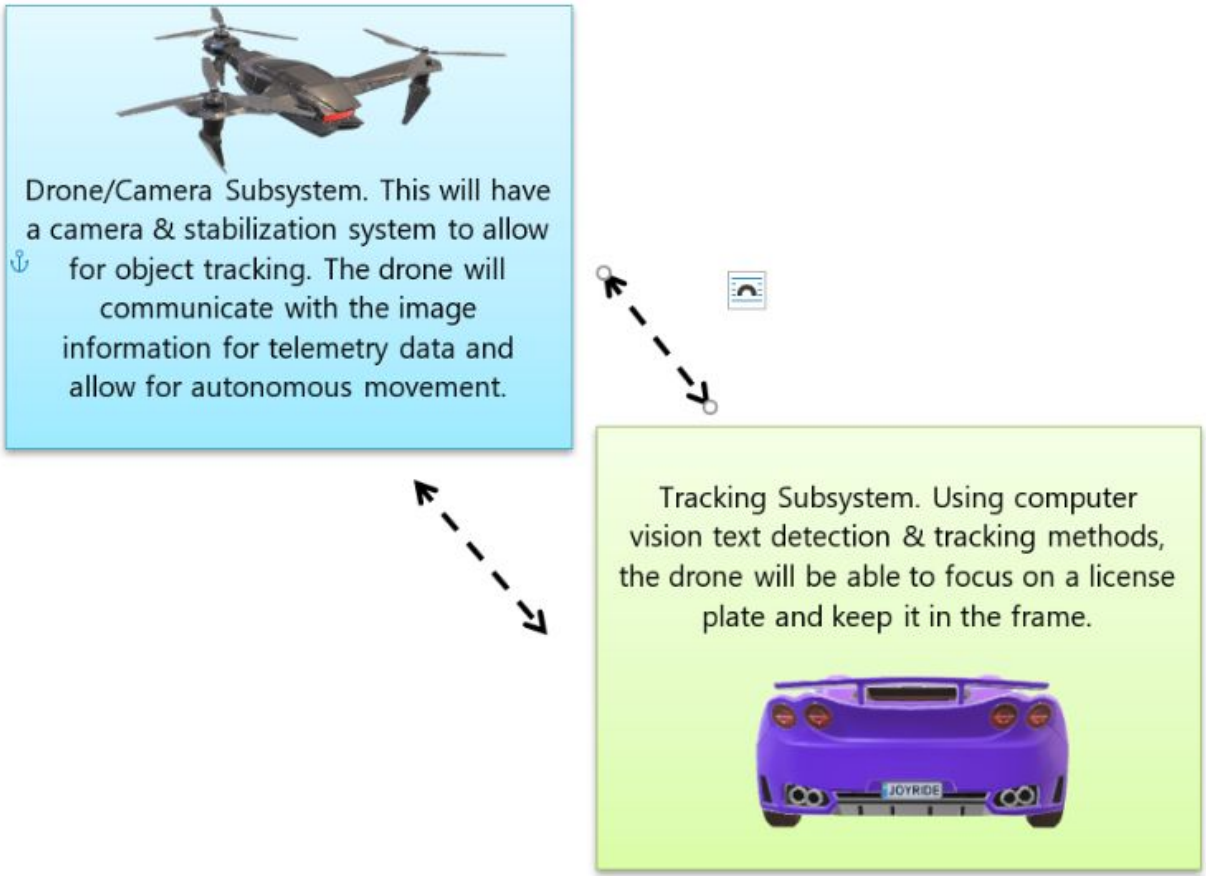

Figure 1: Project Conceptual Image

1.2 Responsibility and Change Authority

Any changes to the specifications or deliverables provided must be approved by the sponsor, Kevin Nowka. The responsibility of the completion of the subsystems is distributed as such:

Kyle Henry- Responsible for the programming and integration of the drone subsystems within the Raspberry Pi, and the autonomous flight control script.

Nabilah Ahmed- Responsible for license plate detection and tracking of target.

Joshua Regresado- Responsible for integration of the camera and camera gimbal systems on the drone, and any additional hardware/ electrical systems.

2. Applicable and Reference Documents

2.1 Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
NFPA 70	2014	National Electric Code
AN-190 ICAO Cir 328	2011	Unmanned Aircraft Systems
802.11	2/9/2011	Wireless Network Management
ANSI C119.6	2011	American National Standard for Electric Connectors
IPC A-610E	Revision E – 4/1/2010	Acceptability of Electronic Assemblies
IEEE 802.11ac	12/11/2013	Specification Driven by Evolving Market Need for Higher, Multi-User Throughput in Wireless LANs

Table 1: Applicable Documents

2.2 Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
FAA 107	2018	Small Unmanned Aircraft Regulations

Table 2: Reference Documents

2.3 Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as "applicable" in this specification are incorporated as cited. All documents that are referred to within an applicable document are considered to be for guidance and information only, with the exception of ICDs that have their applicable documents considered to be incorporated as cited.

3. Requirements

3.1 System Definition

The drone is intended to operate independently of its user. It is expected to find an individual target and to follow it until the battery runs out. The drone can follow a vehicle at speed. The drone will provide a live camera feed of the target and its precise GPS location. The drone will be operated outdoors during the day, in fair weather conditions that allow the drone to fly.

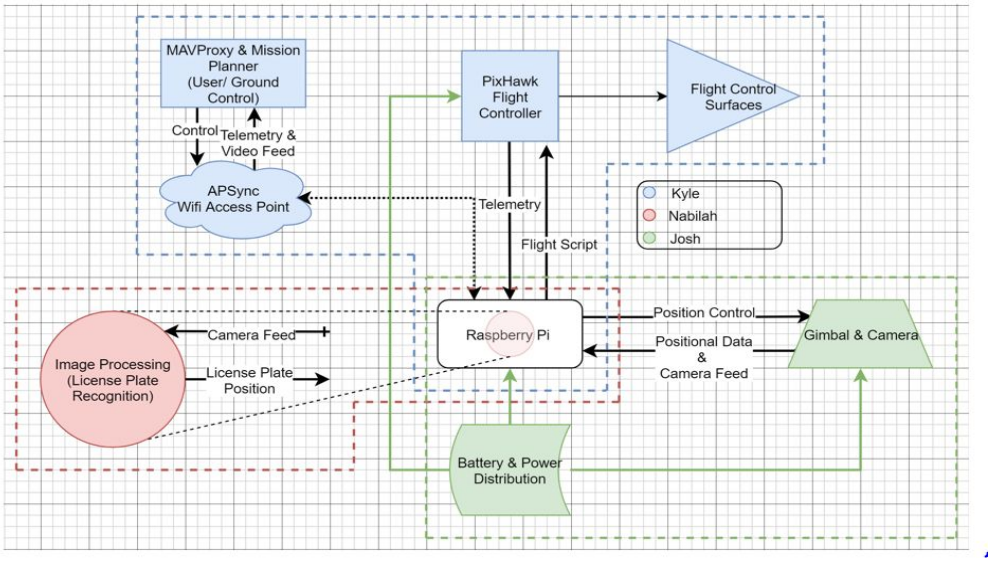

Figure 2: Subsystem Overview

The drone is intended to be able to track a target from the air. The user will set up the drone and give it the command to fly off and find the target and start tracking. The Computer Vision (CV) subsystem is responsible for providing the drone with image processing and a reliable target to track. Video will be taken directly from the CV program and processed in the Raspberry Pi that is attached to the drone to be converted into telemetry data. The Raspberry Pie and software subsystem will be responsible for maintaining visibility of the vehicle, as well as transforming the object's location in the frame into telemetry data for the Pixhawk flight controller. This telemetry data will be used to maintain a chasing flight pattern behind the tagged subject.

The drone can search and follow the target for as long as the battery can provide stable power to the drone and its subsystems. The drone needs to keep track of its location to keep it from flying into restricted areas such as an airport. The weather conditions need to be fair for the drone to fly, such as low wind, no rail, no lightning, etc. In order to detect our target, we need fair lighting. Because of this, our system will work best during daylight, but further testing will be done during the night to further improve the image processing from our code. The vehicle must be traveling less than or equal to the drone's max speed, which for our test drone is around 64-80kmh or otherwise risk losing the target vehicle. The drone must also follow the target from a respectable distance away, a proposed 15m, to allow for rotational corrections in the drone's movement to keep the license plate in view of the camera.

3.2 Characteristics

3.2.1 Functional / Performance Requirements

3.2.1.1 Utilizing "MOSSE" tracking methods in openCV

The drone follower will use license plate image recognition using openCV tracking methods, "MOSSE" to lock onto a user specified licence plate number, which it will then follow using an-board Raspberry Pi running a DroneKit python script.

Rationale: MOSSE tracking offers the fastest image processing for 1080p. At around 20 fps, this tracking method will allow for efficient tracking.

3.2.1.2 Text Detection for license

The drone follower will use a pre-trained neural network "Frozen east Text Detection" in order to track license plates.

Rationale: This pre-trained neural network detects capital letters and numbers, and these characters are common to license plates.

3.2.1.3 Utilizing Python to communicate with controllers

The automation will be done using the Pixhawk flight controller and an on-board Raspberry Pi 3, which will be running a python script written using the DroneKit API. Using frame localization data provided by the image recognition subsystem, as well as vertical and horizontal tilt from the gimbal, the Python script will allow the drone to be completely autonomous via velocity and rotation control.

> *Rational: DroneKit provides all of the necessary control commands needed for the Pixhawk. Therefore, an on-board computer was needed to run DroneKit Python along side image recognition software, which a Raspberry Pi does well. Distance to the target is needed to specify velocity and maneuvering adjustments, and since an accurate distance measuring device of low weight is not readily available within the budget, distance calculation will be done instead using readily available values.*

3.2.1.4 MavProxy for GPS Transmittance & Communication

The Raspberry Pi can then send telemetry information and GPS coordinates of the drone back to a user ground-station using MAVProxy, which will be running mission planner on a computer, as well as a companion debug console application for script output. The user can alternatively use QGroundControl on a mobile device for mobile access to this telemetry data. According to specifications for the Raspberry Pi, the distance of Raspberry Pi communication can be up to 0.8km. This also assumes that there are GPS satellites available for use by the GPS mount.

> *Rational: MAVProxy automatically provides all of the information within the data packets necessary to relay the current GPS position of the drone, as well as altitude, velocity, and mission path. QGroundControl is an extension of MAVProxy that ships with a mobile way to view this data.*

3.2.1.5 Gimbal

The gimbal will also be controlled through the Pixhawk flight controller and Raspberry Pi. The gimbal can be automated to be controlled through DroneKit scripts that are stored on the Raspberry Pi and then relayed to the Pixhawk which sends PWM signals to the gimbal controller board which then interprets those PWM signals to control the brushless motors for the 2-axis gimbal. The DroneKit script will take input from the image recognition in order to keep the license plate within the frame of the camera and within view for the software.

3.2.2 Physical Characteristics

3.2.2.1 Mass Capacity of Drone

The total weight of our drone must be less than 7200 grams.

> *Rationale: Our drone has 6 DJI 2212 (920KV) motors that have a thrust capacity of 1200 grams per motor in order to function as required.*

3.2.2.2 Mounting

The detailed mounting information for the Drone Follower shall be captured in the ICD. The controllers shall be mounted in the center- base of our drone.

Rationale: In order to maintain center of gravity for our drone and prevent tilting of the drone while in the air, the controllers will be mounted on the center. Please refer to ICD for more information.

3.2.3 Electrical Characteristics

3.2.3.1 External Commands

External commands must be sent using the MAVProxy protocol to the Raspberry Pi WiFi receiver. These commands can then be sent to the Pixhawk via the Telem 1 Port spliced to the serial pins on the Raspberry Pi. These commands include mode control, go-to commands, and direct DroneKit functions. The range of the WiFi receiver is specified as 0.8km, assuming there are no obstructions between the user's PC and the drone such as tunnels, large buildings, hills or mountains.

Rational: MAVProxy provides the correct commands and protocols the Pixhawk expects, such as the correct command update frequency and built-in DroneKit commands, while also being able to interface with a Raspberry Pi.

3.2.3.2 Input Voltage Level

The Raspberry Pi will require a +5V input voltage and the Pixhawk will require a +5V input voltage. The gimbal requires a +5V input voltage as well to work. All electronics, save for the motors and ESCs, require a step down voltage because the battery provides 22.2V, which is much higher than needed for them to be powered. For the 6 motors, and ESCs, they will need a 22.2 V connection from the 6s battery.

Rationale: This is the standard input voltage for our components. We will be using a 22.2 V battery for design simplicity (provided).

3.2.3.2 Input Video Feed (Camera)

The camera for image capture and processing is a 1080p 30FPS Raspberry Pi camera. This image data will be processed in the Raspberry Pi via the image processing subsystem. The location of the target in the frame will be used as part of the telemetry data for drone/gimbal movement.

Rationale: The user interface will be 1080p for the video output, and the distance required from the drone to the target to ensure proper frame processing will be maximum 40 feet and 48 degrees. For cost-efficiency, this camera satisfies our requirements and can be easily attached to the Raspberry Pi. Further testing will be done with drone flight to ensure accuracy.

3.2.3.3 Power Consumption

The power consumption of the drone is 177.6 Wh, which means the drone system if left on for an hour will consume 177.6 Watts of energy. This includes all electronics, motors, controllers, antennas, etc.

Rationale: The battery provides the information on the packaging of how much power it can use. Because the drone system will be using the full capacity of the drone, the power rating was used to estimate the power consumption of the drone

3.2.4 Outputs

3.2.4.1 GPS Output

The telemetry data will be sent back to the user using the Raspberry Pi WiFi transmitter using the MAVProxy protocol; the inverse is true for the user as well.

Rationale: MAVProxy provides all of the necessary information within the data packets automatically for this information to be sent back to the user, and will already be used for communication.

3.2.4.2 System Data Output

The video feed and its containing data of interest (object location in frame, GPS location) will be fed back to the user through Mission Planner using MAVProxy.

Rationale: The presentation of the data will be easily digestible through Mission Planner for the user. A GUI may be a secondary goal if necessary.

3.2.4.3 Connectors

The connectors for our Drone shall follow the American National Standard for Electrical Connectors ANSI C119.6-2011.

Rationale: Standard for Connectors.

3.2.4.3 Wiring

The wiring for our Drone that follow the AN-190 ICAO standard for Aerial Systems.

Rationale: Standard for unmanned aircraft systems.

3.2.5 Environmental Requirements

3.2.5.1 Altitude

The WiFi connection to drone is the most limiting factor in this regard, only allowing for up to 0.8km in distance from the ground station.

Rational: Since atmospheric pressure does not change abruptly, distance to ground station is the greatest concern here. In further developments where WiFi range is not an issue, this would be limited by the rotor dimensions and motor lift in certain atmospheric conditions.

3.2.5.2 Thermal

The motors may operate within 0-50C, the battery may operate within 0-35C, and the Raspberry Pi may operate within 0-82C. Therefore, the ambient temperature must at least be within these bounds assuming perfect venting conditions.

Rational: Taken directly from the specification sheets.

3.2.5.3 Rain

This drone shall NOT operate in rainy weather conditions.

Rationale: Due to the various electronics mounted on the drone, this drone will be at risk of damage if flown during rainy weather conditions.

3.2.6 Thermal Requirements

3.2.6.1 Raspberry Pi Cooling

Heat sinks were placed on the CPU and RAM, and a fan was installed on the Pi casing for passive temperature control.

Rational: The Raspberry Pi must be kept under 84C to prevent thermal throttling. This was the most effective way to allow for higher ambient temperatures and prevent overheating while running the specified programs.

3.2.6.2 Cooling of Other Hardware

The remaining hardware shall be passively cooled. No modifications are necessary in this regard given that the temperature ranges specified in the FSR are followed.

Rational: All other hardware on the drone is designed to be passively cooled without modifications. The only concern for overheating is the LiPo battery, which may overheat in direct sunlight or high (>35C) ambient temperatures.

4. Support Requirements

A laptop or PC will be required to interface with the drone follower. For on-site diagnostics, a USB port will be required for Raspberry Pi interfacing, or alternatively a means of using SSH to the Raspberry Pi.

Included in the Drone Follower package is the drone with associated subsystems, a USB to micro-USB cable, and a charging cable.

If there are issues with the Drone Follower, a support technician will be available on call to diagnose the issue over the phone. If the problem were not quickly resolved, the technician will come to the site and diagnose the issue.

Appendix A: Acronyms and Abbreviations

GPS: Global Positioning System
ESC: Electronic Speed Controller
TBD: To Be Determined
FPS: Frames per second
RP: Raspberry Pi
PC: Personal Computer
USB: Universal Serial Bus
p: progressive scan
m: meter(s)
kmh: kilometers per hour
km: kilometers
API: Application Programming Interface

DRONE FOLLOWER

Nabilah Ahmed
Joshua Regresado
Kyle Henry

Interface Control Document

REVISION 2

04 December 2019

Interface Control Document

for

Drone Follower

Team <49>

Approved by:

_____

Project Leader Date

_____

John Lusher II, P.E. Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	9/16/19	All Members		1st pass
2	12/4/19	All Members		2nd pass (Midterm)

Table of Contents

List of Tables

List of Figures

1. Overview

This document will further discuss subsystem interactions in our drone follower as described in the Functional Systems Requirement and Concepts of Operations reports. These subsystems include the Raspberry Pi integration of the drone flight control subsystem, the detection/tracking subsystem, and the gimbaling subsystem. The ICD will include physical descriptions of the subsystems.

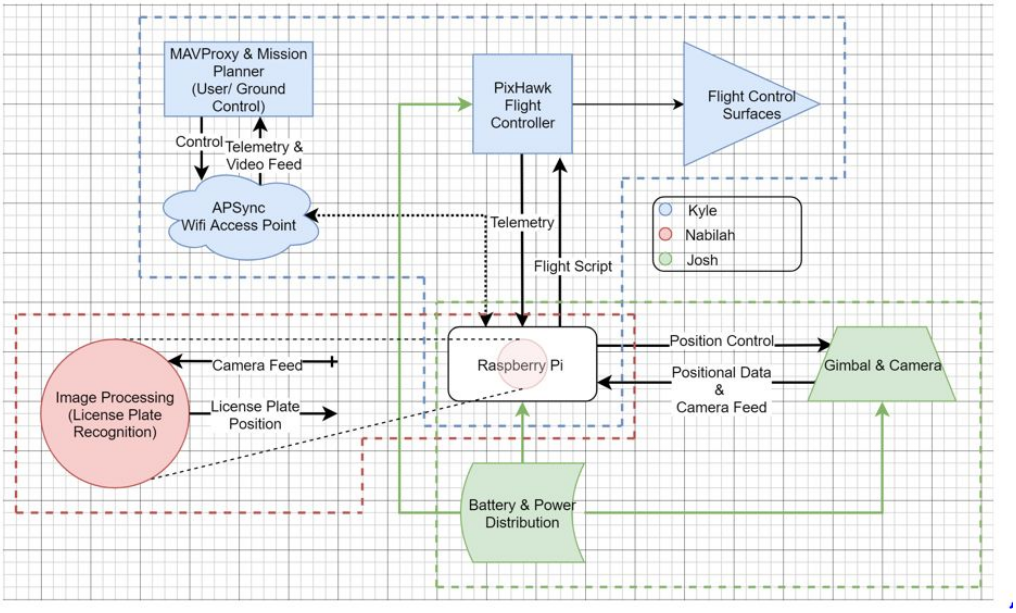

Figure 1: Subsystem Overview

2. References and Definitions

2.1 References

(1) Raspberry Pi Documentation- https://www.raspberrypi.org/documentation/
(2) Open CV Documentaion- https://docs.opencv.org/
(3) FAA 107. Small Unmanned Aircraft Regulations. 1 Jan 2018.
(4) IPC A-610E. Acceptability of Electronic Assemblies. Revision E - 1 April 2018.

2.2 Definitions

- GPS: Global Positioning System
- FPS: Frames Per Second
- CV: Computer Vision
- RP: Raspberry Pi
- CPU: Central Processing Unit
- USB: Universal Serial Bus
- UART: Universal Asynchronous Receiver/Transmitter
- DSLR: Digital Single-lens Reflex

3. Physical Interface

3.1 Physical Weight

Item on Drone	**Weight (grams)**
Drone Body	2000
Gimbal	230
PixHawk Controller	39
Raspberry Pi 3 w/ heatsink	45
DJI Motors (Thrusters)	53 x 6
Raspberry Pi camera	3*
Total	2615 grams (Around 5 lbs.)

Table 1: Weight Specifications

**Please note that due to subsystem changes during the second half of the term, the camera was changed to from a 1080p DSLR camera to a Raspberry Pi 1080p camera . To reduce purchasing costs, we will attempt to integrate our gimbal with this camera. For more information of this subsystem change, please refer to the subsystem report regarding license plate tracking.*

3.2 Dimensions

Item	**L x W x H**
Drone Body (motor thrusters attached)	**Diagonal Wheelbase 550mm (LWH N/A)*
Gimbal	70mm x 70mm x 80mm
Raspberry Pi Camera	*25mm x 23mm x 9mm*
Raspberry Pi	*11 mm x 56.5 mm x 85.6 mm*
Total after attachment:	Diagonal Wheelbase= 550 mm

Table 2: Dimension specifications

**Diagonal Wheelbase* is the specification used to describe drone dimensions. It is the diagonal distance from motor mount to opposite motor mount.

3.3 Mounting Locations

The Pixhawk Controller, Battery,and wiring will be mounted and stored in the central storage bay of the drone. The Raspberry Pi, Radio Receiver, and GPS will be mounted atop the central drone body. The Gimbal will be mounted on the front facing side of the drone on the mounting holes. The motors will be mounted on each of the motor mount locations on each of the six arms of the drone.

4. Thermal Interface

4.1 Raspberry Pi Cooling

The Raspberry Pi must be kept under 84C to prevent thermal throttling. Therefore, heat sinks were placed on the CPU and RAM, and a fan was installed on the Pi casing for passive temperature control.

4.2 Cooling of Other Drone Hardware

The remaining hardware shall be passively cooled. No modifications are necessary in this regard given that the temperature ranges specified in the FSR are followed.

5. Electrical Interface

5.1 Primary Input Power

5.1.1 Raspberry Pi

The Raspberry Pi requires a constant 5V input voltage via USB (regulated). The input current is 2A. The Pixhawk will receive its power from the Raspberry Pi as well as the raspberry Pi.

Component connected to Raspberry Pi	Power
Pixhawk Controller	+4.7-+5.0V, 2A max
Raspberry Pi	+4.9-5.1V, 2A max

Table 3: Power Specifications

5.1.1 Battery

The battery will run off of one 6s, 22.2 V, 177.6 Wh ,8000mAh battery provided. All drone components and subsystems will be powered by the provided battery.

5.2 Signal Interface

5.2.1 Raspberry Pi Camera

The video feed from the camera will be fed into the python CV script for image processing and text detection. The script will display the location of the target in the frame and send this through the Raspberry Pi for further processing of telemetry data.

5.2.2 Gimbal

A MAVLink python script stored on the Raspberry Pi will interface with the Pixhawk flight controller and the flight controller will convert the commands from the script into PWM signals that it sends to the gimbal controller board. The gimbal controller board interprets those PWM signals into gimbal stabilization software to be sends its own PWM signals to the motors which move to stabilize the camera.

5.3 User Control Interface

The only interface with the user will be via a direct connection to the laptop or phone running Mission Planner or QGroundControl respectively. GPS data and video data will be outputted to the user as well as the object being tracked (with a bounding box to clearly see location in frame). The user will have the opportunity to enter a license-plate number it would like to track, and the output (after a max 10 frame delay) will track the object.

6. Communications Protocols

6.1 MAVLink Protocol

All communication between the drone and the ground station will be done using the MAVLink protocol over WiFi using the MAVProxy hotspot setup on the Raspberry Pi. Documentation of this protocol can be found on https://mavlink.io/en/.

6.2 WiFi Communication

The ground station and Raspberry Pi will communicate over a WiFi network from the WiFi hotspot provided by the Raspberry Pi. This communication follows the protocol described in the IEEE 802.11 standard.

6.3 Raspberry Pi & Pixhawk Communication

The Raspberry Pi communicates with the PixHawk via UART serial connections on the TX and RX pins connected to the inverted TX and RX pins on the PixHawk. The UART protocol is defined in Arm PrimeCell ® UART(PL011) Technical Reference Manual Revision r1p5.

6.4 Video Interface

The camera will be attached via a small socket on the Raspberry Pi board on the upper surface and will use a dedicated CSi interface and ribbon cable.

DRONE FOLLOWER

Nabilah Ahmed
Joshua Regresado
Kyle Henry

SUBSYSTEM REPORTS

REVISION 1

04 December 2019

SUBSYSTEM REPORTS

for

Drone Follower

Team <49>

Approved by:

_____

Project Leader Date

_____

John Lusher II, P.E. Date

_____

T/A Date

CHANGE RECORD

Rev.	Date	Originator	Approvals	Description
1	12/4/19	All Members		Creation

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The Drone Follower will be capable of autonomously following a target license plate within the environment. The system does this via image recognition of said license plate using an image recognition algorithm in the on-board the Raspberry Pi to locate a license plate in the video feed. This location is then sent to the flight control subsystem that controls the telemetry of the drone itself. Telemetry information can then be sent back to the user at ground-based location. The system is broken down into the image recognition, hardware, and flight control subsystems.

2. Flight Control Subsystem Report

2.1 Subsystem Introduction

The purpose of this subsystem is to provide flight automation and flight data to the user. The subsystem has two major components. The first is the *automation script* written in DroneKit Python, which takes in locational data from the image recognition subsystem and gimbal tilt data from the hardware subsystem, and provides the drone with appropriate commands to keep it on course towards the target. The second is the *drone communications* which relies on a WiFi connection point from MAVProxy and the MAVLink protocol to send telemetry data back to the user, as well as receive basic commands from the ground station. The Raspberry Pi is the main point of communication and processing, and as such communicates with the PixHawk via serial UART connections on the RPi pins.

2.2 Subsystem Details

A block diagram of the subsystem is shown below.

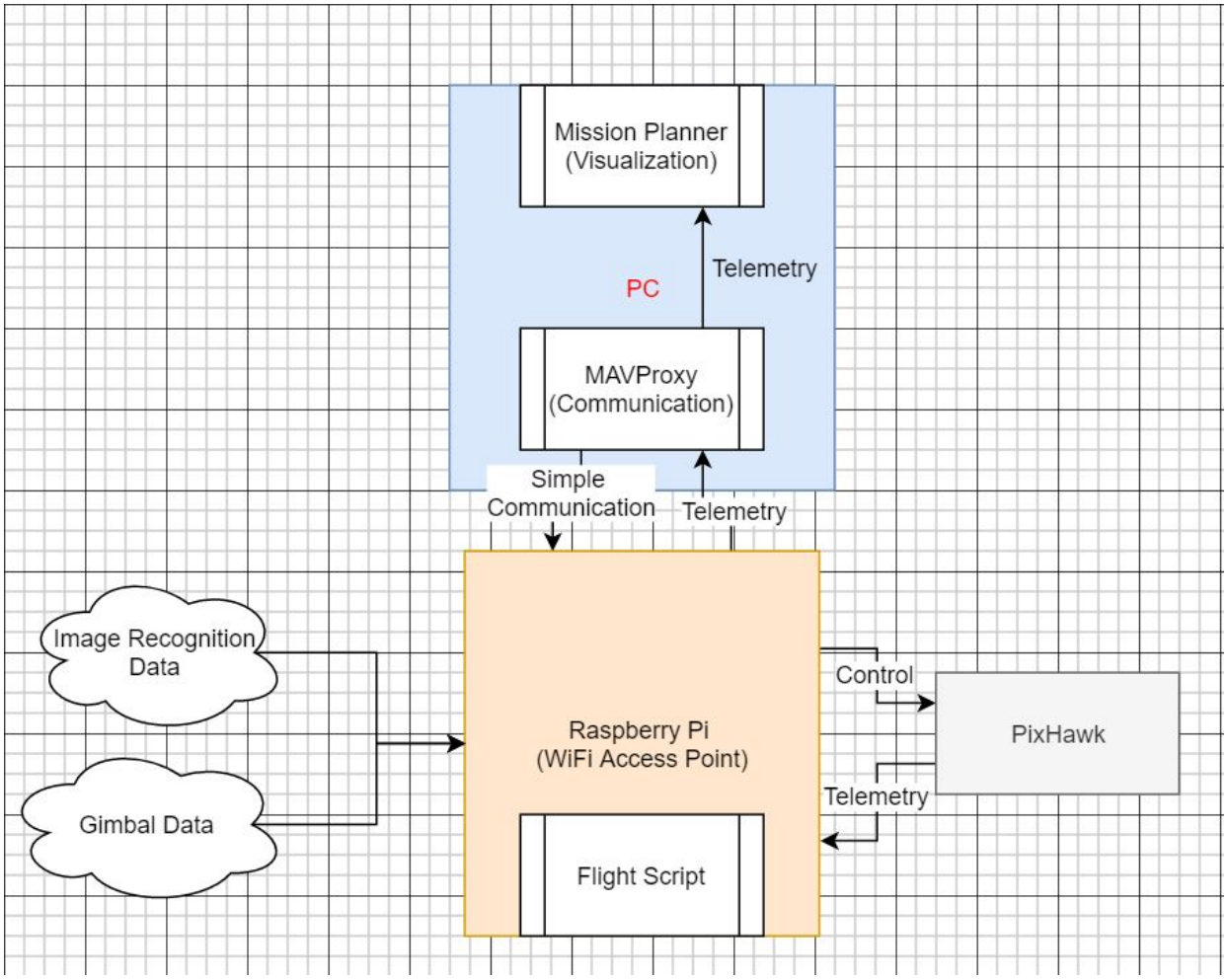

Figure 1: Block Diagram of Flight Control Subsystem

The primary challenge of this subsystem was ensuring all of the components worked together in consort. Most of the protocols and communications had been done previously, which resulted in there being ample documentation towards this goal. A secondary, less difficult task, was writing an automation script that was capable of demonstrating the autonomous capabilities of the drone.

2.3 Subsystem Validation

Since the drone is not yet capable of being flown to test my subsystem without the integration of the other subsystems, I decided to go about my validation in two steps.

First, I tested all of the hardware for the done and the communication protocols. APSync was flashed to the Raspberry Pi, and the serial connection was setup to communicate with the PixHawk. A simple script which preps the PixHawk for launch was then installed on the

Raspberry Pi and run via SSH to test the communication. This communication test was successful when the PixHawk responded to this command by attempting to ready motors. A more complex script would have been preferred, but the PixHawk will not do anything besides launch prep when disconnected from the motors.

The second test involved testing an automation script in a simulated environment called DroneKit SITL. This environment was explicitly designed to test drone automation scripts, with the only difference being the remote communication, and the visualization software being QGroundControl rather than Mission Planner. A block diagram of this test environment is shown below.

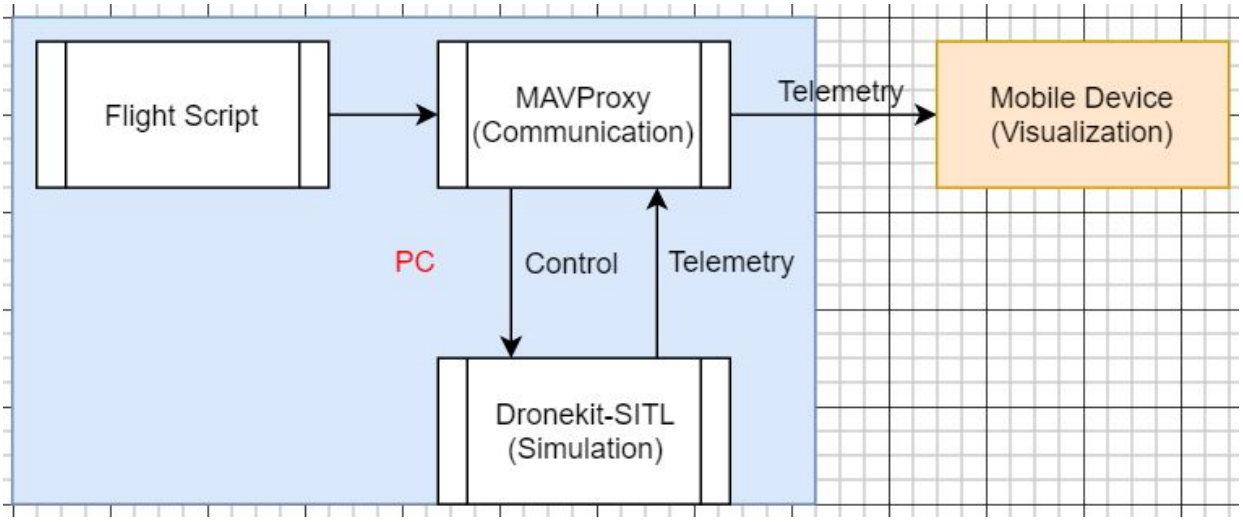

Figure 2: Flight Control Test Environment

This works by using the loopback address of the PC running DroneKit SITL in MAVProxy instead of the Raspberry Pi (connected to the PixHawk). DroneKit SITL simulates the drone's movements and response to commands just as the PixHawk would. I was then able to test a simple automation script, which had the drone perform preflight-checks, take off, control the velocity and pitch of the drone, and the return to launch location using GPS. A drone velocity of up to 40kmh was tested in this simulation.

The only issue I ran into while testing the subsystem was with the simulated altimeter. QGroundControl would lag behind the simulation, resulting negative values when landing and subsequently hitting the virtual ground. Since this is only a simulation, I believe this to be due to lag in the simulated environment, but should still remain a consideration when real-life testing begins. The following chart shows these results.

Test:	Delay (in meters)
1	6.1
2	5.4
3	6.6
4	6.7
5	7.1
6	6.4
7	5.1
8	6.3
9	6.8
10	5.9
AVG:	6.210194905

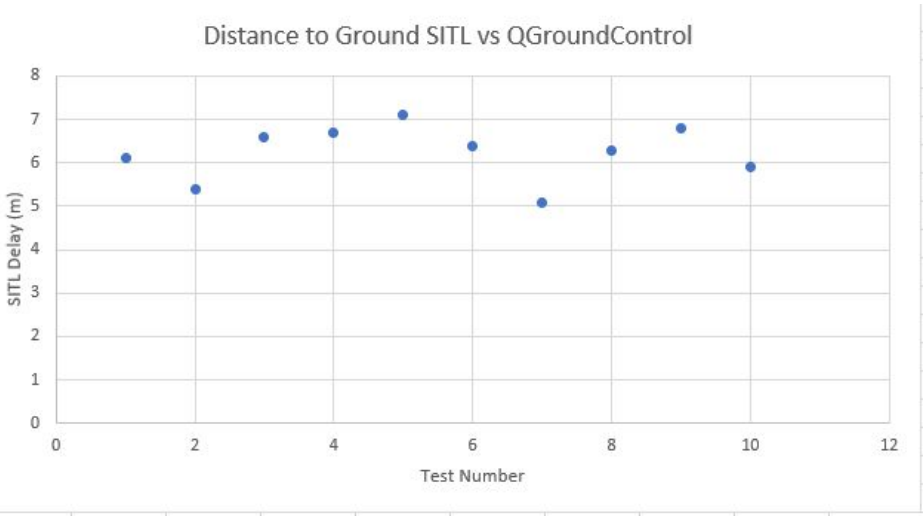

Figure 3: Distance to Ground SITL vs QGroundControl

2.4 Subsystem Conclusion

The communication part of this subsystem was tested by setting up all of the hardware identically to the way it will be setup on the drone. A simple test script was then run to the PixHawk to test for response. The autonomous flight part of the subsystem was tested using DroneKit SITL to pass a DroneKit Python script to the simulated done, which tested takeoff/ landing, velocity, and rotational control. Further development will be required of this script, as currently it does not take into consideration the other subsystems.

3. License Plate Recognition/Tracking Subsystem

3.1 Subsystem Introduction

The purpose of this subsystem is to process the input video feed frame-by-frame in order to detect and track license plates. This subsystem using OpenCV (computer vision) functions available in Python for functionality and communication with the rest of the system. Once the license plate is found in the frame, it will return its location back to the Raspberry Pi for telemetry data. The flight control system/Gimbal will then work to center the target in the frame.

3.2 Original Subsystem

It is important to note that around 10/28/19, there was a major subsystem change. This subsystem originally utilized infrared imaging and a dart/emitter system. A dart would be

attached to the vehicle that emitted infrared light. This light would then be caught by our infrared camera and tracked using a PIXYcam.

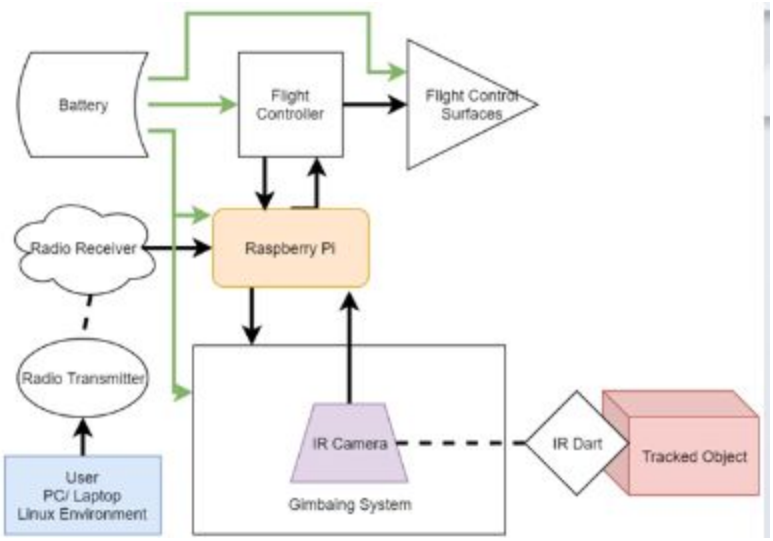

Figure 4: Original Block Diagram for the system

3.2.1 IR Dart/Emitter system

The target of the drone would initially need to have an infrared LED attached to the individual for our camera sensor to start tracking. The emitter will be in the shape of cylinder that is angled upwards at a 40 degree angle for easy detection by the drone. The initial sketch is displayed below:

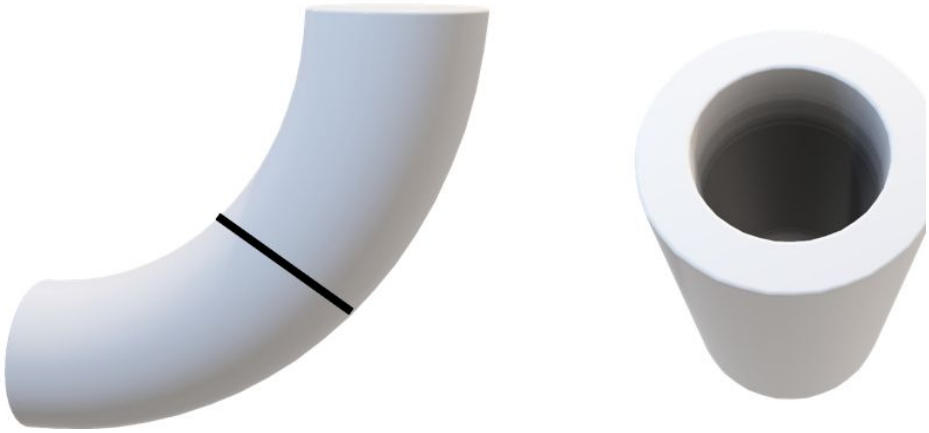

Figure 5: Cylinder Concept Diagram

The diameter of the 3D-printed cylinder during preliminary research was to be around 60 mm. At the black marker in Figure 2, the circuit board for the IR LEDs were to be placed. This cylinder would have an adhesive that could stick to vehicles. The IR LEDs to be used would

emit at a frequency other than 60 Hz in order to avoid confusing the targeting system with the power grid of the city.

Figure 6: IR LED at 940 nm.

These IR LEDs would work emit at near infrared and would be connected together in parallel in order to emit at the highest intensity. These IR LEDs would be connected together on a plate. The plate would comprise of 48 LEDs using a 12 VDC power source. The projection angle of the plate is 90 degrees and plate had an external diameter of 60 mm. Initial research indicated that one plate would be enough for this subsystem. The calculated detection distance with the angle was around 15 meters, however the intensity would drastically decrease (quadratically) per meter. At 15 meters, the intensity would be around 40 nW. This plate would have been powered by a battery placed inside the cylinder, and because the cylinder was hollow, there was minimal indication for heating issues.

Figure 7: Example of single IR LED plate working under infrared filter.

3.2.2. IR Camera detection

In order to detection our infrared emitter, the PIXYcam was going to be used. The PIXYcam is an image and video sensor that can track objects in visible light through training. The open source application PixyMon provided Raspberry Pi connection documentation. To allow for infrared tracking, the lens would be converted to an IR lens. Though an IR filter could be used, this would have affected exposure times and therefore limited our live tracking. PixyMon software would have been manipulated via python in order to automatically detect our IR LED. Once tracked using the PixyCam, the location of the plate would then be transferred to our video interface using a Raspberry Pi camera. This would essentially track our object and our

user would have a high quality video feed. For faster image processing, the PixyCam had a 144p output.

Figure 8: PixyCam with Infrared lens

To initially test the cam, a single IR LED was used (emitting at required wavelength but with 20 nW intensity) to understand the detection capabilities of the camera. By tweaking the exposure of the camera, the perceived ‘brightness’ of the LED was heightened. Lowering the exposure proved optimal.

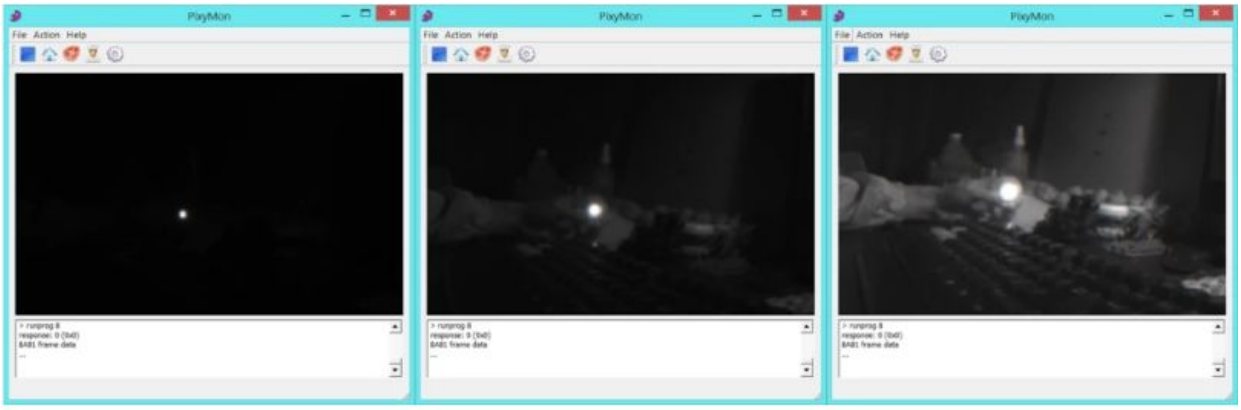

Figure 9: An example of the IR LED appearing bright at lower exposures

3.3 Subsystem Change

There were several issues with this subsystem. The main issue was the emitter design. Later research showed that if we were to detect our emitter at a max range of 15 feet, we would need 5 of these IR LED plates together. This severely affected the cylinder dimensions, as now it

would have to 300 mm in diameter, and each angle of the plates would have to uniquely situated in order to emit in the optimal angle for the Pixycam to detect. Another major issue was the angle of detection. Further research showed that a large object (such as a truck) with reflective surfaces could simply interfere with the camera's view of the emitter and our signal could be lost. This was a major problem as densely populated regions could have trucks. Compared to the alternatives mentioned in the Conops, this solution would be impractical.

3.3.1 New Subsystem

As described in the CONOPS, FSR,and ICD the new subsystem will use computer vision in order to detect the license plate of an image. The tracker could intuitively track the object even if an object is in front of it. Because of the subsystem change late in the semester, this term was used to ensure that the computer vision used could detect and track a license plate.

3.4 Subsystem Validation and Progress

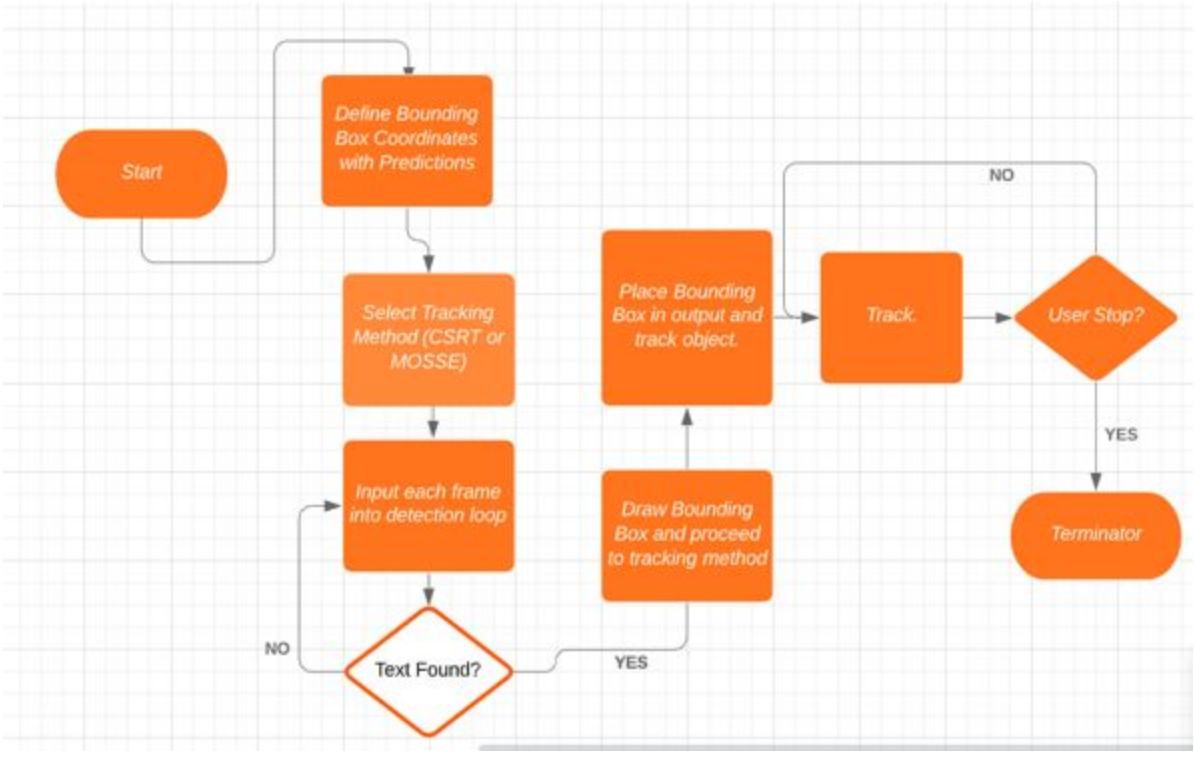

Figure 10: General code flowchart

The code will first initialize the bounding box. Then it will select a tracking method (CSRT or MOSSE) in order to track the object. On the first frame, the code will search the image for a license plate. If the text was detected it would then draw the bounding box around the object and output the video with the box around the text. It will continuously track the object by

utilizing a pre-trained network for text detection (frozen east text detection). Due to timing limitations, we could not train our own network. This would be a goal for ECEN 404. For faster performance, the code would loop every 15 frames.

Figure 11: License Plate Recognition

Output video at 1080p showed that our text was detected with MOSSE and CSRT. Results are depicted below:

Tracker	**FPS (average)**
MOSSE	10
CSRT	7

Table 1: Different trackers and the FPS output

The code will automatically select MOSSE tracking unless the user specifically requests CSRT.

License Plate	**Confidence**
AZM9590	88%
MLK5690	72%
BWE4569	89%

Table 1: Different plates and their confidence

The text detection was satisfactory, and tracking was more than satisfactory. The confidence level indicates the perceived accuracy of the code to its license plate.

3.3.1 Conclusion

Preliminary testing shows that our tracker/detector functions. Future additions to this subsystem will include:

- Minimizing the bounding box *only* to the text
- Displaying the location of the plate in the frame
- Rasbian integration
- Testing with the gimbal and Drone flight system

Validation proved that the code's primary directive is functioning properly and future changes will only require minor tweaking to the code.

4. Gimbal Subsystem Report

4.1 Subsystem Introduction

The purpose of this subsystem is to move the camera around on two axises, pitch and roll, in order to track a target and keep it within view for the image recognition software. The gimbal controller is a dedicated board that is able to take PWM signal input as either from the radio or from the Pixhawk flight controller. For the final subsystem I intend to give have an automated mode that sends signals from the flight controller which converted it from a MAVLink python script on board of the Raspberry Pi, and also a manual mode which can be controlled by the user through a radio controller.

4.2 Subsystem Details

The provided gimbal controller board already interprets signals sent to it to control the gimbal motors to properly stabilize its payload. So the main challenge of this subsystem is to provide useful PWM inputs from either the user or an automated system. The basics of the subsystem are to provide the gimbal controller useful PWM signals that can be used to control the motors, this much hasn't changed throughout the semester. Because the other subsystems had major changes later in the semester, the nature of the needed signals kept changing also. So having to work with and integrate the image tracking subsystem and the flight controller subsystem with the gimbal subsystem is a major challenge. Originally the image tracking system was the PixyCam which had a system that provided PWM signals through the board on the PixyCam. So when the IR system was still our objective, the gimbal controller board didn't have to

interface with the flight controller only the PixyCam IR tracker. Currently now with the license image tracking as our main goal, I will need to write automation scripts which take inputs from Nabilah's software in order to move the gimbal around. It will then interface with Kyle's flight controller in order to output PWM signals to the gimbal controller which will move the motors to stabilize on the target.

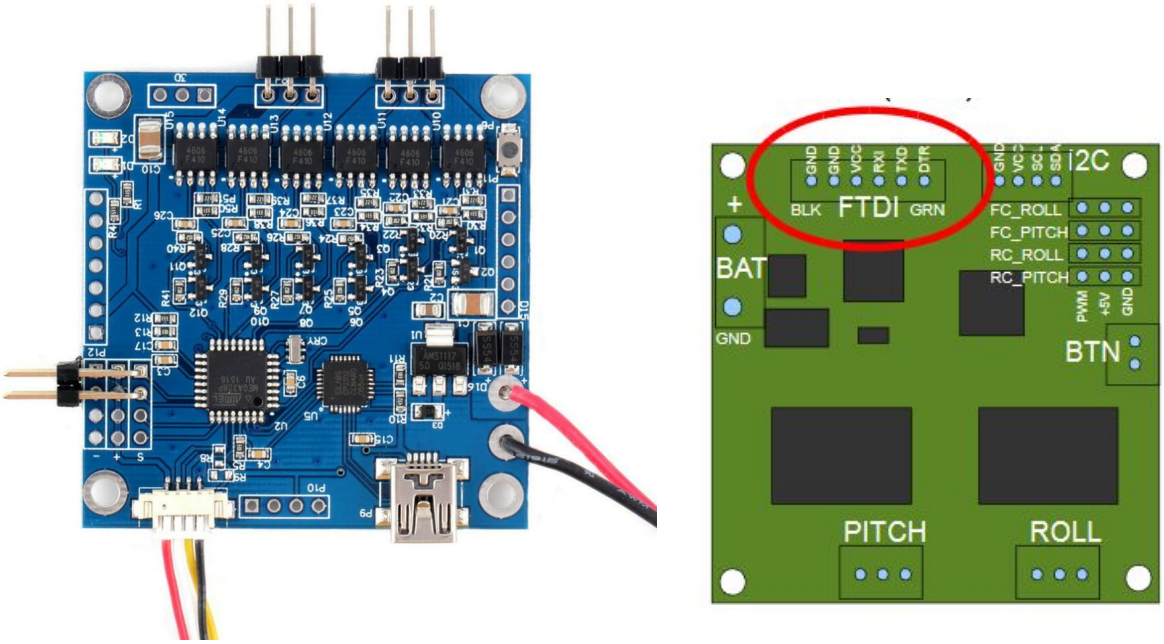

Figure 12: Gimbal Controller Board and Pin Diagram

4.3 Subsystem Validation

There were two ways to validate the subsystem. There was the manual way, which meant that I controlled the movements and placement of the gimbal through an RC controller. I had already validated the gimbal through that method. The other gimbal control method that needed to be validated was through automation. In order to do a real validation test, I would need to integrate the image tracker and the flight controller to provide input to the gimbal controller. But because the other subsystems are currently a work in progress, that isn't viable to do right now. Later when we are approaching the final deadline in 404, it needs to validated this way. The only way to validate the system works automated is through the MAVLink software. I connect the gimbal controller to the Pixhawk flight controller and then I connect to my computer which has the MAVLink software. On the software there are virtual test conditions where I can program the flight controller to tell the gimbal to move around and point at a chosen virtual target. So until the other subsystems are at a further developed level, virtual testing is the only viable method to validate that the gimbal can be automated.

4.4 Subsystem Conclusion

The gimbal subsystem is very much a closed off system. Whereas the other subsystems can receive input and make outputs that can be used elsewhere in other subsystems, the gimbal can only receive inputs in the form of PWM signals , and the gimbal controller only outputs PWM signals to the motors to control it. So the main task with the gimbal is to provide good clean PWM inputs that can control the gimbal motors. In order to do that I will need to integrate data from Nabilah's image recognition so that I have data on where the tracked image is and can move the gimbal to keep it within view. And I will also have to work with Kyle and the Pixhawk flight controller and MAVLink so that I can write scripts that interface with the Pixhawk and the Pixhawk can interpret those scripts into PWM signals sent to the gimbal controller board.

DRONE FOLLOWER

Nabilah Ahmed
Joshua Regresado
Kyle Henry

VALIDATION

REVISION 1

04 December 2019

	Complete
	Terminated/ Not Complete
	Delayed/ Behind Schedule

Task	Subsystem	Date Planned - Date Completed	Team Member
Preliminary Testing of Parts	All	10/23 - 10/24	All
Setup drone control hardware and ensure correct hardware communication	Flight Control	10/27 - 10/27	Kyle Henry
Setup simulation environment for drone control	Flight Control	10/27 - 10/31	Kyle Henry
Simple automation script for subsystem demo	Flight Control	11/04 - 11/03	Kyle Henry
Validate ALPR Code "Test Image"	Detection & Tracking	11/8 - 11/7	Nabilah Ahmed
Validate OpenCV object tracking and OCR separately given input video and Webcam	Detection & Tracking	11/12- 11/14	Nabilah Ahmed
Hybridize object tracking and OCR	Detection & Tracking	11/16 - 11/16	Nabilah Ahmed
Test gimbal stabilization functionality	Gimbal	11/11 - 11/18	Josh Regresado
Send video footage through transmitter to receiver	Camera	11/23 - Next Semester	Josh Regresado
Integrate image recognition into gimbal stabilization	Gimbal and Camera	11/30 - Next Semester	Josh Regresado
Final Testing	All	11/04 - 11/11	All

Performance on Validation Plan

The validation plan was not completed thoroughly, with some subsystems being severely underdeveloped. The flight control subsystem was completed successfully and met all deliverables, as well as the image recognition subsystem. The image recognition subsystem was changed due to concerns over the original plan about halfway through project development, and therefore had reduced deliverables. The gimbal subsystem relied on the gimbal to be delivered, and the distributor of the gimbal was extremely late in this delivery, leaving only a few days for any progress to be made on this front. We plan to meet these missed deliverables over the winter break before starting subsystem integration next semester.