University of New Haven

CAPSTONE PROJECT

DRONE DETECTION

DSCI-6051-02

SUBMITTED BY:
KAVYASREE MOKTALA
TANOJ KUMAR ANAPANA
SAI DEEPAK DEVADARI

SUBMITTED TO:
DR.SHAYOK MUKHOPADHYAY

MAY 11,2025.

Contents

DRONE DETECTION

KAVYASREE MOKTALA
Tagliatela College of Engineering
University of New Haven
West Haven, Connecticut, USA
kmokt1@unh.newhaven.edu

TANOJ KUMAR ANAPANA
Tagliatela College of Engineering
University of New Haven
West Haven, Connecticut, USA
tanap1@unh.newhaven.edu

SAI DEEPAK DEVADARI
Tagliatela College of Engineering
University of New Haven
West Haven, Connecticut, USA
sdeva16@unh.newhaven.edu

***Abstract*—This project presents the design and implementation of a real-time drone detection system aimed specifically at identifying the presence of drones in the environment without any other objects or any security measures. This presents a YOLOv8(You Only Look Once version 8) – based object detection model trained on a custom dataset for accurate drone recognition. The system is built using Raspberry Pi with a camera module and an ultrasonic sensor for drone detection. The trained model detects drones in images by drawing bounding boxes and by labeling them clearly. The goal is to deploy this system on a Raspberry Pi connected to a camera module and ultrasonic sensor for real-time drone detection.**

Keywords—Drone Detection, Raspberry pi, Ultrasonic sensor, Object detection, YOLO, Real-time detection, Roboflow.

I. INTRODUCTION

Drones have become widespread in various fields like photography, surveillance, agriculture and other. Even after the drones are widely used, they are also causing challenges in security, privacy, research and monitoring. Unidentified drone activity can cause serious threats. Thus, developing a accurate real-time drone detection system is necessary.

This project aims to develop a real-time drone detection system using YOLOv8 (You Only Look Once version 8) object detection model. YOLOv8 is the latest version of the YOLO family and it is known for its high-speed, high-accuracy object detection capabilities. A custom dataset was collected consisting the drone and non-drone images, which were later annotated and used to train the model. The training process was conducted in Google Colab. The model was evaluated with the test images to verify its performance. We used this data to train the YOLOv8 model to detect the drone accurately. The system successfully identifies the drone with bounding boxes and labels around the detected objects accurately.

The final goal is to deploy the trained model on a Raspberry Pi connected to a camera module and an ultrasonic sensor. This makes the solution easier for real-time deployment. The model performance was tested on various conditions and found its accurate detection capabilities . That means this model can be used for security, research and monitoring.

INDIVIDUAL CONTRIBUTION OF THE TEAM MEMBERS

Tanoj Kumar Anapana

Project Overview:

- The drone detection project aims to identify and track drones in various environments using a combination of sensors and cameras. The goal is to enhance safety and security by developing a reliable detection system.

Initial Research and Planning:

- Conducted a thorough review of existing drone detection systems to understand their methodologies and technologies.
- Analyzed various approaches to object detection,which helped in formulating a plan for our project.
- Identified the key components needed for the project and how they would interact to achieve the desired outcomes.

Level-0

FIG:1

Component Familiarization:

- Gained hands-on experience with essential components such as:
  Arduino: Understood its programming and input/output capabilities.
  Ultrasonic sensor: Learned how it measures distance by sending out sound waves.
  Raspberry Pi Camera: Familiarized with its integration for image capture and processing.
- Studied wiring diagrams and tutorials to ensure proper connections on a breadboard.

Prototype Development:

- Successfully created a small prototype to test the integration of the Arduino with the ultrasonic sensor.
- Developed a simple program that triggers an LED light to blink when an object is detected, demonstrating the basic functionality of the system.
- Troubleshoot connection issues, enhancing my problem-solving skills and understanding of circuit design.

Advancing Skills with Raspberry Pi:

- Dedicated time to learning more about the Raspberry Pi, focusing on its operating system and programming environment.
- Integrated the Raspberry Pi camera into the project to capture images of detected objects, enhancing the systems capabilities beyond simple detection.

Object Detection Focus:

- Concentrated efforts on the critical aspect of object detection, recognizing its importance for the projects success.
- Collected extensive datasets of drone images accurately, ensuring that the model would be trained on high-quality data.
- Utilized LabelImg to label the images accurately, ensuring that the model would be trained on high-quality data.
- Organized the dataset into training, testing, and validation sets to facilitate effective model training and evaluation.

Model Training:

- Currently developing the code necessary for implementing the object detection algorithm.
- Selected YOLOv5 as the framework for training the model due to its efficiency and accuracy in real-time object detection.
- Engaged in iterative testing and refinement of the model to improve its performance in accurately classifying images as containing a drone or not.

Challenges Faced:

- Initially labeled images as “drone with blades” and “drone with human” but achieved low accuracy.
- Realized that combining these labels was ineffective.

Improved Approach:

- Labeled drone and hand separately using labelImg for better model performance.
- Collected and labeled images for drone and images for hands.

Outcome:

- The combined detection of drones and hands allows the model to identify whether a drone is airborne or held by a hand.

Kavya Sree Moktala

1) *Research and knowledge gaining:*

a) Understanding drone detection:

- Explored different methods for detecting drones like computer vision and sensor based methods.
- Studied research papers and drone detection models to understand the strengths and weaknesses of various approaches.
- Learned about object detection model to detect the drones in real-time.
- Studied various real-world drone detection systems and analyzed their challenges, such as:
- Detecting small drones at a distance
- Handling different lighting and weather conditions.
- Making sure that the model only identifies the correct object we are looking for, rather than birds, planes and etc.

b) Understanding Components:

- Researched and understood how Raspberry Pi can be used for drone detection.
- Learned how Arduino interfaces with sensors and Raspberry Pi.
- Studied how ultrasonic sensors work to detect object distance.
- Learned how to integrate ultrasonic sensor (HC-SR04) with Raspberry Pi.
- Understood the breadboard and its importance in circuit testing.
- Learned about camera modules for Raspberry Pi and how they can be used for object detection.
- Understood how to connect Raspberry Pi and ultrasonic sensors.

2) *Hardware setup and configuration:*

- Connecting the Ultrasonic Sensor to measure the distance of detected objects.
- Find the ways how to setup ultrasonic sensor.
- Ensured that distance data from the ultrasonic sensor was sent to the Raspberry Pi.
- Knew about HC-SR04 Sensor.
- After setting up the hard wares, then proceeded to implement real-time object detection algorithms.

3) *Dataset Creation, Annotation & Model Training:*

- To have some prior knowledge of how actually all the process happens, I worked on creating custom dataset.
- Gathered images from multiple sources to create a dataset including both drone and non-drone images.
- Labelled images using Roboflow for YOLO training.
- Ensured dataset had varied lighting, backgrounds, and drone sizes.
- Uploaded dataset to Google Colab and used yolov8 for training.
- Used transfer learning to fine-tune the model for drone detection.
- Adjusted model parameters to improve precision, recall, and F1-score.
- Evaluated the real-time performance.

4) Further works: (After Mid)

- After several attempts using other YOLO models, found YOLOv8 is compatible for accurate results in object detection and chose the model.
- Now, we have created a real custom dataset containing both drone and hand images.
- Then labelled and annotated the drones, hands and blades of the drones using labelImg.
- This resulted in low accuracy than what we actually require.
- Later we considered only "drones" and "hands", labelled them and gained the good accuracy that a model can perform.
- Trained a YOLOv8 model on google colab using the custom dataset.
- Combined both the drone and hand models to identify whether the detected image is drone or not.
- Evaluated the model using the test images and made the required changes to acquire the required detection accuracy.
- Got the real-time images captured using the Raspberry Pi camera module from our teammate, tested them in the model we have created.

5) Challenges faced:

- I have got both the drone and hand images output, but I did not gain the labels for both the images together to identify whether the output image is drone or hand.
- As we were close to submission, I further moved forward with my team mates work, as he got the both images labelled.
- The reason we found non accurate results is because, the data I trained might not have similar images (different angles, lighting conditions, size) compared to real-time images captured from camera module.

Sai Deepak

1. *Formatted and Set Up the Raspberry Pi*

- Installed the 32-bit Bookworm OS to ensure compatibility with required libraries and hardware components.

2. *Connected the Raspberry Pi to a Laptop*

- Established communication via SSH or MATLAB for remote control, debugging, and code execution.

3. *Installed System Updates and Libraries*

- Downloaded and configured essential updates and libraries (e.g., OpenCV, TensorFlow Lite, etc.) for image processing, object detection, and sensor integration.

4. *Hardware Integration*

- Connected the webcam and ultrasonic sensor to the Raspberry Pi, ensuring stable wiring and signal integrity.

5. *Code Development*

Developed code for:

  - Image capture via the webcam, saving images directly on the Raspberry Pi.
  - Distance measurement using the ultrasonic sensor with precise range calculations.
  - Object detection and image capture triggered when objects are detected within 300 cm to 180 cm.

6. *Circuit Design and Integration*

- Designed and implemented the circuit diagram for connecting the Raspberry Pi to the ultrasonic sensor using jumper wires, a breadboard, and a voltage divider built with three 1-ohm resistors.

7. *Assemble the Drone Components*

- Begin by assembling all parts of the drone carefully according to the manufacturer's instructions. Ensure that all propellers, motors, frames, and electronic boards are securely attached.
- Check Battery Status and Safety: Verify that the drone battery is full charged and securely connected. Inspect the battery for any signs of damage, swelling, or leakage before each use to avoid potential hazards.

8. *Calibrate Sensors*

  - Calibrate all onboard sensors, including the gyroscope, accelerometer, and magnetometer, to ensure accurate and stable flight control.

9. *Learn Basic Flight Controls*

  - Understand and practice using the fundamental controls:
  - Throttle: Controls altitude(up/down)

- Yaw: Rotates the drone clockwise or counterclockwise.
- Pitch: Tilts the drone forward or backward.
- Roll: Tilts the drone left or right.
- Ensure a Safe Flying Environment: Choose a clear, open indoor space for flight practice. Use propeller guards to minimize damage and prevent injury. Keep the are free of people, pets, and fragile objects.

10. *Install Control Software*

- Download and install the Crazyflie PC client or the Crazyflie mobile app(Android only) to interface with and control the drone.

11. *Test Communication and Perform a Dry Run*

- Before actual flight, verify a stable communication link between the controller and the drone. Perform a dry run by spinning the motors without takeoff to confirm proper responsiveness and function.

II. METHODOLOGY

A. Data Preparation

To train a model for drone detection model, a dataset containing images of both drones and non-drone is collected. This dataset was collected from publicly available platforms including Kaggle and Google Images and with various types of drones in different environments and lighting conditions to improve detection accuracy. Non-drone images, are included to help the model distinguish drones from other objects.

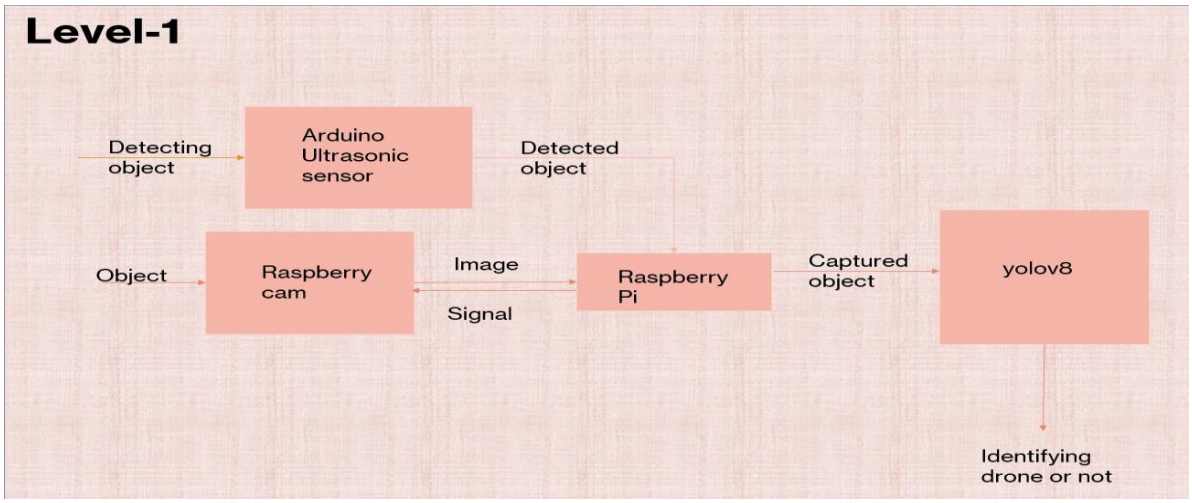

FIG:2

B. Data Preprocessing and Augmentation

Before training the model, the images need to be pre-processed to ensure that all images have the same size, resolution, and format. Annotating is done using LabelImg platform, where bounding boxes are drawn around the drones in the images and label them correctly. The annotated dataset was then exported in YOLO format, which can directly be usable by the YOLOv8 training pipeline. This step is important for training the model to recognize drones accurately. The dataset is then split into training, validation, testing sets to evaluate model performance.

III. MODEL SELECTION

The drone detection system is developed using YOLOv8(You Only Look Once version 8), a deep learning model designed for high-speed and accurate object detection.YOLOv8 is known for its efficiency in detecting objects in real-time across a variety of environments.

The model is trained on a labeled dataset containing both drone and non-drone images, allowing it to learn various features of drones from different distances, backgrounds and angles. After completing the training phase, the model is tested on the new images to verify the detection accuracy and the real-time performance.

A. Model Training:

The labeled dataset is prepared by resizing the images to match the YOLOv8 model requirements. The architecture of YOLOv8 model consists of the below written components:

1) Backbone: A CNN-based feature extractor optimized for faster and more accurate feature learning.

2) Neck: A feature fusion network, such as PANet(Path Aggregation Network),which strengths feature representation across multiple scales.

3)Head: A prediction head that generates bounding boxes, confidence scores, and class probabilities for each detected object.

B. Fine-tuning and Evaluation:

After the initial training process, the model undergoes the fine-tuning to further enhance its performance. This involves applying data augmentation techniques such as flipping, rotation, brightness adjustment, and cropping to create more training examples.

The performance of the trained model is evaluated using object detection metrics:

- Precision: The accuracy of positive predictions.
- Recall: The ability of the model to detect all relevant drones.
- Mean Average Precision(mAP):A comprehensive score that balances precision and recall across different thresholds.

If the performance is not satisfactory, adjustments are made to improve the model.

After successfully training and fine-tuning the YOLOv8 model, the next step involves deploying it to achieve the real-time drone detection.

The optimized model is then loaded on to Raspberry Pi, which is connected to a camera module for a live image capture and an ultrasonic sensor for the distance measurement. The camera captures the real-time frames, which are fed into the YOLOv8 model which is running locally on the Raspberry Pi .The model will processes the every frame then identifies the presence of drones and draws the bounding boxes with their corresponding labels around the detected objects.

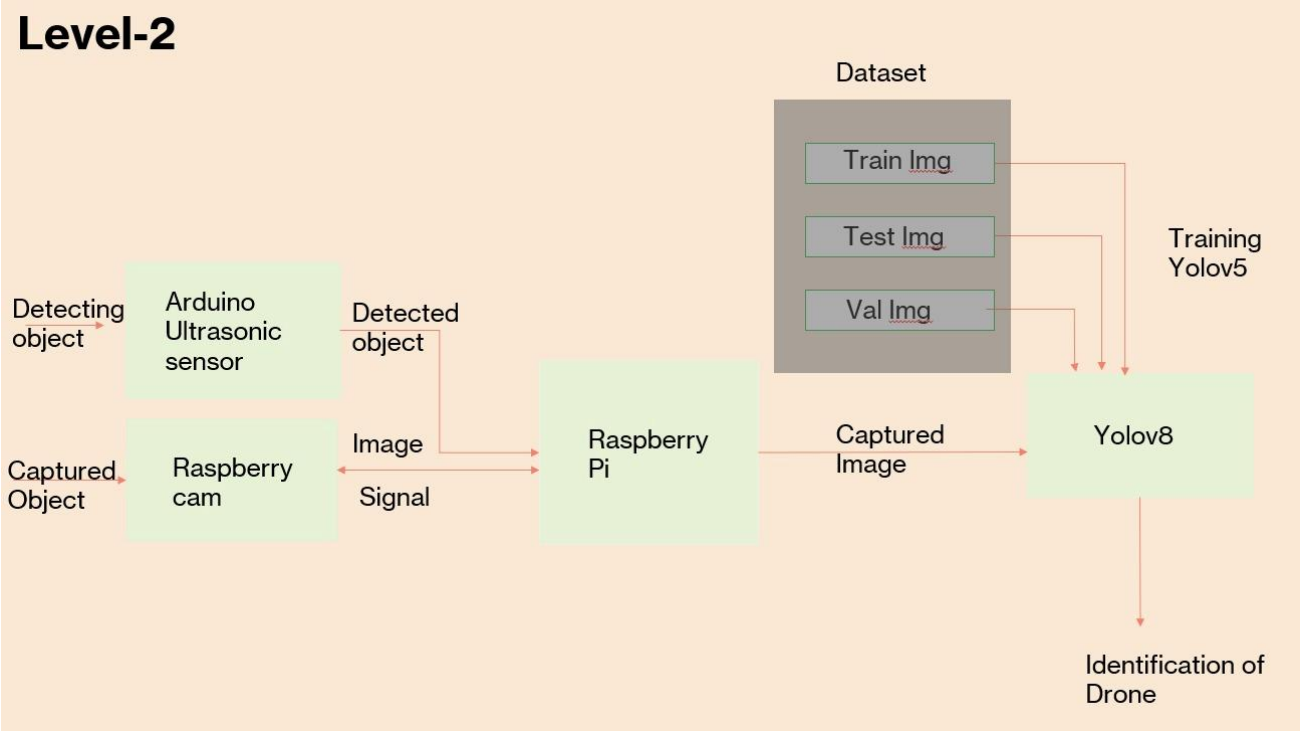

FIG:3

The trained model is optimized for real-time performance on the Raspberry Pi by converting it to a lightweight format to run efficiently on the edge device.

The optimized model is converted into light-weight version compatible with raspberry pi. The final model runs on the raspberry pi, detecting drones in real-time using the camera and ultrasonic sensor. The system is then tested in different environments to ensure reliable drone detection under various conditions.

By the end of the training process, the Yolov8 model is capable of detecting drones accurately in real-time.

C. Deployment:

Custom dataset has 2494 drone images and 3828 hand images and ---------}images of images where both hand and drone are present in the images. We have collected the datasets from the internet.

The drone dataset has 2337 train and 157 valid images. And the hand dataset has 3174 train and 654 valid images. Also added test ----------}images in YOLOv8 model training.

The images are captured from different angles like from the top, bottom, and from side angle. There are 10 images captured from each angle and these are now tested on our YOLOv8 model to gain the required output.

The system is tested under various environments to ensure its robustness,accuracy and the speed.If any false positives or the wrong detections are made during the testing, the minor adjustments are made to the model and sometimes the detection thresholds if necessary.

The final deployed gives the real-time drone detection system capable of operating in different environments showing how well the YOLOv8 model works when used in the real-world situations.

IV. CONNECTIONS AND FLOWCHARTS

- We are able to make connections using the hardwares: Raspberry Pi, Camera Module, Ultrasonic sensor. An are able to capture the images of the objects.
- The below the connections we made:

A. Connections:

Wiring the ultrasonic sensor to Raspberry Pi,

Raspberry Pi as the Main Controller:

HC-SR04 Pin	Raspberry Pi GPIO Pin
VCC	5V
GND	GND
TRIG	GPIO 23
ECHO	GPIO 24

TABLE:1

Raspberry Pi Pin	Arduino Pin
GPIO 14(TXD)	RX(Pin 0)
GPIO 15(RXD)	TX(Pin 1)
GND	GND

TABLE:2

Pi Camera	Raspberry Pi
CSI Ribbon Cable	CSI Port

TABLE:3

Components and Setup:

Hardware Components:

- Raspberry Pi 4: Runs Bookworm OS and processes data.
- HC-SR04 Ultrasonic Sensor: Measures object distances using ultrasonic waves.
- Webcam: Captures visuals for drone detection.
- Voltage Divider: Three 1-ohm resistors step down the Echo pin voltage to 3.3V for Raspberry Pi compatibility.

Voltage Divider Steup:

- Reduces the 5V Echo output to 3.3V using three 1-ohm resistors before connecting to the Raspberry Pi.

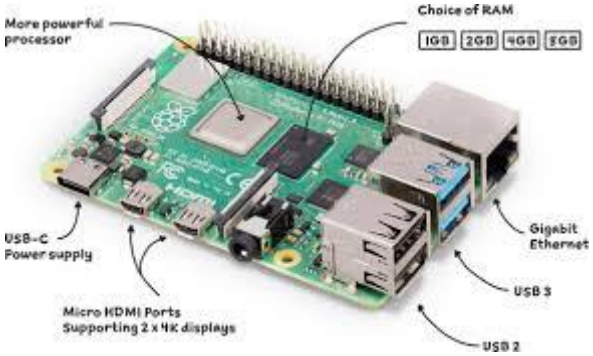

FIG:4

Raspberry Pi 4:

- Specs: Quad-core Cortex-A72 (1.5 GHz), up to 8 GB RAM, USB 3.0, dual micro-HDMI, Gigabit Ethernet, Wi-Fi and Bluetooth.
- Handles: Real-time data and image processing.

HC-SR04 Ultrasonic Sensor:

- Working: Emits a 40 kHz pulse; measures echo return time.
- Specs: 5V, 15 Ma, 2-400cm range, 0.3 cm resolution, 15-degree angle.
- Pins: VCC(5V), GND, Trigger (GPIO 18), Echo (GPIO 24 through voltage divider).

FIG:5

Software Setup:

Required Libraries:

- RPi, GPIO (Ultrasonic sensor interface)
- Time (timing operations)
- cv2(OpenCV for video processing)

Connecting Webcam:

- Plug into a USB port on Raspberry Pi 4.

Code for Image Capturing:

```
import cv2
import time
# Open a connection to the webcam (0 is the default camera)
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open webcam")
    exit()
# Give the camera some time to adjust
time.sleep(2)
# Capture a frame
ret, frame = cap.read()
if ret:
    # Save the image
    filename = f"webcam_image_{int(time.time())}.jpg"
    cv2.imwrite(filename, frame)
```

```
    print(f"Image saved as {filename}")
else:
    print("Error: Could not capture image")
# Release the camera
cap.release()
cv2.destroyAllWindows()
```

Working Principle:

1. The HC-SR04 sensor sends an ultrasonic pulse and measures the time taken for the echo to return.
2. The measured time is converted into distance.
3. If the measured distance is below a specified threshold, the webcam is triggered.
4. The webcam captures images or records video to confirm the presence of a drone.

Object detection:

When the object is detected by the ultrasonic sensor the webcam should capture the image.

The system utilizes a Raspberry Pi along with an ultrasonic sensor and a webcam(or drone camera) to detect objects within a specific range and capture images accordingly. The primary goal is to measure distances and save images when an object is detected within the predefined range.

1. *Ultrasonic Sensor Activation:*

- The ultrasonic sensor continuously scans for objects.

2. *Object Detection:*

- If no object is detected, the system loops back to continue scanning.
- If an object is detected, it moves to the next step.

3. *Distance Measurement Check:*

- If the detected object is within 180cms to 300 cm, the system proceeds.

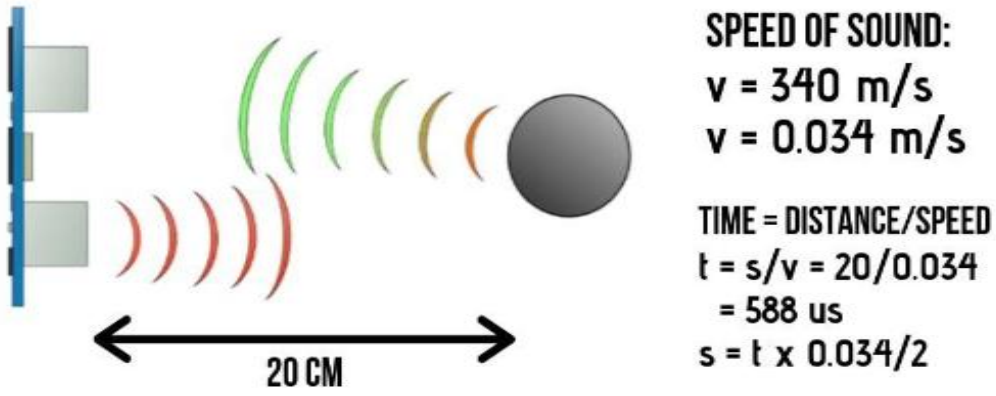

FIG:6

Distance measurement by Ultrasonic Sensor:

Measure the distance to or the presence of a target object by sending a sound pulse, above the range of human hearing(ultrasonic), toward the target and then measuring the time it takes the sound echo to return.

- A drone camera(if available) also captures an image along with distance measurement.

4. Image Storage:

- The captured image is saved to the system.

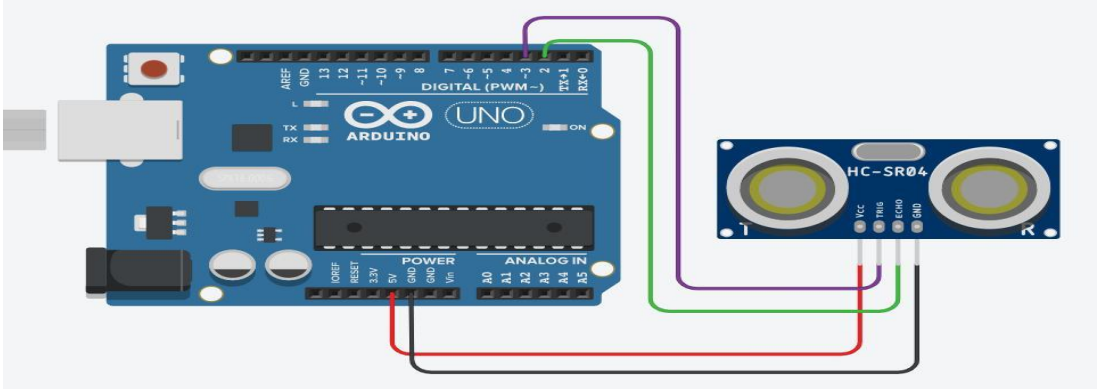

FIG:7

V. COMPONENTS USED

- **Raspberry Pi** – The central processing unit handling sensor data and image capture.
- **Ultrasonic Sensor –** Used for detecting objects and measuring distances.
- **Webcam –** Captures images of detected objects.
- **Drone Camera** (optional) – Captures an aerial image with distance measurement.
- **Storage System –** Saves images for further processing.

VI. APPLICATION AREAS

- Security surveillance
- Autonomous robotics
- Industrial automation
- Distance-based monitoring systems

VII. RESULTS AND EVALUATION

The YOLOv8 model was trained for 50 epochs using the custom dataset. The training was conducted on Google Colab with GPU enabled. The trained YOLOv8 model had successfully detected the drone on both the test images and the real-time camera feeds. The model detected the drone with high accuracy, with bounding boxes and labels drawn around the detected image. The below are the metrics observed:

- Precision
- Recall
- mAP

The system performed accordingly in various environments. The real-time tests on the Raspberry Pi have shown the satisfactory detection speed. The bounding boxes and labels were well aligned with the drones, and false positives were minimal. This indicates the model's efficiency and robustness.

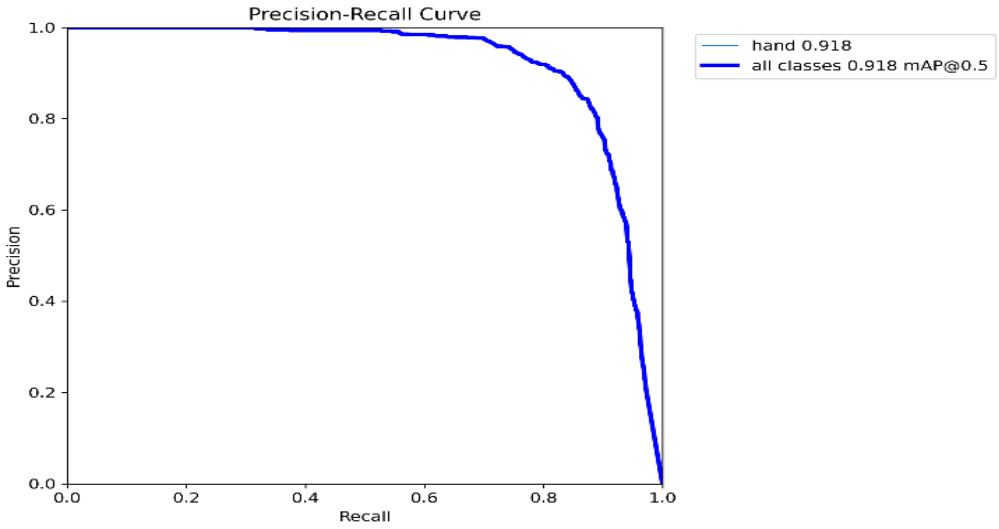

FIG:8

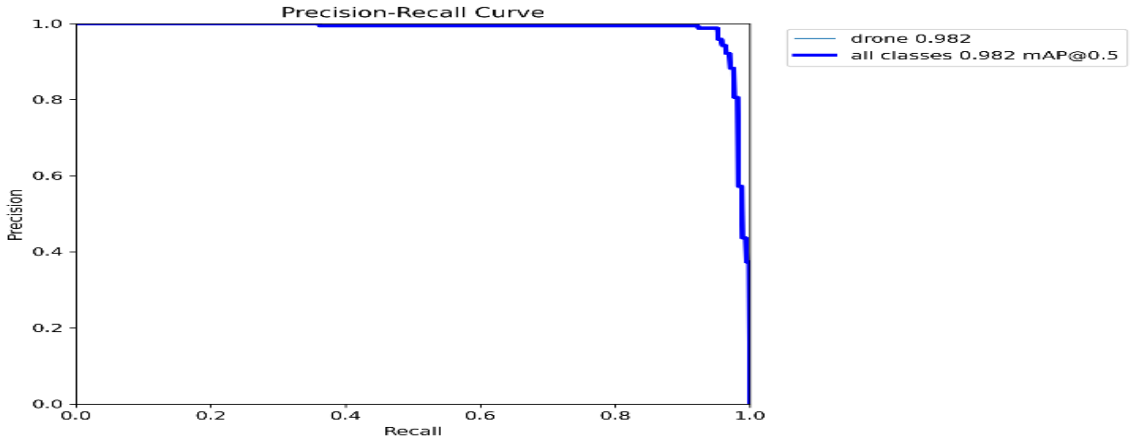

FIG:9

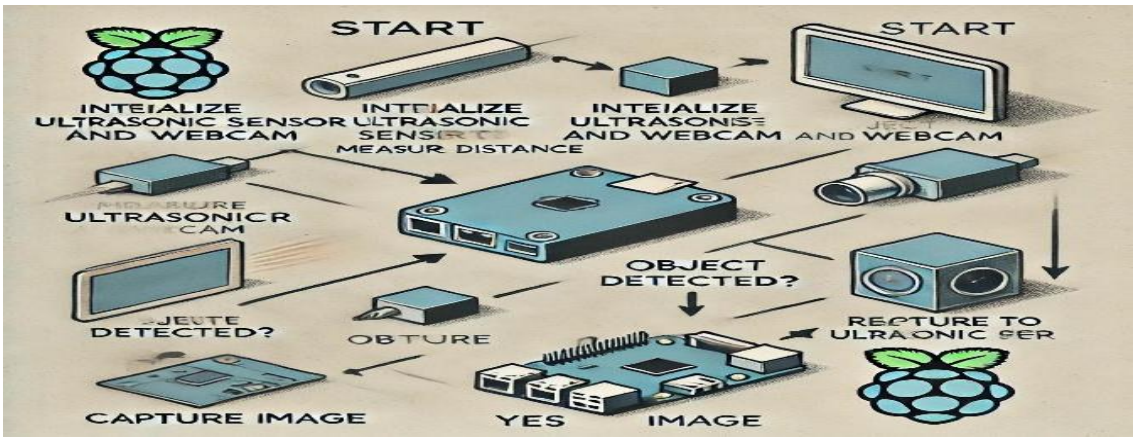

FIG:10

- The model successfully detected drones in various test images. Examples are shown below:

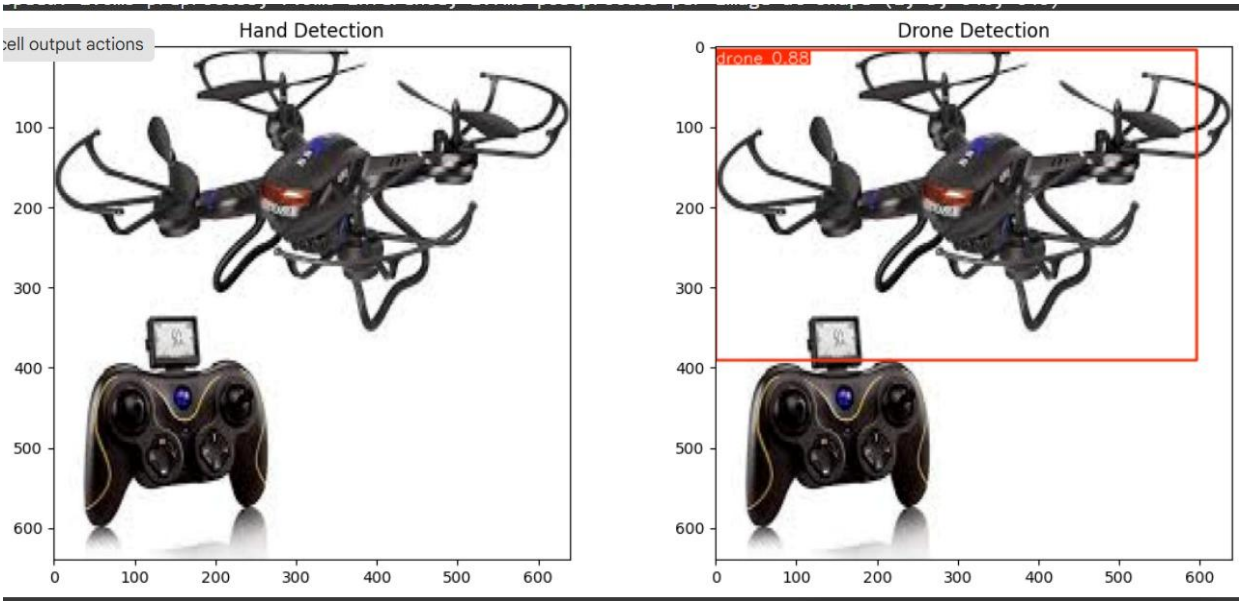

FIG:11

- Fig.11 is the individual result of one the drone image we have trained with the label named “drone” .
- Fig.12 is the individual result of the hand image we have trained with the bounding box.

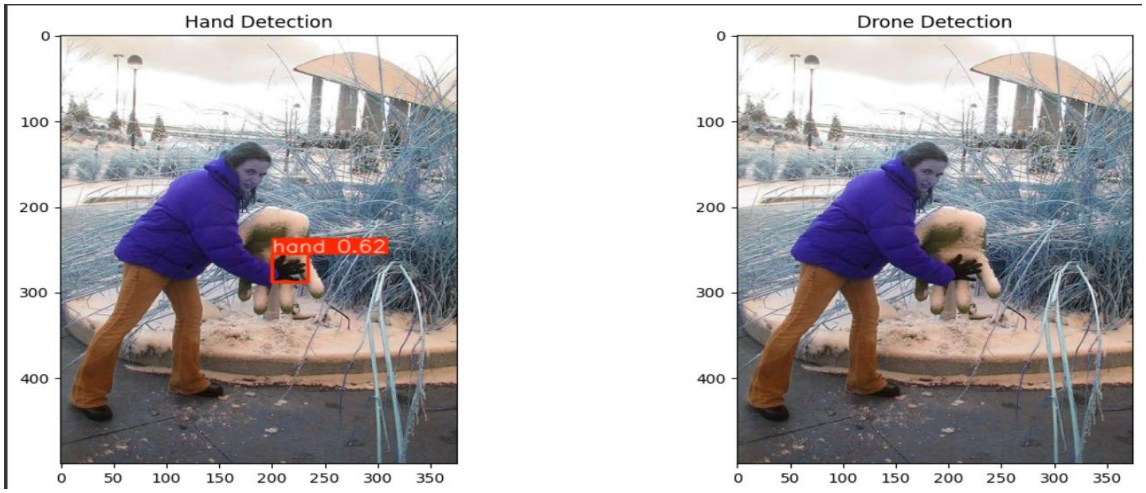

FIG:12

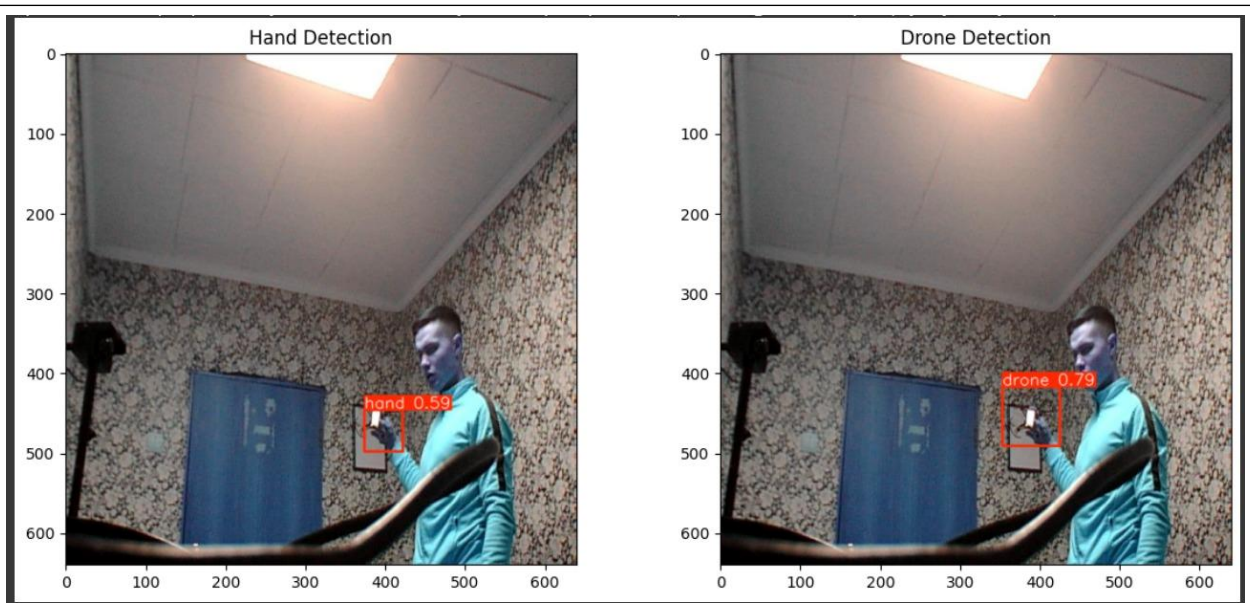

FIG:13

- Fig.13 is the result image gained consisting of both the “drone” and the “hand” are identified.
- The images are also labelled as “drone” and the “hand” and have bounding boxes around the results as we have trained.

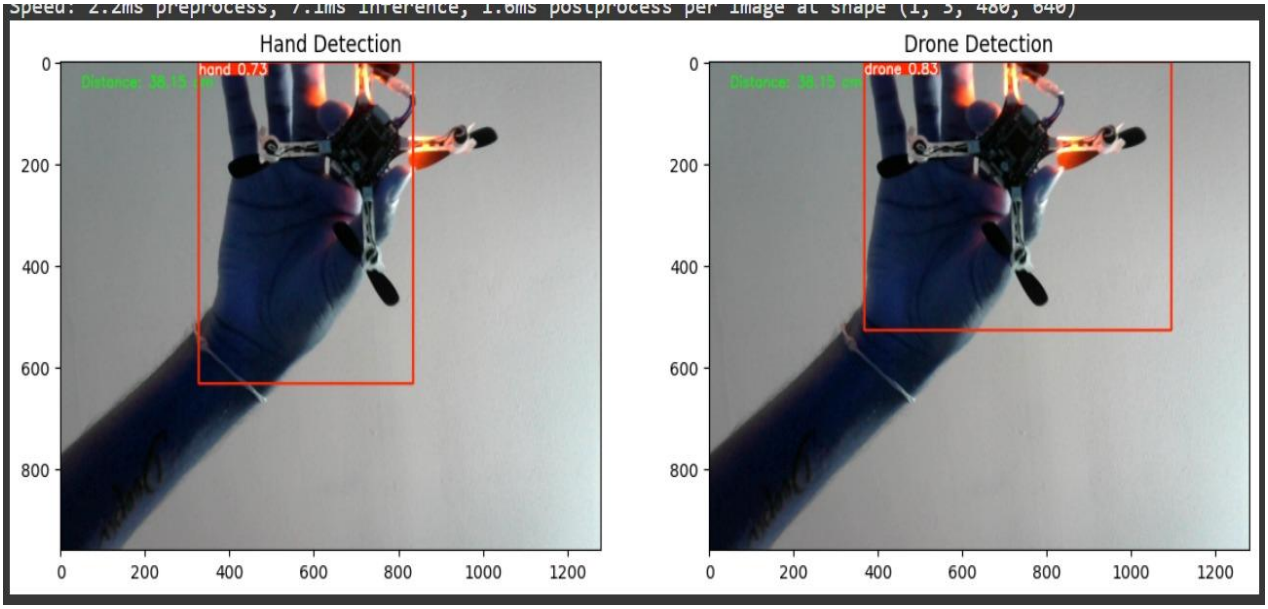

FIG:14

- Now, all these images are the images we have captured through the camera module, the real-time images consisting the “hand” and the “drone”.
- The above Fig.14 is the same image captured from the bottom angle consisting the hand and the drone.
- In the figure, we can observe both the hand and the drone are been presented separately in two different images with the title.
- The “drone” and the “hand” are resulted with the labels mentioned around the images along with the bounding boxes drawn.
- Also, we can find the distance at which the picture has been captured from the ultrasonic sensor and camera module.

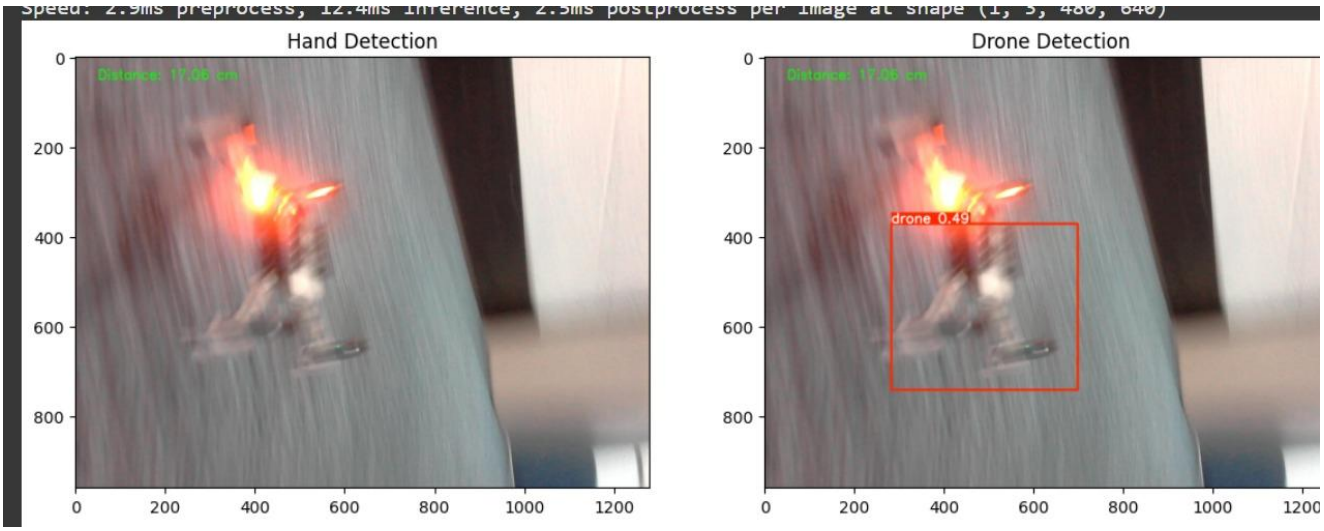

FIG:15

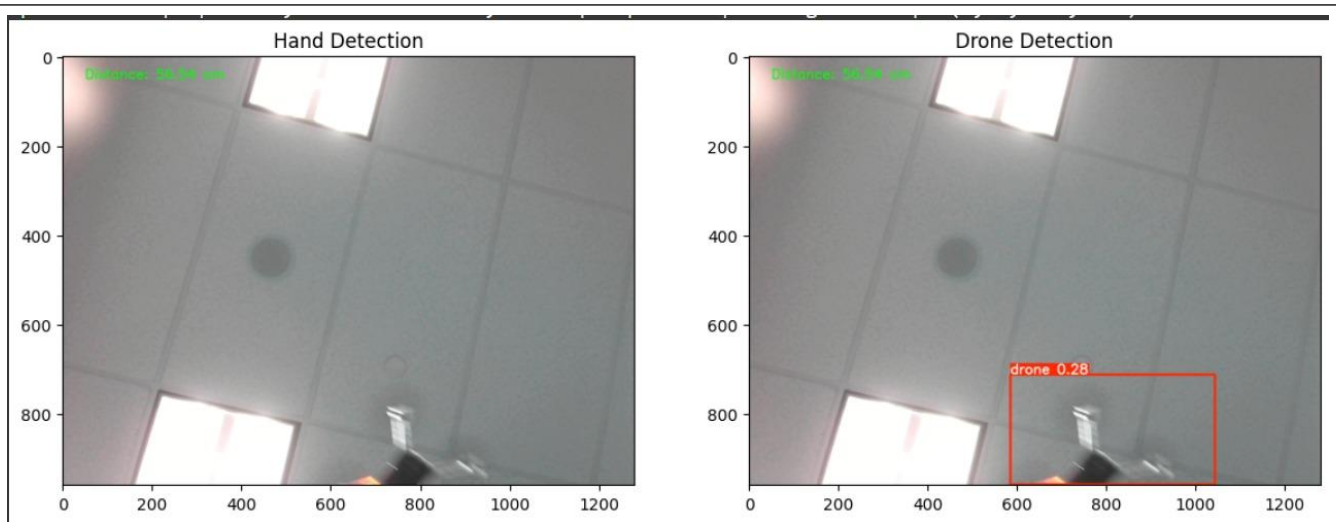

FIG:16

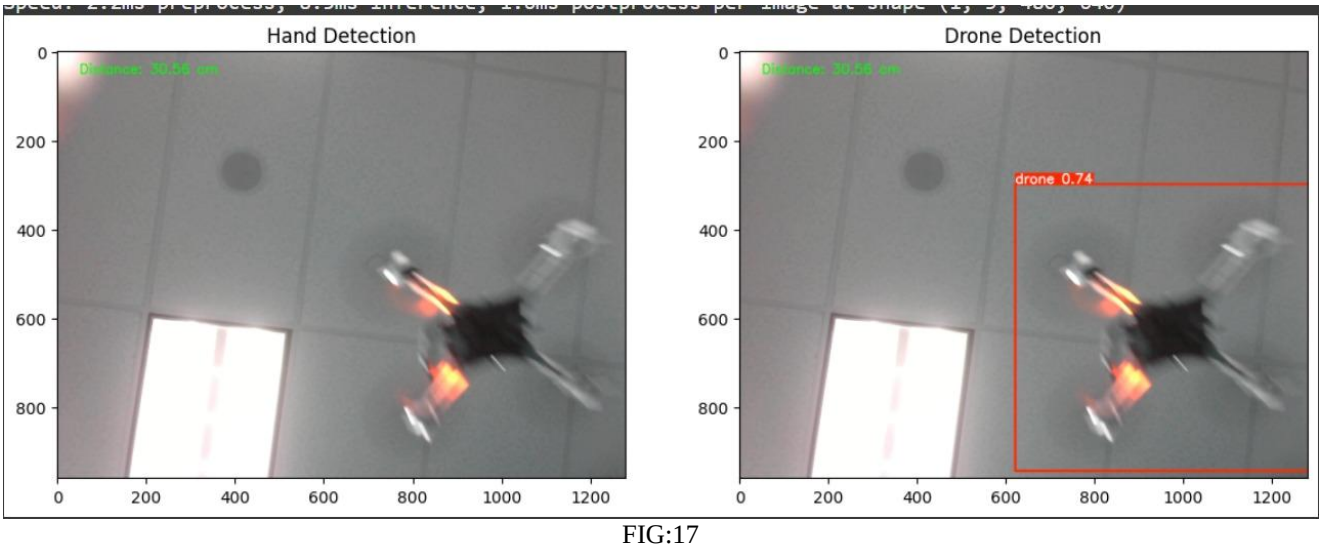

FIG:17

- The above given images Fig. 15,16 & 17 are the real-time test images captured through our camera module.
- In the figure, we can observe both the hand and the drone are been presented separately in two different images with their titles.
- The “drone” and the “hand” are resulted with the labels mentioned around the images along with the bounding boxes drawn around the images.
- Also, we can find the distance at which the pictures have been captured from the ultrasonic sensor and camera module.
- These images are been captured from the different angles to gain the accurate results we aimed for.
- The fig.15 is the image captured from the side angle.
- The fig.16 is the image captured from the bottom angle.
- And the fig.17 is the image captured from the top angle.
- All the images from all the angles have resulted in the desired output.
- Thus, our model has performed well with the good accuracy.

VIII. LIMITATIONS

Despite getting the results of the drone detection system, there are several limitations raised during the model development and testing stages:

- Using a Raspberry Pi for AI model is slow compared to using a powerful GPU. Raspberry Pi’s slowness makes it difficult to process things in real-time.
- It is challenging to capture a clear image at a distance from the camera module due to image noise concerns.
- Due to the various lighting conditions, the color differences made it difficult for YOLOv8 to detect the image.
- Detecting a hand in images captured from a long distance is difficult, where the drone is detectable.
- We faced challenges like various image conditions, lower detection accuracy and also during applying the techniques, data processing and fine-tuning.

IX. CONCLUSION

This project successfully developed a real-time drone detection system using the YOLOv8 object detection model. The custom dataset was created, labeled and used to train the model in Google colab. The model showed the desired accurate results in identifying the drone image with labels and the bounding boxes drawn on the test images.

During the process, there were some challenges we faced. The final system is capable of detecting drones in real-time. We can consider it valuable for applications in security, research and monitoring to some extinct. Further, the system can be improved by training it on more larger data and live video detection by testing it over various environment conditions.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude and thank all our peers and mates for their collaboration, encouragement, and constructive feedback, which have significantly contributed to refining our project.

Our heartfelt thanks to our professor for constant encouragement, suggestions and valuable feedback, and also providing the resources that helped us in shaping this project.

Lastly, we acknowledge the contributions of researches and developers whose work has served as a foundation for our project. Their research in the field of drone detection has provided us with valuable insights and inspiration. This project has been very great learning experience for us.

REFERENCES

The below the references that was helpful for us during the project:

[1] . Delleji, H. Fekih and Z. Chtourou, "Deep Learning-based approach for detection and classification of Micro/Mini drones," 2020 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET), Hammamet, Tunisia, 2020, pp. 332-337, doi: 10.1109/IC_ASET49463.2020.9318281.
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9318281&isnumber=9318217

[2] Ritika Giridhar. (2022). Hand gestures dataset [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/2206851

[3] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Software]. GitHub.
https://github.com/ultralytics/ultralytics

[4] General Hands. (2023, September). YoloV8-W/-Hands Dataset. Roboflow Universe. Available at: https://universe.roboflow.com/general-hands/yolov8-w-hands.

[5] project-iorgd. (2023, November). Hand Dataset. Roboflow Universe. Available at: https://universe.roboflow.com/project-iorgd/hand-hdj9u.

[6] Faizan. (2023, October). Hand Detection Dataset. Roboflow Universe. Available at: https://universe.roboflow.com/faizan-eqxqw/hand-detection-lrbt7.

[7] ITU. (2023, February). Drone Detection Dataset. Roboflow Universe. Available at:https://universe.roboflow.com/itu-t15gy/drone-detection-gayvs

[8] project-986i8. (2023, August). Drone Dataset. Roboflow Universe. Available at: https://universe.roboflow.com/project-986i8/drone-uskpc.

[9] smart robots. (2023, April). drone_detection_2 Dataset. Roboflow Universe. Available at: https://universe.roboflow.com/smart-robots/drone_detection_2.

[10] TRACKER. (2024, November). DRONES_NEW Dataset. Roboflow Universe. Available at:https://universe.roboflow.com/tracker-qjlj1/drones_new.

[11] Y. A. Badamasi, "The working principle of an Arduino," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2014, pp. 1-4, doi: 10.1109/ICECCO.2014.6997578
,URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6997578&isnumber=6997537

[12] C. Zet, C. Foşalău, A. Hariton and G. C. Sârbu, "Improved Distance Measurement Using Ultrasonic Sensors," 2023 International Conference on Electromechanical and Energy Systems (SIELMEN), Craiova, Romania, 2023, pp. 1-5, doi: 10.1109/SIELMEN59038.2023.10290844
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10290844&isnumber=10290142

[13] A. U. Kulkarni, A. M. Potdar, S. Hegde and V. P. Baligar, "RADAR based Object Detector using Ultrasonic Sensor," 2019 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019, pp. 204-209, doi: 10.1109/ICAIT47043.2019.8987259

URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8987259&isnumber=8987232

[14] D. Abuan, M. Lim, C. R. V. Lontoc, J. Sangrador, J. See and E. P. Dadios, "Arduino Based Obstacle Avoiding Robot with Ultrasonic Sensors and Motor Functions," 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 2023, pp. 1-7, doi: 10.1109/ICTBIG59752.2023.10456159URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10456159&isnumber=10455752

[15] A. Tedeschi, S. Calcaterra and F. Benedetto, "Ultrasonic RAdar System (URAS): Arduino and Virtual Reality for a Light-Free Mapping of Indoor Environments," in IEEE Sensors Journal, vol. 17, no. 14, pp. 4595-4604, 15 July15, 2017, doi: 10.1109/JSEN.2017.2708840

URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7934325&isnumber=7956209

[16] Getting started with drone:

https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/