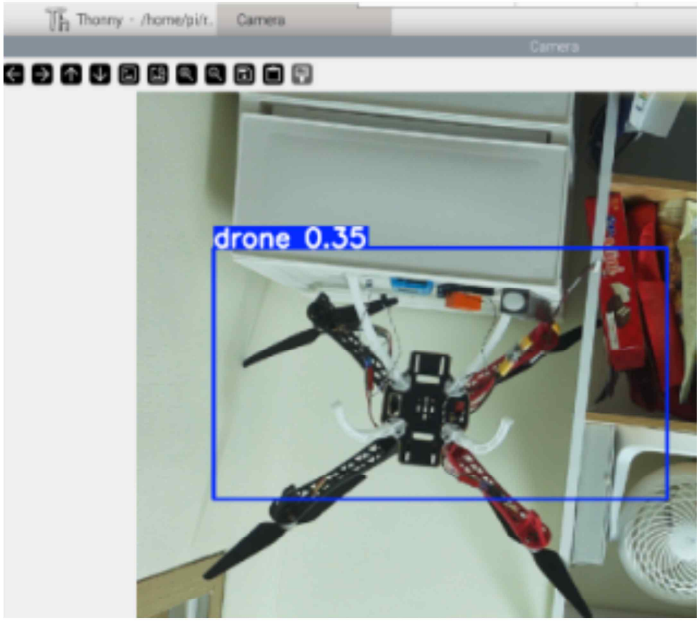
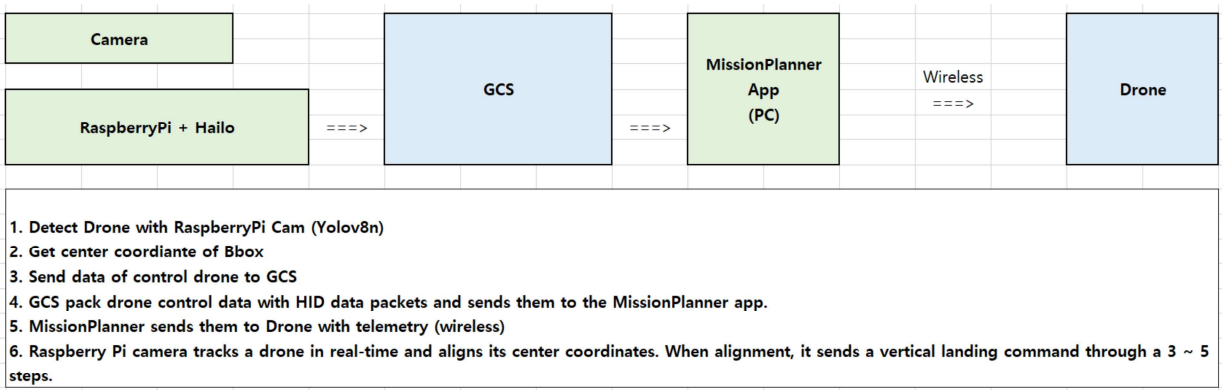


=> 10 -> 30 에 비해 30 -> 100 은 큰 변화량이 없어 보임.

< 4 차 학습 - epochs / 모델 학습 소요시간 : 시간 >

6. 정밀 착륙 유도 시스템 SW 개발

6.1 System Flow



System Flow 를 풀어쓰자면 다음과 같기에, 모델 학습을 병행하며 위 순서대로 개발하게 될 것 같습니다.

6.2 Get center coordinate of BBox

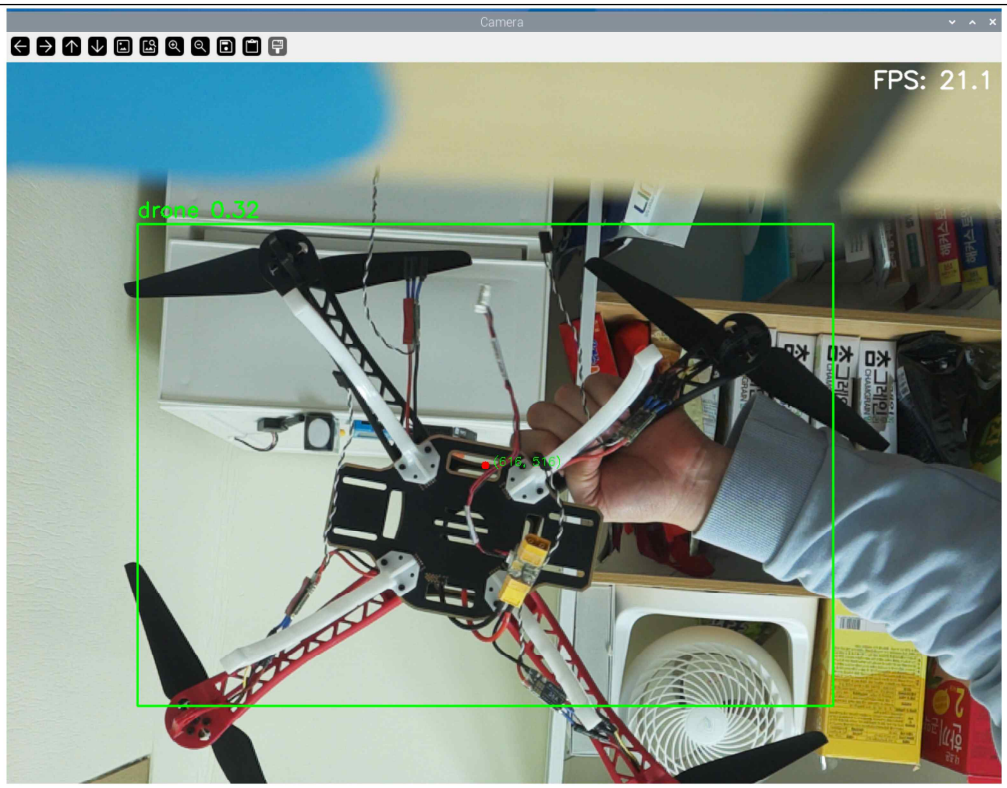
- 위 기능을 위해 실제 BBox 의 좌표값이 필요합니다. 다행히 추가적인 구현없이 객체 인식 결과물에서 산출되는 BBox 의 x,y 최소 최대 좌표값들이 있어 해당 class 함수를 통해 값을 반환받을 수 있었습니다.

```
frame = picam2.capture_array()

results = model(frame)
boxes = results[0].boxes.xyxy
for box in boxes:
    x_min, y_min, x_max, y_max = box[0].item(), box[1].item(), box[2].item(), box[3].item()

    cv2.rectangle(frame, (int(x_min), int(y_min)), (int(x_max), int(y_max)), (0, 255, 0), 2)

    bbox_center_x = (x_min + x_max) // 2
    bbox_center_y = (y_min + y_max) // 2
```



=> 테스트 코드 실행 결과

6.3 Send data of control drone to GCS

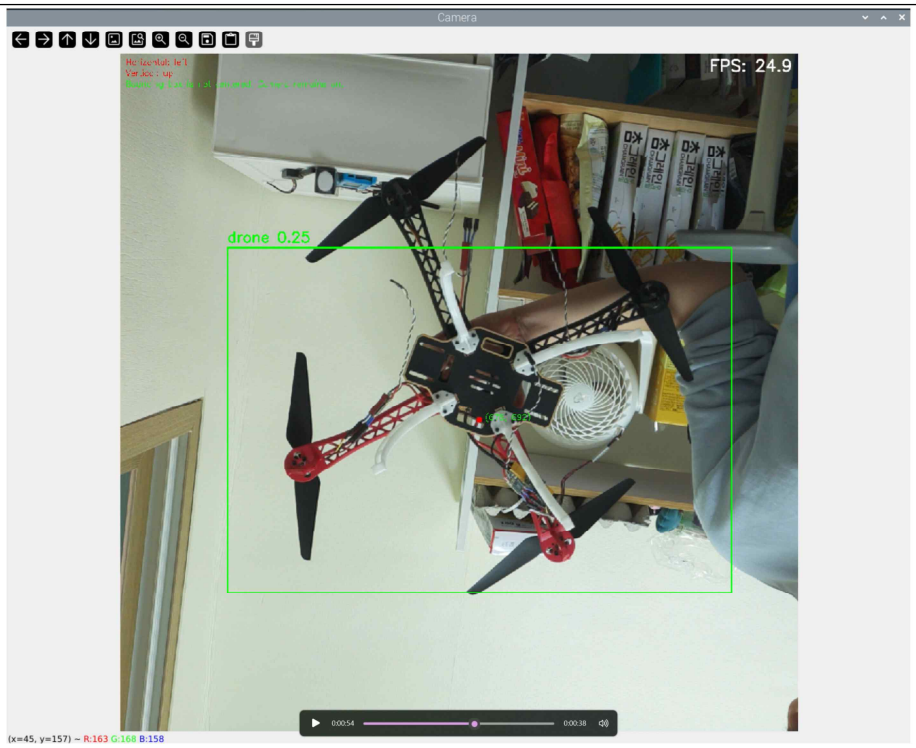
- 위 기능을 구현하기 위해, BBox 에서 산출된 중심 좌표가 카메라 화면의 중심부에 올 수 있도록 신호를 줄 수 있도록 구현할 예정이며. 신호는 특정 digital pin 에 output 을 주거나, Packet 형태로 Serial 라인을 거쳐 전달할 예정입니다.

다만 현재 GCS HW 를 완벽히 구성하지 못해 임시로 화면 좌측 상단에 어디로 움직여야 하는지에 대한 정보를 출력하도록 설정하였습니다.

```
if diff_x < margin and diff_y < margin:
else:
    if diff_x < margin:
        horizontal_direction = "center"
    elif diff_x > margin:
        if bbox_center_x < screen_center_x:
            horizontal_direction = "right"
        else:
            horizontal_direction = "left"

    if diff_y < margin:
        vertical_direction = "center"
    elif diff_y > margin:
        if bbox_center_y < screen_center_y:
            vertical_direction = "down"
        else:
            vertical_direction = "up"

    cv2.putText(frame, f"Horizontal: {horizontal_direction}", (10, 20), font, 0.5, (0, 0, 255), 1)
    cv2.putText(frame, f"Vertical: {vertical_direction}", (10, 40), font, 0.5, (0, 0, 255), 1)
```



=> 테스트 코드 실행 결과 (좌측 상단에 상하, 좌우 기준 어디로 가야할지 적혀있는 상태)

6.4 GCS pack drone control data with HID data packets and sends them to the MissionPlanner app

=> 개발 예정

6.5 MissionPlanner sends them to Drone with telemetry (wireless)

=> 개발 예정

6.6 Raspberry Pi camera tracks a drone in real-time and aligns its center coordinates. When alignment, it sends a vertical landing command through a 3 ~ 5 steps.

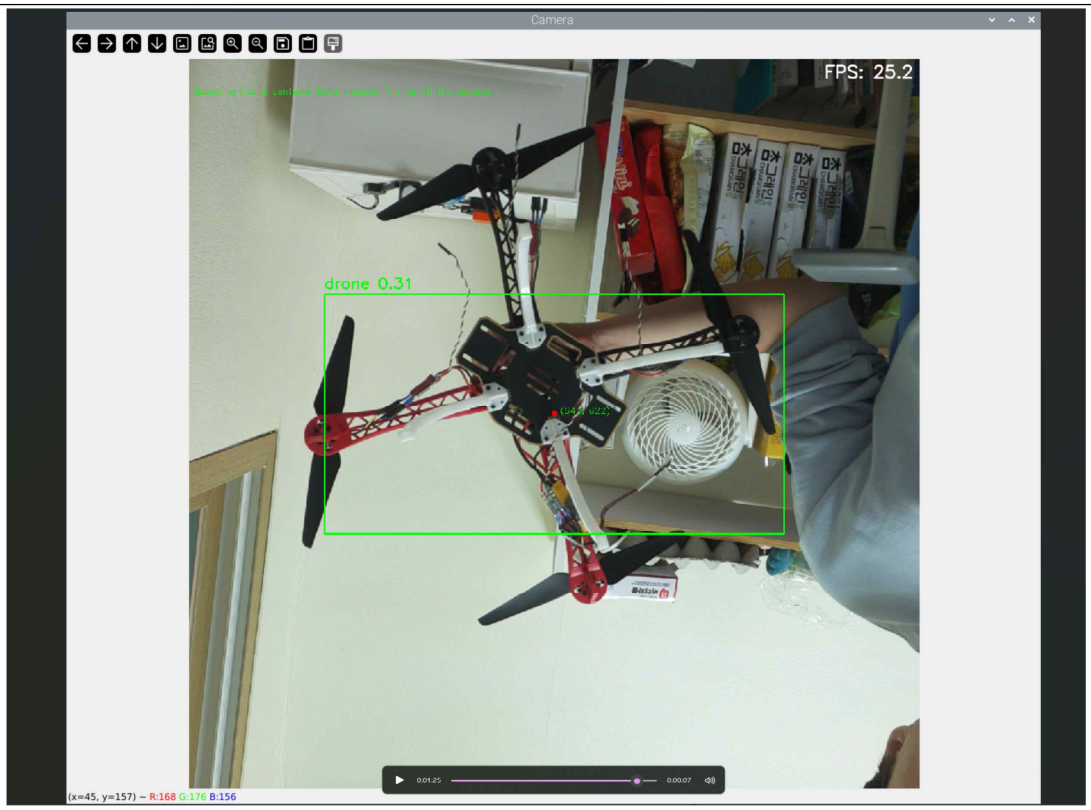
- 현재 수직 착륙 신호는 구성하지 않았으나 중심 좌표 도달 시, 프로그램이 종료될 수 있도록 구성하였습니다.

```
diff_x = abs(bbox_center_x - screen_center_x)
diff_y = abs(bbox_center_y - screen_center_y)

if diff_x < margin and diff_y < margin:
    cv2.putText(frame, "Bounding box is centered within margin. Turning off the camera...", (10, 60), font, 0.5, (0, 255, 0), 1)
    cv2.waitKey(2000)
    camera_on = False
else:
    if diff_x < margin:
        horizontal_direction = "center"
    elif diff_x > margin:
        if bbox_center_x < screen_center_x:
            horizontal_direction = "right"
        else:
            horizontal_direction = "left"

    if diff_y < margin:
        vertical_direction = "center"
    elif diff_y > margin:
        if bbox_center_y < screen_center_y:
            vertical_direction = "down"
        else:
            vertical_direction = "up"

    cv2.putText(frame, f"Horizontal: {horizontal_direction}", (10, 20), font, 0.5, (0, 0, 255), 1)
    cv2.putText(frame, f"Vertical: {vertical_direction}", (10, 40), font, 0.5, (0, 0, 255), 1)
    cv2.putText(frame, "Bounding box is not centered. Camera remains on.", (10, 60), font, 0.5, (0, 255, 0), 1)
```



=> 테스트 코드 실행 결과 (좌측 상단의 좌표 값은 더이상 무의미하므로 센터에 위치시켰다는 메시지만 출력)

7. a
8. a
9. a
10. a
11. a
12. a
13. a
14. a
15.