CS Capstone Design Document

April 20, 2019

American Helicopter Society Micro Air Vehicle Competition

Prepared for

Columbia Helicopters

Nancy Squires _____ Signature _____ Date

Prepared by

Group 14

Beaver Hawks

Anton Synytsia _____ Signature _____ Date

Matthew Phillips _____ Signature _____ Date

Shanmukh Challa _____ Signature _____ Date

Nathan Tan _____ Signature _____ Date

Abstract

This design document outlines the overall software architecture of Oregon State University's Micro Air Vehicle (MAV) system. It outlines the data the software will receive from the MAV's onboard transmitter. Next, the document covers how the received data is processed, organized, and used by each software block. The data pipeline is described in detail, from the origin source, through each sorting and processing step, and finally to useful information displayed for pilot consumption. The design and layout of the GUI is also discussed, including the value of the displayed information to the pilot. Each facet of the MAV's software is covered.

Contents

1 Introduction 5

1.1 Scope . . . 5
1.2 Purpose . . . 5
1.3 Intended Audience . . . 5
1.4 Definitions and Acronyms . . . 5
1.5 References . . . 5

2 System Overview 6

3 System Architecture 6

3.1 System Architecture Overview . . . 6
3.2 Helicopter . . . 7
3.3 Computer . . . 8

4 Component Design 8

4.1 Collision Warning . . . 8
4.1.1 Introduction . . . 8
4.1.2 Structure . . . 8
4.1.3 GUI Element . . . 9
4.1.4 Design Rationale . . . 9
4.2 Height Indicator . . . 9
4.2.1 Introduction . . . 9
4.2.2 Structure . . . 9
4.2.3 GUI Element . . . 9
4.2.4 Design Rationale . . . 10
4.3 Attitude Indicator . . . 10
4.3.1 Introduction . . . 10
4.3.2 Structure . . . 10
4.3.3 GUI Element . . . 10
4.3.4 Design Rationale . . . 10
4.4 Heading Indicator . . . 10
4.4.1 Introduction . . . 10
4.4.2 Structure . . . 11
4.4.3 GUI Element . . . 11
4.4.4 Design Rationale . . . 11
4.5 Speed Indicator . . . 11
4.5.1 Introduction . . . 11
4.5.2 Structure . . . 11
4.5.3 GUI Element . . . 11

4.5.4 Design Rationale . . . 11
4.6 Front Video Feed . . . 12
4.6.1 Introduction . . . 12
4.6.2 GUI Element . . . 12
4.6.3 Design Rationale . . . 12
4.7 Bottom Video Feed . . . 12
4.7.1 Introduction . . . 12
4.7.2 GUI Element . . . 12
4.7.3 Design Rational . . . 12

5 Graphical User Interface 12

List of Figures

1 Overall System Software Architecture . . . 6
2 Helicopter Software Architecture . . . 7
3 Computer View Software Architecture . . . 8
4 Collision Warning Flow . . . 9
5 Collision Warning GUI . . . 9
6 Attitude Indicator . . . 10
7 Attitude Indicator . . . 11
8 Graphical User Interface Mock-up . . . 13
9 Graphical User Interface . . . 13

Section	Original	New
	Apportioning of work section made references to portions of removed sections.	Apportioning of work reflects current document version.
1.1	Reason for project work was ambiguous.	Which competition the team is participating in is specified.
1.4	Contained definitions no longer referenced.	Removed unused definitions.
4.1	Section 4.1 referenced to image recognition. Did not implement.	Removed section.
4.3	Section 4.3 referenced to collision avoidance. Did not implement.	Removed section.
4.3.3	Section 4.3.3 referenced to remote kill switch. Implemented by ECEs.	Removed references.
4.5	Section 4.5 referenced to pickup guidance. Did not implement.	Removed section.
4.10	Section 4.10 referenced to LED control. Did not implement.	Removed section.
4.11	Section 4.11 referenced to speaker control. Did not implement.	Removed section.

Apportioning of work:

Every team member contributed in all portions of this document. Specific contribution tasks are listed below.

Nathan Tan contributed to:

- System Architecture
- Figures
- User Interface Overview

Anton Synytsia contributed to:

- Collision Warning
- Height Indicator
- Attitude Indicator
- Heading Indicator
- Speed Indicator
- Front Video Feed
- Bottom Video Feed
- Graphical User Interface

Matthew Phillips contributed to:

- Abstract
- Formatting/Proof Reading
- Design Component
- Definitions and Acronyms
- System Overview

Shanmukh Challa contributed to:

- Abstract
- Introduction
- Scope
- Purpose
- Intended Audience

1 Introduction

1.1 Scope

The following design document will outline the overall design of the software features for the remote-controlled helicopter. Our goal is to develop a software application that will allow the pilot to successfully navigate through the Vertical Flight Society's Micro Air Vehicle Challenge obstacle course.

1.2 Purpose

This design document will lay out the technical specifications for each component and will expand on the implementation details. The document will outline the decision choices and implementation plans of the software.

1.3 Intended Audience

This document is meant to inform our client, as well as, the Mechanical Engineering team and the Electrical Engineering team of the software design plans so the Raspberry Pi on-board the aircraft can be structured to implement our hardware requirements.

1.4 Definitions and Acronyms

- **GUI**: Graphical user interface.
- **Micro Air Vehicle (MAV)**: A remotely controlled, semi-autonomous, coaxial helicopter.
- **OSU**: Oregon State University.
- **Package**: A sealed paper lunch bag, containing a pamphlet. A braided wire loop is attached at the top of the bag for acquirement.
- **Package A**: A package weighing between 20 and 25 grams.
- **Package B**: A package weighing between 25 and 30 grams.
- **RMC**: Remote computer processing data and providing visual output display.
- **VFS**: Vertical Flight Society.

1.5 References

[1] Vertical Flight Society. (2018). *7th Annual VFS Micro Air Vehicle (MAV) Student Challenge* [PDF file]. Retrieved from https://vtol.org/files/dmfile/7th-annual-mav-student-challenge_v7_oct182018.pdf?fbclid=IwAR3-J_yJIKUKoGPBfsdJFAbjMUzXoxwg1hCXiQi6JnWZRLOnSMBRnMmQGKU.

[2] Standford CS-231. (2018). *Convolutional Neural Networks for Visual Recognition.* Retrieved from http://cs231n.github.io/convolutional-networks/.

[3] Federal Aviation Administration. (2018). *Helicopter Flying Handbook.* Retrieved from https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/media/hfh_ch03.pdf.

[4] jQuery Flight Indicators. (2019). *Flight Indicators - jQuery Plugin.* Retrieved from https://github.com/sebmatton/jQuery-Flight-Indicators.

2 System Overview

The MAV, the software running on the Raspberry Pi, which is onboard the MAV, and the software receiving and processing the MAV data running on a remote computer work together to display data for the pilot to complete an obstacle course competition held by VFS. The software onboard the MAV and the remote software aim to provide maximum value to the pilot during the competition.

3 System Architecture

In this section a high-level overview of the software architecture of our helicopter and other systems is described.

3.1 System Architecture Overview

The software layout for our helicopter will consist of two major block with three external types of sources providing input to our system. The first block of software is run on a Raspberry Pi which operates the helicopter. The next large block represents the code that runs on RMC, which will run a server and a website. There are also three types blocks that will contain no code written by our team, but will send data to our helicopter, one representing the feed from our two cameras, one representing all other types of sensors, and the remote used to pilot the helicopter. The structure is represented in the figure below where the camera feed, sensor feed, remote, and computer send data to the helicopter and the helicopter sends data to the computer.

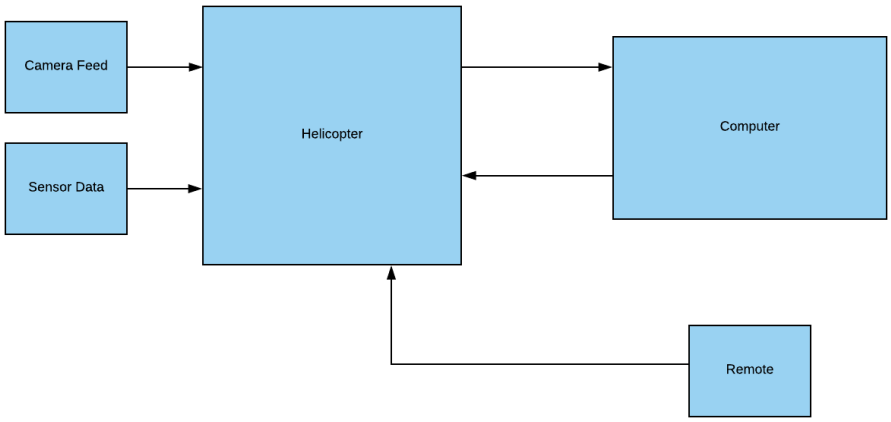

Figure 1: Overall System Software Architecture

The responsibilities of the helicopter is to fly through the obstacle course and send data back to the computer. As a part of flying through the course, we want to implement basic collision warning system for the pilot.

The computer has the capability to transmit data to the helicopter, but this feature is currently unused as the pilot will solely operate the movement of a helicopter with a remote.

The cameras, sensors, and computer are the only external systems that will send information to our helicopter module in the Raspberry Pi. The remote will send signals to the helicopter, but those signals will not be used by any of our modules.

3.2 Helicopter

Our helicopter, operated by a Raspberry Pi Zero and will contain software made in part by us, the Computer Science subsection of the team, and the Electrical Engineer subsection of the team.

The diagram below, figure 2, outlines the intended structure of the helicopter's software. The Raspberry Pi only receives inputs from various sensors, and computer signals from a Wifi transmitter. Outputs are limited to sending signals to handle motor control, and sending sensor and camera data back to the computer. The collision avoidance blocks are unused by our software, but the hooks are still available on the Raspberry Pi. The block will simply pass the data through without modification. Also, the LEDs and speakers were removed due to weight constraints.

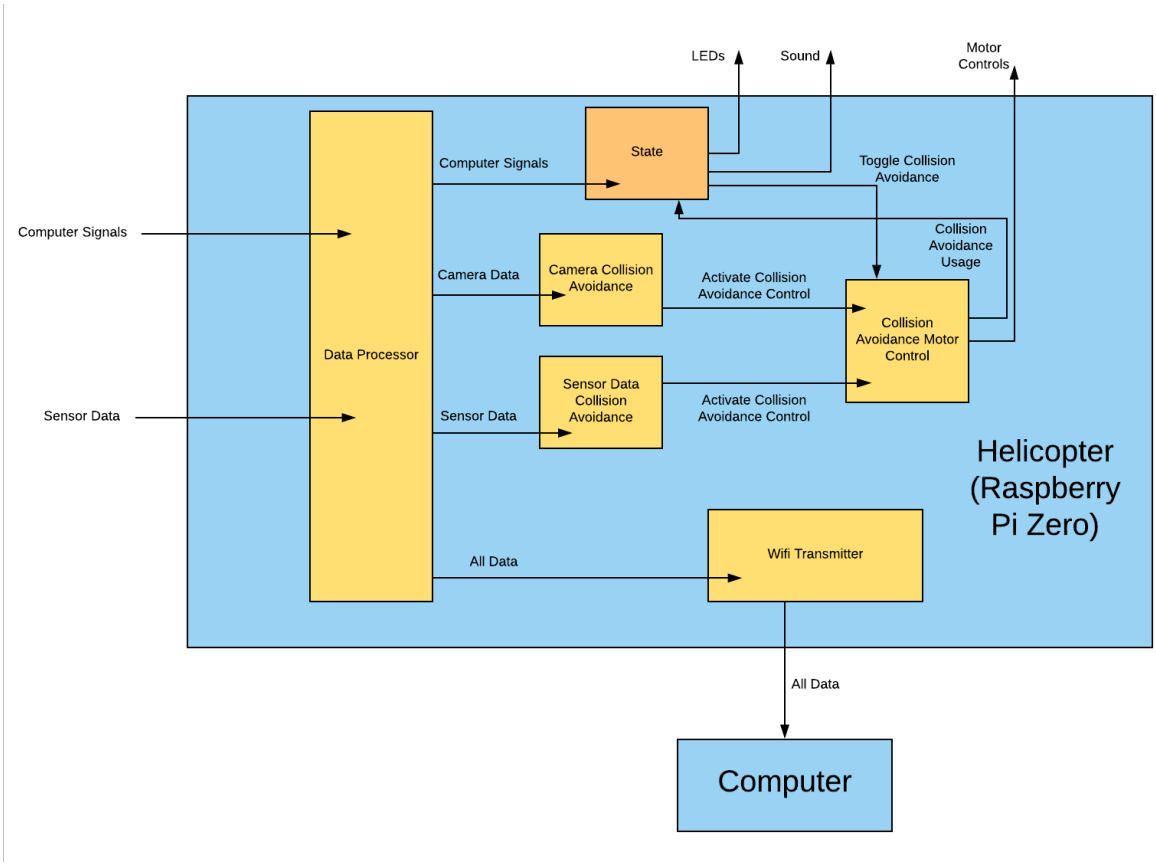

Figure 2: Helicopter Software Architecture

On-board the helicopter, all incoming signals start by entering into a data processing module which ensures that all incoming data is encapsulated in a format the rest of the system can interpret as opposed to possibly a stream of bytes. The data processor is also responsible for keeping the data sorted and outputting the data in an appropriate, timely fashion. The outgoing data will be sent to four different locations, a state module, a collision avoidance module, based on camera data, a collision avoidance module, based on sensor data, and a transmission module which will be responsible of forwarding data on the helicopter to a computer. The state module tracks if the Raspberry Pi has received a proper handshake from the computer and if the Raspberry Pi should be broadcasting or not. It also tracks the orientation, speed, and height of the

helicopter and modifies the state based upon sensor inputs.

To send the camera feeds and various sensor data to the computer the data processor will send the data to the transmitter module. There the data will be sent to a transmitter through an API provided by the Electrical Engineer subsection of our team.

3.3 Computer

Our computer view's main responsibility is to presenting the camera feeds and flight sensor data to the user. The data coming off the helicopter will be sent to a server running locally on the computer. The server calculates the proper display colors for the collision warning system. The path for the control signals is built, but unused in the software.

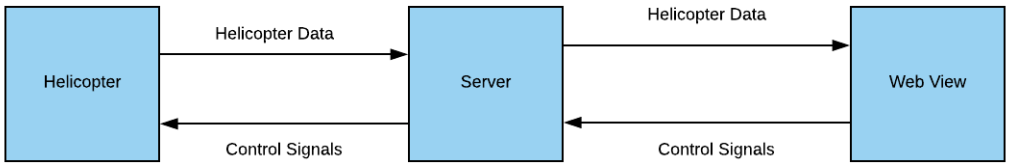

Figure 3: Computer View Software Architecture

4 Component Design

The following section describes the required software and design components.

4.1 Collision Warning

4.1.1 Introduction

Collision warning enhances pilot's awareness of the surroundings, including outside the field of view provided by the camera. To inform the pilot of obstacles surrounding the perimeter of the MAV, we implement a collision warning system.

4.1.2 Structure

The system involves three *LV-MaxSonar-EZ0* ultrasonic range sensors, with one at the front and two at the sides of the MAV. Due to the scarcity of available power and an imposed mass limit requirement, a range sensor at the rear of the MAV is not included. The range sensors detect obstacles within a conical-like field of view. Whenever the MAV is close to the ground, there is a chance for the ground itself to be treated as an obstacle. Using a bottom facing range sensor will allow for temporarily turning off collision warning system whenever the MAV is near the surface of the terrain.

Collision warning system is established at the server side. Range sensor data received from the MAV is processed at the server side and reflected in the graphical user interface, as noted in figure 4. Each pie slice represents the range of the associated ultrasonic range sensor. The range of each ultrasonic sensor, which varies between 6 and 254 inches, is converted to a color. The colors transition from green with the range being the farthest, to yellow with the range being intermediate, and to red with the range being proximal.

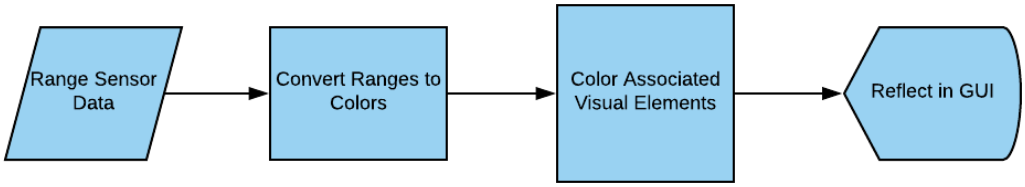

Figure 4: Collision Warning Flow

4.1.3 GUI Element

The GUI consists of a top view of the MAV, surrounded by colored semi-pie sections, shown in figure 5. Each pie section is dedicated to its associated range sensor and is colored in green, yellow, orange, or red, depending on how far an obstacle is. Red indicates obstacle is too close and green indicates no obstacles within a safe radius.

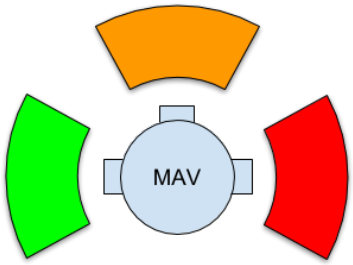

Figure 5: Collision Warning GUI

4.1.4 Design Rationale

The design concept was acquired from the visual display of the road vehicle parking assist software. This allows for the pilot to see where the helicopter is located with respect to obstacles. Additionally, because the field of view of the front facing camera is not wide enough to encapsulate the sides of the MAV, ultrasonic range sensors at the sides of the MAV, to an extent, fulfill missing visual information.

4.2 Height Indicator

4.2.1 Introduction

Height detection provides GUI feedback for the pilot regarding how high the MAV is from the surface of the terrain.

4.2.2 Structure

To inform the pilot of the height, down-facing range sensor is placed at the bottom of the MAV. Range acquired from the down-facing range sensor is transmitted to the server, where it is converted to meters, formatted, and written to a text input element within the GUI.

4.2.3 GUI Element

For the pilot to be aware of the height, a label element, *Height*, along with a height value next to the label is included. The height element is positioned at the upper-left corner of the GUI.

4.2.4 Design Rationale

When flying over obstacles, a pilot must be aware of the range he is from the obstacle below. For the pilot, being aware of height is especially crucial when landing or flying high and solely relying on the graphical user interface for the vision. Visual input coming from the front and package cameras does provide the pilot with the necessary height awareness; however, measured height improves awareness.

4.3 Attitude Indicator

4.3.1 Introduction

Vertical alignment awareness allows for the pilot to know how the MAV is aligned with respect to the horizon. There are two components to the vertical alignment: front-tilt and side tilt. Both of the values are represented by the attitude indicator.

4.3.2 Structure

Data obtained from the accelerometer is converted to pitch and roll angles and transmitted to the server. The server then reflects the pitch and roll in the attitude indicator.

4.3.3 GUI Element

Instead of designing the attitude indicator from scratch, we use an existing jQuery plguin, *Flight Indicators*, by Matton Sébastien. *Flight Indicators* provides a number of flight instruments, including attitude indicator, heading indicator, vertical speed indicator, air speed indicator, and altimeter indicator. The attitude indicator, shown in figure 6, is positioned at the bottom-left corner.

Figure 6: Attitude Indicator

4.3.4 Design Rationale

All manual controlled aviation vehicles have attitude instrument. It is beneficial for the MAV pilot to have access to the vertical alignment data as well.

4.4 Heading Indicator

4.4.1 Introduction

In addition to the attitude indicator, an awareness of the direction the MAV is facing is, too, be suitable for the pilot. We implement a heading indicator, which is a compass.

4.4.2 Structure

Data obtained from the accelerometer is converted to yaw angle and transmitted to the server. The server then reflects the yaw value in the heading indicator.

4.4.3 GUI Element

We implement an existing heading indicator provided by the jQuery plguin, *Flight Indicators*. The heading indicator, displayed in figure 7, is positioned at the bottom-left corner, next to the attitude indicator.

Figure 7: Attitude Indicator

4.4.4 Design Rationale

When navigating through the competition map, it is convenient for pilot to be aware the direction the MAV is facing. Because heading indicator is not as crucial for the pilot as any of the video feeds, the GUI element accumulates a small area on the screen.

4.5 Speed Indicator

4.5.1 Introduction

Awareness of the overall velocity of the MAV is, too, useful for the pilot to know. Although speed indicator is not a crucial component, presenting the flight variable to the pilot allows for the pilot to keep track of the velocity.

4.5.2 Structure

Linear acceleration values obtained from the accelerometer are integrated with Euler integration, and added to the overall velocity vector. The magnitude of the velocity vector is transmitted to the server and reflected in the text label.

4.5.3 GUI Element

The speed label, along with the value, in meters per second, is displayed at the uppler-left corner of the screen.

4.5.4 Design Rationale

Presenting speed in text form is enough for the pilot to be aware of how fast the MAV is moving. Additionally, velocity is not an important indicator and must, therefore, acquire minimal area on the main screen.

4.6 Front Video Feed

4.6.1 Introduction

Front camera video feed is the most essential and a required component of the design.

4.6.2 GUI Element

A background window element with video feed overlaying the entire screen.

4.6.3 Design Rationale

Because video feed is a required component and allows for the pilot to see what the MAV is doing whenever the MAV is not visible to the pilot, the entire video feed should stand out. Overlaying the entire monitor provides a clear visual display for the pilot.

4.7 Bottom Video Feed

4.7.1 Introduction

Down facing camera provides additional visual information to the pilot. The down-facing camera is used to assist the pilot in package pickup and provides awareness of obstacles below.

4.7.2 GUI Element

A smaller rectangular area at the upper-right corner of the screen, displaying the down-facing camera video feed.

4.7.3 Design Rational

Because bottom video feed is not as important as the front video feed, the rectangular area for the bottom video feed must not stand out.

5 Graphical User Interface

The graphical user interface consists of a web-page, with the entire background covered with a front view video feed. Flight instrument component elements and bottom facing camera feed element overlay the sides of the main video feed and may be semi-transparent. When package pickup mode is entered, the bottom facing camera feed window is enlarge sightly and pickup guidance elements are presented. Shown in figure 8 is a mock-up of the GUI. All GUI elements, except for the flight variables, are explanatory. The flight variables section provides the pilot with height, velocity, forward tilt, and a any other non-essential variables.

For the actual deign, shown in figure 9, we develop the front feed and bottom feed elements, the collision warning instrument, and the flight variables element. For flight instruments, we found an existing jQuery plugin, *Flight Indicators*. *Flight Indicators* provides a number of flight instruments, including attitude indicator, heading indicator, vertical speed indicator, air speed indicator, and altimeter indicator. We use attitude indicator and heading indicator. Both of these indicators represent the yaw, pitch, and roll of the MAV, based on the values fed from the accelerometer onboard.

The height flight variable displays the range of the down-facing ultrasonic sensor in meters. Making use of the altimeter indicator for displaying the height would have not been suitable as altimeter measures the altitude from the sea level and not the altitude to the terrain below.

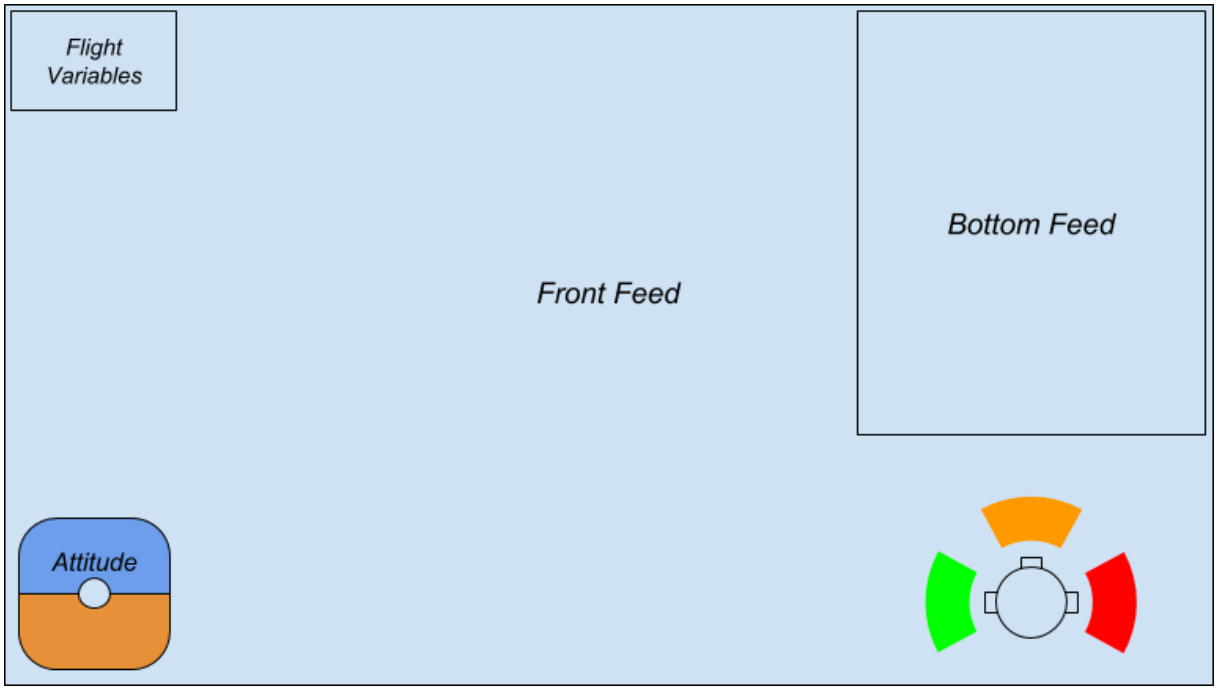

Figure 8: Graphical User Interface Mock-up

The speed flight variable displays the magnitude of the velocity of the helicopter, also obtained from integrating linear acceleration obtained from the accelerometer.

Figure 9: Graphical User Interface

All the values represented in the GUI are obtained at the Raspberry Pi side, transmitted to the server via UDP, and reflected in the GUI. We currently display artificially generated values for the collision warning instrument, attitude indicator, heading indicator, and the flight variables. For consistency and legitimacy of testing, all the generated values are transmitted from the Raspberry Pi. Both, the front feed and bottom feed, display the actual camera feeds transmitted from the Raspberry Pi. We also implement the ultrasonic sensor for displaying the actual height flight variable. The instructions for using the down facing ultrasonic sensor are documented in the *README* of MAV Senior Capstone repository.