

G.O.D. (Gesture Operated Drone)

Group 3 - Pranay Patel, Anshul Devnani, Bernardus Swets

Computer Engineering Majors

Divide and Conquer 2.0

EEL 4914



1. 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. 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. As a group of computer engineers, we have a fundamental understanding of the programming and electrical skills necessary for this project. Furthermore, this project will allow us to work with and learn about popular technologies of the time, including Computer Vision and Machine Learning. 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. Project Definition



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 control or smartphone. 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.

Overall this project incorporates everything we have learned as computer engineers and combines it into one project. The project's main focuses are computer vision, designing microcontrollers, communicating over a network, designing a graphic user interface, programming an embedded system, and PCB routing/design. We will design a graphic user interface that utilizes a webcam which will be used by the person in order to piloting the drone. The various hand gestures will be listed and explained so the user knows the drone's capabilities. After performing the desired action, the command will be processed into information able to be understood by the microcontroller, that will then be wirelessly communicated to the microcontroller and the desired output will be performed.

Testing is an important part of our project and we have put plenty of thought into how we will test our design. The most important thing is to locate an indoor space large enough to fly the drone. Choosing an indoor drone saves us all from requiring drone licenses and makes the project overall more hassle free. Our local gym has multiple indoor facilities with plenty of room for testing. We have reached out to the gym, and confirmed there are certain times when those spaces are empty, and during those times we can freely test our design.

In addition, our project is meant to be a base prototype, and can be expanded upon to outdoor drones. We have defined the project to be on a very small scale to reach a minimum value product, but to get our application onto an outdoor drone will simply involve larger and better motors and sensors to have an increased range and more seamless functionality. Our base prototype is meant to be about building a drone responsive to gestures, rather than building a drone to be able to handle difficult conditions during flight. Drone optimization and upgrading sensors and motors is not particularly our expertise, and so we are looking to build a base product that more experienced drone makers can build upon if they so choose.

With numerous amount of hand gestures, there are plenty of motions you can incorporate into the design. In the table below are a few of the proposed movements we will incorporate. So far we have planned the following movements. In our specific design, we will incorporate directional movements, a takeoff movement, a landing, and miscellaneous other movements. The list of movements is growing and we will most likely add to it and refine it as necessary while we develop the final product. **Table 1** shows the initial set of gestures used for basic drone flight.

User Action	Result
No Gesture	Hover in place/autolevel
	Thrust upwards
	Drone flies forwards






	Drone flies to the left
	Drone flies backwards
	Drone flies to the right
	Thrust Down
	Drone will land at current position

Table 1 Initial Hand Gesture Set

One of the challenges we will face is creating a user interface that will recognize the user's hand gestures. Our plan is to utilize the webcam integrated in a laptop to help achieve hand gesture recognition. In addition to the webcam, concepts in machine learning will be utilized. Neural networks are a core concept in machine learning and by definition are a programming paradigm that allows computers to learn from observational data. Neural networks are usually trained to do a certain task whether it be image recognition, speech recognition, or natural language processing. In our case, we will train the neural network model to recognize hand gestures. The first step to training a neural network is to create a dataset that is used to train the network. The dataset consists of thousands of images that replicate the hand gesture set in Table 1. The second step is to create the neural network model. The purpose of the model is to extract and remember features from the training dataset. The third step is to train the created model. This is done by feeding the training dataset so the model can extract and remember features of each class of gesture. The fourth step is to test our model on

data that our network has not seen in order to check the validity of our model. If our accuracy is bad, we must tweak our model until the best accuracy is achieved. After we are happy with our model, we will use it to predict hand gestures in real time using the integrated webcam on a laptop. Based on the prediction, a signal will be sent to our drone via a wireless communication medium. The drone will then react to the signal it received.

In order to train a more robust and accurate model, a generally big data set is required. The plan is to have at least 2000 images for each hand gesture, this is necessary in order to create a model that is accurate and recognizing hand gesture. The bigger the dataset, the higher chance of accurate prediction. In order to actually create the dataset, manually taking 2000 pictures of each hand gesture will be tedious and time consuming. A better approach is to take a 34 second video at 60 frames per second of each gesture and then split that video up into frames. A 34 second video at 60 frames per second will produce 2040 frames. A simple python program will be written to accurately able to capture video, split the video into frames, and finally save each frame.

There are many APIs and frameworks that help with creating a classification neural network model. Some APIs and frameworks include Pytorch, TensorFlow, Keras, etc. In our project, Keras will be used to create the recognition model. Compared to others, Keras is usually used for smaller data sets. In this case, a small data set consists of thousands of images. Large datasets generally have millions of images. In addition, Keras is more readable and concise, which is not the case with Pytorch and Tensorflow. More readable and concise code leads to easier code debugging which is another big advantage of Keras. Using the Keras API we will be able to create a simple, powerful, and accurate hand gesture classification model.

Building a proper neural network model is essential in creating accurate gesture recognition software. Before starting to create the model, a basic understanding of neural networks is needed. There are many types of neural networks, the one we will utilize in this project is called a Convolutional Neural Network or CNN. CNN's are excellent for video or image recognition. CNN's implement recognition by using convolution layers that extract features of the input image. The model then learns these features so when a brand new input image is passed through the model, an accurate prediction is given. The goal of CNN's is to create a model with parameters that maximize accuracy. This process of maximizing accuracy requires a lot of trial and error of adjusting model parameters.

After the model has the correct parameters and accuracy rate, Keras has a built in function that will predict and classify any input image. For example, if a new input image of a fist is passed as a parameter into the predict function, the function will compute its prediction based on the neural network weights and number of defined

classes. The number of classes in our implementation is 8, one for each gesture. The predict function will assign a probability or value between 0 - 1 to each class. The class with the highest number will be the final prediction of what the model thinks the input image should be classified as. In our project, a prediction will be made in real time for every webcam frame captured. Based on the models prediction, our drone will react accordingly. One main goal and challenge is to maximize our model's prediction accuracy so proper commands are sent to our drone.

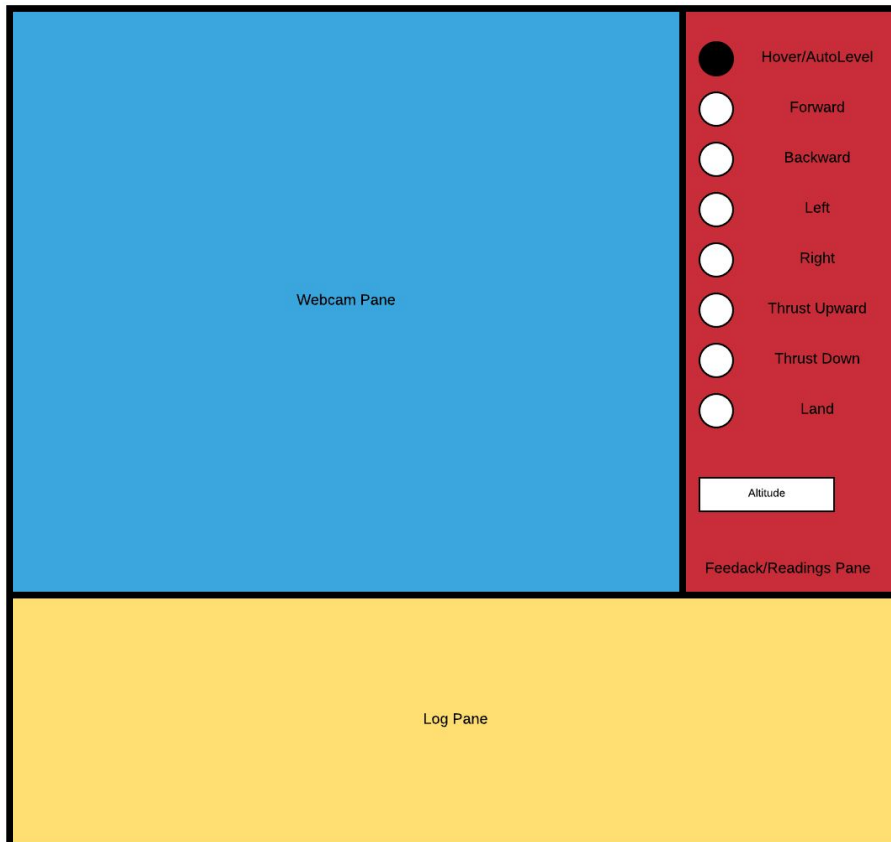


Figure 1 GUI Layout Plan

The GUI will be used to provide more organized user interaction. The proposed layout of the GUI is shown in Figure 1. The GUI will consist of 3 window pane, a webcam pane, a log pane, and a feedback/readings pane. The webcam pane will consist completely of webcam feed from the laptop. This is used so the user can see themselves giving hand gestures. The hand gesture is read from the webcam pane and a prediction is made based on the hand gesture. How the prediction is made is done “behind the scenes” not shown to the user. However, the user will see the result of prediction in the feedback/readings pane. The feedback/readings pane will consist of 8 radio buttons, one for each hand gesture. Based on the prediction made by the neural network model, the corresponding radio button will be filled in signifying the command

the drone will start executing. In addition to the radio buttons, there will be a field that will display the altitude of the drone in real time. Finally, the log pane will be used to give textual feedback to the user. All commands sent between the GUI, Drone and vice versa will be displayed in the log pane. For example, if the user gestures a thumbs up to the webcam, a message along the lines of "Sending Thrust Upwards Command To Drone" will be displayed on in the Log pane. In addition, when the drone receives the Thrust Upwards command, an ACK message will be sent from the Drone to GUI and a message similar to "Drone has received Thrust Upwards Command" will be shown in the log pane. This will be repeated for every hand gesture.

There are various forms of wireless communication, however for this project we have chosen to go with Bluetooth. Bluetooth modules are fairly affordable and relatively easy to incorporate into our design. Bluetooth allows for communication from the user's side to the drone and visa versa, from the drone to the user. This two-way communication can be used to our benefit and can open the door for a better end product. Bluetooth connectivity is now available on most smartphones and laptops. This makes connecting to the device much more user-friendly and allows for some scalability for our project to maybe expand to smartphones. The Bluetooth protocol has been continuously evolving over the years, and it is getting better at using less energy to operate. This is especially important to the G.O.D. because drones use an extremely high amount of power, and most drones can only operate for about 25 minutes from a fully charged battery. Bluetooth is also gaining increased range as the technology evolves, and while we will be operating our drone via a portable laptop computer, it will be ideal for us to control the drone as far as possible. Furthermore, Bluetooth allows each device to remember the other one, meaning that the Bluetooth ID is saved on both devices and means that there is relatively no setup after it has been done initially. 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.

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.

In regards to compatibility and difficulty, we have also researched this topic. Arduinos are very popular for basic DIY projects, and so, the SDE 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 regards 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 the 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.

Using Python or a similar programming language, we will build a simple Graphic User Interface (GUI). Upon launching, it will have the webcam interface, a couple buttons with start and stop capabilities, and also a list of all the available arm movements. There will be an indicator that allows the user to know whether or not the device is connected. The main goal of the platform is to make utilizing the drone as simplistic as possible.

The drone will be built from scratch and since it will be an indoor-based drone, the frame will not be bigger than 150mm. The heart of the drone will be an Arduino (or similar) based microcontroller which will control bluetooth receiving and transmitting, motors, sensors, ESC. Ideally, we would like all sensors to be capable of I2C communication. We chose this particular type of communication because of its low cost, easy extensibility, and since it is more widely used. The specifics of the drone flight will be handled in the embedded drone code. For example, if the user gestures a thumbs

down on the user interface, a signal is sent and decoded by the drone as a thrust down action. The embedded code will then handle slowing down the propellers in order to bring the drone down in altitude. The process will be used for every gesture used. An altitude and gyroscope sensor will be used to ensure a more stable flight.

Our plan is to use a ATmega328p alongside an MPU 6050 to control the drones flight commands and process the movements. A PID controller is important for the drone's stability and the flight controls. Having a closed loop system is essential for the drone's flight. With the inconsistency of the individual motors, a system is programmed to stabilize the drone to stationary stable flight. This task can be challenging but programs such as multiwii facilitate the process. For the trial and error process, the motors can be attached at a single pivot point. While attached the k_p , k_i , and k_d variables will be tested one by one and manipulated until the drone can reset back to neutral with no oscillations. The drone will be designed in a way so when no gesture is being recognized the drone will default to an auto leveled position. The ATmega328p is very good at relaying data to a flight controller, but we have planned to also include the PID looping program. If there are any issues and the ATmega328p isn't powerful or large enough, we can upgrade to the Amtel SAM3XE8. Another solution would be to use the ATmega328p for sending data to a dedicated third party flight controller. If needed we can connect the two via UART.

3. Requirement Specifications

Table 2 defines the initial set of requirements for the project.

The drone frame will be no larger than 150mm
The drone will be able to reach a height of 10 ft
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 be able to receive signals over bluetooth communication from within a range of 20 feet.
The drone will be able to convert the signal received over Bluetooth within 500 ms
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 feedback/reading pane will highlight the correct predicted gesture within 1 second
The user interface will be able to recognize each of the 8 gestures
The drone will utilize propellers of 3 inches or smaller
The time from the user doing the gesture to the drone reacting to it will be a maximum of 2 seconds.
The user interface GUI will consist of a webcam pane, log pane, and miscellaneous pane, as shown in Figure 1 .
The Neural Network will produce an accuracy of a minimum of 95 percent.
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 will utilize electronic speed controllers to help control the propellers.
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 utilize 4 brushless motors with KV above 3000
The drone will utilize an ATmega328P microcontroller

Table 2 Requirement Specification

4. Design Constraints

Table 3 defines the design constraints for the project.

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

5. House of Quality

Figure 2 shows the proposed House of Quality for the project.

Relative Weight	Customer Importance	Direction of Improvement	Functional Requirements										Customer Competitive Assessment			
			Customer Requirements										Gestured Operated Drone	DJI	GoPro	PowerVision
			Weight	Size	Durability	ATmega328p	Gyroscope	Indoor Drone	LiPo Powered	Altitude Sensor	Accelerometer	Brushless Motors				
19%	5	▲											2	5	4	3
4%	1	▼	●	○	▽				▽				3	2	1	5
7%	2	▲		●				○					2	5	4	3
19%	5	▲			▽	●					●	○	2	3	4	5
7%	2	▼	▽	▽	○								4	3	5	2
19%	5	▲			○								5	3	2	1
7%	2	▲						●		▽			5	1	2	3
4%	1	▲						●				●	5	2	4	1
11%	3	▲			○								2	5	4	3
4%	1	▼		▽				▽					2	3	5	4
Importance Rating Sum (Imp)			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 Improvement	
Maximize	▲
Target	□

Figure 2 House of Quality

6. Block Diagram

Figure 3 shows the proposed block diagram for the project. All blocks are currently in the research phase.

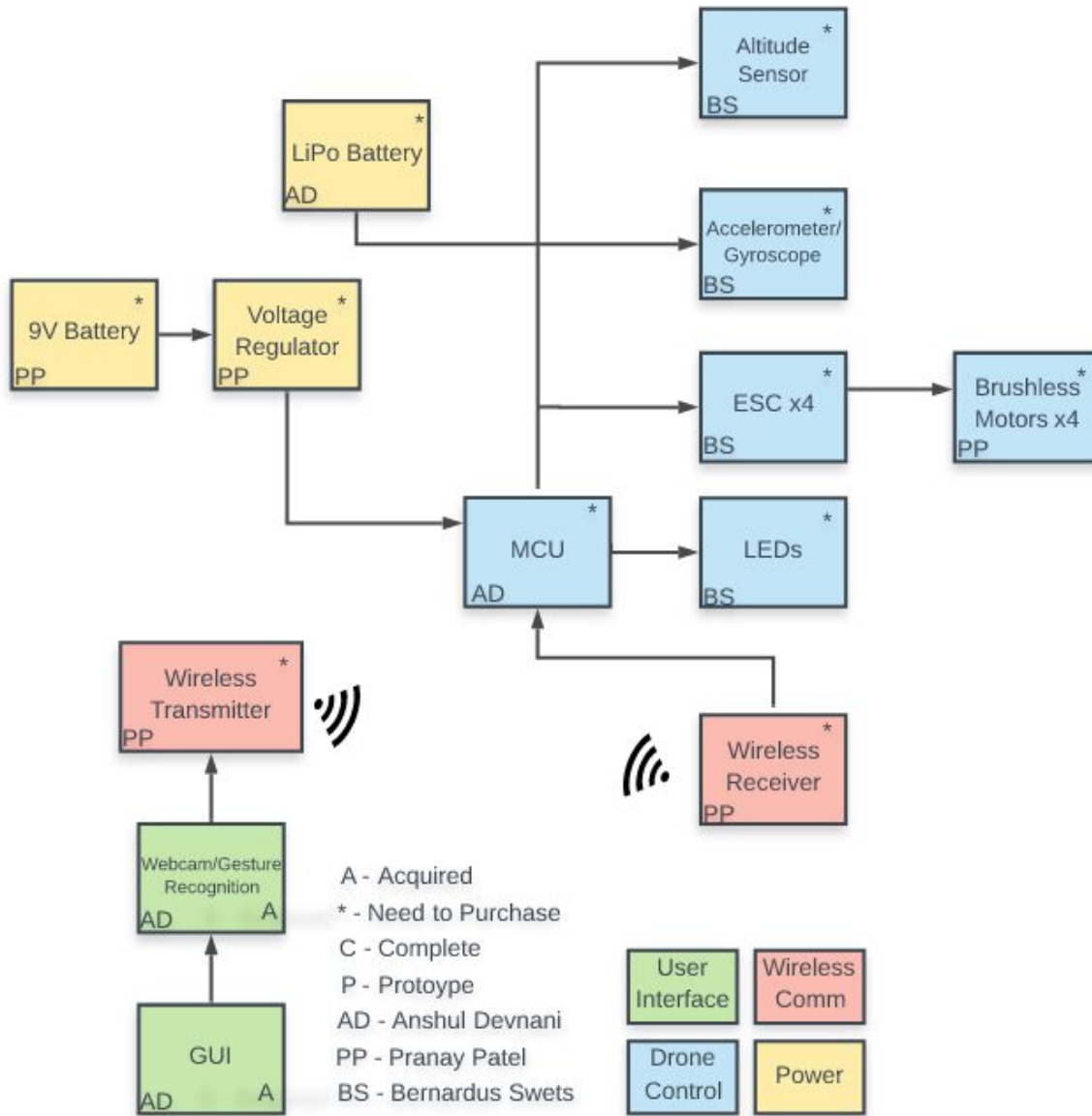


Figure 3 Block Diagram

7. Estimated Project Budget

The following table 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
Miscellaneous Components	\$200.00
Total:	\$525.00

Table 3 Proposed Budget

8. Project Milestones

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.

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	Begin 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 4 Milestones

9. Project Standards

Table 3 defines the project standards for the project.

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