Oregon State University
College of Engineering

CS Capstone Final Report

June 12, 2018

AHS Micro-Air Vehicle Challenge

Prepared for
Nancy Squires, Ph.D.

Prepared by

Group 32
MAV CHALLENGE

Justin Sherburne
Kaiyuan Fan
Yingshi Huang

Abstract

The purpose of this document is to compile all documentation from this project into a single source. This is a culmination of the work contributed by the computer science team for the Micro Air-Vehicle challenge project. We will provide in-depth summaries regarding weekly activities, research, and design attributes for this project. Design choices, key project components, and installation instructions will also be covered in this report.

CONTENTS

List of Figures

REVISION HISTORY

Name	Date	Reason For Changes	Version
Design Document	Nov 29, 2017	Initial Creation	1.0

1 INTRODUCTION

1.1 Project Overview

This project has been proposed by Dr. Nancy Squires. Dr. Squires sponsors a number of collaboratory projects within the College of Engineering, with this being the newest project under her guidance. Dr. Squires served as the supervisor and sponsor for this project. All project planning, design, and implementation was left to the project team members. The goal of this project is to compete in the annual AHS Micro-Air Vehicle challenge[1] and to create additional venues for Oregon State University to represent itself in professional competitions.

1.2 The Project

The Micro-Air Vehicle challenge[1] is an annual collegiate competition hosted by The American Helicopter Society. The competition is conducted by the AHS International Unmanned VTOL Aircraft and Rotor craft Committee. Competitors are tasked with building a vehicle capable of steady-state hovering, avoiding obstacles, and recognizing a drop-off point. This vehicle may compete in either a manual or autonomous flight category. The aircraft is remotely controlled and will have the ability to deliver packages to a specified location.

1.3 Importance

The purpose of this project is to represent Oregon State University in a professional competition. This is OSU's fist attempt at this competition, and we want to go with the best possible aircraft. In the event that the aircraft cannot meet all of the competition requirements we will not attend the competition and pass on knowledge to next year's team.

1.4 Team Members

While this project received additional assistance from underclassmen during initial development, the following list contains the names and roles of the original team.

- Computer Science

  Justin Sherburne - Computer Science team lead and image recognition development.
  Kaiyuan Fan - GUI development and vehicle communication.
  Yingshi Huang - Controls and systems development.

- Mechanical Engineering

  Alec Denhert - Project lead, airframe development.
  Tyler Barrett - Weight balancing and budget management.
  Christopher McBee - Rotor and thrust development.

- Electrical Engineering

  Ian Anderson - Electrical team lead and power management.
  Erik Madison - Ultrasonic and motor controllers.

CS Capstone Requirements Document

Jan 5, 2018

AHS Micro-Air Vehicle Challenge

Prepared for

Nancy Squires, Ph.D.

Prepared by

Justin Sherburne

Kaiyuan Fan

Yingshi Huang

Abstract

This document is intended to define project specifications and requirements. It contains a statement of what requirements need to be met for MAV challenge, how the aircraft works, and technologies and software are used to operate the control board. This document will also include proposed solutions for various challenges, as well as project constraints. It is intended for project sponsors, collaborators, or as a resource to future projects of a similar nature.

REVISION HISTORY

Name	Date	Reason For Changes	Version
Project Requirements	Oct 25, 2017	Initial Creation	1.0
Project Requirements	Jan 5, 2018	Design update	2.0

2 REQUIREMENTS DOCUMENT

2.1 Introduction

The Micro-Air Vehicle challenge is a competition hosted by the American Helicopter Society International. As potential participants in this competition, it is our task to design and build an aircraft capable of competing in the AHS event in May.

2.1.1 Purpose

The purpose of this project is to create a Micro-Air vehicle capable of autonomous navigation. This specific functionality has been replicated on large-scale aircraft, but remains challenging for small-scale applications. The difficulty lies within creating a system that is powerful enough to handle the low-latency calculations required to modify flight trajectories, but light enough to fit onto a vehicle under 500 grams. Applications for this technology could span from package delivery systems to various emergency response and military applications. Similar projects have been undertaken at DARPA[2] and other universities across the united states.

2.1.2 Project Scope

This project intends to create a Micro-Air Vehicle capable of competing in the AHS Micro-Air Vehicle challenge. This means that the vehicle is required to have an emergency shutdown function, and a manual flight overrides. Additionally the vehicle should satisfy the competition requirements regarding weight and size, however those considerations will largely be determined by the mechanical engineering team attached to this project.

This project focuses on an autonomous functionality capable of identifying objects that are important to the competition parameters, and be able to interact with those objects accordingly. This autonomous functionality should be able to sustain autonomous flight for 5 minutes without crashing into any non-moving obstacles.

This project is not intended to create a vehicle capable of navigating outside of the competition parameters, and should not be required to avoid any moving obstacles.

Vehicle requirements will be dependent on competition goals and requirements, and those requirements are subject to change. The success of the project is not dependent on the qualification to compete or performance during the competition, but it is dependent upon the completion and testing of the autonomous functionalities of the vehicle created during this project.

2.1.3 Definitions, Acronyms, and Abbreviations

Definitions

1) **MAV:** Micro-Air Vehicle
2) **AHS:** American Helicopter Society
3) **OpenCV:** Open source Computer Vision
4) **DARPA:** Defense Advanced Research Projects Agency
5) **GUI:** Graphical User Interface

2.2 Overall Description

2.2.1 Product Perspective

The Micro-Air Vehicle Challenge project was sponsored by Nancy Squires Ph.D. This collaborative project was created for the mechanical, electrical, and computer science senior capstone courses with the intention of creating a vehicle capable of competing in the AHS annual Micro-Air Vehicle challenge.

2.2.2 Product Functions

- Able to maintain stable flight for a duration for at least 5 minutes.
- Able to perform takeoff, hover, and land both manually and autonomously.
- Able to stop when the user triggers the KillSwitch function.
- Able to transmit video and sensor data back to a base-station (laptop).
- Able to distinguish between objectives and obstacles while in flight.
- Able to detect and avoid non-moving obstacles.
- Able to pick up and drop off specified objects autonomously.

2.2.3 User Characteristics

We are working under the assumption that all users will have the following characteristics:

- Ability to read and understand English
- Ability to use remote controller to manual flight.
- Familiarity with autonomous navigation requirements and interface.
- Access to any graphical interface associated with the project and familiarity with the functions associated with the interface

Users will not be required to set-up any portion of the communication or flight system, however some initialization steps may be required from a trained operator.

2.2.4 Design and Implementation Constraints

The Micro-Air vehicle is subject to the following constraints based off of previous years competition rules[1]. These constraints will be updated according to the current year's competition once rules are announced

1) The vehicle must weigh less than 500 grams including the battery.
2) The vehicle must be shorter than 17.7 inches in any one dimension.
3) The vehicle is required to operate from an electric power source.
4) It must be able to take off and land vertically, and have the ability to maintain a stable altitude (hover).
5) It must have some sort of on-board camera system featuring at least one camera.
6) It must use standard communication methods, with a preference on 2.4GHz communications.
7) It must have an emergency cutoff switch that will power off the vehicles motors in case of a loss of communication.
8) It must also have a manual control override so an operator can steer the vehicle back to the competition area.

2.3 Specific Requirements

2.3.1 Camera System

1) The cameras should be able to provide video to a remote station.
2) The remote station should be able to process the information from those camera's and send instructions to the vehicle.
3) Object recognition can be handled either by the vehicle or by the remote station.

2.3.2 Control System

1) Manual control should be facilitated through a Microsoft Xbox controller.
2) Autonomous functions should be enabled through a GUI, and should display whether or not it is currently in an autonomous or manual flight mode.
3) The kill switch should override either manual or autonomous flight.

2.3.3 Communication System

1) Must be able to maintain a WiFi connection capable of streaming at minimum: 2 videos at 720p 30fps. This is equivalent to roughly 15Mb/s.
2) Must be able to maintain communication at a distance of 40 ft.
3) In the event of a loss of communication the vehicle will attempt to reconnect for 4 seconds, and then initiate the kill-switch command.

2.3.4 Autonomous System

1) The vehicle should be able to discern what is a target, what is an obstacle, and what is a boundary.
2) Using distance sensors or camera information the vehicle should be able to stay within the competition area without running into an obstacle.
3) Using distance sensors or camera information the vehicle should be able to locate and pick up an object specified by competition rules[1].
4) Using distance sensors or camera information the vehicle should be able to drop off the same object at a specified location

2.3.5 Hardware Specifications

1) Hardware must be small enough to fit within the size/weight constraints specified by the competition rules[1].
2) Hardware must be able to be powered by the on-board power system designed by the electrical engineering team.
3) The hardware must be capable of transmitting data over a 2.4 GHz system.

2.3.6 Software Specifications

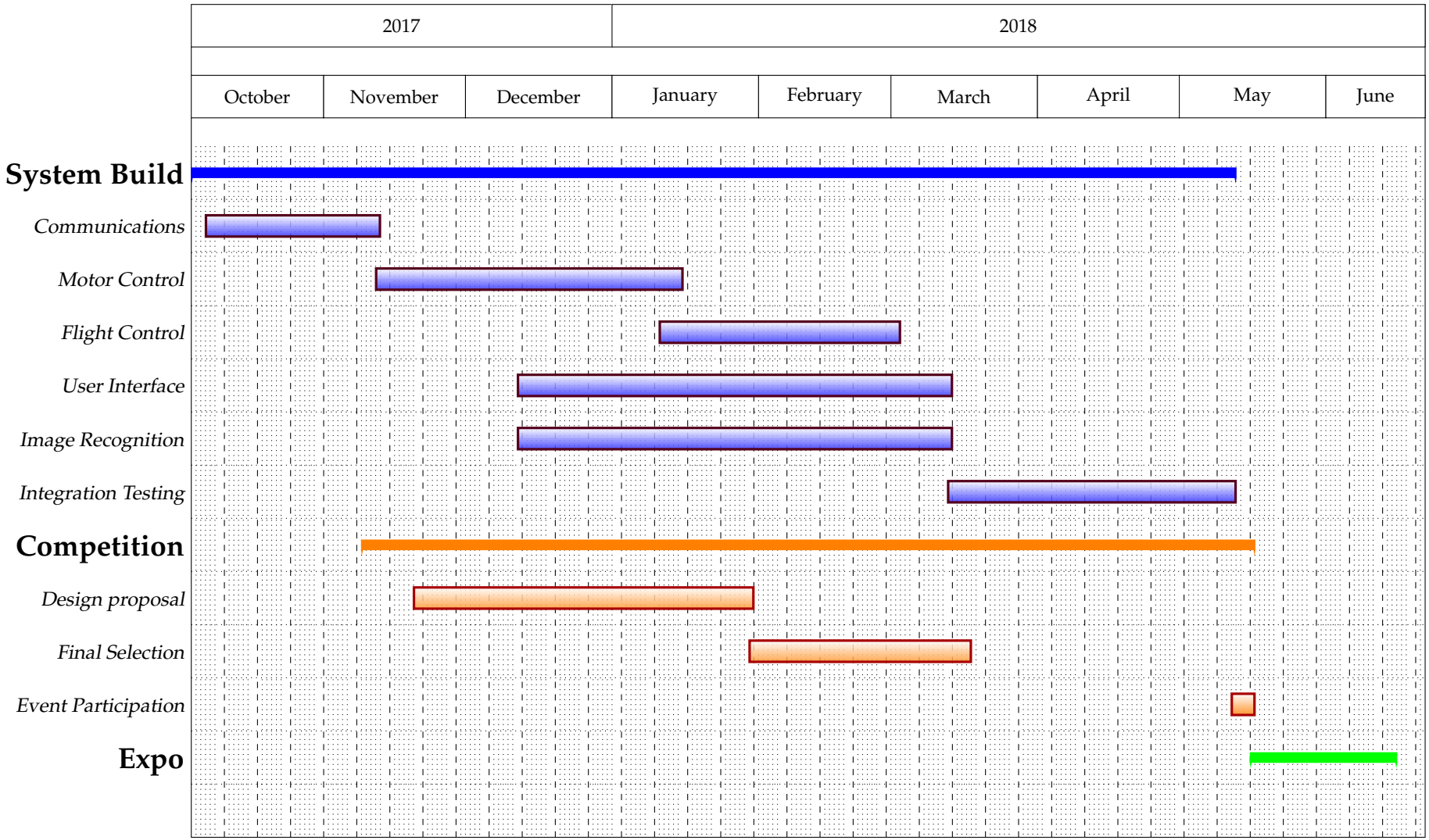
1) The software should be capable of processing images and relay corresponding responses in less than 500ms + communication latency.
2) The software should be capable of translating manual flight commands to motor controller outputs in less than 100ms + communication latency.

3) The software should be able to display two video outputs to a GUI interface at the base station. The video output at the base station should be delayed no less than 500ms + communication latency.
4) The software systems should utilize open source libraries if they are available to minimize the amount of proprietary code implemented in this project. For example OpenCV is an open source library used to translate video imagery to usable information for autonomous system.

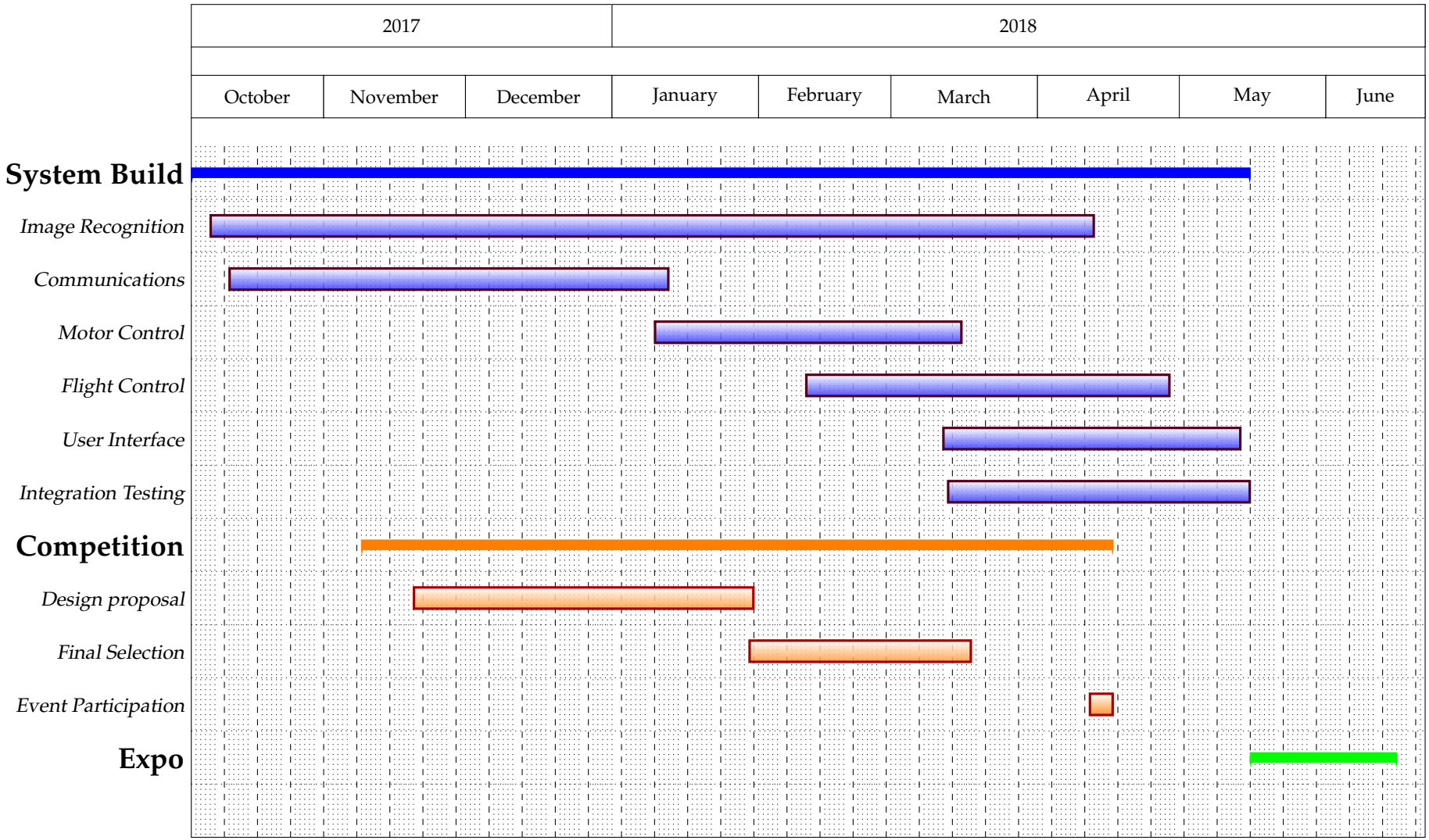
2.4 Document Changes

This document was modified from it's original format on January 5th, 2018. The purpose of this change was to bring this document's title-page and format up to date with the other documents for this project. No requirements were changed in this modification, only stylistic aspects of the document were modified.

2.5 Estimated Timeline



2.6 Actual Timeline



CS Capstone Design Document

Jan 5, 2018

AHS Micro-Air Vehicle Challenge

Prepared for

Nancy Squires, Ph.D.

Prepared by

Justin Sherburne

Kaiyuan Fan

Yingshi Huang

Abstract

The purpose of this document is to elaborate on design concepts related to the implementation of the Micro Air Vehicle project. Our goal is to provide our intended audience with information on the design and implementation of our core features. Here we will outline technical concerns and viewpoints contained within the scope of the Micro Air Vehicle project.

REVISION HISTORY

Name	Date	Reason For Changes	Version
Design Document	Nov 29, 2017	Initial Creation	1.0

3 DESIGN DOCUMENT

3.1 Overview

3.1.1 Scope

This document is intended to inform the reader of the technical goals of the Micro Air Vehicle Challenge project. Here we will describe subcomponents of the project, and how each subcomponent will be implemented into the overall design. It will provide a clear framework for each piece, and provide viewpoints on how each element will be designed. It is not a definition of requirements, or a concrete time line of events leading up to the completion of the project.

3.1.2 Purpose

The purpose of this document is to describe how our project will be implemented to accomplish the tasks defined by the AHS 6th annual MAV challenge. The goal of this project is to create a remotely piloted helicopter with an autonomous flight feature. This document will explain the details of the project's design.

3.1.3 Intended Audience

The intended audience of this design document is project team members, project sponsors or facilitators, and project collaborators.

3.2 Definitions

AHS - American Helicopter Society.

Autonomous - Acting independently from any controlling source.

CSI - Camera Serial Interface. This is the primary interface for the Raspberry Pi cameras.

ESC - Electronic Speed Controller. Used to control motor speed with low voltage logic.

GUI - Graphical User Interface.

GPIO - General-purpose input/output.

MAV - Micro Air Vehicle.

USB - Universal Serial Bus

3.3 System requirements

3.3.1 Functional Requirements

- The vehicle must weigh less than 500 grams including the battery.
- The vehicle must be shorter than 17.7 inches in any one dimension.
- The vehicle is required to operate from an electric power source.
- It must be able to take off and land vertically, and have the ability to maintain a stable altitude (hover).
- It must have some sort of on-board camera system featuring at least one camera.
- It must use standard communication methods, with a preference on 2.4GHz communications.

- It must have an emergency cutoff switch that will power off the vehicles motors in case of a loss of communication.
- It must have a manual control override so an operator can steer the vehicle back to the competition area.

3.3.2 Non-functional Requirements

- The vehicle should be able to navigate the competition area without human intervention.
- The cameras should maintain a minimum quality of 540p at 30fps.
- The image processing system should recognize each of the following objects: landing areas, packages, boundary lines, and large obstacles.
- Hardware should be light enough and small enough to fit within the vehicle limitations.
- WiFi communication should maintain a steady connection at a distance of more than 50 ft.

3.4 Conceptual Designs

3.4.1 Design Overview

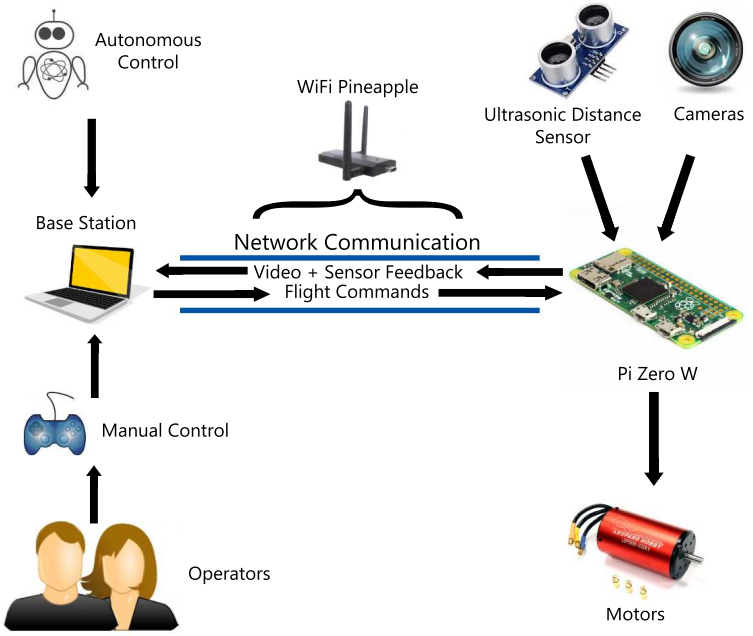


Figure 1. Conceptual Design Overview

3.4.2 Design Components

3.4.3 Device Communication

The communication between the aircraft and the base station will be connected by WiFi. Data is transferred over a 2.4GHz channel, allowing Raspberry Pi[3] board and base station to receive and transmit data when they are in the range of a WiFi network. We will establish a WiFi network by using a router, specifically the WiFi Pineapple[4]. In the event of communication loss, the vehicle should attempt to reconnect for no more than 3 seconds. If the connection is not re-established, the vehicle will shut down the motors.

3.4.4 Hardware

The two main logic control components are the base station and the Raspberry Pi[3]. These two components will be able to communicate to each other through the WiFi Pineapple router[4]. The Base station will handle the difficult computations regarding image processing and flight planning. Additionally, the base station will be able to select between manual control mode or autonomous control mode using a GUI. The Raspberry Pi will handle motor control logic, and elevation control. Video and sensor information will be based from the Raspberry Pi to the base station, and directional commands will be returned from the base station to the Pi.

3.4.5 Flight Control

We will connect motor controllers to the Raspberry Pi with GPIO pins. We will write a GPIO access program to access GPIO pins and establish communication with motors. We can configure the GPIO pins to start and shutdown the motors. Furthermore, we can set the logical voltages either low or high to change the speed. This code to control the motors will be able to handle commands from both manual and autonomous inputs.

3.4.6 Sensors

Camera Sensors are used to easily capture images. For the manual flight, the camera sensor will stream video (image frames) back to the base station. For the autonomous operation, these “images” can be used by video-processing algorithms for recognizing the target, searching the home-base, and operating the hover-hold. We will equip two cameras to our aircraft. One front Camera to process stream video and one at the bottom to find targets and the landing places.

The ultrasonic sensor can emit a specific frequency of sound waves and listen to that sound wave bounce back. The sensor can record the elapsed time between the sound wave being emitted and the sound wave bouncing back[5]. Once we know the speed of the sound wave, it is able to calculate the distance between the ultrasonic sensor and the object. we will use an ultrasonic sensor at the bottom of aircraft to detect the height from the ground and measure the boundary range.

3.4.7 Image Processing

All image processing will be done on the base station to increase calculation performance. Image processing should be able to distinguish four main objects: Delivery targets, packages, boundary lines, and obstacles. Once these objects are correctly identified, the autonomous system will then have to identify where that object is located in correlation to the vehicle. After a general location is established (Left, Right, in front, below, ect.) The autonomous system should then send the appropriate flight commands to the vehicle.

3.4.8 User Interface

The user interface should be simple, with only the necessary information needed to control the micro air vehicle displayed. There should be a white button to enable manual flight control, as well as a red button to stop all flight commands. At minimum the front facing camera stream will be displayed on the user interface. The second camera stream may be added if computational power is not significantly compromised. Additionally, image processing overlays may also be added under the same circumstances.

3.5 Design Viewpoints

3.5.1 Introduction

Viewpoints are an in-depth evaluation for each of our above design components. Here we will break each viewpoint into three primary categories: design concerns, design elements, and design examples. Design concerns address what we are trying to solve, and what problems we are anticipating to come across. Design elements will break down the specific goals of each viewpoint. Within the deign elements we will elaborate on each specific task and a general idea of how it will be implemented. In design examples we will review any existing technologies that might solve the viewpoint we are addressing. If there are no existing technologies, then we will review similar applications and explain what we are doing differently.

Design viewpoint	Design concerns	Example Design Concepts
Communication	Reliability, Fail-safe Operations	Video streaming applications and standard data packet transfer over WiFi.
Hardware	Reliable, Lightweight, Multi-functional	Raspberry pi, CSI cameras, Ultrasonic sensors, WiFi Pineapple
Image Processing	Object Detection, Directional Translation	OpenCV, Matlab Image toolbox, directional markers
Flight Control	Motor control, Logic outline, Input Arguments	ESC motor controller and GPIO pins.
Positioning	Ultrasonic Distance sensors, Image distance calculations	OpenCV distance and positioning calculations, distance based on ultrasonic readings.
User control	User Interface, Manual Control Methods, Emergency Shutdown	Web interface attached to the base station system with basic user controls.

3.5.2 Communication Viewpoint

- **Design Concerns:**

  The main design concern for communication is maintaining a low-latency, high-bandwidth connection between a base station and the vehicle. Design considerations should be made based on what will best address the above design constraint. Additional concerns about communication distance and strength will be secondary considerations for this project.

- **Design Elements:**

  There are three objects that rely on steady communication channels: The base station, the vehicle, and the router. The router will act as an intermediary to the base station in order to boost the signal range, and produce a local network that the vehicle can connect to. The base station will connect to the router via USB or Ethernet cable, and the vehicle will connect over the 2.4GHz wireless channel. This creates a closed network that we can easily secure to communicate from our base station to our vehicle and back. Video and Ultrasonic Sensor information will be passed from the vehicle to the base station, and the flight calculations will be returned to the vehicle for execution.

- **Design Examples:**

Typically MAV's are controlled using radio communication, so our approach to this is not conventional. The main reason behind this is because WiFi has limited range compared to radio communication methods. For our application we will be in relatively close proximity to the vehicle, so this makes WiFi a better option for data throughput.

3.5.3 Hardware Viewpoint

- **Design Concerns:**
  Our primary concerns with hardware is being lightweight enough to fit within the constraints of our vehicle requirements, and being able to communicate to the vehicle reliably. We will elaborate on our hardware decisions below based on those primary requirements. Secondary considerations will be based on ease of use, and project implementation.
- **Design Elements:** Our hardware requirements are limited to a couple different factors. The following components are needed for our implementation: A camera, a distance sensor, a base station, a router, and a vehicle controller. The vehicle controller is our most constrained object because it is limited by our size and weight limitations. This is why we decided to use the Raspberry pi zero. This computer is one of the most powerful computer for it's size on the market, and it is extremely inexpensive. Our camera was also decided base on this choice because is was specifically designed for the CSI input for the raspberry pi. The router (WiFi Pineapple) is based on what was on hand, but it also has the added benefit of not needing an external power source. The base station can be implemented on any laptop as long as it is capable of running Ubuntu or another Lunix distribution.
- **Design Examples:**
  Our design is typical of any server-client relationship. Our base station will act as the sever, while the vehicle is acting as the client. The client (vehicle) relays information from their data sources back to the server. The server will then perform the difficult computations that the client is not capable of handling. There is an intermediary, the router, which is also found in any client-server relationship. This element is less documented, but equally as critical to the overall success of the project.

3.5.4 Image Processing Viewpoint

- **Design Concerns:**
  The concerns of image processing can be separated into two parts: object detection and directional translation. During competition, the object to be detected is the envelop with two colors which are white and red. The envelop might not be in the original video range at the beginning of our flight, so the detection depends on the path of vehicle. During flight, the angle of the cameras will be changing. This causes the view of object to also change. Positions and shapes can increase the difficulty of detecting the object. The directional translation component is based on the relative location within the images received by the cameras. The view from cameras alone cannot estimate exact distances, so sensor and mathematic calculation based on known sizes can decrease the deviation.

- **Design Elements:**
  At the start of the flight, the cameras will capture video. The video data will send to the base station via WiFi router. If the software cannot discover the object, then it will continue to move around the environment. If the

object in the range of camera, base station will calculate a path to pick up the package. According to the path, the vehicle will convert this path into movements. On the way to pick up the object, the micro helicopter will avoid any obstacle it encounters. The base station will keep track of movements and make sure the micro helicopter arrives at the package pick up place. The micro helicopter will then pick up the envelop and attempt to bring it to the corresponding destination.

- **Design Examples:**
  In order to process images, the vehicle needs a program that can handle streaming video in an efficient manner. OpenCV and Matlab Image toolbox can accomplish this task. However, OpenCV is an open source library, while Matlab is a licensed product (which is available to students for free). OpenCV is focuses more on the object detection so it is a better choice.

3.5.5 Flight Control Viewpoint

- **Design Concerns:**
  The main design concern for flight control is maintaining a stable connection between the Raspberry Pi with the multiple ESC motor controllers and motors. Additional concerns will be ESC motor controllers sending the correct pulsing signals to motors with the commands received from the users.

- **Design Elements:**
  Flight control will be accomplished through connecting the ESC motor controllers with the Raspberry Pi through the GPIO pins[6]. Then, we will have each ESC motors controllers with each motor and with an extended power bank. We will write a GPIO access program to enable the GPIO pins to write outputs to the ESC motor controllers, then proceed to signal the motors. Once the connection completed, users can control the flight by sending the command to the Raspberry Pi board.

- **Design Examples:**
  There are many examples of this using typical radio communications to control air vehicles. These are low-latency algorithms used to convert simple directional inputs into frequency based control algorithms for the ESC controllers.

3.5.6 Positioning Viewpoint

- **Design Concerns:**
  The concern for positioning are the measurement accuracy of the ultrasonic sensor and the deviation between the realistic distance with the distance calculated by the image processing algorithm.
- **Design Elements:**
  Camera sensors and ultrasonic sensors will be implemented to measure distance[7]. If we know the dimension of an object and we can use it as a reference to the image that collected by the camera sensor. Distance can be calculated with various of algorithms using OpenCV[8]. The ultrasonic sensor should be connected to the

Raspberry Pi through the GPIO pins. Ultrasonic sensors can measure the distance to the ground by emitting specific frequency of sound waves and recording the elapsed time until sound wave bouncing back.

- **Design Examples:**
  Distance from camera to object can be calculated through triangulation similarity. If we have an object with a known width W. We have initial distance from the object to the camera. Given the initial image collected by the camera sensor, we can measure the pixel width of the object in the image via OpenCV, marked as P. The focal F can be calculated as F = (P * D) / W. Then new distance D' can be measured with the new pixel width in the later image. The formula will be D' = (W * F) / P.

3.5.7 User Control Viewpoint

- **Design Concerns:**
  The User interface should display 4 things. 2 video feeds, a manual override, and a emergency stop. The second window camera interface is completely optional because it is only needed for flight object recognition. The function buttons will take priority over the second window for a camera stream.

- **Design Elements:**
  This is an interface required for the user to control the helicopter through the base station controller. The user will have control through an external controller for forward, backwards, up, down, left , and right. The user control interface also needs to be able to attached with web interface.

- **Design Examples:**
  The physical controller can be na illustration for the user interface. Every controller should include all the basic functions needed, like turn left or right and move forward and backward. Our different directions should be represented by individual keys on the controller. And the emergency shutdown should be accessible through the controller or the user interface on the base station.

3.6 Conclusion

The Micro Air Vehicle challenge not only encompasses the design requirements listed above, but also the design requirements put forth by the Electrical and Mechanical sub-teams. While this design document provides a comprehensive coverage of the logical systems required, there are additional contributions that are not specified above. This document adequately outlines the design requirements for the Computer Science sub-team attached to Oregon State University's Micro Air Vehicle Challenge team.

CS Capstone Technology Review

Nov 20, 2017

AHS Micro-Air Vehicle Challenge

Prepared for

Nancy Squires, Ph.D.

Prepared by

Justin Sherburne

Kaiyuan Fan

Yingshi Huang

Abstract

The purpose of this document is to evaluate technologies related to the implementation of our project. Our goal is to evaluate different technologies and determine the feasibility of each within the scope of our project. Each technology will be evaluated based on our selected criteria.

4 TECHNOLOGY REVIEW

REVISION HISTORY

Name	Date	Reason For Changes	Version
Technology Review	Nov 20, 2017	Initial Creation	1.0

4.1 Justin Sherburne's Role

My role in the project as the computer science team lead is to effectively communicate project responsibilities and plans from the mechanical and electrical teams to my group. In addition to this I will also be developing the image recognition portion of our project. This will involve determining both hardware and software components of this implementation. The image recognition portion of our project will directly tie into our overall success in building an autonomous flying vehicle.

4.2 Hardware Controllers

4.2.1 Overview

We were asked by the Electrical and Mechanical teams for this project to decide what hardware we would require on-board of our Micro-Air Vehicle. This is going to be the brains that directly interfaces with the motors. At minimum it should be able to translate manual commands into usable flight instructions for the vehicle. At most this hardware should be able to handle video and data streaming from a base station while maintaining a stable flight path.

4.2.2 Criteria

1) The controller should be smaller than half of our maximum allowed dimension. This means it should be shorter than 8.85 inches in any dimension. However, smaller boards should be given preference in order to reduce our footprint on the overall design.
2) The controller should not use an excessive amount of power, and should not require an external power adapter.
3) The controller should have a port to attach to a camera. This can be USB, PCIe, CSI, I/O pins, ect.
4) The controller should have I/O pins capable of both transmitting and receiving electrical signals
5) The controller should be capable of maintaining a stable wireless connection via either built-in hardware or using an additional adapter.

4.2.3 Potential Choices

My initial research led me to two very similar options: The Raspberry Pi[3], and the OpenRex. For a third option I am including the Arduino platform due to its versatility. Another possible third option I was considering was the Nvidia Jetson TX1. However, the Nvidia was the largest of the three options, and also required the most amount of power.

4.2.4 Arduino

The Arduino platform is unique in the fact that it can be customized for most applications you could use a Pi or OpenRex on. Adding additional modules called shields to the base Arduino can give it wireless capabilities as well as USB and camera input options. Additionally, the MKR ZERO[9] is a similar size to the PiZero which makes it ideal for our application. Power is supplied via a micro-usb input, and runs on less than 2A. Additional boards may require separate power, however they can be powered in parallel with micro-usb adaptors with the same power supply. I/O

pins would need to be utilized for additional shields added to the Arduino, however a specific shield could also be used for motor control. The Arduino has a micro-controller rather than a processor, meaning that it cannot support a true OS, and will generally be inferior at complex tasks.

4.2.5 Raspberry Pi Model 3

The Raspberry Pi is one of the most commonly used controller boards because it is capable of running a full OS like Raspbian. It's power works identically to the Arduino platform, being supplied via micro-usb. In addition to being able to support an OS, the Pi also comes with additional ports built into the board. The CSI port is capable of handling a video feed from a camera, and USB ports can also accept camera inputs. There are two versions of the Pi that are suitable for our project; the Pi model 3[10], and the Pi Zero W[3]. The Pi Zero is a smaller version of the 3, with less processing power, but still maintaining the on-board wireless capabilities. The processor is reduced on the Pi Zero, which is perhaps the largest difference between the two controllers. Both models also include 20 I/O pins that can be configured as needed. The Pi 3 model B has 4 core processor running at 1.2GHz, however it only supports 1GB of ram. The Pi Zero has a single core processor running at 1GHz, in addition to 512MB of RAM.

4.2.6 OpenRex

The OpenRex[11] board is essentially a more-powerful version of the Pi 3. It includes a couple more ports like SATA as well as an on-board gyroscope. While some of these features could prove useful to the overall project, others are completely unnecessary for our application (like the humidity sensor). Additionally, the OpenRex does not have as much community support as the Pi and Arduino platforms. I/O, power, and size are nearly identical to the Pi, however, the OpenRex does not have built-in wireless communication. The OpenRex has a MX6 processor running 4 cores at 1.2GHz. Additionally, it supports up to 4GB of ram which means it could be capable of handling image processing without a base-station.

4.2.7 Discussion

The Pi is the strongest contender here for a number of reasons. The first being cost. While the Arduino is very comparable at $35, the OpenRex board costs as much as $335 for their top model. The Pi comes with a similar processor, and still includes a processor in its smaller form on the Pi Zero. The Pi Zero costs only $10, and still includes the wireless capabilities that the Arduino and OpenRex do not have. The additional boards needed to run the Arduino and OpenRex add unnecessary complexity to the project.

4.2.8 Conclusion

The Pi Zero W[3] is going to be our board of choice for this project. While it's processor is smaller than the Pi or OpenRex, our solution is to use the processing power of our base-station to our advantage. The small form factor reduces our weight and size while still keeping the camera, wireless, and I/O connections we need. In addition to being small, it is also the cheapest option and has a very large support community should we run into problems that are hardware specific.

4.3 Operating Systems

4.3.1 Overview

For our project we will need 2 operating systems that are capable of interacting with each other in one form or another. One OS is for our controller on-board of the vehicle, and the second is for our base-station. In this comparison, all three OS's we are looking at are able to be implemented on either the base station or the Pi.

4.3.2 Criteria

Operating systems will be evaluated based on the following criteria:

1) The OS being loaded onto our controller must be lightweight and efficient, capable of running on a Pi Zero W
2) The OS running on our base-station must be capable of effectively utilizing the processing power of that system, in addition to being lightweight and stable.
3) Both operating systems should be based on a common platform and share similar file structures.

4.3.3 Potential Choices

There are quite a few Linux distributions floating around that could be very applicable to this project. However, Operating systems that are widely used will take precedence here because generally, they are stable over a variety of platforms. Because of this we will be looking into Raspbian for the Pi, as well as Win 10 iot core. For the base-station we will be comparing Ubuntu to Windows architectures. It is important for these to be similar because it gives us flexibility during development. If we are running Windows we can develop an application on either device without needing to translate code for a different system.

4.3.4 Rasbian

Rasbian[12] is a Debian based operating system that is specifically designed for the Raspberry Pi boards. Since Pi's have been selected as our hardware of choice, it is a logical choice to use Raspbian as our default OS. The problem with Raspbian is that it is loaded with a lot of unnecessary programs that will not be needed for our project. The GUI is beneficial for troubleshooting, however it consumes resources and will ultimately be disabled to favor performance.

4.3.5 Windows 10 / Win 10 iot Core

Windows 10 and Win 10 iot core[13] are both based on the Windows architecture. The Win 10 core is capable of running on the Pi, and interfaces smoothly with other windows devices. Windows is stable and can effectively manage hardware resources for bulky programs. The downside to this is that there is a small community that is using the Win 10 core on the Pi, but support is limited. Additionally, if the Pi is not going to be running a Windows OS, then the base-station should be similarly converted.

4.3.6 Ubuntu

Ubuntu[14] is also Debian based, like Raspbian. This makes it a preferable choice over windows if the Pi is not running a Win 10 iot core. Ubuntu is the most installed linux distro, and has a stable version available for free. There are also lightweight versions of Ubuntu, making it a possible candidate for running on the Pi as well. Ubuntu would allow us to easily communicate, develop, and test using one stable platform.

4.3.7 Discussion

The key in this selection is finding a balance between two operating systems that results in simplicity. There are two clear options here. The first is a Windows 10 base-station combined with a Win 10 iot core running on the Pi. The second option is some sort of Debian - Debian configuration. This could mean Ubuntu to Ubuntu or Rasbian to Rasbian

4.3.8 Conclusion

It comes as little surprise that Raspbian is going to be the preferred OS for the Pi. It is lightweight enough to run efficiently on the Pi Zero, even with a GUI. The community support for Raspbian also gives it an edge over Win 10 iot core and Ubuntu. For our base station we should use Ubuntu, or a similar Debian based OS. Ubuntu is stable, widely supported, and can run alongside Windows on the same machine. Ubuntu will allow us to develop programs for the Pi without needing a special IDE. Additionally, using 3rd party utilities like OpenCV are simpler for Ubuntu / Raspbian because they are designed for a linux system.

4.4 Image Processing

4.4.1 Overview

Our Micro-Air Vehicle is required to have an on-board camera system to stream video back to our base-station. So we are choosing to use this camera with an image-recognition software that is capable to finding the objects we need to look for during the competition. Image processing software is cumbersome, and with our Pi Zero we needed a unique solution to provide adequate processing power. So we are choosing to send the video feed back to our base station for processing, and sending the corresponding commands back to the controller.

4.4.2 Criteria

1) The program or software must be capable of running on a laptop that is not connected to the internet.
2) It must be able to identify the main obstacle located near the center of the competition area.
3) It must be able to identify the boundary lines that mark the edge of the competition area.
4) It must be capable of identifying landing areas, as well as the packages we need to pick up and deliver.

4.4.3 Potential Choices

Writing our own algorithm to locate objects is far beyond the scope of this project, so we need a third-party program for the base architecture. Matlab can be fairly lightweight if you do not include the GUI and additional packages, which makes it a viable option for this project. OpenCV and OpenSLAM are open source projects that could also meet our criteria. OpenCV is specifically designed for image processing, while OpenSLAM is intended for 3-D mapping projects.

4.4.4 Matlab - Image Processing Toolbox

Matlab has many different packages that run with it. One of these packages is the Image Processing Toolbox[15]. This toolbox allows you to take a camera or video input and filter out colors, detect objects, and digitally stabilize video streams. The downside to Matlab is that it is bundled with other packages and isn't necessarily the most efficient processing option. For example, the image processing toolbox recommends being installed alongside six additional toolboxes. It does, however, run on a variety of platforms and languages making it simple to migrate from one device to another in the future.

4.4.5 OpenCV

OpenCV[16] is an open source library the is designed specifically for computer vision and image processing. It runs in C, C++, Python, and Java. It also supports Windows, Linux, MacOS, iOS, and Android platforms. "OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing." . OpenCV is capable of filtering colors to identify specific objects. Additionally, by knowing a size of a specific object, OpenCV can then calculate its distance away from that object.

4.4.6 OpenSLAM

SLAM stands for simultaneous localization and mapping. OpenSLAM[17] is an open-source repository of projects that utilize various SLAM algorithms to map environments. Essentially SLAM sets out to map a complete environment by tracking an objects location in approximation with a moving target. This requires an accurate measure of the vehicles speed, as well as a power computer to run the algorithm. While SLAM algorithms can be tailored to run on the available resources, it is primarily designed for applications like autonomous cars which can be fitted with powerful processing units. Additionally, it is intended to map 3 dimensional environments, not to filter for individual colors.

4.4.7 Discussion

Matlab provides a robust system that can also be used for other sensors and flight information. However, it is not the best option for image processing due to its required dependencies. OpenCV and OpenSLAM are both open source options that are fully capable of identifying the objects we need during the competition. OpenSLAM takes this a step further though and actively maps the environment we are flying in.

4.4.8 Conclusion

OpenCV is going to be our preferred option here because it accomplished everything that the Matlab software can in a smaller package. OpenSLAM could be modified to locate colors during our competition, but the 3-D mapping functionality that it is intended for is not something we need to have for our project. OpenCV is the lightweight and simple option that accomplishes what we need it to do and nothing more.

4.5 Kaiyuan Fan's Role

My role as a member of computer science team in this project is to actively share information with team leader and co-operate with the mechanical and electrical engineering teams. I will mainly work on implementing the autonomous flight of the vehicle.

4.6 Data Collection Equipment

4.6.1 Overview

We plan to design and manufacture a helicopter to participate in the 2018 micro air vehicle challenge competition. We are going to complete a delivery package mission in the competition. The delivery and pickups location areas are roughly known, the exact location of each area of interest is unknown and there has a rectangular obstacle between the base and the delivery area. Therefore, our flying vehicle needs to equip sensors to be able to find the targets, detect the obstacles, stay flight within the mission boundary and search the landing places.

4.6.2 Criteria

1) The sensor should be able to collect data to measure the distance to the ground and the objects, the accuracy, and range of measurement should fit the competition environment.
2) The weight of sensor should be light enough to match the competition requirement.
3) Sensors should provide enough information for both manual fight and autonomous flight.
4) Sensors should be easy to find on market and should not be expensive.

4.6.3 Potential Choices:

The camera sensor is the most common choice for collecting images. By my research, there are various of sensors available on the market and I find the ultrasonic sensor and infrared sensor could be potential choices.

4.6.4 Camera Sensor

Camera Sensors can easily capture the images. For autonomous operation, these “images” can be used by video-processing algorithms for recognizing the target, searching the home-base, and operating the hover-hold. For the manual flight, the camera can collect the most significant real-time stream video and send back to the pilot.

4.6.5 Ultrasonic sensor

The ultrasonic sensor can emit a given frequency of sound waves and listens to that sound wave bounce back. The sensor can record the elapsed time between the sound wave being emitted and the sound wave bouncing back. Once we know the speed of the sound wave, it is able to calculate the distance between the ultrasonic sensor and the object. A minimum distance from the sensor is required to provide a time delay so that the “echoes” can be interpreted. [18] Ultrasonic sensor can measure distance without damage and are easy to use and reliable.

4.6.6 Infrared sensor

Infrared sensors use infrared radiation to detect an object. ”The main use of infrared sensors in robotics is for obstacle avoidance”.[19] It also can use for measuring distance without touching a surface. Similar to the ultrasonic sensor, but two close infrared sensors measure distance by shining a beam of infrared light and uses a photo-transistor to measure the intensity of the light that bounces back. The response speed of infrared sensor can be fast but the range usually under 20 ft.

4.6.7 Discussion

Ultrasonic sensors can be used to control or indicate the position of targets and obstacles. Some objects might not be detected by ultrasonic sensors. This is because some objects are ”shaped or positioned in such a way that the sound wave bounces off the object”, but are deflected away from the Ultrasonic sensor. It is also possible for the object to be ”too small to reflect enough of the sound wave back to the sensor to be detected”[5]. Ultrasonic sensors and infrared sensors can both be used to detect obstacles, but cannot easily to process data to make it suitable for visualization like the camera sensors. Infrared sensors have a relatively short distance measurement. All three types of sensors are cheap and can be easily found at the market, but the camera is the best technology to use for implementation. With the OpenCV support, we can detect the different targets and landing places with their colors. We can even measure the distance to the object if we know the original size of the object. The camera is the essential technology in our implementation.

4.6.8 Conclusion

We are going to implement two cameras in our design. One in front Camera to process stream video and one at the bottom to find targets and the landing places. We also decide to use an ultrasonic sensor at the bottom to detect the height from the ground and measure the boundary range. The helicopter can be stable hover at an altitude and accomplish various tasks with these two technologies.

4.7 Video Stream Interface

4.7.1 Overview

In this project, we are choosing Raspberry Pi as our helicopter's board. Raspberry Pi is a powerful board, we are using Raspberry Pi Camera Module to collect and send stream video (image frames) to our base station. It's necessary to receive the stable and reliable stream video from the helicopter, so the pilot can manual the flight. We need to use an interface to communicate between the camera module to the base station.

4.7.2 Criteria

1) The video stream interface should stream video should under a reasonable latency.
2) The video stream interface should support the UDP (User Datagram Protocol) for data transmission.
3) The video stream interface should support the Raspberry Pi camera module and convey high-quality stream video.

4.7.3 Potential Choices:

We have been implemented the MJPG-streamer and it worked well. I found there are plenty of choices we can choose, I will discuss two common technologies: VLC media player and RPi-Cam-Web-Interface.

4.7.4 VLC media player

VLC media player (commonly known as VLC)[20] is a free, open-source, and cross-platform media player. VLC is available for desktop operating systems and mobile platforms, including the Raspberry Pi. It's easy to install the VLC media player on the Rasberry Pi board and use it to stream video to another device.

4.7.5 MJPG-streamer

MJPG-streamer[21] is also a free and open-source interface. It uses command line application that copies JPEG frames from one or more input plugins to multiple output plugins. It can be used to stream JPEG files over an IP-based network from a webcam to various types of browsers and media players that are capable of receiving MJPG streams. It was originally written for embedded devices with very limited resources in terms of RAM and CPU.

4.7.6 RPi-Cam-Web-Interface

RPi-Cam-Web-Interface[22] is a web interface especially for the Raspberry Pi camera module. It can be used for a wide variety of applications including surveillance and time-lapse photography. It is a free, open-source and highly configurable interface. It can be opened in any browser including the smart phones. The web interface includes motion detection, time-lapse, and video recording.

4.7.7 Discussion

All three interfaces support the user datagram protocol. In a real-time stream, some data loss may be acceptable with the aim of a timely display. VLC is easy to implement but have a relatively high latency compared to other applications. Both MJPG-streamer and RPi-Cam-Web-Interface can send reliable stream video to the base station. There are many other interfaces like pi-camera fits our video stream criteria. MJPG-streamer support various viewports. RPi-Cam-Web-Interface can extend to use motion detection and time-lapse photography.

4.7.8 Conclusion

We have various choices of the interface to stream video via the Raspberry Pi camera module. We will keep testing the best suitable one at the future time. Furthermore, we will be programming to implement OpenCV to detect the objects and measure distance in the streaming video.

4.8 Communication

4.8.1 Overview

Our helicopter need collect data and send to the ground station. The communication between helicopter and ground station should be reliable and stable. Communication requires high speed to process stream video, the communication latency and the connectable distance should be in a reasonable range.

4.8.2 Criteria

1) The communication technology should maintain a transmit speed capable of two videos streaming at 720p 30fps. This is equivalent to roughly 15Mb/s.
2) The communication should be stable under the range of 40 ft.
3) The communication technology should follow the competition rule with the standard communication frequency 2.4 GHz.

4.8.3 Potential Choices:

Our flying vehicle needs be able to remotely communicate with the base station. Wireless communication technology is the prerequisite. The WiFi and Bluetooth are two effective wireless communication technology. While I only find these two technologies, I opened the scope of research and include wired ethernet as the potential choice.

4.8.4 WiFi

WiFi, or Wireless Fidelity[23] is the name of a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections. WiFi technology are commonly using in our daily life. WiFi allows networking of computers and our flying vehicle without the need for wires. Data is transferred over radio waves, allowing WiFi capable devices to receive and transmit data when they are in the range of a WiFi network.

4.8.5 Bluetooth

Bluetooth is another wireless communication technology. It is can exchange data over short distances by using radio waves at the 2.4GHz band. ”By using the 2.4 GHz band, two Bluetooth devices each other can share up to 720 Kbps of capacity”[24]. The connectible range of Bluetooth is relatively low as 10 meters, about 30ft.

4.8.6 Wired Ethernet

Ethernet is a network technology commonly used in local area networks. ”A wired Ethernet connection can theoretically offer up to 10 Gb/s”[25], and the transmit speed is consistent. Wired networks are less expensive, faster, and more secure than wireless networks. However, the wired ethernet provides these advantages, it also has its biggest disadvantage, the immobility.

4.8.7 Discussion

Wired communication is incapable for our project while it provides the best transmit speed and stability. Our helicopter must use the wireless communication technology. The Bluetooth cannot give us enough transmit rate compare with the WiFi, the transmission rate can not able to support two videos streaming. The connectible distance of Bluetooth usually under 30ft. Our helicopter will travel and send data as far as 40ft, the connection will be unstable when the distance increases. WiFi can provide much more reliable and stable communication between the helicopter and the ground station.

4.8.8 Conclusion

We are using Raspberry Pi Zero W which including the wireless network module and Bluetooth. WiFi is the best choice as our communication technology to send and receive data. The network will be provided by the competition area or by our own network router.

4.9 Yingshi Huang's Role

My role is to learn from requirements and help to design and calculate data in our sub-team. I will focus my efforts on controller integration and communication methods.

4.10 Wireless Communication

4.10.1 Overview

According to the requirements from the competition, either a remotely-controlled or full-autonomous system is needed. Regardless of our choice here, a wireless communication system will be required for the micro-helicopter. Wireless communication not only can connect with one devices; it is capable to connect with multiple devices at the same time. Wireless communication is easy to access and convenient compared with other communications. In wireless communication, there are various choices like infrared communication, radio, blue-tooth, WiFi, mobile communication and so on.

4.10.2 Criteria

1) Wireless communication will be the bridge to exchange data between the vehicle and computer.
2) Micro-helicopter will receive data from the wireless communication
3) Wireless communication will be able to receive data from vehicle

4.10.3 Potential Choices

My current knowledge gives me only two options blue-tooth and Wi-Fi. However, the choices two options are not enough, so my research has provided more options like infrared communication, radio, mobile communication.

4.10.4 Infrared communication

Infrared communication is a high-speed short range wireless communication[26]. It needs to transfer data directly from devices to devices, but does not use a traditional network. It is relatively inexpensive, and data will not lose easily. The communication range is ten to thirty meters. The speed of transferring data is fast, with a maximum speed of 100 Mbps. However, the communication can only control one device at a time. Obstacles like doors, walls, bad weather will affect the infrared communication. Infrared communication will also provide a strong power which can be harmful to human body.

4.10.5 Bluetooth Radio

Bluetooth[24] is another example of short range communication between devices. The range of transmission range is within 10 meter in common devices. The distance range can manage by the power of the senders and the receivers. Bluetooth system operates in unlicensed 2.4-GHz frequency. Nowadays, bluetooth is a common tool for transferring data. The normal rate of transmission is 1 Mbps. It is a tool which has replaced the use of infrared communication.

4.10.6 Mobile communication

Cellular systems is the most common communications used by mobile cell. The mobile uses the signal from the signal towers. The area of the signal supply from the tower is like a cell. And there are many signal tower to combine as the whole system. Our team does not have rights to use the signal tower and the usage of the phone company can be expensive.

4.10.7 WiFi communication

WiFi is defined as an abbreviation for wireless fidelity. It is common to use for computer networking. If person has set up WiFi at a place, this person can access the computer network within a certain range of distance. The signal from the WiFi can get through the walls or doors, if they are made of woods. The WiFi communication can even send or receive signal from upstairs or downstairs. The WiFi communication can transmit data at 2.4 GHz frequency. The frequency of the communication can be change by devices. The average rate of transmission data is about 50 Mbps to 100 Mbps. It is easy to set up in a new place.

4.10.8 Discussion

2.4-GHz frequency is recommended from the rules of the competition[1]. So that it becomes one of the reasons that the cellular system has been excluded from the other wireless communications. The second main reason for excluding the cellular system is the budget of using it. Image processing and video processing are operations which need high-speed data transmit tunnel. Infrared communication is not as fast as what operations are expected.

4.10.9 Conclusion

According to the previous consideration, WiFi and bluetooth are better choices than others communications. Both of the communication is easy to purchase on the market and have similar price. Compare with the transmission of the data, WiFi has a higher speed. As a result, WiFi becomes better choice of communications.

4.11 Communication Protocol

4.11.1 Overview

Communications between three hardware: micro-helicopter and base station, base station and remote controller.

4.11.2 Criteria

1) Micro-helicopter will receive data from the base station and respond to movement.
2) base station will get information from the remote controller which provides data to the micro-helicopter for re-positioning.

4.11.3 Potential Choices

Based on our previous choice of WiFi for transmission method; UDP, TCP and Web-Internet will be considered as methods to connect hardwares.

4.11.4 UDP

UDP stands for User Datagram Protocol[27] and uses of data transferring. UDP will not acknowledge the receipt of the sending data. It means a disadvantage that no resends for lost data. On the contrary, an advantage is that save time without re-sending missing data. It is typical for immediate data such as online video, audio transmission, online gaming. Intermittent video or audio is an appearance of Lost data.

4.11.5 TCP

TCP stands for Transmission Control Protocol which is connection-oriented and re-transmittable. An advantage is able to resend data and acknowledge lost segments. A disadvantage is extending sending time. TCP is representative of sending guarantee data like Web-Page, files.

4.11.6 Web-Internet

Web-Internet is a tool using internet via the web to communicate. An advantage is that no need for storage in base station and able to calculate data on other devices. A disadvantage is that all data depend on transferring which requires unhindered signal transmissions.

4.11.7 Discussion

In theory, the computer will need lots of data to calculate distance and movement. Ensure data received is not as important as using data to calculate more rough distance, interrupted data is still value to estimate length and size of target and obstacles. Although 2.4GHz radio might be impacted, this communication is still able to send and receive data between woods and thin walls.

4.11.8 Conclusion

Due to the complexity of the systems being implemented on our project we will likely use a combination of these three technologies. Web-internet will be used for our GUI to interface with the vehicle. UDP will be used for the video-stream to the base-station, and TCP will be used as the transmission protocol between the base station and the vehicle.

4.12 Controller System

4.12.1 Overview

Here I will be analyzing the link between the human giving the commands to remote controller and the reactions of vehicle. Possible remote controllers can be keyboard, a game-pad, and a radio control transmitter.

4.12.2 Criteria

1) The remote controller can send the signals to micro-helicopter, and make the progressions of movements.

4.12.3 Potential Choices

4.12.4 Keyboard

Keyboard has more command keys available than the others. However, the joystick and radio control transmitter can also enough keys for our purposes. The advantage of the keyboard is that the keys are separated and precise. The disadvantage of that relationships is reactions and respond will slower be caused by pressing different keys.

4.12.5 Joystick

A Joystick is easy to learn how to use. However, it is hard to control the joystick with precisely. The advantage is less time of human reaction and easy use by different ranges of ages. The disadvantage is hard to control the micro-helicopter for small movements.

4.12.6 Radio control transmitter

Radio control transmitter is typical to use for micro-helicopter. Typically the manufacturers are selling micro-helicopter and radio control transmitters together. The advantage is a lower cost of material and higher profits. The disadvantage is larger and heavier.

4.12.7 Discussion

According to the requirement of movements of the micro-helicopter, the tunnel has enough space for movement and obstacle are able to avoid with a joystick. Too many connections are not accessible for the pilot to use in the tunnel which needs to avoid the obstacles and picking up target. Radio control transmitter has weaker signal and larger equipment.

4.12.8 Conclusion

Comparing the three possibilities; the keyboard has too many distractions. The Radio control transmitter will add too many weights for the micro-helicopter. So the better choice between increasing the weight and fewer distractions will be the joystick.

4.13 Autonomous System Data

4.13.1 Overview

Non-human control of micro-helicopter moving, pre-written code for distance calculations and exact movements depend on calculations.

4.13.2 Criteria

1) Correct calculations from information
2) Using calculations to make movements

4.13.3 Potential Choices

4.13.4 Data from sensor

Distance sensor is a small tool to get a certain range of distances directly. The advantage to use distance sensor is receiving data directly and no need to wait calculations. The disadvantage for distance sensor is vibration of the micro-helicopter will affect the precision ratio of the distance. Incorrect distance might cause a crash to obstacle or unable to pick up the target which will lead to failure.

4.13.5 Images from camera

Cameras can provide more information than distance sensor. However, more data as input will need more calculation. More calculation will need correspond libraries. And it will need more time and energy to finish the tasks.

4.13.6 Built-in Libraries

Software libraries will help the computer calculate data. Most of the formula are provided in libraries which is used for calculate. However, it require more storage and computer burning. It is a factor to slow down the result of calculations.

4.13.7 Acceleration Positioning

It is a method for calculating from the target or obstacles to micro-helicopter. It need time for computer to process, but all information can be handle by computer it, which mean fewer energy waste.

4.13.8 Discussion

Distance will be a tool to help estimate the distance calculation but not providing correct information. Accelerate positioning depends on the images from cameras, but not necessary depend on the software libraries. However, software libraries can reduce the coding error.

4.13.9 Conclusion

It is better to have correct data than not allowable error data which can lead failure. So using a camera to receive and calculate data is better than receiving direct data from sensor.

5 Weekly Blog Summaries

5.1 Justin Sherburne

5.1.1 Fall 2017

Week 1 - Set-up OneNote, and modify resume for submission.

Week 2 - Groups assigned, first meeting with project sponsor.

Week 3 - Tour of Columbia Helicopters, first group meeting.

Week 4 - Problem statement, initial project design and research.

Week 5 - Order initial equipment for CS, requirements document.

Week 6 - Begin setup of the Raspberry Pi Zero.

Week 7 - Technology Review document, Raspberry Pi streaming to laptop.

Week 8 - Technology Review, more video stream testing.

Week 9 - Thanksgiving week, no meetings or class. Revisiting old documents with new IEEE formatting.

Week 10 - Design document, camera multiplexer arrived for testing.

5.1.2 Winter 2018

Week 1 - Establish winter meeting times.

Week 2 - OpenCV research, multiplexer testing, sponsor approval for modified documents.

Week 3 - OpenCV research on implementation, multiplexer found to be unusable.

Week 4 - OpenCV initial implementation on web-cam. Airframe is still being built.

Week 5 - OpenCV prototype for multiple colors. Motor and ultrasonic testing for the aircraft.

Week 6 - Midterm report and poster rough draft due. March 16 deadline for limited autonomous function.

Week 7 - OpenCV tracking is finalized, begin work on Pi streaming into openCV.

Week 8 - Develop simple OpenCV GUI, including on-the-spot calibration. Streaming from the Pi to the Base station completed.

Week 9 - Beginning to adapt OpenCV to autonomous flight calculations. Helicopter is flying for the first time.

Week 10 - Begin development of the Autonomous flight system. Aircraft motors are experiencing too much current and need replaced. AHS video of competition readiness.

5.1.3 Spring 2018

Week 1 - Autonomous flight progress. Meeting times established. New motors were delayed in shipping.

Week 2 - Withdraw from AHS competition due to motor failure. Continue work on autonomous flight.

Week 3 - Autonomous corner detection is working. New motors have arrived and will begin installation.

Week 4 - Autonomous controls output begins development. Final Expo poster submission.

Week 5 - Modified corner detection for reliability. Autonomous controls established. Helicopter is being re-balanced after motor installation.

Week 6 - Autonomous algorithm development. Flight testing with new motors.

Week 7 - Finalize autonomous flight algorithms prior to code cut-off. Autonomous flight will not be tested on this airframe.

Week 8 - Expo.

Week 9 - Finalize documentation and project presentation.

Week 10 - Finalize documentation and project presentation.

5.2 Kaiyuan Fan

5.2.1 Fall 2017

Week 1 - Create OneNote structure, biography and handed in the resume.

Week 2 - Project was assigned, meet with the group mates and the sponsor.

Week 3 - Tour of Columbia Helicopters, start weekly meeting with TA.

Week 4 - Finish the Problem statement final draft and get verified e-mail.

Week 5 - Meeting with Anna from Columbia helicopter, talked about the funding and design change.

Week 6 - Get approved of our problem statement and ordered equipment online.

Week 7 - Start working on the tech review and install OS for the Raspberry Pi.

Week 8 - Finished Tech review this week and implemented video stream on the Raspberry Pi board.

Week 9 - Thanksgiving week. Attend meeting with ME and EE team on Monday and submit the Tech review on November 21.

Week 10 - Finish the design document within the group and submit on Friday.

5.2.2 Winter 2018

Week 1 - Establish winter meeting times.

Week 2 - First meeting with ME and EE team, discuss the project details.

Week 3 - Big group meeting talks about the team name, document due Jan 31st and equipment funding.

Week 4 - Team name ”Beaverhawk” was made.

Week 5 - Made decision on schedule of the competition. Start to work on the poster and demo.

Week 6 - Wired a DC motor on a breadboard with a power module and motor control board L293D.

Week 7 - Midterm report and presentation. Tested the change direction of DC motor by setting two GPIO pins output LOW and HIGH.

Week 8 - Made a template for motor control in CPP. Sent an intent of video delivery to Dr.Squires.

Week 9 - Did a feature on the user interface, read the user input by keyboard of HTML and pass the command to the Python script to control helicopter motors.

Week 10 - Build EE team motor control code in and add gamepad support to the remote web control system. Record the competition video.

5.2.3 Spring 2018

Week 1 - Establish Spring meeting times. Expo registration.

Week 2 - Did not meet the bar for Gate 2 of the competition, project object change.

Week 3 - New motors are delivered and start the installation.

Week 4 - Added prevent key repeating feature in manual flight code. Tested keyboard event code on Windows.

Week 5 - Added set motor servo back after turning feature in manual flight code.

Week 6 - Submitted midterm progress report and the presentation. Code freeze on Friday.

Week 7 - Tested flight with new motors, have slipping shaft problem. Expo on Friday.

Week 8 - Expo has done.

Week 9 - Finalize documentation and project presentation.

Week 10 - Finalize documentation and project presentation.

5.3 Yingshi Huang

5.3.1 Fall 2017

Week 1 - Read the project and follow OneNote pattern, put biography in file

Week 2 - Project assigned by instructor

Week 3 - Weekly meeting and tour

Week 4 - Problem statement and signature

Week 5 - Sponsor, funding and budget

Week 6 - Research and order equipment

Week 7 - set up equipment and started gitHub

Week 8 - Tech review document and camera

Week 9 - Big group meeting

Week 10 - Submit document and Term review

5.3.2 Winter 2018

Week 1 - Set up meeting time as early as possible

Week 2 - First detail meeting with ME and EE members

Week 3 - Team name and flyer

Week 4 - Testing and flying, competition readiness video is closing

Week 5 - Every member works on competition readiness video

Week 6 - EE team works on equipments connection

Week 7 - Midterm report and presentation

Week 8 - Joystick testing

Week 9 - Second week of joystick testing

Week 10 - Final report and presentation and Whole team final report

5.3.3 Spring 2018

Week 1 - Review last team work and set up meeting with ME team and EE team

Week 2 - Due to late equipments delivery, cancel enter competition

Week 3 - Two language communication found by python extension library

Week 4 - Build up environment of simple statement exchange

Week 5 - Second test of new helicopter

Week 6 - Motor test

Week 7 - switch assignment from language to autonomous

Week 8 - Expo. GitHub filled.

Week 9 - Documation and presentation.

Week 10 - Documation and presentation

6 Final Poster

Figure 2. Expo Poster, Final Draft

7 Project Documentation

7.1 Project Design

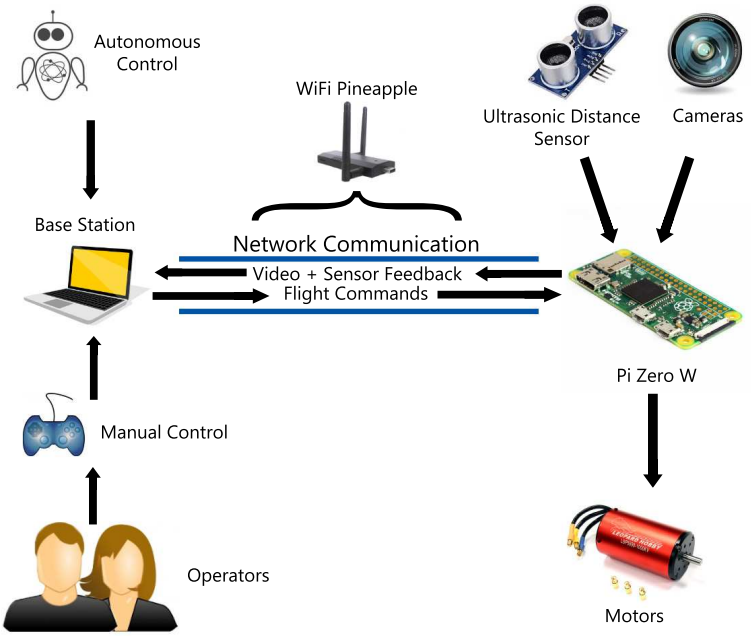


Figure 3. Conceptual Design Overview

For our design we have two distinct systems: The Base Station, and the Raspberry Pi Zero. The Base Station controls the autonomous flight systems, serves as the link for manual control, hosts the GUI, and also runs the image processing system. The Raspberry Pi is the brains of the aircraft; operating the motors, ultrasonic sensors, and streaming the video feed back to the Base Station. These two systems are connected over a standard WiFi connection which can be set up a number of different ways. In our testing we used a device called a WiFi Pineapple, which acts as a short-range, compact wireless router. Above in Figure 3 you can see the overall architecture of our project.

7.2 Software Installation

In this section we will elaborate on how to install and use the software we have developed for this project. We will break the software into two components: The Base Station and the Raspberry Pi. While both devices are used together in this project, the software installation is not the same due to computational limitations on the Raspberry Pi.

7.2.1 Base Station Software

To begin, the base station is simply a computer running a Linux-based operating system. In our case we used a laptop, but a desktop with a monitor could also be used if more processing power is required. The operating system we chose (and that these instructions will be based upon) is Ubuntu. Ubuntu is one of the most supported Linux operating systems, making is ideal for our project. Instructions and additional documentation can be found at https://www.ubuntu.com/.

Once you are running your Linux distro, you will need to install OpenCV. OpenCV is the framework used for the image processing application, and will occupy nearly 2gb of hard drive space. The installation is fairly complex and may

require additional troubleshooting outside of the scope of this document. For detailed installation instructions I would recommend following this guide: https://www.learnopencv.com/install-opencv3-on-ubuntu/ . This step may take up to 20 minutes. To verify that OpenCV has installed correctly I would recommend running some of the included samples or to try our image recognition program directly.

Xdotool is required for the autonomous flight output. It is a Linux-only library and will not work on Windows or OSX. There is a similar library in Windows called Windows.h, however that library is incompatible with Linux. To install Xdotool, simply run: `sudo apt-get install xdotool` in the terminal. Additional resources for Xdotool can be found here: http://xmodulo.com/simulate-key-press-mouse-movement-linux.html .

Lastly, you will need a different web browser than Firefox. The GUI API does not work correctly with the Firefox web browser that is built-in to Ubuntu. Instead we used the Chromium web browser which can be installed via the Ubuntu Software center.

The code developed for this project is located on our git-hub repository: https://github.com/OSU-Capstone-MAV-Challenge/CS_MAV_Challenge. This can be cloned to a directory using the following command:

```
git clone https://github.com/OSU-Capstone-MAV-Challenge/CS_MAV_Challenge
```

You will find the required software for the Base Station in /SourceCode/Base Station/.

7.2.2 Raspberry Pi Software

On the Raspberry Pi we installed Rasbian OS. This is a lightweight version of Linux specifically designed for the Raspberry Pi. It includes native support for all of the Raspberry Pi Zero hardware. Installation instructions can be found here: https://www.raspberrypi.org/documentation/installation/installing-images/README.md . Once installed, it is recommended to remove any large libraries not being used in this project like Math-lab, and Wolfram Alpha. Removing these libraries, in addition to a high-speed SD card will drastically improve startup time.

The following packages will need to be installed on the Raspberry Pi Zero:

PHP 7.0 & Apache2 - Should come pre-installed, follow the following link for additional support: https://www.stewright.
raspberry-pi-3-php-7-powered-web-server/
Tornado Web Framework - Installation instructions: http://www.tornadoweb.org/en/stable/
RPIO - Setup Instructions: https://pythonhosted.org/RPIO/
mjpg streamer - Documentation found: http://sourceforge.net/projects/mjpg-streamer/ or in our git-hub repository.

You will also need to clone the project's git-hub repository onto the Raspberry Pi. We will be accessing the files located under /SourceCode/RaspberryPi/.This can be done using the following command:

```
git clone https://github.com/OSU-Capstone-MAV-Challenge/CS_MAV_Challenge
```

7.3 Software Initialization

While both systems must be powered on and connected to a local area network, the Raspberry Pi Zero must be configured first. During testing the Raspberry Pi can be initialized through a local terminal and the GUI if connected

to an external display. This method will reduce performance, and should be avoided. Instead the configuration will be done through a SSH connection from the base station to the Raspberry Pi. In the future a script could be developed to run at startup to initialize these systems without needing to SSH to the Raspberry Pi.

7.3.1 Raspberry Pi Zero

To initialize the manual flight system, connect Raspberry PI and laptop in a same local area network. SSH into the Raspberry Pi using Putty or a terminal connection. Navigate to the correct subdirectory in the github repository and type `sudo python MotorControl.py`. This initialized the motor control server. To initialize the video streaming server navigate to /SourceCode/RaspberryPi/Streamer/mjpg-streamer/mjpg-streamer-experimental/. In this directory run the startup script to initialize the stream: `startup.sh`.

7.3.2 Base Station

The Index.html file contains the GUI for controlling the aircraft. This can be used with either keyboard input, an Xbox controller, or by manually clicking each button on screen. Simply right-click and open with your desired web-browser.

To run the image processing software, navigate to /SourceCode/Base Station/OpenCV/. Here you can type `make` to compile the program, and `Vision` to run. On line 843 of `main.cpp` there are some toggles to turn on and off various features prior to compiling the program.

> **calibrationMode** - Enables you to reset individual color settings on-the-go. Leaving it enabled with display the trackbars for re-calibrating colors and the filtered video feed; disabling it will hide both of those screens. This feature can also be toggled off at runtime using the Hide Calibration slider.
> **webcammode** - Enabled allows you to use your built-in web-cam for troubleshooting. Rather than needing to connect to a remote camera, it will default to your web-cam video feed. Disabling this will prompt you to connect to a remote camera. In our case this will be the Raspberry Pi. If the video streaming software is already running, you only need to type the IP address of the Raspberry Pi, or the host name of the device.
> **threadded** - true enables the autonomous flight calculations and commands, false disables the autonomous flight functionality. Autonomous calculations can also be stopped after the program has been started by using the autonomous slider in the main video window.

Once the image processing software is running you can click back to the GUI and the autonomous commands will output directly to the web-page. To exit you must type `CTRL+C` to the terminal, or toggle the track-bar labeled kill. To switch to manual flight controls toggle the track-bar labeled autonomous.

8 TEAM RETROSPECTIVE

8.1 Justin Sherburne

8.1.1 Technical Information Learned

During the course of this project I learned a lot about the OpenCV library. It is a powerful tool for image processing applications, but can be just as difficult to learn as a new programming language. While I would have enjoyed diving a little deeper into OpenCV systems, there is simply too many features to try and learn in a term or a year. In addition to OpenCV I also learned about autonomous flight algorithms. While I was developing the flight system for our helicopter I quickly realized how complex an autonomous system really is. On paper it seems simple; given a specific input, output

a direction of movement. In practice there are so many outside factors that you have to try and account for that make it non-trivial. Given autonomous vehicles only have to worry about two dimensions of travel, I grossly underestimated the complexity of this portion of the project.

8.1.2 Non-technical Information Learned

I learned a lot this year about team management and communication. Our team faced many obstacles from both outside and inside sources. Being in an interdisciplinary project has given me great experience in project management. In addition to mt experience in the team, I also learned a great deal about project documentation. Using OneNote was a great way to stay organized and place all of my thoughts and research into one location.

8.1.3 Experiences: Project Work

The amount of documentation that went into this project was unexpected. While I understand the need for elaborate planning for major undertakings like these projects, I believe there are certain parts that are unnecessary.There is quite a bit of redundancy in each document that covers aspects that have already been addressed. Additionally, I do not believe the technology review was adequate research for the project. Personally I spent several hours researching OpenCV and image processing technologies for our project, but my technology review did not reflect that accurately. Instead wrote 2 pages on operating system options, something that is a relatively trivial choice for our application.

8.1.4 Experiences: Project Management

Project management was arguably the most important part of this project, and also one area we struggled with. Having an interdisciplinary project means there needs to be a steady stream of communication between all members of the group. For our team we used Facebook as our primary method of communicating, but not everyone checked our page each week. Posts would occasionally go unnoticed or ignored. A better method for communicating would have been google hangouts, a group text, or even some combination of e-mail and Facebook.

8.1.5 Experiences: Teamwork

Teamwork is another area that our team struggled with this year. A significant portion of this project fell heavily on one or two members of this team at a time, and work was never evenly distributed. I think part of the issue stems from not having clearly defined roles for each team member, and a lack of time management. During the course of this project we acted as individuals contributing to a common goal rather than a team. This means that large portions of the project or documentation were left to an individual as the rest of the team viewed that section as their responsibility or area of expertise.

8.1.6 Things to do differently:

From a non-technical perspective, communication. Many of our problems could have been mitigated with clear communication from start to finish. Defining clear roles for each member is beneficial, but ideally any member should be able to take over development of any section at any point in time. Staying up-to-date with project progress is critical for the success of a team, and I think it also helps motivate each member. Knowing that your teammate is working and actively contributing to your project is an effective motivator for many teams. Communication allows you to keep your teammates up to date or to clue someone in that there is a meeting on Monday morning.

From a technical perspective we could have improved in many areas. Ideally, our image recognition software should be identifying shapes and features using machine learning, or object recognition rather than color. Our autonomous flight algorithm also could have used 2-3 more weeks of development with actual flight testing. One problem that plagued our group was our use of three separate languages in our project. Trying to get all three to work effectively together significantly impacted development. Lastly, knowing how the project progressed this year I would have placed more pressure on the Mechanical Engineering team to finish development. The Computer Science component of this project is the last to be implemented, and on both major deadlines we were given the aircraft with only a week to troubleshoot and get our systems running.

8.2 Kaiyuan Fan

8.2.1 Technical Information Learned

The main technical information I have learned is using a microcomputer board to deliver a practical product. I learned programming on Raspberry Pi to host a web server and browser the webpage remotely to access the Raspberry Pi GPIOs. Through the previous nine months, I have learned to establish a webpage with an open source web framework to communicate with the script on Pi. I have learned to use AJAX to reduce the traffic travels between the client and the server, and I learned using the API in HTML which I have never been used before. In addition, I have learned how to interface different programming languages.

8.2.2 Non-technical Information Learned

The non-technical information I have learned is how to effectively do the research of the unfamiliar topics and properly communicate with the supervisors and my peers. Besides, I have learned to make OneNote blogs to keep track of working logs, and I also got benefits from making various documents.

8.2.3 Experiences: Project Work

The whole project is a long getting new problem and finding solution path. For this challenging project, we start with empty and have overcome many obstacles. We were facing trouble interfacing different programming languages. we got motor issues and there are problems occurred during the flight test. What I have learned is to solve the problems with the knowledge I have and the resources I can find.

8.2.4 Experiences: Project Management

We have our weekly meeting to keep track of the progress of the project. Each of us has the different focus on developing the project, we managed in this way to make steady progress.

8.2.5 Experiences: Teamwork

Through the three terms, we were collaborating with two electrical and three mechanical engineering members. We have set up weekly big group meetings and our CS sub-team meetings. Working in a team helps me learn. Our project is multi-disciplinary, so teamwork performed a significant role in our project. ME team worked on manufacturing and balancing the vehicle, EE team worked on the wiring and motor control script, and our CS team was implementing the autonomous and manual flight systems. We have encountered hardware delay problem by other teams, our CS team could not have sufficient time to test our implementation. We decided to do as much as possible before the hardware available. What I have learned is when we have negative factors within the team, each member should try their best to reduce the negative impact.

8.2.6 Things to do differently:

If I could do it all over, I would be more proactive. I would communicate more often with each other, so we could come out a testable aircraft sooner. I also would like to work on the image recognition portion since I hardly touch this part. In addition, I would like to deliver a more beautiful GUI and add the onboard camera streaming feed into it.

8.3 Yingshi Huang

8.3.1 Technical Information Learned

I have learned using different computer languages which are python and c++ to receive data from a joystick. I have also learned to how to control the pin on the raspberry pi w zero. And I believe raspberry pi is similar to many on market bands like banana pie and so on. The hardest part I learned from the last few terms is to connect two different computer languages. To send data from python to c can use by python extension library. It is an enrichment library, and you can learn a lot from this library. There is much information in it. I will continue to learn from python extension library.

8.3.2 Non-technical Information Learned

I have learned how to use LaTex and shared documentation with others. I also learn how to read lengthy technical documentation and technology guide. Not only learn how to read them I have learned how to write the technical review, verify email, and much other documentation. Not only skills related to academic but also communication and noting ability.

8.3.3 Experiences: Project Work

In the project everyone needs to understand not every part of the plans will fit the procession. In work, the project might be getting into some unexpected, and the problems cannot solve in a long time. It is a reason for the whole team to understand and find another way to cross the obstacles. If the team only have one plan, it is possible to face failure. The project is better to have a second plan and able to extra time to solve unexpected problems.

8.3.4 Experiences: Project Management

Our team did not have proper management, so a lot of works did not have enough time to finish. Excellent project management can help the whole group to work on the project and with better quality. All the assignments must complete by submit date, and each person is better to hand in the job early.

8.3.5 Experiences: Teamwork

For good teamwork, it is crucial to communicate smoothly with other members. Not only talk with your teammates in a right emotion but also make sure your teammate can understand what is your expression otherwise it is a wordless communication. And if your team want to finish the project in a better way, it is good to assign tasks to each member so that people can work on different parts and the project complete by the whole group on time.

8.3.6 Things to do differently:

Do more research before work on the project. The reason to do more research is that I had worked on the project but there is another more straightforward way to figure out the problem, but I missed it. According to the mistake, I understand that do more research can help yourself to know more detail and solve problems faster. Ask for more information from the instructor, be brave to talk with someone who had learned more from me.

9 Appendix A: Essential Code Listings

9.1 Team Git-hub Repository:

https://github.com/OSU-Capstone-MAV-Challenge/CS_MAV_Challenge

9.2 Kyle Hounslow's OpenCV tutorials:

Video Part 1: https://youtu.be/RS_uQGOQIdg

Video Part 2: https://youtu.be/ASCi7J5W1FM

Video Part 3: https://youtu.be/4KYlHgQQAts

Part 3 Code:

https://www.dropbox.com/s/o22cnih7v0mu7gv/multipleObjectTracking.cpp?dl=0

https://www.dropbox.com/s/rck1v0gjv0y1thy/Fruit.h?dl=0

https://www.dropbox.com/s/2bvdrcnlno878s8/Fruit.cpp?dl=0

9.3 Raspberry Pi Resources:

Rasbian - https://www.raspberrypi.org/documentation/installation/installing-images/README.md-

PHP 7.0 & Apache2 - https://www.stewright.me/2016/03/turn-raspberry-pi-3-php-7-powered-web-server/

Tornado Web Framework - http://www.tornadoweb.org/en/stable/

RPIO - https://pythonhosted.org/RPIO/

mjpg streamer - http://sourceforge.net/projects/mjpg-streamer/ or in our git-hub repository.

9.4 Base Station Resources:

OpenCV - https://www.learnopencv.com/install-opencv3-on-ubuntu/

Ubuntu - https://www.ubuntu.com/

Xdotool - http://xmodulo.com/simulate-key-press-mouse-movement-linux.html

REFERENCES

[1] (2017) 2018 micro-air vehicle challenge competition rules. American Helicopter Society. [Online]. Available: https://vtol.org/education/micro-air-vehicle-student-challenge/micro-air-vehicle-student-challenge-2017

[2] (2016) Darpa's fully-loaded quadcopter autonomously navigates an indoor maze at 45 mph. [Online]. Available: https://newatlas.com/darpa-drone-autonomous-45-mph/41810/

[3] (2017) Raspberry pi zero w. Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/

[4] (2017) Wifi pineapple. Hak5 LLC. [Online]. Available: https://www.wifipineapple.com/

[5] (1999-2015) Robotics academy - ultrasonic sensors. Carnegie Mellon. [Online]. Available: http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html

[6] N. PHILIP. (JULY 11, 2017) How to begin with raspberry pi gpio programming using python. [Online]. Available: http://opensourceforu.com/2017/07/introduction-raspberry-pi-gpio-programming-using-python/

[7] Senix. (2015) Distance measurement sensor applications. [Online]. Available: https://senix.com/distance-ranging/

[8] A. Rosebrock. (2015) Find distance from camera to object/marker using python and opencv. [Online]. Available: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/

[9] (2017) Arduino mkr zero. Arduino AG. [Online]. Available: https://store.arduino.cc/arduino-mkrzero

[10] (2017) Raspberry pi 3 model b. Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[11] R. Feranec and M. Murin. (2010 - 2017) Openrex. [Online]. Available: http://www.imx6rex.com/open-rex/

[12] (2017) Raspbian. Raspberry Pi Foundation. [Online]. Available: https://www.raspbian.org/RaspbianAbout

[13] (2017) Microsoft win 10 iot core. Microsoft, Inc. [Online]. Available: https://developer.microsoft.com/en-us/windows/iot

[14] (2017) Ubuntu. Canonical Ltd. [Online]. Available: https://www.ubuntu.com/

[15] (1994 - 2017) Mathworks - image processing toolbox. Mathworks, Inc. [Online]. Available: https://www.mathworks.com/products/image.html

[16] (2017) Opencv. OpenCV Team. [Online]. Available: https://opencv.org/

[17] C. Stachniss, U. Frese, and G. Grisetti. (1999 - 2016) Openslam. [Online]. Available: http://openslam.org/

[18] A. Goldlsmith. (2005) Wireless communications.

[19] G. Benet, "Robotics and autonomous systems," vol. 40, pp. 255–266, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889002002713

[20] RaspberryPiResources. (2013) Streaming video using vlc player. [Online]. Available: http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-camera/streaming-video-using-vlc-player

[21] SourceForge. (2014) Mjpg-streamer. [Online]. Available: https://sourceforge.net/projects/mjpg-streamer/

[22] Elinux. (2017) Rpi-cam-web-interface. [Online]. Available: https://elinux.org/RPi-Cam-Web-Interface

[23] Seattlecentral. (2017) Wi-fi. [Online]. Available: http://resources.seattlecentral.edu/~ymoh/mic265/WiFi/WiFi.html

[24] W. Stallings. (2001) Introduction to bluetooth. [Online]. Available: http://www.informit.com/articles/article.aspx?p=23760

[25] C. Hoffman. (2017) Wi-fi vs. ethernet: How much better is a wired connection? [Online]. Available: https://www.howtogeek.com/217463/wi-fi-vs.-ethernet-how-much-better-is-a-wired-connection/

[26] J. B. J.M. Kahn. (Feb 1997) Wireless infrared communications. [Online]. Available: http://ieeexplore.ieee.org/document/554222/

[27] (August 1980) User datagram protocol. [Online]. Available: https://www.ietf.org/rfc/rfc768.txt