G.O.D. (Gesture Operated Drone)

Group 3 - Pranay Patel, Anshul Devnani, Bernardus Swets
Computer Engineering Majors
Senior Design 1 Final Report – August 2nd, 2019
EEL 4914 Summer 2019

Table of Contents

List of Figures

List of Tables

1.0 Executive Summary

Drones have become increasingly popular over the last decade. Every year their abilities are rapidly increasing and we want to do our part to add to this continuously growing field. With the knowledge we have obtained throughout our studies we want to challenge ourselves and develop a drone that strictly controlled by human hand motions. Our team wanted to build a product that combined every aspect of our computer engineering coursework. With this in mind, we were able to collaborate on the idea to utilize PCB construction, embedded programming, and machine learning to build a useful and sound product.

Because we, as a team, believed that drones can sometimes be difficult and so wanted to offer the ability to control a drone with a much simpler interaction. This allows for us to build in actions for the drone that will automatically account for stabilization and move as expected without having to try to keep the drone flat and level via remote control. This product has not yet found its way on the market and we want to be the first to make this a reality.

Our plan to create a widely marketable product forced us to take into account the usability of the product, the cost, and the ability to use devices that customers were already familiar with to allow for an easier interaction with the drone. We were able to fulfill all of those goals in our project plan. We were able to maximize usability by creating hand gestures to control the drone that are generally universal, meaning that people around the globe can understand many of the basic gestures and would immediately think to use those gestures to operate the drone. In order to make the cost of the drone low, we were taking into account the cost of each and every component when planning our prototype, and once the build process and components list is finalized, we will be able to improve upon that even further. Since our product requires the use of an external device, we decided to make it so that other people can simply install the software required to operate the drone on their local laptops or PCs. This allows for users to interact with something they are familiar with and make the drone user experience more seamless.

Throughout this document we explore why creating this drone would be beneficial, our creative process and map out how we plan to build and test the device. Consisting of four main components, we will have a user interface, the drone flight controls, power system, and the communication network all working in harmony to control the drone. With the simple hand motions explored in the following sections, the drone will be able to perform all the necessary actions needed for flight.

2.0 Project Description

We are proposing a small indoor drone that is entirely driven with hand gestures. The project will have a user-friendly webcam-based GUI, that will communicate with the drone. The GUI's main component will be the webcam, along with other indicators showing the drone's current status and useful live information. From the users end, the user will perform the desired hand gesture and the drone will react accordingly. For example, the user will signal a thumbs up and the drone will respond, within a reasonable response time, and increase its flying altitude. We have a set number of hand movements we plan to incorporate. As the project develops, we will add functionality and push ourselves to create as many movements as possible. As a group of computer engineers, we have a fundamental understanding of the programming and electrical skills necessary for this project. We plan to develop the flight controller ourselves and incorporate and apply the corrective features of a closed loop system. Furthermore, this project will allow us to work with and learn about popular technologies of the time, including Computer Vision and Machine Learning. Both are emerging industries and are growing rapidly. As this is a self-funded project and there are currently no sponsors, our goal is to make this project as low cost as possible. This will ensure that the product can be supremely accessible to the public and can be improved upon moving past our first prototype.

2.1 Project Motivation

When discussing all the options for possible senior design projects, we had a couple ides of varying complexities and price points. This project was on the more complex and relatively more expensive, however we were most enthusiastic about researching and creating this project. With all three of us equally eager to research and plan this project, it justified the higher cost and the increased complexity. As strictly computer engineers our education covered a wide variety of topics and overall this project incorporates everything we have learned. In the latter half of our education we took many courses regarding microcontrollers, communication networks, creating graphic user interfaces, programming embedded systems, and PCB routing/design. Individually we also chose to study computer vision and linear control systems which play key roles in the project. With this knowledge we will build a product that, from our extensive research, has not been built before. This is also another motivating factor. Both drones and computer vision are extremely popular, and in some cases the two have been combined for tracking purposes. That being said, there is not yet a commercial product that is strictly controlled by hand movements. Being the first to achieve this would be an extremely satisfying accomplishment.

Flying a drone for the first time can be fairly complicated and can give a user a lot of trouble. With the use of your own hand movements, it can add a sense of ease and fluidity not found in a typical hand-held controller or smartphone. In addition, our solution involves controlling the drone with one hand, which is unlike traditional drones in which you use both hands to operate a physical remote controller. Our solution will allow the user to only need to use one hand to control the drone, as long as that hand stays in the correct field of vision. This will allow for freedom of motion for their alternate hand, which is something that is overlooked often when it comes to drones. Oftentimes, drones are used to record something in motion, whether it be action sports outdoors or photographers and videographers trying to get a birds eye view that isn't easily attained without one. Given this, allowing for a free hand will immediately be beneficial to drone users.

2.2 Goals and Objectives

Our objective is to create a low-cost hand gestured controlled drone. We have limited time to complete the project, and we want to make the most of our time. After spending the last few months researching and planning the project, our goal is to start building in early August 2019 and have a working prototype by the end of September 2019. Once we have a working product, our goal is to add as many hand signals as possible. We are starting with eight essential hand signals and we strive to get that number up to around 15 different hand signals. Another objective of ours is to make the build process as simple as possible and as repeatable as possible. During our production process we will most likely need to spend more money on replacement parts during testing and other unexpected factors. Once we get the working product, we can limit our design to the bare minimum of what needs to be completed and make the project as affordable as possible.

2.3 Requirement Specifications

When describing our requirements, we did our best to ensure every requirement was abstract and quantifiable. **Table 1** shows software requirement specifications, **Table 2** depicts hardware requirement, and **Table 3** displays system requirement specifications. As we get more involved in the project, we may notice that some limitations we set might be extremely lenient or we might have set the bar too high. Because of this, we are open to altering or adjusting our requirements as we see fit.

2.3.1 Software Requirements

Table 1 specifies the software level requirements for our project

The drone will be able to convert the signal received over Bluetooth within 500 ms.
The user interface will be able to recognize each of the 8 gestures.
The feedback/reading pane will highlight the correct predicted gesture within 1 second.
The Neural Network will produce an accuracy of a minimum of 95 percent.
The user interface GUI will consist of a webcam pane, log pane, and miscellaneous pane.

Table 1 Software Requirements

2.3.2 Hardware Requirements

Table 2 specifies the hardware level requirements for our project

The drone frame will be no larger than 150mm.
The drone will not weigh more than 2 pounds.
The drone will be powered by 3.7V lithium polymer batteries.
The microcontroller will be powered by a 9v DC battery.
The drone will utilize propellers of 3 inches or smaller.
The drone will utilize 4 electronic speed controllers to help control the propellers.
The drone will utilize 4 brushless motors with KV above 900.
The drone will utilize an ATmega328P microcontroller.

Table 2 Hardware Requirements

2.3.3 System Requirements

Table 3 specifies the system level requirements for our project

The drone will be able to receive signals over Bluetooth communication from within a range of 20 feet.
The drone will be able to react to commands within 1 second.
The drone will be able to land, and motors will terminate within 5 seconds.
The drone will be able to take off to 3 feet within 3 seconds.
The Bluetooth signal will maintain connection within 15 feet.
When the drone's Bluetooth signal is lost, the drone will hover in place and land within 10 seconds.
The time from the user doing the gesture to the drone reacting to it will be a maximum of 2 seconds.
The drone will communicate its current altitude to the GUI with a maximum latency of 3 seconds.
The drone will maintain its altitude when moving left, right, forwards, and backwards.
When the drone accelerates in a specific direction, it will rotate less than 90 degrees to perform the given action, as to not tip the drone over.
The drone's altitude will be able to be read with a maximum 1 second delay on the miscellaneous pane of the GUI.
The drone will be able to reach a height of 10 ft.

Table 3 System Requirements

2.4 House of Quality

The image below, **Figure 1**, is the proposed house of quality for our drone design. We compare the model we are planning to create with top market competitors including DJI, GoPro and PowerVision.

			Functional Requirements										Customer Competitive Assessment			
		Direction of Improvement	▲	▼	▲	□	□	□	▲	□	□	□				
Relative Weight	Customer Importance	Customer Requirements	Weight	Size	Durability	ATMega328p	Gyroscope	Indoor Drone	LiPo Powered	Alitiude Sensor	Accelerometer	Brushless Motors	Gestured Operated Drone	DJI	GoPro	PowerVision
19%	5	Long Battery Life							•				2	5	4	3
4%	1	Lightweight Frame	•	○	▽			▽					3	2	1	5
7%	2	Bluetooth Connectivity		•			○		○				2	5	4	3
19%	5	Auto-Leveling				▽	•				•	○	2	3	4	5
7%	2	User Friendly Design	▽	▽	○								4	3	5	2
19%	5	Remoteless Controls				○							5	3	2	1
7%	2	Short Range Communication						•		▽			5	1	2	3
4%	1	Low Noise						•				•	5	2	4	1
11%	3	Quick Reaction Time				○							2	5	4	3
4%	1	Visually Appealing		▽				▽					2	3	5	4
		Importance Rating Sum (Impo	40.74	88.9	25.93	107	189	107.4	188.9	7.407	166.7	88.89				
		Relative Weight	4%	9%	3%	11%	19%	11%	19%	1%	16%	9%				
		Our Product	1	2	3	4										
		Technical Competitive Assessment														

Correlations	
Positive	+
Negative	-
No Correlation	

Relationships		Weight
Strong	•	9
Medium	○	3
Weak	▽	1

Direction of Improver	
Maximize	▲
Target	□

Figure 1 House of Quality

3.0 Standards and Constraints

3.1 Constraints

Along with our planning comes a lot of different constraints that govern the choices we make and the path we take with our project. Those range in various types such as economic, environmental, ethical, health, manufacturability, safety, social, and sustainability. This project is fully funded by our group. We came up with the idea of this project as a group and did not involve any third parties. As a result, we do not have any sponsors for our project. It is nice to plan the project ourselves however the assets from a sponsor would alleviate some of the economic stress. We understand that we will have to allocate a lot of money to fund the project. As students we have limited funds and want to do our best to make our project as affordable as possible. There are a lot of steps we can take to make that more plausible. A lot of the components we are buying are fairly sensitive and need to be taken care of properly. If we can avoid breaking pieces unnecessarily, we can save a lot of money in the long run. Also, if we do more research beforehand, we cannot waste money on the wrong parts. Another benefit of scanning the market thoroughly is finding the best balance of price and quality where we can obtain the best option possible. Having to buy replacements is inevitable but limiting the number of mishaps will lessen the economic constraints. We do not have unlimited funds and we kept that in our minds when we choose our parts.

Environmentally our drone will be constrained by its ability to only be flown indoors. It is not east to find indoor spaces where flying the drone is allowed without permission. It is important that our drone is capable of performing well in tight situations. Having the constraints of four walls around the drone can complicate some of the testing. To work around this, we received permission from our local gymnasium that is going to work with us. They have extra indoor basketball courts that are used throughout the day, but they have given us the times when the gym is typical empty and free for us to fly our drone around. When we are unable to use the gym, we can use our own personal garages. These have far less room to work with but have enough space to practice basic maneuvers and test what needs to be looked at. Environmentally we are also legally constrained. To fly the drone out doors in the state of Florida a license is required. To save money and time we decided to avoid flying the drone outdoor completely and to focus on only flying indoors. The benefit of this is the controlled environment that we have indoors. There are no factors like wind and rain to worry about. Having an indoor drone will lessen the constraints of the more unpredictable conditions of the outdoors.

There are also socially acceptable and ethical places to fly the drone. Before flying a drone in any location, it is important to have permission, whether this is a public gym or our personal apartments, it is essential we let everyone know that we will be flying a drone. It is not socially acceptable to fly our drone over people.

Drones sometimes have a negative connotation and are often banned, as they can be a disturbance. Drones are typically associated with having cameras and even though our drone does not have a camera, people may feel as if we are spying on them. It would also be unethical of us to fly our drone in certain places. If we fly our drone in the wrong places not only can be it be illegal but also offensive. Because of this we are going to limit the places we fly our drone. We are going to ensure that we do not cross any ethical or social borders when testing and flying our drone.

Drones can be dangerous, and we need to know the safety and health constraints going into our building process. There are extremely fast-moving parts that can be damaging if touched. We cannot cover the propellers and they need to be exposed for the drone to function properly. Knowing this we will stay clear of the drone while in flight or while the propellers are turned on. Luckily our lightweight indoor design does not pose as much a threat as some of the heavier commercial drones. If a collision were to unfortunately happen, there would most likely not be any major injuries however the possibility is out there. Other than injuries from the drone, there are no other health and safety constraints.

When manufacturing the drone there are a couple constraints that we need to be aware of. One of the main constraints is the range of our device. Bluetooth has become more advanced and can range quite far however we believe that once we cross 200ft, our design will no longer be able to connect. As this is an indoor drone, we will most likely not exceed these limits however, in the right setting, that might be a possibility and we need to be aware of this. As we are classifying this as a lightweight indoor drone, we have size constraints to fit that classification. We do not want to have drone that is heavier than two pounds. Some more advanced better functioning components are heavier, and more expensive, so this constraint encourages us to get the most efficient cost-effective part. We also do not want the frame to exceed 150mm.

Two of the largest constraints that we have to work around are the number of recognizable gestures and the battery life. It is important that the hand gestures we define are different enough to be recognized by the webcam. If a hand gesture is too close to another there could be a mistake that occur. As we test our initial flight gestures, we will have a better understanding of how similar we can make them. Our initial gestures are very different, but as we increase movements, we will need to find hand gestures that are unique enough to not interfere. The battery life is another obstacle we have to work around or try to overcome. Further into the document we discuss potential way to increase the battery life however for the time being we need to work with the limited battery life we are expecting. Implementing rechargeable batteries will help a lot some of the financial stress replacing batteries will cause.

In the table below is a more concise and clearer version of some of the topics discussed above. A lot of these values are more quantifiable. As the project goes on, we might discover that some of the values we found in research or predicted might be wrong and are subject to change.

Constraint	**Value**
Drone laws	Flying outdoors
Wireless range	Less than 200ft
Drone Frame Size	Less than 150mm
Drone Battery Runtime	20 minutes
Drone Weight	Less than 2 pounds
Number of gestures	At least 7 gestures, but limited, as similar gestures may be hard to differentiate by webcam
Budget	Affordability

Table 4 Project Constraints

3.2 Project Standards

When working on a project, standards are essential as they create a level of quality and expectation across the board. It also helps make the project adaptable and easy to incorporate. If another company or team were to incorporate our project, they would easily be able to adapt to our industry standard protocols. In **Table 5** below, we map out the standards that we are following.

I2C Communication Protocol
IEEE 802.15.1 (Bluetooth)
UART Communication Protocol
ISO/TC 20/SC 16 (Unmanned Aircraft Systems)
IPC-A-610

Table 5 Project Standards

Both I2C and UART are very common communication peripherals, and most third-party sensors and devices we are using are compatible. Most flight controllers utilize UART, while all of the sensors we have looked into our I2C. Using the I2C bus can significantly simplify and make our design more efficient.

Drones all must follow the ISO/TC 20/SC 16 standard for unmanned aircraft systems. A drone is an unmanned aircraft system and these standards map out what is allowed and what is not allowed in regard to locations to fly your drone. This allows for a more responsible and better educated population of drone operators, which is especially important as drones gain popularity.

IEEE defines Bluetooth as a standard for Wireless Personal Area Network (WPAN). We decided to use this standard for our wireless communication because it is heavily supported and continually updated. This allows for us to implement a technology that is familiar to the common user and is a respected engineering standard.

IPC-A-610 is the Acceptability of Electronic Assemblies. This standard will verify that our product has a highly reliable printed wiring assembly. This is a crucial criterion for our project to meet because it will verify our product even further to allow it to be more marketable. This will also allow us to proceed to manufacture the product faster because it already meets the industry standard and does not need to be verified in that regard again.

4.0 Project Design

Information in this section outlines our approach in designing our Gesture Operated Drone prototype. Majority of the research we have done regarding the project will be in this large section. This covers all the flight controls, physical drone properties, communication and the computer vision aspects of the project.

4.1 System Block Diagram

Figure 2 depicts the proposed block diagram for the project. All blocks are currently in the research phase. Our system design is divided into 4 groups, Application/GUI, Power, Drone Hardware, and Wireless Connectivity.

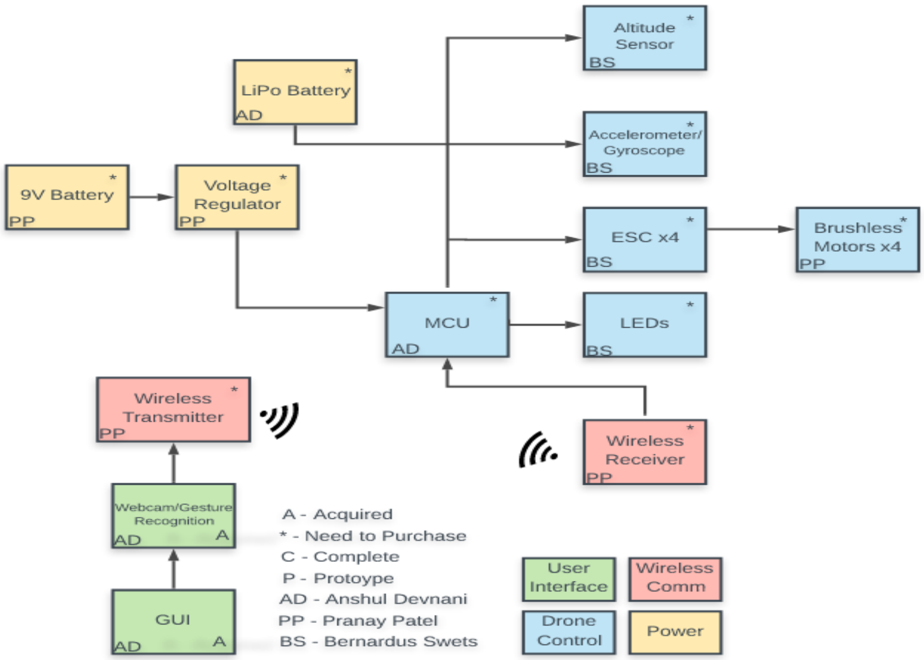
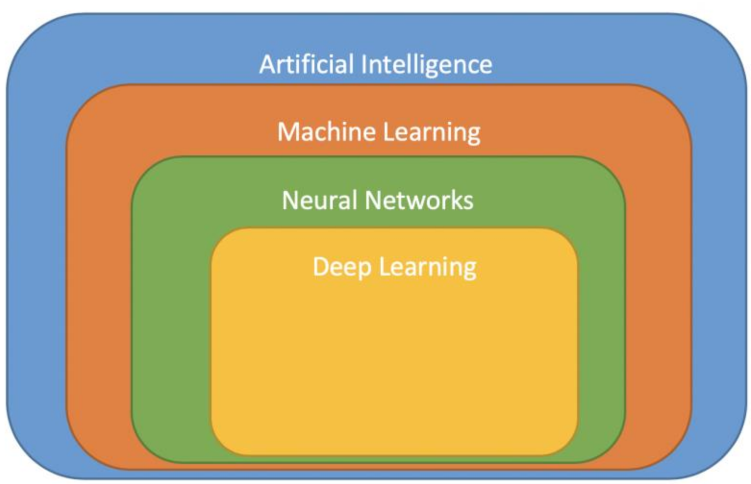


Figure 2 System Level Block Diagram

4.2 Neural Networks Overview

Before building a Neural Network application, understanding of the components, features, and constraints of Neural Networks is necessary. Neural Networks are a subset of Machine learning in terms of hierarchy. **Figure 3** shows the hierarchy of different concepts in artificial intelligence.



1

Figure 3 Hierarchy of AI

Artificial Intelligence is a broader group that encompasses Machine learning. Machine learning allows a system to learn and progress from past inputted data without being explicitly programmed. Neural Networks are a subset of machine learning because certain components and properties of Neural Networks allow for this learning to occur. Essentially, a Neural Network is a set of algorithms that are designed to recognize patterns and learn from these patterns to perform some task without being explicitly programmed to do so. Some popular applications of Neural Networks include speech recognition, object detection, image processing, and text recognition. Neural Networks are modeled after our brain and how our brains processes information. They consist of interconnected nodes or neurons that take in input from and give output to different neurons. All nodes are connected via weighted edges. A weight represents the strength of a connection between nodes and governs how much influence one node has on another. The higher the weight between two nodes the higher the influence that node has on the other. Neural Networks are typically trained on a some set of data, while this training is occurring the weights are updated in order to give optimal results. Neural networks are also split up into 3 generalized layers, the input layer, the hidden layers, and the output layer. **Figure 4** depicts the general architecture of a neural network. The input layer provides the initial data for the neural network. The hidden layers are the between the input and output layers and is where all the computation and learning is done. The more hidden layers that exist, the deeper we say the Neural Network is. The number of hidden layers in a network all depends on the machine learning application itself. The output layer is the final layer in the network and produces a final result. The idea of having a machine train itself to process and learn from data without explicitly teaching the machine is known as deep learning. The hidden layers of the neural network allow for this learning to occur.

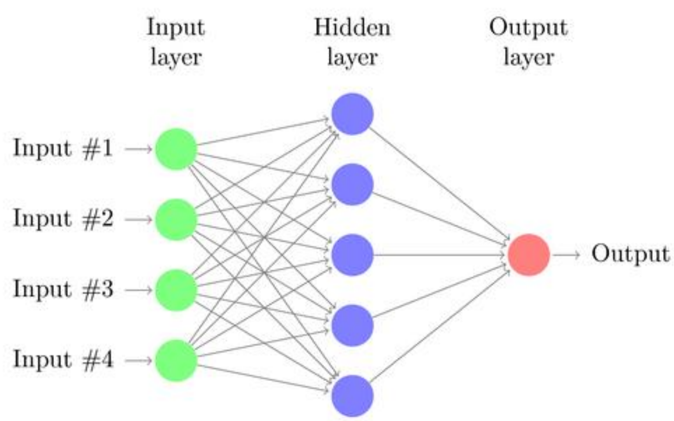


Figure 4 Neural Network Architecture
Permission to use from open source

4.2.1 What are Convolutional Neural Networks (CNNs)

In today's day in age, there are many different types of neural networks, some examples include, Recurrent Neural Network, Long/Short Term Memory, Convolutional Neural Networks, etc. For our project, the neural network that we will choose to implement is the Convolutional Neural Network. This specific type of neural network help bridges the gap between computer vision and deep learning. Convolutional neural networks have proven to be effective in areas related to image recognition and classification and have been very successful in tasks related to object detection. We chose to implement a Convolutional Neural Network in our project because of these facts. The challenge of accurately recognizing and classifying hand gestures in real time can easily be solved by training a Convolutional Neural Network. **Figure 5** shows the basic architecture of a CNN.

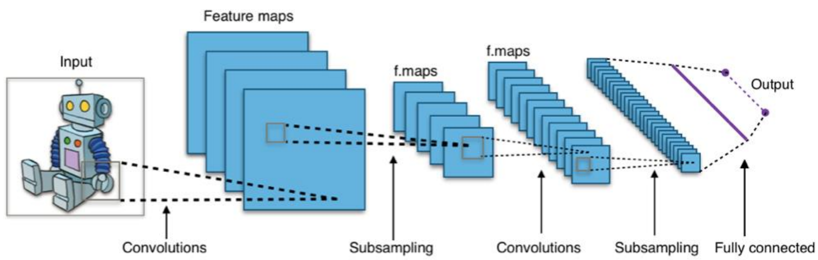


Figure 5 Basic CNN Architecture
Permission to use from open source

CNNs take an image in as input, in our project this will be an image of a hand gesture. Next, the image is sent through hidden layers where the image is broken down and different features of the hand gesture image are extracted and learnt by the network. For example, some features that can be extracted are edges and corners. A close fist hand gesture image will have different looking edges than an open palm hand gesture image. As the features are being extracted and learned, the weights associated with each node in the network are modified to account for newly learnt features. This is referred to as the feature learning stage. The classification stage is where the network makes a prediction on what it thinks the input image is or classifies the image based on the features the network extracted. In our project, an input hand gesture image can only be one of eight different hand gestures therefore the network will need to classify the input hand gesture image as one of eight different classes. The specific components that go into feature learning and classification are known as the building blocks of the CNN and will be discussed in section 4.2.2.

4.2.2 Building Blocks

Before building a Convolutional Neural Network, understanding of the certain building blocks is necessary. With a proper understanding of each building block, it is possible to create a robust and accurate Neural Network. In the subsequent sections, characteristics of each main building block will be explained as well as how each building block will be used in creating the Gesture Recognition Neural Network.

4.2.2.1 Convolutional Layers

Convolutional Layers are an essential part of Convolutional Neural Networks. The main purpose of convolutional layers is to extract features and detect patterns from the input image. Patterns in images can be anything from edges, corners, circles, squares etc. A specific filter is used within convolutional layers to detect specific patterns. A filter is essentially a matrix that is used to convolve over the input image matrix. **Figure 6** portrays what computation occurs in the convolution layers.

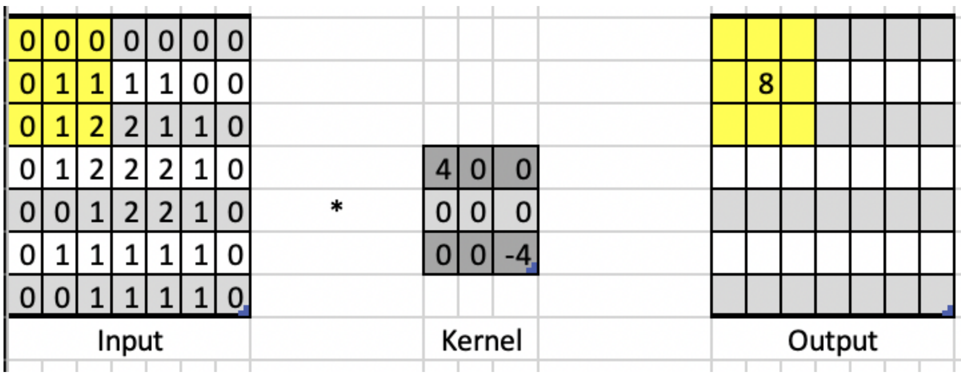


Figure 6 Convolution Layer Computation

The convolution layer essentially does the convolution operation on two matrices. One of these matrices is the kernel or filter and is usually a 3x3 matrix. The other matrix is the image in matrix form. The values that make up the image matrix are all the pixel intensity values. For example, a 50x50 image is converted to a 50x50 matrix with 2500 different pixel intensity values ranging from 0 to 255. The convolution is the dot product of the two matrices. According to **Figure 6**, the kernel can only perform the dot product on a 3x3 region of the image matrix at a time. After the convolution operation is complete, the result is saved into a new matrix and the 3x3 kernel acts like a sliding glass window and shifts over one pixel to the right. The convolution process then repeats itself, saving the result in a different matrix, until the whole input image matrix has convolved by the kernel. The matrix in which the results of the convolution operation is saved is known as the feature map. A feature map or activation map is a mapping of where different kinds of features are found in the input image. In essence, within the convolutional layers, there are different filters that are used to extract different features of the input image. The number of feature maps is determined by the number of filters used in the convolutional layer. There is one feature map per filter used. Convolutional layers will be used in our model architecture as they prove to be the most efficient way to extract different features from our hand gesture dataset.

4.2.2.2 Pooling Layers

The one limitation of feature/activation maps is that they are sensitive to the location of features in the input image. For instance, the feature map of a closed fist hand gesture will look different than the feature map of another closed fist that is slightly rotated. The goal is to create a model such that the correct hand gesture regardless of translations, a closed fist should be recognized as a closed fist regardless of how its rotated or translated. To solve the sensitivity issues, pooling layers will be used in our model architecture. Pooling layers solve this issue by essentially down sampling images. By down sampling images, small features will not be captured and only the more robust and general features are retained. This idea is referred to as local translation invariance, minute features should be ignored but broader features should be captured.

Pooling works by summarizing the features present in feature maps in patches and is used on the feature maps after the activation function has been applied. **Figure 7** shows an example of the pooling process

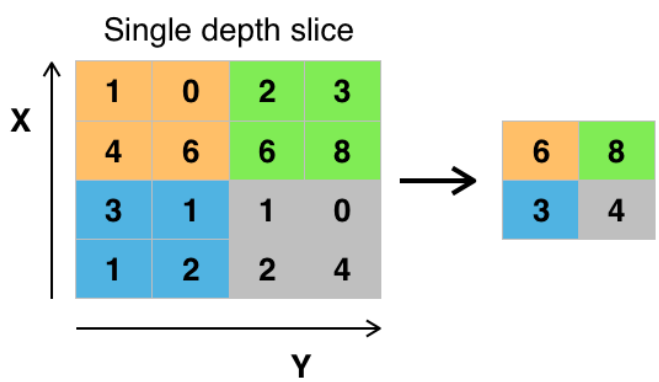


Figure 7 Pooling Process
Permission to use from open source

Essentially, pooling works by splitting the feature map matrix on the left into patches, According to **Figure 7**, these patches are 2x2 boxes. The highest pixel value is taken from each patch and copied to a new matrix on the right which is ¼ the size of the feature map matrix. This process is repeated for every 2x2 patch until the down sampled matrix, on the right, is completely filled. The resulting pooled matrix is fundamentally a summary of the features detected in the input and helps provides invariance to small changes or translations in the input. If the input is translated a small amount, the pooled matrix values should not change.

4.2.2.3 Fully Connected Layers

Fully connected layers are typically used at the end of the model architecture in the classification stage. The convolutional and pooling layers allow the model to detect features, but the fully connected layers use the detected features to classify the input images. The output of the feature learning phase is set of feature maps that have been through multiple convolutional, activation, and pooling layers. In order to achieve classification, these feature maps need to be flattened and mapped to a N dimensional vector. N represents the number of classes the model can assign an input image to. In other works, if the last layer of the feature learning phase outputs a 14x14x3 volume, it means there are 3 feature map matrices all of size 14x14. This output volume is then mapped and connected to vector of size 588 since 14 * 14 * 3 equals 588. This vector is again mapped to another fully connected layer known as the output layer of dimension N. For our project, the fully connected output layer must be of dimension 8 since there are 8 different potential hand gestures that can be recognized. The actual classification occurs when the output layer is applied a SoftMax activation function. By applying a SoftMax activation function to the output layer, the output vector is transformed into a vector of probabilities of what class the model believes the input image belongs to. In our project, fully connect layers will be used with SoftMax activation in our Convolutional Neural Network model because if provides us an efficient way to achieve classification within the model itself as opposed to using an external conventional classifier, like a Support Vector Machine, which adds to the complexity of the code and overall computation time.

4.2.2.4 Activation Functions

Activation functions are critical to the learning performance of a convolutional neural network. These functions are inspired by certain activity in our brain. Different brain neurons are activated by different triggers. The main purpose of an activation function is to convert an input signal of a node to an output signal so it can be used in the next layer in the model architecture. The weighted sum of each node in the network is inputted into the activation function, the resulting output is a number bounded between a lower and upper limit and is used in the next layer of the model. In Convolutional Neural Networks, activation functions are used after convolutional layers and fully connected layers. If activation functions are not applied to layers, output signals between nodes would be a linear function. Linear functions are constrained by their complexity and will not be as powerful when learning features from image data. Therefore, in order to make the model more robust and powerful in its ability to learn from image data, it is essential to introduce non linearities in our model. Non linearities are introduced in our model by using activation functions as it makes the easy for the model to adapt to different types data. The most common activation functions include Sigmoid, TanH, and ReLU. **Figure 8** shows the graphs of these activation functions.

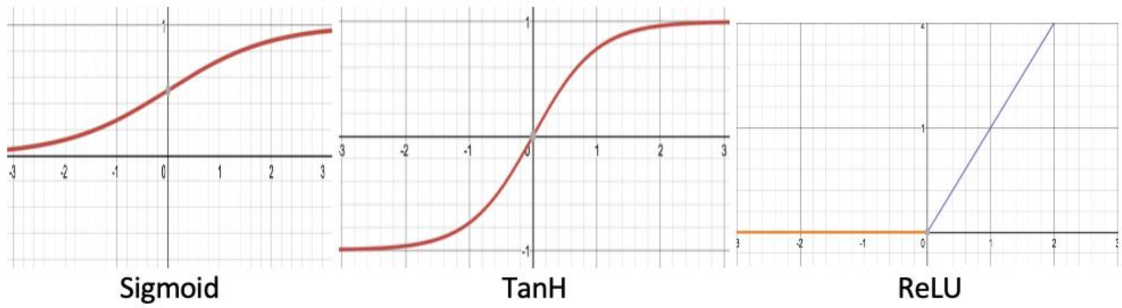


Figure 8 Common Activation Functions

The Sigmoid activation function takes in an input signal of a node and transforms the signal between 0 and 1. If the input signal is a negative number, this number will be transformed to a value close to zero. If the input signal is a positive number, the signal will be transformed to a value close to 1. If the input signal is close to zero, it will be transformed to a value between 0 and 1. The closer the transformed signal is to one, the more “firing” or active the node is in the network. If the transformed signal is close to zero, the less active the node in the network is. Since the sigmoid activation functions maps signals between zero and one, it is typically used for models that predict probabilities because probability of something is always between zero and one. In practice, the sigmoid activation function suffers from many issues such as the vanishing gradient problem which makes this activation function not has popular today.

The Tanh activation function is preferred over the Sigmoid function due to the fact that it is zero centered meaning the function is bounded between -1 and 1. Very negative input signals get mapped to -1 whereas very positive input signals get mapped to 1. Input signals close to zero are mapped to values close to zero. The Tanh activation function, however, still does not solve the vanishing gradient problem.

The Rectified Linear Units or ReLU activation is the most popular activation function used today. If an input signal is zero or negative, it will be mapped to the value of zero. If the input signal is greater than zero it will be mapped to that same value. Therefore, this activation function only has a lower bound of zero. The one main advantage of the ReLU activation function is that is solves the vanishing gradient problem.

For our project, the plan is to use ReLU after each convolutional layer and fully connected layer. Since the ReLU activation involves simpler mathematical operations it proves to be more efficient and less computationally expensive than the Sigmoid and TanH activation functions. Because of this fact, using ReLU activation can lead to better model performance.

4.2.2.5 Putting It All Together

By combining these layers in a certain order, the model architecture is built. Typically, in a conventional convolutional neural network the order in which the programmer places the layers are as follows, the convolutional layer followed by the activation layer followed by the pooling layer. An activation does not follow a pooling layer due to the fact that the pooling layer only down samples the feature maps and its outputs don't need be normalized by an activation layer. Fully connected layers are typically found at the end of the network and are typically followed by the output layer or more fully connected layers. The big question when putting together the different layers to create the model architecture is how many different layers to use. There is no set standard on how many layers to use as its all based on the application and characteristics of the dataset. For our project, we don't foresee using a lot of layers since our application of the neural network, which is to recognize hand gestures in real time, will not need many layers of abstraction to accurately differentiate between gestures. We are confident that keeping our network shallow, ie. Not using as many layers, will meet our requirements of accurately recognizing different hand gestures and doing so in real time. The specifics on what layers our model will utilize and the order the layers will be arranged are presented in section 4.3.4.

4.2.3 How do CNNs Learn/Train

Convolutional Neural Networks learn through a process called Backpropagation and takes place during the training of the neural network. This process is split up into 4 different stages, the forward pass, the loss function, the backward pass, and weight updating. Throughout the forward pass stage, the input data is passed through the model. In our project, the input data that we will pass through our model are hand gesture images. Because the weights are randomly chosen at the very beginning of the model training phase, the output classification predictions or probabilities will be very uniform in nature. For instance, if an image of a closed fist hand gesture is sent through our model in the earlier stages of the model training phase, the expected output classification probabilities would be around 15 percent for each class of hand gestures. Having uniform classification probabilities specifies that the model, with its current node weights, can't extract enough features from the input image to help make an educated prediction about what the classification of the image may be. The loss function is then computed to measure how different the predicted classification is from the actual ground truth label of the input image. The more different these two are, the higher the loss value. The lower the loss value, the more accurate the model is. There are many popular loss functions we can configure our network to use but the one that we will use in our model architecture is known as the Cross-Entropy Loss function. This loss function is popular to use with classification problems because the loss value increases as the predicted classification probability deviates from the ground truth label. One important aspect of using this loss function is that it penalizes severely classification predictions that are confident by wrong. For example, the loss value will be extremely high if the neural network model predicted a thumbs up hand gesture, but a closed fist was actually gestured by the user to begin with. Every time a loss value is calculated, the goal is to find which weights or nodes contributed most to the loss in the network, this occurs during the backward pass stage. During the backward pass stage, the weights that effected the loss the most are found by taking the gradient of the loss function at each weight. The gradient of the loss function is simply the derivative of the loss with respect to weight of each specific node or, $\frac{d(L)}{d(W)}$ where L represents the loss and W represents the weight of the specific node. After the derivative is calculated, the last step is to perform an update of the specific weight value tied to each node. In order to calculate the new weight value for each node, the value of the derivative is multiplied by a number known as the learning rate. Choosing the learning rate value is up to the programmer. A good learning rate value will allow for the model to converge on an ideal set of weights that gives the best prediction accuracy. A learning rate that is too high will result in big changes in weights which will lead to non-optimal results. For our project, we will start by using a learning of .001 and will adjust this value if the loss in the model is not improving. As stated before, the gradient of the loss function or derivative is multiplied by this learning rate to achieve a new weight value. The new weight replaces the old weight associated with the node. The process of backpropagation occurs at the end of each training iteration and is repeated until all weights are updated to achieve minimum possible loss and highest possible accuracy for the model. **Figure 9** illustrates a flowchart that describes the backpropagation process of one iteration.

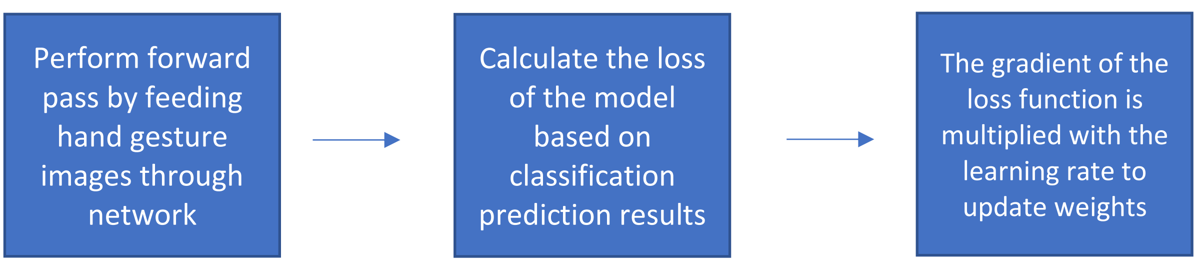


Figure 9 Backpropagation Flowchart

4.3 Gesture Recognition Neural Network

Building a good gesture recognition application is an immensely important aspect of this project. Failure to create a robust recognition application will not only lead to wrong gesture recognition predictions but also lead to drone control issues. One of our main goals of this project is to create a model that will produce extremely accurate gesture predictions based on the users given gesture. Machine learning and Neural Networks are great for applications in which classification of data is involved. For example, you are creating an app that can classify what dog breed a specific dog is in real time using your phone camera.

Trying to approach a solution to this classification problem without using machine learning would prove to be time consuming and inefficient because the developer will need to come up with and hard code complex algorithms in order to teach the computer how to differentiate between different dog breeds. Using Machine learning and Neural Networks, the developer can give the computer the chance to learn what all the different dog breeds look like beforehand so when given new input data, ie. A picture of a German Shepard, the output prediction will be a German Shepard. In our project, the classification task at hand is categorizing different hand gestures in real time. Just like in the example given above, trying to use non machine learning techniques would pose to be extremely difficult and complex. Therefore, our solution to this classification problem allows for the computer to learn the physical characteristics of a set of different hand gestures and will be able to accurately predict a newly inputted hand gesture, in real time. There are many aspects into creating and deploying a robust and accurate neural network application. If the steps in creating a Neural Network are followed correctly it can be surprisingly simple to achieve a highly accurate predictions (97% accurate or more). These facets will be explained in detail in subsequent sections.

4.3.1 Hardware Requirements

Solutions to classification problems using Machine learning and Neural Networks are extremely computationally expensive. The main reason being that the basic building blocks for machine learning computation is matrix multiplication and convolution. These tasks may not seem as computationally demanding in itself but when training a neural network, specifically a convolutional neural network, millions or even billions of these matrix multiplications and convolution operations need to be completed. The training of a neural network can take days even weeks on a basic office computer with average hardware specifications. Therefore, it is imperative that the correct hardware is used so that Neural Network training time and prediction time is minimized.

There are many different types of hardware that can be used to successfully create and run Machine learning applications. Some of the different types of hardware include Central Processing Units or CPU's, Graphical Processing Units or GPU's, Field Programmable Gate Arrays or FPGA's, or Specialized Accelerators. When it comes to Machine learning we need the right hardware that will be able to lower prediction time, achieve higher throughput through training, and lower power costs. Being able to speed up the matrix multiplication and convolution operations will ultimately lead faster training time. Since training a Neural Network takes the most time in creating and deploying Machine learning applications choosing the right hardware to help minimize computation time is key. Out of the hardware types listed above, the Graphical Processing Units are used the most used in the Machine learning world with Central Processing Units being second most popular. The main advantage of Graphical Processing Units is that they handle mathematical computation significantly faster. Computer graphics in general, involve an immense amount of matrix mathematical functions therefore these Graphical Processing Units are designed specifically to minimize computation time. Because of this fact, Graphical Processing Units are far superior to any other Machine learning hardware when it comes to training Neural Networks as most of the intense computation is done during this stage. As mentioned before, the deeper the Neural Network, the more intense the computation gets. The Central Processing Unit can also be used to train Neural Networks and is used most on systems with integrated cards.

For our project, the system that will be used to train our Neural Network will be a 2017 MacBook Pro. The basic specifications for this system are shown in **Table 6**.

Processor	3.1GHz dual-core Intel Core i5
Memory	16GB 2133MHz LPDDR3
Graphics	Intel Iris Plus Graphics 650 (Integrated)
Storage	512GB SSD

Table 6 Host Computer System Specifications

Since the graphics card on the system is an integrated graphics card, the processor will be used as the computation source when training the network. Due to hardware restrictions and our budget, we do not believe it is feasible to buy an expensive GPU just to train the model. As mentioned before, the CPU can handle Neural Network training computation, just not as fast as a GPU. Training time is highly dependent on training data dimensions and size as well as network architecture. For example, a network with 100 layers and 5000 images of input data with dimensions 720 x 480 will train a lot slower than a network with 50 layers and 5000 images of input data with dimensions 50 x 50 if the same hardware is used to train the model. A CPU can be used for our application since the input training data will be small and the network architecture will not be as deep. The specifics of the training data and network architecture will be discussed in later sections. Overall, the 3.1 GHz dual-core Intel Core i5 will be a capable processing unit that will be able to train the model with an estimated training time of less than 24 hours.

In the event that the MacBook Pro CPU cannot handle the computational requirements of training and real time recognition of hand gestures, we will be forced to consider other approaches to help boost our computational power. There are many options to help solve the computational restrictions we may face during the training and deployment of our gesture recognition network. One option is to buy an external GPU and connect it to the MacBook Pro to help give enough computation power in order to speed up training time and the deployment of the gesture recognition application. The major downfall of this approach is the cost of acquiring this hardware. External GPUs tend to cost around $500 USD which will essentially double are proposed budget. Due to budget restrictions, acquiring and using an external GPU will not be the approach to solve potential computational restrictions. The other option we can turn to for solving this issue is to use Machine learning as a Service or MLaaS. MLaaS provides users with Machine learning tools and algorithms via a cloud computing service. Some of the best know providers of MLaaS include Microsoft Azure, Amazon Web Services, and Google Cloud. Amazon Web Services offer an abundant amount of services geared towards machine learning. One popular service AWS offers is Amazon SageMaker which allows one to build, train, and deploy machine learning models. The big advantage of using this service is that a developer does not need to learn complex machine learning algorithms as there are tools and wizards that allow you to create the machine learning model without generating any code. Google Cloud's machine learning engine is another popular cloud computing service for machine learning tasks. This engine is built upon the TensorFlow framework which makes this engine highly flexible. Google Cloud's machine learning engine allows users to both use a GUI to implement neural network models or use an environment dedicated to coding the model from scratch. Microsoft Azure's ML studio is Microsoft's version of implementing machine learning tools in the cloud. The main disadvantage is that there is a steep learning curve in using ML studio and everything from data preprocessing to exploring the model results need to be done manually. ML Studio's GUI interface, however, allows for easy building, training, and deployment via its drag and drop GUI mechanism.

For our project, the first option is to use the existing hardware, the MacBook Pro, to train and deploy our model as this is the most cost-effective approach. In the event that our hardware does not meet the computational requirements of performing gesture recognition in real time, we will explore the options described earlier. Google Cloud's machine learning engine will be option we will choose if we need to upgrade our computation throughput. Google Cloud offers the cheapest price point for using its machine learning tool with monthly fees of $52 per month. Another attractive aspect of Google Cloud's machine learning engine is that it provides environments to both code and use a GUI to create neural network models. In our opinion, being able to code Neural Networks from scratch allows for better flexibly during the development stage of the model. Again, using these cloud computing services is a backup plan if our current hardware does not meet computational and accuracy requirements. However, we are confident that our existing hardware will perform well enough to meet these requirements.

4.3.2 Software Choices

In order to start building a Machine learning application, software related decisions need to be made. When starting a new Machine learning project selecting the right programming language, development environment, and API/Framework are all crucial decisions that can either allow for seamless creating and deployment of a Machine learning application or cause the developer many issues if wrong decisions are made.

There are many factors that go into choosing the right programming language for a Machine learning application. Factors such as robustness, readability, ease of coding, experience with the language, documentation/support, and most importantly, compatibility with Machine learning APIs and frameworks. Some of the most popular programming languages for machine learning today are Python, Java, R, Lisp, and Prolong. Lisp is one of the oldest AI suited language. Some features of Lisp include ease of creating new objects, ability to process symbolic information, automatic garage collection, and good prototyping capability. Prolog is similar to Lisp in the machine learning aspect. Features of Prolog include automatic backtracking, tree-based data structuring, and efficient pattern matching. R is a programming language that is used mainly for statistical data manipulation. With the right packages installed R can be a powerful tool for machine learning usually with raw data. Java is one of the more popular general-purpose programming languages. In addition to the easy use, widespread support, and the number of packages available, Java can handle computation required by machine learning such as search algorithms and neural network model building. Python is another popular general-purpose programming language but has even more regard in the machine learning world. Python has a very simple syntax which, in turn, allows for readability and coding ease. In addition, there is an immense number of libraries that make programming certain tasks easier. Most importantly, popular machine learning API's and Frameworks are compatible with Python. Based on these factors, Python is the language that we will choose to code our Neural Network application.

After selecting the right programming language, where you develop the application, or the development environment is an important software choice in the overall software development lifecycle. Choosing the right development environment can save the developer and immense amount of time especially when creating a Neural Network model. There are two options for development environment either an Integrated Development Environment (IDE) or a Text Editor. Some examples of IDE's include PyCharm, Eclipse, and Visual Studio. Examples of text editors include Atom, Sublime Text, and Visual Studio Code. A pure text editor is just a place for one to write code. There is no ability to run code from within the text editor application or check for syntax errors before run time. Usually when one wants to run code written in a text editor, the command prompt is used to call and run the code. Text editors are used mainly for coding small programs and typically not used for big projects. Integrated Development Environment are far superior to basic text editors as IDE's contain all the functionality of text editors and much more. A big feature of IDE's is that most comprise of built in debuggers. A developer can code and debug their program within the IDE as opposed to having a separate compiler when using a text editor. Some other features of IDE's include automatic code completion, built in project file explorer, package installers, and being able to run code with a click of a button. Therefore, an IDE will be used for the development environment of this project. The only restriction when it comes to selecting an IDE is programming language. It is imperative to select an IDE that is compatible with the programming language being used for development. The PyCharm IDE will be used for our development environment. PyCharm is an IDE created by Jet Brains and is an IDE geared towards developing Python and Django projects. PyCharm is compatible with Mac OS, which is the operating system that our project will be developed on. In addition to having all the features described above, the main reason PyCharm was selected is because of the free educational license Jet Brains offers for students. With this license we are given the full product at no cost.

Now that the programming language and development environment choice has been made, the next major software decision is selecting a machine learning API/Framework. There are many different APIs and frameworks geared towards Machine learning that allow for one to create Neural Network models easier. Some of the most popular API's and frameworks include TensorFlow, PyTorch, and Keras. TensorFlow is an open source library developed by Google that is used for building Neural Networks. PyTorch is another open source machine learning library specifically for Python and was developed by Facebook. Keras is an open source neural network API that is built on top of TensorFlow and is primarily used to create and experiment with deep neural networks. There are many factors that go into selecting the right Machine learning API/Framework for the project such as ease of use, debugging, and dataset considerations. As for ease of use, these API/Frameworks all operate on different levels of abstraction. Keras is a higher-level API where commonly used functions are wrapped in callable functions. PyTorch is a lower level API where the programmer can do more customization when creating the Neural Network Model architecture. TensorFlow is more of a middle ground between Keras and PyTorch in terms of abstraction. **Figure 10** and **Figure 11** show code for creating a simple Neural Network Model using Keras and PyTorch respectively.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPool2D())
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

Figure 10 Example Keras Code for Creating a Model

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3)
        self.conv2 = nn.Conv2d(32, 16, 3)
        self.fc1 = nn.Linear(16 * 6 * 6, 10)
        self.pool = nn.MaxPool2d(2, 2)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 6 * 6)
        x = F.log_softmax(self.fc1(x), dim=-1)
        return x
model = Net()
```

Figure 11 Example PyTorch Code for Creating a Model

It is clearly shown using Keras is easier to both read and code. Which ultimately leads to easier debugging. Keras is said to be the easiest to debug whereas TensorFlow is the hardest with PyTorch coming in as the middle ground. The final consideration when choosing the right machine learning API/Framework is the dataset. The input dataset is the data being fed into the network in order to train the model. Keras is used when dataset is typically small. For example, if the input dataset consists of thousands of images, Keras would be a good choice as it is comparatively slower. PyTorch and TensorFlow are optimized for speed therefore a good choice for larger dataset, usually millions of input dataset images. Our dataset will be relatively small, consisting of thousands of images of different hand gestures. Given all the stated considerations, Keras will be the API/Framework used to build, train, and test our Neural Network model.

4.3.3 Building the Dataset

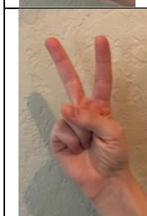
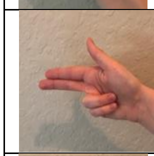
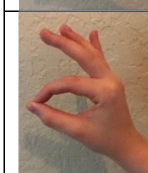
User Action	Result
No Gesture	Hover in place/autolevel
	Thrust Upwards
	Drone flies forwards
	Drone flies to the left
	Drone flies to the right
	Drone lands in current position
	Drone flies backwards
	Thrust down

Table 7 Initial Hand Gesture Set

The first step for creating a Convolutional Neural Network is building and preprocessing the dataset. This input dataset set will be used to train our model. In Convolutional Neural Networks the main goal is to create an input dataset that has good coverage so the model will be able to achieve maximum prediction accuracy when faced with brand new input. We will use supervised learning in our model. Supervised learning is the idea where all training data is associated with a label identifying what the training data represents.

For this project, our dataset will consist of thousands of different hand gesture images. The set of hand gestures that our application will be able to recognize are shown in **Table 7**. In order to build our dataset, we plan on using our MacBook Pro webcam. Using this webcam, we can manually take thousands of pictures of different hand gestures, but this task will prove to be tedious and time consuming. To improve efficiency, given that the MacBook Pro webcam has the capability to record at 60 frames per second, it makes more sense to record a 17 second video of someone doing a specific hand gesture. With that video, we can process each frame individually for a total of 17 * 60 = 1020 images of a specific hand gesture. This approach is less time consuming then the manual approach described above. One main challenge we will be faced with will deal with processing each frame. Our Convolutional Neural Network needs to be able to universally recognize hand gestures no matter the users skin color or changes in users background environment. For example, our model shall be able to predict the correct hand gesture of someone of dark skin sitting outside and do the same when faced with a user of light skin sitting indoors. It will be inefficient and virtually impossible to train a model taking into account all skin color and environment variables. So how do we train our model in a way that it does not need to take such variables into account? Our plan to simply extract and threshold the hand gesture from each frame before sending it through our neural network for training and testing. In order to extract the hand gesture, the idea of image background subtraction will be used. In essence, you capture the background of your environment before you hand is the frame. This will create a "mask" that will remove or subtract everything but your hand. If background subtraction is done correctly, the resulting image will be just the hand gesture with a black background. This solves the problem of varying environments. To solve the issue of varying skin colors, binary thresholding is used on the already background subtracted image. By using a binary threshold, we can segment an image based on a certain pixel intensity. Given the background subtracted image we can threshold the image such that all the dark black pixels remain black and all every other pixel will be converted to white. This will create a silhouette of the hand gesture. The process of extracting and thresholding an image is shown in **Figure 12 through Figure 14**.

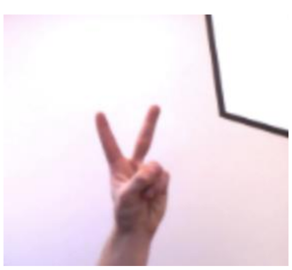


Figure 12 Original Image

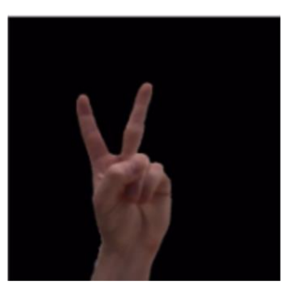


Figure 13 Background Subtraction

Figure 14 Binary Threshold

Our Neural Network should only be fed images that have been background subtracted and been applied a threshold to ensure skin and environment independency which we believe will maximize our model's total prediction accuracy.

A utility written in Python will be used to create our and organize our dataset. A basic flow diagram of this utility is shown below in **Figure 15**.

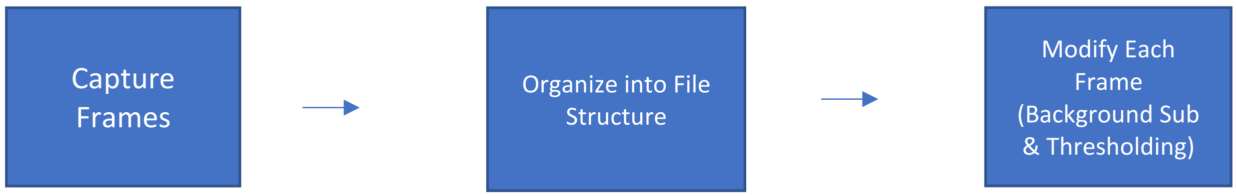


Figure 15 Utility Flowchart

OpenCV is an open source computer vision library that can be used to interface with the computer webcam. The webcam will record a certain number of frames and the utility will load all the captured frames into directory associated with the hand gesture being recorded. The directory name will serve as the label for each specific frame. For example, frames that show a closed fist will be put into directory named *thrust_upwards*, this name will also act as the label for each of the frames residing in that directory. The utility will then transverse through the created directory and modify each frame using background subtraction and thresholding to create frames that are both skin and background environment independent. The resulting image will be cropped so that only the hand gesture is shown and then resized to 50 x 50 pixels to ensure uniformity across all dataset images. This same process will be executed for each hand gesture. Pseudo-code for this utility is shown in **Figure 16**.

```
For each Gesture
    Enable Webcam
    Capture and save background image
    Capture at least 1000 frames
    Save in directory label

    For each captured frame in directory label
        Subtract with previously captured background image
        Threshold background subtracted frame
        Crop frame so just gesture is shown
        Resize 50 x 50
        Save new image
```
Figure 16 Dataset Creator Utility Pseudo-Code

4.3.4 Building the CNN Model

After our training dataset is created, the next step is to create our Convolutional Neural Network Model architecture. There are two approaches to creating the model architecture. One option is to create our own model architecture or the other option being using an already defined architecture. The main advantage to creating your own model is that you have full freedom to use the different building blocks, as discussed before, in any way. However, the main disadvantage is optimizing the architecture if needed. In building a Convolutional Neural Network Model, the developer doesn't know how good the model will perform without taking the time to train the model. In some cases, this could take days and if accuracy is low and optimization to different layer parameters is needed, it could take weeks before the model is producing the right accuracy. In essence, implementing our own Convolutional Neural Network Model from scratch will involve a good amount of trial and error and with the hardware being used to train and test our model, the process will not be time efficient. Based on this fact, we will use an already defined model. This approach is much better since these models have been created, tested, and optimized for accuracy by experts in the machine learning field. In addition, there are an immense amount of defined architectures to choose from. Of course, no matter what defined architecture we choose, there will be some tweaking of some parameters that will allow for compatibility between our dataset and model architecture itself. In general, the more layers a model has the more computation is needed however the model accuracy is generally higher in deeper networks. For our project, we want to stay away from using deep networks due to our hardware constraints and due to the fact that the model needs to produce prediction results in real time. Given these constraints we need to base our model after a predefined model that is shallow (less layers) and produces the best accuracy for our application.

There are many well defined Convolutional Neural Network model architectures that are optimized for different Machine learning applications. LeNet-5 and AlexNet are two Convolutional Neural Network architectures that pose to be a good fit to implement for our gesture recognition application. LeNet-5 is one of the earliest CNN model architectures and its main advantage being how shallow the network is. LeNet-5 consists of 7 total layers and was originally used to classify handwritten or machine printed digits. A representation of the Le-Net architecture is shown in **Figure 17.**

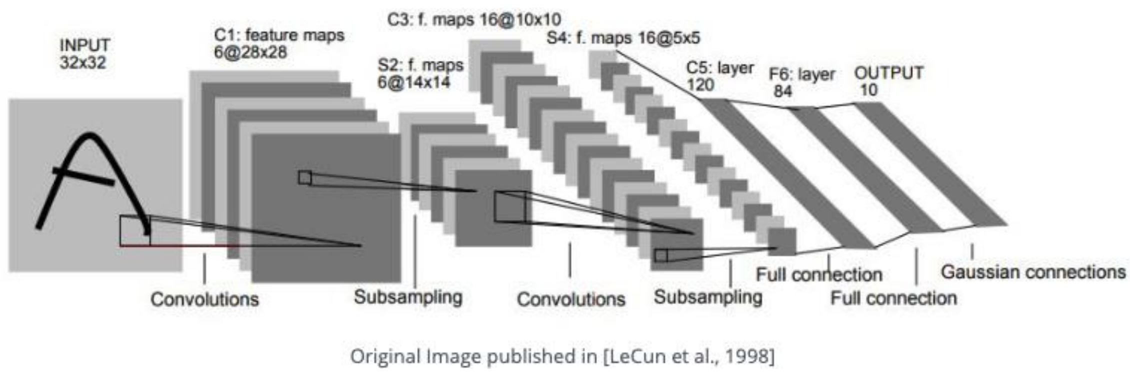


Figure 17 LeNet-5 Architecture
Permission to use approved

The input image to LeNet is a 32x32 greyscale image and the architecture consists of 3 convolutional layers (C1, C3, and C5), 2 subsampling layers (S2 and S4), and 1 fully connected (F6) followed by the output layer. What makes this architecture attractive for our application is that the model itself is shallow therefore training time will be comparatively shorter and predictions can occur in real time. In terms of error rate/accuracy, Le-Net-5 was able to achieve an error rate below 1% on certain datasets. AlexNet is another considered Convolutional Neural Network architecture for our gesture recognition application. AlexNet has a similar architecture to LeNet-5 but it is deeper (has more layers) than LeNet-5. AlexNet also outperforms LeNet-5 in terms of accuracy due to the fact that AlexNet is a deeper network architecture. **Figure 18** shows the architecture of AlexNet.

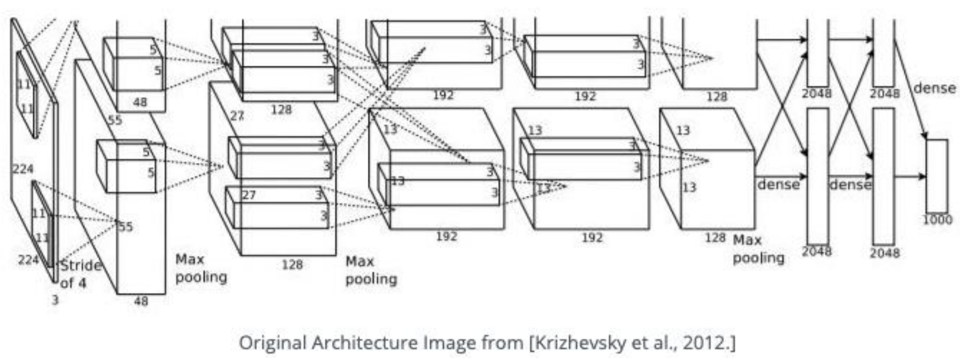


Figure 18 AlexNet Architecture
Permission to use approved

AlexNet consists of 4 convolutional layers, 3 subsampling layers, and 3 fully connected layers followed by an output layer. AlexNet is typically used for classification of high-resolution colored images and due to the fact that AlexNet is deeper than LeNet-5, it could cause slower prediction time given our hardware constraints.

Both LeNet-5 and AlexNet are good defined Convolutional Neural Network architectures that have been proven to produce accurate predictions. For our project, we plan on implementing the LeNet-5 architecture first to see what results we can achieve. LeNet-5 is a simple and shallow network that we believe can produce accurate results in real time. In addition, LeNet-5 was designed for a dataset consisting of greyscale and low-resolution images. Our dataset falls into this category since our input images will also be in greyscale and of size 50 x 50. AlexNet is a very capable architecture but given the characteristics of our dataset and our hardware constraints, we believe using AlexNet for our real time gesture recognition application could be overkill. However, if using the LeNet-5 architecture does not meet our accuracy requirements, we will be forced to use AlexNet or similar architecture as it is more robust and capable of achieving higher prediction accuracies.

To build/code the model we will use Keras, TensorFlow, and Python as mentioned in earlier sections. The plan is to create a single Python file that will contain code to preprocess our dataset, the model architecture in code form, and commands to initiate training as well as saving our model weights after training is complete. Coding the model architecture will be done completely using Keras since it provides the simplest and readable way to create Neural Network Models.

4.3.5 Training the Built Model

Once our Neural Network model architecture is defined, the we can begin training the Convolutional Neural Network. Our input dataset of hand gesture images will be used to train and test our model. Before the training begins, the input data set in split into two parts, train data and validation (or test) data. Typically, there is more train data than validation data, a 9 to 1 split. For example, if there are a total of 10000 input dataset images, 1000 of those images will be grouped into the validation data and the remaining 9000 will be grouped into the training data. The training data is used to help the model learn whereas the validation data is used to test the model’s accuracy at that point in the training process. For our project, the dataset will consist of about 1000 images of each hand gesture for a total of around 8000 images. The plan is to split the dataset, grouping 7000 images to be used for training the Neural Network and the remaining 1000 images will be used as validation data. An example of a Convolutional Neural Network being trained using Keras is show in **Figure 19**.

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 22s 363us/step - loss: 1.3991 - acc: 0.8830 - val_loss: 0.0882 - val_acc: 0.9738
Epoch 2/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0712 - acc: 0.9790 - val_loss: 0.0874 - val_acc: 0.9729
Epoch 3/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0484 - acc: 0.9854 - val_loss: 0.0898 - val_acc: 0.9757
<keras.callbacks.History at 0x7fc38442e240>
```

Figure 19 CNN in Training Phase

The example shown in **Figure 19** has an input dataset of 7000 samples, 6000 of these samples are used for the training dataset whereas 1000 samples are used for the validation or test dataset. This model is trained for 3 epochs. An epoch is essentially the number of times the model cycles through all the data. Within each epoch the same 6000 samples are used to train the model and the same 1000 samples are used to test the model's accuracy at that specific epoch. In general, the more epochs that are run, the more the model's accuracy will increase. However, there is an upper bound on the number of epochs that can be run until there is no more improvement in accuracy. There is no way of knowing what this upper bound is, so it is a general rule to set the number of training epochs to a high value, around 50 epochs. There is always an option to stop training if there are no noticeable or decreases in accuracy. At the end of each epoch the model evaluates its performance and performs backpropagation to update the weights, the specifics of backpropagation are mentioned in Section 4.2.3.

4.3.6 Testing the Neural Network

Testing of the Neural Network itself occurs at the end of each epoch. The model first trains itself using the training dataset and immediately after the model tests itself on the validation data. This process occurs during every epoch. After each epoch, metrics are calculated to show how well the model is responding to the training and testing. The loss metric is the output of the loss function which measures how well or how poorly the model behaves by finding the difference between the predicted value and the ground truth value of an image and is used to optimize the model. Accuracy measures how well the model performed by taking the number of correct prediction and dividing by the total number of predictions. The accuracy metric is not taking into account when optimizing the model as it is just a metric for us to reference in order to see if the model is training well. If a model is training well, we see a decrease in the loss and an increase in accuracy. **Figure 20** depicts the ideal trends of a model in the training phase if the loss metric and accuracy metric is plotted. It is shown that accuracy generally increases, and loss generally decreases as number of epochs increase. According to **Figure 19**, the loss and acc metrics are calculated from using the training dataset images to test the model whereas the val_loss and val_acc are calculated from using the validation dataset images to test the model. The metrics that we are most interested in are the val_loss and val_acc because they measure how good the model is performing to seeing new hand gesture images. For our project, we will make sure monitor these different metrics during the training because they are the only indication of how good the model is responding to training.

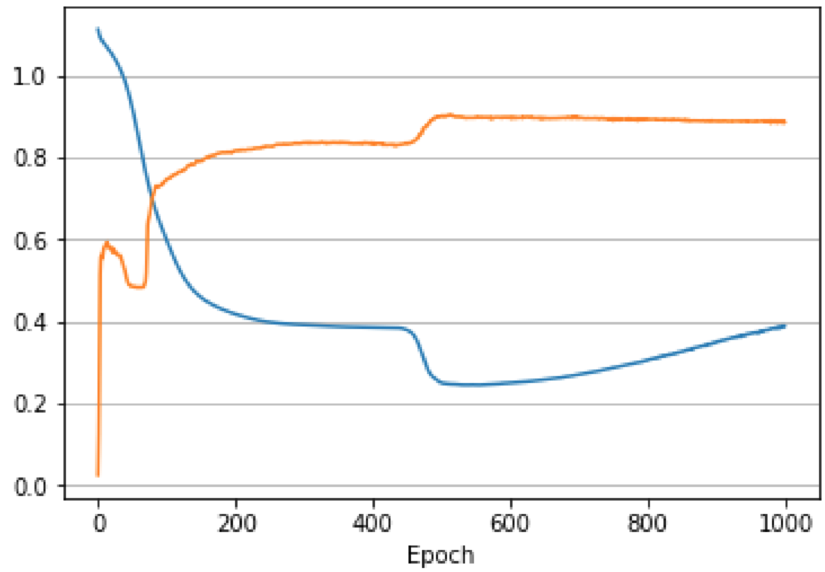


Figure 20 Accuracy (Orange) and Loss (Blue) vs Epoch
Permission to use requested

4.3.7 Real-Time Recognition

One major aspect of our project which also poses to be a main challenge achieving real time recognition of hand gestures so our drone can also be maneuvered in real time. Given that our model trains successfully and produces an acceptable accuracy it is imperative that we use our model in a way to achieve real time results.

Once our model is done training and the model's architecture, weights, and optimizer state is saved into a .h5 file, the plan is to create a python program that handles the real time recognition of hand gestures and the sending of messages over Bluetooth. **Figure 21** shows a flow chart of the proposed program

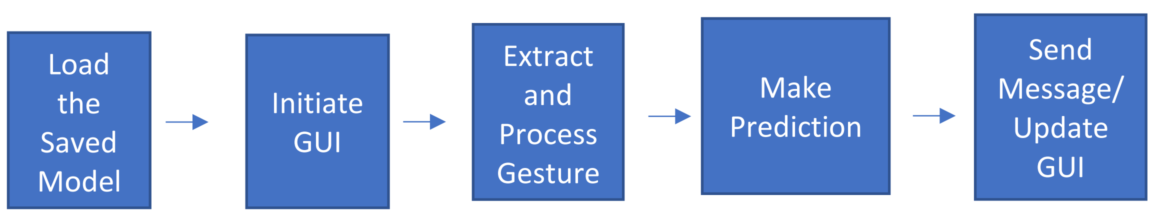


Figure 21 Recognition Program Flowchart

The first step is to load the previously saved model architecture, weights, and optimizer state. In Keras, the function *load_model(filepath)* can be used to load the model at the specified file path. The next step is to initiate and load all visual aspects of our graphical user interface or GUI. The specifics of the GUI will be covered in section 4.4. The third step is to extract and process the hand gestures shown to the webcam. Each frame is first background subtracted, applied a threshold, resized to 50 x 50 resolution, and finally converted to greyscale. After the frame containing the hand gesture is processed, it is passed through our loaded model in order to get a prediction of what the hand gesture is. In Keras, the *model.predict* function is used to get prediction results on the inputted image. This function returns an array of size equal to the number of classes. In our implementation, there is a class per hand gesture totaling to 8 gestures. The values in the array represent how close the model thinks the input image is to belonging in the specific class. The higher the value, the more the model thinks the input image belongs to the class. For example, in our project we have 8 gestures which corresponds to 8 different classes. When *model.predict* is called on a new input image it will produce an array of 8 values, [.21, .26, .56, .86, .95, .12, .03, .42]. The model predicts that the input image is closest to class 5 since that value is highest. The 5th class could represent a closed fist so therefore a closed fist is the final prediction. After the prediction is made the next step is update the GUI to reflect this. In addition, a corresponding Bluetooth message will be sent to the drone to specify what maneuver the drone must perform based on the recognized hand gesture. The Extract and Process Gesture, Make Prediction, and Send Message/Update GUI steps all should be done in real time.

4.3.8 Foreseeable Issues

One obvious issue that could potentially arise during the model training and testing is bad prediction accuracy. Bad prediction accuracy can be a caused by a handful of things such as characteristics of the dataset, model architecture, model parameters. If the dataset doesn't provide good converge over the different classes, there is a potential for some accuracy issues. For example, if our dataset contains 1000 images of a closed fist hand gesture but only 50 images of an open palm hand gesture, the model might run into accuracy issues when trying to classify open palm hand gestures because it was not given much data for the particular hand gesture to be trained on. In our project, by creating and using a dataset that is composed of an equal number of images per hand gesture, and having an abundant number of images per gesture, we will eliminate the possibility that bad prediction accuracy will be caused by the dataset. The model architecture itself can lead to bad prediction accuracy. Having too many layers or having a sparse number of layers in your model can affect how accurate the model is. If an insufficient number of layers are used then the network will have a hard time recognizing features that make each image different, thus leading to bad prediction accuracy. Having too many layers in a model or having a very "deep" model can result in longer training time but even worse, longer prediction time. Because we need the hand gesture predictions to be made in real time, we must avoid building an architecture with too many layers. Model parameters such as the number of filters in each convolutional layer or the learning rate value can either contribute to good model prediction accuracy or bad model prediction accuracy. The science behind choosing right parameters for your model is still a field in machine learning research as it is highly dependent on the application. As of right now, there is no general standard to use when defining parameters in the model and is essentially a trial and error process in order to achieve maximum accuracy. As stated before, we plan on using a predefined model, LeNet-5, that has been researched and optimized for performance. Of course, it is possible that we will need to tweak model parameters or completely change the model architecture if prediction accuracy is low. We are confident that by using the LeNet-5 architecture, our prediction accuracy will be high enough and there won't be a need to completely change the architecture of our model.

Overfitting is one of the most common and most researched problem with neural networks. The model is overfitting when the training dataset accuracy continues to increase or stay the same while the validation dataset accuracy declines. This means the model is memorizing rather than generalizing. Since the training data is the exact same for each epoch, the model is memorizing the training data and therefore when tested on the same training data the prediction accuracy will be high. When tested on new data, the model will not perform well and will show a decrease in validation accuracy due to the fact the model is not generalizing well enough. Detecting overfitting in a model is straightforward. By line plotting the training data accuracy and the validation data accuracy, it can easily be shown if a model is overfitting. **Figure 22** shows a line plot of a model that is overfitting.

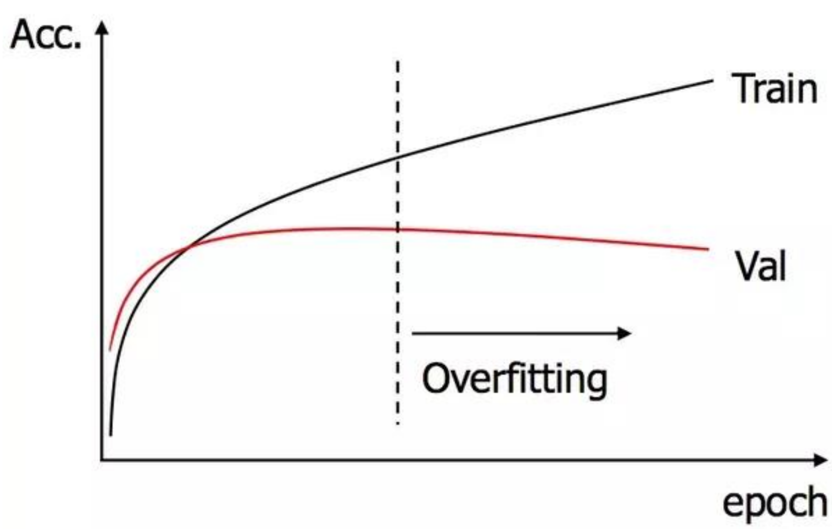


Figure 22 Overfitting Graph
Permission to use requested

For our project, we will implement different overfitting prevention techniques if necessary. One way to prevent overfitting is to implement early stopping. Early stopping essentially stops model training when the validation accuracy starts to decline rapidly. We can specify in the Keras code that we want to apply early stopping right before when the validation test accuracy starts to decline. Another popular technique that we will consider using in the event of overfitting is the use of dropout layers. Dropout refers to ignoring a random set of nodes during the model training phase. Each node is either kept or removed from the network with a certain probability. This helps prevent overfitting because in an overfitted network some nodes that make up the network are too dependent on other nodes in the same network which leads to memorizations. Strong dependences between nodes are denoted by higher weighted edges between the two nodes. By dropping some nodes from the network, the dependences between nodes are broken forcing nodes not to rely on each other by distributing weights evenly across all nodes. Dropout can easily be implemented in Keras by simply adding a Dropout Layer. In essence, if our model does overfit during training, we will use the early stopping technique first, as it's the easiest to implement. Using dropout layers will be our second option as choosing the right amount of dropout layers and the right probability that a node will be dropped will involve some trial and error.

4.3.9 Other Approaches to Gesture Recognition

Computer Vision is still a up and coming field in research therefore than isn't a lot of other approaches to gesture recognition. The one notable approach that differs from the Machine learning approach is the use of the python library known as OpenCV. OpenCV offers users an abundant number of functions that help with computer vision applications. OpenCV can be used to recognize simple hand gestures by essentially counting the number of finger tips is sees. This is a big limitation since there can only be six recognized hand gestures. In order to be able to count the number of finger tips in the image, a contour or outline of the hand must be found first. The next step is to find the edges of the found contour, this is effective in trying to find the fingertips of the hand. After the edges are found, the next step is to ignore all edges that are not fingertips. This is done by computing the angle between two edge points, if the angle is small enough, the edge points will be considered an edge point. Some advantages to this approach compared to the machine learning approach is there is no need to gather a dataset to train a model, this cuts back on development time as the OpenCV approach is more of a "plug and play" way of recognizing gestures. In addition, the speed of producing a prediction is generally quicker using OpenCV than using the Machine learning approach. Some disadvantages of using OpenCV to recognize hand gestures is the limitations on the different hand gestures that can be used, and the code complexity that goes into distinguishing between different hand gestures. The algorithms needed in order to achieve gesture recognition are more complex and the gestures are limited to the basic numeric gestures with OpenCV. The more advanced gestures used the more complex the algorithms get in order to distinguish between the set of hand gestures. The most prominent difference between OpenCV and Machine learning approaches is that with OpenCV, the developer is essentially teaching the computer how to distinguish between hand gestures by hard coding the characteristics to look for that differentiates each hang gesture. Therefore, the complexity the of code is dependent on how many gestures are used in the application. Machine learning, on the other hand, allows the computer to learn the different features of each hand gesture. The code complexity is not dependent of the set of hand gestures as the same model architecture is used to train any set of hand gestures. This approach is more robust, allowing the developer to add or remove hand gestures from the dataset with little code modifications.

4.4 Graphical User Interface

4.4.1 GUI Overview

The graphical user interface will act as what the user interacts with to communicate with the drone. The goal is to keep this interface as simple and user friendly as possible. **Figure 23** shows the initial layout of the GUI.

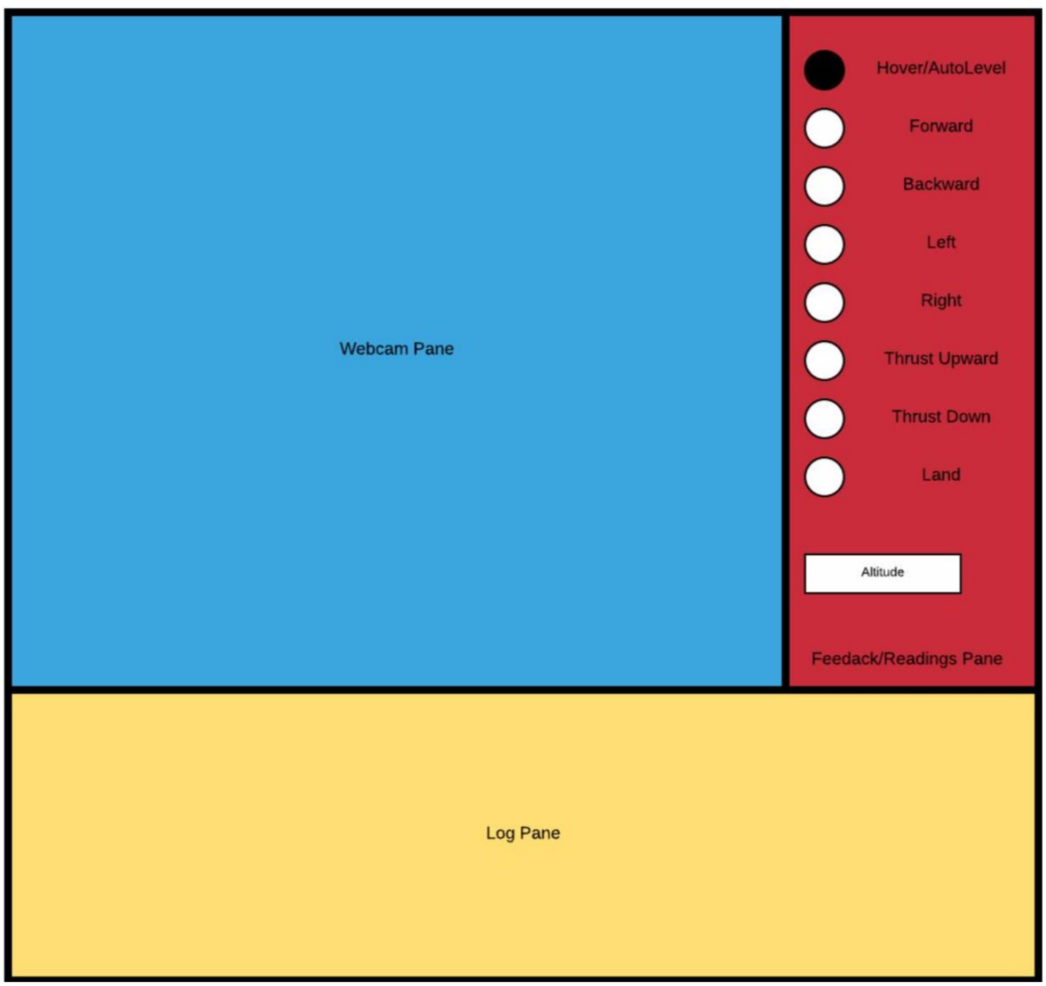


Figure 23 GUI Layout

The GUI will consist of 3 different sections or window panes the webcam pane, feedback/reading pane, and a log pane.

4.4.2 Webcam Window Pane

The webcam window will consist of a real time feed of the webcam and take up majority of the overall GUI space. This is where the user's gestures will be displayed and captured so the captured gesture can be processed by the Neural Network in the backend. The real time feed will be displayed with a green box overlay. **Figure 24** displays an example of what the webcam window will show. The green box overlay acts as a region of interest. This region is where the user will display their gesture and will ultimately be cropped out, processed, and sent through the Neural Network model for a real time gesture prediction.

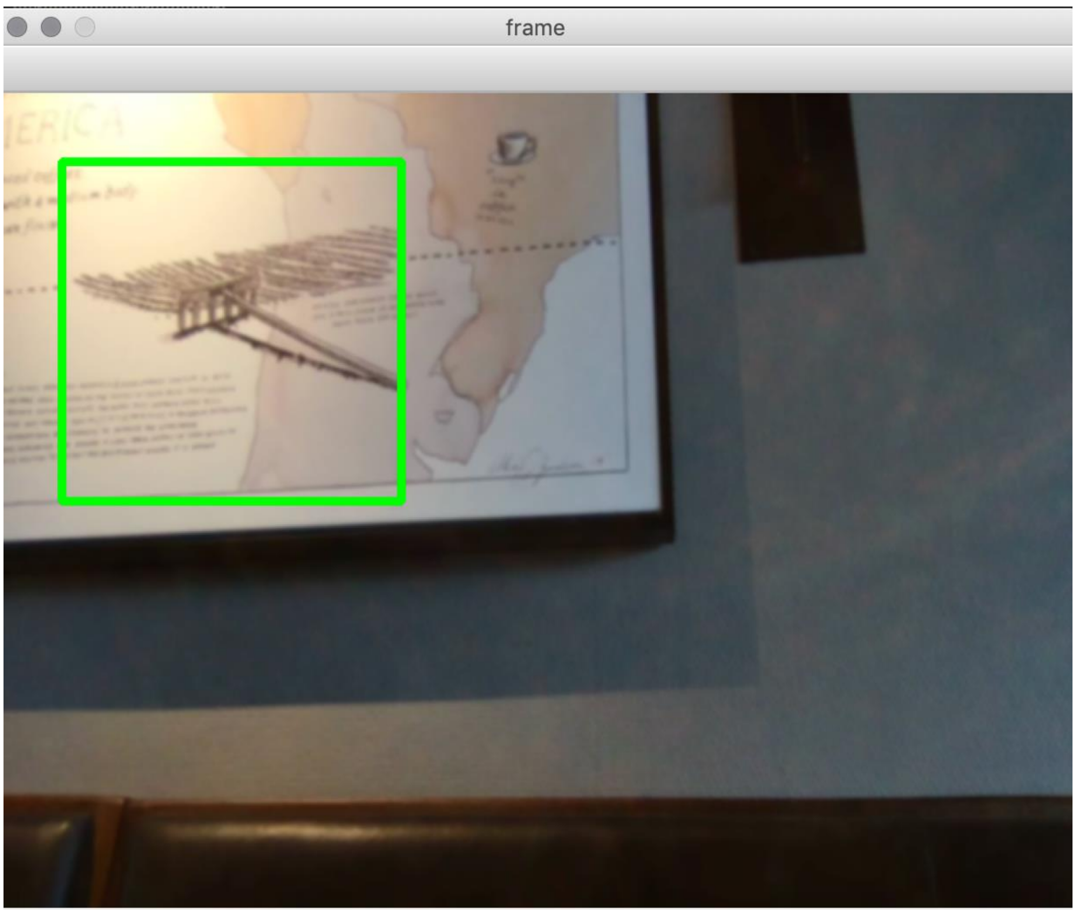


Figure 24 Webcam Window Real Time Feed

4.4.3 Feedback/Readings Window Pane

The feedback/reading window pane will give the user a visual representation of the prediction result after been passed through the Neural Network model. This window pane will consist of radio buttons, one for each gesture, with labels identifying the specific drone action Ideally, while the user gives the specific gesture that correlates to having the drone thrust upwards, the radio button labeled Thrust Upwards will be filled. Only one radio button can be filled at a time as only one drone action can be done at once. In the feedback/reading pane there will be one radio button for every drone action/gesture. In addition, there will be an altitude field that displays the drone's altitude in real time. An altitude sensor on the drone side will communicate the drone's altitude to the graphical user interface via Bluetooth.

4.4.4 Log Window Pane

The log window pane will serve as a textual representation of all actions being performed and be located at the bottom of the GUI. Essentially, every action taking place in the system should be recorded in the log window. From experience, if debugging is needed for your system, looking at log files is a good place to start. We decide to implement the idea of using logs to ease the debugging process and to have a good idea of commands being sent throughout the system. **Table 8** shows the initial set of log messages based on actions being performed in the system.

System Action	Example Log Message
Host system successfully connects to drone via Bluetooth	Bluetooth pairing successful
Users hand gesture is recognized	User displayed a Closed Fist
Mapping of hand gesture to drone action is sent to drone	Host sent Thrust Upward command to drone
Drone sends an acknowledgement back to host after receiving command	Drone received Thrust Upward command
Drone sends altitude data to the GUI	Received drone altitude data – 5 ft

Table 8 Log Message Format

In essence, all log messages will be written to a specific log file. This log file will be monitored, and its contents will be displayed, in real time, to the log window. The *tail -f* command will be used to achieve a real time log feed.

4.4.5 Building the GUI

For our project, we want to build a GUI as it will provide the user a simple and attractive way to interact with the system with all different components consolidated in one GUI window. Since our project will be coded in Python, we will be consistent and also use Python to create our GUI. Fortunately, there are many open source libraries that assist with creating a GUI using Python. Some main open source Python GUI frameworks include Tkinter and PyQt. Tkinter is native to Python and is a basic GUI package and provides common GUI elements that is used to build the interface. Some elements include buttons, entry fields, display areas, etc. These elements are also referred to as widgets.

Some main advantages of Tkinter include that it is part of Python and there is nothing extra to download. It also has a very simple syntax and provides an abundant number of widgets. Some main disadvantages of Tkinter is that the graphics look old and outdated and it can be difficult to debug. PyQt is a set of Python bindings for the popular Qt application framework. It is not native to Python and requires extra downloads. It is easier to design GUI's with PyQt and is typically used to design more advanced GUIs. In contrast, Tkinter is generally used for smaller, less advanced, GUI applications

We plan on using Tkinter to create our GUI as no extra installs are needed and due to the fact that our GUI itself will not be advanced and the look of the GUI is not an important aspect to us. Essentially, the purpose of the GUI is a create some organization of all the different aspects of the gesture recognition processes. When creating a GUI, the concept of event driving programming will be utilized. Event driven programming is a programming paradigm where the flow of the program is determined by events. Events can be mouse clicks, messages from other threads, key presses, etc. Usually, when creating a GUI there is a main loop that waits and listens for events to occur. When a specific event occurs, a callback function is triggered to appropriately respond to that event. In our GUI implementation, a specific event will occur every time a new hand gesture is predicted. There will be 8 events, one for each hand gesture. If a closed fist hand gesture is recognized, the callback function/event handler will send a specific message to the drone via Bluetooth, update the Feedback/Readings window pane to fill in the closed fist radio button, and write a message to the log file that will be shown in the Log window pane. Another foreseeable event that will occur is updating the altitude field. The drone will send its altitude via Bluetooth to the GUI. Upon receiving the altitude information from the drone, and event will be triggered to update the altitude field in the Feedback/Reading window pane to reflect the newly updated altitude of the drone.

One potential optimization in building and executing our GUI is to incorporate multithreading to our GUI application. Multithreading has the potential to improve computational performance by using different CPU cores in parallel. In our GUI implementation multithreading could be used perform tasks that do not depend on each other. For example, reading in altitude data from the drone and updating the altitude field in not dependent on recognizing hand gestures, sending the specific message via Bluetooth and updating the GUI fields. Therefore, a single thread can handle receiving and processing the altitude data and another single thread can be used to handle gesture recognition. If multithreading isn't used a potential blocking scenario can occur. For example, if the execution of the program is currently processing a frame for recognition and at the same time the drone sends altitude data to the GUI, the updating of the altitude field will be blocked since the CPU is currently executing instructions to process the frame. The updating of the altitude field will have to wait until the frame recognition is complete before continuing. Therefore, with multithreading, productivity and response time of certain GUI aspects will be increased.

4.5 Wireless Communication

Because a drone is controlled by RC, we needed to plan for some sort of wireless communication. The requirement of ours to control the drone from a remote location is imperative to a drone project, because there are several safety concerns involved with operating an unstable drone due to the speed and torque that the motors spin. To circumvent the safety concerns, we are going to operate the drone wirelessly, which will allow us a safer testing environment and a more usable product overall. Additionally, it is one of the main features of any drone on the market to be wireless, because the idea of a drone is to be able to fly independently. One of the core ideas for our project is to build a drone that is very user-friendly and easily manageable, which goes hand-in-hand with being wireless. This means that we had to perform thorough research on forms of wireless communication to carry our data to and from the drone and decide which particular medium is best for our implementation.

4.5.1 Possible Connection Mediums

When it comes to the wireless communication for the project, we researched each and every one of our options, because we wanted to ensure that we were using the most beneficial medium possible. Our possible options boiled down to Wi-Fi, Radio, Zigbee/Z-Wave, and Bluetooth. We explored each of the mediums of wireless communication, but we found that Wi-Fi and Bluetooth were the leading ones, so we weighed out the advantages and disadvantages for the two.

The industry leading form of wireless communication is undoubtedly Wi-Fi, as it is a household term known world-wide, and even is seen by some as hard to live without. This is because it is our way to connect to the Internet, and for it to be so widely used, Wi-Fi must be reliable and fast. For us to leverage the advantages of Wi-Fi, we would need either a common Wi-Fi network for both the drone and the master computer for quick communication across the same network. Alternatively, we could have the devices connect to the Internet, from different or same access points, and communicate via an API microservice to communicate with POST requests from the computer to be received by the drone. This implementation would allow us to control the drone from long distance remote locations. We could also have built the drone to be a Wi-Fi access point, which would have allowed us to just connect to it from the laptop computer and directly communicate to the drone so long as we stay in range of the Wi-Fi access point, we could control the drone that way. This would have been similar to the way that you connect to a Google Chromecast, in which the device contains a Wi-Fi access point and you connect to it to feed it information to set it up. We decided against Wi-Fi because of a few reasons. The implementation that would allow us to control the drone from a long distance did not interest us, due to the fact that we are operating a drone, and you would always want to at least see the drone to understand its surroundings to not bump into anything and be able to navigate properly. Another reason we decided to move away from Wi-Fi was that it would have been a much more complicated configuration and is much less cost effective. The Wi-Fi access point configurations mentioned above would all require at least one Wi-Fi access point and one Wi-Fi receiver for the communication to work properly. While Wi-Fi is great in being speedy and communicating large amounts of data in short periods of time, popularity of Bluetooth for wireless communication for projects similar to this one is much higher than Wi-Fi.

One of the primary advantages of Wi-Fi would have been the ability to connect a vast number of devices to the network, however for our particular implementation this would not have been beneficial. This is due to us not wanting to send the drone commands from multiple sources, which would cause the drone to behave unpredictably and could result in injury or damage to the surroundings because it does not know which signals to prioritize. As this device is meant to be a personal drone, we only plan on having one device connected to control it, as having multiple devices connected and sending signals will cause the drone to perform unpredictable behaviors. This requirement actually lends itself nicely to one of the primary limitations of Bluetooth.

4.5.2 Why Bluetooth

Bluetooth is also being used because of how popular it is in everyday life. Bluetooth has been around since the early 2000s and has continually been maintained and upgraded through the past 2 decades. This technology remains a worldwide wireless standard and it is evident why it is when one understands the power and ease-of-use of it. It is completely standardized and has been continually optimized to reduce interference, reduce cost, increase data throughput, and reduce power usage. This continual optimization provides us another reason to use this technology, because it is only evolving more in the future, we are protecting our product for the future as it can be upgraded to the newer Bluetooth version without much difficulty. This is opposed to using other forms of wireless communication such as infrared signals or satellite communication.

4.5.2.1 Complexity

In regard to compatibility and difficulty, we have also researched this topic. Arduinos are very popular for basic DIY projects, and so, the Software Development Environment they provide to program it is very intuitive and hundreds of thousands of projects have been done developing on them. Because of this, there are a plethora of resources in regard to establishing the Bluetooth connection, as well as communicating data via Bluetooth. There are plenty of samples of source code for various projects that will provide us a great start on the embedded code that we will need to implement on the Arduino. As the sole data that is being communicated between a laptop computer and the Arduino is the gesture and the altitude sensor, we should not have many issues in data loss. In essence, we will be sending a code from the laptop, after deciphering the correct gesture, with a dictionary for the code implemented on the Arduino board, to determine the action that the drone should perform. From the drone, all we will be sending is the value for the altitude that will be directly read from the sensor. Simply put, we will only need a single byte going each way in terms of immediate data transfer. The fact that we are deciphering the gesture on the laptop allows for the computation done for the drone to be focused on maintaining flight and performing the actions. Our plan is to continue sending the signal from the laptop to the Arduino to tell the drone what to do. For example, if you give the 'thumbs up' gesture, the determinant code for that action will be communicated over Bluetooth to the Arduino continually, until the gesture is changed or until there is no gesture. In either of those cases, we will then switch to sending the appropriate signal continually until the signal is either changed or no longer shown. We are anticipating that we may experience some data loss because Bluetooth is not 100% efficient and reliable in certain conditions, however, we do not expect to experience data loss for longer than 500ms, because we are continually sending the signals. By this I mean that we will be sending several signals, and so it will not be a large issue if one of those signals gets lost, because they are being sent many times per second.

4.5.2.2 Bluetooth Version

Bluetooth has various versions available, as the people maintaining and upgrading the technology have been making it faster, allowing for more range, and increasing the reliability of it. This means that it is continually updating and there are many different versions of it. We will be using Bluetooth 4.x, which is the release that has been out for nearly 10 years now. We chose this particular implementation of Bluetooth because it will be, without a doubt, the cheapest version for us to use. Bluetooth modules with Bluetooth 4.x are extremely widely available and cheap. This allows us to keep the cost of the project down, which will increase the accessibility of the final product. Using an older implementation of Bluetooth also allows for the most available support regarding troubleshooting issues we may have when building it out. Bluetooth 5.0 has only recently been making its way as a standard in the market, as the technology actually came out in late 2016 but companies generally take some time to actually add it in to all of their products. We also chose this version for its Low Energy feature, which allows for us to preserve the drone battery for actually operating the drone, which is one of the major pain points of drones. We decided against Bluetooth 5.0 as well due to its only new feature being Slot Availability Masking, which detects and prevents interference on neighboring frequency bands. This feature is not a particularly necessary thing for us because we are looking to stay very close to the drone when operating it, as it is not meant to travel so far.

4.5.3 Pairing Setup

In regard to our particular implementation of Bluetooth, we will be pairing the drone to the laptop computer that is reading the hand signals. This pairing is a very simple process that nearly everyone with a smartphone is familiar with. It involves putting the Arduino (with the Bluetooth module) into pairing mode and searching for available ('visible') devices from the laptop computer. This process need only be done a single time, because after the first connection, each device will have the other device's Bluetooth ID saved and stored. This will allow the devices to connect automatically going forward, so long as Bluetooth is enabled on both devices. This also is not limited to connecting to one device. If we decide to run our software on different machines, we need only to pair the devices once again per device. To clarify, the drone will only be connected to one device at a time and will only be receiving signals from one device at a time.

4.5.3.1 Trusted Devices and Security

The pairing system usually has a built-in security check, which allows for external devices that you do not want to connect to your device to be filtered out. The usual process is, upon the pairing request, a security passkey is requested. This allows for some sense of security with the data being exchanged, because if we had some external device sending signals to our drone, it could malfunction, and the damages could be costly and/or dangerous. This built-in security check lets us make sure that only the devices we want connecting to our drone will be able to send it signals. **Figure 25** shows what the pairing request looks like from the HC-05 module to an Android phone, however it is very similar to a computer.

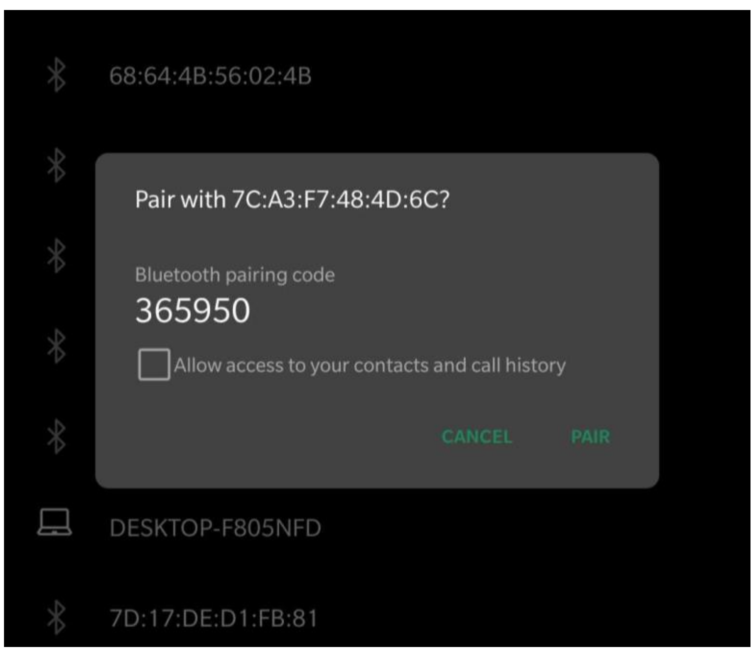


Figure 25 Bluetooth Pairing Request

4.5.4 Limitations

There are several limitations to Bluetooth, and so we will discuss what in particular we will be limited by in our project specifically. Our project in regard to wireless communication is quite simple, but there are a few hurdles to get over in regard to the data that is communicated and reliability.

4.5.4.1 Data Limitations

In specific, the amount of data being sent over Bluetooth is a limitation. Bluetooth does not have a very high data throughput, and so we cannot send large amounts of data quickly and efficiently. We considered this limitation of Bluetooth when scoping out the project, and so we decided to make the data communicated very small and simple. We plan on communicating only bytes of data, because the data will be sent very often, so we want to send small amounts of data for it to be communicated quickly and efficiently. Sending data over Bluetooth is via radio waves, and so it is difficult to send large data through the air whilst blocking out any interference. Additionally, we want the signals to be sent rapidly so that the drone is able to respond quickly to a new command. The signal needs to be read quickly and then communicated to the drone quickly to ensure that the drone is moving in a near real-time response. This requirement for us dictates that we needed to send small data but extremely fast.

As far as code complexity goes for the Bluetooth communication, we will be only communicating bytes of data, as the information coming from the drone will be numbers for the altitude sensor, and the information coming from the computer will just be a 4 digit code to determine the action that the drone will need to do. This allows for very small data and will ensure that the on-board memory will not be exceeded, and data will not be lost.

4.5.4.2 Range Limitations

Another limitation of Bluetooth is the range through which it can reliably communicate. This tends to vary from module to module, but generally, Bluetooth 4.0 is meant to have a limited range of roughly 300 feet, which is determined by the Bluetooth Special Interest Group (SIG). It will be vital for us to find a reliable Bluetooth module due to this limitation, as there is a possibility for an increasing number of interferences with the signal. A rapidly growing number of devices are communicating through radio waves in this time, and so interference-blocking is a key feature that signals need to have. The expected range is actually determined by the Power Class, which is a standard in Bluetooth that allows you to determine the difference between the capabilities of certain Bluetooth modules. Power Class 1 has a maximum range of 100 meters, while Power Class 2 has a maximum range of 10 meters, and Power Class 3 has a maximum range of 10 centimeters. Based on this, we will be absolutely unable to use a Bluetooth module that falls under Power Class 3, and so will be looking to find something in Power Class 1-2. However, the further the range, the more power that the Bluetooth module will use, which is one of our primary project constraints because drone flight time is very hard to maintain. Due to this, we will be prototyping a Bluetooth module in Power Class 2 first, to determine if the range is enough for us to maintain decent functionality of the drone. If that does not work, we will then fall back on trying something in Power Class 1 to be able to communicate the signals at a larger distance.

4.5.4.3 Interference Limitations

Another primary concern that will be on our minds is avoiding the heavy amount of interference that we will deal with when it comes to an indoor drone. Building an indoor drone helps us greatly with avoiding drone laws that would impede our product's use, but it also has its cons. The main con to building an indoor drone is that we have to block out an extreme amount of interference. This is due to the fact that in a room there are several wireless signals that are transferring very large amounts of data at all times. This will especially be the case when we are presenting our project for the Senior Design showcase, and so we will need to build a fool-proof way to send a strong signal that does not get interrupted or lost along the travel to and from the drone. This will without a doubt be a great challenge to us, and we will have to test in order to make sure that our drone is easily able to communicate with the laptop computer while we are feeding the signals to the camera.

4.5.4.4 Device Count Limitations

Bluetooth 4.1 allows for a maximum of seven devices connected at one time. This is a limitation that Bluetooth has due to the signal frequencies it has available to it. However, this limitation actually is not a constraint for us, because in our implementation of our gesture-operated drone, we plan on restricting data emissions to one device, meaning that only one device can control the drone. In doing so, we will build in the functionality to disallow any more than one device connected via Bluetooth. We are required to do this to make sure that the drone does not fly uncontrollably and is not confused as to the action that the drone should perform.

4.5.4.5 How Will We Accommodate

Because we want to keep the data being communicated to a minimum to ensure a faster delivery, we are going to be using a library built for altitude sensors that will convert the data that it receives via its sensors and converts it into one floating value. This will allow us to communicate the small amount of data rapidly and repeatedly so that we will be able to see near real-time updates of the altitude of the drone, with a relatively quick response time.

4.5.4.6 Dictionary Setup

Because we will be receiving data via an integer, we will be creating an on-board dictionary of sorts, so that the data we receive can automatically be converted to a maneuver/motion for the drone to perform. The dictionary will be defined based on **Table 9** provided below:

Dictionary Value	**Maneuver**
0000	Hover in place/auto level
0001	Thrust upwards
0002	Drone flies forwards
0003	Drone flies to the left
0004	Drone flies backwards
0005	Drone flies to the right
0006	Thrust Down
0007	Drone will land at current position

Table 9 Dictionary for Drone Commands

This table dictates what will happen based on each signal sent by the laptop computer after the gesture is converted to one of the 8 above numbers.

Below, in **Table 10**, naming each mode, describing the mode, and describing each use-case that our drone will be using for each.

Mode Name	Mode Description	Drone Usage
Active	Regular connection mode, device is actively communicating data to paired device	This will be the mode that the drone is in most often during the prototype stage, further into later implementations we will use this mode less to preserve battery
Sniff	Power-saving mode, checking for transmissions at a set interval, this mode is activated when the data is not actively being communicated/transferred	This is the ideal mode for the drone to be in for most of the time. As we are able to configure the interval for the check for transmissions, we will be continually altering this to make our drone response be a reasonable time while also saving as much energy as we can
Hold	Different power-saving mode, device sleeps for a set interval and returns to active mode after that, master can command the slave device to go into hold directly	This mode may be used when the drone has landed initially, and after a certain amount of time we can send the drone's Bluetooth module into "Park" mode
Park	Deep sleep power-saving mode, master can directly put slave device in Park Mode to deactivate the slave device until told by master to wake up	This mode will be used when the drone has been grounded for a longer interval, and so it is unlikely that the user is going to return to use the drone anytime soon, and will receive a signal from the master to wake back up when they need to launch the drone again

Table 10 Bluetooth Modes

4.5.4.7 Bluetooth Modules

There are several Bluetooth modules available to use in conjunction with the Arduino board. Majority of them are very simple to setup as they are made to use with the simple-to-use Arduino, but they all have varying libraries, configurations, and ranges to make each one different.

4.5.4.8 Module Limitations

There are several limitations that we have to consider when choosing our exact Bluetooth module.

For example, we need to take into account the cost of the module because we are completely funding this project ourselves and are trying to make our product as accessible as possible to introduce the value of our product.

Another limitation we should be considering is the amount of power drawn by the module roughly. While this depends heavily on how we are using the Bluetooth connection and how often we are communicating with it and what mode it stays in, particular modules do use different amounts of power because they can communicate either more reliably or are able to communicate over longer distances.

Range is the second most important limitation of this choosing, because we need to ensure that we are able to at least communicate to the drone at a reasonable distance, because a drone is not often controlled at a distance of under one foot. If we were limited to that kind of range, it would be hazardous to even operate the product due to the rapidly spinning propellers that could catch body parts and maybe even injure people nearby.

The most important factor in choosing a Bluetooth module is its ability to communicate signals without interference causing the signal to be lost on the receiving end. This could be very dangerous as well because a user could ask the drone to increase its altitude and interference could cause the signal to be altered and then the drone would receive, say, an incorrect command to speed up forward, which could be hazardous to people and objects nearby. This means that we absolutely must ensure that the drone operates based on the user's commands with 100% precision, hence why we will require that the Bluetooth module be able to communicate the signals with 100% precision.

4.6.4.9 Module Options

Primarily, for these types of projects the most common module to use is the HC-05 module. The reasons for this are that the module is able to communicate reliably within about 30ft and can work as either a master or a slave. This would mean that the module is able to create its own piconet as a master, and several external slave devices would be able to connect to this module. This is a functionality that we would not need, however with this particular module, since it is so popular for DIY projects, would have the most support for in regard to troubleshooting issues that we may have with it.

Another popular option for these projects is the HC-06 module. This one in particular is very similar to the HC-05 module, as it has the same range and brand, but simply without the functionality to operate as a master device. This one is very suitable for our use case because we only need the Arduino to be a slave device to the computer that is configured as a master device.

If we find that we require a greater distance for the connection, we can rely on switching to the BlueSMiRF Bluetooth module, because that module is able to communicate over 100 meters.

Another option for us is to use the BLE Link Bee Bluetooth module. The downside to this module is that it is relatively new, and so there will not be as much support and tutorials when we are trying to configure or troubleshoot errors with it. However, it has many benefits to it. Primarily the range that is twice that of the HC modules of 60 meters, along with a typically rare functionality for Bluetooth modules of having an integrated voltage regulator that supports both 5V and 3.3V MCUs. This functionality will be very beneficial to us as we begin building the prototype because we will likely end up going back and forth between different power configurations.

4.6.4.10 Reasons for Choosing

We have decided to go with the HC-06 module is very simple, we only need to connect it directly to the Arduino board (as shown in the picture below), configure the module, then continually read from the module through the Serial object to read the input. We can also send the altitude data through in the same loop as we are using the Bluetooth connection as a Full Duplex connection. The configuration we will be using will need to be tested when we build the prototype, however the default baud rate is 9600. We only need to tell it to save the connection info so that the connection is easier to setup next time. We decided on this module because it is extremely cheap and will allow us to very easily set up the Bluetooth connection in the beginning when prototyping, and because our drone will be operating indoors it may have a greater range. We will likely need more range than this module provides, but this module is so cheap that we can at least use it for testing because it is so easy to configure. This also allows us to hit the ground running faster in testing the flight control components, which will undoubtedly be the most difficult part of the project to figure out. Below you can see the configuration of the Bluetooth module connected to an Arduino board, as only 4 jumper cables would need to be used, this is a very simple configuration with few external parts needed.

4.5.5 Low Power Mode

Because power usage is such a prominent issue in all technological devices in this day and age, we are trying to create the most efficient product possible, to allow for the power-on time to be maximized. As this is especially important in drones that use high-power motors to keep the device suspended in the air or thrust upwards, we are trying at every step to preserve as much power as possible. Luckily, Bluetooth offers several low-energy modes that allow users to preserve power in their implementations of the technology. This was especially a focus on the Bluetooth 4.0 version because of the ever-growing requirement to save battery to increase efficiency of technology. Due to this constraint, we will be using Bluetooth's several modes to our advantage. These modes have primarily been made to preserve as much energy as possible.

4.6 Drone Hardware Design

4.6.1 Model Overview

Our drone design is a classic quadcopter with four arms, four brushless motors, and four dual blade propellers. Each motor is accompanied by its own ESC which are all powered by rechargeable lithium batteries. The ESCs are connected to the flight controller, which communicates to the user via Bluetooth. Each command is received by the Bluetooth module and interpreted by the microcontroller on our printed circuit board. **Figure 26** depicts an overview of the drone design.

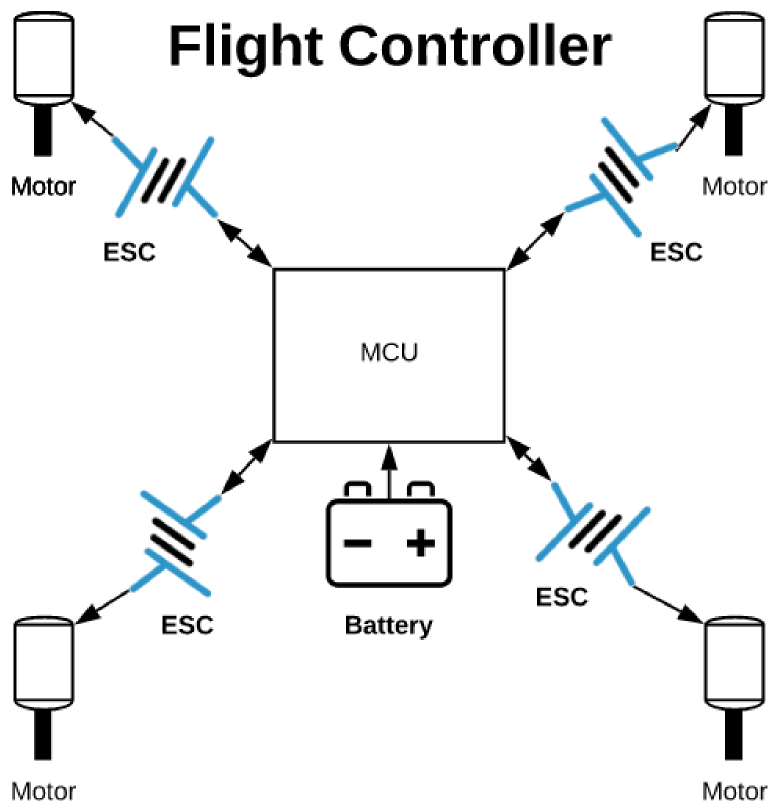


Figure 26 Drone Design

4.6.2 List of Materials

Below is a list of all the major components used to build the drone.

- Usmile 450 Quadcopter Drone Frame
- RC 1000KV Brushless Motor
- 30A Electronic Speed Controllers
- 3 Cell-Lithium Battery
- MPU 6050 - Accelerometer & Gyroscope
- Printed Circuit Board
- Bluetooth Module
- Altitude Sensor

4.6.3 Drone Frame

The drone frame is an essential aspect of the design. It is the core foundation. Even with sound electronics, a weak or misaligned drone frame can lead to future complications. There are a couple features we kept in mind when choosing our drone frame. The two most important aspects we had to decide were the size of the drone and what material the frame was made of. Our decision processes and decisions are mapped out in the sections below.

4.6.3.1 Dimensions

Our drone is designed to be flown indoors and that was an important consideration when choosing parts. Being in a confined space the smaller the design the better. A large bulky frame will limit the room we have to fly indoors. While a small frame might be easier to fly, we need enough room to mount all the components. The arms need to be large enough for the ESCs while the middle needs to be big enough to house the batteries and the PCBs. Without the need for a camera or a gimbal, commonly found on commercial drones, we do not need an extended landing gear to account for the added depth.

Most drones are measured in millimeters and are measured across horizontally/vertically. Each drone arm is the same length and the overall square design, means drones are measured with only a single value. We have decided that a 450 mm is small enough to fly indoors but will have enough space for all the materials. If we build our drone and have an excess of space, or a cumbersome design, we can always decrease the length of the frame.

Drones can have varying number of arms extending from the base. Some drones have as few as three and others have up to eight drone arms. When you increase the number of arms, the drone becomes more powerful and the thrust increases. Since our drone is strictly for indoor flight only, we are not overly concerned about making our drone very powerful. Four arms will give us plenty of thrust, allow for a more efficient design, and will make designing the flight controller much more feasible.

4.6.3.2 Frame Material

The material of the drone plays a large role in a drone's design and can have various effects on the flight and strength of the device. Frames that tend to be stronger are often heavier, while those that are lighter, are generally weaker. The best combination is a lighter drone that is sturdy enough to withstand minor crashes and stiff enough to have minimal bending. A lot of high-end drones use carbon fiber. Carbon fiber can be a great option, but it can be more expensive than other materials. It is both very strong, hard to damage, and also extremely light. The added strength might not be worth the increase in budget as our drone is not going to be flown in extreme conditions. On the other side of the spectrum would be a wooden drone. It is very cost efficient, but wood is extremely heavy and not strong enough to withstand crashes we are going to expect during the testing phases.

The best option for our design would be a fiber reinforced plastic drone. Strong enough to withstand the impact of minor crashes we might experience indoors and fairly light weight. There are plenty of frames on the market that are affordable and made of reinforced plastic. Another benefit of using plastic over the more expensive carbon fiber, is plastic does not have any communication issues. Carbon fiber is notorious for blocking radio waves and can cause complications when controlling the drone. When using carbon fiber, it is important to place electronics in a way where the signal will not be blocked by the frame. Plastic is a good lightweight and sturdy alternative to the other frame materials on the market.

4.6.3.3 Drone Assembly Process

When assembling the drone, it is important to ensure all the components are properly balanced and tightly secured. There are a few bad side effects on flight as a result of an unbalanced drone. A poorly aligned drone can lead to shaking. As a result, certain electrical components can be loosened and give off incorrect readings. Shaking can especially throw the gyroscope off, which is a key part of a stable flight. Shaking is not the only problem; bad alignment can cause undulations throughout the drone that could result in sporadic flight patterns. If you assemble the drone properly, this issue can be avoided. Eliminating these issues will make testing and debugging a lot easier.

There are two options we considered when discussing the frame of our drone, was whether we should custom the drone and 3D print it ourselves or build a pre made frame. Customizing the drone ourselves would give us a lot of design freedom but would require a lot of excess work. Many drone frames on the market are well made and we would not be gaining much of an advantage designing it ourselves. Assembling a premade drone would be a time saving options and also a reliable option. Putting the drone together ourselves we can be extra careful to ensure all components are aligned properly and all the components are tightly secured in place.

4.6.4 Motors

4.6.4.1 Overview of Motor Orientation

Of the four motors at the end of each drone arm, the direction of the spin is extremely important. A drone consists of three main types of movements. The first is the drone's ability to vertically change height, the second is rotation, and the last type of movement is a directional change. Below, **Figure 27**, is an image showing the orientation we are going to use for our design. The four motor positions are front left, front right, rear left and rear right. These can be represented with the following abbreviations, FL, FR, RL, and RR, respectively.

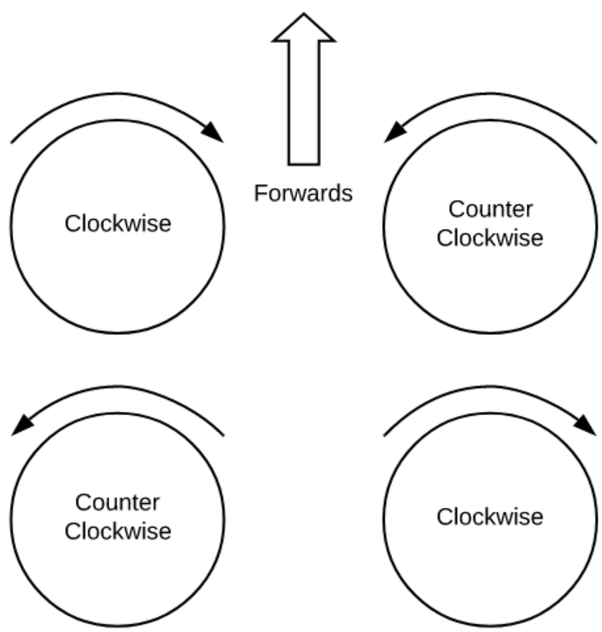


Figure 27 Motor Orientation

For the drone to change height, it uses the speed of the motors to control how much air is being pushed. The thrust of the motors and the drone's vertical flight path go hand-in-hand. When the motors are spinning there are two main forces present. The air being pushed down, and the counter force expressed in Newton's third law. When these two forces are equal, the drone remains level. With an increase in thrust, the force pushes downwards, and the drone starts to rise. The opposite happens when the thrust is decreased.

For the drone to not rotate the angular momentum needs to be zero. If the drone needs to be rotated, the angular momentum needs to be changed. This change can be done through a change in speed of one of the motors. If a singular motor has an increase in speed, the drone would rotate but it would also cause the drone to move vertically. As a result, the drones motors work in pairs to prevent this from happening. The FR and RL motors are pair while the other two are also a pair. While one set of opposing motors are decreased or increased the opposite occurs to the other pair to prevent a change in height. For a drone to rotate to the right, the front right and rear left motors will increase thrust while the, front left and rear right motors decrease their speed. If the front left and rear right motors did not decrease their speed, the drone would begin to rise. This decrease in speed counteracts this motion.

When talking about directional movements, it does not matter what way the drone moves, as the drone is symmetrical, it is the same explanation for all directions. Rotational uses diagonal pairs while, directional movement uses adjacent pairs. These pairs will change depending on which direction. If the pilot wanted to move the drone to the left, it would increase the speed of the front left and the rear left motors. If only the two motors increased speed, the drone would life up. To compensate for this, similarly to rotation, the other two motors decrease their speed. This will keep all other forces zeroed out and will just move the drone in the desired direction. All of balancing is going on simultaneously, and with the correct orientation everything will work in harmony. Prior to first flight it is crucial that we take the time to ensure all the motors are properly orientated.

4.6.4.2 Electronic Speed Controller

Motors rely on electronic speed controller to function properly. Essentially the electronic speed controller, abbreviated ESC, communicates between the motors and the flight controller. It governs the speed that the motors spring and can be programmed to perform as desired. Both brushless motors and brushed motors require different types of ESC. In our case, we are using a brushless motor, so we need the corresponding ESC. ESC for brushless motors are easy to distinguish as they have three motor wires, as opposed to the two motor wires on a brushed ESC. These wires will carry the signals from the flight controller to the motor. A stronger signal will spin the motors faster, this is all determined by the flight controller which receives the instructions from the user.

Inside of an ESC there are six MOSFET transistors that are all chained together. Certain combination of transistors when activated will correspond to a specific phase inside the motor. It is programmed to take the signal given from the flight controller and performs the correct gate changes to output the desired rotation. The higher the signal, the faster the cycle of phases will occur. It is important that we position our ESCs in a way where they will be exposed to open air to prevent them from overheating.

Choosing ESC can be a difficult task. There is a large variety of electronic speed controllers on the market all that have their pros and cons. After searching through a bunch of ESCs we narrowed our selection to a couple. Areas we put our focus on while searching was the compatibility, size, amperage rating, and the weight. Compatibility is important and the software needed to program the ESCs can play a large role in our decision making. If we are not comfortable with the corresponding software, we would be hesitant to choose the ESC.

The size of the ESC needs to be able to fit securely in the arm of the drone frame. The more powerful the ESC, the larger it is typically. ESCs can get big and the last thing we want are the ESCs to be protruding from the arms of the drone. We are looking for a good combination of power and size. Having a drone designed for stable indoor conditions, we can sacrifice the power for the overall size of the electronic speed controller. With a smaller size, the weight will also be decreased. Weight and size are directly related and the more lightweight our drone is, the more efficient our design will be.

Lastly, amperage rating is very important. These will be drawing the majority of our batteries power and minimizing this can elongate our battery life. ESCs are rated by the maximum number of amps allowed. A higher amperage ESC can run at a lower amperage, but once the value is exceeded, there is a risk of overheating or destroying the ESCs. For our design we limited our search for ESCs with at least an amp reading of twenty Amps. We are most likely not going to exceed thirty amps but during our search we did not limit ourselves to ESCs over 30 amps. Below, in **Table 11**, is a comparison of potential ESCs.

	Size	Weight	Price	Amp Rating	Compatibility
Emax BLHeli	3.1 x 2.0 x 3.1 inches	4 oz	$40	20A	✔
RC Electric Parts	2.1 x 1.0 x 0.5 inches	4.5 oz	$16	30A	✔
Crazepony	1.0 x 0.5 x 0.2 inches	1 oz	$45	35A	✔

Table 11 Comparing ESCs

All three options are good options that could work with our design. The first options by EMAX, is slightly bulkier than the other two. This added size did not come with another strong advantage. The price was on the more expensive side while the amp rating was the lowest. It was a good option but was not the best choice. Craze pony’s 35 A model was extremely lightweight and compact. This would be the most efficient choice; however, the cost is three times more than the other RC Electric Parts option. We decided that the added benefits of the Crazepony design did not justify that increased cost. At almost three times the cost, it would have been a lot more expensive to choose the Craze pony. Needing possibly more than four for backups or testing purposes, the Crazepony ESCs put us at risk of using much more of the allocated funds on ESCs. The RC Electric Part ESC is small enough with a high enough amp rating. The one downside is that they are the heaviest of the three designs. This added weight does not work in our favor; however, it is still fairly light, and the difference is rather negligible. **Figure 28** shows of the ESC designed by Electric Part that we will use in our drone design

Figure 28 RC Electronic Part ESC

4.6.4.3 Brushless Motors

Brushless motors are the best option for our drone because they have a longer life-span than brushed motors. With no internal friction, the motor will not deteriorate as quick as a brushed motor would. Brushless motors use magnetic power which waste less energy and is more reliable. Brushless motors can be divided into two categories, in runner and out runner motors. Performance wise they are both very similar however in our case we are using an out-runner motor. Outrunning motors are commonly used for drones. In running motors tend to be taller and narrower, however with the extension of our drone arms, space is not an issue. The wider outrunning motors are more suitable for drones. They are slightly less efficient but are capable of producing more torque.

4.6.4.4 Motor Power

Section 4.6.6 goes further in depth about our power design. Our ESCs will be connected to our lithium polymer battery power source. The motors we are using are 1000KV motors designed for RC quadcopters. 1000KV motor will produce more than enough thrust, however if in the future we run into issues, we always have the option of using more powerful motors. The motors and the ESCs draw a lot of power however our plan is to implement rechargeable batteries. With a rechargeable battery, it will save us from having to buy batteries every time the drone is dead. We are going to start with three 3.7V batteries and work our way from there.

4.6.4.5 Propellers

Propellers come in various shapes and sizes. The number of propeller blades per motor is a tradeoff between efficiency and thrust. Motors with more propellers have more thrust but are more inefficient. For our design we are choosing to use dual blade propellers because sacrificing efficiency for thrust is not worth it for our drone designed for indoor use only. It is important to position the propellers properly depending if the motor is spinning clockwise or counter clockwise. The image below shows the orientation of the propeller depending on the direction of spin.

Similar to the number of blades, the longer the propeller the more thrust it gives however it comes at the cost of efficiency. Bullnose propeller are shorter and have a more square cut off however we are using longer propellers that may draw more current however the added thrust will help our dual blade propellers.

4.6.4.6 Motor of Choice

Choosing motors has a large effect on the drone’s performance. The motor is one of the key elements that governs choices for many other decisions. The size of the motor determines the length of the propellers while the type of the motor must match the ESCs. Starting with weight, this is one of the more important aspects to determine. The motor must be chosen with the frame size in mind. The size of your motors should be relative to how big your drone frame is. With a lot of room to work with on our drone phase, we can get a fairly larger motor. Besides weight, the power of the drone and how efficient it is are also selling points. Some of the original motors that caught our eyes are listed in **Table 12**.

	Weight	Type	Price	Strator Size	KV
Hobbypower	1.5 oz	Brushless Outrunner	$40	2212	1000
LiTacc Model	2.0 oz	Brushless Outrunner	$48	2212	1200
Woafly	2.5 oz	Brushless Inrunner	$31	2212	920
abcGoodefg	1.44 oz	Brushless Outrunner	$45	2212	2200

Table 12 Motor Comparison

If we were to choose any of the following four motors, we would have a solid product, but out of the four we were able to narrow our selection down to one. All motors are within our desired rotor size range. The LiTacc Model was a very good choice, relatively lightweight, however it was the most expensive model out of the four, and the increased cost was not justified as the other models had similar or better specs. The Woafly was significantly cheaper than the other two options, however it was also the heaviest motor. Not only was it more weight, but it is also the only inrunner out of the four. Even though we preferred an outrunner, we did not rule out all inrunner motors. We did not feel that the heavier weight and brushless inrunner motor was worth the decrease in price. Both the Hobbypower and abcGoodefg were very similar but the increased voltage on the abcGoodefg was not what we were looking for.

In summary, we are looking for a prop size ranging from eight inches to ten inches. With the prop size you can determine the desired size of the motor's stator. We are shooting for something greater than 2200. Another important specification is that our individual motors do not exceed 2oz. With these in mind, our motor of choice is the A2212 1000KV by Hobbypower shown in **Figure 29**. With a stator size of 2212, we have the option of using our desire propeller size range. We can vary the size of the propellers and compare the efficiency; our concrete propellers size will be determined in our testing phase. This motor weighs roughly 1.5oz which is below are constraint of 2oz. With a diameter of just over an inch, we will have plenty of room to mount it on our frame. It is also fairly shallow with a height just under 2 inches. One of the most attractive aspects of this motor is the low cost.

Figure 29 A2212 1000KV Hoppypower RC Motor
Permission to use requested

4.6.5 Sensors

4.6.5.1 Overview of Drone Sensors

The drone has a total of three different sensors all serving their own each individual purpose. The three sensors are the gyroscope, accelerometer, and the altitude sensor. They are all connected to our flight controller and their data is used to balance the drone and move the drone to the user indicated position. There are a lot of varying types of sensors on the market and a big part of our research was going through all the options and figuring out which were the best. Some of the factors we considered were the cost of the sensor, the functionality of the sensor, the overall size, and the communication protocol. Our decision processes are mapped out below, along with the sensor we will be building our prototype with.

4.6.5.2 Gyroscope

Gyroscopes come in various different types. Space shuttles use laser gyros while something more common like your car uses a vibration gyroscope. The gyroscope used by our drone is also a vibration gyro and it is an essential part of our design. It is especially important for the PID control loops that will keep the drone’s flight stable. The MPU-6050 is a popular option and we are going to use it for our design. The MPU-6050 has more than just a gyroscope, section 4.6.6.2 dives further into greater detail regarding all the chips functionality. One of the main reasons it is widely used is because it has a very helpful auto leveling compatibility which will facilitate with the balancing process. The sensor vibrates in certain way when the device is rotating. The gyroscope feeds this information to the flight control and the appropriate action is taken. The MPU-6050 is pictured below in **Figure 30**.

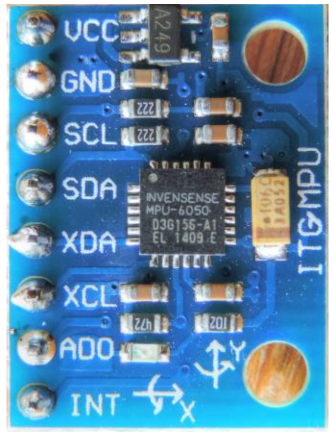


Figure 30 MPU 6050 Gyroscope and Accelerometer
Permission to use from open source

It is important to keep in mind when the gyroscope is being mounted, to ensure the gyroscope is aligned with the frame of the drone. Otherwise the drone will balance incorrectly. The sensor measures the how fast the drone is rotating. The rotation of the drone and angular velocity is explained in greater detail, in section 4.6.4.1. There are three coordinates the gyroscope measures, x, y, and z. Both x and y can be determined depending on which way the gyroscope is rotated. That being said, it must be sat perfectly flat so the vertical measurement, z measurement, is accurate.

When looking through motion sensors, it is important to look at the number of axes on the chip. Starting at three, a three-axis motion sensor only measures position and functions as an accelerometer. A six-axis motion sensor now adds the gyroscope rotational measurements. For the purpose of this project, we will be using a 6-axis motion sensor. A nine-axis motion sensor includes a magnetometer which for the purposes of this project we do not need.

4.6.5.3 Accelerometer

The accelerometer we are using shares the same chip, MPU-6050, as the gyroscope. Accelerometers are equipped on most electronic devices that are moving. They used on most aircrafts and measure both orientation and a devices acceleration. The accelerometer is constantly measuring all of the forces acting on the drone. Some are constant, like gravity, while others are user induced. Newton's second law defines acceleration as the net forces divided by the mass of the object. The accelerometer works in the same manner. There is a mass attached that measures the change in forces and determines the acceleration value. This is given to the flight controller and is used to move as desired and prevent the drone from tilting.

Doing research, we have some slight doubt regarding the effectiveness of the MPU-6050. The combined accelerometer and gyroscope are an ideal set up.

This being said, we run the risk of increased noise and disturbed signals. An alternative solution if we run into problems are buying separate accelerometers and gyroscopes. Two good reliable backups would be Invensense ICM-42605 or the MPU-9250.

The MPU-6050 board can connect to the Arduino board. These connections are very simple, as the Vcc input and Gnd pins are used for power, the SCL (I2C Clock) and SDA (I2C Data) pins are connected to the general GPIO pins on the Arduino board, and the Arduino SDE will allow us to work with the data passing through those pins directly.

4.6.5.4 Altitude Sensor

Measuring altitude can refer to many different things. Simply speaking it is a measurement of how high the drone is flying. However, it is all based off a reference point. This can either be absolute or relative. Different ways to measure altitude include measuring atmospheric pressure, height above sea level or height above the ground. For the use of controlling the drone’s height, we would need a drone that measures to the barometric pressure. We need to ensure the height doesn't exceed a certain value from the starting point. When the drone is launched, the current altitude needs to be stored. The drone can be given a limitation as to how much room it has to safely fly. This value can be changed and will be assigned prior to flight.

The altitude sensor our drone is using is the BMP180 as shown in **Figure 31**. This is a small yet powerful device that measures pressure and altitude. Additionally, it also measures the temperature of the surrounding air, but for the purpose of this project, that measurement is of no use. This altitude sensor is fairly inexpensive and supports I2C which will integrate nicely in our PCB. The layout of this sensors connection to our PCB is discussed in section 5.0.

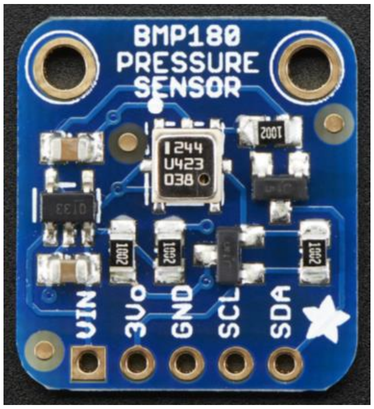


Figure 31 BMP180 Altimeter
Permission to use from open source

4.6.5.5 Indicators

The drone will be equipped with numerous LED indicators to help the user understand which direction the drone is flying and what state the drone is in. The drone will have an indicator that will turn on when the drone is powered on. This will be a small LED that will flash three times when the drone is powering on. Once on, the LED will remain illuminated and will turn off if the drone loses power or the drone is turned off. This is a good quick indicator to the user the state of the drone. When the drone is turned off, the LED will flash twice and promptly turn off, demonstrating the drone is now powered off. The LED will have various colors to indicate the battery life. **Figure 32** below shows the different colors and their corresponding battery value. Our drone and most drones on the market do not have very long-lasting batteries. It is important that the user knows the state of the battery and knows how long he has until the device loses power.

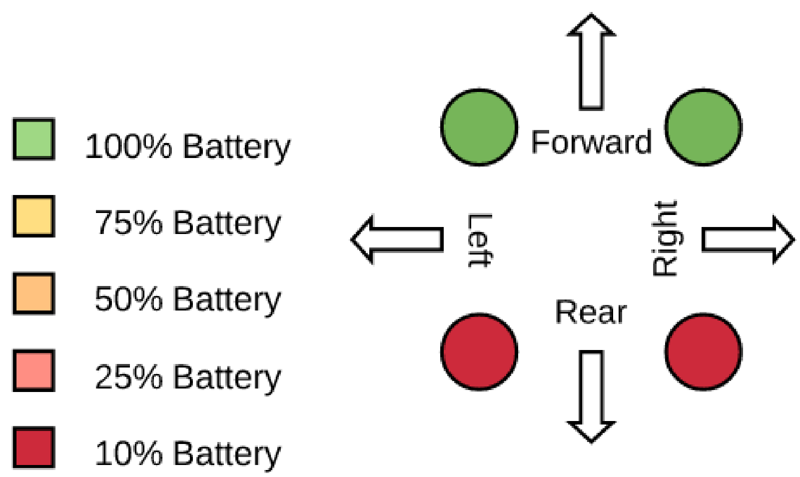


Figure 32 LED Indicators

Another indicator we are using, is a directional indicator. On the bottom of the drone will be four LEDs. The two front motors, FR and FL, will both have a green LED, while the other two will have a red LED. This will be on the ends of the arms, so the user can have easy visibility to the directional LEDs from every angle. These LEDs can be referenced when the drone is being directed. Moving the drone forward, is in reference to the two green LEDs. Regardless of where the drone is rotated, the forward movement will always be directed towards the two green LEDs. The image below shows how to read the LEDs and determine which direction needs to be shown to move the desired direction.

4.6.6 Power

4.6.6.1 Overview of Power

Power is one of the most important things to have a deep understanding of when it comes to this project, as we want to create a product that is as efficient as possible while retaining a low cost due to the project being locally funded and maintaining a good value to make it accessible. All of these things are heavily dependent on power when it comes to drones because in-air flight time is the greatest limitation when it comes to drones, given that most professional drones only have roughly a 30-minute flight time, which is industry leading. While that timing is for outdoor drones that reach a very high elevation, we are building an indoor drone, but the drone will have a lot of things pulling power from the power source. Overall, the power source will be providing voltage to the drone's propellers and the Arduino board (which in turn powers the altitude sensor, the Bluetooth module, the gyroscope/accelerometer, and the flight controllers). This shows that there are several things at work that require power here, and so our power source is going to need to be reliable and large. Unfortunately we are also limited to how large the battery can be because this will weigh the drone down, and our motors will need to use more power to keep the drone in air (if they can even lift it off the ground) and it will put more stress on the motors too. This would also cause the product to run much hotter, which can result in unsafe conditions for the other electrical components and eventually a malfunctioning of several parts of our drone.

4.6.6.2 Gyroscope

For the gyroscope, we will be using the MPU-6050. This particular component has a low power mode built into it, in which it will draw under .1 milliamps. This very low current draw is ideal for our implementation because we are trying to be as preservative as possible with our power. This .1 milliamp is being used up by .02 milliamps for the low power mode and roughly .06 milliamps for the voltage regulator that is built into the MPU-6050. Because the MPU-6050 is being powered by the Arduino, this will cause our Arduino to draw less power, which allows for more power to be used to keep the drone in the air, which will increase the runtime for our product.

4.6.6.3 Lithium Polymer Batteries

To power our drone, we will be using a Lithium Polymer (or LiPo) battery. We will be using this particular type of battery because they are much more efficient and powerful. The downside to using them is that they tend to run up the price of the drone, however because power is one of the biggest limitations when it comes to drones, we have decided as a group to invest well into a good battery so that our product can have a longer runtime. The type of battery is primarily determined by the motors and propellers that we are choosing, since heavier and more powerful ones would require a heavier and bigger battery. LiPo batteries are differentiated by the Capacity, C Rating, and the Cell Count. The Capacity of the battery is generally measured in milli-ampere hours (mah) which is generally the measurement that you often see to measure store-bought AA and AAA batteries. This measurement means that your device could draw that number of milliamps for one whole hour to drain the battery from 100% to 0%. The C Rating gives the maximum discharge current that can be drawn from the battery without damaging it. This just ends up being the Cell Count multiplied by the Capacity. The Cell Count is just that, the number of cells that the battery contains in total. Generally, you will find 3S or 4S, aka 3 cells or 4 cells, where each cell has a nominal voltage of 3.7V.

4.6.6.4 Our Choice

We will be experimenting with these LiPo batteries to build our product, but for now the battery we will begin testing with a few 3S batteries. We will essentially buy one of a lower capacity and then continue testing to figure out if we need a bigger one or can continue to use that one. Ideally, we would like at least roughly 5 minutes of flight time for our first prototype. In the future, we can work on upgrading the battery and bettering our product to be able to have a longer flight time, however our drone is meant to be very miniature, and so we are hoping that a 900mAh capacity 3S LiPo battery will suffice for our first prototype.

4.6.6.5 Rechargeable Battery

For our product, we intend to use a rechargeable battery. The reasoning for this is that we are building this product for hobby use, rather than competition use because the idea of this product is to use it indoors and build a new way to control a drone. Generally, we are not building this drone for a one-time use, and so it would be a great hassle to need to replace the battery every time you are done using the drone. Therefore, we will be using a rechargeable Lithium Polymer battery (specifications previously discussed). This will also help us in testing and keeping the cost down, because we will undoubtedly need to test the drone’s flight and runtime vigorously and having to buy new batteries every time it is discharged will drive the price of our drone up exponentially.

We will be purchasing a Lithium Polymer battery charger that is able to charge our battery safely along with the battery. The charger comes with connectors that plug into the battery and provides a screen interface so that you can view the battery percentage or the battery content. This feature will be especially useful in our power testing because we will be able to use the screen to tell us how much battery is left in the drone after we perform a controlled test with the drone and it still has power after landing. This will greatly help us in writing out usage instructions for the end-user.

4.6.6.6 Voltage Regulator

For each component receiving power, mainly the motors and the Arduino, we will need to regulate the voltage so that we are not over-supplying them and in turn damaging the components. While the power supplied to the motors will not need to be very strictly regulated, the power to the Arduino will need a very defined voltage regulation.

4.6.6.7 Battery Life

As previously mentioned, the battery life of the drone is one thing that we expect to be our greatest limitation for the drone. The reason for us thinking this is because drones typically do not have a very long battery life even when professionally made. Furthermore, we plan on using a very small battery. We hope to have at least an estimated 5 minutes of battery life from 100% battery to 0%, but we will need to see how close we are to that in the first prototype when we start testing out batteries and combine all of the electrical components together to see the amount of power that they are going to draw. Our plan is to use the low power modes in all of the sensors and the Bluetooth wireless communication to give us the most efficiency with our battery, and we also plan on testing how long the drone can hover, how long it can continually move forward, and other similar tests so that we have a very good idea of how the drone can operate and to understand how to instruct the drone user when a good time to bring the drone back closer to the user so that the drone can land before the battery is completely discharged and has a crash landing. Because of this, we will need to test the drone’s flying time with the motors connected but without the propellers, allowing us to determine the kind of flight time before actually letting the drone fly by itself, which would help us reduce damage costs.

4.6.6.8 Altitude Sensor

Measuring altitude can refer to many different things. Simply speaking it is a measurement of how high the drone is flying. However, it is all based off a reference point. This can either be absolute or relative. Different ways to measure altitude include measuring atmospheric pressure, height above sea level or height above the ground. For the use of controlling the drone’s height, we would need a drone that measures to the barometric pressure. We need to ensure the height doesn't exceed a certain value from the starting point. When the drone is launched, the current altitude needs to be stored. The drone can be given a limitation as to how much room it has to safely fly. This value can be changed and will be assigned prior to flight.

The altitude sensor our drone is using is the BMP180. This is a small yet powerful device that measures pressure and altitude. Additionally, it also measures the temperature of the surrounding air, but for the purpose of this project, that measurement is of no use. This altitude sensor is fairly inexpensive and supports I2C which will integrate nicely in our PCB. The layout of this sensors connection to our PCB is discussed in section 5.0.

The MPU-6050 board can connect directly to the Arduino board. These connections are very simple, as the Vcc input and Gnd pins are used for power, the SCL (I2C Clock) and SDA (I2C Data) pins are connected to the general GPIO pins on the Arduino board, and the Arduino SDE will allow us to work with the data passing through those pins directly.

4.6.6.9 BMP180 Code

The exact code uses an altitude function that takes in two parameters of the atmospheric pressure of your location and then the atmospheric pressure that the sensor is reading to determine the altitude based on the difference. This simple and easy to use library allows for our code to be complete, readable, and concise. The code will be running on a while loop that will continuously serve the pressure data converted to an altitude number in meters for us to provide to the laptop to show in the GUI.

4.7 Drone Software Design

4.7.1 Flight Controls

4.7.1.1 Overview of Flight Controller

Flight controllers control the speed of all the motors, dissect commands from the user and balance the drone. They are vital to the drone and can vary in functionality. For our design we have the option of programming our own flight controls or using a pre-programmed flight controller. Flight controllers use the onboard sensors and constantly feedback information for correction purposes. This is how the drone remains level. This PID tuning process allows us to customize how our drone reacts to certain movements and gives us a lot of freedom when designing our drones flight controls.

4.7.1.1 Dedicated Flight Controller

Drones have become widely popular over the last few years and the market is saturated with various kinds of drones. With all these drones, there are a ton of preprogrammed flight controllers to choose from. If we were to use a dedicated flight controller, it would save us the trouble of having to balance the quadcopter ourselves. If we did use a dedicated flight controller, the integration process would be more difficult. Communicating our controls to the already preprogrammed device, would limit our freedom and could also lead to future complications. This is not our first choice. We understand that complications down the road might lead us towards a dedicated flight controller.

The dedicated flight controller is just another microcontroller that is pre-programmed to have a stable drone. We thought it was important to have a couple options selected in case we needed to use a dedicated flight controller. Light a lot of the other components in this project, there are a lot of different kinds and it took a lot of research to find the best fit for our project. One component that will help is having a floating-point unit, abbreviated FPU. This is a component to speed up the computation of floating-point numbers. With an FPU, the mathematics will be calculated at a faster rate and will alleviate stress on the MCU and allow for quicker corrections.

Flight controllers are measured by their speed on a scale ranging from F1 to H7. These values determine a lot of components regarding the drone, but the higher the value, the more functionality and the better the processor. For our case, we do not need more advanced than a F3 processor. These processors are powerful enough and come with all the necessary components, including the FPU. The Frsky Rx & OSD V2 is an F3 flight controller that would be a good option if we decide to go the dedicated flight controller route.

4.7.1.2 Combined Flight Controller

Instead of purchasing a preprogrammed flight controller, using the Arduino platform we can develop our own. Having full control will allow us to expand our capabilities to the fullest potential. The commands will be received and directly converted into the desired reaction. There will be no integration process with an external flight controller. This will make our design simpler and limit the number of components controlling the drone. That being said, combining our flight controller and all the other devices under one MCU, might be a lot for the microcontroller to handle. It is important that device we pick is powerful enough to handle everything. It is also important that we have a backup plan in case, we cannot make it work. Our microcontroller decision is further explored in section 4.7.1.4.

4.7.1.3 Flight Control Schematic

If we decide to go with a dedicated flight controller there will be a change reflected in our block diagram. This will the separation of the MCU and the flight controller. The flight controller will be connected to the ESC and given instructions from the MCU. **Figure 33** shows and updated block diagram.

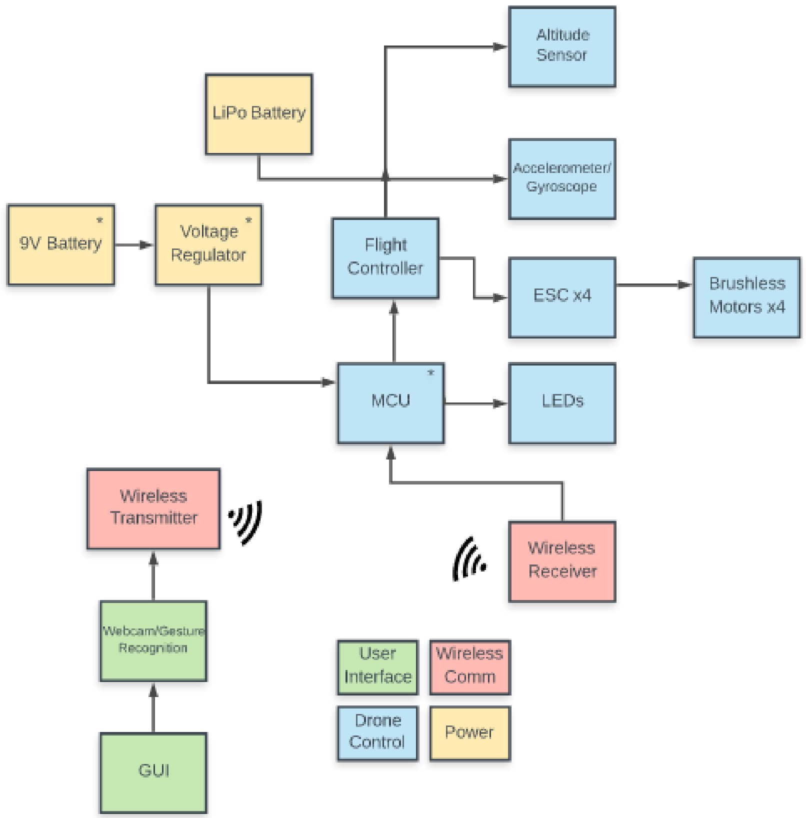


Figure 33 Dedicated Flight Controller Schematic

4.7.1.4 Microcontroller

The microcontroller is the brain of our drone and connects the ESC to the user giving them control of the device. The microcontroller we are planning to use the ATmega328p. The ATmega328p has a clock rate of 16 MHz, 2KB RAM, and 32KB of storage. We do not need much storage space as the code we are using is concise. We have predicted the processing power will be fine to work as a flight controller and read in commands from the user.

If the processor turns out to not be powerful enough to fly efficiently, the AT91SAM3X8E is a safe backup. It has significantly more space, higher clock rate and a lot more ram. The larger microcontroller will most likely not be needed. However, another option is to use the ATmega328p to feed information to a dedicated flight controller. The ATmega328p is quite good at relaying information. We are trying to avoid a dedicated flight controller but, in the scenario, where we need more power, this could be a feasible option. In **Table 13** below we list the three options we are interested in. The ATSAMD21G18 is a happy medium between the ATmega328p we plan to use and the backup AT91SAM3X8E. If the ATmega328p is slightly overworked, upgrading to the ATSAMD21G18 would be a smarter move than the bigger change to the

AT91SAM3X8E.

	Clock Rate	**RAM**	**Flash**	**Price**
ATmega328p	16 MHz	2 KB	32 KB	$10
AT91SAM3X8E	84 MHz	96 KB	512 KB	$20
ATSAMD21G18	48 MHz	32 KB	256 KB	$15

Table 13 MCU Comparison

4.7.1.5 ESC Calibration

Calibration of the ESC is extremely important. Subtle differences have large negative effects on the drone's stability. All the motors need to be in unison and spinning at the same speeds. The calibration process is crucial but it's fairly straightforward. No external program is used to calibrate the ESCs. All the calibration is programmed through the Arduino platform. A certain value needs to be set to determine what no throttle is and what maximum throttle is. This will vary depending on the motor. In our case, when the ESC gives a signal of 500 microseconds, the throttles are not spinning, and maximum speed when the ESCs send a signal of 1500 microseconds.

4.7.1.6 Balancing the Propellers

Once the propellers are arranged properly, they must be balanced. Just like most other components, symmetry and balance is a must. Using the accelerometer discussed in section 4.6.5.3 and the Arduino platform, the number of vibrations can be measured. Each motor can be isolated and checked to ensure not much vibration is being produced. Too much vibration will affect the flight of the drone and will be extremely difficult to control. With constant starting and stopping and adding small increments of weight to the appropriate side of the propeller they can be balanced. For the smoothest possible flight, the level of vibration should be as minimal as possible and equal across the four motors. If all four motors have little vibration the sensor can run without being disturbed. Taking our time on balancing the propellers will benefit us greatly in the long run with an extremely steady flight.

4.7.1.7 Explanation of Flight Control Code

Arduino programming language works very well with servo motors. It makes it very simple to program the ESCs as necessary. After including the servo package, you have the ability to use very helpful built in functions to control the motors. This will especially be helpful when controlling the PID loops. The actual

PID tuning process is discussed more in section 4.7.2. However, the following is a general summary of what the Arduino program will consist of. Initially the servo libraries are imported, opening the door to plenty helpful functions. The next step is to define values that will be used throughout the program globally so they can be referenced from across the board. It is important to look at the data sheet and observe how the information from the different sensors are given and how we can convert that to helpful information that can be used to create calculations. This information is discussed in more detail in the individual sensor section 4.6.5.

Now that all the preliminary information is taken care of, the next step is to define each of the motors and assign them the necessary signal to be in the off position. The program will run in an endless loop that will constantly be looking for a direction. The values received via Bluetooth we correspond to a signal and a conditional statement will match the value with a movement. Each digit will correspond to a different hand movement. When no action is being received, it will remain stationary. At this point the PID loops will constantly be giving feedback correcting the drone's movement and ensuring it remains horizontal. While the drone is hovering, the device is constantly getting feedback from all the sensors, this is the data being used to calculate the PID value. The program will also monitor the battery level of the drone. The different levels will correspond to different LED colors specified in section 4.6.5.5. All the other LEDS will also be controlled by the flight program. These include the power LED and the directional LED. The directional LEDs will remain the same color while the other LEDs will be constantly changing depending on the state of the drone.

4.7.2 PID Tuning

4.7.2.1 Introduction to PID Tuning

PID is an acronym for Proportional Integral and Derivative. In a closed loop system these values can be used to control the flight and allow the drone to make corrections as quickly as possible. This control system is constantly getting feedback and correcting errors. Changing the values of P, I and D will change how quickly and how the drone fixes these errors. Setting your own PID values can give us a lot of freedom to have the drone react best to our motions. Ideally the drone should not oscillate and move right back to an auto leveled position once the drone has finished its action.

Starting with the P value, it monitors the current error. A drone without any PID tuning would not correct itself. Including the P value will cause the drone to start oscillating. At this point it reads the error that the drone is too far to one side and tries to compensate. The higher the value, the more it tries to correct. This correction alone will not be enough. It will try to correct and over compensate causing a continuous back and forth action.

The I value monitors the past corrections and applies it under the situation where external forces are applied to the drone. Initially the I value is not a necessity. The P and D values are the first priority. The D value looks at potential future errors and correct accordingly. The combination of the P and D values are what create the quick reactive correction that is common across commercial drones. If the D value is increased it will work harder to stop the over corrections caused by a higher P value.

4.7.2.2 PID Schematic

Below, in **Figure 34**, a representation of the basic PID model. It shows how the output is fed back into the controller and altered by the PID values, affecting the output.

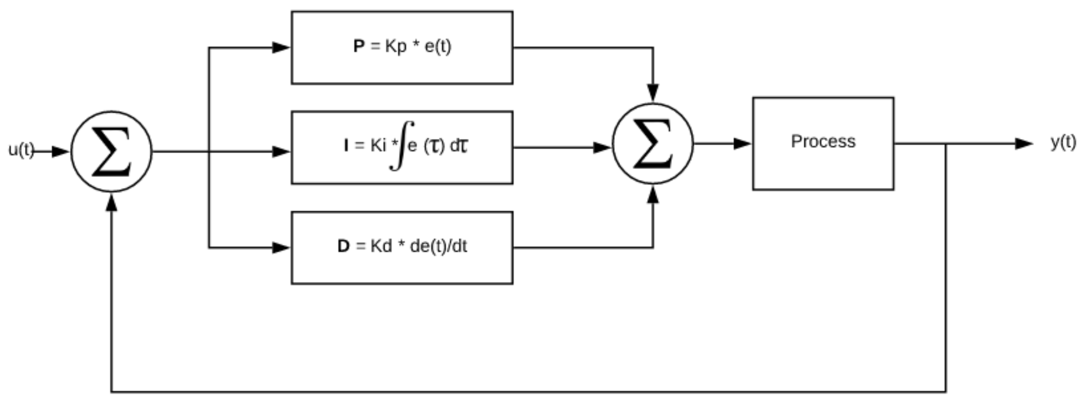


Figure 34 PID Model

4.7.2.3 Using Multiwii to Balance the Drone

There are numerous programs that facilitate with PID tuning including control station, MathWorks and MultiWii. After looking through various different options and possibilities we decided the best third-party tool to help balance the drone would be MultiWii. MultiWii is a tool designed specifically for RC drones and has a wide variety of helpful capabilities. It gets its name as it was originally based upon a component of Nintendo's game console Wii, which heavily used motion tracking abilities. It does a good job graphing the PID process and these visuals will help give us insight on how we can improve our design. With the useful Horizontal Situation Indicator (HSI) and all the angular measurements calculated, getting the Kp, Ki, and Kd values through MultiWii will be made much simpler. This is explained further in section 4.7.2.4. MultiWii integrates extremely easily with the Arduino software and the two will make balancing the drone much easier.

4.7.2.4 Process for tuning PID Loops

In order to test the motors, we have a structure designed to hold the drone in place. From this stationary position the drone will be easy to see where the corrections need to be made. Either this can be used, or the drone can be held and as the motors are increased, it can be moved to feel for the drone to be corrected. This process works but having a consistent stationary mount would be the most effective. Starting out we can isolate the test to one axis. Starting with two motors we can find the best working PID values and apply that to the other axes. Everything thing done on the first axis, can be applied to the second axis. After setting the untested axis, check to see if the values work and adjust the value accordingly. It will most likely work at first assuming the two axes are close to identical, but the values might need some subtle adjustments.

There is not a combination of PID value that is universally correct. There are guidelines to help guide in the right direction. Every motor is different, and every drone will have its own unique inconsistently. Separate motors draw varying amounts of power, and the stronger motor will cause the drone to lift towards the more powerful motor. This is corrected with the PID tuning. The best PID tuning process is starting with the P value. Increase the P value till a steady oscillation is obtained. This oscillation should be relatively quick. It will bounce back and forth but will take a very long time to get stable. Once the drone is oscillating from the overcorrections, the D value can be introduced. The D value will monitor the time a PID loop takes and relate that to the current angle. Now the drone should correct faster. With that information it will be able to prevent the drone from over correcting drastically and will limit the time of complete correction. The next process can be tedious but through trying different P values and corresponding D values, the high function set of values can be found to balance the drone as quick as possible. When the D value is too low, it will almost be negligible, and when it is too high the system will act unpredictably. Having an understanding of this can help you determine the next move to find the correct P and D values. Including the I value can also tighten up the drone's corrective process. The drone could balance itself without the I value although the I value makes subtle important changes. Once the drone is balanced close enough to zero the P and D values are no longer of use. Here is where the I value comes into play and can make those subtle changes that ensure the drone is perfectly horizontal.

It is important that the drone's range doesn't exceed a real angle of -45 degrees or +45 degrees, and also that the drone remains perfectly horizontal when no movement is occurring. If the drone were to exceed these ±45-degree angles, the drone would flip over. Once the drone flips over, it will not be able to hold itself up and crash. If the drone does not remain at a real angle of 0, it will float around and won't remain still which will make controlling the drone difficult. The importance of PID tuning cannot be stressed enough and we know this will be a major part of our project. This will take up a large amount of time and needs to be started as early as possible to get a working drone. Once balanced, we will have more time to focus on all the other aspects and improve our design to the best of our ability.

4.7.2.5 Explanation of the PID Code

The program needs to read in the values from the motion sensor and use that information to actively balance the drone. The value received can be broken into two categories, the accelerometer readings and the gyroscope. The accelerometer has a 16-bit value that can be converted to a more digestible unit of the pull of gravity. Gravity is roughly 32 ft/ s^2, and a register value of 16384 is equal to 1g. When the gravity is equal to 1g, the drone is level and a change in value most likely means the drone is changing direction. Using the pull of gravity, you can calculate the angle of the drone using basic trigonometric functions. These angles are important and can be used with the gyroscope to monitor the position and angular changes in the drone. The information can be extremely sporadic, and it is important to use filters to clean up the data received from the motion sensor. Through a combination of low pass and high pass filter, we can take that information and eliminate most noise and random errors that might occur.

Using the Arduino environment controlling our flight controls, we can use that data to help with the PID tuning. The three main PID constants are represented by Kp, Ki, and Kd. Kp corresponds to the P value, Ki corresponds to the I value, and Kd corresponds to the D value. It is important to track the error, how far off the drone is from stable and use the PID values to correct that. The desired angles are simple, either 0 for horizontal flight or 45 degrees in the desired direction. The PID controllers' job is to compare this value and correct accordingly. It is important to define which side is positive and which side is negative. Directionally, the forwards and right position are positive on the y and x axis, respectively. The backwards and left position are negative on the y and x axis, respectively. Each k value is individually calculated and summed together to create the singular PID value. This one PID value can then be used to monitor and correct the motor speeds. The error is constantly fed back into the closed loop system, the difference is taken for the desired value and the necessary corrections are made.

4.7.2.6 Effects of the Battery Life on the Motors

As the battery life declines, the power being delivered to each motor will decrease. This needs to be addressed in the flight controller as this will have a relatively large effect on the drone's flight pattern. With a short life span, the drone batteries are draining quite rapidly. As the battery declines in overall charge, the motors will all together deliver more power. This inconsistency needs to be closely monitored and expected. As the power starts to drain, the PID loops will correct the speeds of each motor to ensure the drone remains stable. The power of the drown is explained more in section 4.6.6.

4.7.3 Initial Flight Testing

The initial flight testing is being performed in a local gym. We needed a space with plenty of room to try out different flight control settings. We chose an inside setting as the drone is designed for flying indoors and we eliminate all the hazards and excess forces outdoors. The space we used had heights exceeding our max height requirement so we were able to confirm our drone could reach the desired height of 10 feet. After PID settings and all the equipment is mounted, we took the drone to our testing location and flew around observing how the drone reacts to certain motions and how it corrects. Observing these components, we would keep that in mind for what corrections would need to be made. Using gym mats, we covered the floor and did our best to keep the drone low. Are biggest goal with testing is to do as little damage to the drone as possible. Without the mats, the hard floor below could potentially break a propeller or damage other extending parts of the drone. The mats will absorb some of the impact and prolong the life of our drone in the event of a crash landing. For the majority of the testing we will keep the drone fairly low to the ground and avoid exceeding certain heights to prevent major crashes. The huge amount of space allocated for a testing helps minimizes unwanted crashes.

We want the flight to be demonstrable in a small indoor environment. When the project is to be showcased, it is important to have a fail proof procedure to show all the functionalities. That means our battery must last a certain amount of time and all the gestures must be repeatable upon command. During our testing this is what we are looking out for. Exhaust a battery and become aware of what time span we are limited too. If this time is shorter than we presumed, adjustments will need to be made. Make our design more efficient or add batteries to the drone.

We also need to make sure each individual hand gesture is easily distinguishable. If two separate hand gestures are too similar and the drone is performing the wrong action, the gesture may need to be changed to a more recognizable one. These initial tests are important to lay the groundwork for improving our design and minimize failure. Once we reach a comfortable position we can then start adding to our design. Adding extra hand gestures and adding functionality to our drone would be a fun way to test how far we can take our project.

6.1.6 Bluetooth Testing

In regard to testing with Bluetooth, we will be able to test that the module is connected to the laptop computer by seeing the connection indicator LED on the module, as well as verify that on the laptop computer. We will also be able to verify this easily when setting up with new devices as we have several laptop computers, and we are able to connect to the module without it being connected to the drone. We will rigorously test the connection of the module at varying distances, with various obstacles in the way, as well as test the communication of the data with these variables to clearly define the limitations around our product. These are very important things to verify because the drone, if it gets out of control, can be very hazardous to surrounding objects and people. When we want to test the communication, we can simply communicate the data across without the drone actually flying or spinning the motors. As we are only communicating numbers across the Bluetooth connection, we can very easily output the numbers to the console of each respective device to ensure that all of the data is coming through accurately and precisely.

6.1.3.1 Altitude Sensor Testing

In regards to testing with the BMP180 barometric pressure sensor, we can verify this very easily by connecting it to the Arduino, and reading the output of the sensor in the Arduino's console that is accessible by the Arduino Software Development Environment, as that is how you debug most Arduino programs and we will be commonly using that tool. We will develop tests in which we elevate the sensor to previously measured heights and then look at the output given in the console to ensure that it is accurate to the level described in the technical specs of the sensor, which is a very low difference of 18 centimeters. The output described will essentially output the elevation level in relation to the provided atmospheric pressure reading at the floor level. It will output its estimated elevation level after a set interval that we provide it using the Spark Fun library to convert the sensor's data to a readable elevation level. Below is similar to the output that we are expecting to see in the Arduino SDE when debugging/testing the altitude sensor. **Figure 35** is a screenshot of the log output showing from the Arduino's Integrated Development Environment. The output is what we expect to receive using Sparkfun's library to show the exact altitude at a given time.

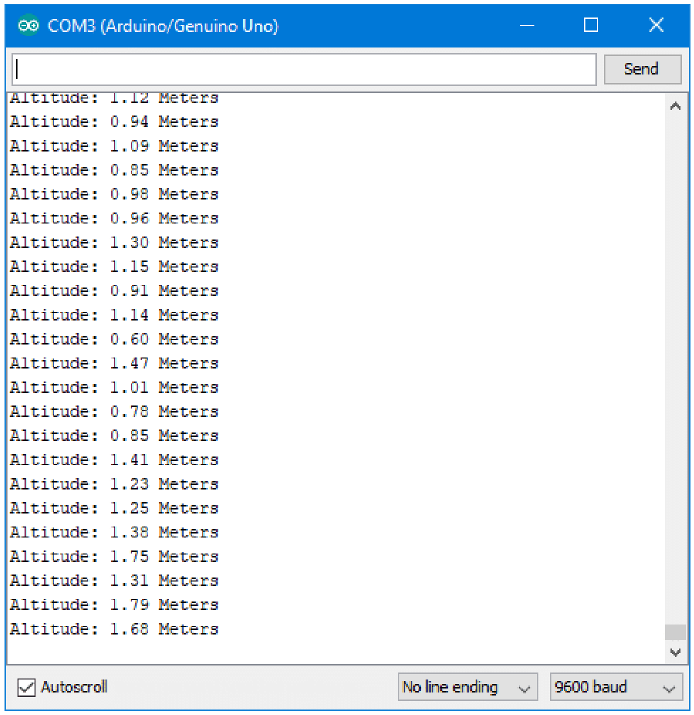


Figure 35 Altitude Sensor Output

4.7.4 Expected Adjustments

Building our project, we are going to run into dead ends and times we need improvements. There are some areas where it is safe to assume, changes may need to be made. It is important to be open to change and not get to focused on something stopping the progress of the project. The first change we may need to work around, is upgrading to a microcontroller with more space and a stronger processor. If our current processor cannot manage both flight controls and communicating to the user, we will have to upgrade our processors. Another change may be adding more power or making our design more efficient. Even commercial drones are notorious for having fairly short battery life. Having multiple batteries standby is another solution however, making our system as efficient as possible is the best option.

We may also need reprinted PCB. Errors can occur during the production phase or we can damage the board ourselves. Regardless of how the issue was caused it is important to have a reliable backup. Most sites are fairly reliable and can give us a low-cost board in a time no longer than a week. This is comforting as changes may need to be made or fixed. Same goes to say with individual components. Are goal being to choose reliable sensors and other key functional parts, however incidents happen. Crashes may cause breaking or defects can cause other issues, therefore we can have a backup readily available to avoid wasting time. With testing out main priority is to not waste time waiting for parts.

Being constantly at a stage of testing and improving will lead to the best result and possible outcomes.

Another area we have room to explore is what to do when the drone loses connection. Whether this is caused by a loss in power, out of range or interference, the drone needs a safe solution to land properly. The simplest solution would be once connection is lost, or when the battery reaches are certain percentage, the drone will perform a landing sequence and power down. This solution will be simple to implement but could also have some downfalls. The drone might land over something it should not land, causing harm to someone or the drone. This can be avoided by the user, when flying to always keep the drone out of danger and over a safe place to land. Another option would be to incorporate a back-tracking flight path, that can lead the drone closer to its initial starting path. This will require a more advanced fix and possibly a more powerful flight controller. If the starting position were marked, it could maneuver its way back to the starting position. This feature is seen in commercial drones and would be a possible addition to our drone. There is a lot of room to explore in this category and once our drone is working properly, we can focus on improving this. Once the drone loses connection or powers off mid-flight, there should be a clearly defined response.

4.7.5 Research and investigations

Drones have become increasingly popular in recent years, along with computer vision technology. When the idea came up for this project, as a group we researched the web for similar products. Although the market is super saturated with drones, there was no product exactly alike what we are attempting to design. This exclusivity is enticing and is another driving factor to do our best at designing and improving this product.

That being said computer vision and drones do go hand in hand. On the market, a popular product is a drone that follows motion. These are mainly used for tracking purposes and the drone will choose a target and track its movement. This utilizes the same computer vision concepts but differs in the sense that, our design will be independent of the user movement unless directly intended for the drone.

5.0 Printed Circuit Board

5.1 Printed Circuit Board Overview

We will be ordering the printed circuit board from online in search of a very low cost and a relatively short delivery time. This will help us combine the electrical components that we need to work together to get the drone up and running.

5.1.1 Ordering the PCB

To order the PCB, we need to build the PCB design, and then upload the design to the website that we are ordering from. We also will need to know how many pieces exactly that we are ordering, the number of layers we want, and the thickness of the chip. There are several other options that can be customized when it comes to PCBs, but for the purpose of our project we do not believe that we will need anything particularly customized.

5.1.1.1 PCB Company Options

When it comes to choosing the company that we will order the PCB from, there are limitless options. The primary requirements that we have are a relatively quick turnaround time and a low cost for a low volume. We only plan on ordering 2 boards, one for our main use and one backup in case something goes wrong in the mounting process for the components. We hope to find the board for under $10 each and to receive the board within, at most, two weeks from order time. We leveraged a website called pcbshopper.com in order to determine which company we should purchase the PCB from that meets our needs.

The most popular company to order PCBs from is JLCPCB.com, which is a company that has a special offer to provide prototype PCBs for roughly $2 each, however their minimum order count is 5 pieces. This company simply wants us to upload a ‘gerber’ file, which is the industry-standard file type for PCB designs. This website is especially appealing to us because of their low cost for each board, however we likely will not need 5 PCBs. We found this company to be able to deliver at a cost of roughly $10 for normal US registered mail delivery, which offers a turnaround time of 23 days.

An alternative to JLCPCB is Elecrow, which is a company based in China that will offer a total price of $13 with a turnaround of 16 days. This offer is compelling if we are on more of a time crunch, however with this company the minimum order is for 5 pieces still. The primary quality of Elecrow that would be useful to us is the quality of the product, as the reviews are much higher for Elecrow as compared to JLCPCB and other companies.

After looking at more results from the pcbshopper.com resource, it seems that most other companies are offering higher prices for similar products, and there are no companies that have 100% satisfaction with the product. If we were to find that then we would be able to completely eliminate the need for adding in buffer time for testing the PCB when we mount the components onto it.

5.1.2 Building PCB Design

We will need to build a very comprehensive design for the PCB in order to provide to our company of choice, in the format of a gerber file. This design will need to be built with a specific PCB design software, which we will be able to upload to the company that we want to order it from so that it can be built exactly to our expectations. For this, there are several options of software to build the designs. We will also be searching for free software only, as we are trying to keep the budget for this project at a minimum.

5.1.2.1 PCB Design Software Options

PCBWeb is one application that allows for designing electronics hardware. It has a very fast and user-friendly wiring tool to allow us to route all of our components to each other easily. This software offers also a parts catalog which will likely allow us to find common parts that users choose and offer much more information about (i.e. which pins are for Vcc and Gnd and more) the components that we are using, if they are available in their library of components.

Another option is called ZenitPCB, which is a user-friendliness focused program that will allow us to quickly spin up our PCB design, and it offers the functionality of directly converting the schematic design to a PCB if we are able to provide that to the software. It is unlikely that we will take advantage of this feature, however if we were on a large-scale this feature would be very useful because we would very likely have the schematic to convert automatically.

Several other tools are available, however many of them are Operating System specific, which is not ideal as well as being less commonly used and so there is less support in the online community for them. Because of this, we will be building our PCB design with PCBWeb, as it seems to be the most popular option and is the highest rated free software. From PCBWeb, we will generate a gerber file for us to upload to the company that we will be ordering the boards from.

5.1.3 Mounting Parts on PCB

In order to mount all of our parts onto the PCB, we will be taking advantage of a service offered nearby us called Quality Manufacturing Services, Inc. This incorporation offers a free service to mount most of our parts onto the PCB, so long as we provide the PCB and the design of where the components are meant to be fitted. This will allow for a professional to mount our components, which will reduce the likelihood of error that would occur if we were to mount it ourselves with our lack of better equipment and experience. This is a very common service that many students use to mount components onto their PCBs for their Senior Design projects and so we will also be following suit to ensure a better product.

The PCB will need multiple components mounted onto it. The largest portion of the PCB will be taken by the Arduino board, as that is the primary component that is powering everything and feeding most of the parts data and receiving all of the data. The Arduino is then complemented by the Bluetooth module, the accelerometer, and the altitude sensor in terms of components that will be feeding it data. Then, the ATMega328p will also be connected as the flight controller and will then communicate to the ESCs to provide the drone motors the appropriate speeds they should fly at. The voltage regulator will both need places on the PCB to be connected, as the ESCs will be directly connected to the motors.

5.2 Hardware Requirements

Because this project is heavily hardware-focused, we will have requirements based solely on the hardware. We will have our requirements directly line up to our tests, so that our tests are exactly testing our requirements, and ensuring functionality. Our core functionality however is divided up into Software Requirements, Hardware Requirements, and General Requirements. Our hardware requirements are more focused on how the hardware parts work together.

5.3 Potential Risks

While building this project, there are numerous potential risks. Primarily, the risks involve failing parts and electrical hazards. We intend to prototype at various levels to mitigate these risks. We are planning on testing each individual electrical component, such as the Arduino, the ATmega328p, the motors, and the ESCs. We intend to mitigate any risk of electric shock by measuring via multimeter the output of each component with the battery connected to it. This will allow us to understand exactly the output of the components to build a project that does not electrically fail or short. We also plan on testing in a closed environment due to the danger of drone motors hurting people. This will also allow us to fly the drone legally because we are doing it indoors and without anybody in the room, as to protect their privacy.

5.3.1 Drone Laws

Currently, in Florida, drones are not able to be flown without restrictions. We are required to submit for permission to fly our drone via the Federal Aviation Administration, which will require a registration process and a description of the drone and why we intend to fly it. Flying a drone and invading someone’s privacy is illegal, and because of the risk of several drone operators committing this crime, there have been laws placed to circumvent this risk. The laws are about registering the specific drone, following safety guidelines, keeping the drone within the line of sight, getting authorization to fly in any controlled airspace, and flying below 400 feet in any uncontrolled airspace. Because we are not building this product for outdoor flight, we do not need to worry about this risk. Therefore, we will be flying our drone completely indoors with authorization from the entire room without breaching any person's privacy. Being indoors also means that we are able to fly a drone without any licensing or certification. This allows our product to be much more marketable and accessible. The indoors setting also means that the airspace is private, which means that our drone's flight is not regulated by the FAA, however fault is still gone to the pilot if anyone is harmed during the drone's flight.

6.0 Prototype Construction

We will be constructing the prototype after all of the individual parts have undergone their associated tests. Similar to how a software application is tested with unit tests and system tests, we will perform the same type of testing on our project to ensure that all of our parts work individually, and then that they all work together. We will construct the prototype by first connecting the motors to the ESCs, then mounting those parts to the frame. From there we will connect the PCB that we will design for the combination of microchips that we will be using for the project. We will then connect power and test the drone out.

In order to build the prototype, we will need to go through a number of steps. Primarily, we will need to start with the drone motors connecting to the ESCs. Then we will test operation of the motors directly through the ESCs and the ATmega328p to see how the motors work. From there we will need to attach the motors and ESCs with wiring to the drone frame that we purchase.

Next, we will need to mount all of the components onto our designed PCB which is then to be mounted onto the drone frame in the very center, attempting to do our best to keep the weight balance as central as possible to allow for the smoothest balancing mechanism for the drone's motors and flight controller.

Once those parts are mounted, we will move on to the software side of the project. This part of the project can be done in conjunction to the actual hardware construction of the drone, as they are two separate parts that are not dependent on each other until we actually want to fly the drone.

The Arduino and the ATmega328p will need to be programmed along with the sensors and Bluetooth module connected, which we will need to work on having them all working in conjunction simultaneously.

The neural network will first need to be trained, and then we will need to create the GUI that shows the signal that it is reading from the webcam, along with the altitude of the drone and a log output to verify that the computer is sending the appropriate signals and that the drone is receiving those signals clearly.

Once the hardware and software have been completed, we will connect to the drone via Bluetooth and verify whether the drone is able to respond to our hand gestures passed to our laptop's webcam.

This will conclude our process of building our prototype. Of course, this is a very high level and very simple method of building our prototype, but it will be our responsibility to be flexible to account for unknown hurdles that we may come across and not let any bumps in the road keep us from progressing our project.

7.0 Owner's Manual

Our product was built with ease-of-use as a primary benefit of our drone compared to other drones available in the market. However, taking off for the first time does involve a generally lengthy process. In order to setup the drone for its first flight, the following steps will need to be followed:

1. Install the GUI software on the controlling laptop computer
2. Test every one of the hand motions in the table provided containing the hand gestures and make sure the correct output is shown in the log output
3. Charge the drone's battery to 100%
4. Turn on the drone's power
5. The drone will automatically go into pairing mode if it does not detect a nearby previously connected device
6. Open the controlling computer's Bluetooth settings
    a. Select G.O.D. to connect to the drone
    b. Enter the provided PIN to authenticate
7. Verify that the Bluetooth connection was successful on the log output of the GUI
8. Verify that the altitude sensor feature is providing output on the GUI, and that it is responding to actual changes in altitude for the drone (this can be done by manually picking up the drone or by flying it using the motors)
9. Start giving the drone commands via hand gesture

This product can be very dangerous, so please use caution when flying. As this product is a prototype, please fly this product in an area clear of animals and obstacles to prevent any injury or damage to the surroundings or the drone.

During prototyping, the laptop computer used was a 2017 MacBook Pro, with a 3.1GHz CPU. While you are welcome to use your own laptop, please understand that we do recommend something similar to what we were prototyping with or better to ensure that your hand gestures are read in and converted to commands to the drone quickly and efficiently.

Upon opening up the GUI, please verify that all components are visible and are structured like the picture shown below in section 4.4.3. This will allow you to make sure that the software has been installed correctly.

7.1 Troubleshooting Steps

If you find that the GUI software is not responding, please try all troubleshooting steps below:

1. Uninstall and reinstall the software
2. Restart your laptop computer
3. Check your camera settings for your specific computer.

If there are issues in which the hand gestures are not being recognized by the GUI software:

1. Verify that you are allowing the GUI to utilize the camera on your laptop computer
2. Please check every gesture to see if any of them are registered
3. Please verify that you have a high contrast between your hand and the background, so that the camera is able to clearly distinguish your gesture
4. Try using your alternate hand to mimic the gestures

If there are issues with connecting to the drone via Bluetooth, verify the below:

1. Is the blue LED light on the drone flashing when the drone has been turned on?
2. If the blue LED is flashing rapidly, the drone is still in pairing mode, which means that it is searching for a new device to connect to
    a. Ensure that your controlling device is within 30 feet of the drone
3. If the blue LED is flashing slowly, it has connected to a previously connected device and is awaiting input from that device
4. Should that be the incorrect device, please disable Bluetooth on the incorrect device to put the drone back into pairing mode so that your desired controlling laptop computer can see the drone in the Bluetooth settings

If the drone is not able to fly based on your commands, please verify the below:

1. Is the drone's battery charged to 100%?
2. If the motors are not moving at all, there is likely an issue with an on-board connection to the ESCs or from the ESCs to the motors
3. Verify that the log output shows the signal that the command has been received
4. If none of the above work, please reset the drone and quit the GUI
    a. Open the GUI
    b. Connect again via Bluetooth
    c. Verify on your laptop that you are connected to the correct device
    d. Verify the camera is enabled and your hand is visible in the frame
    e. Send a 'fly upwards' command

8.0 Administrative Content

8.1 Evaluation Plan

Our evaluation plan will consist of key points that our project is meant to meet, evaluation questions that will be answered with measurable outcomes, and an evaluation design that highlights our project's objectives and addresses the key shortcomings.

8.2 Key Evaluation Points

The key evaluation points that we are trying to reach with our project are the ability to fly the drone, the ability for our neural network to recognize the hand gesture passed to the camera, and the ability to communicate the action paired with that hand gesture to operate the drone.

8.3 Evaluation Questions

In order to meet those evaluation points, we need to be able to measurably test those points. Below are the measurable key questions that we aim to be able to answer positively after the successful construction of our project:

1. Can the neural network recognize our hand gesture within 500ms?
2. Can the drone react to our hand gesture within 1s?
3. Can the drone ascend with full stabilization, maintaining level during ascent?

8.4 Evaluation Design

For us to be evaluated, we will be reaching out to one of our professors or a provided proctor to go through all of our measurable requirements shown in the beginning pages of this report and verify that we have met those requirements. Of course, those requirements can change along the way, however we will be sure to provide the appropriate fields for the evaluator to fill in so that the input is unbiased when it comes to measurements and having a holistic understanding of our project. We will also include a list of the primary constraints that we were tied to during the making of the project and will leave a field for the evaluator to input how we circumvented those constraints in order to complete our project. This will allow the proctor to gain a much better understanding of the project, why everything has been done in the way it has been done, and the ideology behind what we plan to develop with our prototype.

Furthermore, to allow for more freedom for the evaluator, we will offer a form in which they can list their specific concerns. Our team will answer those concerns upon hearing them, and if our project does not address those concerns then we will be able to collect that as data from the evaluator to be addressed in the next iteration of the project.

8.5 Proposed Schedule

Every week we plan on meeting at least twice and discussing our research and progress. On top of these meetings we will strictly follow our milestone plan listed below in **Table 14**.

Date	Semester	Milestone
May 27, 2019	Summer 2019	Divide & Conquer 1 Assignment
May 28, 2019	Summer 2019	Approve projects and begin research
June 3, 2019	Summer 2019	Begin our individual writing parts
July 2, 2019	Summer 2019	Complete parts and share content
July 4, 2019	Summer 2019	Begin integrating all three parts
July 16, 2019	Summer 2019	Finalize document and print final copy
July 30, 2019	Summer 2019	Submit the final document
August 27, 2019	Fall 2019	Order all the necessary parts
September 3, 2019	Fall 2019	Being assembling drone
September 10, 2019	Fall 2019	Ensure individual components are working
September 17, 2019	Fall 2019	Build the first prototype
September 24, 2019	Fall 2019	Test prototype
September 26, 2019	Fall 2019	Use following time to redesign and rebuild
November 19, 2019	Fall 2019	Finalize drone for final presentation
November 26, 2019	Fall 2019	Present our project

Table 14 Milestones

8.6 Budget and Finances

Budgeting was a very large concern when it comes to this project. Most drone projects end up costing a very large amount, and we found in our research that this was mostly due to parts breaking and expensive drone parts. Other reasons for this can be due to students buying pre-built drones and add to its functionality. This was among our options, however we decided to try and build as much of the drone as we reasonably could, so that we would be able to have the drone be marketable and cost us less in prototyping.

Because we were not buying a drone that was already built and just integrating our solution to control it via controlling the given controller, we circumvented the entire need of buying a controller as well, since our laptop computer is directly controlling our drone now, with no middle man that could have been a dependency for our project.

Furthermore, since we avoided buying an already built drone, we are able to deeply understand all of the working parts of our drone, which allows us to understand what is broken and how to fix or replace that piece that is broken, whereas if we were to buy a built drone, we would have to dismantle the entire thing to fix something that could be going wrong with the drone.

While we saved greatly on buying the drone in parts, we also tried to minimize our spending on a part-level basis. This means that cost-effectiveness was a factor when it comes to each and every part that we chose to use to build our product.

Table 15 maps out our expected budget for our project. These are estimated costs and we expect our total budget to be around the value listed in the last row.

Component	**Estimated Cost**
Development Equipment	$100.00
Bluetooth Module	$10.00
Brushless Motors x4	$70.00
Propellers x4	$20.00
Lightweight Drone Frame (150mm)	$20.00
Electronic Speed Controller x4	$100.00
Voltage Regulator	$5.00
Batteries	$40.00
Accelerometer Sensor/Gyroscope	$20.00
Altitude Sensor	$10.00
PCB Printing	$30.00
ATMega328p	$10.00
Miscellaneous Components	$200.00
Total:	**$635.00**

Table 15 Proposed Budget

8.6.1 Software Development

When it comes to the GUI, we developed the entire thing by ourselves. This was attributed to the fact that among us are 3 computer engineers, and so we have a strong background in software development. This allowed us to build the GUI for the drone all with the use of free tools and the development was done entirely by us.

In regard to the neural network, one of the members of the group, particularly Anshul Devnani, had previous coursework experience in Machine Learning, and so he was knowledgeable to help the 3 of us build the neural network free-of-charge through the use of Keras.

The minimal hardware necessary for the neural network recognition system was simply a computer that has a webcam and a GPU. We have not yet tested the neural network's speed from one laptop's GPU specs to another, however we believe there to be a difference in time taken to train the model as well as the time taken to recognize the gesture. Despite that, most laptop computers are quite capable of handling this task if it is the only one currently running on the computer and there are not several other tasks running on the GPU. Since our intention was to minimize the spending, we avoided the need of purchasing new hardware just to control the drone, and so we decided to use the computers that we already owned, as they already had capable GPUs and had built-in webcams.

8.6.2 Wireless Communication

For the wireless communication, we decided to use Bluetooth, which we found to be very cheap and accessible. Most of the Bluetooth modules were available for under $40, however we decided to go with a very popular choice that was only a fourth of that price. We chose this one because it was popular with DIY projects, and it was cost-effective for us to use simply the module tacked on top of the Arduino rather than buy an Arduino with Bluetooth already built in. This form of communication also proved cost-effectiveness in the reasoning that most computers have Bluetooth built-in, so it was something that computers already have installed and would not require any additional hardware for the computer for the drone and the computer to connect.

8.6.3 Battery

In terms of battery, we designed our product to be able to use a rechargeable battery. This means that we are not required to spend large amounts of money on several batteries that will only last a 15-20 minute flight and then be unusable afterwards. Most drones typically use a rechargeable battery, and so we will be using the same. We were very unsure about the capacity of battery we needed, so we are starting with a 3S Rechargeable Lithium Polymer battery that is only a 900mAh capacity to start, however we will likely need to upgrade the battery as we go on. Fortunately, the low amperage batteries are relatively cheap, most being under $20. We will certainly need to spend an additional amount on recharging the battery, which we plan on prioritizing despite price due to us trying to avoid destroying or damaging any of the batteries we use to test.

8.7 Division of Labor

A lot of work on the project is performed as a team although for individual research we broke down the topics into sections and divided the work. We all chose the topics that best suited our expertise and strengths. Anshul Devnani has a passion for computer vision and is hoping to pursue a career in the field. As a computer engineer, he is currently working as intern at Leidos as a system integration engineer and previously work as a CWEP for two years. With his professional programming experience and computer knowledge, gained through courses at UCF, he took on the task of planning the graphic user interface and research deep neural networks and computer vision.

Pranay Patel has a history in network communication and power systems. He is a computer engineering major and is researching all the different types of network communications, memory management, PCB construction, and system power. He currently works at Darden Restaurant as an implementation engineer intern working with a skilled team of engineers to maintain and improve a back-end system for digital marketing. He is using the knowledge he has gained from work and class to research what is necessary to connect the drone to the user interface and how to power the drone.

Bernardus Swets is a computer engineering major at UCF, following the digital track, and took on the task of researching the flight controls. He focused on looking into the different drone designs and corresponding hardware. He also looked into the flight control software. Having experience with linear control system, he researched what it would take to balance our drone using PID loops. He works as a system engineer CWEP at Lockheed Martin. Between his knowledge gained from professional experience and in class he will focus on designing and balancing the drone.

9.0 Conclusions

To conclude, our project is going to be a new way to interact with drones and can pave the way for a new way to interact with other machines as well. The extensibility of gesture-controlled devices is rapidly growing, and it is also extremely beneficial to those with disabilities regarding sound. Because those with disabilities regarding sound tend to communicate through sign language or the like because they are unable to talk, this will allow them an easy way to communicate with devices via gestures that they are already very familiar with. Our project, a gesture-operated drone, is simply an implementation of a gesture-controlled device. The Gesture Operated Drone allows for an extremely simple way to operate a drone in comparison to the unwieldy RC remotes that commonly come with drones to operate them.

Our gesture schema has been set up to be accessible to any and all that have full motion of all of their fingers. This schema allows for an extremely wide market for our drone, because most people are able to do all of these simple gestures with ease. In addition, our target market for the product is people that want to pick up drones as a hobby or for a just-for-fun purpose. This target market is extremely wide because drones are a relatively new concept and there are an increasing amount of people taking up photography and videography in today’s time. Generally, most drones are either flown for fun or to get a photograph or video from an angle that mimic’s a bird’s eye view. This allows for a very unique picture or video and so is desired by many people exploring the hobby of photography.

While there are many people that want to take up flying a drone for whatever purpose, unfortunately it can take time to learn how to operate the drone safely with the common RC remote that comes with most drones, and drones can be very dangerous if flown incorrectly or if it goes off-course due to the operator not knowing how to use the remote control. Our project is designed to skip this extreme learning curve by making predefined actions for the drone, such as elevate, de-elevate, move left, right, forward, and backward, so that the user can simply pick up the drone and start using it without the worry of accidentally thrusting the drone into a tree, damaging the several hundred dollar drone that they just bought minutes after using it. On top of that, the user does not even need to operate any extraneous hardware to perform those actions, they simply need to only use their hands in front of their computer screen to make them happen.

Of course, there is one particular limitation that we immediately noticed when compared to using a remote control that seemed to be a disadvantage to our solution. That limitation would be the latency with which the signal is received. In particular, there is much more computation going on when it comes to the two forms of operating the drone. Both using a remote control and our solution of communicating from a laptop computer requires a wireless communication method to command the drone to act. However, they differ in the computation required to create the command to the drone. Our solution must convert the input hand signal from a camera to match a model, while the remote control simply needs to convert analog input to digital input and send the appropriate command accordingly. Our projection was that this would cause an extreme latency that would impact the drone's functionality and wieldiness of the controls. However, as we are training the model, we are seeing a much faster response time than we previously expected.

This project is very helpful in combining our multitude of coursework to produce a working drone from scratch and a GUI built from scratch with integrated Machine Learning and Computer Vision applications. We were also able to communicate between the two wirelessly. Every part of our project could of course be improved upon, and when going to market we would be able to minimize costs by manufacturing in bulk and not wasting as many parts as we did during testing. This would allow for the product to be much more marketable.

Additionally, we could offer the product as a modular product, in which we offer the drone with better Bluetooth modules for extended range, a faster chip to handle flight control, better motors, a larger frame, and the like. This would make the customer able to make the product more attuned to their use case.

Another addition that we could implement to our drone could be the ability for the drone to have the camera on board, with a built-in processor able to handle the neural network processing. This would allow the drone to be 100% hands free, and completely without a remote at all. The primary condition that we would run into there is keeping the subject's hand in view at all times, or to train a neural network to recognize hands despite all of the extraneous input received via the camera's lens.

To sum up, this project was extremely educational and allowed us to follow the lifecycle of an integrated project from start to finish. We were able to generate actual user requirements based on measurable items, and we were able to build a prototype that can scale to multiple things. This project can evolve in hundreds of ways and can really make a large impact on consumer tech worldwide if it was to reach the global market. The reasoning behind that is that as we progress in technology, we are decreasing the direct touch interaction continuously. One particular example of that is how fast voice recognition technology is spreading and ramping up. However, our product is able to target those that are not able to speak fluently or do not speak a common language that is supported for most voice-recognition services out of the box. This allows for us to target a near-universal market, because humans everywhere can understand some hand signals, and we have designed our product to account for using hand signals that are understandable no matter what differences a person may have origin-wise.

Appendix A Resource and Citations

/@piotr.skalski92. “Preventing Deep Neural Network from Overfitting.” *Medium*, Towards Data Science, 4 Jan. 2019, towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a.

“Convolutional Neural Network.” *Wikipedia*, Wikimedia Foundation, 29 July 2019, en.wikipedia.org/wiki/Convolutional_neural_network.

/@_sumitsaha_. “A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way.” *Medium*, Towards Data Science, 17 Dec. 2018, towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

“Tkinter vs PyQt Detailed Comparison as of 2019.” *Slant*, www.slant.co/versus/16724/22768/~tkinter_vs_pyqt.

CS231n Convolutional Neural Networks for Visual Recognition, cs231n.github.io/convolutional-networks/.

“Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson.” *AltexSoft*, www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/.

“KDnuggets.” *KDnuggets Analytics Big Data Data Mining and Data Science*, www.kdnuggets.com/2018/01/mlaas-amazon-microsoft-azure-google-cloud-ai.html.

“AlexNet.” *Wikipedia*, Wikimedia Foundation, 26 June 2019, en.wikipedia.org/wiki/AlexNet.

Rizwan, Muhammad. “LeNet-5 - A Classic CNN Architecture.” *EngMRK*, 30 Sept. 2018, engmrk.com/lenet-5-a-classic-cnn-architecture/.

Jeremy Jordan. “Common Architectures in Convolutional Neural Networks.” *Jeremy Jordan*, Jeremy Jordan, 20 Oct. 2018, www.jeremyjordan.me/convnet-architectures/.

“Keras vs TensorFlow vs PyTorch: Deep Learning Frameworks.” *Edureka*, 22 May 2019, www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/.

/@DevEconomics. “What Is the Best Programming Language for Machine Learning?” *Medium*, Towards Data Science, 7 Jan. 2019, towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7.

“A Beginner's Guide to Backpropagation in Neural Networks.” *Skymind*, skymind.ai/wiki/backpropagation.

/@avinashsharmav91. “Understanding Activation Functions in Neural Networks.” *Medium*, The Theory Of Everything, 30 Mar. 2017, medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

“Online Diagram Software & Visual Solution.” *Lucidchart*, www.lucidchart.com/.
Clausing, John R. HauserDon. “The House of Quality.” *Harvard Business Review*, 1 Aug. 2014, hbr.org/1988/05/the-house-of-quality.

Brokking, J.M. “Project YMFC-AL - The Arduino Auto-Level Quadcopter.” *Brokking.net - Project YMFC-AL - The Arduino Auto-Level Quadcopter - Home.*, www.brokking.net/ymfc-al_main.html.

Ryan, and General Electric. “Brushless Inrunner vs Outrunner Motor?” *Go Back to the Front Page*, 24 Aug. 2018, www.radiocontrolinfo.com/brushless-inrunner-vs-outrunner-motor/.

Osmanbasic, Edis. “Three-Phase Electric Power Explained.” *Engineering.com*, www.engineering.com/ElectronicsDesign/ElectronicsDesignArticles/ArticleID/15848/Three-Phase-Electric-Power-Explained.aspx.

“Choosing the Right Quadcopter Frame.” *QuadHangar*, 14 June 2016, www.quadhangar.com/choosing-the-right-quadcopter-frame/.

“Best Indoor Drones - Fly in the Living Room without Wrecking the Place.” *Drone Rush*, 30 July 2019, dronerush.com/best-indoor-drones-fly-living-room-11926/.

Allain, Rhett. “How Do Drones Fly? Physics, of Course!” *Wired*, Conde Nast, 3 June 2017, www.wired.com/2017/05/the-physics-of-drones/.

“Quadcopter PID Explained.” *Oscar Liang*, 20 Jan. 2019, oscarliang.com/quadcopter-pid-explained-tuning/.

“Electronic Speed Control.” *Wikipedia*, Wikimedia Foundation, 3 June 2019, en.wikipedia.org/wiki/Electronic_speed_control.

Corrigan, Fintan. “How A Quadcopter Works With Propellers And Motors Explained.” *DroneZon*, DroneZon, 18 July 2019, www.dronezon.com/learn-about-drones-quadcopters/how-a-quadcopter-works-with-propellers-and-motors-direction-design-explained/.

"How Gyroscopes Work." *Robot Academy*, 30 July 2018, robotacademy.net.au/lesson/how-gyroscopes-work/.

"A-610 Acceptability of Electronics Assemblies Training and Certification Program." *IPC*, 22 Apr. 2019, www.ipc.org/ContentPage.aspx?pageid=IPC-A-610.

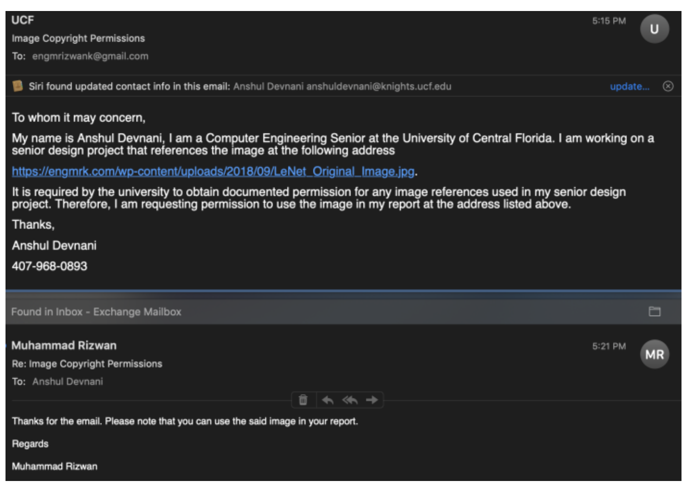
Henney, Marc. "Bluetooth Versions Comparison & Profiles." *RTINGS.com*, 6 July 2017, www.rtings.com/headphones/learn/bluetooth-versions-comparison-profiles.

"How to Set Up the BMP180 Barometric Pressure Sensor on an Arduino." *Circuit Basics*, 13 Aug. 2018, www.circuitbasics.com/set-bmp180-barometric-pressure-sensor-arduino/.

Murison, Malek, and Malek MurisonMalek Murison. "ISO Proposes Global Drone Standards." *DRONELIFE*, 22 Nov. 2018, dronelife.com/2018/11/22/iso-proposes-global-drone-standards/.

"The Guide to Bluetooth Modules for Arduino." *Into Robotics*, Into Robotics, 18 Jan. 2015, www.intorobotics.com/pick-right-bluetooth-module-diy-arduino-project/.

Appendix B Copyright Permissions


UCF
5:15 PM
Image Copyright Permissions
To: engmrizwank@gmail.com

Siri found updated contact info in this email: Anshul Devnani anshuldevnani@knights.ucf.edu update...

To whom it may concern,

My name is Anshul Devnani, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

https://engmrk.com/wp-content/uploads/2018/09/LeNet_Original_Image.jpg.

It is required by the university to obtain documented permission for any image references used in my senior design project. Therefore, I am requesting permission to use the image in my report at the address listed above.

Thanks,

Anshul Devnani

407-968-0893

Found in Inbox - Exchange Mailbox

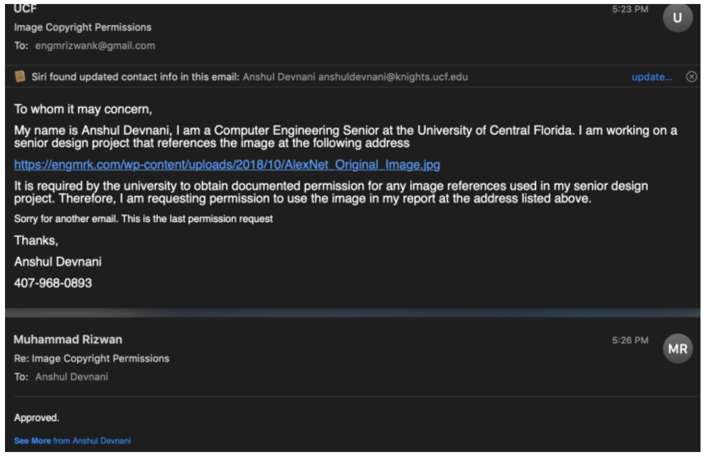
Muhammad Rizwan
5:21 PM
Re: Image Copyright Permissions
To: Anshul Devnani

Thanks for the email. Please note that you can use the said image in your report.

Regards

Muhammad Rizwan

Figure 36 Copyright Permission LeNet-5 Architecture


UCF
Image Copyright Permissions
To: engmrizwank@gmail.com
5:23 PM
U

Siri found updated contact info in this email: Anshul Devnani anshuldevnani@knights.ucf.edu update...

To whom it may concern,

My name is Anshul Devnani, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

https://engmrk.com/wp-content/uploads/2018/10/AlexNet_Original_Image.jpg

It is required by the university to obtain documented permission for any image references used in my senior design project. Therefore, I am requesting permission to use the image in my report at the address listed above.

Sorry for another email. This is the last permission request

Thanks,

Anshul Devnani

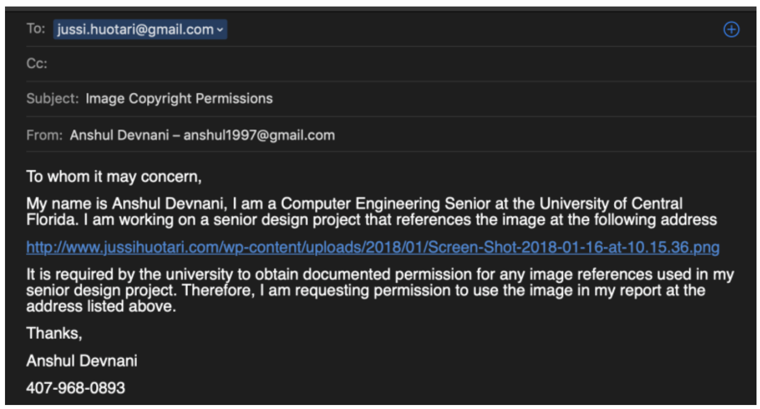
407-968-0893

Muhammad Rizwan
Re: Image Copyright Permissions
To: Anshul Devnani
5:26 PM
MR

Approved.

See More from Anshul Devnani

Figure 37 Copyright Permissions AlexNet Architecture


To: jussi.huotari@gmail.com

Cc:

Subject: Image Copyright Permissions

From: Anshul Devnani – anshul1997@gmail.com

To whom it may concern,

My name is Anshul Devnani, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

http://www.jussihuotari.com/wp-content/uploads/2018/01/Screen-Shot-2018-01-16-at-10.15.36.png

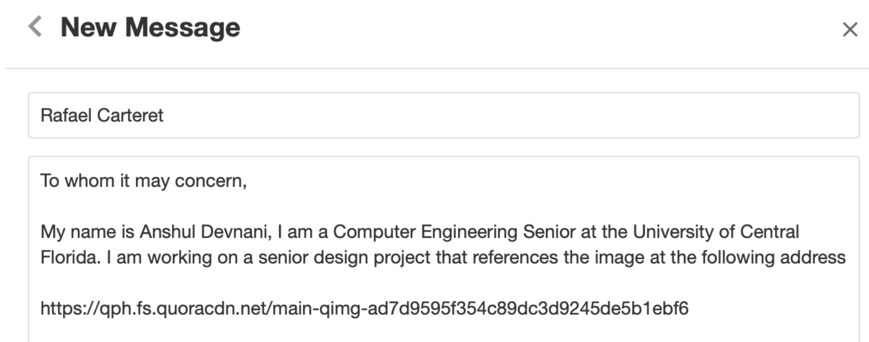
It is required by the university to obtain documented permission for any image references used in my senior design project. Therefore, I am requesting permission to use the image in my report at the address listed above.

Thanks,

Anshul Devnani

407-968-0893

Figure 38 Copyright Permission Request Loss & Accuracy vs Epoch


New Message

Rafael Carteret

To whom it may concern,

My name is Anshul Devnani, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

https://qph.fs.quoracdn.net/main-qimg-ad7d9595f354c89dc3d9245de5b1ebf6

Figure 39 Copyright Permissions Request Overfitting Graph

ben Swets
Thu 8/1/2019 5:38 PM
SUPPORT@EPOWERHOBBY.COM

To whom it may concern,

My name is Bernardus Swets, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

https://www.amazon.com/Hobbypower-1000kv-Brushless-Multicopter-Quadcopter/dp/B00E7LG85O/ref=sr_1_1_sspa?crid=1138GF78SI572&keywords=1000kv+brushless+motor&qid=1564695388&s=gateway&sprefix=1000+kv%2Caps%2C250&sr=8-1-spons&psc=1

It is required by the university to obtain documented permission for any image references used in my senior design project. Therefore, I am requesting permission to use the image in my report at the address listed above.

Thanks,

Bernardus Swets

407-968-0893

Figure 40 Copyright Permissions Request Brushless Motor

ben Swets
Thu 8/1/2019 5:32 PM
service@dhgate.com

To whom it may concern,

My name is Bernardus Swets, I am a Computer Engineering Senior at the University of Central Florida. I am working on a senior design project that references the image at the following address

https://www.dhgate.com/product/hot-sale-rc-bec-30a-esc-motor-speed-controller/403693034.html

It is required by the university to obtain documented permission for any image references used in my senior design project. Therefore, I am requesting permission to use the image in my report at the address listed above.

Thanks,

Bernardus Swets

407-968-0893

Figure 41 Copyright Permission Request ESC