Fire Detection and Suppression Drone

A Major Qualifying Project

Submitted to the Faculty of Worcester Polytechnic Institute

in partial fulfillment of the requirements for the Degree in Bachelor of Science

in

Electrical and Computer Engineering

By

Nicholas Janco

Xiaoyi Long

Elizabeth Walling

Date: 3/31/2020

Project Advisor:

Professor Maqsood Mughal

This report represents work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review. For more information about the projects program at WPI, see http://www.wpi.edu/Academics/Projects.
Abstract

The goal of this Major Qualifying Project (MQP) is to create a drone equipped with an automatic flame sensing and water spraying system to overcome the major problems faced during firefighting. These problems faced by firefighting include lack of knowledge about the building construction, fire intensity, and knowing about the building tenability and victim viability. Work on the drone deals with the proof of concept, exhibiting the scalability of this system for large scale fires. During testing of heights 10 feet or below, the drone performed most effectively. The drone that we built was capable of suppressing small fires at a distance of 4 feet from the fire. This is achieved with a custom-designed spraying system controlled by a flame sensor.
Acknowledgments

It has been our honor and privilege to be a part of the Electrical and Computer Engineering (ECE) program at WPI as undergraduate students. We like to express our heartfelt gratitude to all of those who were influential in helping us to complete this MQP.

First, we would like to thank our project advisor: Professor Maqsood Mughal from the ECE department, who had provided us helpful advice and assistance to complete this project through A, B, and C terms, including areas of setting the project scope, providing research resources and prototyping materials, and advising our final deliverables. He first initiated this firefighting application drone project, and immediately caught our interest for its hands-on ability and potential wide range of applications.

Second, we are thankful to have Mitra Anand as another useful resource of ours when it comes to SolidWorks design and 3D printing. Since we had little experience in SolidWorks or 3D printing as ECE majors, Mitra had guided the team in the mechanical design by reviewing and modifying our drone frame design, suggesting printing material, and setting up printing procedures.

Third, we would like to thank Professor William Michalson and Professor Stephen Bitar for their insightful advice on parts of our project. At the early stage of our project when we were collecting information on how to build a drone, Professor Michalson suggested that system thinking was a key to making a drone. As a professor in Robotics Engineering (RBE), who has rich industrial and drone projects advising experiences, he gave us a list of things to consider, such as the targeted purposes of our drone, and nozzle placement’s impact on drone movement. More importantly, he reminded us to minimize the complexity of the mechanical design; instead, we should concentrate on the ECE aspects, things like flying, communication, electrical design for the flame sensing and water spraying system. Speaking of electrical design, Professor Bitar was a big contributor. As a huge enthusiast and expert in analog electronics, he patiently went over and revised our circuit every time we brought our designs over. With his extensive guidance, we finally came up with a circuit for our flame sensing and water spraying system, which makes logical sense, and optimizes the functionalities.

Additionally, we would like to extend our appreciation to Mr. William Appleyard for the ECE shop, since he had helped us a lot with providing parts and keeping an organized record of our parts purchases.

Last but not least, we really appreciate the help from our friend Gabriel Rodriguez. As a senior double-majoring in Mechanical Engineering (ME) and RBE, he has studied drones as a hobby and had genuinely offered so much help when we were stuck in problems. He helped us look at the design of the drone, and suggested possible solutions for drone not taking off, and drifting problems. He even lent us some parts such as the XT60 connector and screws. We were fortunate
enough to have his advice from a ME and RBE perspective, which was what we lack as ECE majors.

We sincerely offer our thanks and gratitude to all the people mentioned above, who had taught and helped us so much throughout the project, without whom, our MQP experience would not be as great.
Executive Summary

The number of fires and wildfires occurring has been slowly increasing in the world today ("Wildfires and Climate Change", 2019). With climate change and global warming, wildfires have become more and more relevant, especially in California and Australia. Stories about wildfires have arisen in the news quite frequently throughout the past couple of years. Fire department used to face the challenges of combating massive fires, and the deficiency of tools that could help them locate and extinguish fire accurately and efficiently. Fortunately, the emergence of firefighting drones has greatly improved the effectiveness and efficiency of fire combating. And the goal of our MQP is to design and implement a firefighting drone that detects fires and suppresses it with a spraying system.

The firefighting drone we created was all built from scratch. For first-time drone builders, we had encountered a lot of challenges throughout the project, but we were able to overcome each of them and came up with a finished product. The main feature that makes this drone unique is the flame sensing and water spraying system. Our drone is capable of sensing fire that is 4 feet away. The water pump carried by the drone can pump out about 240 liters per hour of liquid. Our drone was created as a proof of concept. The process involved a large amount of research, along with several tutorial videos on YouTube. A good amount of time was spent troubleshooting and understanding why certain components did not operate the way they should.

Our firefighting drone consists of two systems - the flying system and fire sensing with the water spraying system. Our design idea is to have the flame sensor continuously detect fires while the drone is manually controlled by a person on the ground. Once a fire is detected, the spraying system would automatically pump out water to put out the fire.

The flying system mainly consists of a flight controller (FC), motors, electronic speed controllers, and a 3S LiPo battery. The first important aspect of building a drone is understanding the weight distribution. Before building the drone, it is critical for us to make logical guesses of how much the drone weighs itself, and the payload that the drone is going to carry, so that we can pick the components accordingly. The components would have the safest and most optimal performances under specified range of weights. The physical arrangement of the parts can also affect the drone performance. We have to design a way to distribute the total weight evenly throughout the aircraft, so that it stays balanced in the air. Another important aspect is the power consumption. The flying system needed to consume a minimal amount of power in order to get the longest flight time. Our projected flight time without any payload is about 4.2 minutes. Lastly, building a drone requires a clean solder on all the parts, since one short circuit would cause flight failure.

The fire sensing and spraying system consists of the Arduino controller, a water pump, a flame sensor, a servo motor, a water container, a hose, and its own power system. We chose an IR sensor as our flame detector due to its easiness of communicating with the Arduino and budget constraint;
however, we did realize its limitations of filtering sunlight, whose wavelength falls within the sensor’s range of spectral bandwidth, which could be the biggest distraction when the sensor is looking for flame. The fire sensing system is consistently on, while the water spraying system goes on automatically with a high flame reading. We have the nozzle and flame sensor both attached to a platform which is moved by the servo, located at the front of the drone. Under the normal situation, the platform will constantly swirl 180 degrees in the X plane to search for fires. If high-intensity light is detected, the platform will stop and swivel ± 5 degrees around the angle where the light reading is the highest. Then the water pump located underneath the drone will pump and spray water through the hose.

After multiple troubleshooting and testing, we concluded that a drone of this capability is possible. Our drone may have been demonstrated on a small scale but can be easily scaled up to fight larger fires. Due to our budget and time constraints, we were not able to create a drone to the size we wanted to. Overall, our project was successful and could pave the way for other projects to improve and size up.
# Table of Contents

Abstract ii
Acknowledgments iii
Executive Summary v
Table of Contents vii
List of Figures x
List of Tables xi
Chapter 1: Background and Introduction 1
  1.1 Background 1
  1.2 Introduction 1
    1.2.1 Motivation 1
    1.2.2 Creative Process 2
    1.2.3 Electrical System 4
    1.2.4 Mechanical System 4
    1.2.5 Software 5
    1.2.6 Application 6
Chapter 2: Parts and Components 7
  2.1 Parts Selection 7
    2.1.1 Flight Controller 8
      2.1.1.1 FC Specifications 9
        2.1.1.1.1 FC firmware 9
        2.1.1.1.2 Processor 9
        2.1.1.1.3 UART 10
        2.1.1.1.4 IMU sensor 10
        2.1.1.1.5 Layout 11
        2.1.1.1.6 Size and Mounting Pattern 11
      2.1.1.2 Candidates and Value Analysis 11
    2.1.2 Transmitter and Receiver 13
    2.1.3 Frame 14
    2.1.4 Electronic Speed Controller (ESC) 15
    2.1.5 Motors 16
    2.1.6 Batteries 17
    2.1.7 Flame Sensing MCU 18
    2.1.8 Flame Sensor 19
    2.1.9 Servo Motor 19
  2.2 Parts Assembly & Configuration 20
    2.2.1 BetaFlight Configurator Installation 20
    2.2.2 Parts Configuration 20
    2.2.3 Spraying Platform Assembly 21
    2.2.4 Configuration Tests 21
2.2.4.1 Testing Single Motor and ESC 22
2.2.4.2 Testing Channel Mapping 23
2.2.4.3 Test All Motors on FC using Transmitter 23

Chapter 3: System Architecture 26
3.1 Flight System 26
  3.1.1 Drone Structure 26
  3.1.2 FC Electrical System 28
    3.1.2.1 System Schematics 29
    3.1.2.2 Power Rating 29
    3.1.2.3 Flight Performance Calculations 30
    3.1.2.4 FC Programming on BetaFlight 31
    3.1.2.5 Troubleshooting 31
  3.1.2 Spraying System 32
    3.2.1 Spraying System Structure 32
    3.2.2 Electrical System 33
    3.2.3 Software 34
    3.2.4 Second version improvements 35
  3.3 Combined System 35

Chapter 4: Tests and Results 36
4.1 Takeoff and Landing 36
4.2 In-air Performance 37

Chapter 5: Discussions and Recommendations 39
  5.1 Flight Safety 39
  5.2 Fire Extinguish 40

Chapter 6: Conclusion 41

Bibliography 42
Appendix A: FC Pros & Cons 44
Appendix B: Installing & Configuring BetaFlight 45
Appendix C: Parts Configuration 50
  C.1 Bind Transmitter and Receiver 50
  C.2 Assign Auxiliary channels 51
  C.3 Set Up Aux Mode in BetaFlight 52
  C.4 Set Failsafe 54
  C.5 Calibrate FC in BetaFlight 55
  C.6 Verify Channel Mapping 57
  C.7 Verify Channel Directions 57
  C.8 Adjust Channel Centers 58
  C.9 Adjust Channel Endpoints 58
  C.10 Configure Receiver Protocol in BetaFlight 59
Appendix D: 1st Model Flight Performance Calculation 61
  D.1 No payload 61
List of Figures

Figure 1 Aerial, and front view of the first and second design of our drone ........................................... 3
Figure 2 Flysky FS-i6X 2.4GHz transmitter with IA6B 2.4GHz receiver .................................................... 13
Figure 3 True X quadcopter frame design from different angles ................................................................. 15
Figure 4 The stator width and height measurements of a brushless motor .................................................. 16
Figure 5 Evolution of motors ......................................................................................................................... 17
Figure 6 Goldbat 11.1 V 2200 mAh 3S 35C LiPo battery ........................................................................... 18
Figure 7 Arduino Nano V3.0 ......................................................................................................................... 19
Figure 8 DFR0073 Analog flame sensor ......................................................................................................... 19
Figure 9 TowerPro SG90 (left) vs. MG92B (right) servo .............................................................................. 19
Figure 10 Major steps of the configuration of BetaFlight prior to the FC programming .............................. 20
Figure 11: Mounted platform on the servo ..................................................................................................... 21
Figure 12 Complete platform on the drone ................................................................................................... 21
Figure 13 ESC wire connection ...................................................................................................................... 22
Figure 14 Flysky-i6AB receiver pin connections .......................................................................................... 22
Figure 15 Connection among battery, receiver, motor, and ESC ................................................................. 22
Figure 16 X-shape drone motors spinning direction ...................................................................................... 23
Figure 17 Switch motor for opposite motor direction .................................................................................. 23
Figure 18 Wiring of ESCs, motors, and FC .................................................................................................... 24
Figure 19 BetaFlight interface to control motors .......................................................................................... 24
Figure 20 Systematic view of the project ....................................................................................................... 26
Figure 21 Drone electrical system connections ............................................................................................ 29
Figure 22 Adjust Angle and Horizon mode parameters on BetaFlight .......................................................... 32
Figure 23 BJT switch for water pump ........................................................................................................... 33
Figure 24 Block diagram of electrical spraying system .............................................................................. 34
Figure 25 State diagram for arduino operation ............................................................................................ 34
Figure 26 Final spraying system installed on drone ....................................................................................... 35
Figure 27 90 Degree Angle of Antennas ....................................................................................................... 40
List of Tables

Table 1 FC weighted scores assignments ................................................................. 7
Table 2 FC Processors Comparison ........................................................................ 9
Table 3 Number of UART’S for different processors .............................................. 10
Table 4 Detailed value analysis for FCs ................................................................. 13
Table 5 Comparison between different quadcopter designs ................................ 14
Table 6 General guideline on picking motors based on frame size ....................... 17
Table 7 Comparisons on Two Models’ 3D-Printed Parts ...................................... 28
Table 8 Total Current and Voltage Consumption of FC Electrical System ............ 29
Table 9 Calculated results for hover flight time and thrust-weight ratio under five weight conditions ........................................................................................................... 30
Chapter 1: Background and Introduction

1.1 Background

Within the past two years, three major wildfires ravaged our world: the 2018 California wildfires, 2019 Amazon rainforest wildfires, and 2019 Australia bushfires. These fires have created unprecedented damage, destroying more than 26 million acres of land and killing more than 30 people and an estimated 1 billion animals (Zee & Torres, 2020). The 2018 Camp Fire in California resulted in estimated damages of $16.5 billion (Nagano, 2019). In the Amazon, around 76,000 fires burned throughout the rainforest. This number showed an increase of more than 80% fires compared to the year before (Borunda, 2019). These warning signs were foreseen in Australia as researchers warned years ago about an increase in bushfires due to a decrease in rainfall and increase in intense and hot temperatures (Yu, 2020). The California Department of Forestry and Fire Protection stated the seven most destructive California fires occurred in the last decade (Nagano, 2019). It is believed that there will be a 15-70% increase in high-to-extreme fire risk days by 2050, and increase by over 100% at 2100 in comparison with the 2010 fire season (Yu, 2020). So much effort and money have been put in to stop the ravaging fire, including the help from thousands of firefighters. Additional to the firefighters present at the fire scene to combat the fire, the emergence of high-tech firefighting has significantly increased the safety for the fire service, improved their effectiveness and saved countless lives (Gasior, 2018). Firefighting drones have been playing a major role in fire safety specifically.

1.2 Introduction

1.2.1 Motivation

The firefighting drone we decided to build had several motivations attached to it. In the current state of the world, with global warming becoming an increasing issue as the years go on, wildfires around the world have become a large issue (Lieberman, 2019). As of recently, Australia has been engulfed in flames due to long droughts and increased temperatures (Law, 2020). Just like for any technology in its development phase, there is always a need for improvements in the fire technology as well. Hoping to help with these advancements, we researched how to build a drone that detects and suppresses fire. We wanted to create a prototype quadcopter drone on a small scale to show proof of concept and help others expand and create this drone on a large scale. The idea is to keep firefighters from having to get in close contact with these massive fires, and suppress the flames enough for firefighters to safely combat the fires. We cannot completely remove firefighters from fire suppression as they play vital roles in decision making and the elimination of these fires. But we can help by making their jobs easier and safer, aiming to prevent serious injuries. With our drone, we will be able to provide great detail on how to correctly build the
suppression drone and help others to build upon our foundation. This project also challenges our electrical engineering skills on being able to create a full electrical system for a product.

We also hope that our drone could be the base model for drones that could be used for other public safety applications, such as agriculture, search and rescue, law enforcement, and disaster responses.

1.2.2 Creative Process

The creation of the drone fire detection and suppression systems took an extended amount of planning, research, failures, and overcoming obstacles. Coming into the project, our group had no experience in drone building. We initially concentrated on the drone system and getting the quadcopter to fly in the air, which would involve putting parts together, and programming the FC. We started with research, and found that how-to-videos and community forums were vital in the progression of familiarizing, designing and putting together the drone as common mistakes and techniques are discussed in detail. Without the details we discovered in various articles and videos, we would have encountered many more obstacles.

In our research, we looked into what parts we would need to build the drone, and other factors, such as, temperature, voltage/current ratings, weight, and costs of components. The important part of this research was to determine parts that would work in our system as a whole. Focusing on the specifications of each part ensured we would have the correct voltage, current, temperature, and durability ratings to fit in our system. We did realize one big challenge during the drone building process is that since none of the resources online was tailored to the exact same parts that we were using, it was hard for us to follow guidelines given different versions written by different people. After having several candidates for each part in mind, we performed the value analysis to finalize the candidates that would best suit our drone. We came together and created system-level designs to confirm that all drone parts would work together. When the system-level designs were complete, we went ahead and ordered parts within our budget. The first thing we did after receiving the items was to make sure they functioned the way they should, and there were not any defects. Having confirmed that the FC and motors worked properly, we then moved to programming the FC to correctly communicate with all parts of the system. A good amount of time was spent figuring out communication between parts as there were constant troubleshooting and soldering issues that prohibited the system to work. When the drone programming was complete, we soldered everything together then attached the system to the frame of the drone. When the drone system was completely built, we did several flight tests. The first few tests resulted in imbalanced motor speeds and crashes. But these issues were greatly improved with program troubleshooting and parts replacement, primarily the motors, propellers, and the overall drone frame, which significantly improved the balance and rigidity of our drone. We hoped to test the drone at much higher altitudes (around 30 feet) once the drone reached a stable performance. But due to time and budget constraints, we were only able to get the drone up to approximately 10 feet from its initial
take-off stage. However, we were able to develop the flame sensing and water spraying system alongside the drone. The spraying system provided about the same amount of challenges as the drone system as we had to continuously troubleshoot the programmable code. With the spraying system, we went through a similar process as with the drone system in terms of research, and parts selection. We had researched online and consulted professors at the WPI ECE department, and finalized a circuit that involved power regulating, and automatic controlling. In order to minimize our expenses, we took advantage of the electronic components from our ECE lab kit used in a previous class to construct the circuit.

We performed tests for each individual system before combining them together. We troubleshooted if anything failed; otherwise, we considered if there could be improvements to the current working design. We had been conducting tests for our drone’s aviation performance once the drone was put together and programmed, and for our flame detecting and spraying system once we finished building the circuit. When both systems were created successfully and combined together, we tested the performance of the drone and the accuracy of the spraying system. Learning our lesson from the first model drone, we developed a second model drone, and proved it to have a higher performance than the previous model. The figures below show the aerial, and front view of our first and second design of our drone (Figure 1). Chapter 5 will include all the details of our tests throughout this MQP.

Figure 1 Aerial, and front view of the first and second design of our drone
1.2.3 Electrical System

Our group members were all ECE majors, and the main focus of building the drone was to demonstrate our skills learned from our major. Throughout the report, the emphasis will be on all the electrical systems and coding from the drone.

The electrical system consists of two main parts: the drone system, and the flame sensing with a water spraying system. The drone system consisted of powering the motors, regulating the motors via electrical speed controllers (ESC), allowing the drone to fly, and control through the flight transmitter. The transmitter could arm the drone, and change between manual or auto flight modes with a switch. This worked with the communication side of the drone with the receiver. Altogether, the drone system could fly, communicate with the transmitter, activate different flight modes, and move in any direction.

The flame sensing with a water spraying system consisted of a water pump, a hose, a IR flame sensor, and a servo motor, allowing the drone to spray out a fire suppressant to a spot with high IR readings. The circuitry of this system was placed inside of the drone power compartment, and worked as its own system with its own power source. One main issue we encountered while designing the electrical system for the flame sensing and water spraying system was providing enough voltage to the water pump. In our spraying system, the Arduino was powered with two batteries and the Arduino provided voltage to the other components through its programmable digital ports. The port voltage and current from the Arduino were not enough to turn on the water pump. In order to get the right amount of voltage and current to the water pump, we designed a circuit that increased the input voltage and current to the level to turn on the water pump. Detailed logistics of the electrical system will be discussed in Chapter 3.

1.2.4 Mechanical System

The mechanical system consisted of the structure and frame of the drone, and the physical placement of the parts for the water spraying system. Due to our minimal knowledge, we started with a design already made for a quadcopter and made minor changes. The frame was then printed with 3D printers. Once we had the complete design, we printed our drone frame parts from the WPI Foisie Makerspace with the staff's help and online training for 3D printing on Canvas. With the successfully-printed drone parts, we needed to combine and screw together the parts. A challenge that came with this was that some screw holes were not aligned, and to fix this, we adjusted the design for our second model and hand drilled holes after printing the parts and used different sized screws to properly install motors. This process took several reprints, design adjustments, and testing multiple screw sizes to see which fit best. Another challenge we faced was to have a total weight that would be less than the thrust power of the motors. We took this into consideration when choosing the structure of the drone. Along with this challenge, we also needed to secure all parts safely, even in the event of a crash. So for our second model, we chose much
powerful motors and propellers, and reprinted frame parts that were more rigid. One specific adjustment to the frame was to increase the modify the design so that some parts were thicker and to also increase the infill percentage. Additionally, one of the joints was particularly strained so heated inserts were added to strengthen the screw joint. In terms of the physical placement of the parts for the water spraying system, some challenges arose with our minimal knowledge of mechanical engineering, but we were able to overcome them with the help from outside sources and the supportive staff at WPI. The first challenge that we encountered while building this system was to seal the fire suppressant liquid in a bottle while preventing leaks. We went about this with melting plastic and sealant. Since this model was a prototype, the container we used was a 300mL simple plastic drinking bottle. The hose attachment was pulled through the opening of the bottle and attached to the water pump inside the bottle. An extra hole was created to pull the wire from the water pump out to the Arduino. The spraying system was then attached to the bottom of the drone with supportive straps. The second challenge we ran into was to figure out a way to mount the sensor, hose and servo together, while ensuring the hose could be able to move as minimum tension as possible. Since our idea was to have the flame sensor, hose, and servo constantly swirl altogether for 180 degrees to search for fires, we used a thin piece of wood as a platform, driven by the servo, to attach the hose and sensor. And this synchronized the movement of those three parts. Additionally, due to the position of the bottle, the hose coming out from the water bottle had to wrap around one of the legs of the drone to make it to the front of the drone. The platform and the bottom of the bottle not being on the same level hindered the hose’s smooth motion because of the stiffness of the hose. We came up with a mount that brought the platform to the same level as the bottom of the bottle, which greatly alleviated the friction between the hose and the bottle. Chapter 3 will have much more details about the overall mechanical design.

1.2.5 Software

The code for flying the drone was created with the BetaFlight software. BetaFlight is a programmable software meant for use in the creation of aviation products. The software has a user-friendly interface with the programming done through a graphical user interface (GUI). Some of the issues we found with the flight settings were having the correct speeds for the motors, moving the motors in the right directions, and fail-safe functions. These issues came and were taken care of through learning how to operate BetaFlight. Most of our initial flight issues were due to wrongly adjusted flight settings.

The code we created for the spraying system was made in the Arduino integrated development environment (IDE). The program worked by a servo motor moving the flame sensor in a pivot of 180 degrees in the X plane searching for a high reading. Once the high reading of 850 analog output from the flame sensor was found, the system would trigger the water pump to spray with a ± 5-degree rotation in the direction of the high reading. The program automatically turned off once
the flame sensor read a value below the analog threshold value of 750. When we felt comfortable with the spraying system code we started to build the spraying system.

1.2.6 Application

The application of this quadcopter drone is to be used in any type of fires, ranging from house fires to wildfires. Based on the drone size it could be used in different sized fires. We wanted to build a drone that would have several features essential to firefighting. With the surge of drone popularity and the various ways they can be used, we felt that this would be a product essential in future technologies (Fotouhi, 2017). Another factor that took into consideration was the surge in wildfires occurring such as the California fires and the Australian bush fires (Yu, 2020). We want this drone project to someday help prevent houses, human lives, and communities from being lost.
Chapter 2: Parts and Components

We will discuss our choice of components for both the flight system and flame sensing along with the water spraying system in this chapter. The flight system consists primarily of the FC, transmitter and receiver, frame, ESC, motors, and battery. And the flame sensing and water spraying system includes a microcontroller (MCU), IR sensor, and servo motor.

2.1 Parts Selection

Throughout the process of selecting parts, we took careful consideration of what we needed on our drone and researched precisely to get the correct parts by doing value analysis. Value analysis generated a total score for each candidate, which indicated the overall performance of that candidate. And this allowed us to determine the combination of elements that would fit the best in our drone system and was the most effective implementation. We assigned the weighted scores for each major component, including transmitter, receiver, FC, flight simulator, batteries, and frame, based on the importance of the features, shown in the table below. The higher the score is, the more important the feature is in a component.

Table 1 FC weighted scores assignments

<table>
<thead>
<tr>
<th>Item</th>
<th>Transmitter</th>
<th>Receiver</th>
<th>Flight controller</th>
<th>Flight simulator</th>
<th>Batteries</th>
<th>Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>X 5</td>
<td>X 10</td>
<td>X 10</td>
<td>X 50</td>
<td>X 40</td>
<td>X 30</td>
</tr>
<tr>
<td>Range</td>
<td>(power of signal) 25</td>
<td>(sensitivity )35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power consumption</td>
<td>X 10</td>
<td>X 10</td>
<td>X 50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>X 15</td>
<td>X 5</td>
<td>X 10</td>
<td>X 10</td>
<td>X 10</td>
<td>X 15</td>
</tr>
<tr>
<td>Usability</td>
<td>X 10</td>
<td>X 30</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shape</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X 30</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>X 30</td>
<td>X 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of channels</td>
<td>X 20</td>
<td>X 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X 25</td>
</tr>
<tr>
<td>Susceptibility to noise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Data storage size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
</tr>
</tbody>
</table>
As the brain of the drone, the FC sends commands to other parts on the drone. For example, the FC controls the motors and regulates them to stabilize the drone. To fit our needs and budget, we picked the Matek F405 CTR FC. This controller had all the capabilities we wanted and was programmed to be user-friendly with BetaFlight. The controller had several 5 V ports to power other parts of the drone and included a barometer to get altitude readings during flight.

The four brushless motors we used for our first model were from Crazepony rated at 1700KV with a size of 2306, paired with their 6-inch three-blades propellers. After the failures we experienced during the testing with our first model, we decided to use DYS 1120KV motors, with a size of 2836, and 9-inch two-blade propellers to provide enough thrust and power for our second model. Overall, the components that we decided on were selected with careful consideration of the advantages and disadvantages of each type in order to create an outcome that will perform well, while still having all the necessary features. We looked in detail on what parts fit best in our system and had the necessary specifications to work as a whole. While selecting parts, we looked into datasheets, compared different types and brands of each part, and performed a value analysis on which type of part would bring us the most efficiency with the least cost. And we will discuss in detail our parts selection in the following sections.

2.1.1 Flight Controller

The FC is the brain of the drone, which basically is a circuit board with sensors that detects orientation changes of the drone. It also receives user commands and controls the motors in order to keep the drone in the air.

To figure out what was the best FC for our drone, we must first figure out what exactly we wanted to accomplish. As beginners trying to build the first drone, we were aiming to build one that had the most basic setup. This setup included a fairly heavy payload and a flame detecting and water spraying system for firefighting purposes. Our drone didn't need to operate at racing speed, because in firefighting, we would rather have the drone stable and balanced than fast. Ideally, we would like to equip our drone with a camera so that it could send the operating team on the ground real time videography. Another factor we needed to consider was the compatibility with the flight simulation software. A FC that was difficult to set up was less valuable than one that was easy to configure. Additionally, we needed to think about the budget of our project, since we only had $600 for the entire project. Last but not in any way least, we must consider the capabilities that the FC supported, given our needs for this firefighting drone.

We had used online resources and reached out to people who could give us professionals opinions on building drones, such as our project advisor Professor Mughal who has built drones himself, and Professor Michalson who has advised two firefighting-drone-related MQPs. Given our project outline and budget, we outlined the higher-level requirements for the FC as follows:
● Compatible with BetaFlight configurator
● Must contain some sensors for drone movement measurement and regulation
● Fairly lightweight and small-sized
● Support Flysky i6-AB receiver IBUS protocol

More details, including typical FC specifications, and value analysis among several candidates will be discussed in the following sections.

2.1.1.1 FC Specifications

When choosing a FC, we needed to compare the specifications of different FC to see which one met our needs.

2.1.1.1.1 FC firmware

First thing we looked at was the FC’s compatibility with the flight configurator. As we had decided BetaFlight as our flight configurator, we must select a FC that could be programmed on BetaFlight in order to get it to fly the way we wanted.

2.1.1.1.2 Processor

The processor is the brain of a FC, similar to the CPU in a computer. F1, F3, F4, and F7 are the different processors that are commonly used in racing drone FC. F1 has the slowest processor speed and least flash memory, while F7 is the fastest and holds much more flash memory.

<table>
<thead>
<tr>
<th>Processor (example chip)</th>
<th>Processor Speed</th>
<th>No. of UART on FC</th>
<th>Flash Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 (STM32F103CBT6)</td>
<td>72MHz</td>
<td>2</td>
<td>128KB</td>
</tr>
<tr>
<td>F3 (STM32F303CCT6)</td>
<td>72MHz</td>
<td>3</td>
<td>256KB</td>
</tr>
<tr>
<td>F4 (STM32F405RGT6)</td>
<td>168MHz</td>
<td>3</td>
<td>1MB</td>
</tr>
<tr>
<td>F7 (STM32F745VG)</td>
<td>216MHz</td>
<td>8</td>
<td>1MB</td>
</tr>
</tbody>
</table>

Based on our online research, it was recommended to get an F3 or F4 FC for now, as we had reached the limit of F1, and F7 FC was still new and needed time to be improved research (Liang, 2020).
2.1.1.1.3 UART

UART stands for Universal Asynchronous Receiver/Transmitter, the hardware serial interface that allows us to connect external devices to the FC, such as, serial radio receivers, telemetry, race transponder, and VTX control.

The number of UARTs on an FC depends on the design of the board, and the processor used. The table below shows the number of UARTs that different-processor FCs have.

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F3</th>
<th>F4</th>
<th>F7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 UARTs</td>
<td>3-5 UARTs</td>
<td>3-6 UARTs</td>
<td>7+ UARTs</td>
</tr>
</tbody>
</table>

Our drone was designed to have the features of radio receiving and video control; however, the more UART’s the FC has, the more flexible it is. After discussion within the team, we figured it’d be safer and sufficient to have 3-6 UARTs.

2.1.1.1.4 IMU sensor

The job of an inertial measurement unit (IMU) sensor is to measure the drone’s movement and orientation. An IMU sensor contains both an accelerometer (ACC) and gyroscope (Gyro). Acro mode (manual) uses only the gyro, while the angle mode (self-level) uses both the ACC and gyro. Given the firefighting application of our drone, it would be helpful to have a self-leveling mode to adjust drone’s movement and orientation, so we preferred our FC to come with both accelerometer and gyroscope.

There were two main properties of IMU we needed to consider in a FC: max sampling rate, and how susceptible to the noise they were (both electrical and mechanical noise). Currently, the most widely used IMU is the MPU6000 as it supports up to 8KHz sampling rate, and is proven to be one of the most robust IMU against noise. The general consensus is to avoid MPUs including MPU6500 and MPU9250 which are noisier despite the higher sampling speed (Liang, 2018). The communication protocol between the IMU sensor and the processor, also known as bus, can also have effects on the effective sampling rate and therefore the maximum FC loop time. The preferred bus is SPI, which allows us to run gyro at a much higher refresh rate than I2C which has a limit of 4KHz.
2.1.1.5 Layout

Layout is where the pins/solder pads are located on the board, and how easy it is to connect the components. Ideally, we preferred all the pads were all located on the edges, which would avoid messy wiring.

2.1.1.6 Size and Mounting Pattern

The common sizes for FCs are 30.5×30.5 mm, 20x20 mm and 16x16 mm. Five inches and larger aircraft normally use 30.5×30.5 mm while anything smaller uses 20x20 mm. We tried to minimize the weight of the FC as much as possible.

Having all the typical FC specifications above, and based on our needs, we came up with a list of features we would look for on a FC:

- Priced between $40-$50
- Fairly fast speed to process data. F4(168MHz) or F7 would be a good choice
- Equipped with ICM Gyro’s – allows 16KHz or 32KHz Looptime
- Preferred BUS is SPI, which allows you to run Gyro at a much higher refresh rate
- Supports, 4S, 5S and 6S
- Soft mounted in some way

2.1.1.2 Candidates and Value Analysis

With our requirements above, we did thorough research on FC on the market and picked out three that suited our needs the best. They were CL Racing F7, Holybro Kakute F7 AIO, and Matek F405 CTR.

Detailed analysis of their pros and cons can be found in Appendix A. However, in order to pick the best one out of those three candidates, it is necessary for us to perform a value analysis against the standards above. After considering the basic specifications of the FC and the needs of our project, we used the weighted points below, with a higher number meaning more important to the analysis:

- Weight (10)
- Power consumption and regulation (10)
- Cost (10)
- Usability (30)
  - Compatible with BetaFlight, and the rest of the system
  - Voltage/current regulator
  - Clean layout
  - Adequate sensors
- Susceptibility to noise (20)
• Data storage size (25)

Then, based on the criteria above, we assigned different value points ranging from 1-3 to each level. And the Table 4 below shows the detailed value analysis for FC.

• Weight: the weight and size of the controller
  ○ Smallest size and lightest 3
  ○ Reasonably small and light 2
  ○ Quite large and heavy 1

• Power consumption and regulation: the power that the controller needs to consume and can provide to other peripherals
  ○ Most effective 3
  ○ Somewhat effective 2
  ○ Least effective 1

• Cost: How much is the component
  ○ Affordable 3
  ○ Somewhat affordable 2
  ○ Least affordable 1

• Usability: how compatible and supportive it is to the rest of the system
  ○ Most compatible and supportive 3
  ○ Somewhat compatible and supportive 2
  ○ Least compatible and supportive 1

• Susceptibility to noise: how susceptible to noise its sensors are
  ○ Least susceptible 3
  ○ Somewhat susceptible 2
  ○ Most susceptible 1

• Data storage size: how much data the controller can store
  ○ Largest storage 3
  ○ Somewhat large storage 2
  ○ Least storage 1
Table 4 Detailed value analysis for FCs

<table>
<thead>
<tr>
<th>Item</th>
<th>Weighted Score</th>
<th>CL Racing F7</th>
<th>Holybro Kakute F7 AIO</th>
<th>MATEK F405 CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Power consumption</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Cost</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Usability</td>
<td>30</td>
<td>2</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Susceptibility to noise</td>
<td>20</td>
<td>2</td>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>Data storage size</td>
<td>25</td>
<td>2(1G flash)</td>
<td>3(SD)</td>
<td>3(SD)</td>
</tr>
<tr>
<td>Total</td>
<td><strong>100</strong></td>
<td><strong>210</strong></td>
<td><strong>220</strong></td>
<td><strong>290</strong></td>
</tr>
</tbody>
</table>

Finally, based on the scores above, Matek F405 CTR surpasses the other two candidates. This controller had all the capabilities we wanted.

2.1.2 Transmitter and Receiver

The flight transmitter we decided on was the Flysky FS-i6X 2.4GHz transmitter with IA6B 2.4GHz receiver (Figure 2). We picked this transmitter due to its large capabilities and relatively low price. The transmitter has six default channels but up to ten channels to use for functions and control parts of the drone. The modulation that the transmitter uses is Gaussian Frequency Shift Keying (GFSK). There are also many failsafe features on the transmitter to prevent serious damage to the drone. The transmitter came along with the IA6B receiver which takes in signals from the transmitter. Having the transmitter and receiver as a bundle was very helpful as we did not need to research what receiver would pair with our transmitter. These two components also were easy to identify and program in BetaFlight. The transmitter runs on 4 double-A batteries and the receiver runs on 5 V DC power. The stick resolution of the throttle and thrust knobs is at 4096 bits. The transmitter has a 26mm dual antenna capable of reaching long distances. When the battery level reaches below 4.2 V, a warning would appear to replace the batteries. The RF power is less than 20dBm. The transmitter has two two-position switches, two three-position switches, four buttons, and two knobs that can all be programmed for certain functions. With all these features and the low price of the transmitter and receiver, we decided this would be a good fit for our project.

![Figure 2 Flysky FS-i6X 2.4GHz transmitter with IA6B 2.4GHz receiver](Source: amzn.to/flysky-trans-recev)
2.1.3 Frame

The first decision to be made about the type of frame is how many motors it will use. The three types of drones that we considered were a tricopter, a quadcopter, and a hexacopter. If a drone has fewer motors, it is cheaper to make, but it is also less stable and has less thrust. Quadcopters can have a decent amount of thrust if the propellers and motors are selected wisely and they are generally stable. Thus, because a quadcopter design would be sufficient while being cheaper to make than a hexacopter, it was the type of design that we selected.

Among quadcopters, there are a variety of different frame shapes that allow for various motor placements. Table 5 shows several of the frame shapes we considered along with notes regarding how the shape affects performance. Based on this information we chose a True X frame design.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>True X</td>
<td>Design places all motors perpendicular so that they are evenly spaced about the center. This allows for equal stability along all axes.</td>
</tr>
<tr>
<td>The Dead Cat</td>
<td>Shape is typically favored by larger designs. The goal of this design was to remove propellers from the camera angle.</td>
</tr>
<tr>
<td>The aptly named H shape</td>
<td>Popularity due to being bulky.</td>
</tr>
</tbody>
</table>

Before picking the final exact frame design we wanted to use, there were a few additional factors we took into consideration. Firstly we needed to ensure that there would be space in the frame to mount all the equipment, such as a battery, and our spraying system. Next is the construction method. We were planning to 3D print the frame, and we had to decide if we wanted to print a unibody shape or one with separate arms. A unibody shape would have to fit inside of one 3D printer but would be easy to use and would require much less assembly. A design with separate arms would have less of a size constraint and would be less costly to repair as only the broken piece would need a replacement; however, this design is more complicated to create and assemble.
The team determined that the size flexibility and ease of replacement far outweighed the more complicated assembly and chose to use a separate arm design. Additionally, smaller things like the orientation of the arms: vertical arms create less drag without affecting flight stability or structural integrity were considered while evaluating frames.

The team did not have much of a background in ME or CAD and did not feel confident designing a frame from scratch so we used the selected specifications and researched drone frame CAD files that were offered to the public until we found one that fit everything we were hoping for. Figure 3 depicts the model of the design from different angles we chose. However, this model did not upload the corresponding files. We used similar designs, one which included the files for the body of the drone, and one which included the files for the legs.

![Figure 3 True X quadcopter frame design from different angles](image)

2.1.4 Electronic Speed Controller (ESC)

The ESC must meet specifications based on the power supply and motors it will power. The FC, which is the important media to distribute power to all the ESCs, and the motors were selected before the ESCs. This meant that all the specifications were already determined and an ESC meeting those specs needed to be found and purchased. The voltage of the battery should not be higher than the ESCs rating and the current of the battery should be larger than that of the ESC (GetFPV, 2018). Additionally, the maximum output voltage of the ESC should not exceed the motor rating and the ESC current should be larger than the motor. Thus, using these specs, we selected the 40A ESCs to work with our FC and motors, and the least expensive ESC with these qualified ratings and reasonable shipping time was purchased.
2.1.5 Motors

The motor is positioned immediately under the quadcopter propeller and controls its rotation and speed. There are two types of motors used in RC, brushless and brushed motors. Generally, brushless motors are used on larger models, such as racing drones, and any bigger models. We would need 4 brushless motors for our drone. There are several things to consider when picking motors. The first rule of thumb is that the thrust to weight ratio should be 2:1, which allows the motors to produce 50% more thrust than the multirotor, so the drone can hover in midair at about half throttle. With a higher thrust to weight ratio, a drone would have greater agility and acceleration, but it might become harder to control as well (Liang, 2019). We had a drone with all the flying system components that weighed around 963 grams. If we shoot for a 2:1 thrust to weight ratio, the drone motor should be efficient enough to produce a thrust of about 481.5 grams each or a total of 1.926 kg on the whole. Once we were clear about our stand on the thrust to weight ratio, it is now important to look into the pole count of the motors. Motors with a higher pole count would require more voltage during its flight but produce a greater motor torque. But with these conditions comes the lower revolutions per minute (RPM) of the motors. On the other hand, motors with a lower pole count would have a higher RPM. But these devices would then require smaller blades, coming with a smaller torque or lift off the ground. In most cases, drones are seen using motors with higher pole counts, i.e lower RPM, in order to eliminate the necessity of a gearbox. And that fit our need of lifting the flame detecting and water spraying system with the drone. Next up is the size of the motor. The size of the brushless motors is normally indicated by a 4-digit number – AABB. “AA” is the stator width (or stator diameter) while “BB” is the stator height, both are measured in millimeters (Liang, 2019). Figure 4 on the right shows the stator width and height measurements of a brushless motor.

Increasing either the width or height of the motor stator would increase the stator volume, size of the permanent magnet and electromagnetic stator coils. This would eventually increase the overall torque of the motor so it could spin a bigger and heavier propeller (Liang, 2019).

The next big question at hand is the KV value of the motor or the RPM/volts. KV refers to a constant that gives us the RPM of the motor when a potential difference of 1 volt is applied across the motor. The RPM value is equal to the product of KV value and power source in volts. Generally speaking, heavier quadcopter usually pair with medium to low KV motors, lighter quad usually use high KV motors (Liang, 2019).

Figure 4 The stator width and height measurements of a brushless motor (source: http://bit.ly/bmotors)
Overall, to decide the motors that fit our quadcopter, we had to first know the frame size to estimate the motor size and the RPM that each motor has to determine if they could generate thrust efficiently. We also had to make sure that the motors could produce enough torque to spin the propeller. Generally, a bigger stator size and higher KV mean more current draw.

The table below gave us a general guideline on picking motors based on the frame size measured by the diagonal motor to motor distance. Because of miscalculation, we picked much smaller motors and propellers for our first model, while we were supposed to go with motors with a size of 26XX or larger, and a KV value of 1200KV or lower, given our 19-inch-arm frame. We did get the correct size motors and propellers for our second model, and they were proved to be the correct configuration from much more successful flight tests of the second model. Figure 5 shows two selections of motor and propellers for our two models, and the detailed comparison between these two selections will be discussed in Chapter 3.

Table 6 General guideline on picking motors based on frame size (source: http://bit.ly/bmotors)

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Prop Size</th>
<th>Motor Size</th>
<th>KV</th>
</tr>
</thead>
<tbody>
<tr>
<td>150 mm or smaller</td>
<td>3” or smaller</td>
<td>1105-1306 or smaller</td>
<td>3000KV and higher</td>
</tr>
<tr>
<td>180 mm</td>
<td>4”</td>
<td>1806, 2204</td>
<td>2600KV-3000KV</td>
</tr>
<tr>
<td>210 mm</td>
<td>5”</td>
<td>2205-2208, 2305-2306</td>
<td>2300KV-2600KV</td>
</tr>
<tr>
<td>250 mm</td>
<td>6”</td>
<td>2206-2208, 2306</td>
<td>2000KV-2300KV</td>
</tr>
<tr>
<td>350 mm</td>
<td>7”</td>
<td>2506-2508</td>
<td>1200KV-1600KV</td>
</tr>
<tr>
<td>450 mm</td>
<td>8”, 9”, 10” or larger</td>
<td>26XX and larger</td>
<td>1200KV and lower</td>
</tr>
</tbody>
</table>

2.1.6 Batteries

There are a number of factors to consider when it comes to choosing a Lithium Polymer (LiPo) battery, such as, voltage, capacity, weight, physical size, discharge, and compatibility with the FC, motors, and ESCs.

First, all LiPo cells have a nominal voltage of 3.7 V per cell. The number of cells is usually labeled as S. For example, 3S means that it is 3 cells x 3.7 V /cell, which is 11.1 V. Second, capacity is
the amount of energy the battery can hold, measured in milliamp/hour (mAh) or amp/hour (Ah). For example, 2200mAh is 2200 milliamp hours or 2.2 amp-hours. That is, this battery will give 2.2 amps of current for one hour. Or 4.4 amps for 1/2 hour, or 8.8 amps for 15 minutes (Genstattu, 2019). Third, the discharge rate determines the amount of power the battery can supply the power system. For example, a 20C battery can discharge at 20 x 2,000 mAh which is 40,000 mAh or 40 Amps. We had a fairly large system, with a big frame, and payload. It would be better if we could find a battery that discharges at a moderate speed. Fifth, since a battery has a huge effect on the drone’s power to weight ratio, we needed to pick out a fairly light battery that could fulfill our needs. Sixth, considering that we had limited space for the battery on the drone, we needed to select a battery that had a reasonable size. Finally, after looking at the requirements of the FC, motors and ESCs, we determined that our battery needed to be around 11.1-22.2 V (3-6S), and have a continuous output current bigger than the total of four ESC’s, which is 160A.

Knowing all the basic specs of the battery, we should also think about how we should power our FC. Fortunately, our Matek F405 CTR FC had a built-in battery eliminating circuit (BEC), a 5 V voltage regulator that would convert our battery voltage down to a clean and constant 5 V; otherwise, it would not be safe to power the FC directly from the battery. We had browsed through a lot of 3-6S batteries on Amazon, and finally selected the Goldbat 11.1V 2200mAh 3S 35C LiPo battery (Figure 6) for the these reasons: Proper supply voltage, reasonable weight and size, moderate discharge rate, good capacity, fair price, and good user reviews.

2.1.7 Flame Sensing MCU

Due to the simplicity of our flame sensing and water spraying system, we did not need a complex MCU. We looked for something that was light weight, power efficient, and had at least five programmable digital I/O pins and two analog pins. Fortunately, our advisor professor Mughal had extra Arduino Nano from his lab class that we could borrow, and it turned out to be a versatile and effective MCU for our system. This MCU takes an input voltage between 7 to 12 V, and can regulate it to a DC voltage of 5 V. It has fourteen digital I/O pins and eight analog input pins, which are more than enough for our application. It has a size of 0.73inch x 1.70 inch, and weighs 7 grams, meeting our requirement of light weight and small size. Notably, the current and voltage ratings for each programmable digital I/O are 40 mA and 5 V. The 5 V would be enough to support the parts in the flame sensing and water spraying system, but the 40 mA would require some sort of current amplifier to be able to power our water pump.
2.1.8 Flame Sensor

The IR array flame detector we chose were Arduino flame sensors manufactured by DFRobot. It is also known as visual flame detectors, employing flame recognition technology to confirm fire by analyzing near IR radiation using a charge-coupled device (CCD). It is especially able to monitor flame phenomena, without too much hindrance from water and water vapor. In the fire-fighting drone game, this flame sensor would play an important role in the probe, which could be used as the drone's eyes to find a fire source.

Within the operating temperature -25 degrees Celsius to 85 degrees Celsius, it can detect light whose spectral bandwidth ranging from 760nm to 1100nm. The sensor has a probe angle of 60 degrees, and takes a supply voltage between 3.3 V to 5 V. The flame sensor would be connected to both analog and digital pins of our Arduino Nano. The digital connection would be used for providing a constant 5 V to the sensor while the analog connection would be used for IR readings of the fire for the user.

We also considered other options as the sensor to detect fire, including smoke sensor and heat sensor. But we decided on this flame sensor since it could often respond faster and more accurately than a smoke or heat detector due to the mechanisms it uses to detect the flame. However, we did realize that this IR sensor could be interfered by sunlight whose bandwidth also falls between 760nm to 1100nm. We kept in mind that when we did testing, we ought to eliminate sunlight as much as possible.

2.1.9 Servo Motor

The purpose of the servo motor was to move the flame sensor and hose in the same direction. For our first model, we used TowerPro SG90 servo. It took a supply voltage of 4.8 to 6 V, and had a torque of 1.8 kg/cm, meaning one centimeter away from the pivot, it could lift about 1.8 kilogram of weight. It experienced some difficulties of spinning that much weight, so we got a stronger servo from professor Mughal. The new servo had a torque of 3.1kg/cm when supplied with 5 V, almost a double of the old servo.
2.2 Parts Assembly & Configuration

In this section, we will briefly talk about steps we took to program the FC, including the configurator and driver installation, and parts configurations.

2.2.1 BetaFlight Configurator Installation

BetaFlight is the configurator we used to program the FC. Besides the installation of Betaflight, we also needed to install and configure the related drivers to be able to flash the FC before programming it. The figure below shows the major steps of this process, and more detailed steps of installation on Windows OS would be discussed in Appendix B.

![Configuration Flowchart]

Figure 10 Major steps of the configuration of BetaFlight prior to the FC programming

2.2.2 Parts Configuration

After installing the configurator and related drivers, we started with configuring the transmitter, receiver, and FC with the proper system and function setup, so that their optimal functions could contribute to the successful flight each time. Here are the major setups that we changed, which are explained in detail in Appendix C.

1. Bind transmitter and receiver
2. Assign auxiliary channels 5 and 6 to switches A and C
3. Set up auxiliary mode in BetaFlight
4. Set failsafe
5. Calibrate FC in BetaFlight
6. Verify channel mapping
7. Verify channel directions
8. Adjust channel centers
9. Adjust channel endpoints
10. Configure receiver protocol in BetaFlight

Some of the lower-level and system tests we performed during the configuration are discussed in the following section.
2.2.3 Spraying Platform Assembly

While we were assembling and configuring parts for the drone flight system, we had multiple discussions within the team about how to implement our idea of having the flame sensor and hose moving at the same time. One teammate proposed an idea of creating a platform that can be spun by the servo, hold both the flame sensor and hose at one place. Thus, the spinning servo would synchronize those two components’ direction of pointing. We first built a prototype of such platform with a short piece of hose (around 5 cm), a thin 4 cm x 4 cm piece of wood, servo, and the flame sensor. We mounted the servo arm on one side of the wood (Figure 11), and on the other side, secured our hose to the wood with metal screws. On top of the hose, we glued the flame sensor. The complete platform is shown in Figure 12.

![Figure 11: Mounted platform on the servo](image1)

![Figure 12: Complete platform on the drone](image2)

2.2.4 Configuration Tests

We had done some testing throughout the process of configuring the parts. When we finished the binding between transmitter and receiver, we tested every one of the motors separately connecting with the receiver and ESC. This step was critical because we wanted to find out the broken parts first out of the merchant so that we could find replacements.

Additionally, after we set up all the configuration on BetaFlight, we tested if the parts work well together. This inspection included the tests for the following items.

1. Channel mapping
2. Test all motors on FC using the transmitter

We will also discuss some issues we ran into during testing in this section.
2.2.4.1 Testing Single Motor and ESC

In order to test whether the ESCs and motors worked properly together, first, we soldered the 3 motor wires to the motor soldering pads on one side of the ESC-motor soldering tabs. We connected the first wire from the motor to the first connection on the ESC, middle wire from the motor to the middle tab on ESC, and last wire from the motor to the bottom tab on ESC.

Then on the other side of the ESC, we connected the red and black wires to the power source in the lab. Each 40A ESC would need a 2S input, equivalent to 7.2 V. In between the ESC power wires, there was one wire tangled by a black and a white wire shown in Figure 8, which should be connected to power ground and signal on the receiver, respectively.

Based on the pin connection of the receiver shown in Fig 14, we connected the Ground port to power source ground, V+ port to power source positive, and the S port to the ESC signal wire. The overall connection is shown in Fig. 15.

After we got all the connections set up, we turned on the power source and pushed the left stick on the transmitter slightly up. The motor indeed started spinning as intended. As we pushed the stick more upwards, the speed of the motor increased. This connection yielded the motor spinning counter-clockwise.

Knowing that we needed to have an equal number of clockwise (CW) spinning motors and counter-clockwise spinning (CCW) motors for our X-shape drone (Fig.16), we simply switched the two outermost wires on two of the four motors as shown in Fig.17.

We then tested the rest of the ESCs and motors using the same method above, and they all worked properly.
2.2.4.2 Testing Channel Mapping

Testing the channel mapping is critical, because incorrect channel mapping would cause failures when taking off, and the problem of drifting. We needed to make sure all the channels were mapped properly to lay the solid base for drone flying.

After we connected the FC to BetaFlight and provided an 11.1 V DC voltage to it, under the ‘Receiver’ tab in BetaFlight, as we moved the sticks and switches on our transmitter, the bars did move as intended. The Aux 1 represents arming. When the left-most stick is switched up (value 2000), the drone would be armed; when the stick is switched down (value 1000), the drone would be disarmed. Moving the left stick up moves the Throttle up (higher number); moving the left stick left moves the Yaw down (lower number); moving the right stick up moves the Pitch up (higher number); moving the right stick left moves the Roll down (lower number).

2.2.4.3 Test All Motors on FC using Transmitter

Now that we knew the FC was calibrated properly, and four motors working separately, we moved on to getting all motors work, connected to the FC. The goal was to have all 4 motors spinning at the desirable directions - CW on one diagonal, CCW on the other diagonal - at the same time controlled by the transmitter. This test would address these questions:

- If the wiring among ESCs, motors, FC, and receiver is correct
- If the transmitter and receiver bind correctly
- If the arming/disarming switch is assigned properly
- If the FC channel mapping is correct

The Matek F405 CTR controller came with detailed documentation for the wiring with different flight parts. We followed their wiring guidelines to first connect the ESCs, motors, and receiver to the FC, as shown below in Figure 18.
After we connected all the ESCs to the FC, we tested on BetaFlight whether four motors would spin correctly. The first step was to test using the sliders on BetaFlight. We kept the default order of the motors on Betaflight - top left motor as #4, top right motor as #2, bottom left motor as #3, bottom right motor as #1, connected the FC to the laptop via USB cable and provided an 11.1V from the power source. The user interface of BetaFlight is shown below.

Before we turned on the power source, we made sure all the propellers were removed from the motors for safety reasons. The motors did successfully spin up in the correct directions. And we could also change the spinning speed of each individual motor on BetaFlight by moving the slider on BetaFlight.

Next, we tested the motors with the transmitter. Using the same setup, we overwrote the BetaFlight slider controller by turning on the transmitter. Having the Switch A pushed away, we armed the
quadcopter, which is equivalent to sliding the bar at the lower right corner shown on the screen above. By moving sticks, we were able to change the Pitch, Yaw, Roll, and Throttle of the motors, which are reflected on the slider bars on the left of the screen above.
Chapter 3: System Architecture

The following sections describe the systematic considerations of this project. We began with identifying what subsystems this project consisted of, along with the functionalities and constraints of each of them. Then we considered what different physical and software components were being used in the subsystems.

Our project consists of three major systems: flight system, flame detecting system, and water spraying system, shown on the right. As their names suggest, the flight system controls the overall movement and orientation of the drone; flame detecting system takes charge of sensing the flame using the near-infrared (IR) array flame detector; water spraying system will turn on the water pump as soon as the flame detector reading is above threshold value 850. We will discuss the water spraying and flame detecting system together since those two systems go hand in hand, controlled by the Arduino Nano microcontroller.

3.1 Flight System

If we look at the drone itself, it involves both mechanical and electrical constructions, which go hand in hand. The mechanical structure greatly impacts the performance of the electrical system in ways, for example, a poorly-constructed frame would cause so many vibrations that would interfere with the communication of the IMU sensors with the microprocessor on the FC. The electrical structure consists of all the components that require power to operate. With an inadequate electrical structure, the mechanical system could get damaged or not perform to its full use. Together these systems work to give functionality to the drone. The systems can only perform properly if both systems are up to the highest standard. Therefore, it was important for us to have a clear system-level picture of the drone structure.

3.1.1 Drone Structure

The drone we created was made in two versions. The first version was equipped with the Crazepony 2306 1700KV brushless motors, the Crazepony 6-inch 3-blades propellers, and 19-inch arms. The second version was upgraded with DYS 2836 1120KV brushless motors, 9-inch 2-blade propellers, and the same length but higher density of arms. The drone structure interlocks all the parts together. The structure is the foundation for the system and determines the layout of the parts.
Determining which structure to use for the drone came down to several factors, such as weight distribution, motor power, propeller size, and durability. The weight of the drone needs to be evenly distributed throughout or balanced so the drone can stay hovering in the air. The motor power of the drone needs to have a thrust-weight ratio above the weight of the drone so that it can fly without restrictions and no swaying. The structure needs to be durable enough to hold all the components of the drone and not easily break with landing impacts or crashes. All the parts need to be properly secured on the structure.

The parts that make up the drone structure include the arms, legs, bases, screws, washers, and zip ties. The four arms that hold all the motors and the ESCs are attached to the top and bottom bases (see section 2.1.4). The arms place the motors symmetrically and allow them to run properly. The four legs attach to the bottom base and stand the drone 5 inches above the ground. The legs absorb impact, which allows the drone to land smoothly. The assembly process of putting the structure together required access to 3D printing, a screwdriver, a drill, and a wrench. Once the parts were printed, it was quite simple to put everything together. The parts were printed with screw holes already in. Some holes did not print precisely and needed a drill to clean the holes up. The screws were put in with either a screwdriver or wrench.

We made several adjustments on the mechanical structure for our second model to improve the performance of the drone, by taking these aspects:

- Thickened the bases to increase durability and security
- Increased the infill of arms
- Reprinted arms for a cleaner print
- Used longer screws for attaching legs
- Used same type of screws for same parts to ensure even weight distribution
- Added washers and nuts to make the joints more secured

Below is a table showing comparisons between the two models of our drone in terms of their 3D-printed parts:
### Table 7 Comparisons on Two Models’ 3D-Printed Parts

<table>
<thead>
<tr>
<th>Parts</th>
<th>Arms</th>
<th>Bases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First Model</td>
<td>First Model</td>
</tr>
<tr>
<td></td>
<td>Second Model</td>
<td>Second Model</td>
</tr>
<tr>
<td><strong>Dimension</strong></td>
<td>40.72 mm x 213.55 mm x 55.05 mm</td>
<td>BOTTOM: 182.00 mm x 118.00 mm x 1.60mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOP: 108.15 mm x 108.15 mm x 1.65mm</td>
</tr>
<tr>
<td><strong>Infill</strong></td>
<td>25%</td>
<td>15%</td>
</tr>
<tr>
<td><strong>Screw sizes</strong></td>
<td>1.6mm (Leg-arm): 2.85 mm x 12 mm; M3 inserts</td>
<td>1.6 mm (Base-arm): 2.75 mm x 18 mm</td>
</tr>
<tr>
<td><strong>Misc.</strong></td>
<td>Longer screws to secure, added washers and heated inserts</td>
<td>Thicker bases to add stability and decrease strain on joints</td>
</tr>
</tbody>
</table>

The projected flight time with the first model structure is around 4.2 minutes. This number was calculated with several payload calculations. The payload is determined by the weight distribution, weight to thrust ratio, the type of the motors, the type of battery, the ESC power, the propellers, and the frame size. The detailed calculation can be seen in Appendix D, calculated by a payload calculating software called eCal.com. We planned on calculating the projected flight time for the second model, but due to the outbreak of COVID-19, we were unable to perform the calculation.

### 3.1.2 FC Electrical System

The flight controller electrical system includes a 3S (11.1V, 2200mAh) LiPo battery, four brushless motors, four 40A ESCs, IA-6B 2.4GHz receiver, and an F405-CTR FC. In order to get all those parts to work well together, we carefully performed soldering, and configurations that syncs the parts together. Although we soldered as soon as we started the project, we could not start configuring the electrical system until the drone frame was done being constructed. We needed to calibrate the FC at its final position so that the accelerometer and gyroscope on board could recognize the FC placement with respect to the frame. And the FC placement during the calibration should be consistent with the placement during takeoff.
3.1.2.1 System Schematics

Below is a diagram showing the connections among all components within the FC electrical system. The FC is the brain of this system, powered by the system’s only power source - 3S LiPo battery. The built-in power distribution board (PDB) on board allows each connected ESC to draw up to 40A to power the motors. In order to let the transmitter, control the motors, a receiver needs to be bonded to the FC. Since our receiver uses an iBus protocol, we needed to connect it to the RX2 port on the FC.

![Diagram of drone electrical system connections](image)

**Figure 21 Drone electrical system connections**

3.1.2.2 Power Rating

The maximum current rating on the FC is 184A (recommend 179A), and the battery voltage sensor is rated 110V. We calculated that the overall current being drawn and voltage being consumed are shown in the table below.

<table>
<thead>
<tr>
<th>Item</th>
<th>ESC</th>
<th>Receiver</th>
<th>Camera</th>
<th>System Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quantity</strong></td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Current Rating</strong></td>
<td>40A</td>
<td>2A</td>
<td>0.28A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Voltage Rating</strong></td>
<td>11.1V</td>
<td>4-6.5V</td>
<td>3.3-5.5V</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Total Current Draw</strong></td>
<td>160A</td>
<td>2A</td>
<td>0.28A</td>
<td>162.28A</td>
</tr>
<tr>
<td><strong>Total Voltage Consume</strong></td>
<td>44.4V</td>
<td>6.5V</td>
<td>5.5V</td>
<td>50.6V</td>
</tr>
</tbody>
</table>

Table 8 Total Current and Voltage Consumption of FC Electrical System
All of the components above could be powered by the FC properly within the FC I/O pin output current and voltage rating - 5V@2A (maximum 3A), and the maximum total current and voltage rating.

3.1.2.3 Flight Performance Calculations

We made use of eCal.com, a web calculator for quadcopters to project our drone’s flight performance, by providing the following specifications. The information was based on the parameters of the first model of our drone, which were equipped with the 2306 1700KV motors and 6-inch propellers:

- Model weight
- Battery configuration and capacity
- Controller current
- Motor KV
- Propellers type

The software calculated the projected hover flight time, thrust-weight ratio, electric power, and other important indexes, which can be found in Appendix D. We had done a couple of calculations with five different weight conditions: just the drone itself, carrying an empty bottle and all other necessary components for the flame detecting and water spraying system, carrying a bottle of 4oz water, carrying a bottle of 8oz water, and carrying a bottle of 16oz water. The table below shows the results for hover flight time and thrust-weight ratio for these five conditions.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>No Payload</th>
<th>w/ Empty Bottle</th>
<th>w/ 4oz Water</th>
<th>w/ 8oz Water</th>
<th>w/ 16oz Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hover Flight Time (minutes)</td>
<td>5.5</td>
<td>5.5</td>
<td>4.6</td>
<td>3.9</td>
<td>3</td>
</tr>
<tr>
<td>Thrust-weight ratio</td>
<td>1.8</td>
<td>1.8</td>
<td>1.6</td>
<td>1.4</td>
<td>1.2</td>
</tr>
</tbody>
</table>

The thrust-weight ratio is a good indicator of how possible the drone can take off. The recommended thrust-weight ratio for a quadcopter is 2:1. With the setup of our first model, when it carries no payload, our drone could take off fine and get a decent flight time. As the payload increases, the thrust-weight ratio starts decreasing. So we decided that in order to ensure the performance of our drone with payload, we could either use a battery with higher voltage, or get more powerful motors and propellers.
3.1.2.4 FC Programming on BetaFlight

As mentioned before, BetaFlight is the configurator we used to program the FC. And the main programming we did on BetaFlight involves the following:

- Serial ports configuration: configure the connection to each serial port (UART).
- Accelerometer calibration: let FC recognize the levelness
- Switches setup: set up switches for arming/disarming, and flight modes
- ESC calibration: make sure all the ESCs work properly
- Receiver channel mapping: verify transmitter sticks are tied to the desired channels through receiver
- Motor testing: make sure your motors are working and are spinning in the correct directions

We went through all the FC programming when we first started constructing the first model of our drone. And repeated the same process as we built our second model with more powerful motors and propellers.

3.1.2.5 Troubleshooting

After several fly tests, we kept encountering issues like the drone tipping off or not being able to hover in the air. With full stability, we had to redo several of the calibration steps, as well as other fixes to make sure the FC performs the way it should. For issues where the drone is not taking off, or tipping off during the takeoff process, we performed these debugging steps:

- Checked the motors, propeller blades, and ESCs, and made sure they weren’t broken.
- Calibrated gyros and accelerometer, and ensured the internal sensors were calibrated for level flying.
- Made sure propellers direction matched - two propellers spin Clockwise (CW) and two propellers spin Counter Clockwise (CCW).
- Calibrated transmitter, made sure all calibration offset were in the center.

We went through the list above one at a time to determine which step could fix the failure of taking off. We noticed that every step above did improve the drone performance every time, but the critical step that made the most improvement of the drone was calibrating the accelerometer. We learned that we were supposed to calibrate the accelerometer before every flight. And since the gyro got recalibrated automatically every time we reconnected the battery, it was recommended to not touch the drone for a while after being connected to the battery.

As the drone got better at taking off, we ran into another issue of severe drifting in the air, so we took the following steps:
- Adjusted the trim settings, made sure all trims were placed at the center position, so all channels pitch, yaw, and roll were set at 1500.
- Calibrated transmitter, made sure all four calibration offset were in the center.
- Calibrate accelerometers
- Changed Angle and Horizon mode parameters shown in the figure below

![Figure 22 Adjust Angle and Horizon mode parameters on BetaFlight](image)

We noticed that the accelerometer calibration and change in flight mode parameters did ameliorate the drone drifting issue. However, we suspected that the unbalanced weight distribution might contribute to the drifting as well. So in our second model, we would focus on balancing the weight distribution, and adding more thrust to the drone. This additional stability and thrust greatly improved the performance of the drone. While the first version of the drone required full thrust before any of the spraying system load was added, the modifications added to the second version increased the thrust enough that it could take off with the added load. The drone was much more maneuverable with less drifting until the spraying system was added. The sloshing of the water did decrease the stability but it was still more stable that the first version had been.

### 3.2 Spraying System

In addition to flying, the drone is required to also have a system which can detect flames and then deploy some flame retardant. This system was created with an independent power source and therefore, it is electrically isolated from the components controlling the flight of the drone.

#### 3.2.1 Spraying System Structure

Once the type of flame retardant was decided we were able to begin designing the system. Attaching a sensor and the end of the hose to a servo creates a wide angle of detection. Once a flame has been detected, the servo reduces to a much smaller angle of rotation, and the retardant begins spraying until the flame is no longer detected. The three near-infrared (IR) array flame detectors we chose were Arduino flame sensors manufactured by DFRobot. Within the operating temperature of -25°C to 85°C, they can detect light in a spectral bandwidth ranging from 760 nm to 1100 nm. The hose selected is an 8 mm silicone hose. The servo we used is a towerPro micro servo which has 1.6 kg/cm of torque, which is strong enough to move the hose and sensor with 180 degrees of rotation. An Arduino nano was selected to control all the components.
3.2.2 Electrical System

The sensor and servo only required 5V and not much current, so the Arduino board was able to supply the power. However, the water pump required more current than the board would be able to supply. We originally designed a complicated circuit where the I/O pin of the Arduino would control the operation of the pump without supplying the power. Professor Bitar helped us design a much simpler circuit which did not require as much hardware. The diagram of this simple circuit can be seen in Fig 23 on the right.

In this diagram the R2 represents the pump which has an internal resistance of 34.28Ω. The Shockley diode is a safety feature for the circuit. An NPN transistor was used as such that when V2 (the voltage at the I/O pin of the Arduino) is low, the collector voltage is pulled high and therefore there is no voltage drop across the load, and the pump is off. When the V2 voltage is high, the collector current is pulled low and the pump is turned on. A diagram of how all the electrical components are connected can be seen in a block diagram in Fig 24.
The system required two 7.4 V LiPo batteries, one for the Arduino, and one for the water pump. The signal pins on the Arduino control the operation of both the servo and the water pump, based on the signal received from the flame sensor. This algorithm is discussed in more detail in the next section.

### 3.2.3 Software

The goal of this system was to find fire and deploy the flame retardant. To do this the flame sensor and water hose were connected to the servo and moved with 180 degrees of rotation until a fire was detected at which point, the pump turned on and the angle of rotation reduced to ±5 degrees from when the flame was detected. It stays in this state until the flame is no longer detected. The state diagram for the Arduino can be seen in Fig 25 on the right. Thus, in the No Flame state, the servo is moving with 180 degrees of rotation and the pump is off. In the Flame state, the servo angle of rotation has been greatly reduced and the pump is turned on. The
reading on the flame sensor determines which state the Arduino is in. The threshold for turning on the pump is higher than the threshold required to switch states and turn the pump back off.

3.2.4 Second version improvements

When the spraying system was assembled and installed on the drone, it became clear that the with the design, the servo was not strong enough to move the hose the full 180 degrees. This was due to a combination of the servo torque and the flexibility of the hose. We upgraded to a servo which had more torque. Changing the servo required a change in the software so that the correct libraries were used, because the original servo was a 360-degree servo and the new one was a continuous servo. Additionally, a new hose which was 1mm thinner was installed allowing for more slack between the tank and nose. The combination of thinner walls and more slack created more flexibility along the hose. The increased flexibility and torque allowed the system to have the planned angle of rotation. Figure 26 shows the final spraying system after it was installed on the drone.

Figure 26 Final spraying system installed on drone

3.3 Combined System

The combined system creates the entire drone. All of these parts work together to complete the combined system. The combined systems include the drone, spraying, and software systems. The drone system includes the structure, flight controller, and electrical systems. The spraying system includes the spraying structure and the electrical system. The drone system allows for the quadcopter to fly and maneuver in the air, while the spraying system detects and suppresses flames. The component that bridges these systems together is the software, and power components. The BetaFlight software allows for all parts of the drone system to work with each other and properly calibrate in the air. All these components are powered through the flight controller. The Arduino software allows for the proper identification of flames and allows for the water pump to spray out liquid to the fire. All the spraying system components are powered through the Arduino controller except for the water pump. To bridge the spraying system to activate with the flight transmitter, a port of the Arduino is connected to the flight controller to start up the spraying with a switch on the transmitter. All these systems work together to make the complete system.
Chapter 4: Tests and Results

As we were designing and constructing our drone, we empirically tested all components of the drone, including the FC system, flame detection and water spraying systems. We performed all testing on the open fields either on or near the WPI campus, since large open fields are cleared of all obstacles, such as trees, wires, etc. This was critical to ensuring safe flights and protecting electronics on the drone.

Our testing with the final model frame, fully-functioned flame detection and water spraying systems were performed throughout the week of March 2nd, 2020. A brief breakdown of the results of different aspects of the system throughout our project is provided below.

4.1 Takeoff and Landing

There were several factors affecting the performance of the takeoff and landing:

1. The sturdiness of the drone frame
2. The security of the joints on the frame
3. The power of motors and propellers
4. The weather on the days that the drone got tested

The levelness of the surface from which the drone took off and landed on

The take-off at first was not properly functioning as the propellers and motors were not the proper size for the weight of the drone. Along with unfit propellers and motors, improperly stabilized FC on the drone resulted in tipping off. With the strong vibration of the drone when trying to take off, and without secured mounting to the drone frame, the FC either tilted or slipped off from the center of the drone, which caused the FC to think that the drone needed more thrust on one side. One other issue faced was a poor connection to the battery of the drone. The wires were not properly attached to the battery, causing a loose connection and the power easily shutting off.

The landing of the first model drone was not ideal, since the drone would never land safely and would crash or land on its side. The first factor that caused these failures was the flying environment. Due to the weather and location constraints, we could only test our drone originally on uneven surfaces, areas surrounded with shrubs and trees, and on cold, windy days. The environment greatly affects the landing as a clear landing area with stable weather conditions is needed for a successful landing. The second factor is the pilot controlling the drone. It is essential to know how to fly a drone, or else the landing could end in a crash or land with heavy impact, damaging the drone. Just as the take-off, a loose connection of the FC and battery would cause issues and result in tilting or crashing. In the end the landing was much more improved than the ones tested at the early stage of our project after fixing these factors. Several of the same issues occurred while the drone was in the air, which will be further explained in the next section.
Based on the testing results of our first model mentioned above, we improved our drone design in these aspects. And detailed changes between the first and second model can be seen in Table 7.

- Used larger propellers and more powerful motors
- Used a 3D-printed FC mount
- Increased infill and thickness of the drone frame
- Secured screws and joints on the frame
- Secured battery connection to the FC

By using 9-inch propellers, 1120KV motors with a size of 28*36 mm, and a 3D-printed FC mount, we were able to give the drone enough lift and keep the FC in a fixed position at the center of the drone. We also redesigned our drone frame by increasing the infill and thickness of the drone frame to ensure its sturdiness, so that the vibration could be significantly ameliorated. To fix the issue of poor connection between the battery and the drone, we used both the male and female XT60 battery connectors with sufficient soldering and heat shrinks. Additionally, to become better pilots, we had been using the simulation software FPV Freerider to practice flying in a virtual environment to have minimal damage as possible and understand how to properly land. The adjustments we made to our second models did greatly improve our drone’s take-off and landing performance.

4.2 In-air Performance

Our main methods to record our drone’s in-air performance testing results were through observation, and timing. In this section, we will talk about these aspects related to our drone’s in-air performance:

- Communication
- On-board sensors - how sensitive are the IMU sensors; how fast does the flame sensing system detect the flame.
- Calculated flight time vs. measured flight time for both models

When testing our drone’s communication with the ground operation, we looked at how well it responded to the transmitter signal. The Flysky - IA6B receiver provides six channels and has an RF sensitivity of -105dBm, and a working range of 500 ~ 1500m in the air. The radio control receiver protocols are iBUS, a new digital serial protocol specific to Flysky. iBUS is a two-way protocol that can send and receive data. While we were able to send data through transmitter to our receiver on board, we could also receive information sent from the receiver back to our transmitter, such as the receiver battery voltage level. For both models, we did not experience any significant difficulties in communicating with the drone while it was in the air. However, with our first model, we encountered severe drifting issues after the drone had taken off. Besides the ones already mentioned in section 4.1, the adjustments below gave our second model more stable, and smoother in-air performances.
- More even weight distribution
- Improved receiver antenna placement
- More frequent calibration of the FC accelerometer and ESCs
- Cleaner wiring and more secure connections

Our IMU sensors on FC include accelerometer, gyroscope, and barometer, each of which measures angular rate, force, and atmospheric pressure respectively. At the early stage of our testing, due to our inefficient calibration of the accelerometers, the drone had a problem identifying the movement change. Eventually we got that problem fixed, and the accelerometer could properly sense the angular movement change, and regulate the throttle of motors accordingly. We did not get a chance to test our drone with the flame detecting and water spraying system until the second model since we were worried that the motors on our first model would not be able to carry that much payload. When the second model was done constructing, we put on the flame detecting and water spraying system. Initially, the spraying system did not work as the servo could not provide the required torque. Then the second version of the spraying system was created and installed. This version had additional torque as well as a more flexible hose. Once this version was installed, it worked correctly.

Lastly, we recorded footage and flight time when the drone took off successfully. For the first model without any payload, we were able to get the drone up 13 feet in the air for a duration of 10 seconds, which was a lot lower than the projected flight time of 5.5 minutes. We did experience a crash from that test flight due to the drifting issue. Again, due to the outbreak of COVID-19, we were unable to calculate our measured flight time for zero payload, and with payloads of 4oz, 8oz, 16oz.
Chapter 5: Discussions and Recommendations

In this section we will cover what aspects of the system that performed notably well or otherwise. Additionally, we will provide any recommendations for future work or things we would try differently if we had the opportunity to redesign.

5.1 Flight Safety

Flight safety is a very important factor for drones. Without proper flight safety, the drone would be destroyed and even cause damages potentially to properties on the ground every flight. We could have definitely improved on our flight safety and allow for better flight tests. To improve on flight safety, we recommend prior to purchasing parts and building the drone to do extensive research and calculations on weight-to-thrust ratios, the sizes, weights, the voltage and current rating of each part, and their compatibility of other parts within the drone system. With the purchased parts make sure every part has proper documentation, so that there is no vagueness and difficulty finding specifications. With researching parts, find parts that are rigid and able to withstand force. We suggest looking into metal props and a structure made with a tough, but lightweight material such as carbon fiber. When the parts are all purchased, we recommend determining the best layout of the drone that allows for even distribution of the weight, and allowing the antennas to work to its full potential. To keep the antennas working properly, keep the two antennas as straight as possible, otherwise the control range is reduced. The antennas should be placed at a 90-degree angle to each other, as seen in Figure 27. It is important to keep the coaxials not bent further than 90 degrees. In order to have proper communication, the antennas must be at least 15 cm away from conductive materials such as metal and carbon. The last part to ensure the antennas working correctly, is to keep them away from the motors, propellers, ESC, and other parts creating loud noises. The next suggestion we have is to all be properly trained in controlling and flying the drone. Flying the drone is no easy task with no experience. We suggest finding online classes, purchasing a drone to practice flying, or use simulations to fly in a virtual space. It is also essential to fly the drone in a large open area so that there are no obstructions. To prevent bad crashes, setup a failsafe on the drone in case there is a disruption in communication
between the transmitter and receiver. Next, we will provide recommendations on the spraying system.

5.2 Fire Extinguish

In this part, we will talk about some possible actions to improve our drone’s fire extinguishing ability. First, we can start with using a larger water tank. Due to the payload constraint, we attached a 16oz-water bottle as our water tank. In the future, one should consider using a water tank with bigger capacity and lighter material for applicable use. Second, one could use some fire retardant instead of water. We decided to use water as our fire retardant throughout our project because it was the most affordable and accessible option. The other fire retardants are commonly accomplished by chemical reactions that reduce the flammability of fuels or delay their combustion. They are available as powder, to be mixed with water, as fire-fighting foams, fire-retardant gels, and even as coatings or sprays to be applied to an object. In reality, for forest-fire fighting, especially aerial firefighting, early fire retardants were mixtures of water and thickening agents, and later included borates and ammonium phosphates. Finally, using a flame detector that can filter out sunlight would greatly improve the fire-fighting ability. Our flame sensor is an IR array sensor that detects light in the spectral bandwidth ranging from 760 nm to 1100 nm. Since sunlight is composed of more than half of infrared light whose bandwidth is above 700 nm, it becomes the biggest distraction during our flame detecting system testing. In order to fight against sunlight, we came up with two alternatives based on some research. First, use the IR narrow bandwidth diodes that detect the intense long wave IR spectra instead of wavelength. Sunlight has ca 6000C but flame ca 1000 C. Therefore, the frequency will be shifted hardly. Additionally, the IR flux that flame has is hundreds of times stronger than that of sunlight. Second, use a narrow band interference filter and bolometer type sensor (Janis59, 2017).
Chapter 6: Conclusion

Drones are getting more and more prevalent in the public safety domain, especially firefighting since fire departments are seeing large benefits from the use of drones during the search and rescue missions in fires (Hayes, 2018). Being intrigued by this advanced tool used in the firefighting industry, we started building a prototype drone that can detect and suppress fires. For the past six months, we had conducted extensive research on topics including, the basic steps of building an RC drone, parts comparisons and analysis, and payload and flight time calculations. Then, we got our hands-on 3D-printing and assembling the drone frame, designing and testing the flame detecting and water spraying system circuit. We finally tested two different models of our drone. The goal for our first model was to get the drone up in the air, so it was not equipped with the flame detecting and water spraying system. We had encountered lots of obstacles, and challenges during the tests, but we were able to tackle each of them down, and roughly got the first model to take off. Learned from our first model, we enhanced our design for the second model with more secured joints, sturdier drone frame, and more powerful motors and propellers. These adjustments improved drone performance significantly. As we were more confident towards our second model’s flying performance, we attached the flame sensing and water spraying system onto the drone as well. We broke down the tests for the second model into drone flying with no water, flame sensing and water spraying system only, drone flying with a full tank of water, and lastly, a drone carrying a full tank of water and flame sensing and water spraying at the same time. We learned from each test either it was a failure or success, and our perseverance ultimately led us to the success of our model.

After completing this project, we suggested a couple of points that will be much helpful to start or expand on this project. First of all, it is essential to gather all research and calculations prior to ordering any parts. The most important part of it all is to understand the weight-to-thrust ratio. Without the proper ratio, the drone will not be able to fly. We also suggest using a durable and sturdy frame, so that the drone will not break when landing, or take minimal damage during crashes. To prevent crashes, we suggest that everyone is trained in drone flying to properly get the drone to take off, hover, and land without damage. One final suggestion is to use a water tank with more capacity and power to take down larger fires. We suggest this by using a larger model of the drone and using a fire suppression liquid other than water that would better combat the fires. With these suggestions, we hope that our project can be expanded upon and improved for further research and use.

Overall, it was a fruitful and rewarding experience participating in this MQP, as we had a lot more exposure to the applications of drones and the engineering field. We were able to put our electrical and computer engineering skills to use and demonstrate our success and understanding in the field. With this project, we are confident in our major and are eager to put our skills to use in the real world. And we also hope that there could be more improvements based on our project in the future.
Bibliography


### Appendix A: FC Pros & Cons

<table>
<thead>
<tr>
<th>FC Candidates</th>
<th>CL Racing F7</th>
<th>Holybro Kakute F7 AIO</th>
<th>MATEK F405 CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>Fast processor, high gyro sampling frequency, and good performance gyro with less oscillation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clean and convenient layout with well-sized solder pads</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soft mounting grommets to cut noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Built-In Camera Control for OSD plan display timer on Betaflight</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Built-in voltage regulator 3.3V@250mA, 5V/1.5A BEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1G Flash memory for BlackBox - maybe enough memory, adequate for 60+ min flight 4K recording</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4in1 ESCs. The ESC header supports both external current sensor input and ESC telemetry, so can connect motors directly</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has VTX Pitmode built-in – a feature to turn your VTX on and off using a switch on your radio. Pitmode is basically a feature to turn down VTX power near zero when you are not flying. In order to minimize interference to other pilots.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Affordable: $35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soft-mounted Gyro sensor allows good protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Betaflight OSD can display timer on the configurator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integrated PDB, with ESC built-in</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comes with voltage regulator 3.3V/0.2A, 5V/2A BEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enough storage for data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Onboard BMP280 Baro Sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>BlackBox memory is only supported by Betaflight 4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No baro sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The processor is a little bit more sensitive to noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complicated wiring, affect gyro detecting. Furthermore, you cannot put anything on top of the FC with things like a receiver or VTX.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The ribbon cable which might feel a little fragile and could break in a crash.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Little bit expensive: $50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Doesn’t support PPM and PWM receivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Long shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soft-mounted Gyro sensor allows good protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dedicated ESC telemetry pads right next to the ESC signal pads, which will make soldering much cleaner and easier.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comes with voltage regulator 3.3V/0.2A, 5V/2A BEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Onboard BMP280 Baro Sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Betaflight OSD can display timer on the configurator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reasonable price $40</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fairly lightweight</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No onboard 9V regulator for camera control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No ESC built-in</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Installing & Configuring BetaFlight
(Cited from https://www.youtube.com/watch?v=xmaTq4JgTXI)

BetaFlight is managed using the BetaFlight configurator application, also known for short as the BetaFlight GUI. The BetaFlight GUI is a Chrome app. To run it, first, we need to install Google Chrome on the computer using this link:

https://www.google.com/chrome/browser/desktop/index.html

Some people do not like Google and object to installing Google Chrome on their computers. There are a few options for configuring BetaFlight without Chrome, but they are mostly intended for quick changes in the field, not full-time use.

Next download the BetaFlight Configurator GUI from the Chrome App Store:

https://chrome.google.com/webstore/detail/betaflightconfigurator/kdaghagfopacdnlgbohiknlhcocjccjao?hl=en

Finally, run the BetaFlight Configurator GUI:

1. Enter the URL chrome://apps/ in Chrome’s address bar.
2. Click the icon for the BetaFlight Configurator.
3. The configurator will start.

To save time in the future, we can right-click the app icon in Chrome and choose “Create Shortcuts”. This will create shortcuts in our start menu and on our desktop, to launch the app directly.

More importantly, we need to install related drivers to get the optimal features of the configurator. The drivers we need are listed on the Welcome page of BetaFlight as shown:

![Figure 1: Required drivers for BetaFlight](image)

Figure 1: Required drivers for BetaFlight
The first driver we need is CP210, so click ‘here’ to go to the download page. Once we are on the download page, scroll down to the option for our operating system. We only want the VCP option for Windows 7/8/8.1/10 as shown in Figure

![Download for Windows 7/8/8.1/10 (v6.7.3)](image)

**Figure 2: Download options for Windows 78/8.1/10**

Once the download is finished, open up the zip file. We recommend to extra all the files to a designated folder on your laptop. Depending on the system type, run the x86 executable file if the laptop is 32 bits; run the x64 file if the laptop is 64 bits as shown below.

![Figure 3: Installed options for CP210 driver](image)

The next thing we need is the VCP driver. Click ‘here’ on the Welcome page in BetaFlight to go to the download page. Once we are on the STM-VCP download page(Figure), we have to first accept the license agreement by first clicking the ‘Get Started’ button then click the ‘Accept’ button on the top right corner.
After your first name, last name and email address are filled out, you will get a download email for the installation. You need to be aware that when the download is finished, the executable file is not the actual driver installer; it is a self-extracted archive. When you run the executable file, you just unpacked the archive on your desktop. Once the archived is unpacked, you should go into the folder and run the actual driver installer.

Notice that you only need to install the drivers one time on any given machine. After that, all you have to do is plug in any new FC that you get and the drivers will install themselves.

Now you can plug in the FC, and two LEDs should be lit up. The blinking LED (usually red or green) tells you that the FC is in a FC mode; while the static LED (usually blue) tells you the board status is fine. When you first plug in the FC through a USB cable, the blinking LED would flash quickly then slowly three times. That means the gyro calibration is finished and indicates the board is acting normally as a FC.

It is always recommended to flash the FC before programming, and the board must be in bootloader mode to be flashed. There are two ways that the FC connects to the laptop. One way is through a virtual com port (VCP) that does USB enumeration. The microprocessor on the FC is handling the USB protocol through VCP. The other way is through the CP210 chip on the board. Since the CP210 chip is handling all the USB protocol, the microprocessor on the FC has no idea that it is talking over USB. If you see a ‘USB VCP’ port under the ‘Ports’ tab in BetaFlight, that means the FC has VCP; otherwise, the FC uses the CP210 chip to handle the USB protocol.
With the VCP, there is one more step to do - download the Zadig. In order to flash the board, the board needs to be in bootloader or DFU mode. When it goes into DFU mode, and if the laptop has a virtual com port, the Windows drivers don’t work. So you need to download a different device driver in order to flash the PCB board on Windows. So click on the download link to Zadig, a tool that lets you install the driver. Once the executable file finishes downloading, you run the file and will see the window shown below.

![Figure 5: Zadig window](image)

First, click ‘Options’ > ‘List all devices’, and find your board. Do NOT pick ‘STM32 Virtual COM Port’, since that indicates the board is NOT in bootloader mode, and we do not want to overwrite the ‘STM32 Virtual COM Port’ with Zadig driver. The correct way to do it is to unplug the board from the USB port, hold down the bootloader button shown in Figure and plug the board back in. Now the LED does not flash, and it only gets one solid blue LED. That tells us that the board is in bootloader mode.

![Figure 6: Matek F405-CTR bootloader location](image)

Now that the board is in bootloader mode, we go back to the Zadig window. And pick the ‘STM32 BOOTLOADER’ as shown below. We can now click ‘Replace Driver’, and the driver installation should begin. When it is done installing, you may need to reboot your computer to get this work.
So now unplug the board, hold down the bootloader button on the board, and plug the board back. There should be a solid blue LED that tells you the board is in bootloader mode. Start BetaFlight configurator. On the top right corner, it should say DFU, which means that BetaFlight agrees that we are in bootloader mode. We can go to ‘Firmware Flasher’ to flash the board.
Appendix C: Parts Configuration

C.1 Bind Transmitter and Receiver

We first paired the transmitter and receiver with a bind plug, so that they know that they are talking to each other, even other signals present. The detail steps and corresponding figures are the following:

1. Plug the bind plug in the receiver. The first column from the right is the binding port.
2. We used the lab power supply to provide 4 V DC by connecting the positive and negative to the left-most pin on the second and third row on the receiver respectively because the second row from the bottom is V+ and the third row is Ground.
3. If the LED on the receiver is flashing, the receiver is ready for binding.
4. Now turn the radio on while holding the bind button, once started the screen should read RX binding.
5. Once the LCD display on the transmitter shows the RX bar on the top right corner, that means our transmitter and receiver are correctly binding together. And we can let go of the ‘Bind’ button.
6. After we remove the bind plug, the constant LED means that the transmitter and receiver are bound now. And the transmitter screen displaying both the transmitter and receiver battery also indicate the successful binding of the transmitter and receiver.

Figure 1: (Step 1) Insert bind plug
Figure 2: (Step 2) Flysky i6AB pinout

Figure 3: (Step 4) Press bind button to start binding
Figure 4: (Step 6) Successful binding of TX and RX
C.2 Assign Auxiliary channels

By default, the Flysky radio assigns Auxiliary channels 5 and 6 to the knobs A and B on the transmitter. Since it is rather difficult to precisely rotate those knobs while flying to change flight modes or other parameters, we assigned the Aux channels to switches A and C. We dedicated the switch A, the left-most two-position switch on the transmitter for arming and disarming, and switch C, the second switch from the right on the transmitter, the only three-position switch for flight modes, as shown in the figure below:

![Figure 5: Aux channels set to switches A and C](image)

The detailed steps are as follows:

1. Long press the OK button on the transmitter to access the radio main menu.
2. Navigate to the ‘Functions setup’ menu.
3. Scroll all the way down to the last option in the menu called ‘Aux switches’.
4. By default, all of the switches were disabled for some reason, but we wanted to enable switch A and switch C shown below. Long press the ‘Cancel’ button to save and return to the menu.
5. Now navigate to the ‘Aux. channels’ option.
6. Set channel 5 to the 3-position switch C for flight modes, and channel 6 to the left-most 2-position switch A for arming. Long press ‘Cancel’ to save the changes.

![Figure 6: (Step 2) TX ‘Function setup’ menu](image)  ![Figure 7: (Step 3) TX auxiliary switches option](image)
C.3 Set Up Aux Mode in BetaFlight

We need to sync up the auxiliary mode in BetaFlight with the following steps:

1. In BetaFlight Configurator, go to the ‘Modes’ tab, and click ‘Add Range’ next to the ‘Arm’ tab. The screen will change to look as shown below:

Figure 10: Add range for arming/disarming in BetaFlight

2. Turn on the transmitter and connect the FC to BetaFlight via USB. The small yellow indicator will move to indicate the current position of the Aux1 channel (Switch A) as shown:
3. Put the arm switch in the position that we intend to use as “armed”, which is commonly pushed away from the operator. By this logic, the switch pulled towards the operator should be disarmed, because it is easier to slap the switch that direction as we are trying to disarm in a hurry. Now we should see the yellow indicator move as we move the switch (This won’t work if you haven’t bound the transmitter and receiver.)

4. With the switch in the “armed” position, drag the yellow range marker to the end of the channel (value of 2100) where the yellow indicator is. Now put the switch in the “disarmed” position, drag the yellow range marker to the point with a value of 1000 shown below.
5. Click ‘Save’ in the lower-right of the window to save all changes.

C.4 Set Failsafe

Finally, we set up the failsafe feature, which can be triggered by a switch, the loss of radio link or an unexpected RC command pulse. In any of those cases, the quadcopter will know and can take appropriate action, instead of just flying away. The basic rule: it is better to drop from the sky unarmed than have a flyaway, randomly chasing people (Rowe, 2019). We only enabled the failsafe on the throttle channel (channel 3) so that the FC will know when it has lost the link with the radio. We set up the failsafe mode by following these steps:

1. Go to ‘System setup’ > ‘RX setup’ > ‘Failsafe’ menu on the transmitter as shown
2. Scroll to ‘Channel 3’ and press the ‘OK’ button. Move the throttle stick all the way to the bottom, and turn the failsafe on.
3. Long press the Cancel button to save, and now you should see that Channel 13 is set to -100% (Figure 15). If this is not the case, try again, making sure the throttle stick is all the way down and the trim is also all the way down.

4. Lastly, we enabled and set up the failsafe on the FC. After the FC is connected to BetaFlight, click the ‘Enable Expert Mode’ button on the top right corner as shown (Figure 16) to obtain the full menu.
5. Now go to the ‘Failsafe’ tab as shown below, and look at the ‘Channel Fallback Settings’. Set ‘Roll[A]’ to ‘Auto’, ‘Aux 1’ to ‘Set’ with a value of 1100, and ‘Aux 2’ to ‘Set’ with a value of 1100.
6. Under the ‘Stage 2 - Failsafe Procedure’, select the ‘Drop’ option as shown below. Lastly, click ‘save and reboot’ to apply all changes.

C.5 Calibrate FC in BetaFlight

When we first got the Matek F405 CTR FC, we placed it on a level table, pointed it in a direction that we set as the front, and connected it to the laptop via a USB cable. BetaFlight simulation software can automatically detect the factory configuration of the controller when connected. Before we did everything else, we wanted to make sure the FC is stationary on a very level surface. So we went to the ‘Setup’ tab in BetaFlight (Figure 20) and clicked the ‘Calibrate Accelerometer’ button (Figure 21).
By doing so, we let BetaFlight know the correct orientation of our quadcopter. The 3D model of our FC (Figure 22) in the middle of the screen on BetaFlight stimulates the real-life drone movement. For example, when we tilted the FC down, the 3D model also tilted down, which proves that the accelerometer on the FC functions properly.

Shown above right, the green arrow on the 3D stimulation indicates the forward direction of our quad in real life.

We also discovered that we could manually type in the sensor configuration to tell BetaFlight the correct orientation. This is done in the ‘Configuration’ tab and, more specifically, in the ‘Board and Sensor Alignment’ box (Figure 23). We need to change the angle (in degrees) until the model moves correctly. The most common correction needed is to add some Yaw offset. So if the model on screen rolls when we pitch the quad forward, we would need to change the yaw degree by 90 degrees. The most convenient way is to use the drop-down box to select a preset angle offset. Lastly, hitting ‘Save and Reboot’ after the changes would allow us to save the changes.
C.6 Verify Channel Mapping

Next, we must verify that our transmitter is configured correctly for the default configuration with these following steps:

1. Connect the Matek F405 CTR FC to laptop via a USB cable
2. Start the BetaFlight Configurator. Click the ‘Connect’
3. Go to the ‘Receiver’ tab. Move the transmitter sticks, and make sure the sticks move the corresponding sliders as following:
   a. Moving the left stick up should move Throttle up (higher number).
   b. Moving the left stick left should move Yaw down (lower number).
   c. Moving the right stick up should move Pitch up (higher number)
   d. Moving the right stick left should move Roll down (lower number)

4. If any of the controls move the wrong channel, we need to re-order the Channel Map parameter in the top-center of the ‘Receiver’ tab screen to correct them. The channel map parameters are: AETR1234
   - Channel 1 = Aileron = Roll = Right stick left/right axis
   - Channel 2 = Elevator = Pitch = Right stick up/down axis
   - Channel 3 = Throttle = Left stick up/down axis
   - Channel 4 = Rudder = Yaw = Left tick left/right axis

   For example, if our Yaw and Roll were swapped with each other, we would change the Channel Map from AETR1234 to RETA1234. The ‘1234’ at the end of the map will always be the same.

Rearrange the A, E, T, and R in the channel map until the controls are as described in step 3

![Channel Map Setting Example](image)

C.7 Verify Channel Directions

If the channels are mapped correctly but move in the wrong direction, one must reverse the channel in his/her transmitter. Following are the instructions for reversing channels on a FlySky FS-i6 transmitter:

1. Power on the radio
2. Then hold down the ‘OK’ button to enter the ‘System Setup’ screen
3. Press the down button to highlight the ‘Functions Setup’ option
4. Press ‘OK’ to enter the ‘Functions menu’
5. The selection arrow will be pointing at ‘Reverse’. Press ‘OK’ to enter the Reverse menu
6. Press ‘OK’ to cycle through the channels (Figure 25). Press Up or Down to choose whether the channel direction is Normal or Reversed. Reverse the required channels so that the channels move as described above in the BetaFlight ‘Receiver’ tab. Figure 25: Reverse channels directions

C.8 Adjust Channel Centers

If the channels do not center properly, our drone will drift when we center the sticks. To check whether our channels are adjusted correctly, first, we need to center our transmitter sticks.

After connecting the FC to BetaFlight, we went to the ‘Receiver’ tab, and looked at the Roll, Pitch, Yaw, and Throttle channels. These channels should be at exactly 1500. If they are within the range of about 1498 to 1502, then it is OK to skip this step. If they are much different than 1500, the channel center points must be adjusted with the following steps:

1. Power on the radio
2. Then hold down the ‘OK’ button to enter the ‘System Setup’ screen
3. Press the down button to highlight the ‘Functions Setup’ option
4. Press ‘OK’ to enter the ‘Functions menu’
5. Press the Down key to move the selection arrow to ‘Subtrim’. Press OK
6. Center all of the transmitter sticks. The spring-loaded sticks will auto-center. And the Throttle stick needs to be centered manually.
7. In the Subtrim screen, ‘OK’ cycles through the channels. Up/Down raises or lower the channel center. With the transmitter sticks centered, adjust the sub-trim for channels 1, 2, 3, and 4 so that the channel reads as close to 1500 as possible in the BetaFlight ‘Receiver’ tab.

C.9 Adjust Channel Endpoints

If the channel endpoints are not adjusted correctly, the quadcopter may rotate faster in one direction than the other. In extreme cases, we may not even be able to arm. To check whether our channel endpoints are adjusted correctly, first have all propellers removed and plug the FC into a laptop via a USB cable.

Then go to the BetaFlight ‘Receiver’ tab. Lower the throttle stick all the way down. The Throttle channel should have a value of 1000 at the lowest point. Raise the throttle all the way up. The Throttle channel should have a value of 2000 at the highest point.
Now push the yaw stick all the way to the left. The Yaw channel should have a value of 1000. Push the yaw stick all the way to the right. The Yaw channel should have a value of 2000. The same should be true for Pitch and Roll: down or left should be equal to 1000; up or right should be equal to 2000.

If the channel endpoints are not equal to 1000 and 2000 when we fully deflect the sticks, we need to adjust the channel endpoints with following steps:

1. Power on the radio
2. Then hold down the ‘OK’ button to enter the ‘System Setup’ screen
3. Press the down button to highlight the ‘Functions Setup’ option
4. Press ‘OK’ to enter the ‘Functions menu’
5. Press the Down key to move the selection arrow to ‘Endpoints’. Press OK
6. In the ‘End Points’ screen(Figure), OK scrolls between channels. Up/Down adjusts the endpoints. We control whether we are adjusting the top or bottom endpoint by raising or lowering the stick for the channel we are adjusting. This can be a little confusing because it is probably not obvious to most people which channel goes with which stick. The default channel order for the Flysky radio and for BetaFlight is:
   - Channel 1 = Aileron = Roll = Right stick left/right axis
   - Channel 2 = Elevator = Pitch = Right stick up/down axis
   - Channel 3 = Throttle = Left stick up/down axis
   - Channel 4 = Rudder = Yaw = Left tick left/right axis
7. For each of the channels 1, 2, 3, and 4, adjust the top and bottom endpoint. Do this by holding the correct stick full left or down to adjust the bottom endpoint and full right or up to adjust the top endpoint. Then use the up/down button on the FlySky FS-i6 to adjust the endpoint percent (%) until the BetaFlight Configurator ‘Receiver’ tab shows 100 for the bottom endpoints and 2000 for the top endpoints.

C.10 Configure Receiver Protocol in BetaFlight

After we successfully bound all the motors to the FC, and controlled them on BetaFlight, we moved on to controlling the motors with the transmitter. Firstly, we needed to build the connection between the receiver and FC by using three female to male cables. Based on the receiver and FC documentation, for an iBus protocol, we ought to connect the rightmost three pins on the first row on the receiver (Figure 26) to the following pads on the FC (Figure 27).
We double-checked that all three of the wires were secured and correctly connected, we ran the BetaFlight simulator and connected the FC to the laptop via a USB cable. In order to build the communication between the FC and receiver, we went to the ‘Ports’ tab to configure the receiver protocol. Since our FlySky FS-iA6B Receiver has an iBus protocol, we enabled the ‘Serial RX’ column under default UART2 port (Figure 28).

After clicking the ‘Save’ button, we went to the ‘Configuration’ tab. Under the receiver section, we needed to tell BetaFlight what type of receiver we use by selecting the ‘Receiver Mode’ to ‘Serial-based Receiver’, and ‘Serial Receiver Provider’ to ‘iBus’ as shown in Figure 29. Once done press the ‘save’, we clicked the ‘reboot’ button to apply the settings.
Appendix D: 1st Model Flight Performance Calculation

D.1 No payload
D.2 With empty water tank
D.3 With 4oz water
D.4 With 8oz water
D.5 With 16oz water
Appendix E: Spraying System Arduino Code

```c
#include <Servo.h>

#define flame_pin A7
#define servo_pin A5
#define pump_pin A4
// #define flame_inputV 3.3
// #define servo_inputV 5.0
#define servo_start_angle 0
#define servo_end_angle 45
// #define interval_num 1
#define low_threshold 750
#define high_threshold 850

Servo servo_test; // initialize a servo object for the connected servo
int constant_reading = 0;
int direct = 1;
int angle_found;
int count;
int angle = 0;
int sig;
int pwmVal;

void setup() {

  // set pins as outputs
  pinMode(flame_pin, INPUT);
  pinMode(servo_pin, OUTPUT);
  pinMode(pump_pin, OUTPUT);
  servo_test.attach(servo_pin);

  Serial.begin(9600); // BAUD rate
}

void has_found_flame() {
  while (1) {
    servo_test.write(sig);
    constant_reading = analogRead(flame_pin);
    Serial.println(constant_reading);
    if (constant_reading < low_threshold && count > 300) {
      angle = 0;
      break;
    }
    angle += 1;
    if (angle >= 10) {
      angle = 0;
      servo_test.write(95);
      if (sig == 0) {
        sig = 180;
      } else {
        sig = 0;
      }
    }
  }
  digitalWrite(pump_pin, HIGH); // water pump is powered
}
```
Serial.println(constant_reading);
delay(10);
count++;
}

//Constantly read flame sensor analog value 0 - 1024
void loop() {
  digitalWrite(pump_pin, HIGH);
servo_test.write(sig);
digitalWrite(pump_pin, LOW); //water pump is not powered
constant_reading = analogRead(flame_pin);
Serial.println(constant_reading);
if (constant_reading > high_threshold) {
  count=0;
  angle_found = angle;
  has_found_flame();
}
angle += 1;
if (angle >= 35) {
  angle=0;
  servo_test.write(95);
  if (sig == 0){
    sig = 180;
  }
  else{
    sig=0;
  }
}
delay(10);
}