

# Gesture Operated Drone

Drones have become increasingly popular over the last decade. Every year their abilities are rapidly increasing and we wanted to do our part to add to this continuously growing field. With the knowledge we have obtained throughout our studies we challenged ourselves to develop a drone that is strictly controlled by human hand motions/signals. 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, Python development for a GUI, and machine learning to build a useful and sound product.

Because we, as a team, believed that drones can sometimes be difficult to operate, we wanted to offer the ability to control them with a much simpler interaction. This allows for us to build in automatic stabilization and maneuvers 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 wanted 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 mimic all of the 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 our prototype build process was completed, we will be able to refine that process to reduce time to build and increase cost efficiency. 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.

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. We plan to develop the flight controller ourselves to apply the corrective features of a closed loop system. As this is a self-funded project and there are currently no sponsors, our goal was to make this project as low cost as possible. This ensures that the product can be supremely accessible to the public and can be improved upon moving past our first prototype.

Throughout this document we summarize how the drone was built and the motivation behind each component

we used to build it. Consisting of four main components, we will have a graphical user interface, the drone flight controls, power system, and the communication network all working in harmony to control the drone.

## Project Design

Information in the following sections outline our approach in designing our Gesture Operated Drone prototype.

## Gesture Recognition Neural Network


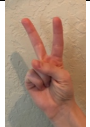


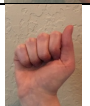
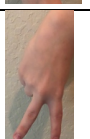

Building a good gesture recognition application was an immensely important aspect of this project. Failure to create a robust recognition application would've not only lead to wrong gesture recognition predictions but also lead to drone control issues. In our project, the classification task at hand was categorizing different hand gestures in real time. Our solution to this classification problem allowed for the computer to learn the physical characteristics of a set of different hand gestures and was able to accurately predict a newly inputted hand gesture, in real time.

Because classification problems using Machine Learning and Neural Networks are extremely computationally expensive, since it uses matrix multiplication and convolution, we utilized our MacBook Pro's GPU to do the computation. We had accounted for the possibility of the MacBook not being enough, and so had kept a backup option of using Google Cloud, but we found the MacBook's to be enough for a decent response time.

As for the implementation of the Machine Learning application, we decided to write it in Python, utilizing the Keras library to make our code as minimal and readable as possible. Keras is a library that is built on top of TensorFlow, which is a Machine Learning library that has been developed by Google. Additionally, we chose Keras because our dataset is relatively small for this application, limiting it to only thousands of images as compared to larger scale Machine Learning applications using millions of images.

## Gestures

As for the actual gestures, the below table shows the commands we will be providing to the Neural Network:

Dictionary Value	User Action	Result
0000	No Gesture	Hover in place/autolevel
0001		Thrust Upwards
0002		Drone flies forwards
0003		Drone flies to the left
0004		Drone flies to the right
0005		Drone lands in current position
0006		Drone flies backwards
0007		Thrust down

For this project, our dataset will consist of thousands of different hand gesture images. 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. It will be inefficient and virtually impossible to train a model taking into account all skin color and environment variables.

Our plan was to simply extract the hand gesture from each frame before sending it through our neural network, which means that we used the idea of background subtraction. For this, we set a background, and that causes everything except the hand signal to appear as black. In order to account for varying skin colors, we used the binary thresholding from the background subtraction to make the subject of the image filled with white pixels, making a silhouette of the hand gesture.



For training, we used OpenCV. OpenCV is an open source computer vision library that can be used to interface with the computer webcam. The webcam records a certain number of frames and the utility loads all the captured frames into directory associated with the hand gesture being recorded. For our project, we wanted 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. In terms of error rate/accuracy, the Le-Net-5 architecture was able to achieve an error rate below 1% on certain datasets.

Before the training began, the input data set was 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 our project, the dataset consisted of about 1000 images of each hand gesture for a total of around 8000 images. Testing of the Neural Network itself occurs at the end of each epoch (number of times the model cycles through the data). The model first trains itself using the training dataset and immediately after the model tests itself on the validation data. After each epoch, metrics were calculated to show how well the model is responding to the training and testing.

## Graphical User Interface

The graphical user interface is what the user interacts with to communicate with the drone. The goal was to keep this interface functional and user-friendly. The webcam window consists of a real time feed of the webcam and takes 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 feedback/reading window pane will give the user a visual representation of the prediction result after been passed through the Neural Network model. The log window pane serves as a textual representation of all actions being performed and is located at the bottom of the GUI. Essentially, every action taking place in the system should be recorded in the log window. We decided to implement the idea of using logs to ease the debugging process and to know the commands being sent throughout the system.

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. Tkinter 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.

## Wireless Communication

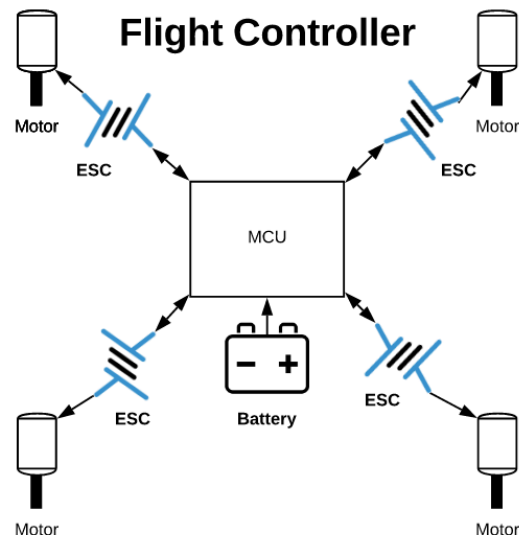
Because a drone is controlled by RC, we decided to go with Bluetooth as the form of communication because of how commonly it is used for small projects on this scale, how familiar it is to setup, and the fact that we do not need to send large packets of data. Bluetooth is completely standardized and has been continually optimized and updated. In regard to our particular implementation of Bluetooth, we will be pairing the drone to the laptop computer that is reading the hand signals. In our solution, we are pre-programming the Bluetooth ID of the HC-05 module attached to the PCB so that we have no issue connecting in our prototype. We decided that we only will be communicating one byte of data for the command to be sent, so that it can be sent quickly and very often. This is because we tested and plan to showcase in a space with relatively few interferences, and no concrete walls in between or obstacles of the like.

The previous table shows the values that we are utilizing for our drone configuration, meaning the laptop computer will be sending these values to the drone after the CNN computation has

completed and the drone will be reacting accordingly:

## Drone Hardware Design

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.

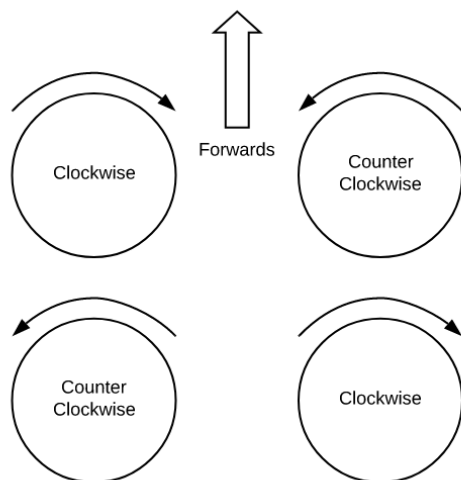


**Figure 1 Drone Design**

We chose to use a fiber reinforced plastic drone frame. Carbon fiber is a popular alternative used widely in commercial drones. Our reason for avoiding carbon fiber was the decreased cost. Carbon fiber drones are also notorious for blocking radio waves. We avoided this issue by utilizing the reinforced plastic frame. This frame was strong enough to

withstand the impact of minor crashes we experienced indoors and was fairly light weight. When assembling the drone, we ensured all the components are properly balanced and tightly secured. We experienced in testing, even the slightest loose component will significantly increase the vibrations across the drone. A poorly aligned drone can lead to shaking. Shaking especially threw the gyroscope off, which is a key part of a stable flight. This is one of many things we did to avoid vibrations. Our other solutions are also explored in this section.

Of the four motors at the end of each drone arm, the direction of the spin is extremely important. Shown in **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 2**  
**Motor Orientation**

Motors rely on electronic speed controller to function properly. Essentially the electronic speed controller, abbreviated ESC, communicates between the motors and the flight controller.

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. It is important that we position our ESCs in a way where they will be exposed to open air to prevent them from overheating.

Brushless motors were the best option for our drone because they have a longer life-span than brushed motors. Brushless motors use magnetic power which waste less energy and is more reliable. 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. 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. The direction of the propeller is extremely important. The leading edge much be facing the direction of the rotation. Having a misplaced propeller will result in no lift.

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. 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 laser sensor, the Bluetooth module, the gyroscope/accelerometer, and the flight controllers). There are two separate power sources. The PCB has a 9V battery that powers the microcontroller and all the sensors. This power is divided into 5V and 3V which is then routed to the specific components. The lithium polymer battery at the base of our drone is what is used to connect to the ESC. Our drone frame had a built in power distribution board that split the battery equally and powered the ESCs. We also had two separate wires coming from the battery that were used to measure the battery life of the battery. Once the battery life reached a certain point, an indication would inform us, and we knew to change the battery. The battery voltage was also live fed into the GUI, giving us a clear representation of how much time we had left.

For the gyroscope, we will be using the MPU-6050. Its very low current draw of 0.1 milliamps is ideal for our implementation because we are trying to be as preservative as possible with our power.

To power our drone, we will be using a rechargeable Lithium Polymer (or LiPo) battery. We will be using this particular type of battery because they are much more efficient and powerful.

Our goal with mounting our flight controller to the drone frame, was to dampen the vibrations as much as possible. For starters we positioned the flight controller as central as possible. At the center of the drone is where the vibrations are the least. We also avoided using any metal to mount the controller.

Using anti vibration foam we soft mounted our flight controller to the drone.

We also wanted to tackle the drone's vibration directly at the source. Motors are the root cause of vibrations through the drone. With the use of Silicon TPC mounting pads, we did our best to absorb the motors vibrations. Between the soft mounted flight controller and the anti vibration pads under the motors, we had a sound design, that allowed the gyroscope to read accurate angles.

## **Drone Software Design**

Flight controllers control the speed of all the motors, dissect commands from the user and balance the drone. 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. Instead of purchasing a preprogrammed flight controller, using the Arduino platform, we developed our own. The commands will be received and directly converted into the desired reaction.

The microcontroller is the brain of our drone and connects the ESC to the user giving them control of the device. The microcontroller we used is 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.

Calibration of the ESC is extremely important. All the motors need to be in unison and spinning at the same speeds. All the calibration is programmed through

the Arduino platform. 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.

PID is an acronym for Proportional Integral and Derivative. In a closed loop system these values were 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.

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. There is not a combination of PID value that is universally correct. Every motor is different, and every drone will have its own unique inconsistencies. 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.

Initially our gyroscope had some issues with drift. We used a Kalman filter to eliminate this drift. On top of this, it also helped handle gyro inconsistencies caused by drone vibrations. The filter was customizable and we found the best values to eliminate drift and stabilize the gyroscope. On top of this we were also able to desensitize the gyroscope.

The initial flight testing was 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.

## PCB

In order to get our PCB made, we created the schematic we needed in Eagle software and uploaded the gerber file to JLCPCB.com, from which we were able to order 5 PCBs from. We did run into issues with our first PCB, in which we were unable to load the bootloader onto the ATmega328p chip that was embedded on the PCB, and so we swapped out the embedded chip and opted to go with drop sockets for us to remove the ATmega328p from the Arduino we were using to program it, and placing it onto the PCB for use afterwards.

## Conclusion

All of the above components were combined to create our Gesture-Operated-Drone, which meets several industry standards and is on par with a legitimized engineering project. We found that this project continually challenged us however we were absolutely able to realize each and every part of our 4 year college career in the ECE department was utilized in the making of this project. Even though we all had our own specialties, most of the project was performed as a group. We gained experience working as a team and troubleshooting together. As we hit hurdles, being able to brainstorm amongst each other allowed for us to reach solutions we may not have come up with by ourselves. We all learned a lot from this project and will use the skills in our future professional endeavors.



## Biography



Pranay Jay Patel will graduate with his Bachelor of Science in Computer Engineering in December 2019. He currently works for Darden Restaurants as a Software Engineering intern. He plans on working for a large tech company for a few years before going back to get his Master's degree in Business Administration, with the intention to start his own company in the future.



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.



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.