

Colour detection using UAV for **Smart Agriculture**

B22RVP03

Guide: Dr. Y. Raja Vara Prasad

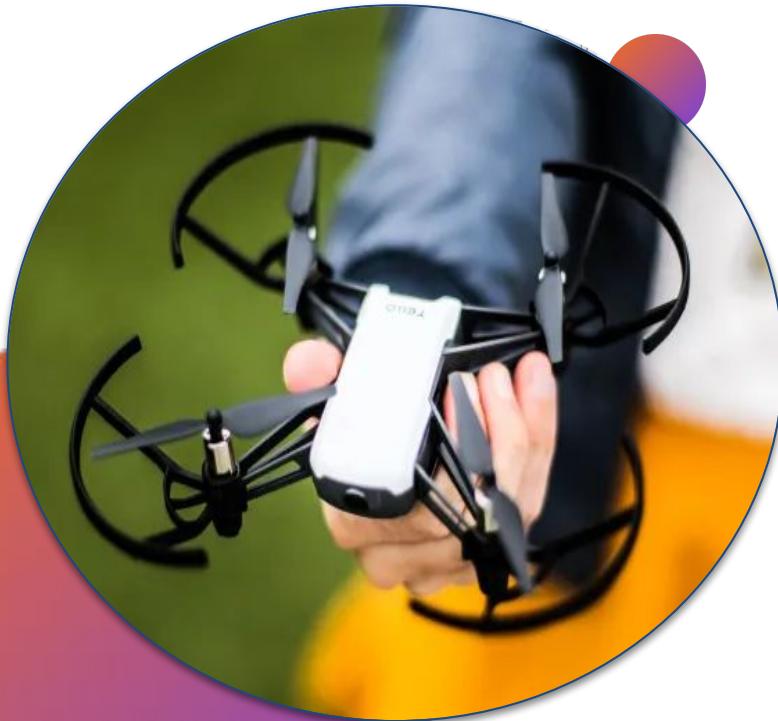
Team members:

Kamarthi Litheesh Kumar - S20190020218

TU Surya Sathvik - S20190020254

Outline:

1. Problem Statement
2. Timeline
3. Recap Phase 1,2,3 + Results
4. Work done in Phase 4
5. Methodology
6. Results
 - a. Drone path
 - b. Python code
 - c. CV outputs
 - d. Cloud results
 - e. Website screenshots
7. Video link
8. Future Explore
9. References

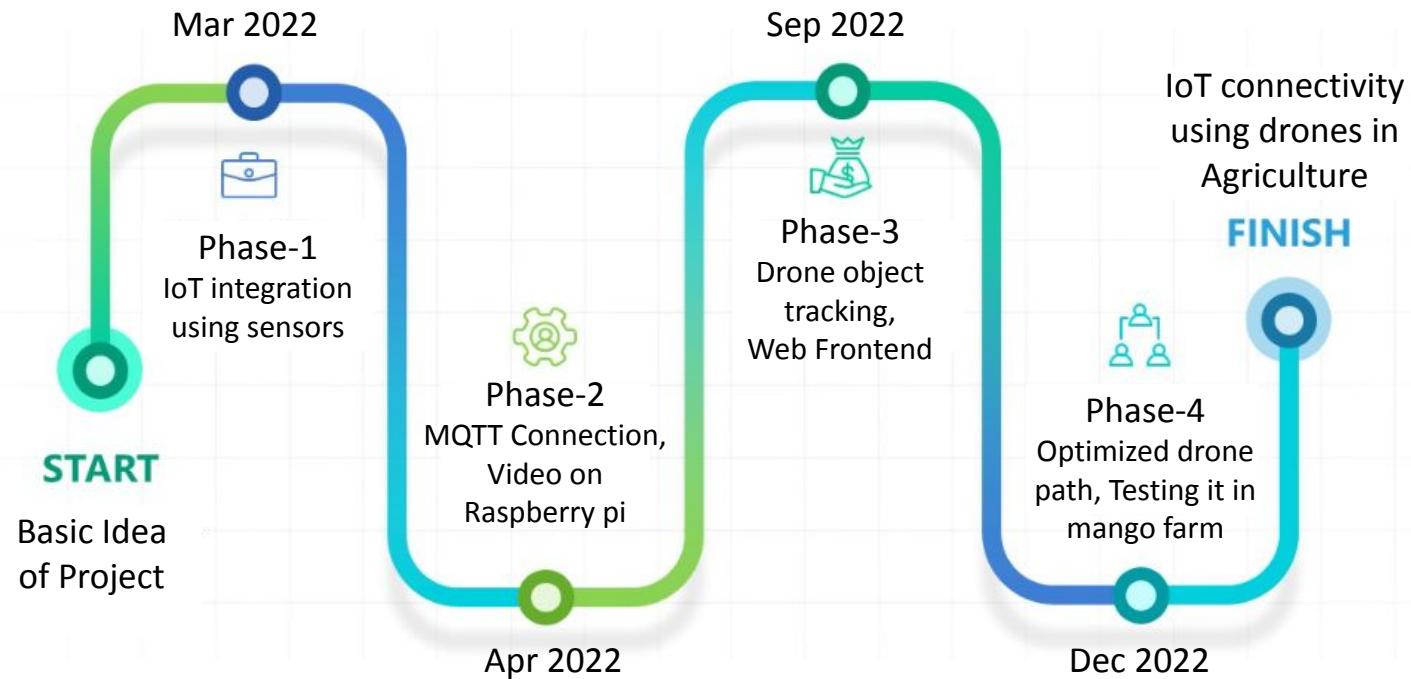


Problem Statement

The connectivity limitations of drone applications in smart agriculture and its solutions are analyzed in this project.

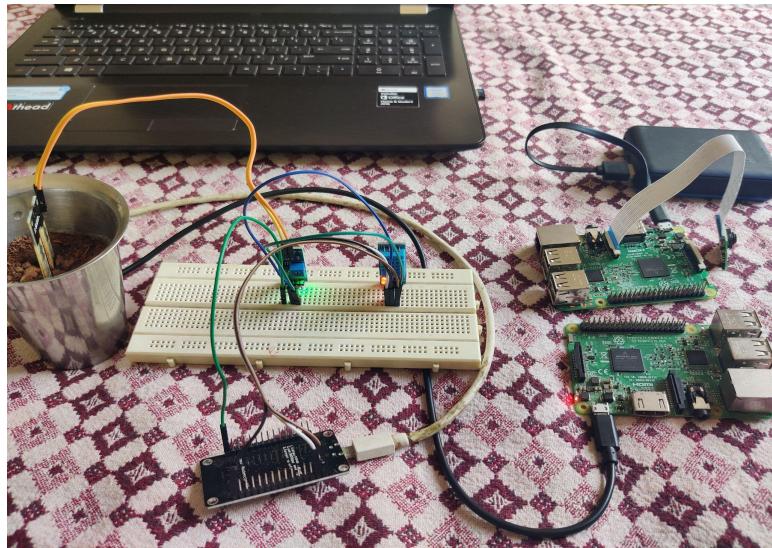
- Writing optimized python code for automating drone path through computer vision.
- Testing the tello drone in mango farm to check its accuracy in object tracking while flying.

Timeline

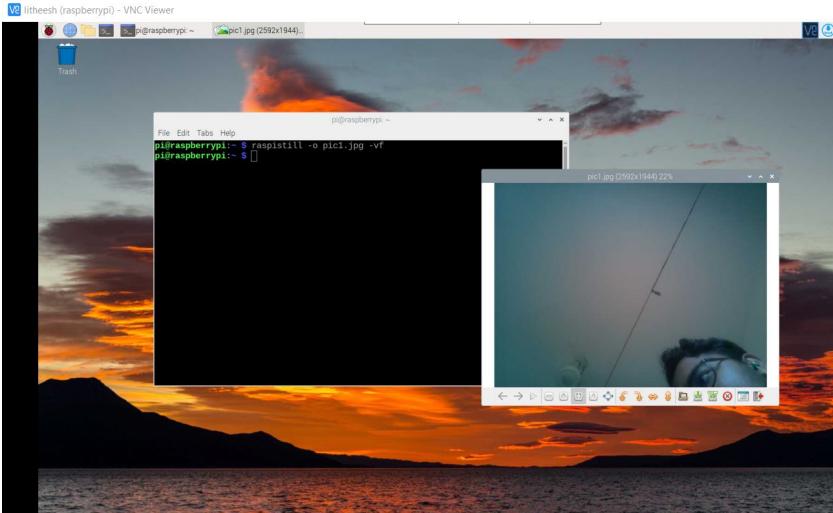


Recap: Phase 1, 2 works

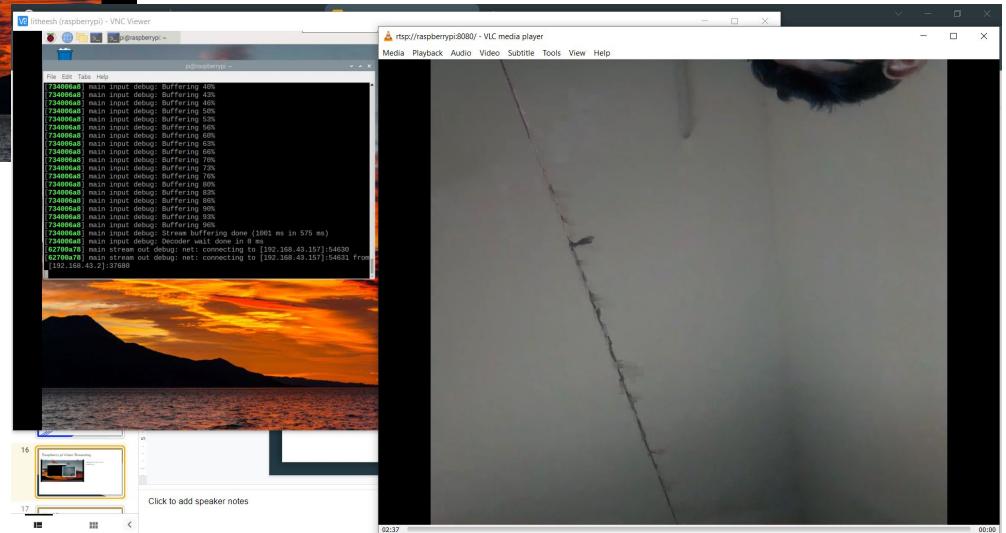
- Sensors are connected to Raspberry pi through NodeMCU.
- Pump Motor is connected to another NodeMCU.
- Both NodeMCUs are connected to Ground Raspberry pi using MQTT.
- Ground Raspberry pi changes to client from broker
- Drone Raspberry pi acts as MQTT broker and published sensor data on cloud.
- Fixing a camera module to Drone Raspberry
- Analyzed the delay in streaming in various resolutions.



Video & Photo

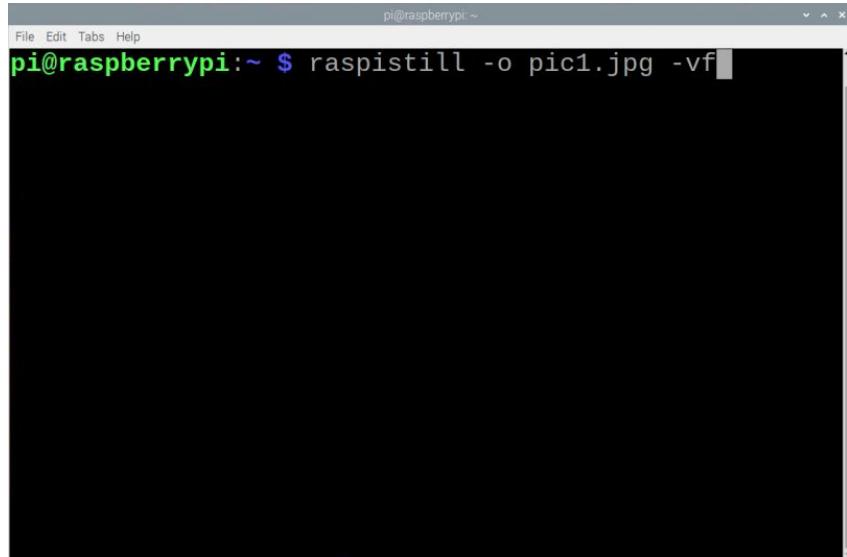


Taking photos from drone raspberry pi



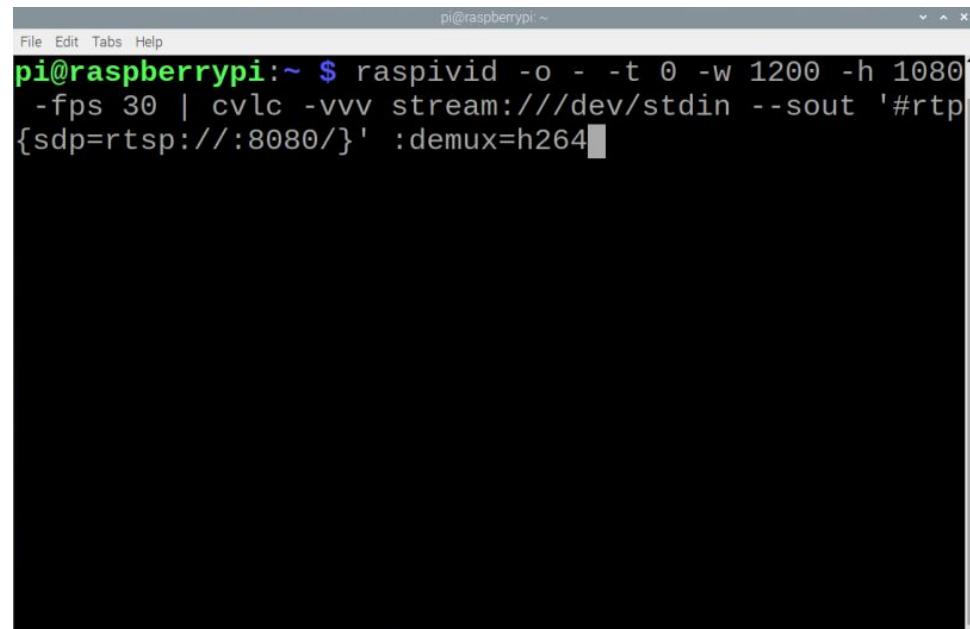
Live streaming video from
drone raspberry pi

Raspberry pi Video Streaming



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

Command line for taking photos from drone raspberry pi



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

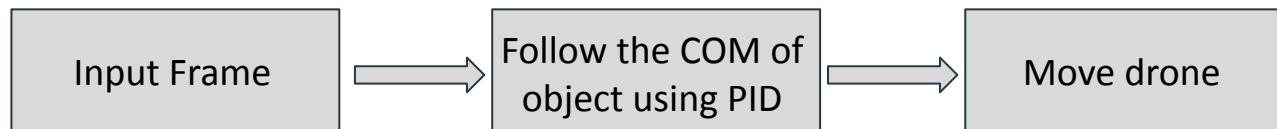
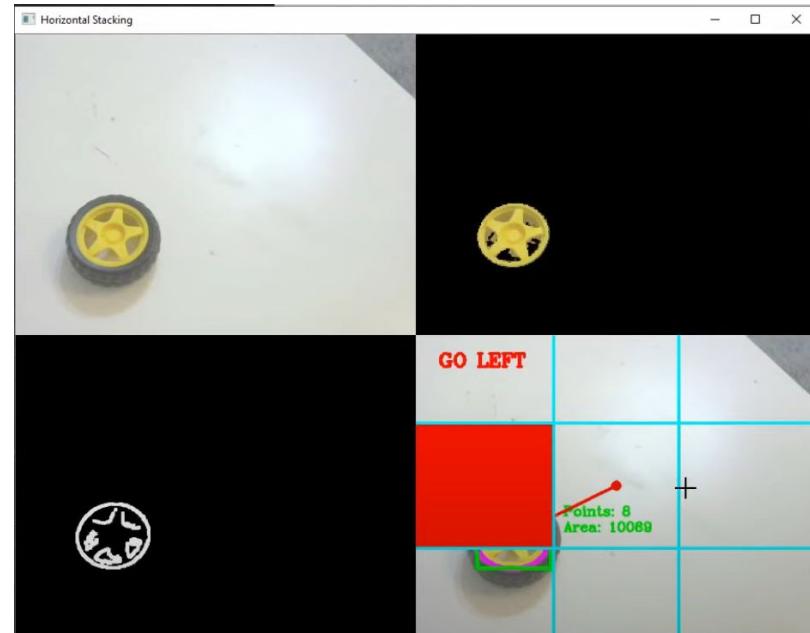
Command line for Live streaming video from drone raspberry pi

Recap: Phase 3 works

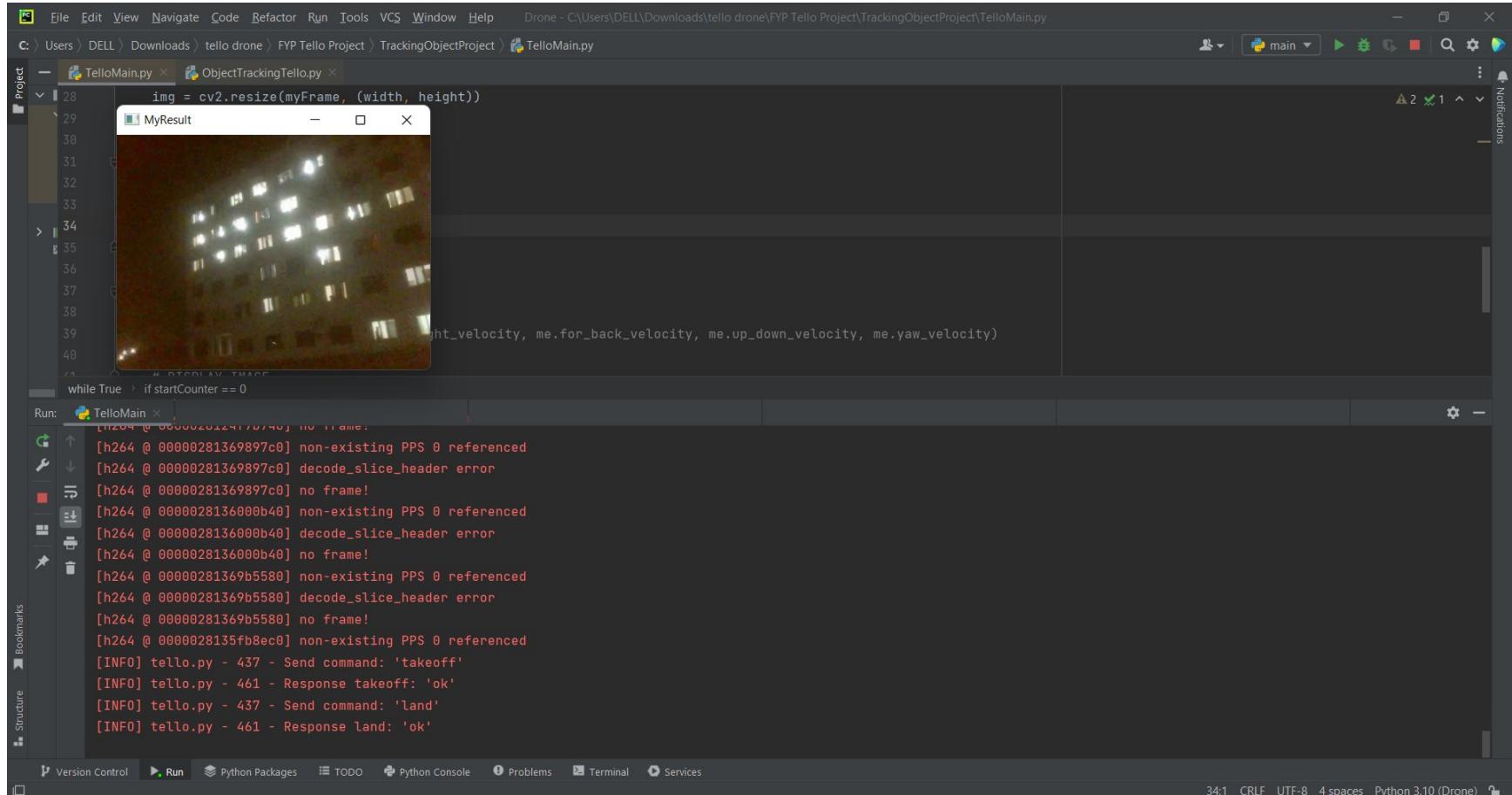
- We used Tello drone
- It can be flown both through app and commands
- Wrote Python code for automating flying mode
- It detects the object based on colour by Computer Vision(CV)
- follows the objects and reaches the nearest path to the object
- And take photos and videos at that particular location.
- Front End of the Website.

Object Tracking Drone

- Identify different objects in the captured image based on the color
- 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
- Now we can calculate the speed and movements of the drone

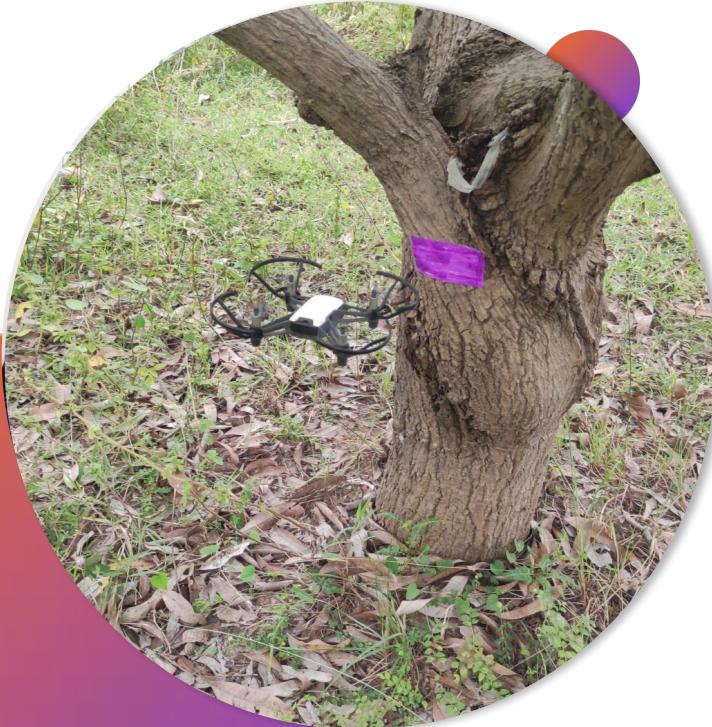


Python output:



The screenshot shows a Python development environment with the following details:

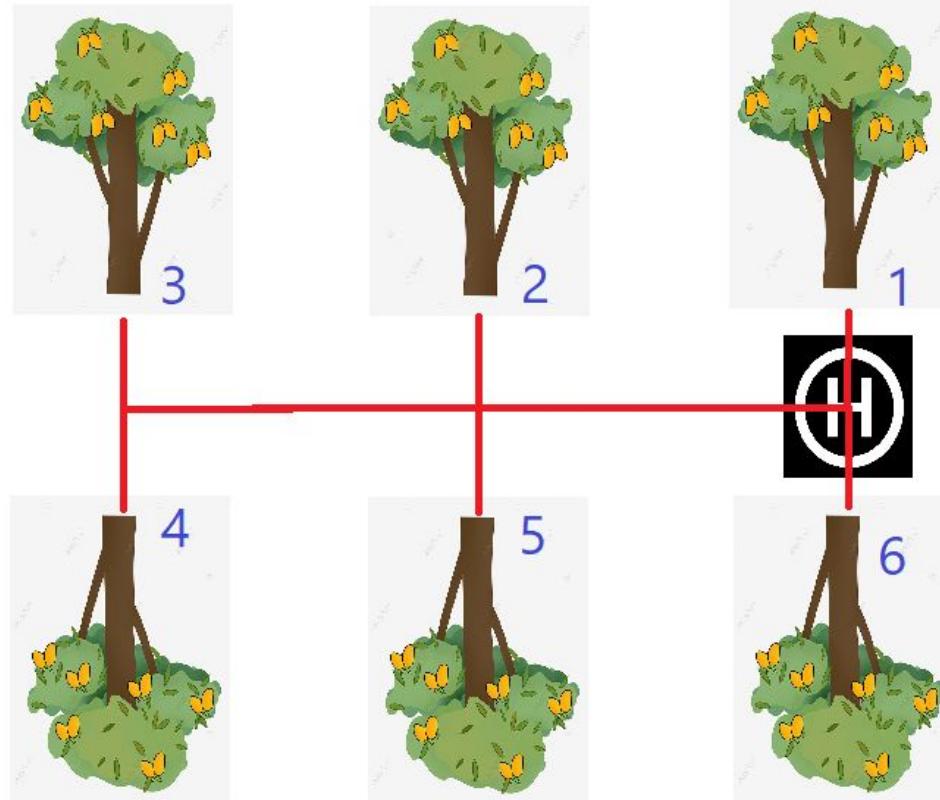
- File Path:** Drone - C:\Users\DELL\Downloads\tello drone\FYP Tello Project\TrackingObjectProject\TelloMain.py
- Code Editor:** The code is for a Tello drone project, specifically for tracking an object. It includes imports for cv2 and numpy, and a loop that processes frames and updates the drone's position.
- Terminal Output:** The terminal shows the following log:
 - Frame processing errors: [h264 @ 00000281369897c0] non-existing PPS 0 referenced, [h264 @ 00000281369897c0] decode_slice_header error, [h264 @ 00000281369897c0] no frame!, [h264 @ 0000028136000b40] non-existing PPS 0 referenced, [h264 @ 0000028136000b40] decode_slice_header error, [h264 @ 0000028136000b40] no frame!, [h264 @ 00000281369b5580] non-existing PPS 0 referenced, [h264 @ 00000281369b5580] decode_slice_header error, [h264 @ 00000281369b5580] no frame!, [h264 @ 0000028135fb8ec0] non-existing PPS 0 referenced.
 - Drone control commands: [INFO] tello.py - 437 - Send command: 'takeoff', [INFO] tello.py - 461 - Response takeoff: 'ok', [INFO] tello.py - 437 - Send command: 'land', [INFO] tello.py - 461 - Response land: 'ok'
- IDE Features:** The interface includes a Project view, a Run tab with a Tello icon, and various status indicators like notifications and build logs.



Phase 4 works:

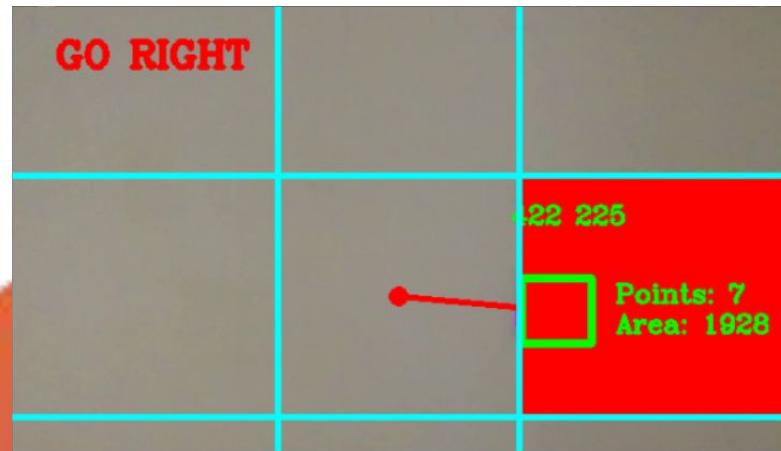
- Optimized object detection and tracking.
- We have given a path to drone.
- Tello drone follows that path by using Python code.
- If anywhere drone misses its path it uses computer vision to correct its path.
- And by reaching end of the path it collects data and passes it to cloud.
- Values will be updated on the website.

Given Drone path

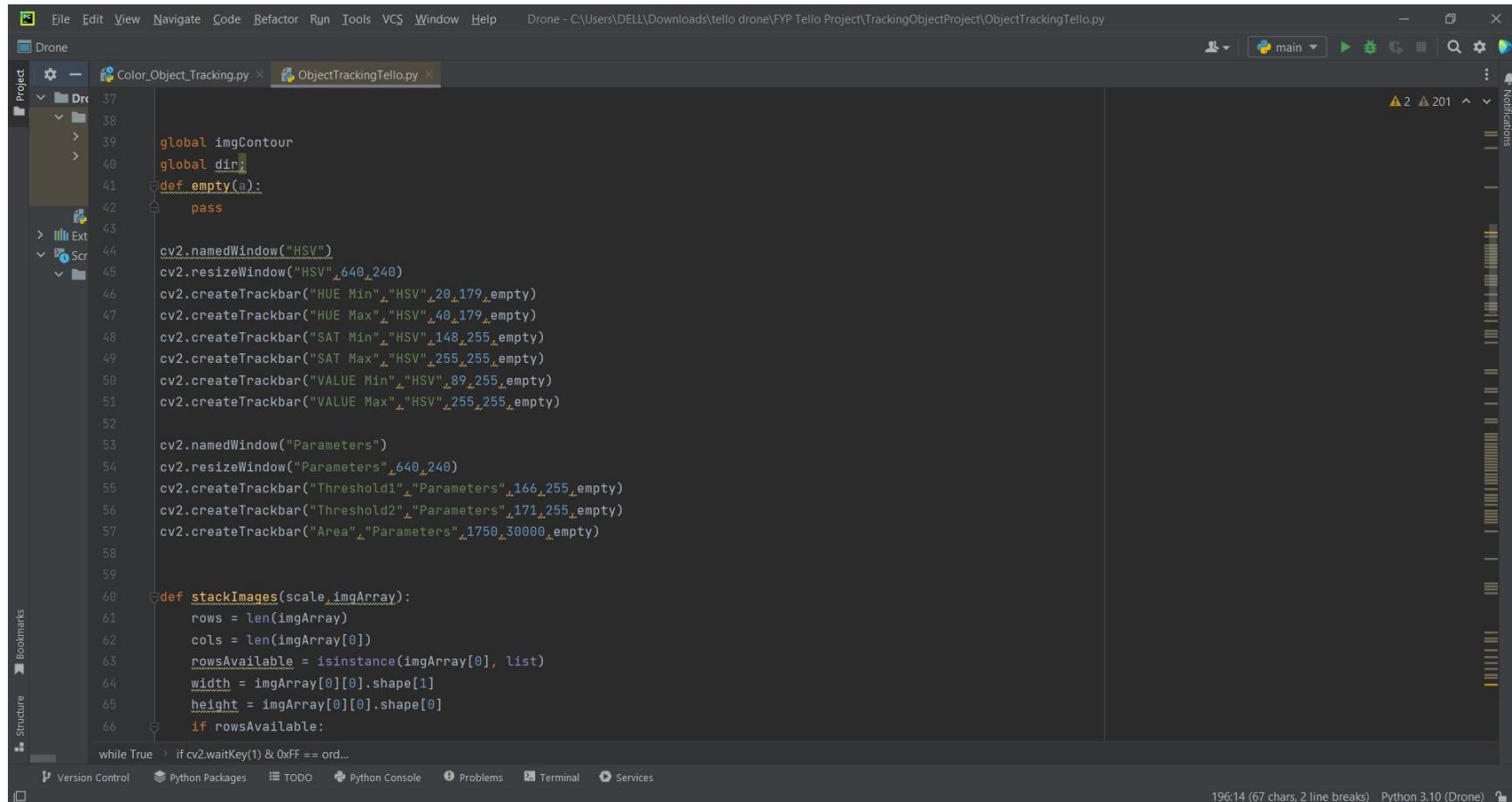


Methodology

- First 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.
- Will blur together within the largest area violet color and draw a boundary.
- Using offset coordinates of the center of the object and the center of the frame.
- The drone should move either up, down, left, or right by observing the sign of the velocity.



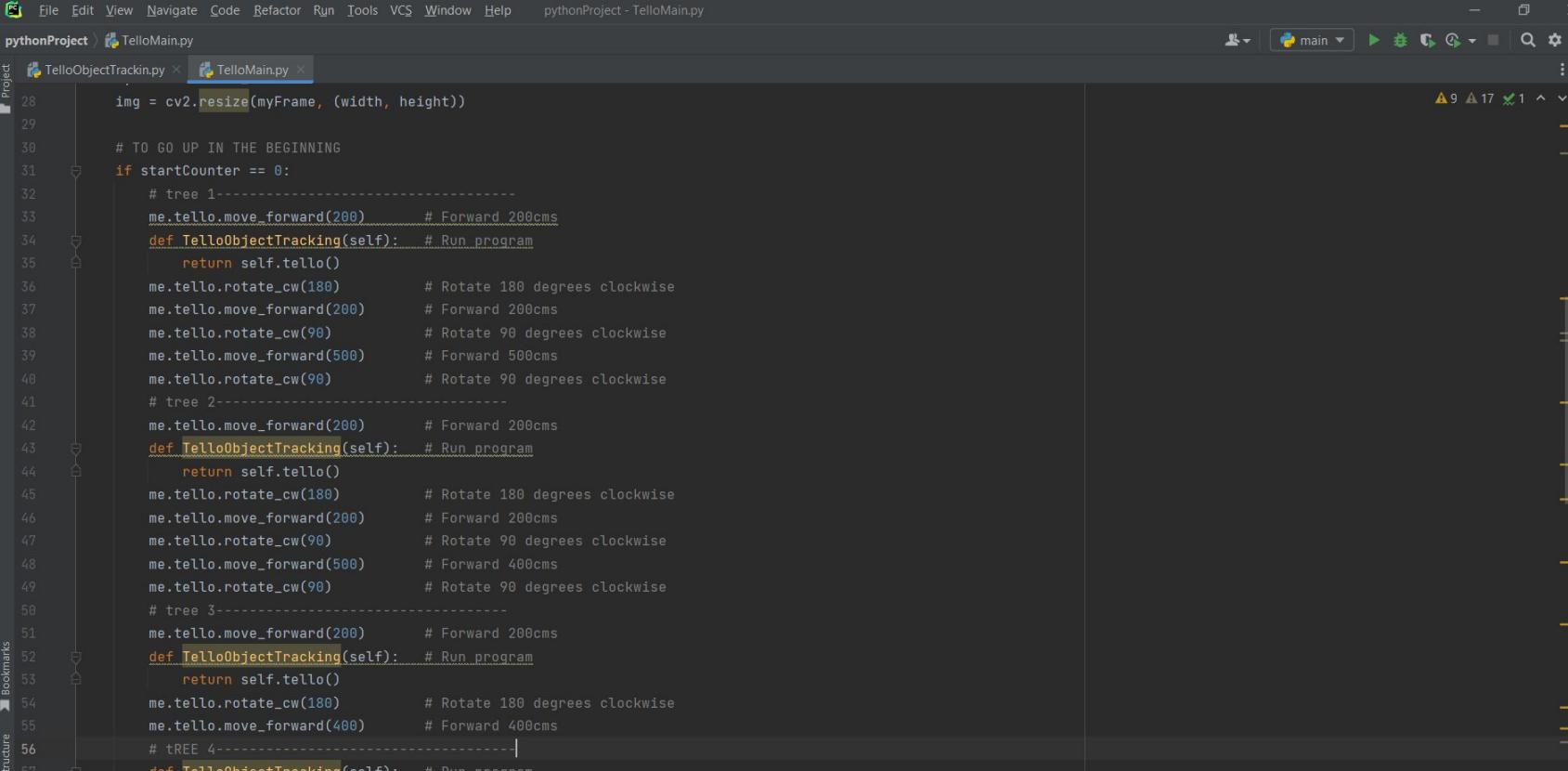
Python Code:



The screenshot shows a Python code editor interface with the following details:

- Project:** Drone
- Files:** Color_Object_Tracking.py (active tab), ObjectTrackingTello.py
- Code:** The code is a script for object tracking using OpenCV. It includes:
 - Global variable declarations for `imgContour` and `dir`.
 - A `def empty(a): pass` block.
 - Code to create a `HSV` window and trackbars for HUE, SAT, and VALUE channels.
 - Code to create a `Parameters` window and trackbars for Threshold1, Threshold2, and Area.
 - A `def stackImages(scale, imgArray):` function.
 - Loop logic to wait for a key press (code: `if cv2.waitKey(1) & 0xFF == ord...`)
- Editor Status:** 2 errors (indicated by a red icon), 201 lines of code.
- Bottom Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services.
- Right Sidebar:** Notifications, a vertical ruler, and a color palette.

Python Code:



File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - TelloMain.py

pythonProject > TelloMain.py

TelloObjectTracking.py < TelloMain.py

```
28     img = cv2.resize(myFrame, (width, height))
29
30     # TO GO UP IN THE BEGINNING
31     if startCounter == 0:
32         # tree 1-----
33         me.tello.move_forward(200)      # Forward 200cms
34         def TelloObjectTracking(self): # Run program
35             return self.tello()
36         me.tello.rotate_cw(180)        # Rotate 180 degrees clockwise
37         me.tello.move_forward(200)    # Forward 200cms
38         me.tello.rotate_cw(90)        # Rotate 90 degrees clockwise
39         me.tello.move_forward(500)    # Forward 500cms
40         me.tello.rotate_cw(90)        # Rotate 90 degrees clockwise
41         # tree 2-----
42         me.tello.move_forward(200)    # Forward 200cms
43         def TelloObjectTracking(self): # Run program
44             return self.tello()
45         me.tello.rotate_cw(180)        # Rotate 180 degrees clockwise
46         me.tello.move_forward(200)    # Forward 200cms
47         me.tello.rotate_cw(90)        # Rotate 90 degrees clockwise
48         me.tello.move_forward(500)    # Forward 400cms
49         me.tello.rotate_cw(90)        # Rotate 90 degrees clockwise
50         # tree 3-----
51         me.tello.move_forward(200)    # Forward 200cms
52         def TelloObjectTracking(self): # Run program
53             return self.tello()
54         me.tello.rotate_cw(180)        # Rotate 180 degrees clockwise
55         me.tello.move_forward(400)    # Forward 400cms
56         # tREE 4-----|
57         def TelloObjectTracking(self): # Run program
```

while True > if startCounter == 0

Version Control Python Packages TODO Python Console Problems Terminal Services

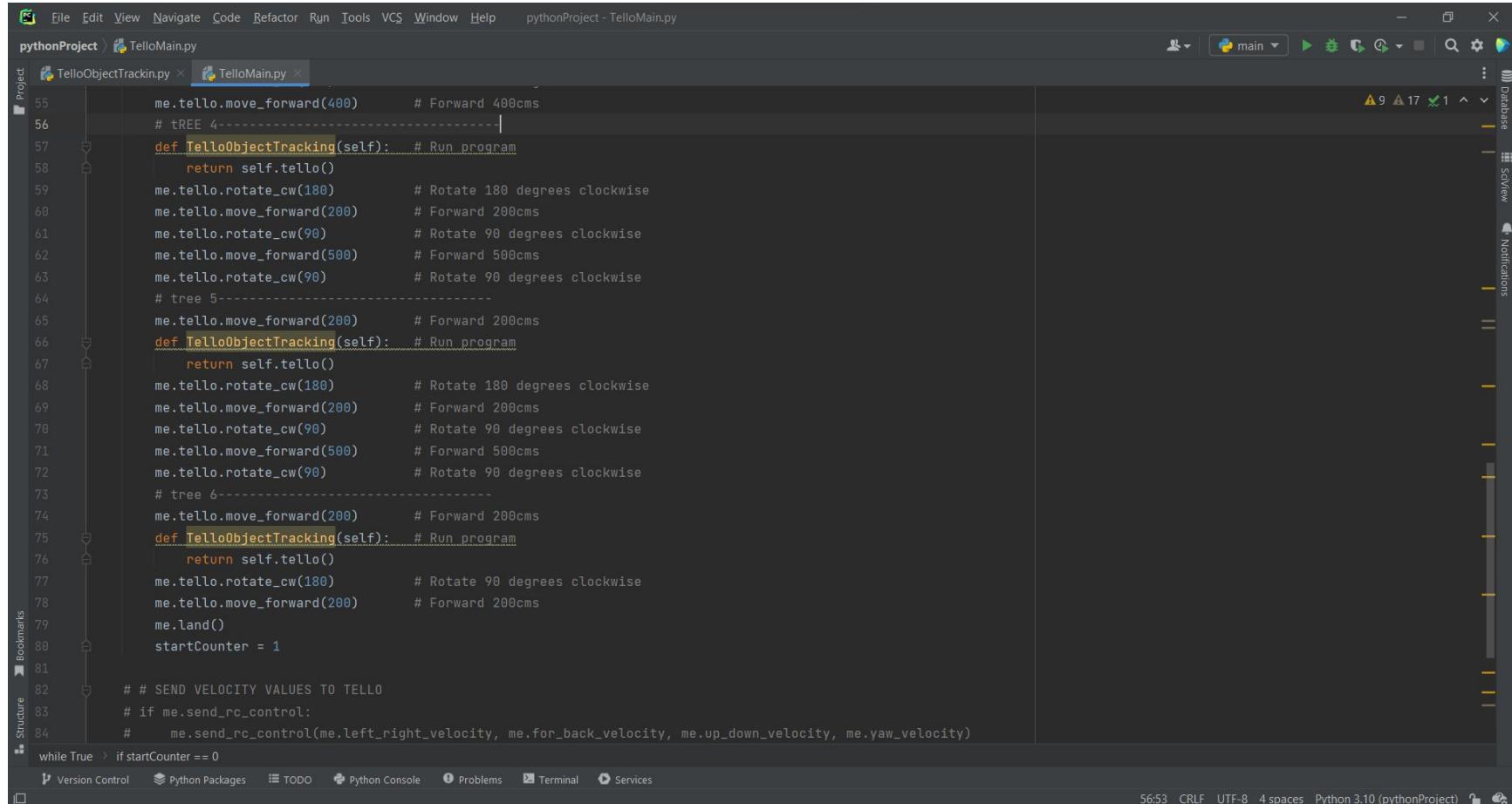
9 17 1

Database

SCView

Notifications

Python Code:



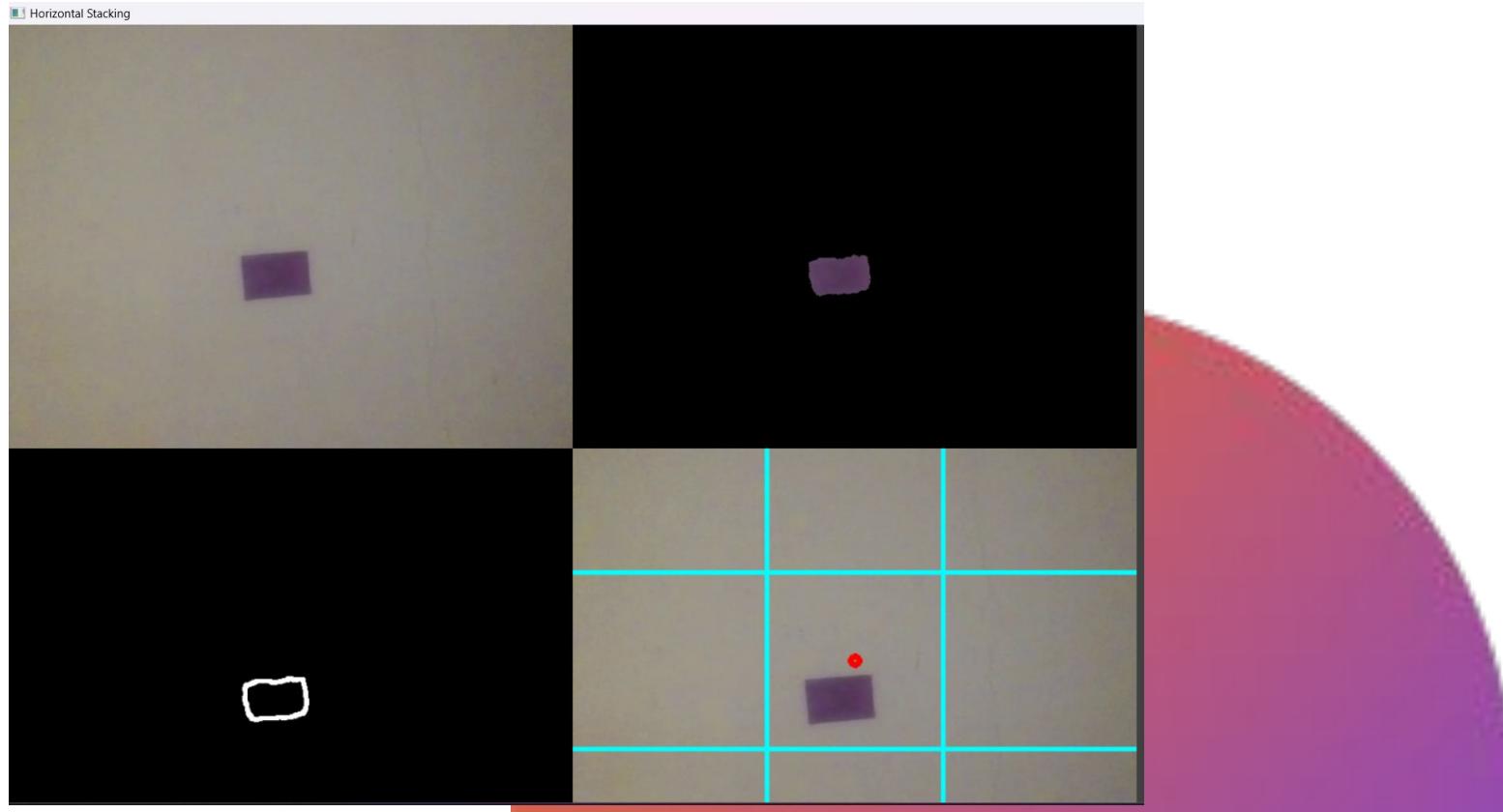
The screenshot shows a Python code editor interface with the following details:

- Project:** pythonProject
- File:** TelloMain.py
- Code Content:**

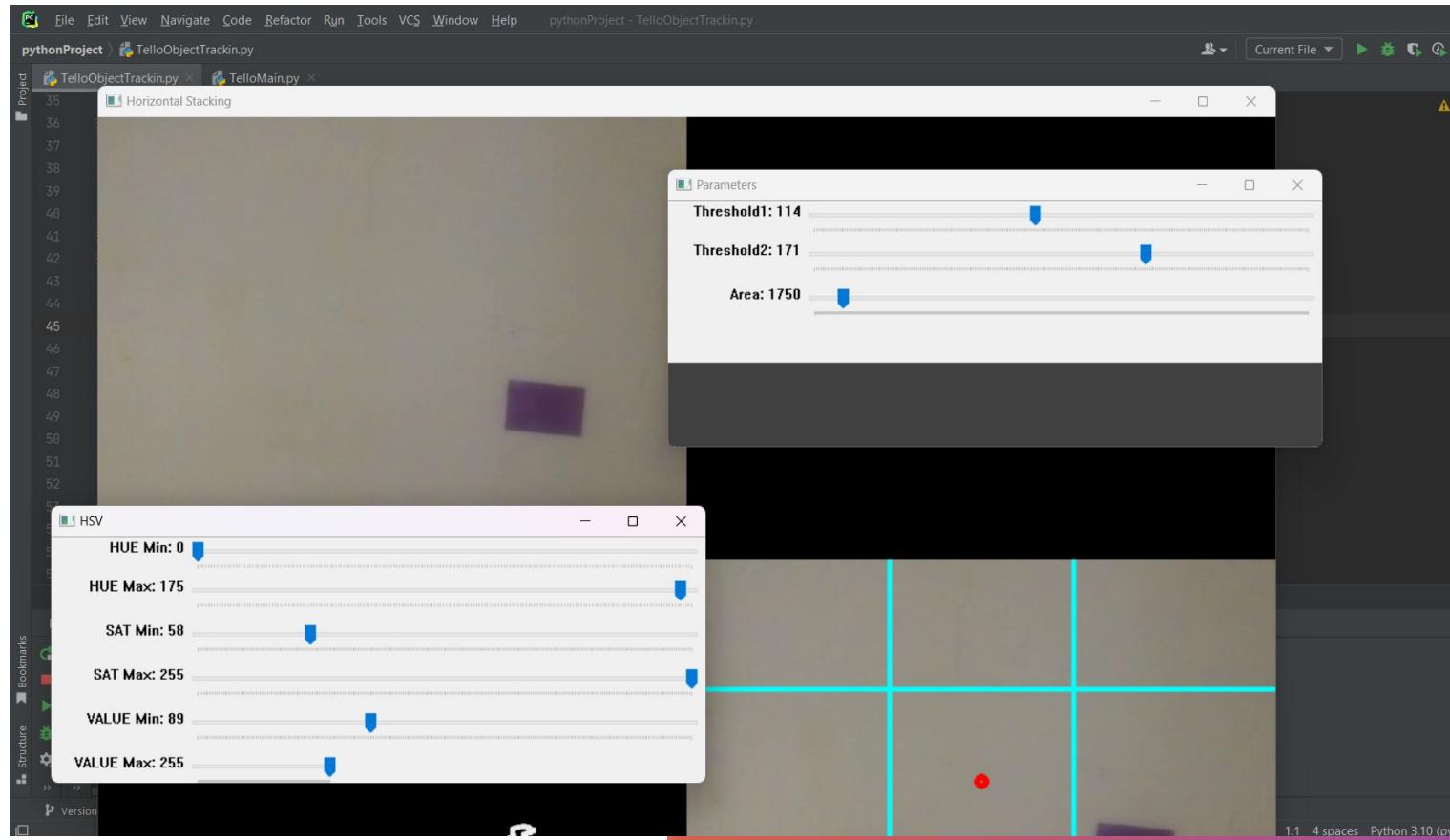
```
55     me.tello.move_forward(400)      # Forward 400cms
56
57     def TelloObjectTracking(self): # Run program
58         return self.tello()
59
60         me.tello.rotate_cw(180)      # Rotate 180 degrees clockwise
61         me.tello.move_forward(200)    # Forward 200cms
62         me.tello.rotate_cw(90)       # Rotate 90 degrees clockwise
63         me.tello.move_forward(500)    # Forward 500cms
64         me.tello.rotate_cw(90)       # Rotate 90 degrees clockwise
65
66         # tree 5-----
67
68         me.tello.move_forward(200)    # Forward 200cms
69         def TelloObjectTracking(self): # Run program
70             return self.tello()
71
72             me.tello.rotate_cw(180)      # Rotate 180 degrees clockwise
73             me.tello.move_forward(200)    # Forward 200cms
74             me.tello.rotate_cw(90)       # Rotate 90 degrees clockwise
75             me.tello.move_forward(500)    # Forward 500cms
76             me.tello.rotate_cw(90)       # Rotate 90 degrees clockwise
77
78             # tree 6-----
79
80             me.tello.move_forward(200)    # Forward 200cms
81             def TelloObjectTracking(self): # Run program
82                 return self.tello()
83
84                 me.tello.rotate_cw(180)      # Rotate 90 degrees clockwise
85                 me.tello.move_forward(200)    # Forward 200cms
86
87                 me.land()
88
89                 startCounter = 1
90
91
92             # # SEND VELOCITY VALUES TO TELLO
93             # if me.send_rc_control:
94                 #     me.send_rc_control(me.left_right_velocity, me.for_back_velocity, me.up_down_velocity, me.yaw_velocity)
95
96 while True > if startCounter == 0
```

- Toolbars and Status Bar:** The status bar at the bottom shows the time as 56:53, encoding as CRLF-UTF-8, and the Python version as 3.10 (pythonProject). Other tabs include Version Control, Python Packages, TODO, Python Console, Problems, Terminal, and Services.

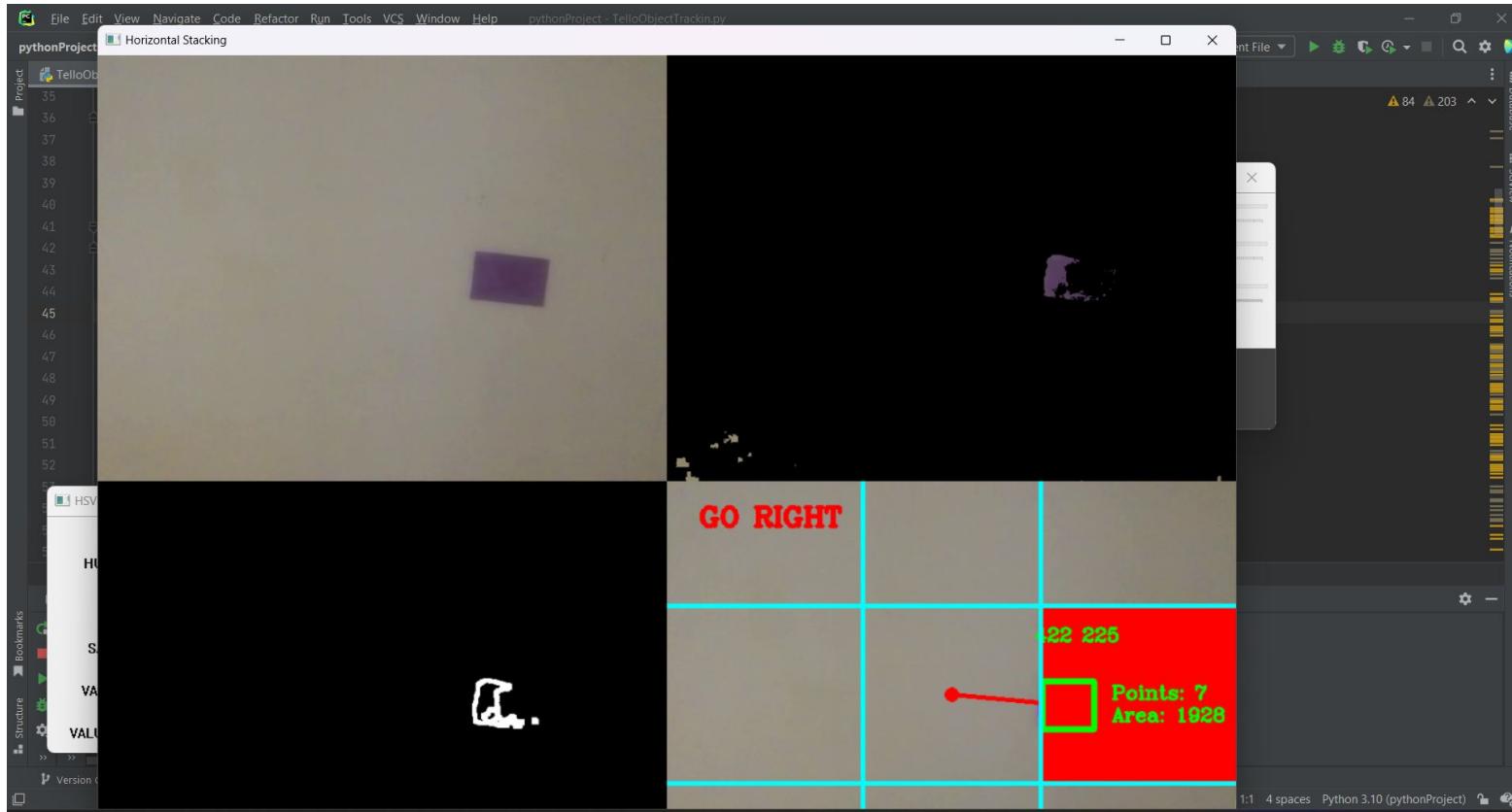
Python Results:



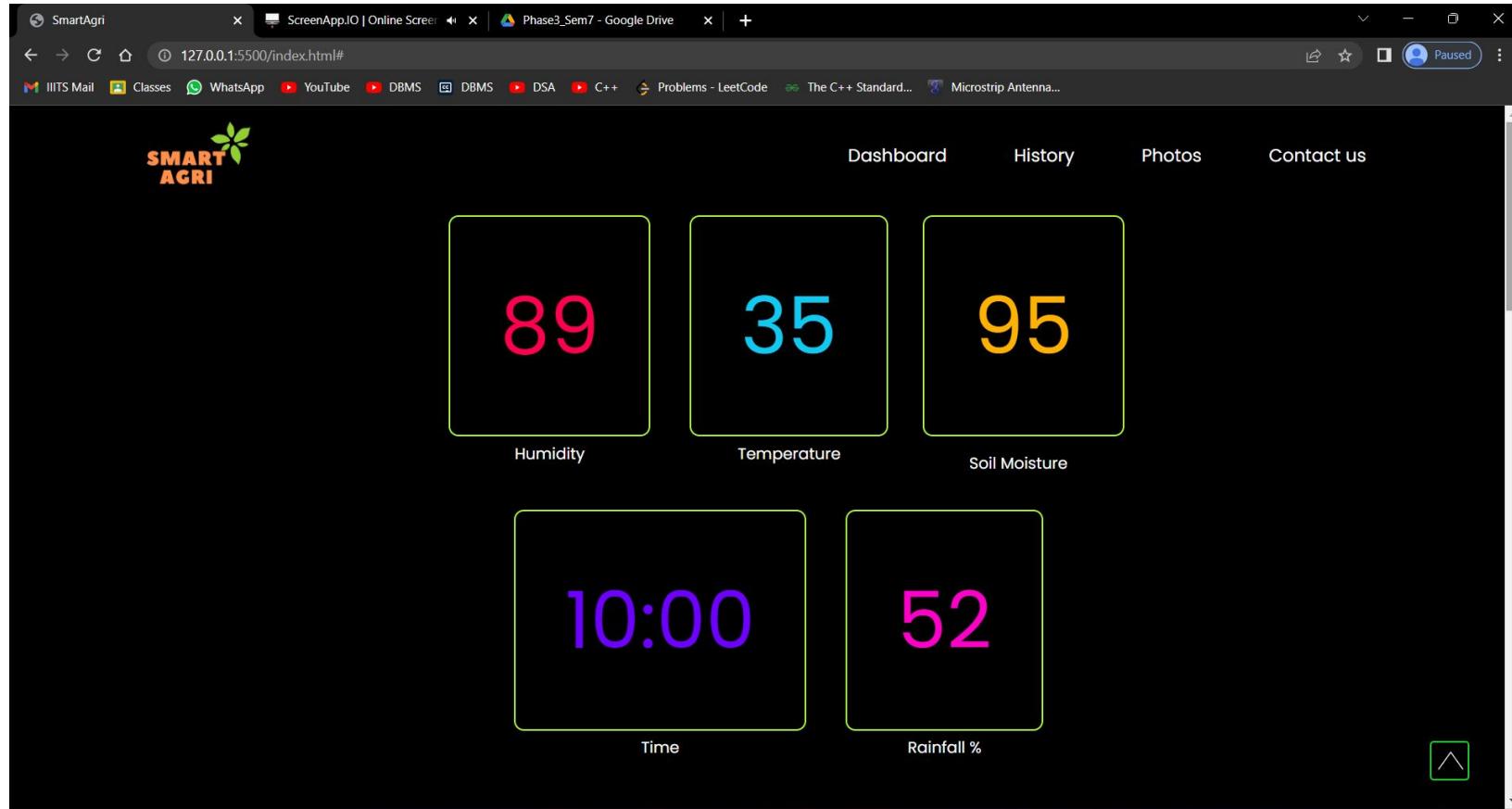
Python Results



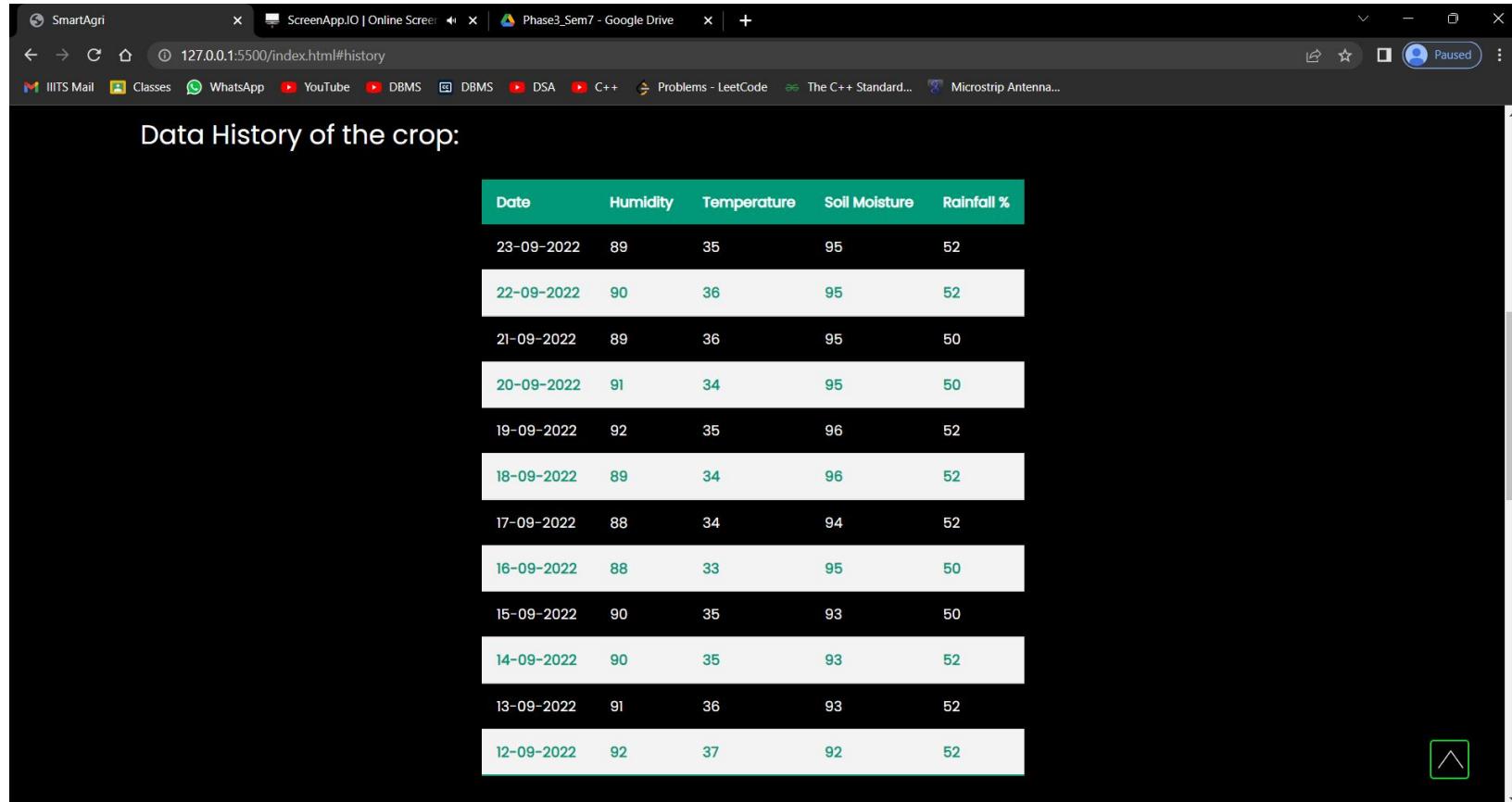
Python Results



Website Screenshots:



Website Screenshots:



The screenshot shows a web browser window with a dark theme. The title bar includes tabs for "SmartAgri", "ScreenApp.IO | Online Screen", and "Phase3_Sem7 - Google Drive". The address bar shows the URL "127.0.0.1:5500/index.html#history". The page content is titled "Data History of the crop:" and displays a table with the following data:

Date	Hurnidity	Temperature	Soil Moisture	Rainfall %
23-09-2022	89	35	95	52
22-09-2022	90	36	95	52
21-09-2022	89	36	95	50
20-09-2022	91	34	95	50
19-09-2022	92	35	96	52
18-09-2022	89	34	96	52
17-09-2022	88	34	94	52
16-09-2022	88	33	95	50
15-09-2022	90	35	93	50
14-09-2022	90	35	93	52
13-09-2022	91	36	93	52
12-09-2022	92	37	92	52



Video link:

[Click here to Watch](#)

Links:

Python code link:

<https://github.com/litheeshkumar/IoT-Connectivity-for-Drone-applications/tree/main/Python%20Codes>

Website source code link:

<https://github.com/litheeshkumar/IoT-Connectivity-for-Drone-applications/tree/main/SmartAgri%20Website>

Google drive link for pics and video taken by Tello:

https://drive.google.com/drive/folders/1LUn7OpOfWLix3Sq-68jW5SKrzOUXE_A9?usp=share_link

Challenges faced:

- The issues of dying batteries and overheating.
- If the battery life of the Tello is too low, the drone will not take flight.
- Before running code, charge all battery packs and have replacements if drone will not lift off.
- The DJI Tello will not take flight if it is overheating.
- Landing of drone is one of the main problem - placing a notebook as shown - as a launch surface.
- Finally, several libraries must be downloaded to run the Tello with CV2.



Future Scope:

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



References:

1. Jeffrey Kantor. CBE 30338 Visual Tracking of an Object with a Drone. *IEEE Access* 2019, 7, 129551–129583. [[CrossRef](#)]
2. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. [[CrossRef](#)]
3. AlDahoul, N., Sabri, A. Q., & Mansoor, A. M. (2018). Real-time human detection for aerial captured video sequences via deep models. *Hindawi*, 2018, 15. [[CrossRef](#)]
4. S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. [[CrossRef](#)]



Thank You