

Exploratory Project Report

Project Aim

To make a real time hand gesture controller for drone which is robust in any environmental changes.

Submitted By:

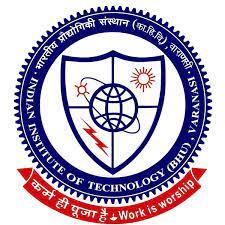
Yash Upadhyay (19095111)

Yatharth Bhargava (19095112)

Under the Guidance of

**Dr. Sanjeev Sharma**

**(Department of Electronics Engineering)**



**Department of Electronics Engineering**

**IIT(BHU) Varanasi**

**CERTIFICATE**

This is to certify that this project report “**To make a real time hand gesture controller for drone which is robust in any environmental changes**” is submitted by **Yash Upadhyay** and **Yatharth Bhargava** who carried out the project work under the supervision of **Dr. Sanjeev Sharma.**

We approve this project for submission of the Exploratory Project, IIT(BHU) Varanasi.

**Signature of Supervisor**

**Dr. Sanjeev Sharma**

Department of Electronics Engineering

IIT(BHU) Varanasi

**Acknowledgement**

It give us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide, **Dr. Sanjeev Sharma,** for his valuable guidance, encouragement, and help for accomplishing this work. His useful suggestions for this whole project are sincerely acknowledged.

We would also like to express our sincere thanks to all others who helped us directly or indirectly during this project work.

**Students Name:**

Yash Upadhyay – 19095111

Yatharth Bhargava – 19095112

**ABSTRACT**

This report presents the design, and prototype implementation and testing of a hand gesture-controlled drone such that one doesn’t require a controller .

This project went in 3 phases:

**Phase One :**

Studying about various gesture detection algorithms available which provide more robust results regarding hand tracking and its posture detection and also applying one.

**Phase two:**

Selecting drone simulation software such that the environment represents more realistic drone . Also, programming its basic movement control system.

**Phase three:**

Connecting drone simulation with hand gesture recognizer and executing proper movement command for each gesture.

**INTRODUCTION**

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.

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. 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.

We are proposing a drone that is entirely driven with hand gestures which works fine even in some bad lighting condition.

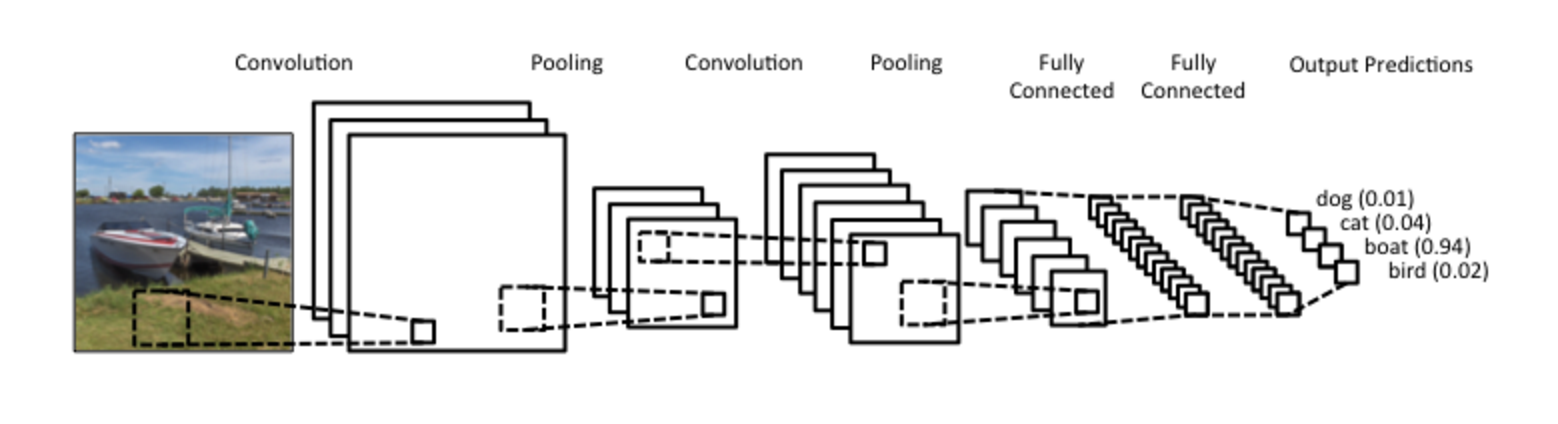
Our solution will allow the user to only need to use one hand to control the drone, as long as that hand stays in the correct field of vision. This will allow for freedom of motion for their alternate hand, which is something that is overlooked often when it comes to drones.

As a self-funded project, we are not able to work on real drone components but we have tried our best to cope up with this problem by proposing and testing every component as we can.

Report on Presentation Investigation

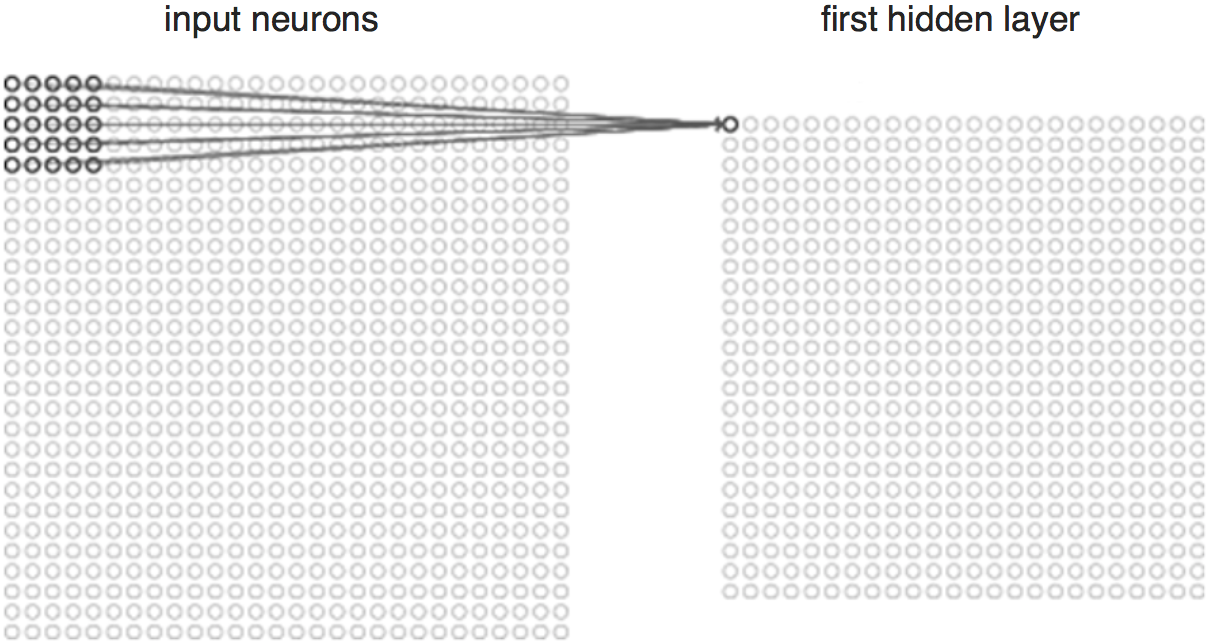
1. Convolutional Neural Network:

Most of the Computer Vision tasks are surrounded around CNN architectures, as the basis of most of the problems is to classify an image into known labels. Algorithms for object detection like SSD(single shot multi-box detection) and YOLO(You Only Look Once) are also built around CNN.



Artificial neural networks were great for the task which wasn’t possible for Conventional Machine learning algorithms, but in case of processing images with fully connected hidden layers, ANN takes a very long time to be trained. Due to this, CNN was used to first reduces the size of images using convolutional layers and pooling layers and then feed the reduced data to fully connected layers.

**Convolution layer:**

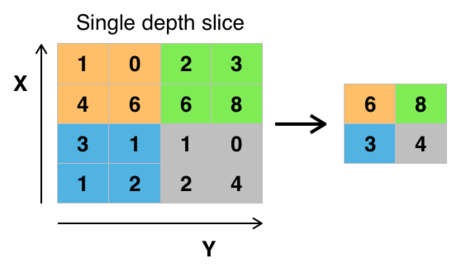


To perform convolution operation, a filter (A smaller matrix)is used whose size can be specified. This filter moves all over the image matrix and its task is to multiply its values by the original pixel values. All these multiplications are summed up to one number at the end. The filter moves further and further to its right by n units(can vary) performing a similar operation. After passing across all the positions, a matrix is obtained which is much smaller in size than the input matrix.

## Nonlinear layer:

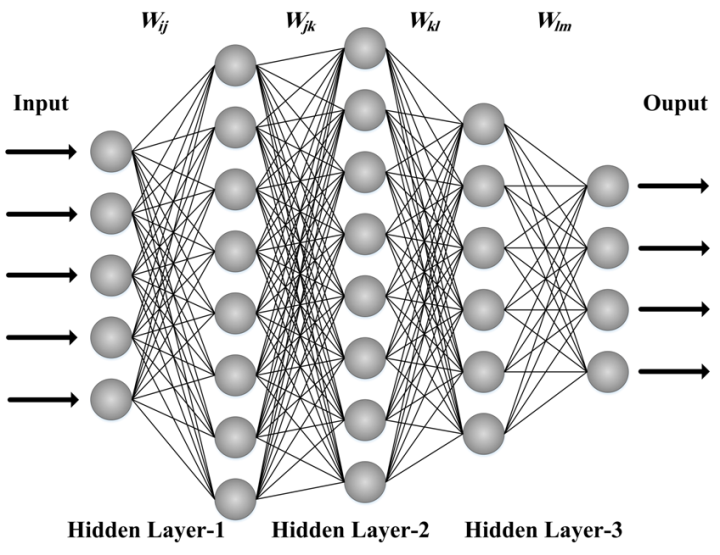
This layer is added after each of the convolution layers. It uses an activation function to bring non-linearity to data. Non-linearity means that the change of the output is not proportional to the change of the input. We require this nonlinearity because if the network was linear, there would be no point in adding multiple layers (multiple linear layers are equivalent to a single layer). By increasing the nonlinearity, a complex network is created to find new patterns in the images. The activation function here can be Rectified Linear Unit (ReLU), Tanh or any other nonlinear activation function.

## Pooling Layer



Pooling layer is used to further downsize the matrix. The most common form of a pooling layer is with filters of size 2×2; applied with a stride of two down samples at every depth slice in the input along both width and height, discarding 75 per cent of the activations. Pooling layer is generally used to select the most important pixels by using Max pooling function which only selects the highest value pixel present in the filter. This reduces the amount of computation required for training, hence reducing the time taken for training the neural network significantly.

**Fully Connected layers:**

In Fully Connected layers, every neuron from one layer is connected to every neuron in another layer. Principally, FC acts similar to as the traditional neural network, Multi-Layer Perceptron (MLP). However, the only difference is that the inputs would be in the shape and form created by the previous stages of a CNN.

1. Hand Detection in Mediapipe :

Mediapipe offers cross-platform, customizable ML solutions for live and streaming media.

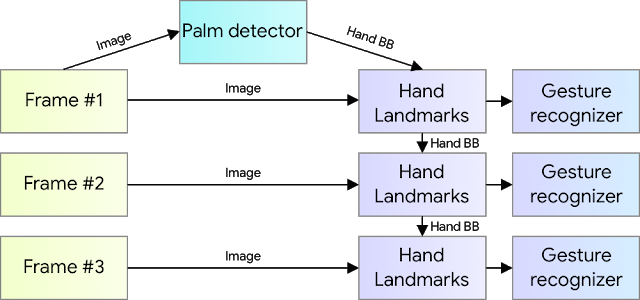
While coming naturally to people, robust real-time hand perception is a decidedly challenging computer vision task, as hands often occlude themselves or each other (e.g., finger/palm occlusions and handshakes) and lack high contrast patterns.

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame.

Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, this method achieves real-time performance on a mobile phone, and even scales to multiple hands.

MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g., rotations, translation and scale) and instead allows the network to

dedicate most of its capacity towards coordinate prediction accuracy. In addition, in the pipeline the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection invoked to re-localize the hand.

**Models**

**Palm Detection Model:**

Detecting hands is a decidedly complex task: our model has to work across a variety of hand sizes with a large-scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands.

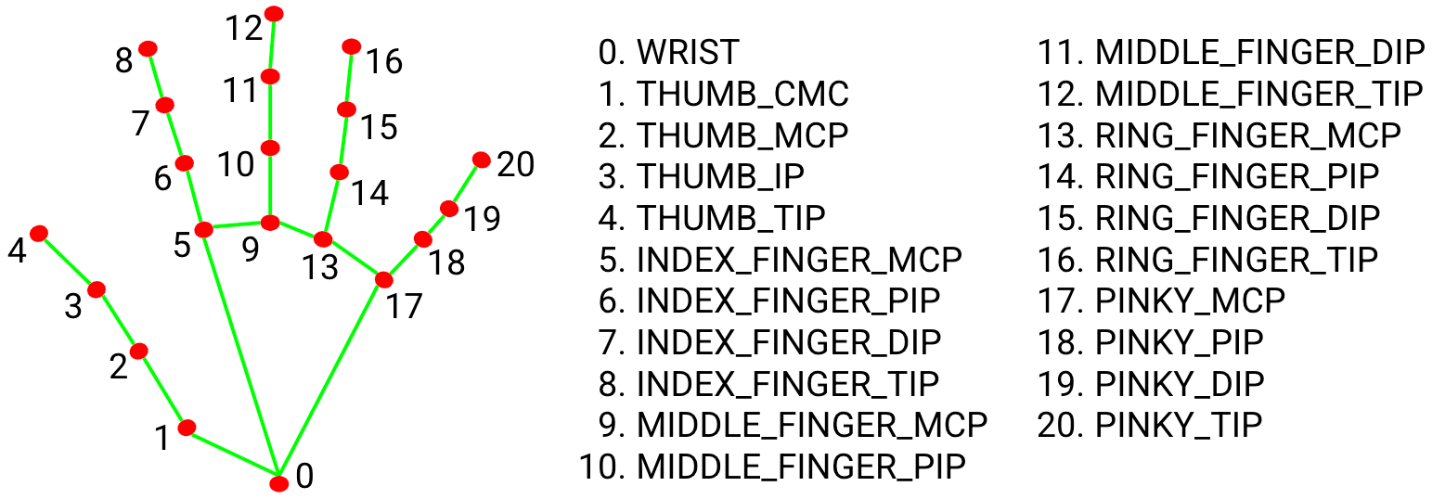
Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.

Mediapipe method addresses the above challenges using different strategies. First, a palm detector is trained instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, the focal loss is minimized during training to support a large amount of anchors resulting from the high scale variance.

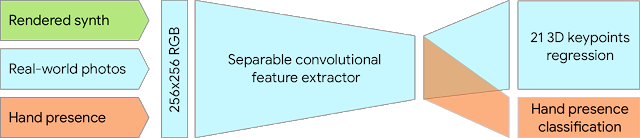
With the above techniques, we achieve an average precision of 95.7% in palm detection. Using a regular cross entropy loss and no decoder gives a baseline of just 86.22%.

**Hand Landmark Model:**

After the palm detection over the whole image the subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, about ~30K real-world images have manually annotated with 21 3D coordinates, as shown below .To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, a high-quality synthetic hand model renders over various backgrounds and map it to the corresponding 3D coordinates.

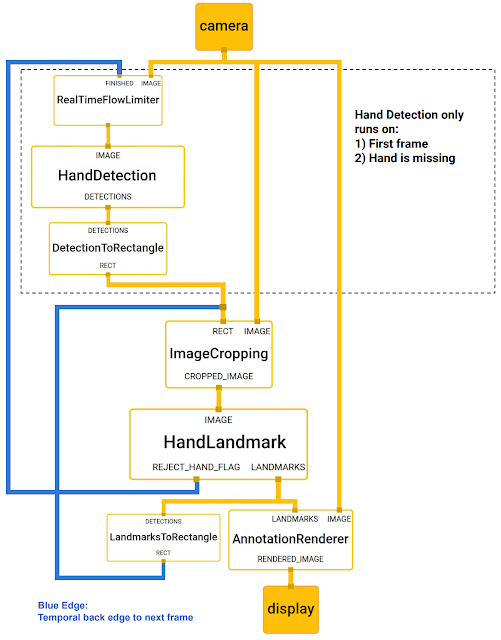
However, purely synthetic data poorly generalizes to the in-the-wild domain. To overcome this problem, a mixed training schema is utilized. A high-level model training diagram is presented in the following figure.



**Implementation via MediaPipe:**

With MediaPipe, this perception pipeline can be built as a directed graph of modular components, called Calculators. Mediapipe comes with an extendable set of Calculators to solve tasks like model inference, media processing algorithms, and data transformations across a wide variety of devices and platforms. Individual calculators like cropping, rendering and neural network computations can be performed exclusively on the GPU. For example, we employ TFLite GPU inference on most modern phones.

Our MediaPipe graph for hand tracking is shown below.



**OUR APPROACH:**

* Hand Tracking and Landmark Detection
* Control Logic to Control Drone
* Controlling Drone in Simulator Using Script in Simulator

**CONTROL LOGIC**

* We have implemented 2 logics to control drone.
  + Finger – count Approach
  + Hand Position Based Approach
* Commands from this logic are fed into the script in simulation software, where the drone maneuver accordingly.

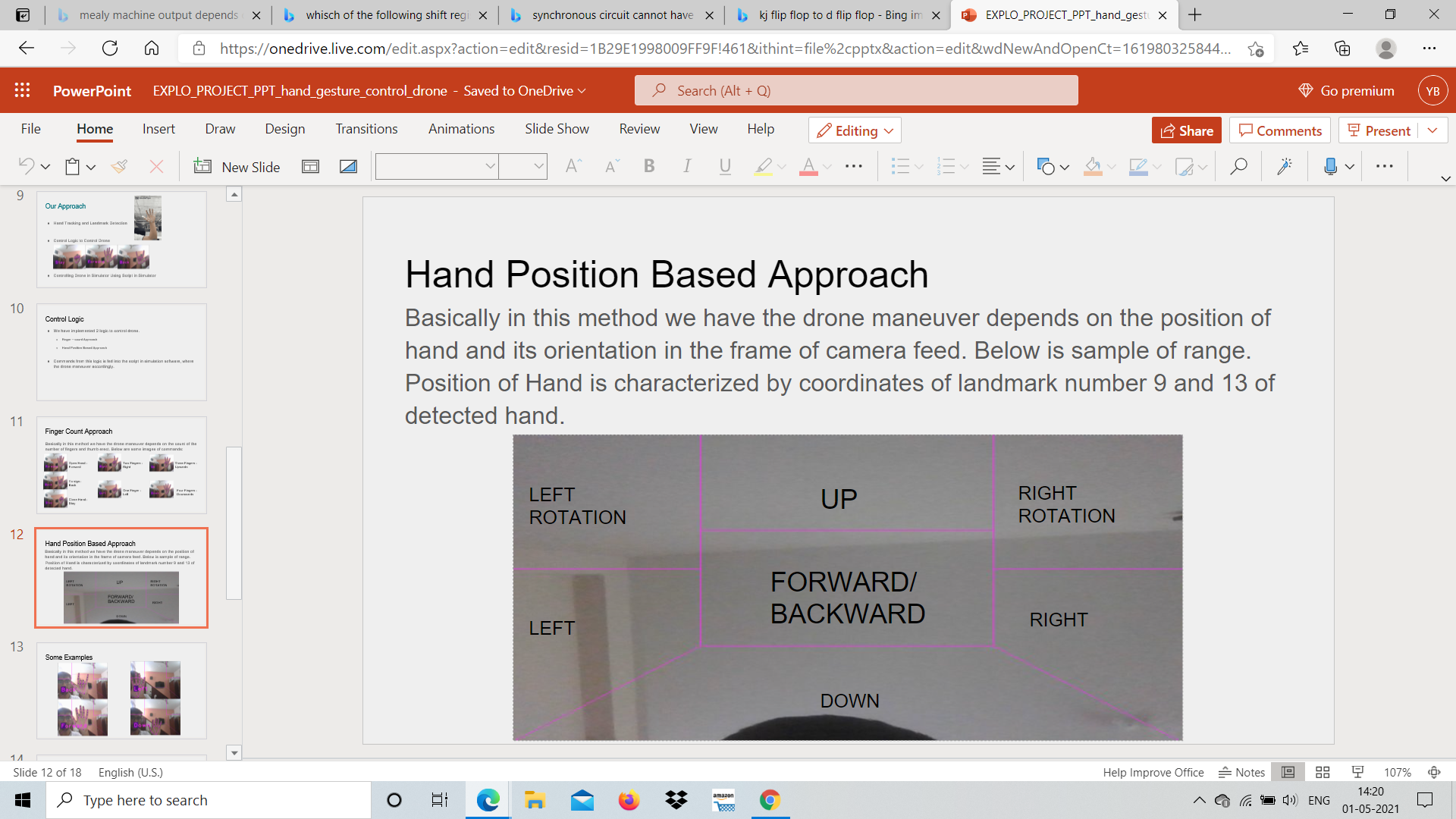
1. Finger Count Approach

Basically, in this method we have the drone maneuver depending on the count of the number of fingers and thumb erect. Below are some images of commands:

|  |  |
| --- | --- |
|  | Open Hand:  Forward |
|  | Two Fingers:  Right |
|  | Three Fingers:  Upwards |
|  | Yo sign:  Back |
|  | One Finger:  Left |
|  | Four Fingers:  Downwards |
|  | Close Hand:  Stay |

1. Hand Position Based Approach

Basically, in this method we have the drone maneuver depending on the position of the hand and its orientation in the frame of camera feed. Below is a sample of range. Position of Hand is characterized by coordinates of landmark number 9 and 13 of the detected hand.



Some Examples:

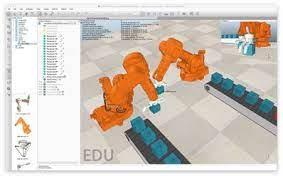
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Drone Simulation software**

For drone simulation we searched for various simulators and came across some of the versatile and mostly used once e.g., Webots, Gazebo, Coppeliasim etc.

We In this project used Coppeliasim due to following reasons:

1. It is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS, a remote API client, or a custom solution.
2. Its compatibility with both Windows and linux.
3. Multiple language support for controllers.
4. Fast and lightweight.
5. Small size and ease of installation.
6. Its large support base and proper documentation.

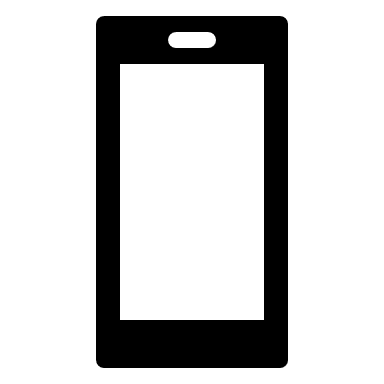


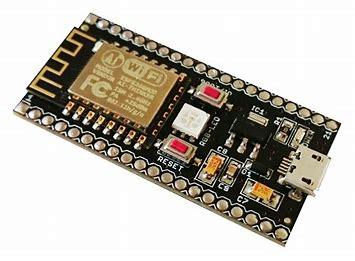
1. **Proposed Complete Structure with real drone**

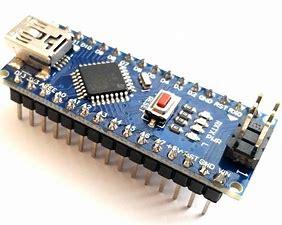
Using mediapipe we can implement hand recognition using our android phone after which it will send a motion command to the drone using WIFI module.

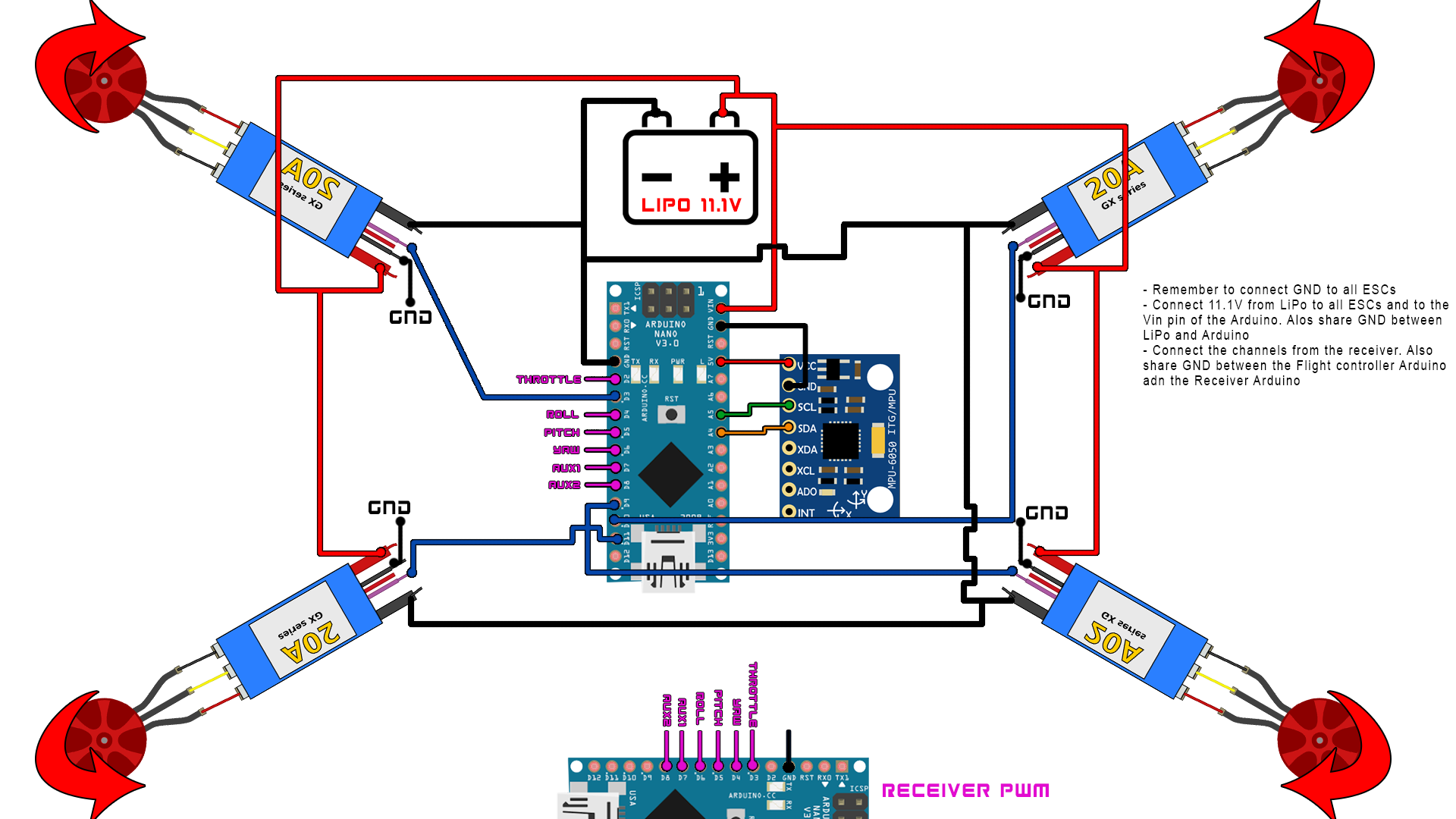
List of all the components required :

1. Smart Phone .
2. Node MCU (ESP8826)
3. Arduino microprocessor
4. Drone basic kit







**Drone Control using Arduino:**

**Results and Discussions**

Machine Learning and Computer Vision-based method for drone control is very simplistic and beginner friendly also it adds new features to drone such as one-handed control and performing very touch stunts just using simple gestures. But Practical application and simulation testing are two different systems.

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.

**Conclusion**

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.

**References**

1. <https://mediapipe.dev/>
2. <https://www.coppeliarobotics.com/>
3. <https://www.researchgate.net/publication/324485264_Hand_Gesture_Controlled_Drones_An_Open_Source_Library>
4. <https://medium.com/alumnaiacademy/introduction-to-computer-vision-4fc2a2ba9dc>
5. <https://www.ece.ucf.edu/seniordesign/su2019fa2019/g03/Documents/GOD_Senior_Design_1Final_Document_Group3.pdf>
6. [Sensors | Free Full-Text | Real-Time Human Detection and Gesture Recognition for On-Board UAV Rescue (mdpi.com)](https://www.mdpi.com/1424-8220/21/6/2180)

**GitHub Repository of our Work**

* [**https://github.com/Yashupadhyay603/Hand\_gesture\_drone**](https://github.com/Yashupadhyay603/Hand_gesture_drone)