

OBJECT TRACKING USING DRONE FOR SMART AGRICULTURE

A BTP Report

By

**Kamarthi Litheesh Kumar, S20190020218
TU Surya Sathvik, 20190020254**



**INDIAN INSTITUTE OF INFORMATION
TECHNOLOGY SRI CITY**

Date 20/12/2022

Final Report



INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**OBJECT TRACKING USING DRONE FOR SMART AGRICULTURE**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2022 to December 2022 under the supervision of Dr. Y Raja Vara Prasad, Indian Institute of Information Technology Sri City, India.

I have not submitted the matter presented in this report for the award of any other degree of this or any other institute.

Signature of the student with date

K.Litheesh Kumar
20/12/22

Kamarthi Litheesh Kumar, 20/12/2022

Surya
20/12/22

TU Surya Sathvik, 20/12/2022

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Reef

Signature of BTP Supervisor with date

Dr. Y Raja Vara Prasad

ABSTRACT

Owing to effective and flexible data acquisition, unmanned aerial vehicle (UAV) has recently become a hotspot across the fields of computer vision (CV) and remote sensing (RS). Inspired by the recent success of deep learning (DL), many advanced object detection and tracking approaches have been widely applied to various UAV-related tasks, such as environmental monitoring, precision agriculture, and traffic management. This project provides a comprehensive survey of the research progress and prospects of DL-based UAV object detection and tracking methods. More specifically, we first outline the challenges, and statistics of existing methods, and provide solutions from the perspectives of DL-based models in three research topics: object detection from the image, object detection from the video, and object tracking from the video.

Open datasets related to UAV-dominated object detection and tracking are exhausted, and four benchmark datasets are employed for performance evaluation using some state-of-the-art methods. Finally, prospects and considerations for future work are discussed and summarized. It is expected that this project can facilitate those researchers who come from the remote sensing field with an overview of DL-based UAV object detection and tracking methods, along with some thoughts on their further developments.

Purpose: Computer vision in drones has gained a lot of attention from artificial intelligence researchers. Providing intelligence to drones will resolve many real-time problems. Computer vision tasks such as object detection, object tracking, and object counting are significant tasks for monitoring specified environments.

Methodology: In this project, a detailed literature review has been conducted focusing on object detection and tracking using UAVs concerning different applications. This study summarizes the findings of existing research papers and identifies the research gaps.

Contribution: Object detection methods applied in UAV images are classified and elaborated. UAV datasets specific to object detection tasks are listed. Existing research works in different applications are summarized. Finally, a secure onboard processing system on a robust object detection framework in precision agriculture is proposed to mitigate identified research gaps.

Index Terms - Object detection, object tracking, remote sensing, unmanned aerial vehicle, video.

Contents

1. INTRODUCTION	6
2. LITERATURE SURVEY	8
3. CONTRIBUTIONS	9
4. METHODOLOGY	10
5. RESULTS	28
6. WORK DONE FOR FINAL EVALUATION	34
7. CONCLUSION	34
8. FUTURE WORK	36
9. LIST OF ABBREVIATIONS	37
10. ACKNOWLEDGMENTS	38
11. REFERENCES	39

1. INTRODUCTION.

Object detection and tracking, as an important research topic in the field of remote sensing, has been widely investigated and applied to various civil and military tasks, such as environmental monitoring, geological hazard detection, precision agriculture, and urban planning. Traditional object acquisition methods derive mainly from satellites and manned aircraft. Normally, the two types of platforms run on a fixed rail or follow a predetermined path, or temporarily change the running route and hover according to a commissioned task, e.g., city planning and mapping, or performing object observation in a harsh and inhospitable environment.

However, the cost of satellites and manned aircraft, and the potential safety issues of pilots inevitably limit the application scope of such platforms. With the development of microelectronic software and hardware, navigation and communication technology renewal, and breakthroughs in materials and energy technology, the unmanned aerial vehicle (UAV) platform already an international research hotspot in remote sensing has rapidly emerged. A UAV remote sensing system is a high-tech combination of science and technology integrated UAVs, remote sensing, global positioning system (GPS) positioning, and inertial measurement unit (IMU) attitude determination means. It is a dedicated remote sensing system with the goal of obtaining low-altitude high-resolution remote sensing images. Compared with traditional platforms, UAVs make up for information loss caused by weather, time, and other limitations.

Computer vision is already achieved remarkable achievements due to advancements in deep learning algorithms, hardware requirements, and the availability of datasets. Object detection is the most investigation task done by researchers as it has many applications. The objective of object detection is to detect objects from a certain category (for example human beings, dogs, vehicles, motorcycles, or cats) in a picture and if any, return the area and extent of each instance of objects. It constitutes the backbone for solving complex and high-level computer vision tasks such as object tracking, segmentation, event detection, image captioning, scene comprehension, crowd monitoring, and activity recognition. Researchers started to overcome the challenge of developing general object detection systems that can detect categories of objects which match human ones. The field of object detection has been greatly enhanced. However, object detection in drone applications is yet to research. From surveillance to agriculture, all applications need accurate object detection to serve their purpose.

Precision agriculture is expected to see significant growth compared to other applications, as the use of UAVs is becoming one of the most crucial parts of managing farm tasks. Precision agriculture is a collection of methods for tracking crops, collecting data, and carrying out informed crop management tasks such as the optimal water supply, and the selection of effective pesticides. UAVs can assist farmers in a vast range of activities such as planning and analyzing crop plantation, and farm monitoring to assess crop growth and health. The benefits of using aerial services in agriculture favored the expansion of the use of fertilizers from the air in 1940 to other activities, such as top dressing. A drone manufacturing company called DJI developed drones by integrating sensors especially for protecting crops from insects or weeds.

Although drones are popular now, their history is long years back. It can be categorized based on flying speed, stabilizing its position, hovering, fixed or loitering, and flying environment. Different types of UAVs have different characteristics. Single-rotor UAVs can carry heavy payloads but it has mechanical complexity which leads to high cost. Multirotor UAV is a common type as its usage applies to both professionals and common people. It can hover or move along the given target.

However, changes in viewpoint and changes in scale make the object detection task in drones more challenging than traditional object detection. In a traffic surveillance scenario, drones record traffic from a bird's eye view. The advantage of this is to 100 m in height to record vehicle traffic. Object detection from a bird's eye perspective is more challenging than the front-parallel view for the following aspects,

- The dynamic transition of moving objects.
- Changes in aspect ratio and image scale.
- Abrupt camera motion.
- Severe perspective distortion.
- Motion blur.
- The high density of objects.
- Complex background.

In addition to these difficulties, object detection studies in aerial view are also confronted by the biased dataset problem. To avoid this problem, the dataset must be annotated which reflects real-world applications. Therefore, it is not unusual that the object detection models learned from traditional images are not appropriate for aerial images.

2. LITERATURE SURVEY

2.1. Small-Object Detection Algorithm:

Most state-of-the-art detectors are based on deep-learning methods. These methods mainly include two-stage detectors, one-stage detectors, and anchor-free detectors. Two-stage detectors first extract possible object locations and then perform classification and relocation. The classic two-stage detectors include spatial pyramid pooling networks (SPPNet), faster region-CNN (RCNN), etc. The one-stage detectors perform classification and positioning at the same time. Some effective one-stage detectors mainly include the single-shot multi-box detector (SSD), You Only Look Once (YOLO) series, etc. Anchor-free detectors include CenterNet, ExtremeNet, etc. These methods do not rely on predefined anchors to detect objects. In addition, some scholars have introduced transformers into the object detection field, such as detection with transformers (DETR) and vision transformer faster RCNN (ViT FRCNN), which have also achieved good results.

However, the detection objects of the general object detectors are multi-scale. They are not designed for small-object detection specifically. Small-object detection algorithms can be mainly divided into two kinds. One is to improve the detection performance of small objects with multiple scales in a video or image sequence. The other is to improve the detection performance of small objects with only a scale in an image. The improved detection methods of small objects with multiple scales mainly include feature pyramids, data augmentation, and changing training strategies.

In 2017, Lin et al. proposed feature pyramid networks (FPN), which improve the detection performance effect of small objects by fusing high-level and low-level features to generate multi-scale feature layers. In 2019, M. Kisantal et al. proposed two data augmentation methods for small objects to increase the frequency of small objects in training images. B. Singh et al. designed scale normalization for image pyramids (SNIP). SNIP selectively backpropagates the gradients of objects of different sizes, and trains and tests images of different resolutions, respectively.

The research object of Drones 2022, 6, 308 3 of 19 these methods is multi-scale objects, which cannot make full use of the characteristics of small objects. The approaches that only detect small objects with a scale are mainly of three kinds: designing networks, using context information, and generating super-resolution images. L. Sommer et al. proposed a very shallow network for detecting objects in aerial images. Small-object detection based on network design is few and immature. J. Li et al. proposed a new perceptual GAN network to improve the resolution of small objects to improve detection performance. In this paper, we focus on algorithms that are suitable for small-object detection in UAV reconnaissance images.

2.2. Object Detection Algorithm for UAV:

The object detection algorithms used in UAVs are mainly designed based on the requirements of the task scenarios. M. Liu et al. proposed an improved detection algorithm based on YOLOv3. The algorithm first optimizes the Res-Block and then improves the darknet structure by increasing the convolution operation. Y. Li et al. proposed a multiblock SSD (MBSSD) mechanism for railway scenes monitored by UAVs. MBSSD uses transfer learning to solve the problem of insufficient training samples and improve accuracy. Y. Liu et al. proposed multi-branch parallel feature pyramid networks (MPFPN) and used a supervised spatial attention module (SSAM) to focus the model on object information. MPFPN conducted experiments on a UAV public dataset named VisDrone-DET to prove its effectiveness. These algorithms are combined with UAV application scenarios but are all based on classic methods. They cannot work well when inferencing against untrained small objects

3. CONTRIBUTIONS

- **Kamarthi Litheesh Kumar (S20190020218)** - Written Arduino code, Python code for Object tracking and detection, and Website code for printing values.
- **TU Surya Sathvik (S20190020254)** - Written Python code for Drone path tracking, Hardware connections of Raspberry pi.

Problem Statement:

The object recognition and tracking of a specific object using a Tello require a method to communicate changes in the object's position as well as manipulation of the RBG values in a CV2 code. For object recognition, Tello will need to recognize the specific color of the object. The RBG values chosen for the object are unlikely to be exact, but it will need a specific range to be close enough in order for Tello to recognize the specific object. To track the object, the Tello will need to process that there is an offset in the original distance of the object from the center of the frame and react accordingly to refocus.

4. METHODOLOGY

In this project, We want to collect data without any loss for that we have node points where the total data from the sensors like Temperature sensor, Humidity sensor, Rainfall sensor, and Soil Moisture sensors will be calculated. From these sensors, data will be collected by ground Raspberry pi and then send to the cloud using the MQTT protocol. When a drone having raspberry pi reaches the range of ground raspberry pi, the data will be transmitted and the drone raspberry pi will send that data to the cloud. In the cloud, we are training the data and after identifying that the reading is out of thresholds then we can activate the process manually when the soil moisture is very low then we can turn ON the pump motor which can wet the soil and help us to get a good yield.

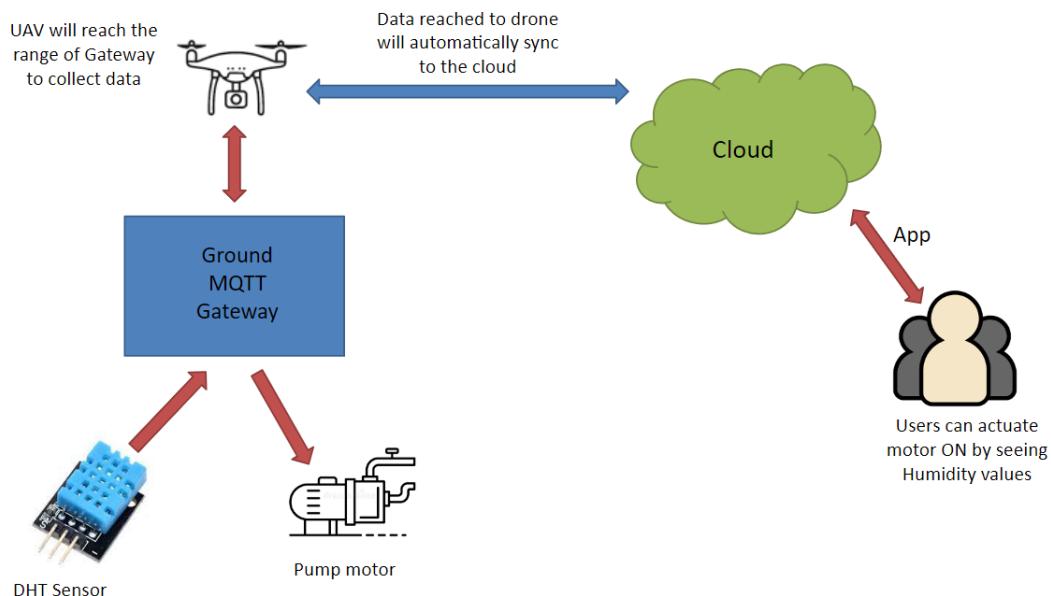


Figure1: Flow chart of ground Raspberry pi connections

4.1. Definition:

For the transmission of data we are using the MQTT protocol. MQTT(MQ Telemetry Transport) is a lightweight, publish-subscribe, machine-to-machine network protocol for Message queue/Message queuing service. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP.

4.2. Overview:

The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device (from a microcontroller up to a fully-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.

If a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message. A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for the selected topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic. This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher.

When a publishing client first connects to the broker, it can set up a default message to be sent to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker. Clients only interact with a broker, but

a system may contain several broker servers that exchange data based on their current subscribers' topics.

A minimal MQTT control message can be as little as two bytes of data. A control message can carry nearly 256 megabytes of data if needed. There are fourteen defined message types used to connect and disconnect a client from a broker, to publish data, to acknowledge receipt of data, and to supervise the connection between client and server. MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth.

4.3. MQTT broker:

The MQTT broker is a piece of software running on a computer (running on-premises or in the cloud), and could be self-built or hosted by a third party. It is available in both open-source and proprietary implementations. The broker acts as a post office. MQTT clients don't use a direct connection address of the intended recipient but use the subject line called "Topic". Anyone who subscribes receives a copy of all messages for that topic. Clients can subscribe to a topic from a single broker, and single client can register subscriptions to topics with multiple brokers.

Each client can both produce and receive data by both publishing and subscribing, i.e. the devices can publish sensor data and still be able to receive the configuration information or control commands (MQTT is a bi-directional communication protocol). This helps in sharing data and managing and controlling devices. A client can not broadcast the same data to a range of topics and must publish multiple messages to the broker, each with a single topic given.

With the MQTT broker architecture, the client devices and server applications become decoupled. In this way, the clients are kept unaware of each other's information. MQTT, if configured, can use TLS encryption with the certificate, username, and password-protected connections. Optionally, the connection may require certification, in the form of a certificate file that a client provides and must match with the server's copy.

In case of failure, the broker software and clients can automatically hand over to a redundant/automatic backup broker. Backup brokers can also be set up to share a load of clients across multiple servers on-site, in the cloud, or a combination of these. The broker can support both standard MQTT and MQTT for compliant specifications such as Sparkplug. This can be done with the same server, at the same time, and with the same levels of security.

The broker keeps track of all the session's information as the device goes on and off, in a function called "persistent sessions". In this state, a broker will store connection info for each client, topics each client has subscribed to, and any messages for a topic with a QoS of 1 or 2.

The main advantages of an MQTT broker are:

- ❖ Eliminates vulnerable and insecure client connections if configured to
- ❖ Can easily scale from a single device to thousand devices
- ❖ Manages and tracks all client connection states, including security credentials and certificates, if configured to
- ❖ Reduced network strain without compromising the security if configured to (cellular or satellite network)

Here we are using two NodeMCUs as end clients. The NodeMCU which is connected to the LED bulb acts as a Subscriber and another one which is connected to the DHT sensor acts as Publisher and Raspberrypi links both ends using the MQTT protocol to act as MQTT broker. The publisher wants to publish data sensed from the DHT sensor and LED will subscribe to the data from raspberry pi which tells the subscriber to turn it ON then LED will glow. These MQTT connections help us to transmit the data from one endpoint to another point.

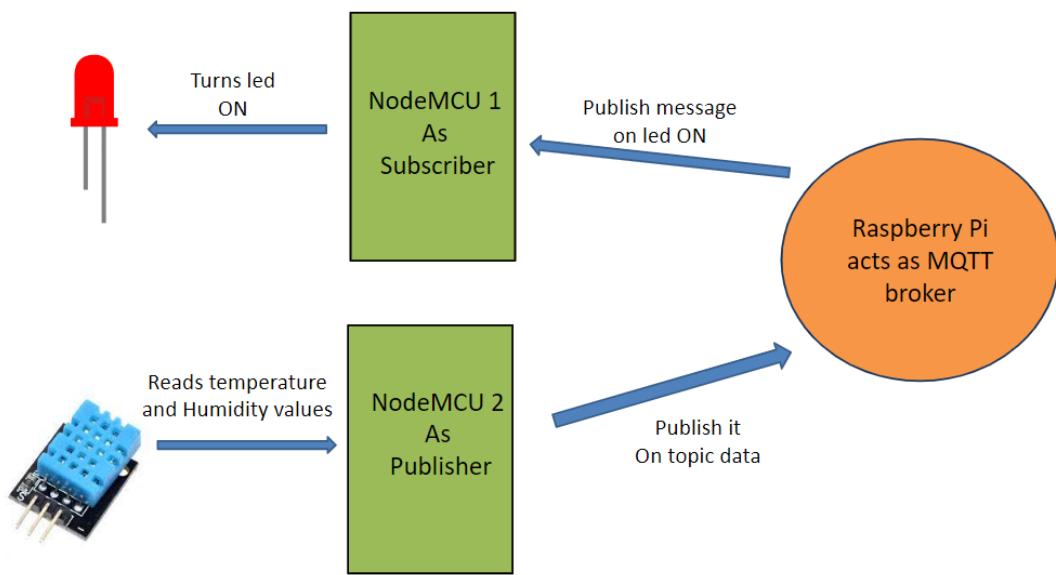


Figure2: Basic MQTT connection between 2 NodeMCUs

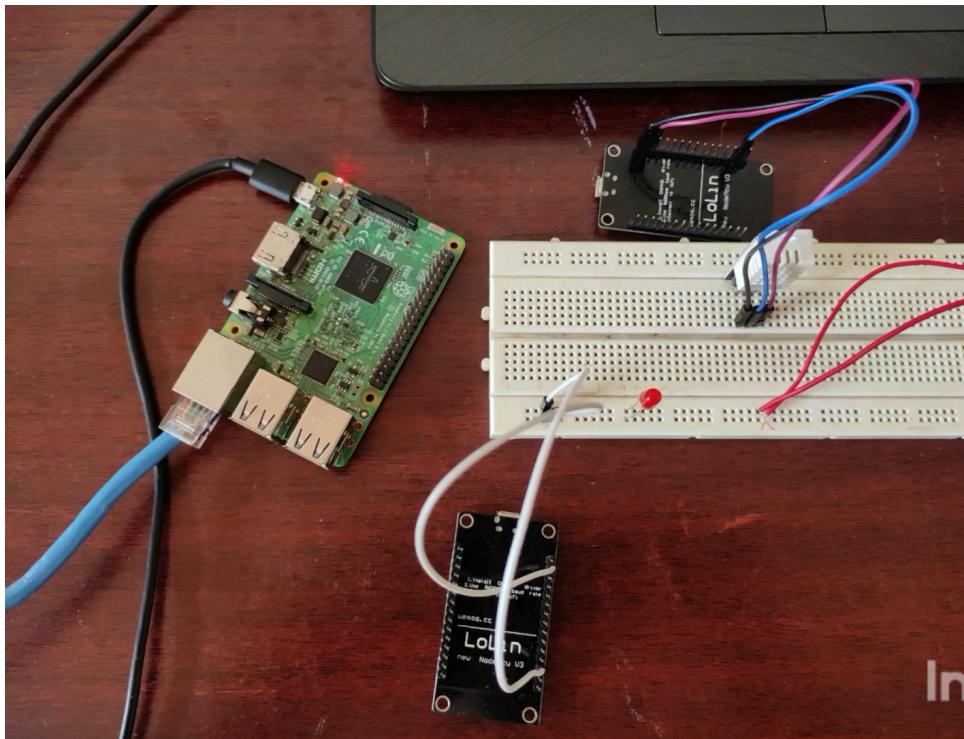


Figure3: Hardware of figure 2 MQTT connections

- 4.4. After that we extended the scale to litter higher, Now we are streaming the video taken from the raspberry pi camera module and can show photos taken by it. For that we have followed the below steps:

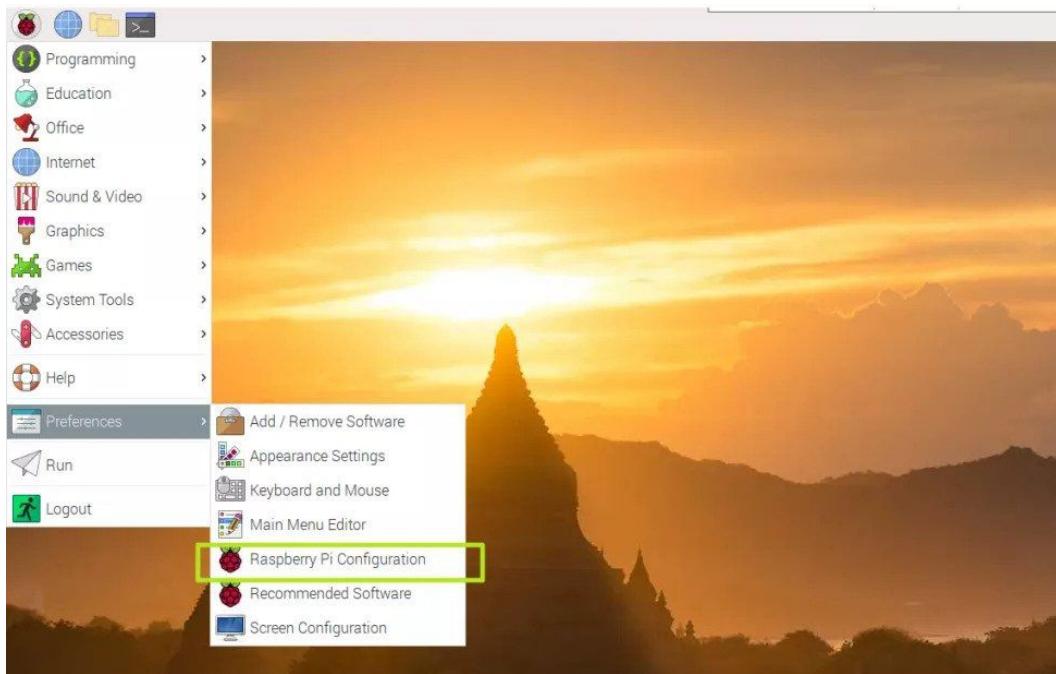
1. Open the camera port by gently lifting the plastic lock upwards.



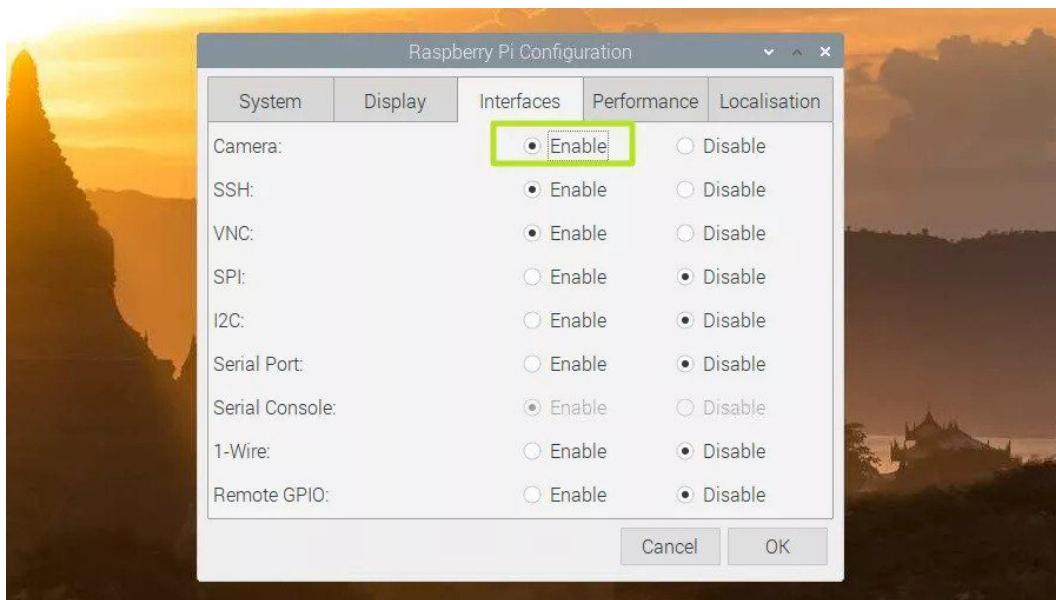
2. Insert the ribbon connector with the blue tab facing the USB / Ethernet ports.



3. Close the lock on the connector and give it a very gentle pull to make sure it is in place.
4. Power up your Raspberry Pi and then go to Preferences >> Raspberry Pi Configuration.



5. Click on the Enable button for the Camera found in the Interfaces tab.



6. Click Ok and reboot the Pi.
7. Open a Terminal and type the following command to take a quick picture to test the camera.

```
$ raspistill -o test.jpg
```

After five seconds have elapsed, an image will be taken and saved as test.jpg. Using the file manager check that the image is correct before moving on.

4.5. Testing a Stream

To start a stream we need to open a terminal and enter a rather long command. Ensure that your Raspberry Pi is connected to the network. For the best performance use an Ethernet cable, Wi-Fi will work, but you may see dropouts.

1. **Get the hostname of your Raspberry Pi.** Open a terminal and type in this command for your hostname. Make a note of the hostname. You may need to add “.local” to the end, depending on your network.

```
$ hostname
```

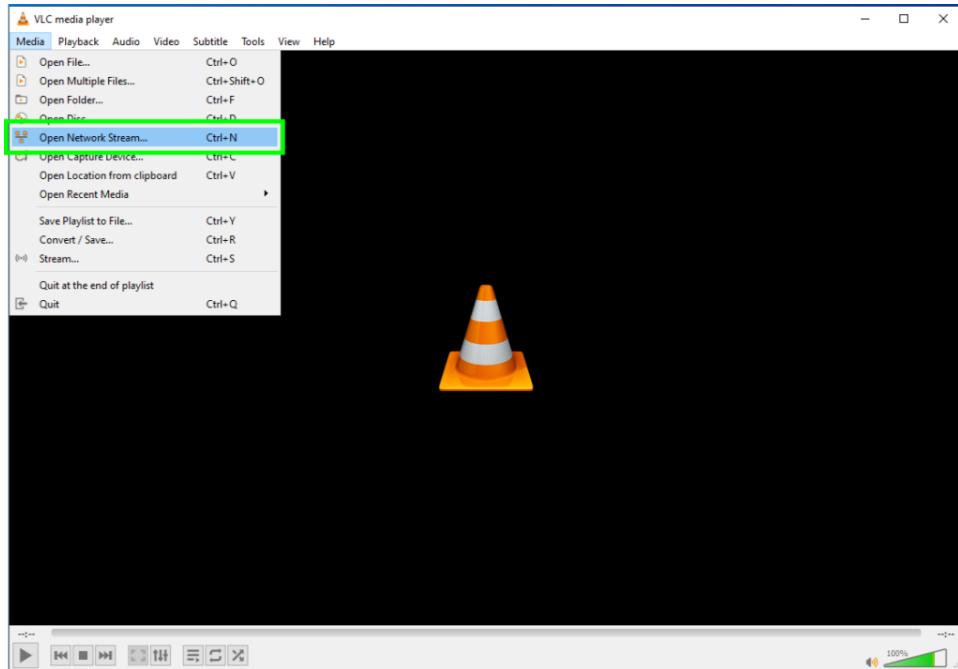
2. **Run the streaming command.** The one-line command to run a video stream live from the camera is rather long, so let's go through the command before we run it.

-o is our output, in this case, set to none.
-t is the length of the video clip, using zero will set this to infinite.
-w and -h are the width and height of the video, in this case, 800 x 600.
-fps is the frames per second for the video stream, a lower value should minimize dropouts.

Run this command in a terminal on your Raspberry Pi.

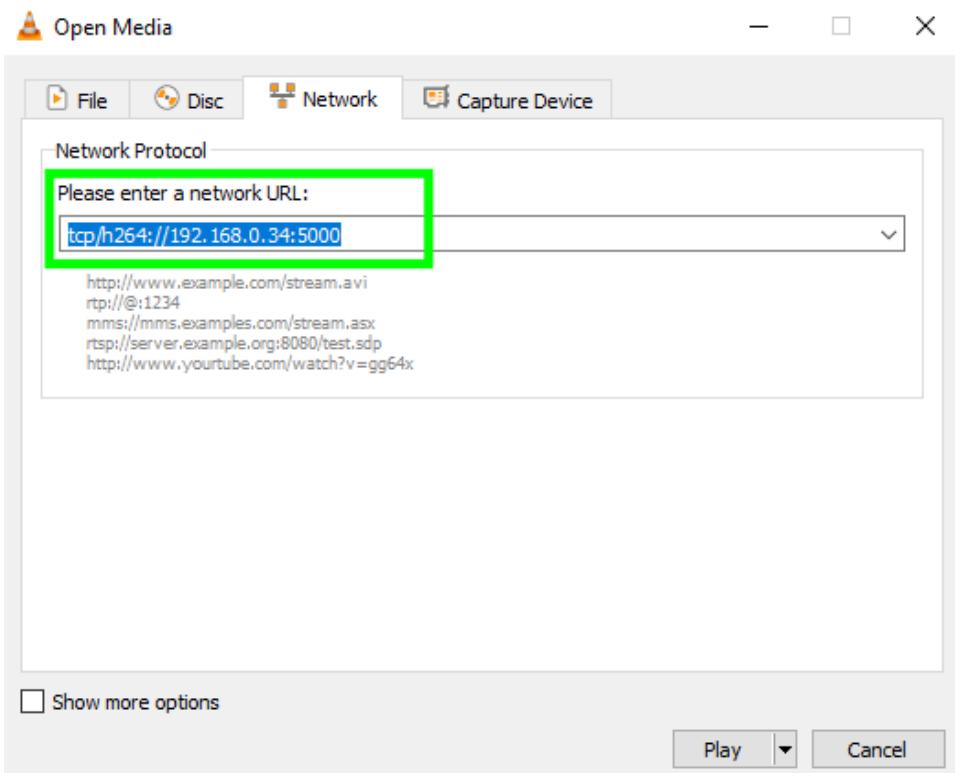
```
raspivid -o - -t 0 -w 800 -h 600 -fps 12 | cvlc -vvv
stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8080/}'
:demux=h264
```

3. On your Windows / Mac / Linux computer install VLC and then open VLC.
4. Go to Media >> Open Network Stream, or press CTRL + N.



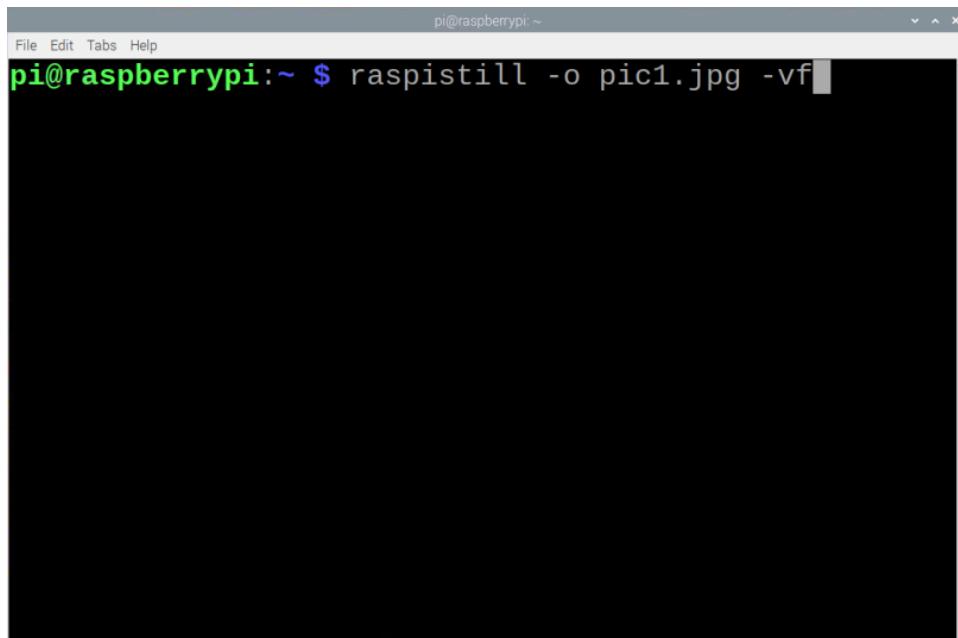
5. Enter the rtsp address and port of your Raspberry Pi. The rtsp address is your Raspberry Pi's hostname, prefixed with rtsp://, and the port was set to 8080. Click Play to start the stream. Here is our example rtsp address.

rtsp://raspberrypi:8080/



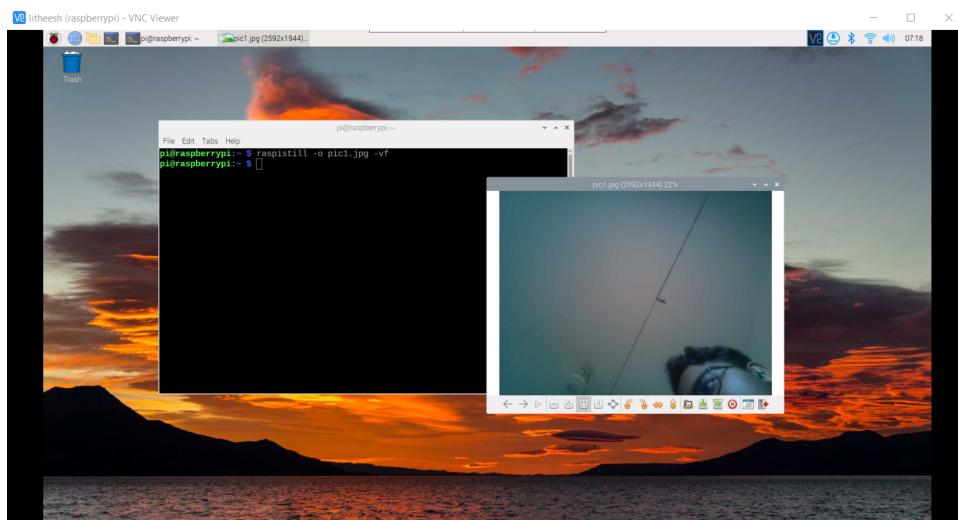
It will take VLC a few seconds to catch up with the stream, be patient. Soon you will see video streamed from your Raspberry Pi, with a delay of around 5 to 10 seconds.

6. To record a stream, click on Playback >> Record and then specify a filename. The recording will end when the stop button is pressed.



```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ raspistill -o pic1.jpg -vf
```

Command for Raspberry pi camera module taking photos



Raspberry pi taking photos

4.6. Streaming Video With RPi

1. Use raspivid to create an infinite stream at a resolution of 800 x 600 at 15 fps. The -ifx switch will be populated by a randomly chosen effect from the array To stream live video, this time we use a standard TCP stream. It works a little faster than rtsp but your stream may have artifacts. The Pi listens (-l) for connections from any local IP address.

```
raspivid -t 0 -w 800 -h 600 -ifx ${array[$index]} -fps 15 -l  
-o tcp://0.0.0.0:5000
```

2. Save the code to the /home/pi/ directory and exit from the editor.
3. Open a terminal and use this command to make the code executable.

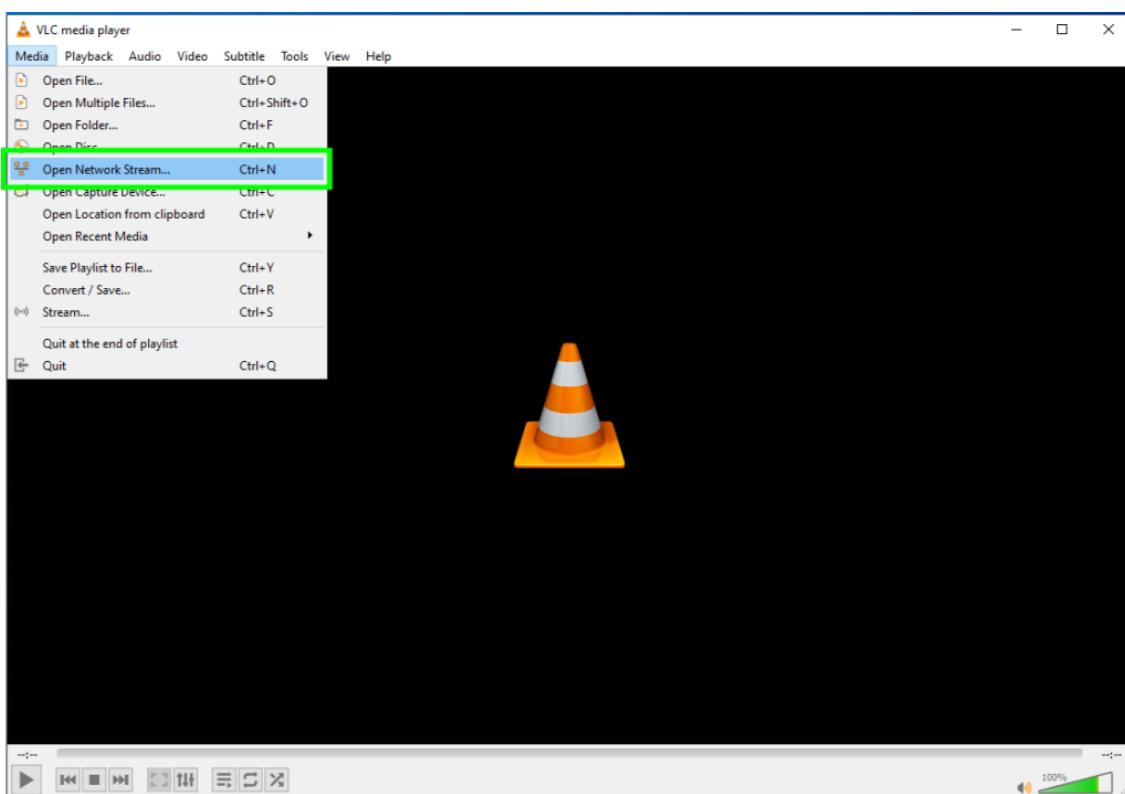
```
$ chmod +x random_stream.sh
```

4. Run the code.

```
$ ./random_stream.sh
```

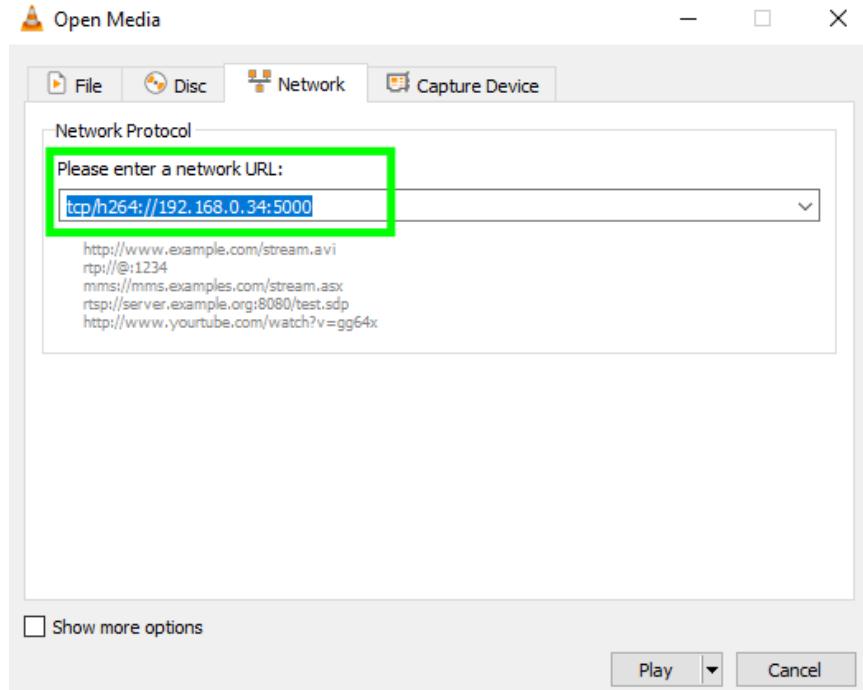
5. On your Windows / Mac / Linux computer open VLC.

6. Go to Media >> Open Network Stream, or press CTRL + N.

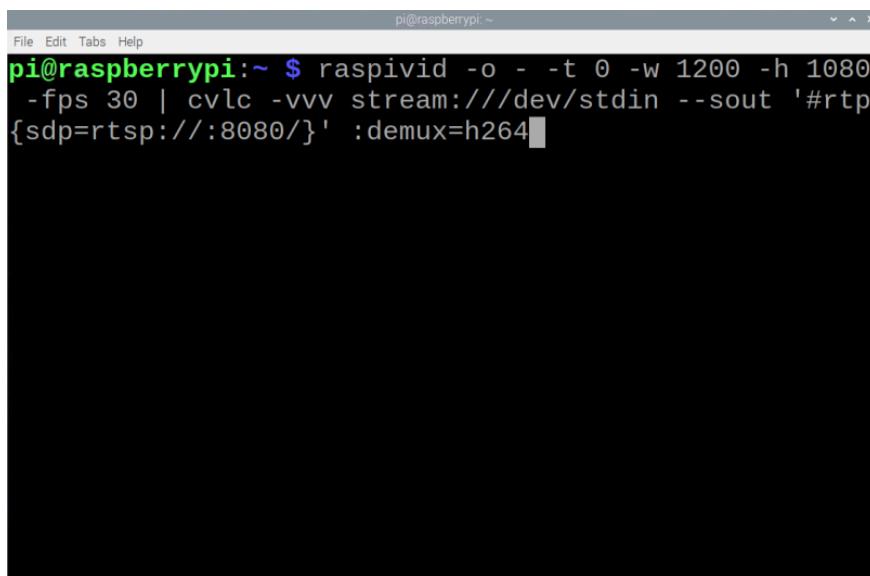


13. Enter the TCP address and port of your Raspberry Pi. The TCP address is the Raspberry Pi's hostname, prefixed with `tcp/h264://`, and the port is set to 5000. Click Play to start the stream. Here is our example TCP address.

`tcp/h264://raspberrypi.local:5000`

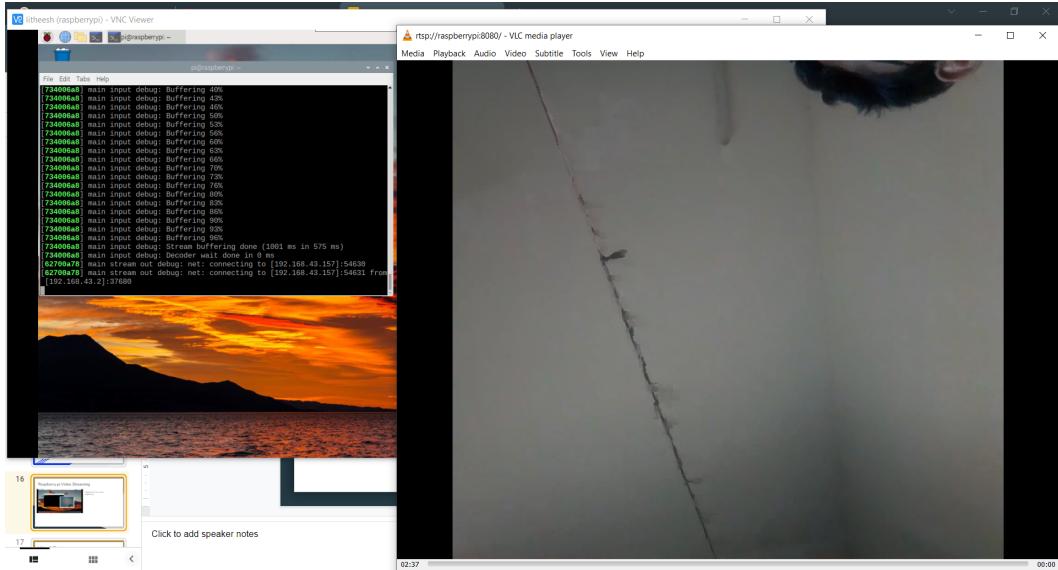


14. To record a stream, click on Playback >> Record and then specify a filename. The recording will end when the stop button is pressed.



```
pi@raspberrypi:~ $ raspivid -o - -t 0 -w 1200 -h 1080 -fps 30 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8080/}' :demux=h264
```

Command for raspberry pi for video streaming



Live streaming from raspberry pi in VLC Player

Here is the full flow chart of our project which consists of all submodules from starting where IoT connection is given using MQTT protocol and the data will be sent to drone raspberry pi. Video streaming by drone raspberry pi camera module. Data from the cloud will be visible on the Website. When the algorithm identifies outliers then we can actuate the processes like turning ON pump motor.

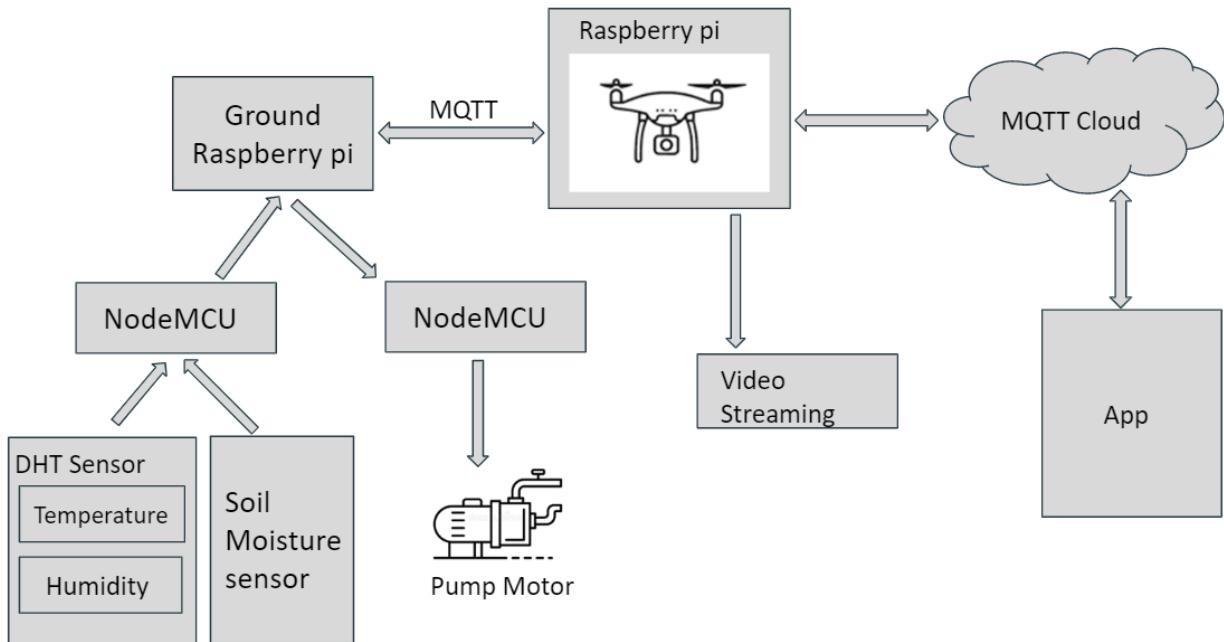


Figure4: Full flow chart of the project

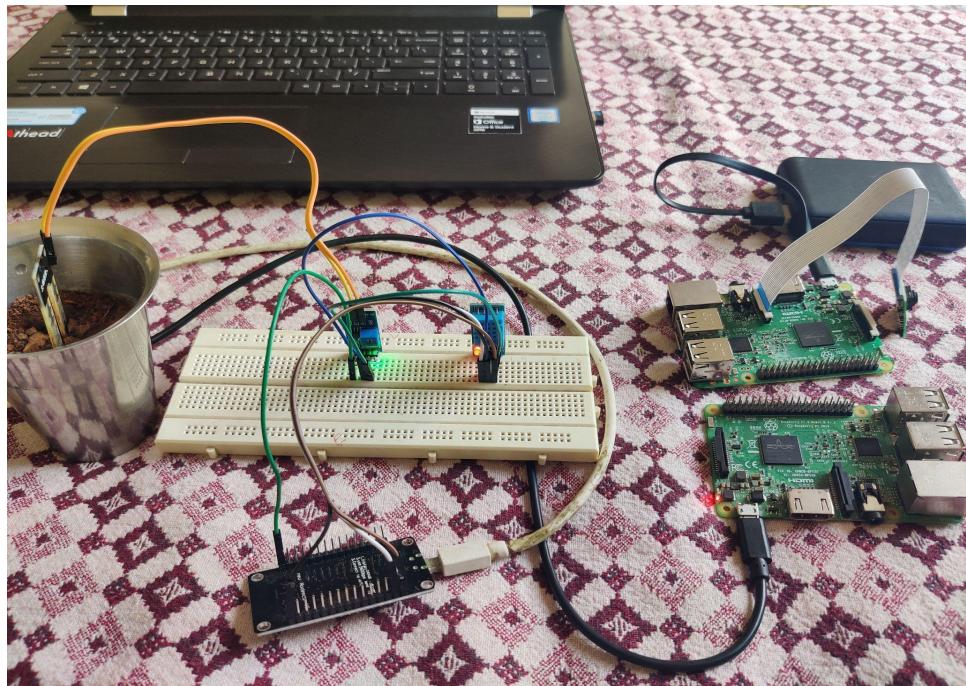


Figure5: Hardware connections of the project

Links for Arduino coding [[Click here](#)]

4.7. Tello Drone Object Tracking:

You may actually use any programming language that can establish a UDP connection to send commands to the Tello drone, however, for this project, we will use Python to send commands to the drone. Python programming is intermediate to advanced subject.



Once the DJI Tello can be controlled remotely, the next step to color tracking is to display the camera feed that the Tello sees on the computer using CV2. The pixels in the image can be searched for pixels of a certain color by analyzing RGB values. In this project, we used a range of violet values. The code will blur together pixels within the largest area blue color range and draw a circle around the violet object we hope to track.

Tracking the object is done using a PID control loop incorporating the offset coordinates of the center of the object and the center of the frame. This is done by using the coordinates from the previous frame and comparing it to the current frame. These coordinate distances are computed in pixels. The goal of tracking is to track where the blue object is "now" based off where it was in the last frame.

The control loop uses the offset coordinates and the distance from the center to find velocities. These velocities are used to command the drone to move left, right, up, or down. It runs the loop for each frame and moves the drone accordingly. The drone should move either up, down, left, or right by observing the sign of the velocity in either the x or y direction. The velocity at which the drone moves is the magnitude of the value computed by the control loop. All of this should result in a drone tracking a violet object.

From tello we can take photos and record video manually and automatically by python coding. Here are the links for python coding we used for operating the Tello drone in this project

Link for Python code [[Click here](#)]

4.8.Basic Steps for Object tracking:

- Identify different objects in the captured image based on the color.
- Convert image to an HSI plane (HUE, SATURATION, and INTENSITY)
- We get the binary mask of our object.
- Try to put the COM of the object in the frame.
- Use PID to calculate the speed and movements of the drone.

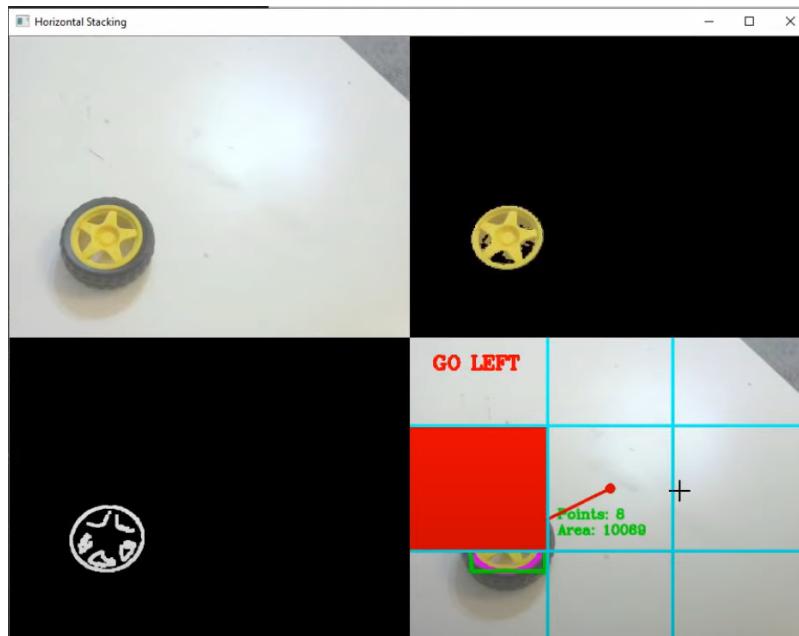
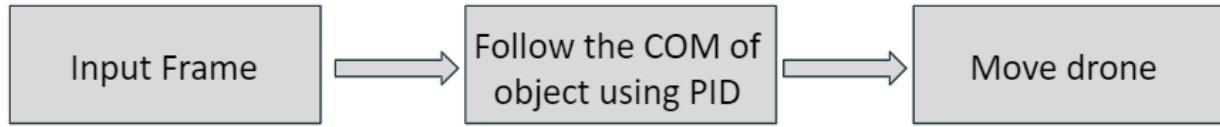


Figure6:Example of Stacking of all results

Quadrant1: Image was taken by drone

Quadrant2: Blurring all colors other than required color

Quadrant3: converting an RGB image to HSI image

Quadrant4: Now we draw a boundary region around all required points. If the center of the object deviates from the center of the frame then it popups to correct its frame by saying GO LEFT, GO RIGHT, GO TOP, GO BOTTOM, etc...

Link to Python Code to Object Detection and Tracking [[Click here](#)]

Gdrive link of photos taken by Tello drone [[Click here](#)]

Video link of the Phase 2 Work [[Click here](#)]

4.9. Dronepath tracking:

We assigned Tello a dedicated path and Tello uses computer vision algorithms in python to detect objects using colors and moves along this path when it reaches the node point of the IoT connection. Given the drone, the path is in figure 10. So at first drone starts from its starting position(Helipad) and reaches the first tree (numbered 1) and collects data from its node and sends the data to the cloud and the same way it collects data from all the trees and reaches its starting position.

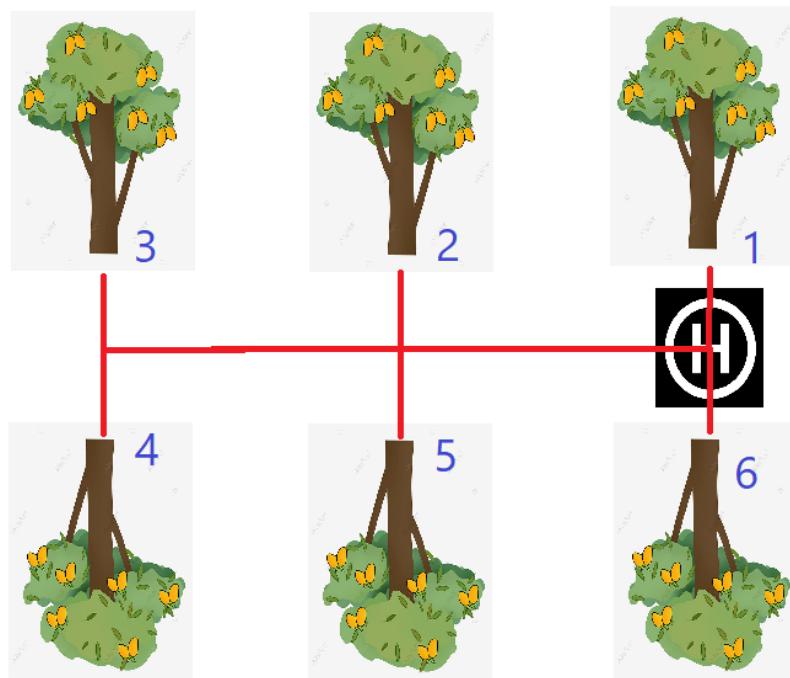


Figure7. Assigned Tello drone path

Link for python code of drone path [[Click here](#)]

Google drive link for pics and video taken by Tello [[Click here](#)]

Link for Website Source code [[Click here](#)]

Explanation of Code:

The code below will use Python to fly your Tello drone on a pre-determined flight path. There are four main Python files.

- [stats.py](#) defines the Stats object to store statistics on command requests and responses
- [tello.py](#) defines the Tello object to abstract and represent the Tello drone
- [command.txt](#) stores the commands that will be sent to the Tello drone
- [app.py](#) is the application program, or the driver, that will be executed as an entry point to start sending commands

Here, we can have the sample code of drone while the drone is flying to the tree (numbered 1). Firstly, we want to know the measurements of the experiment that is the distance between both the trees (numbered 1 and 2) is 500 centimeters, and the trees (numbered 1 and 6) are 400 centimeters.

```
# tree 1-----
    me.tello.move_forward(200)      # Forward 200cms
    def TelloObjectTracking(self):# Run program
        return self.tello()
    me.tello.rotate_cw(180)         # Rotate 180 degrees clockwise
    me.tello.move_forward(200)      # Forward 200cms
    me.tello.rotate_cw(90)          # Rotate 90 degrees clockwise
    me.tello.move_forward(500)      # Forward 500cms
    me.tello.rotate_cw(90)          # Rotate 90 degrees clockwise
```

The drone starts flying from the starting position (Helipad symbol) travels to each tree as numbered and for that drone will move forward 200 centimeters ie. (from helipad to tree1) then it will collect data from sensors and send to the cloud which will be done by running [TelloMain.py](#) and the stabilization and object tracking for to get into its trajectory will be done by [TelloObjectTracking.py](#). After that drone will rotate 180 degrees clockwise, then moves forward to reach the horizontal drone path line ie. 200 centimeters far from its last position.

Now drone will rotate 90 degrees to face the path, then moves forward 500 centimeters to reach another tree path. After that, it will rotate 90 degrees to face the tree. In the same way, the drone will collect all the data from the trees and reach its starting destination. All the codes are in the above-given links.

4.10. Data visible on the Website:

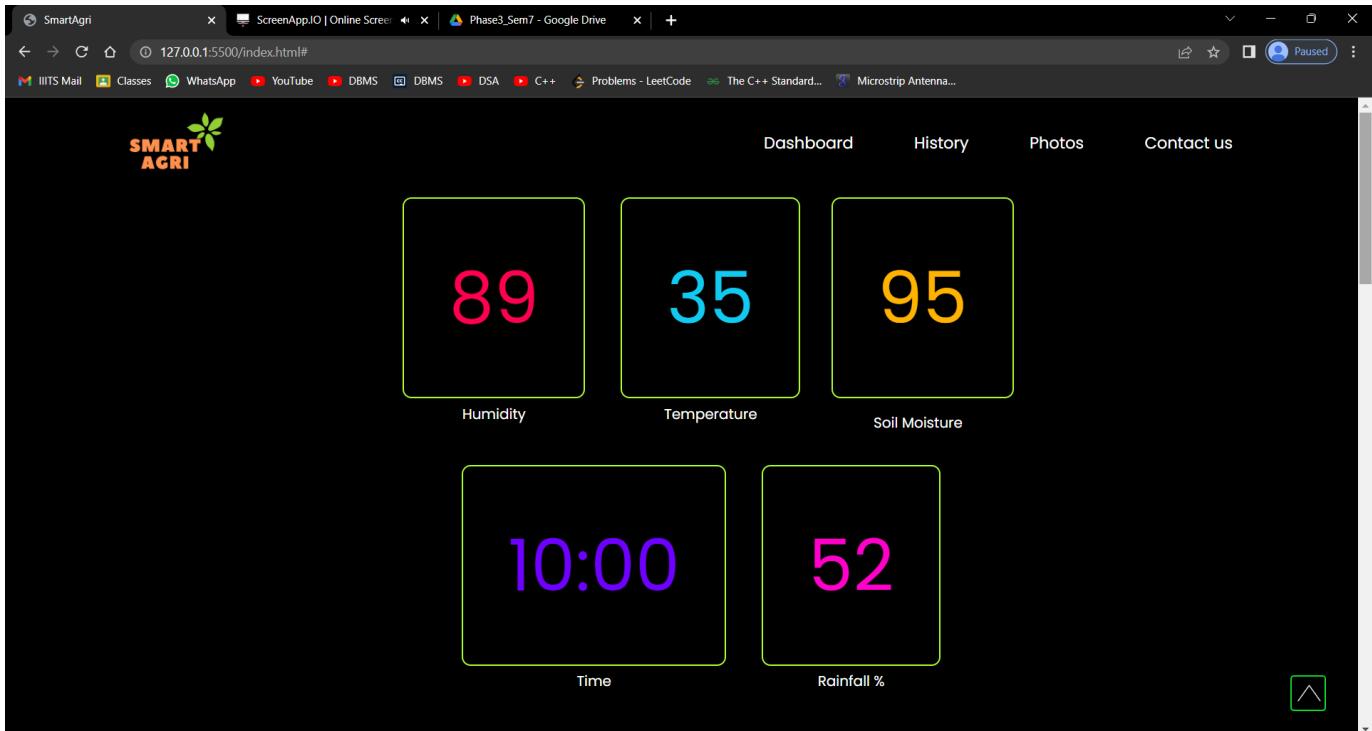


Figure8. Screenshot of Website dashboard

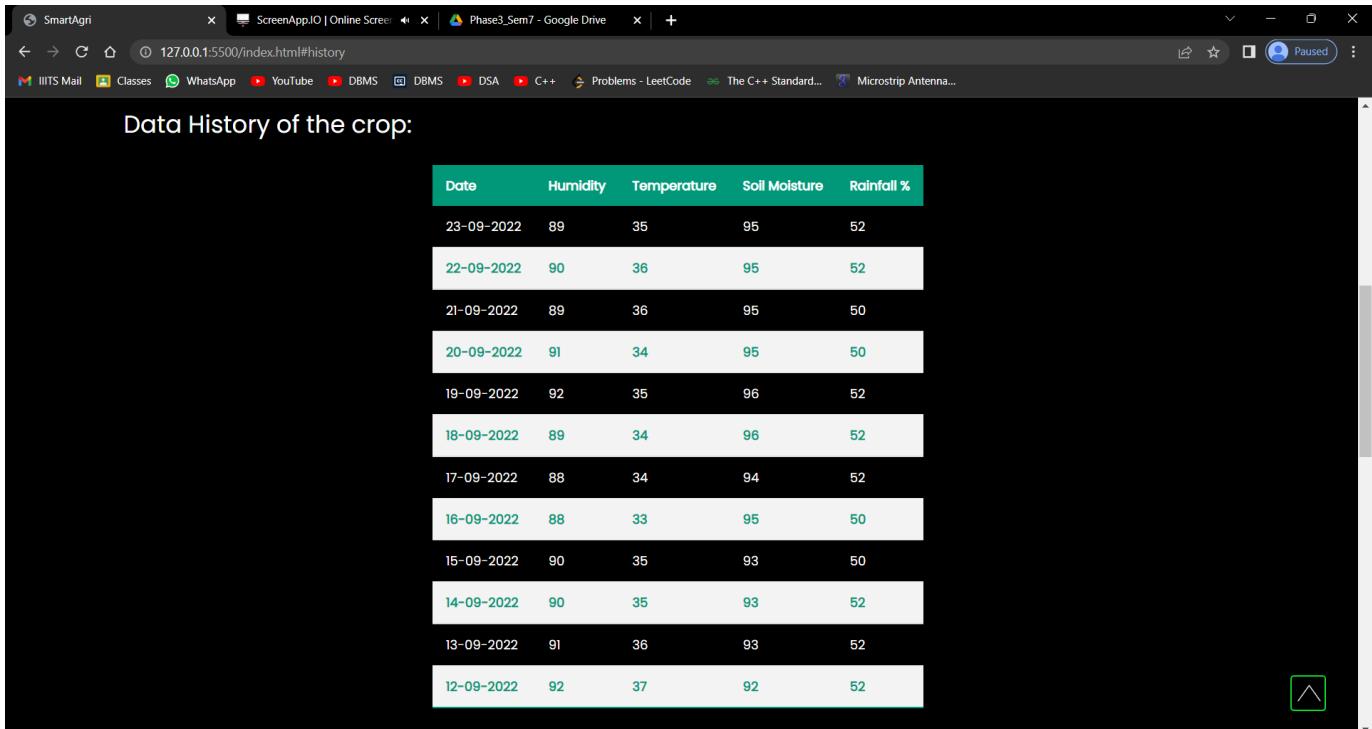


Figure9. screenshot of website data history column

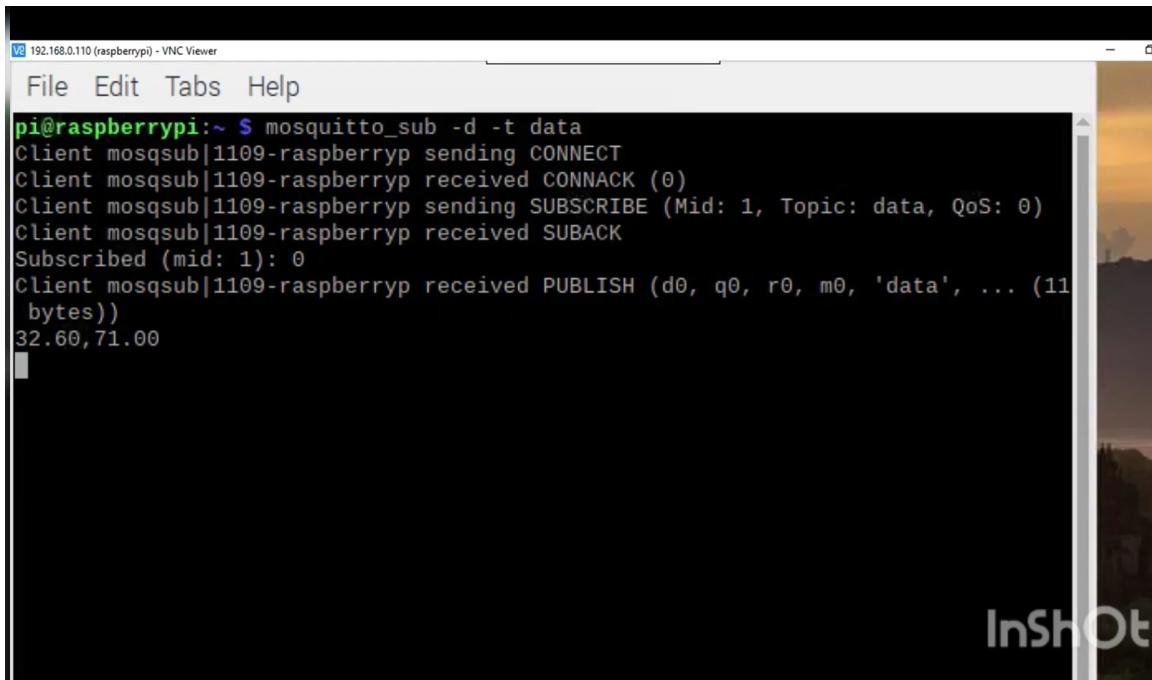
5. RESULTS

Video links of final results [[Click here](#)]

Table 1: Delay in streaming video and taking photos using raspberry pi camera module

Resolution	Delay in Streaming Video	Delay in taking Photo
1080x700	4 sec	0.5 sec
2500x1080	5 sec	1.2 sec
2592x1944	6.5 sec	3 sec

5.1. Results from MQTT connection from figure 3:



```
pi@raspberrypi:~ $ mosquitto_sub -d -t data
Client mosqsub|1109-raspberryp sending CONNECT
Client mosqsub|1109-raspberryp received CONNACK (0)
Client mosqsub|1109-raspberryp sending SUBSCRIBE (Mid: 1, Topic: data, QoS: 0)
Client mosqsub|1109-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|1109-raspberryp received PUBLISH (d0, q0, r0, m0, 'data', ... (11 bytes))
32.60,71.00
```

Figure10. MQTT Publisher part output

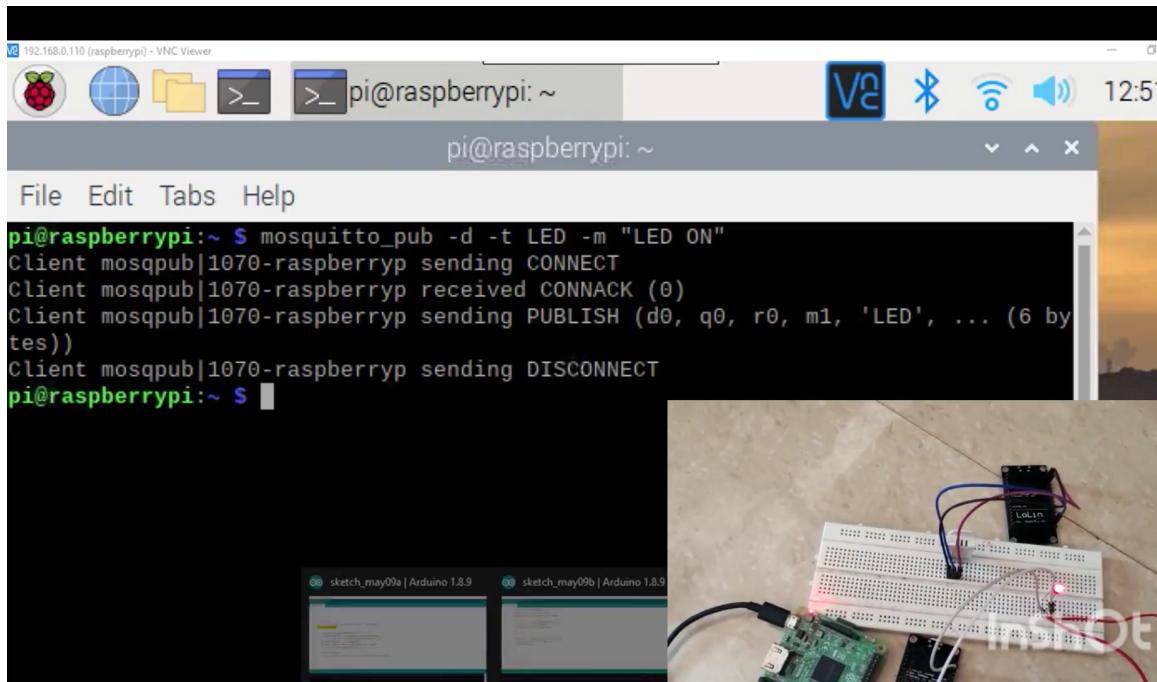


Figure11. MQTT Subscriber part output

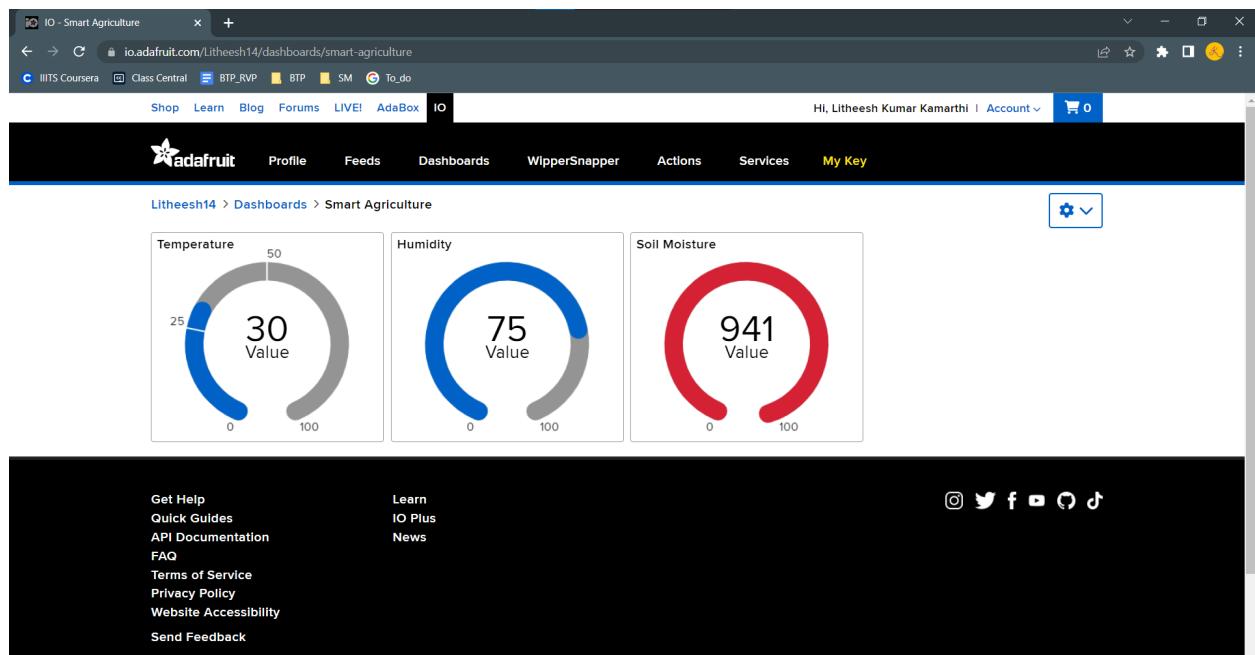


Figure12. Adafruit Cloud output

5.2. Results from Tello drone Python coding:



PyCharm IDE screenshot showing a Python project for a Tello drone. The code in `TelloMain.py` includes image processing and drone control logic. The terminal output shows the drone's response to commands like 'takeoff' and 'land'.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Drone - C:\Users\DELL\Downloads\tello drone\TYP Tello Project\TrackingObjectProject\TelloMain.py
C: Users DELL Downloads tello drone FYP Tello Project TrackingObjectProject TelloMain.py
Project - TelloMain.py ObjectTracking\Tello.py
  28 img = cv2.resize(myFrame, (width, height))
  29
  30 cv2.imshow("MyResult", img)
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  42
  43
  44
  45
  46
  47
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  829
  830
  831
  832
  833
  834
  835
  836
  837
  837
  838
  839
  839
  840
  841
  842
  843
  844
  845
  846
  847
  847
  848
  849
  849
  850
  851
  852
  853
  854
  855
  856
  857
  857
  858
  859
  859
  860
  861
  862
  863
  864
  865
  866
  867
  867
  868
  869
  869
  870
  871
  872
  873
  874
  875
  876
  877
  877
  878
  879
  879
  880
  881
  882
  883
  884
  885
  886
  887
  887
  888
  889
  889
  890
  891
  892
  893
  894
  895
  896
  897
  897
  898
  899
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  909
  910
  911
  912
  913
  914
  915
  916
  917
  917
  918
  919
  919
  920
  921
  922
  923
  924
  925
  926
  927
  927
  928
  929
  929
  930
  931
  932
  933
  934
  935
  936
  937
  937
  938
  939
  939
  940
  941
  942
  943
  944
  945
  946
  947
  947
  948
  949
  949
  950
  951
  952
  953
  954
  955
  956
  957
  957
  958
  959
  959
  960
  961
  962
  963
  964
  965
  966
  967
  967
  968
  969
  969
  970
  971
  972
  973
  974
  975
  976
  977
  977
  978
  979
  979
  980
  981
  982
  983
  984
  985
  986
  987
  987
  988
  989
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1017
  1018
  1019
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1027
  1028
  1029
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1037
  1038
  1039
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1047
  1048
  1049
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1057
  1058
  1059
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1067
  1068
  1069
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1077
  1078
  1079
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1087
  1088
  1089
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1097
  1098
  1099
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1107
  1108
  1109
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1116
  1117
  1118
  1118
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1126
  1127
  1128
  1128
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1136
  1137
  1138
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1145
  1146
  1147
  1147
  1148
  1149
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1156
  1157
  1158
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1165
  1166
  1167
  1167
  1168
  1169
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1175
  1176
  1177
  1177
  1178
  1179
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1186
  1187
  1188
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1194
  1195
  1196
  1196
  1197
  1198
  1198
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1205
  1206
  1207
  1207
  1208
  1209
  1209
  1210
  1211
  1212
  1213
  1214
  1214
  1215
  1216
  1216
  1217
  1218
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1224
  1225
  1226
  1226
  1227
  1228
  1228
  1229
  1230
  1231
  1232
  1233
  1234
  1234
  1235
  1236
  1236
  1237
  1238
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1244
  1245
  1246
  1246
  1247
  1248
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1254
  1255
  1256
  1256
  1257
  1258
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1264
  1265
  1266
  1266
  1267
  1268
  1268
  1269
  1270
  1271
  1272
  1273
  1273
  1274
  1275
  1275
  1276
  1277
  1277
  1278
  1279
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1285
  1286
  1287
  1287
  1288
  1289
  1289
  1290
  1291
  1292
  1293
  1294
  1294
  1295
  1296
  1296
  1297
  1298
  1298
  1299
  1300
  1301
  1302
  1303
  1304
  1304
  1305
  1306
  1306
  1307
  1308
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1314
  1315
  1316
  1316
  1317
  1318
  1318
  1319
  1320
  1321
  1322
  1323
  1323
  1324
  1325
  1325
  1326
  1327
  1327
  1328
  1329
  1329
  1330
  1331
  1332
  1333
  1334
  1334
  1335
  1336
  1336
  1337
  1338
  1338
  1339
  1340
  1341
  1342
  1343
  1343
  1344
  1345
  1345
  1346
  1347
  1347
  1348
  1349
  1349
  1350
  1351
  1352
  1353
  1354
  1354
  1355
  1356
  1356
  1357
  1358
  1358
  1359
  1360
  1361
  1362
  1363
  1363
  1364
  1365
  1365
  1366
  1367
  1367
  1368
  1369
  1369
  1370
  1371
  1372
  1373
  1374
  1374
  1375
  1376
  1376
  1377
  1378
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1384
  1385
  1386
  1386
  1387
  1388
  1388
  1389
  1390
  1391
  1392
  1393
  1393
  1394
  1395
  1395
  1396
  1397
  1397
  1398
  1399
  1399
  1400
  1401
  1402
  1403
  1404
  1404
  1405
  1406
  1406
  1407
  1408
  1408
  1409
  1410
  1411
  1412
  1413
  1413
  1414
  1415
  1415
  1416
  1417
  1417
  1418
  1419
  1419
  1420
  1421
  1422
  1423
  1423
  1424
  1425
  1425
  1426
  1427
  1427
  1428
  1429
  1429
  1430
  1431
  1432
  1433
  1434
  1434
  1435
  1436
  1436
  1437
  1438
  1438
  1439
  1440
  1441
  1442
  1443
  1443
  1444
  1445
  1445
  1446
  1447
  1447
  1448
  1449
  1449
  1450
  1451
  1452
  1453
  1454
  1454
  1455
  1456
  1456
  1457
  1458
  1458
  1459
  1460
  1461
  1462
  1463
  1463
  1464
  1465
  1465
  1466
  1467
  1467
  1468
  1469
  1469
  1470
  1471
  1472
  1473
  1474
  1474
  1475
  1476
  1476
  1477
  1478
  1478
  1479
  1480
  1481
  1482
  1483
  1483
  1484
  1485
  1485
  1486
  1487
  1487
  1488
  1489
  1489
  1490
  1491
  1492
  1493
  1493
  1494
  1495
  1495
  1496
  1497
  1497
  1498
  1499
  1499
  1500
  1501
  1502
  1503
  1504
  1504
  1505
  1506
  1506
  1507
  1508
  1508
  1509
  1510
  1511
  1512
  1513
  1513
  1514
  1515
  1515
  1516
  1517
  1517
  1518
  1519
  1519
  1520
  1521
  1522
  1523
  1523
  1524
  1525
  1525
  1526
  1527
  1527
  1528
  1529
  1529
  1530
  1531
  1532
  1533
  1534
  1534
  1535
  1536
  1536
  1537
  1538
  1538
  1539
  1540
  1541
  1542
  1543
  1543
  1544
  1545
  1545
  1546
  1547
  1547
  1548
  1549
  1549
  1550
  1551
  1552
  1553
  1554
  1554
  1555
  1556
  1556
  1557
  1558
  1558
  1559
  1560
  1561
  1562
  1563
  1563
  1564
  1565
  1565
  1566
  1567
  1567
  1568
  1569
  1569
  1570
  1571
  1572
  1573
  1574
  1574
  1575
  1576
  1576
  1577
  1578
  1578
  1579
  1580
  1581
  1582
  1583
  1583
  1584
  1585
  1585
  1586
  1587
  1587
  1588
  1589
  1589
  1590
  1591
  1592
  1593
  1593
  1594
  1595
  1595
  1596
  1597
  1597
  1598
  1599
  1599
  1600
  1601
  1602
  1603
  1604
  1604
  1605
  1606
  1606
  1607
  1608
  1608
  1609
  1610
  1611
  1612
  1613
  1613
  1614
  1615
  1615
  1616
  1617
  1617
  1618
  1619
  1619
  1620
  1621
  1622
  1623
  1623
  1624
  1625
  1625
  1626
  1627
  1627
  1628
  1629
  1629
  1630
  1631
  1632
  1633
  1634
  1634
  1635
  1636
  1636
  1637
  1638
  1638
  1639
  1640
  1641
  1642
  1643
  1643
  1644
  1645
  1645
  1646
  1647
  1647
  1648
  1649
  1649
  1650
  1651
  1652
  1653
  1654
  1654
  1655
  1656
  1656
  1657
  1658
  1658
  1659
  1660
  1661
  1662
  1663
  1663
  1664
  1665
  1665
  1666
  1667
  1667
  1668
  1669
  1669
  1670
  1671
  1672
  1673
  
```

Figure13. Image taken by in automated python command mode



Figure14. Tello drone flying



Figure15. Photo of Tello drone tracking its assigned color region

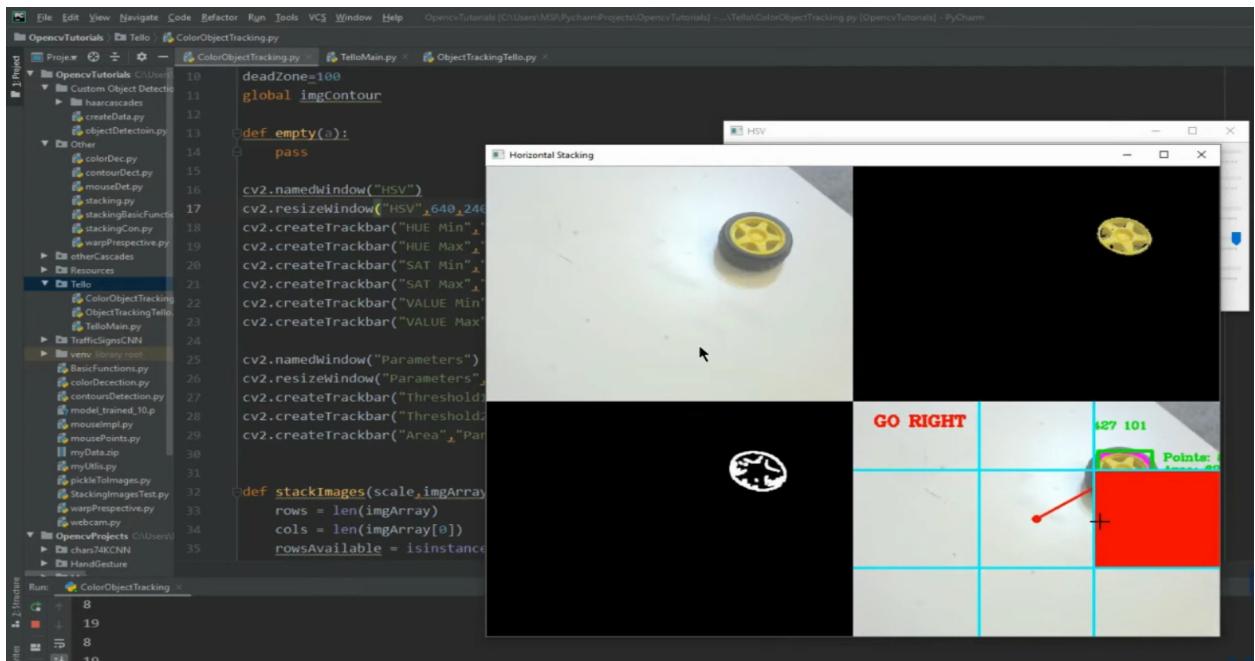


Figure16. Screenshot of stacking

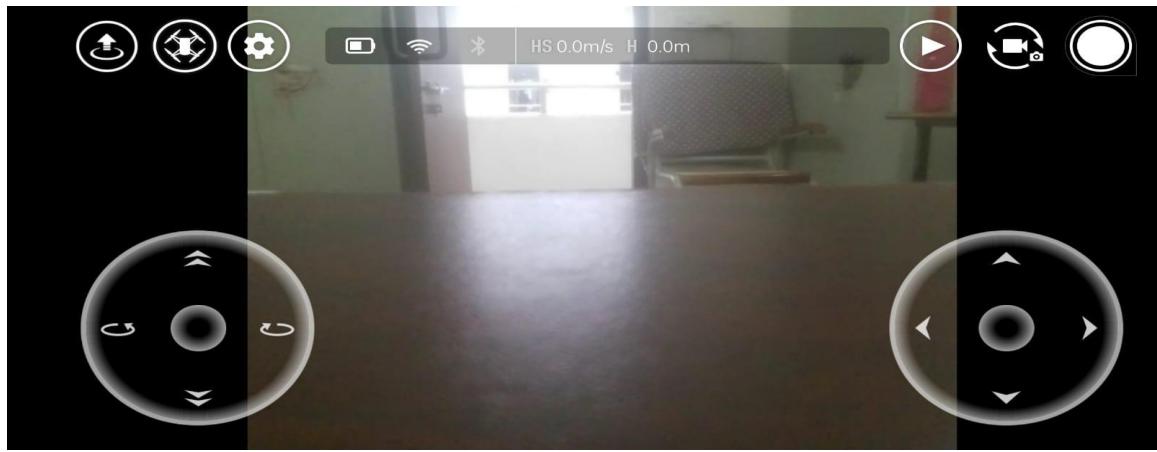


Figure17. Tello drone manual mode flying joy stick in Tello mobile app

5.3. Results from Tello drone Object tracking:

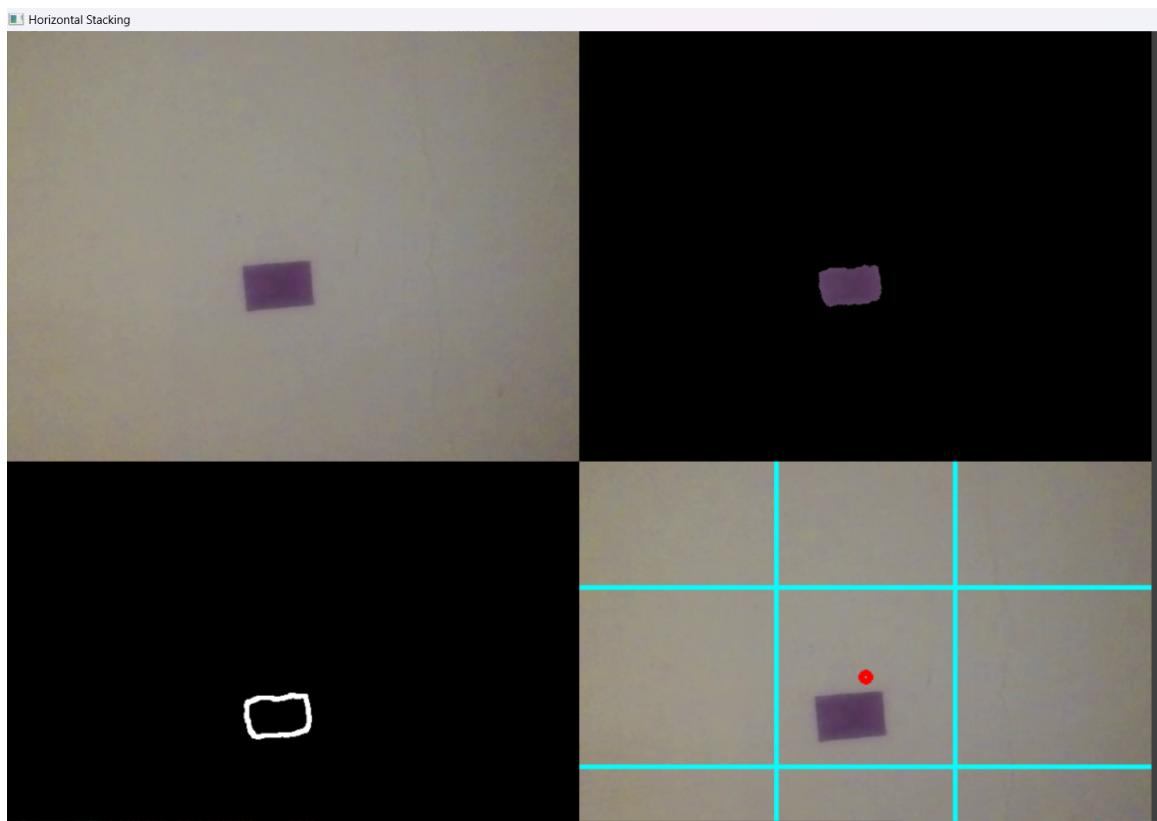


Figure18. Stacking of images

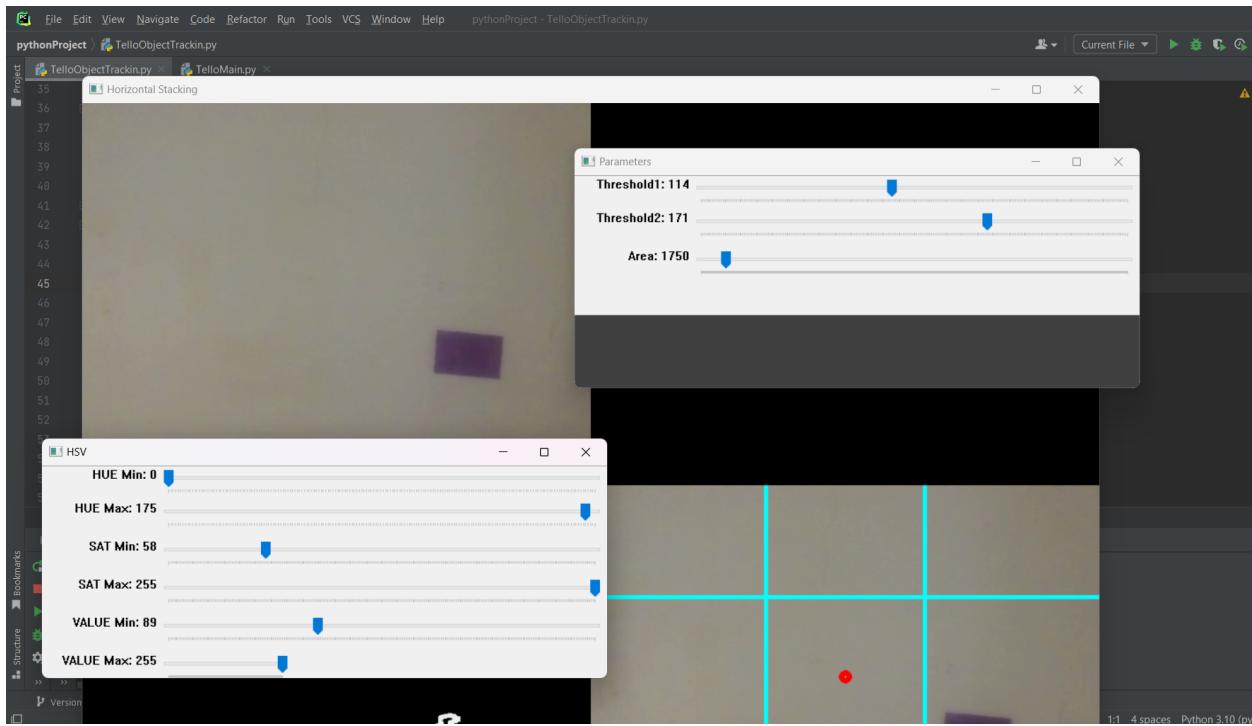
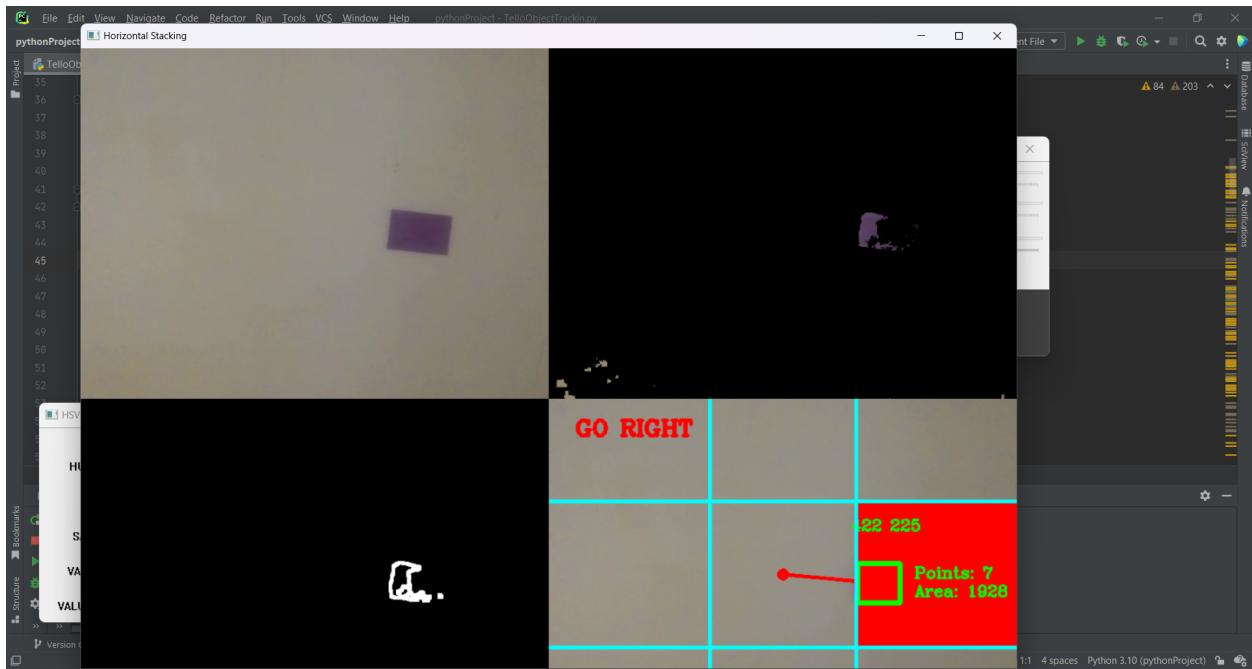


Figure19. Adjusting Threshold values



6. WORK DONE FOR FINAL EVALUATION

- ❖ We are using the Tello drone in the Final Evaluation.
- ❖ It can be flown both through the app and commands.
- ❖ Python code for automated flying mode.
- ❖ It detects the object by Computer Vision(CV).
- ❖ follows the objects and reaches the nearest path to the object.
- ❖ And take photos and videos at that particular location.
- ❖ Optimizing CV (Computer Vision) model for object tracking
- ❖ Coding for giving instructions to ground level Actuators like pump motor etc..
- ❖ Optimized object detection and tracking.
- ❖ We have given a path to drone.
- ❖ Tello drone follows that path by using Python code.
- ❖ If anywhere drone misses its path it uses computer vision to correct its path.
- ❖ And by reaching the end of the path it collects data and passes it to the cloud.
- ❖ Front End, and Backend of the Website, Values will be updated on the website.

7. CONCLUSION

Operating the DJI Tello is an important first step to completing the goal for the tracking of an object. The keyboard commands help the user to place the drone where he or she prefers, (proper height, angle, distance from the object, etc.) in a ready position for tracking.

Once the DJI Tello can be controlled remotely, the next step to color tracking is to display the camera feed that the Tello sees on the computer using CV2. The pixels in the image can be searched for pixels of a certain color by analyzing RGB values. In this project, we used a range of blue values. The code will blur together pixels within the largest area of the violet color range and draw a boundary around the violet object we hope to track.

Tracking the object is done using a PID control loop incorporating the offset coordinates of the center of the object and the center of the frame. This is done by using the coordinates from the previous frame and comparing it to the current frame. These coordinate distances are computed in pixels. The goal of tracking is to track where the blue object is "now" based on where it was in the last frame.

The control loop uses the offset coordinates and the distance from the center to find velocities. These velocities are used to command the drone to move left, right, up, or down. It runs the loop for each frame and moves the drone accordingly. The drone should move either up, down, left, or right by observing the sign of the velocity in either the x or y direction. The velocity at which the drone moves is the magnitude of the value computed by the control loop. All of this should result in a drone tracking a violet object.

In this project, we proposed a method and built an IoT connection using autonomous UAV tracking and landing system that enables UAV tracking and the landing of a designated target in an environment with multiple interference targets. The image information transmitted by the camera was processed by Tello and coordinate transformation to obtain the spatial coordinate information of the specified target; then, the discrete PID control method was adopted to realize tracking and the landing of the UAV on the designated target. Extensive simulation experiments were conducted to evaluate the system's performance. The effectiveness and reliability of the algorithm and system were analyzed qualitatively and quantitatively with regard to various aspects, such as the target pixel coordinates, the position of the UAV relative to the target, and the tracking and landing process. The landing error of the UAV during the landing process after the target was stationary did not exceed 0.01 m. In addition, the advantages and disadvantages of the system were discussed at the level of both processor hardware and algorithm design, and corresponding suggestions were made for different problems.

Although drones are small in size and have restrictions on power, it is important to design an efficient algorithm for drone object detection. Variations viewpoint is another significant challenge to be addressed by the object detection algorithms. In future work, we can aim to investigate object detection using UAVs in agriculture applications. Precision agriculture deals to monitor the fields

precisely by collecting and analyzing the data. Acquiring aerial images using UAVs is less expensive than satellites. Object detection in UAV aerial images in precision farming is an important research area to be investigated. We proposed a secure onboard object detection framework in precision agriculture in this project and implementing it will be our future work.

8. FUTURE WORK

- ❖ Drone object detection has many applications in real time.
- ❖ We think that deep learning algorithms perform better than image processing methods.
- ❖ Although the drones are small in size and have restrictions in power, it is important to design an efficient deep learning algorithm for drone object detection.
- ❖ As future work, we can aim to investigate object detection using UAVs in various other fields.
- ❖ Precision agriculture is to monitor the fields precisely by collecting and analyzing the data.

9. List of Abbreviations

UAV = Unmanned Aerial Vehicle

CV = Computer Vision

DL = Deep Learning

RS = Remote Sensing

GPS = Global Positioning System

IMU = Inertial Measurement Unit

SPPNet = spatial pyramid pooling networks

RCNN = faster Region-Convolutional Neural Networks

SSD = Single-Shot multi-box Detector

ViT FRCNN = Vision Transformer Faster RCNN

FPN = Feature Pyramid Networks

SNIP = Scale Normalization for Image Pyramids

MBSSD = MultiBlock SSD

MQTT = Message Queuing Telemetry Transport

TCP/IP = Internet Protocol Suite

QoS = Quality of Service

TCP = Transmission Control Protocol

UDP = User Datagram Protocol

TLS = Transport Layer Security

10. ACKNOWLEDGMENTS

We would like to thank our mentor, Dr. Y. Raja Vara Prasad for guiding us in this project and providing inputs and suggestions during evaluations and working. All team members have contributed equally, sharing the workload since the beginning.

11. REFERENCES

- [1] M. Shahbazi, J. Theau, and P. Menard, “Recent applications of un-manned aerial imagery in natural resource management,” *GISci. Remote Sens.*, vol. 51, no. 4, pp. 339–365, 2014.
- [2] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,” *IEEE Access*, vol. 7, pp. 48572–48634, 2019.
- [3] J. Hao, Y. Zhou, G. Zhang, Q. Lv, and Q. Wu, “A review of target tracking algorithm based on UAV,” in *Proc. CBS*, pp. 328–333, IEEE, 2018.
- [4] D. Hong, N. Yokoya, N. Ge, J. Chanussot, and X. Zhu, “Learnable manifold alignment (LeMA): A semi-supervised cross-modality learning framework for land cover and land use classification,” *ISPRS J. Photogramm. Remote Sens.*, vol. 147, pp. 193–205, 2019.
- [5] D. Hong, L. Gao, N. Yokoya, J. Yao, J. Chanussot, Q. Du, and B. Zhang, “More diverse means better: Multimodal deep learning meets remote-sensing imagery classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4340–4354, 2021.
- [6] AlDahoul, N., Sabri, A. Q., & Mansoor, A. M. (2018). Real-time human detection for aerial captured video sequences via deep models. Hindawi, 2018.
- [7] Chen, N., Chen, Y., You, Y., Ling, H., Liang, P., & Zimmermann, R. (2016). Dynamic urban surveillance video stream processing using fog computing. In *Proceedings of the 2016 IEEE second international conference on multimedia big data (BigMM)* (pp. 105–112).
- [8] Robicquet, A., Sadeghian, A., Alahi, A., & Savarese, S. (2016). Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European conference on computer vision* (pp. 549–565).