

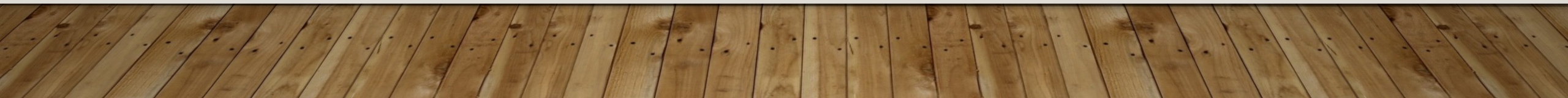
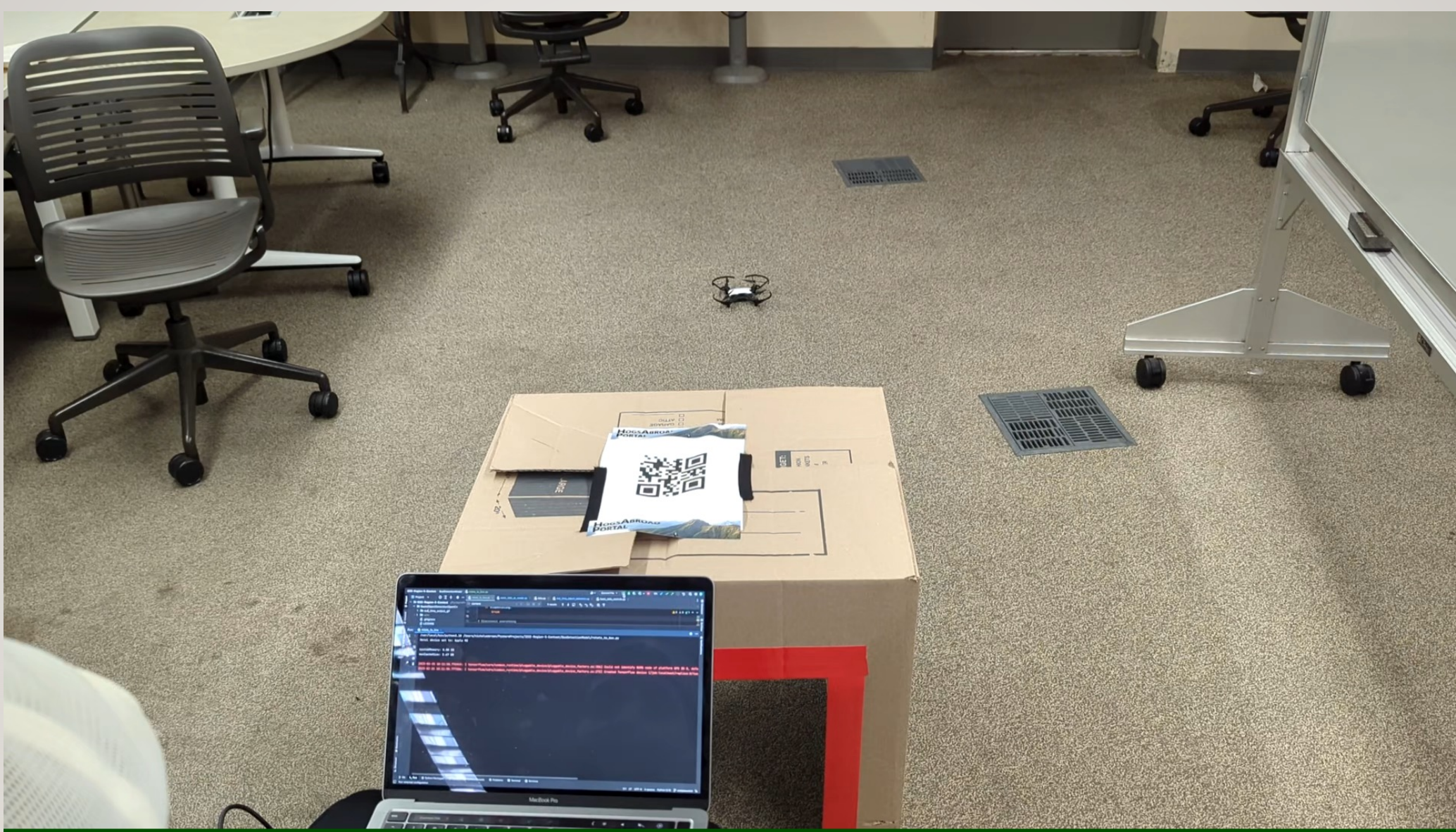
TEAM 8 EE IEEE ROBOTICS COMPETITION

NICHOLAS BROWN, CALLUM BRUTON, JASE CORNETT, AUSTIN FLYNN, STEPHANIE STOCK



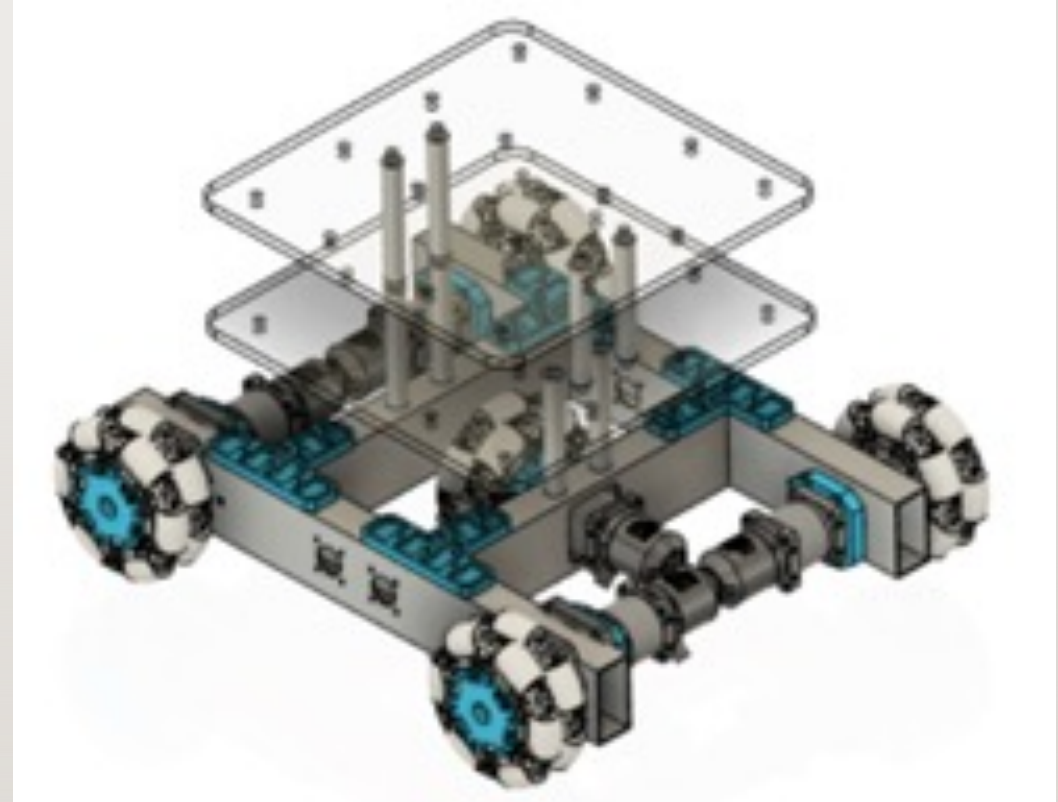
DJI RYZE TELLO DRONE

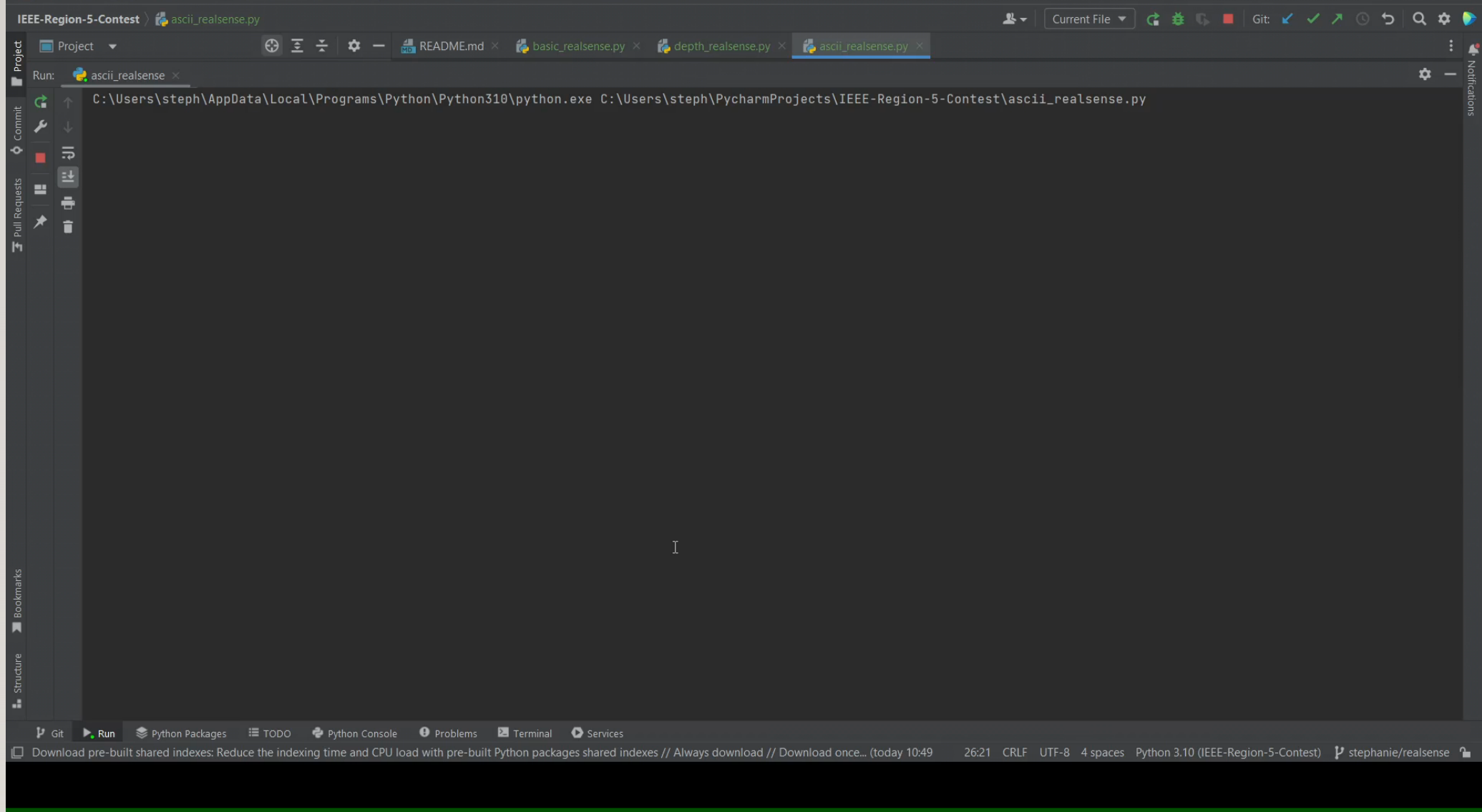
- Has a pre-made Python SDK with function that allow us to control the drone
- Uses a finite-state machine to operate autonomously
- Reads QR codes using OpenCV with a pre-made QR reader
- Uses object detection model to locate box or ground robot and fly to it
- Sends crucial data back to Nvidia Jetson over UDP



GROUND ROBOT

- Aluminum and Acrylic Frame
- Five 4" Omni-wheels
- Sensors
 - Ultrasonic
 - Intel RealSense D415i
 - Raspberry Pi Camera V2





The image shows a top-down view of an NVIDIA Jetson Nano development board. The board is populated with various components, including a large black heat sink covering the central processor, several peripheral chips, and various connectors along the edges. The text 'NVIDIA JETSON NANO' is overlaid on the left side of the board in a white, sans-serif font. A vertical white line is positioned to the right of the title, separating it from the list of features.

NVIDIA JETSON NANO

- Main processing unit for drone and ground robot.
- Takes input from the drone using UDP, Raspberry Pi Camera module V2, and ultrasonic sensor
- Outputs commands to the drone and ground robot using Python
- I/O used for this project include wireless networking, USB, CSI connector, and GPIO pins

sketch_mar9b | Arduino IDE 2.0.3

File Edit Sketch Tools Help

Arduino MKR1000

Serial Monitor

Serial Monitor

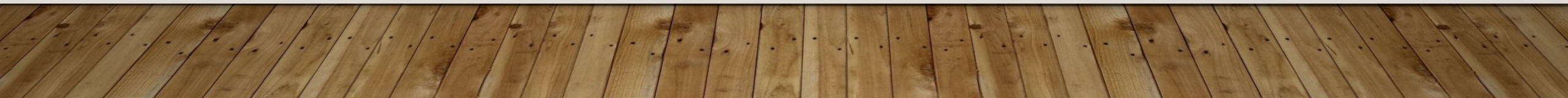
```
sketch_mar9b.ino
1 // defines pins numbers
2 const int trigPin = 7;
3 const int echoPin = 6;
4 // defines variables
5 long duration;
6 int distance;
7 void setup() {
8   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
9   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
10  Serial.begin(9600); // Starts the serial communication
11 }
12 void loop() {
13   // Clears the trigPin
14   digitalWrite(trigPin, LOW);
15   delayMicroseconds(2);
16   // Sets the trigPin on HIGH state for 10 micro seconds
17   digitalWrite(trigPin, HIGH);
18   delayMicroseconds(10);
19   digitalWrite(trigPin, LOW);
20   // Reads the echoPin, returns the sound wave travel time in microseconds
21   duration = pulseIn(echoPin, HIGH);
22   // Calculating the distance
23   distance = duration * 0.034 / 2;
24   // Prints the distance on the Serial Monitor
25   Serial.print("Distance: ");
26   Serial.println(distance);
27   delay(250);
28 }
```

Output

```
[===== ] 91% (192/209 pages)
[=====] 100% (209/209 pages)
done in 0.104 seconds

Verify 13372 bytes of flash with checksum.
Verify successful
done in 0.017 seconds
CPU reset.
```

Ln 27, Col 12 UTF-8 Arduino MKR1000 on COM3 4:27 PM 3/9/2023



```
print battery
```

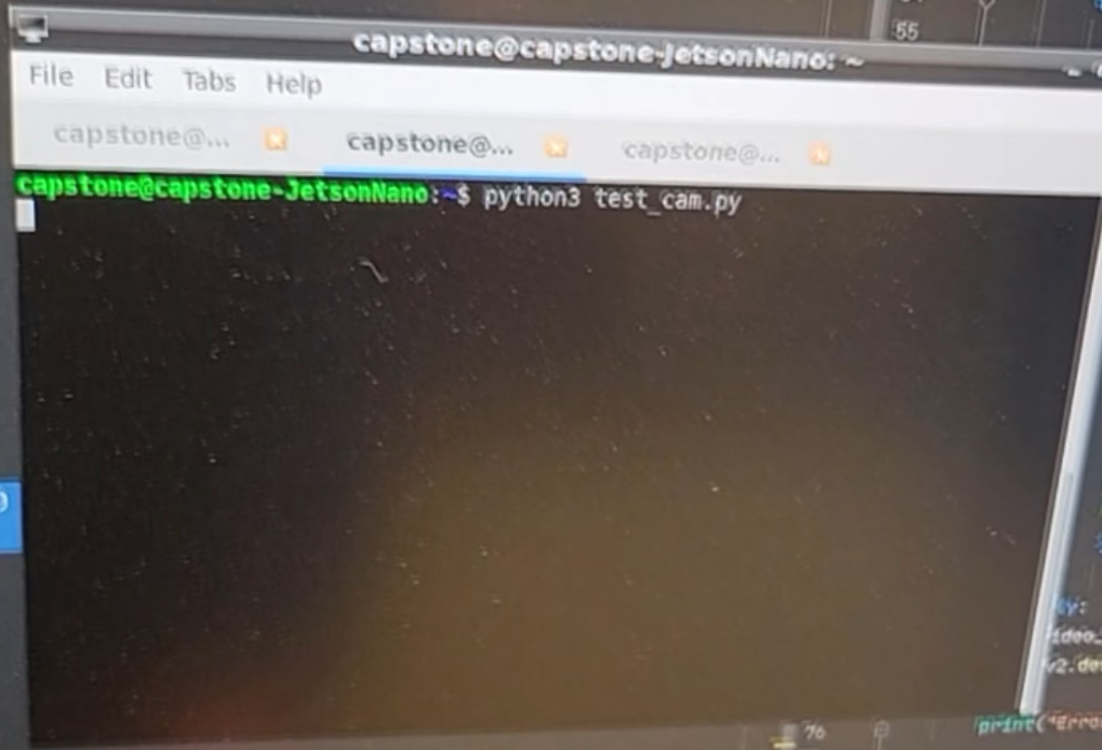
```
) .frame
```

```
detector.detectAndDecode(img)
```

```
240))
```

```
ord('q')):
```

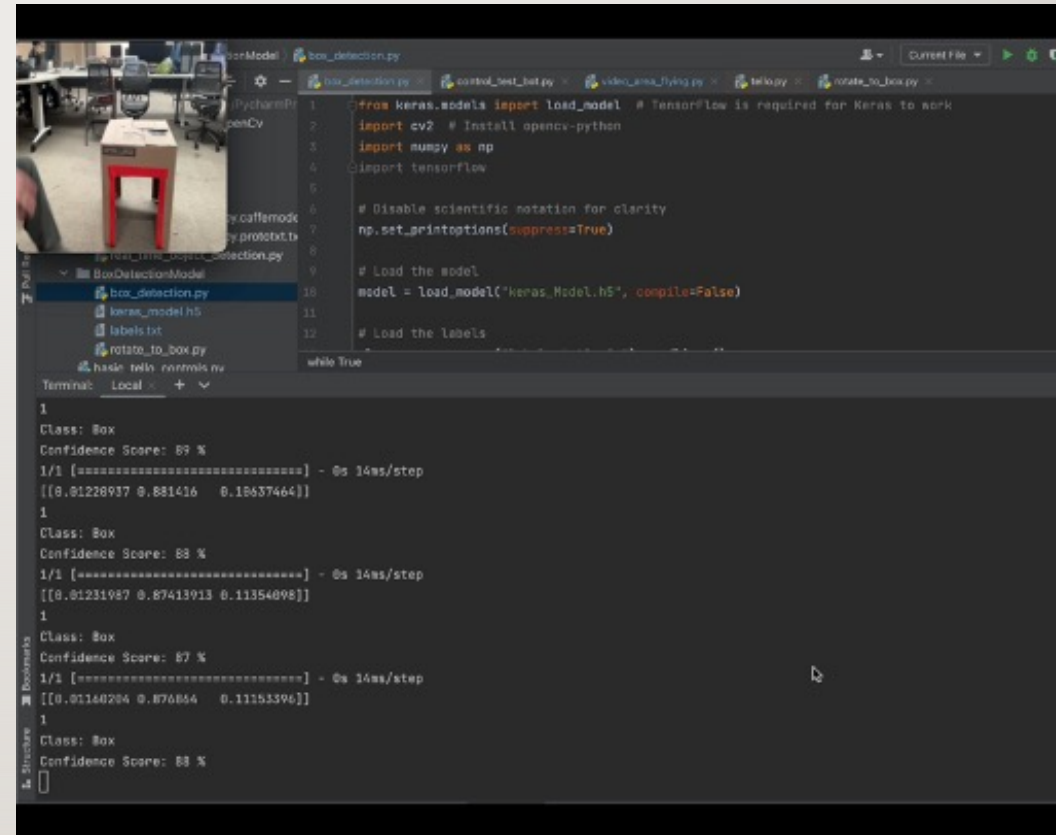
```
done
```



```
50 # To flip the image, modify the flip_method/p
51 print(gstreamer_pipeline(flip_method=0))
52 video_capture = cv2.VideoCapture(gstreamer_pipe
53 if video_capture.isOpened():
54     try:
55         window_handle = cv2.namedWindow(window_title)
56         cv2.resize(window_handle, (width, height))
57         while True:
58             ret_val, frame = video_capture.read()
59             # Check to see if the user closed the window
60             # Under GTK+ (Jetson Default), WND_PROP_FULLSCREEN
61             # GTK+ Substitute WND_PROP_AUTOSIZE to detect
62             if cv2.getWindowProperty(window_title, cv2.WND_PROP_FULLSCREEN) == cv2.WND_PROP_FULLSCREEN:
63                 data, bbox, straight_qrcode = detector.detect(frame)
64                 if len(data) > 0:
65                     print(data)
66                     cv2.imshow(window_title, frame)
67             else:
68                 break
69             keyCode = cv2.waitKey(10) & 0xFF
70             # Stop the program on the 'ESC' key or 'q'
71             if keyCode == 27 or keyCode == ord('q'):
72                 break
73     except:
74         print("Error: Unable to open camera")
75         video_capture.release()
76         cv2.destroyAllWindows()
77
78 # detector = cv2.QRCodeDetector()
79 show_camera()
80
```


OBJECT DETECTION MODEL

- Created using Teachable Machines
- Exported for Keras that can be used in Python with supporting packages
- Can differentiate between the boxes, drones, and everything else
- Is used with the drone's camera and on the Intel RealSense



```
1 from keras.models import load_model # TensorFlow is required for Keras to work
2 import cv2 # Install opencv-python
3 import numpy as np
4 import tensorflow
5
6 # Disable scientific notation for clarity
7 np.set_printoptions(suppress=True)
8
9 # Load the model
10 model = load_model("keras_model.h5", compile=False)
11
12 # Load the labels
13 labels = open("labels.txt", "r").readlines()
14
15 while True:
16     1
17     Class: Box
18     Confidence Score: 89 %
19     1/1 [=====] - @s 14ms/step
20     [[0.01228937 0.881416 0.10637464]]
21     1
22     Class: Box
23     Confidence Score: 88 %
24     1/1 [=====] - @s 14ms/step
25     [[0.01231987 0.87413913 0.11354098]]
26     1
27     Class: Box
28     Confidence Score: 87 %
29     1/1 [=====] - @s 14ms/step
30     [[0.01140204 0.876854 0.11153396]]
31     1
32     Class: Box
33     Confidence Score: 88 %
```

The image shows a PyCharm IDE window with the following components:

- Project View (Left):** Shows the project structure for "IEEE-Region-5-Contest". The "BoxDetectionModel" folder is expanded, showing files like "box_detection.py", "keras_model.h5", "labels.txt", and "rotate_to_box.py".
- Code Editor (Center):** Displays the code for "box_detection.py". The visible code includes:

```
1 from keras.models import load_model # TensorFlow is required for Keras to work
2 import cv2 # Install opencv-python
3 import numpy as np
4 import tensorflow
5
6 # Disable scientific notation for clarity
7 np.set_printoptions(suppress=True)
8
9 # Load the model
10 model = load_model("keras_Model.h5", compile=False)
11
12 # Load the labels
13 while True
```
- Terminal (Bottom):** Shows the execution of the script. The output includes:

```
nicholasbrown@nicholass-mbp BoxDetectionModel % python3.10 box_detection.py
Metal device set to: Apple M2

systemMemory: 8.00 GB
maxCacheSize: 2.67 GB

2023-03-09 16:21:18.070658: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:306] Could not identify NUMA node of platform GPU ID 0, defaulting
to 0. Your kernel may not have been built with NUMA support.
2023-03-09 16:21:18.071055: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:272] Created TensorFlow device (/job:localhost/replica:0/task:0/de
vice:GPU:0 with 0 MB memory) -> physical PluggableDevice (device: 0, name: METAL, pci bus id: <undefined>)
```