EE 701 - Internet of Things

Project Report

Object Recognition on Drone

By:

Rohith Srinivas Raghavan
rohiths2@buffalo.edu

Naga Kiran Reddy Karnati
nkarnati@buffalo.edu

Shanmugam Vignesh
vshanmug@buffalo.edu

Rohith Kalige
rohithka@buffalo.edu

Contents

List of Figures

1 Project Description

Drones have been around for a long time, yet it is interesting to see the uses cases that are emerging, especially on the Internet of Things realm. Drones are fun toys, and many people fly them as a hobby. But over time more commercial and professional drones are being deployed for a range of business applications and critical missions — everything from sports photography to science, research, military missions, medical purposes and search and rescue. So, it is better to call them IoT drones.

To make these achievable, these IoT Drones must be able to carry some heavyweights. A USRP N200 or N210 weighs approximately 1.2 kg and with a dimension of 22 cm x 16 cm x 5 cm which will result in a primary requirement of a large frame. A frame that can fit a USRP and be able to hold it tight with no movement in the air. The IoT Drones need a large battery that can be used to travel long distances. The IoT Drones also require some powerful motors and a power distribution board to be able to lift all these weights. There are some basic requirements for IoT Drones such as Pixhawk 4, microcontroller, GPS module, and speed controllers to control the speed for motors.

We want to use the lab quadrotor UAV, which is equipped with an on-board computer, a camera, an altitude sensor (i.e., Lidar sensor). We want to receive the lidar sensor data and video data captured from drone camera and process that data on ground station (i.e., base computer).

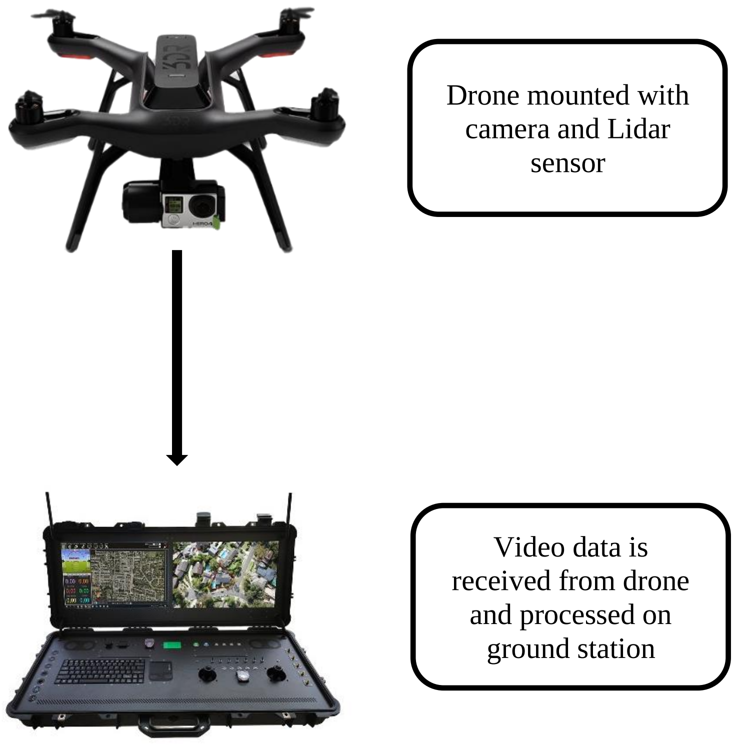

Drone mounted with camera and Lidar sensor
Video data is received from drone and processed on ground station

Figure 1: System Schematic [1]

2 Process Flowchart

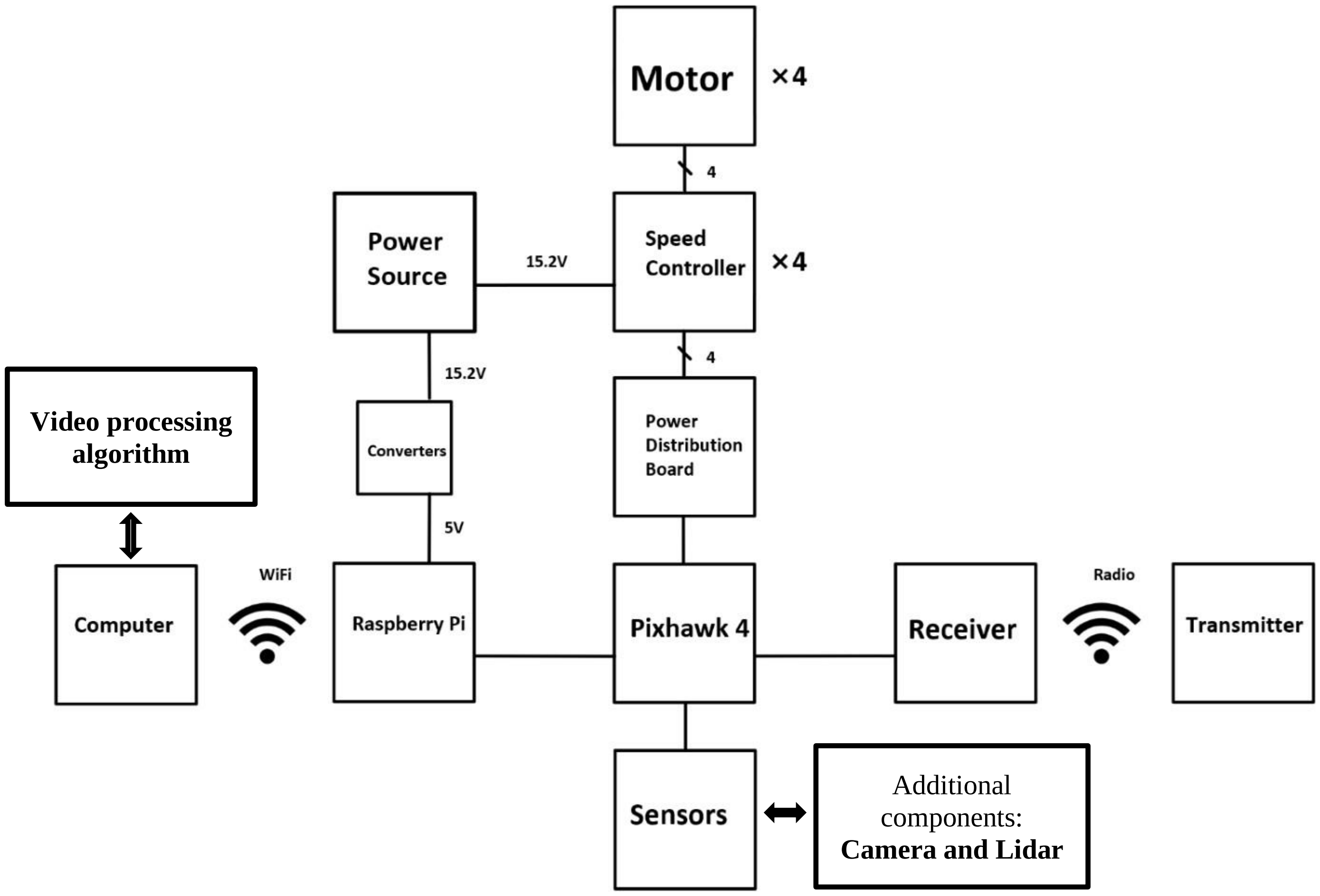


Figure 2: System Flowchart

3 Project Overview

The processes involved in the IoT Drone are:

- Data Acquisition
  Camera module and Lidar sensor mounted on drone is connected to Pixhawk 4. The Pixhawk 4 also connects to the GPS module to receive the GPS location. There are some built-in sensors in Pixhawk 4 such as accelerometer, gyroscope, barometer, and magnetometer which is the compass so it will receive data from all these sensors.

- Data transmission
  The Pixhawk 4 sends all data to Raspberry PI 4 and transfer to a ground station.

- Data Processing
  The video data and lidar data is processed and analyzed by Raspberry PI 4 using an algorithm to uncover useful insights from the received data

- Data Visualization
  The video data from the arducam is live streamed on a monitor and we performed object recognition on that data. For that we have used a pretrained ssd model which can detect up to 80 objects and deployed it on TensorFlow lite platform running on raspberry pi 4b. Our model can detect and draw a bounding box around the object with label and confidence score on top of the bounding box.

4 Hardware Setup

Raspberry pi setup:

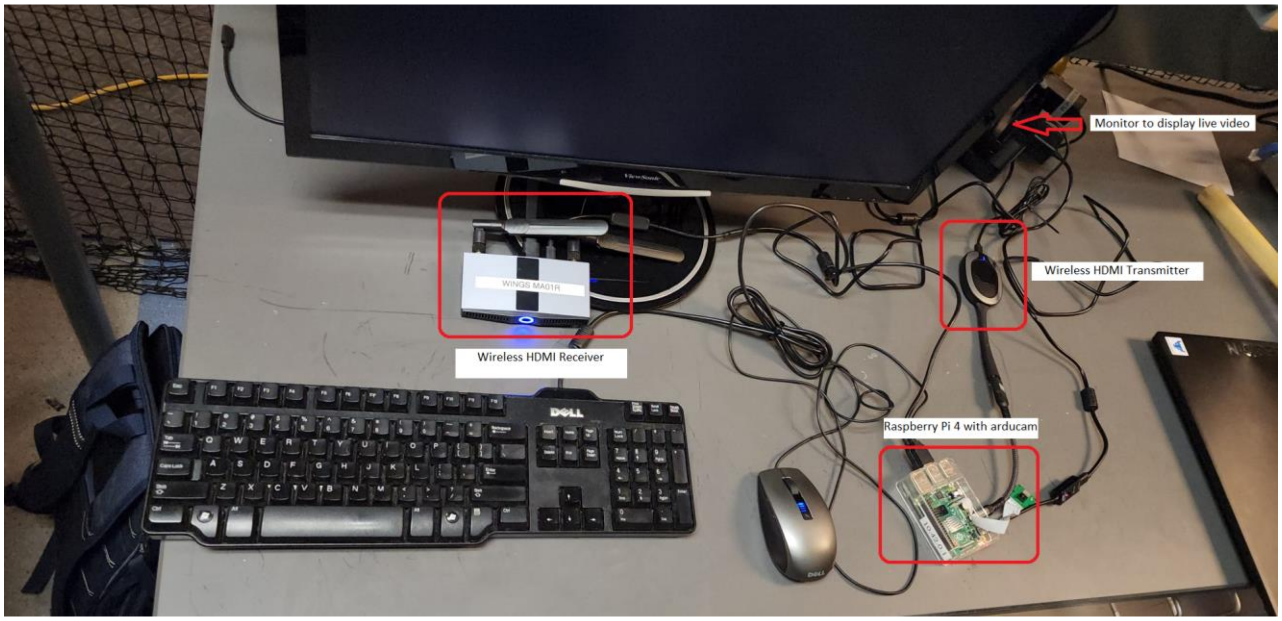

Monitor to display live video
Wireless HDMI Receiver
Wireless HDMI Transmitter
Raspberry Pi 4 with arducam

Figure 3: Hardware setup

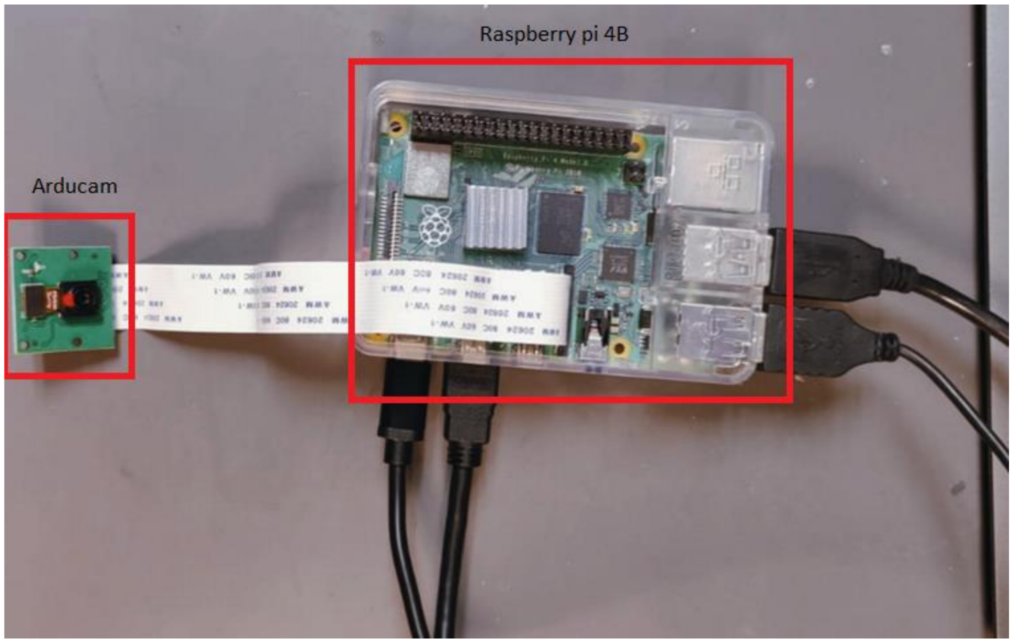
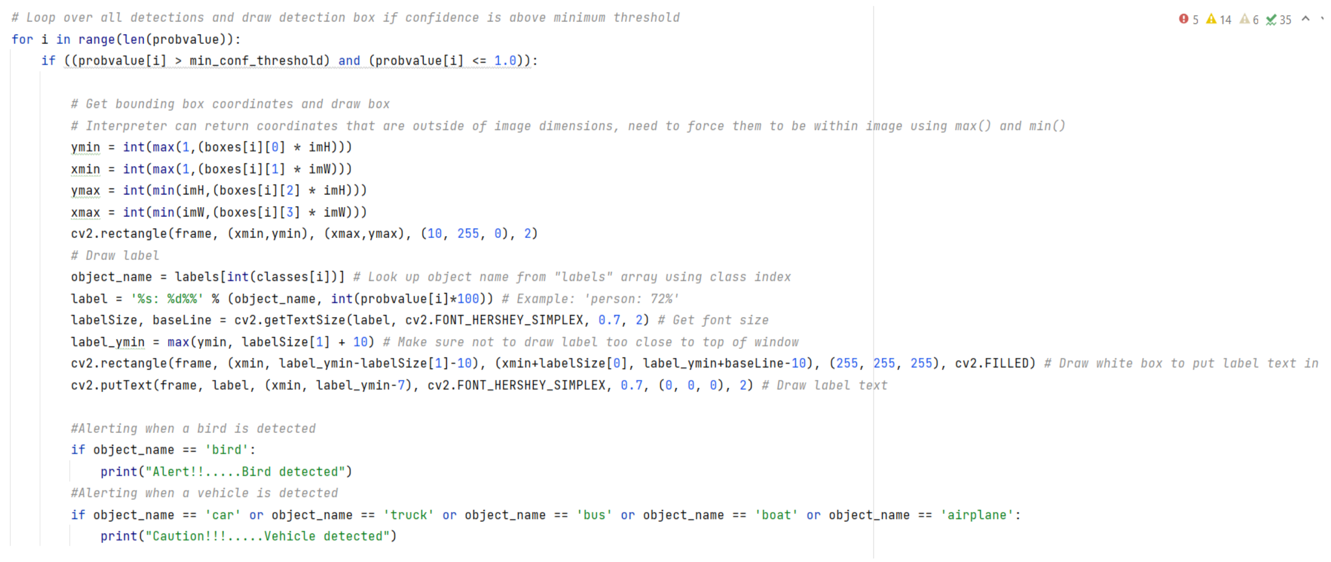


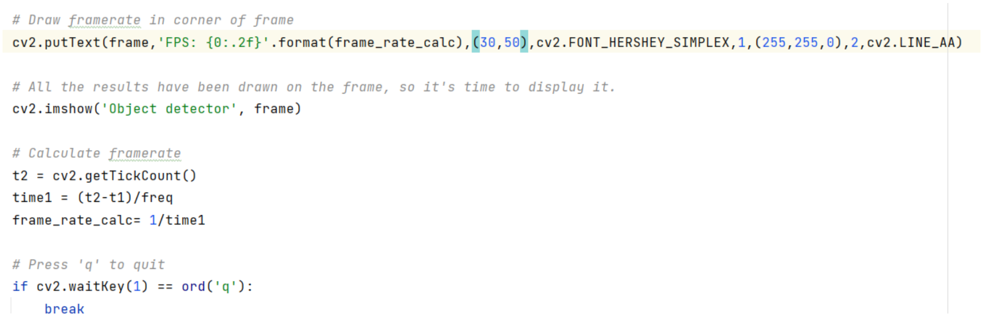
Figure 4: Interfacing camera module with Raspberry Pi 4b

5 Code snippets:

Main code to alert the user and draw bounding boxes, display label and confidence score is shown below:



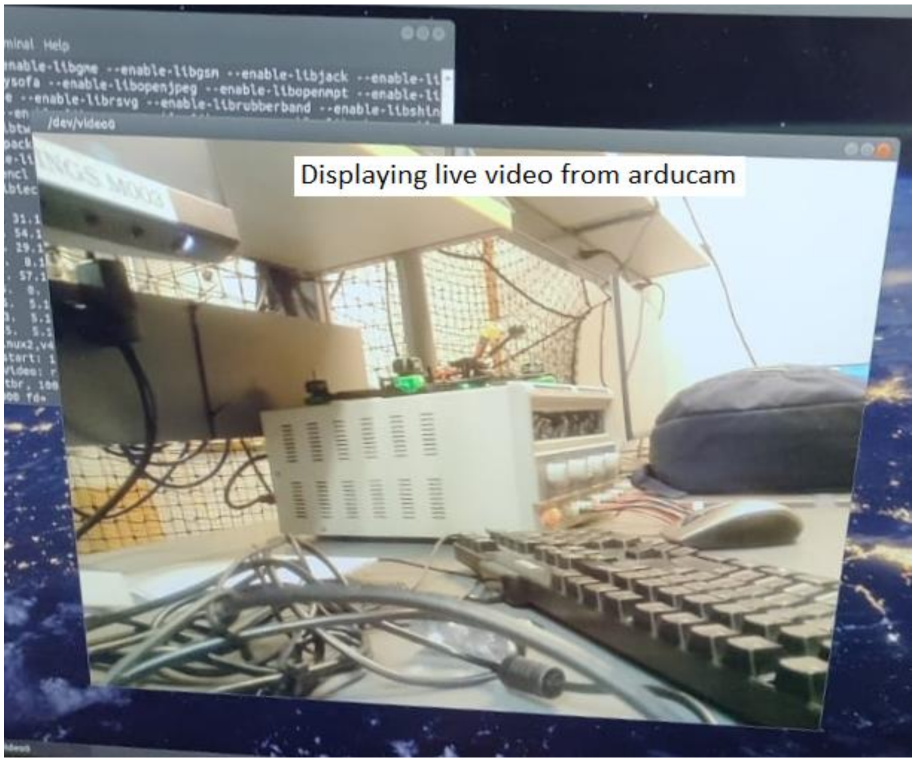
```
# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(probvalue)):
    if ((probvalue[i] > min_conf_threshold) and (probvalue[i] <= 1.0)):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))
        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
        label = '%s: %d%%' % (object_name, int(probvalue[i]*100)) # Example: 'person: 72%'
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
        cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

        #Alerting when a bird is detected
        if object_name == 'bird':
            print("Alert!!.....Bird detected")
        #Alerting when a vehicle is detected
        if object_name == 'car' or object_name == 'truck' or object_name == 'bus' or object_name == 'boat' or object_name == 'airplane':
            print("Caution!!!.....Vehicle detected")
```



```
# Draw framerate in corner of frame
cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)

# Calculate framerate
t2 = cv2.getTickCount()
time1 = (t2-t1)/freq
frame_rate_calc= 1/time1

# Press 'q' to quit
if cv2.waitKey(1) == ord('q'):
    break
```

6 Results

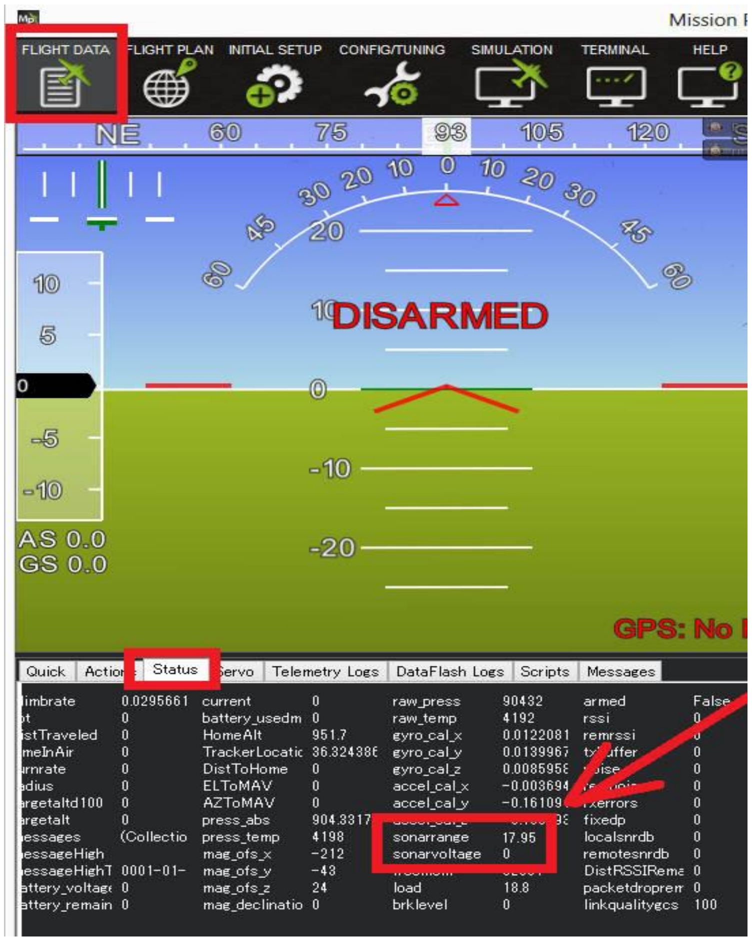
Live video from camera:



```
enable-libgme --enable-libgsn --enable-libjack --enable-li
ysofa --enable-libopenjpeg --enable-libopenmpt --enable-li
e --enable-librsvg --enable-librubberband --enable-libshin
```

Figure 5: Displaying live camera feed from Arducam on ubuntu mate

Lidar data on Mission Planner:



```
limbrate       0.0295661  current         0          raw_press     90432      armed          False
ot             0          battery_usedm   0          raw_temp      4192       rssi           0
istTraveled    0          HomeAlt         951.7      gyro_cal_x    0.0122081  remrssi        0
meInAir        0          TrackerLocatic  36.32438€  gyro_cal_y    0.0139967  txbuffer       0
ırnrate        0          DistToHome      0          gyro_cal_z    0.0085958  [illegible]    0
adius          0          ELToMAV         0          accel_cal_x   -0.003694  [illegible]    0
argetaltd100   0          AZToMAV         0          accel_cal_y   -0.16109   rxerrors       0
argetalt       0          press_abs       904.3317   [illegible]   [illegible] fixedp        0
essages        (Collectio press_temp      4198       sonarrange    17.95      localsnrdb     0
essageHigh                mag_ofs_x       -212       sonarvoltage  0          remotesnrdb    0
essageHighT    0001-01-   mag_ofs_y       -43        [illegible]   [illegible] DistRSSIRem   0
attery_voltage 0          mag_ofs_z       24         load          18.8       packetdroprem  0
attery_remain  0          mag_declinatio  0          brklevel      0          linkqualitygcs 100
```

Figure 6: Leddar One data on Mission planner

Object recognition:

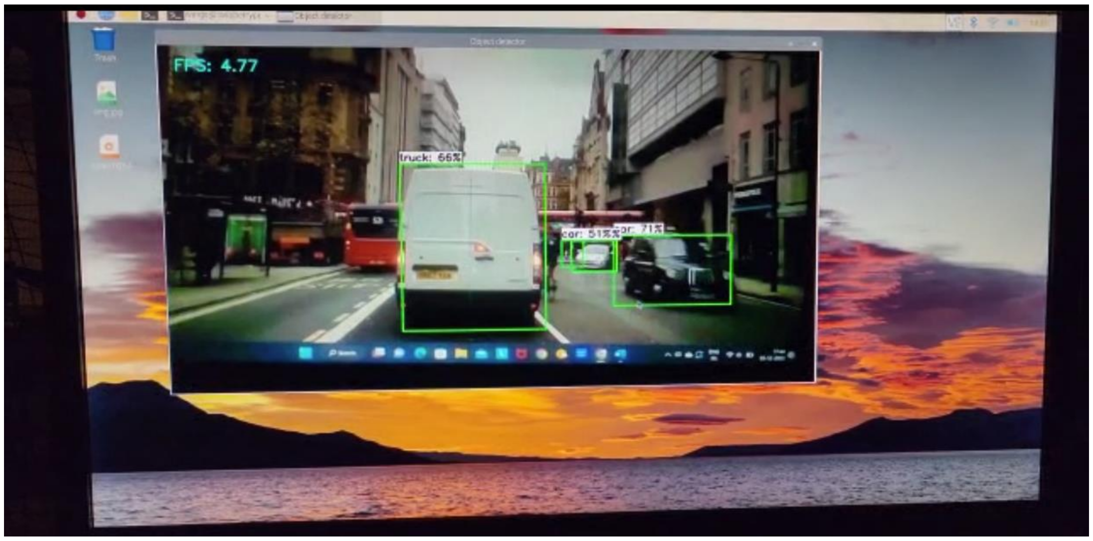
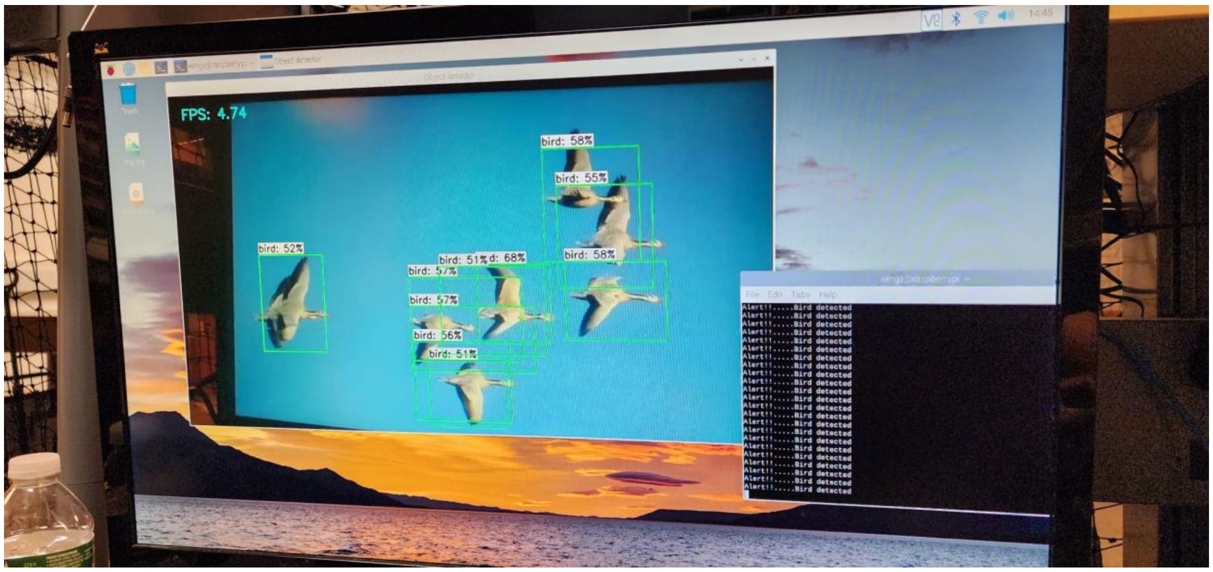


Figure 7: Model detecting vehicles in a scene

Bird detection:



```
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
Alert!!.....Bird detected
```

Figure 8: Alerting when a bird is detected

7 Conclusion

We were able to locate the objects or nearby drones from video data, captured from camera module, and lidar data captured from lidar sensor mounted on the drone.

References

- Song Han, William Shen, Zuozhen Liu. " Deep Drone: Object Detection and Tracking for Smart Drones on Embedded System" Stanford University, 2016 https://web.stanford.edu/class/cs231a/prev_projects_2016/deep-drone-object__2_.pdf