라즈베리파이 기반 AI 실시간 관제경보 시스템

AI 시스템 반도체 설계 2기
안재용 고종완 배상원 정지원

CONTENTS

01 | 개요

프로젝트 목표

개요

- 실시간 객체 인식과 온디바이스 경보 시스템 구축을 통해 공항 주변의 공공안전 및 자동화된 위험 인식 체계를 개발

주요 목표

- 실시간 영상 스트림에서 특정 개체(비행기, 새, 사람, 버스)를 탐지하고, 미리 정의된 위험 발생 시 즉각적인 경보 발생

분석 방법

- OpenCV를 통해 실시간 카메라 영상을 처리하고, GPIO 모듈을 이용해 감지 결과에 따른 경고 제공

사용 모델

- YOLOv8 딥러닝 모델을 사용하여 영상 내 세 가지 객체를 정확하게 식별

01 | 개요

개발 환경

하드웨어

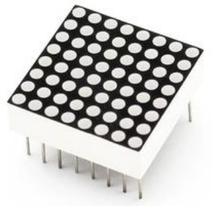
- 메인 디바이스 : Raspberry Pi 5 8G
- 카메라 입력 : USB Webcam 2대
- 출력 장치 : LED, LED Dot Matrix, Buzzer
- 기타 I/O : Raspberry Pi 5 GPIO interface

소프트웨어

- 운영체제 : Ubuntu, Raspberry Pi OS
- 프로그래밍 언어 : Python3, JavaScript
- AI 모델 : YOLOv8, DenseNet기반 학습모델

RaspberryPi

ubuntu



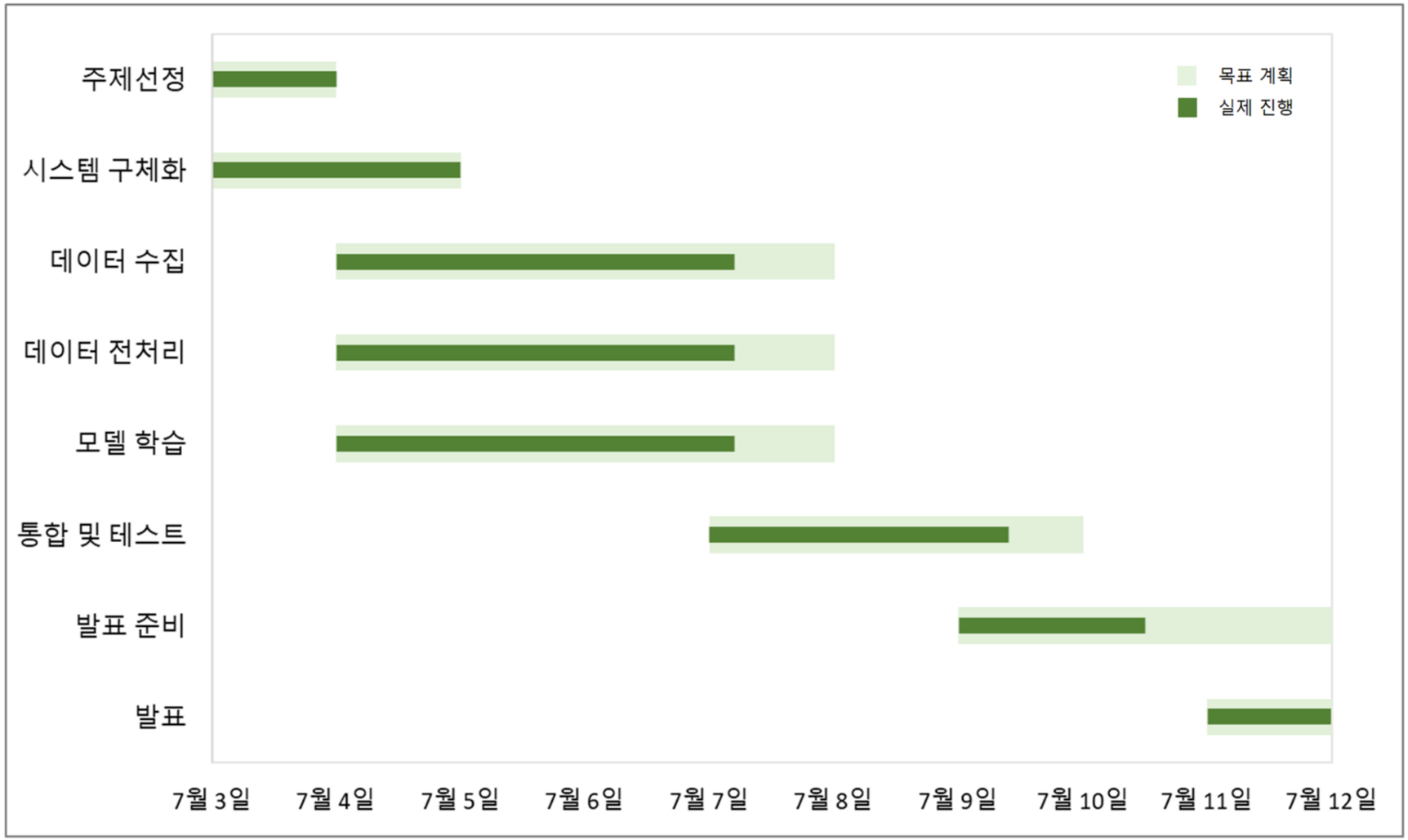
python

JavaScript

ultralytics
YOLOv8

01 | 개요

개발 일정


주제선정
시스템 구체화
데이터 수집
데이터 전처리
모델 학습
통합 및 테스트
발표 준비
발표
목표 계획
실제 진행
7월 3일
7월 4일
7월 5일
7월 6일
7월 7일
7월 8일
7월 9일
7월 10일
7월 11일
7월 12일

02 | 모델 소개

YOLOv8 : You Only Look Once, 객체 탐지를 위한 실시간 딥러닝 모델
PyTorch 기반으로 가볍고 빠르며, 한 번의 신경망 연산으로 이미지에서 객체의 위치와 종류를 동시에 예측하는 방식

특징	설명
높은 정확도	다양한 버전으로 정확도와 속도 조절 가능
실시간 속도	GPU 뿐만 아니라 CPU/라즈베리파이에서도 빠르게 실행
간편한 학습	커스텀 데이터로 쉽게 재학습 가능
쉬운 활용	Python 스크립트 (detect.py, train.py) 로 바로 실행 가능

02 | 모델 소개

탐지 대상 클래스

클래스명	설명	COCO 클래스 번호
PERSON	활주로 주변 인물 탐지	0
AIRPLANE	항공기	5
BUS	활주로 주변 버스 탐지	6
BIRD	조류 충돌 위험 판단	14

02 | 모델 소개

- **Airplane Classification** : 비행기 기종 분류를 위한 DenseNet201 기반 모델
  FGVC 데이터셋을 통해 100개의 클래스를 학습

- **탐지 대상 클래스**

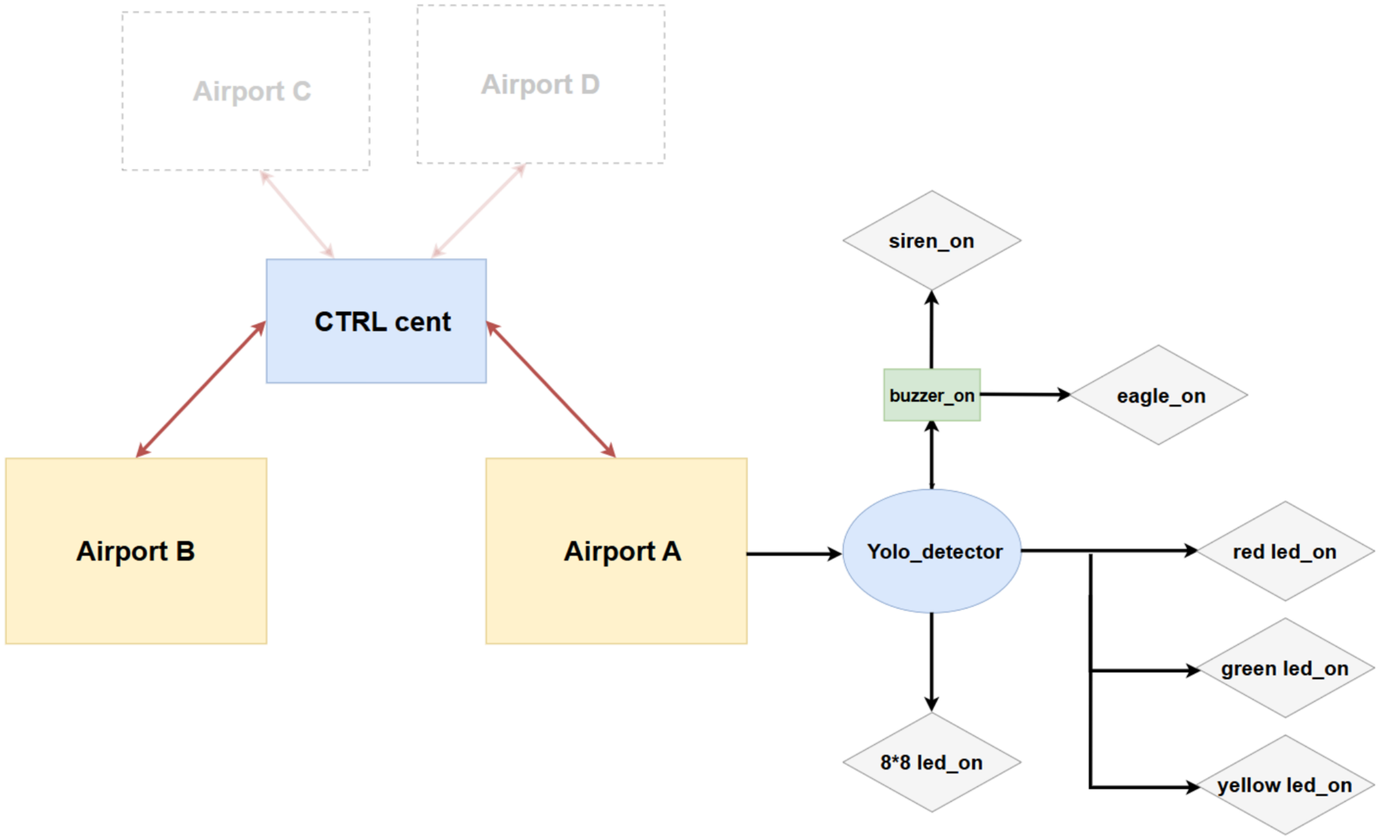
0:	"707-320"	25:	"A320"
1:	"727-200"	26:	"A321"
2:	"737-200"	27:	"A330-200"
3:	"737-300"	28:	"A330-300"
4:	"737-400"	29:	"A340-200"
5:	"737-500"	30:	"A340-300"
6:	"737-600"	31:	"A340-500"
7:	"737-700"	32:	"A340-600"
8:	"737-800"	33:	"A380"
9:	"737-900"	34:	"ATR-42"
10:	"747-100"	35:	"ATR-72"
11:	"747-200"	36:	"An-12"
12:	"747-300"	37:	"BAE 146-200"
13:	"747-400"	38:	"BAE 146-300"
14:	"757-200"	39:	"BAE-125"
15:	"757-300"	40:	"Beechcraft 1900"
16:	"767-200"	41:	"Boeing 717"
17:	"767-300"	42:	"C-130"
18:	"767-400"	43:	"C-47"
19:	"777-200"	44:	"CRJ-200"
20:	"777-300"	45:	"CRJ-700"
21:	"A300B4"	46:	"CRJ-900"
22:	"A310"	47:	"Cessna 172"
23:	"A318"	48:	"Cessna 208"
24:	"A319"	49:	"Cessna 525"

...

76:	"Fokker 100"
77:	"Fokker 50"
78:	"Fokker 70"
79:	"Global Express"
80:	"Gulfstream IV"
81:	"Gulfstream V"
82:	"Hawk T1"
83:	"Il-76"
84:	"L-1011"
85:	"MD-11"
86:	"MD-80"
87:	"MD-87"
88:	"MD-90"
89:	"Metroliner"
90:	"Model B200"
91:	"PA-28"
92:	"SR-20"
93:	"Saab 2000"
94:	"Saab 340"
95:	"Spitfire"
96:	"Tornado"
97:	"Tu-134"
98:	"Tu-154"
99:	"Yak-42"

03 | 시스템 구성

전체 시스템 구성도



CTRL cent (Control Center)

- 여러 공항(Airport A, B, C, D)과 연결되어 상호 채팅·통신
- 각 공항의 탐지 상태, 경고 정보, 로그 등을 실시간으로 공유

Airport A: YOLO 탐지 기반 관제

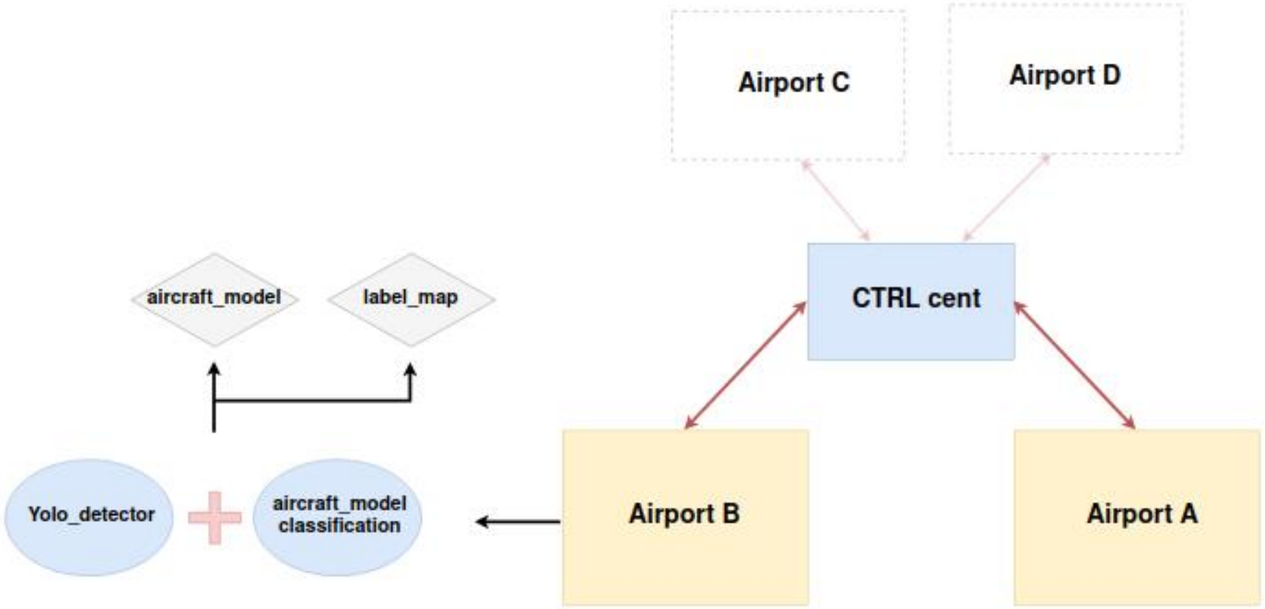
→ 탐지 조건에 따라 다양한 출력장치 제어

Siren : 비행기 + 사람 동시 탐지 등 조건 충족 시 작동
Eagle Sound : 새 떼 탐지 시 버저 제어 출력
8x8 LED Matrix : 비행기 탐지 시 항공기 안내 화살표 표시
LED :

- Red LED → 비행기
- Green LED → 버스
- Yellow LED → 비행기+버스 동시 탐지 시 작동

03 | 시스템 구성

전체 시스템 구성도



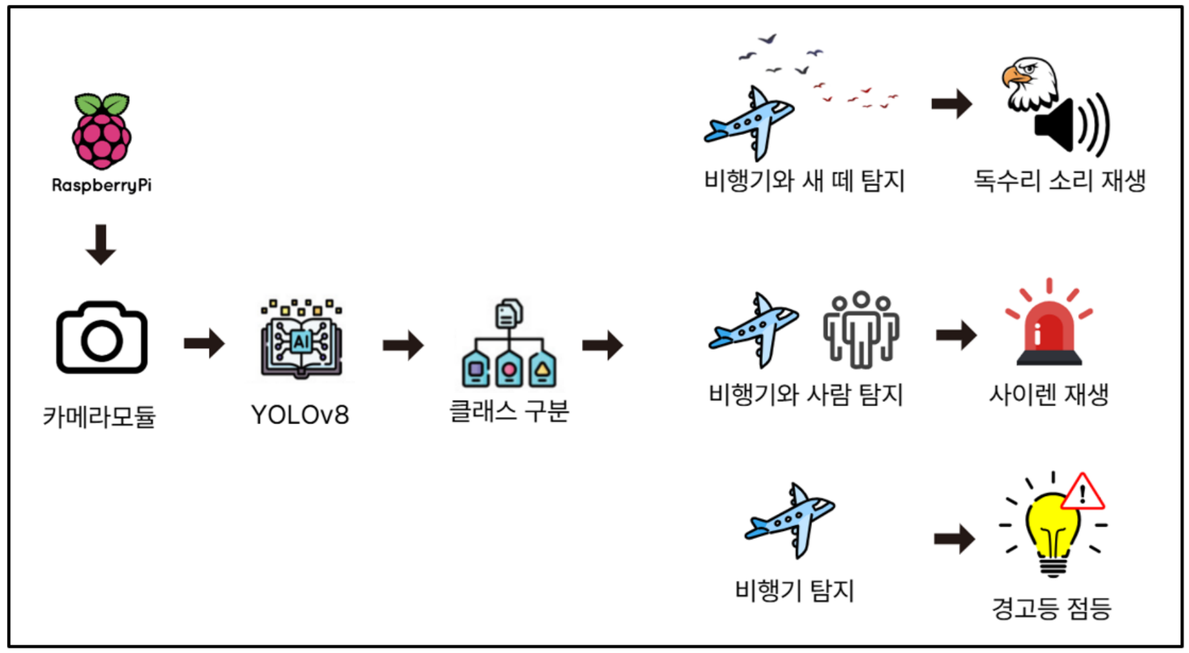
Airport B: YOLO & aircraft_model_classification
→ detection & classification 구조

aircraft_model.keras: 비행기 기종 100개의 클래스, 클래스 당 100장의 image로 구성된 FGVC 데이터셋으로 학습한 모델

label_map: FGVC 클래스를 구분하기 위한 .json 파일

03 | 시스템 구성

YOLO Detector의 제어 시나리오



1. YOLO 탐지

- YoloDetector로 프레임에서 객체(Label) 추출
- 조건: 새 5마리 이상, 비행기+사람, 버스, 비행기 등

2. 동작 분기

- 탐지 결과에 따라:
  - LED 제어 (버스, 비행기, 동시)
  - 8x8 화살표 표시
  - 사이렌
  - 독수리 소리
- process_frame()에서 판단 후 OpenCV로 박스 그려서 전송

03 | 시스템 구성 - CTRL center

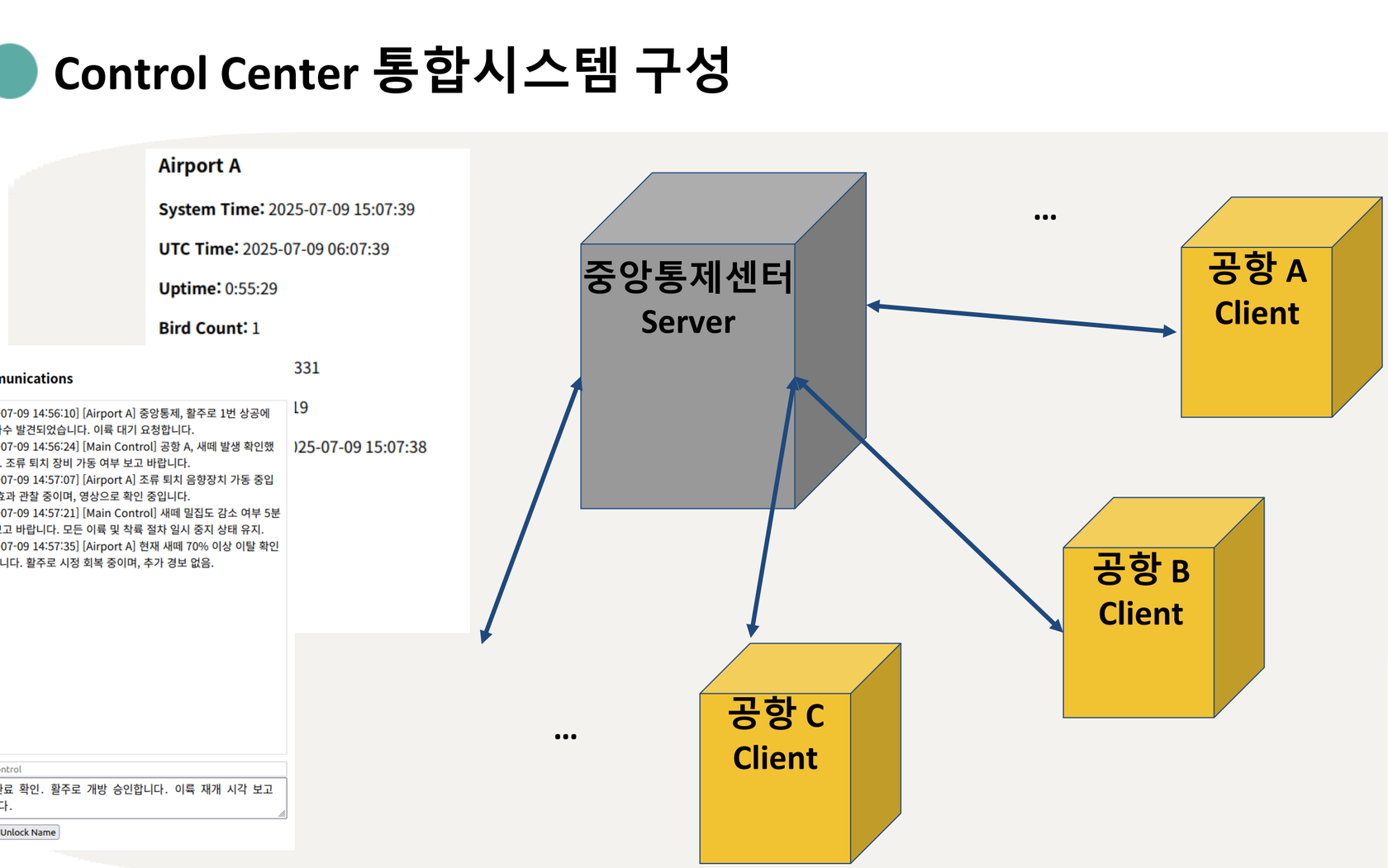
Control Center 주요 화면 & 기능

- **Aircraft Control System :**
  - **실시간 모니터링:** 각 공항의 탐지 상태, 카메라 상태, LED/사이렌 동작 등을 한 화면에서 확인 가능
  - **원격 제어:** API 버튼으로 각 공항의 LED, 사이렌, 독수리 소리 등을 직접 수동 제어 가능
  - **로그 관리:** 탐지 및 제어를 실시간으로 확인 가능
  - **채팅 기능:** Control center 및 각 공항 간 실시간 메시지 전송 가능

- **Control Center Dashboard :**
  - **실시간 모니터링:** 각 공항의 탐지 상태, 카메라 상태, LED/사이렌 동작 등을 한 화면에서 확인 가능
  - **로그 관리:** 탐지 및 제어를 실시간으로 확인 가능
  - **채팅 기능:** Control center 및 각 공항 간 실시간 메시지 전송 가능

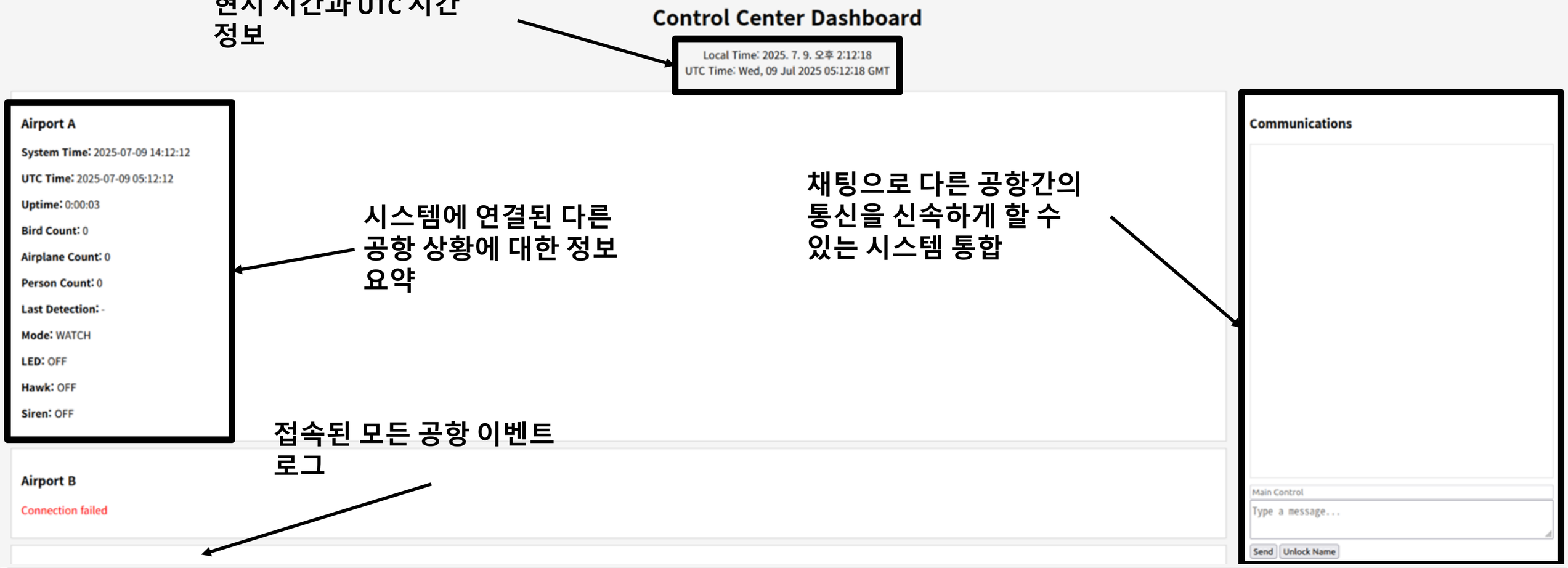
03 | 시스템 구성 - CTRL center

Control Center 통합시스템 구성



03 | 시스템 구성 - CTRL center

현지 시간과 UTC 시간 정보



시스템에 연결된 다른 공항 상황에 대한 정보 요약

채팅으로 다른 공항간의 통신을 신속하게 할 수 있는 시스템 통합

접속된 모든 공항 이벤트 로그

03 | 시스템 구성 - airport A

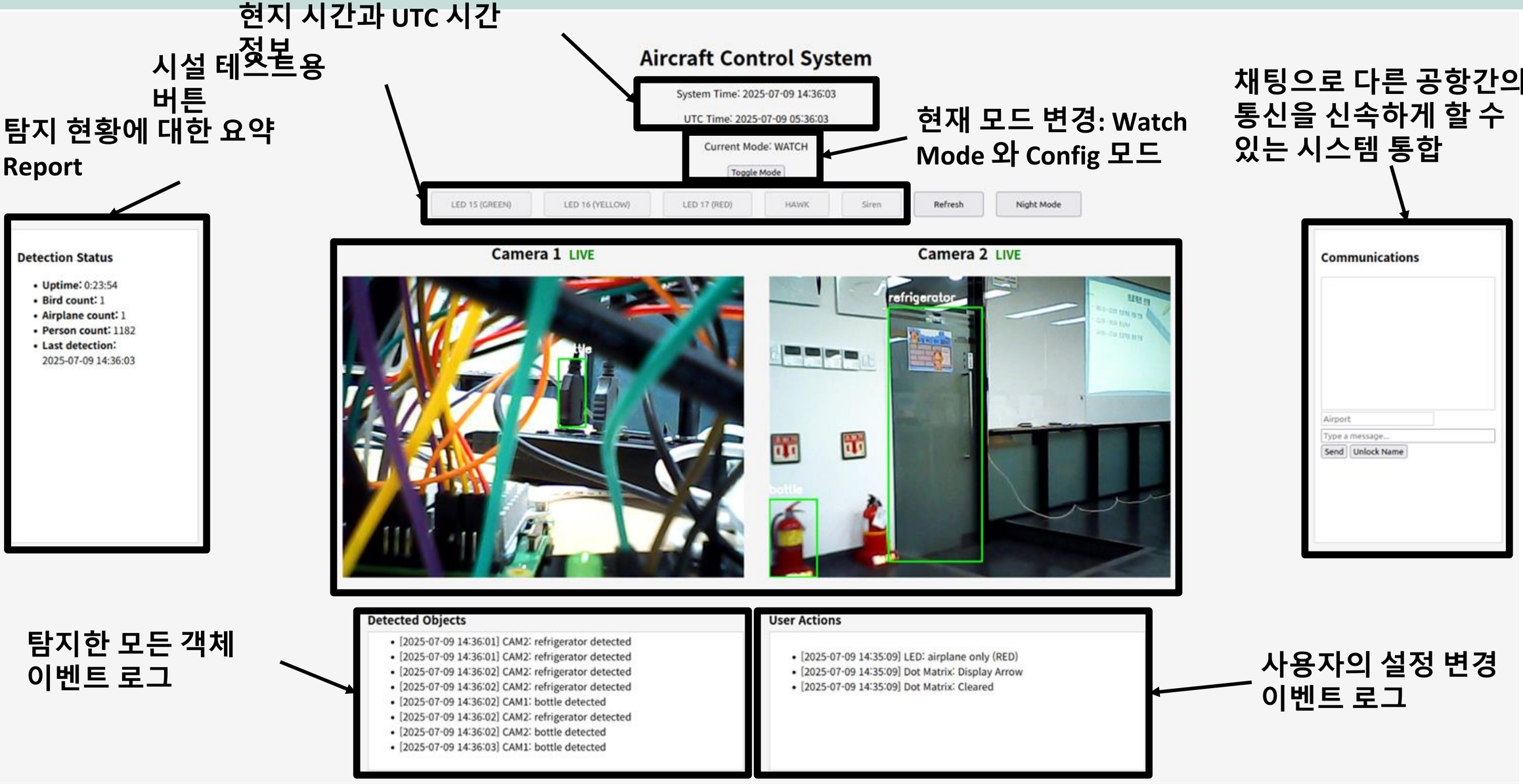
현지 시간과 UTC 시간 정보

시설 테스트용 버튼

탐지 현황에 대한 요약 Report

현재 모드 변경: Watch Mode 와 Config 모드

채팅으로 다른 공항간의 통신을 신속하게 할 수 있는 시스템 통합



탐지한 모든 객체 이벤트 로그

사용자의 설정 변경 이벤트 로그

03 | 시스템 구성 - airport A

주요 코드 (main_yolo.py)

```python
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/video0')
def video0():
    return Response(generate_frames(cap0, "CAM1", "camera0"),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/video1')
def video1():
    return Response(generate_frames(cap1, "CAM2", "camera1"),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/api/status')
def api_status():

@app.route('/api/mode/toggle')
def api_mode_toggle():

@app.route('/api/led/test/<int:num>')
def api_led_test(num):

@app.route('/api/hawk')
def api_hawk():

@app.route('/api/siren')
def api_siren():

@app.route('/api/chat/send', methods=['POST'])
def api_chat_send():

@app.route('/api/chat/messages')
def api_chat_messages():
```

- /api/status
→LED, 사이렌, 탐지 통계, 로그 등 JSON 반환
- /api/led/test/<num>
→ 특정 LED 직접 켜기/끄기
- /api/siren → 사이렌 수동 실행
- /api/hawk → 독수리 소리 수동 실행
- /api/chat/send → 메시지 전송
- /api/chat/messages → 메시지 목록 확인

- generate_frames():
  OpenCV로 프레임 캡쳐 → YOLO → 탐지 → JPEG 인코딩
  Flask Response로 클라이언트에 전달
- /video0, /video1 : 각 캠별 스트림

- 실시간 탐지 + 웹 제어 + GPIO interface를 통한 동작 기능

03 | 시스템 구성 - airport A

주요 코드 (yolo_detector.py)

```python
class YoloDetector:
    def __init__(self, model_path='yolov8n.pt'):
        self.model = YOLO(model_path)

    def detect(self, frame):
        results = self.model(frame, verbose=False)[0]
        boxes = results.boxes
        labels = [results.names[int(cls)] for cls in boxes.cls]
        coords = boxes.xyxy.cpu().numpy()
        return list(zip(labels, coords))
```

- ultralytics 라이브러리로 실시간 객체 탐지 수행
- YoloDetector 클래스에서 카메라 프레임 입력 → detect()로 Label과 좌표 추출
- 탐지된 Label에 따라 시나리오 동작

03 | 시스템 구성 - airport A

주요 코드 (led.py)

```python
from gpiozero import LED

led_airplane = LED(17)          # 비행기만
led_bus = LED(15)               # 버스만
led_bus_airplane = LED(16)      # 버스+비행기 동시

def set_led(airplane=False, bus=False, bus_airplane=False):

    if bus_airplane:
        led_bus_airplane.on()
        led_bus.off()
        led_airplane.off()
    elif bus:
        led_bus.on()
        led_bus_airplane.off()
        led_airplane.off()
    elif airplane:
        led_airplane.on()
        led_bus.off()
        led_bus_airplane.off()
    else:
        led_airplane.off()
        led_bus.off()
        led_bus_airplane.off()
```

- 탐지 결과에 따라 LED 상태 결정
  비행기 탐지 → red led on
  버스 탐지 → green led on
  비행기 + 버스 동시 탐지 → yellow led on

- set_led() 함수로 조건별 제어

03 | 시스템 구성 - airport A

주요 코드 (dot_matrix.py)

```python
from gpiozero import DigitalOutputDevice
import time

# BCM 기준 핀 번호
ROW_PINS = [2, 3, 4, 5, 6, 7, 8, 9]
COL_PINS = [10, 11, 12, 13, 19, 20, 21, 23]


ARROW_PATTERN = [
    0b11100111,
    0b11000011,
    0b10000001,
    0b00000000,
    0b11000011,
    0b11000011,
    0b11000011,
    0b11000011,
]
def display_arrow(refresh_rate=0.002):
    for row_idx, row_data in enumerate(ARROW_PATTERN):
        light_row(row_idx, row_data)
        time.sleep(refresh_rate)
```

- 비행기 탐지 시 항공기 안내 화살표 표시
- display_arrow : 행/열 핀 제어하여 패턴 출력
- ARROW_PATTERN으로 화살표 모양 정의

03 | 시스템 구성 - airport A

주요 코드 (siren.py , speaker.py)

```python
from time import sleep
from actions.buzzer_hw import buzzer

def trigger_siren(on=True):
    if on:
        buzzer.value = 0.7

    else:
        buzzer.value = 0
```

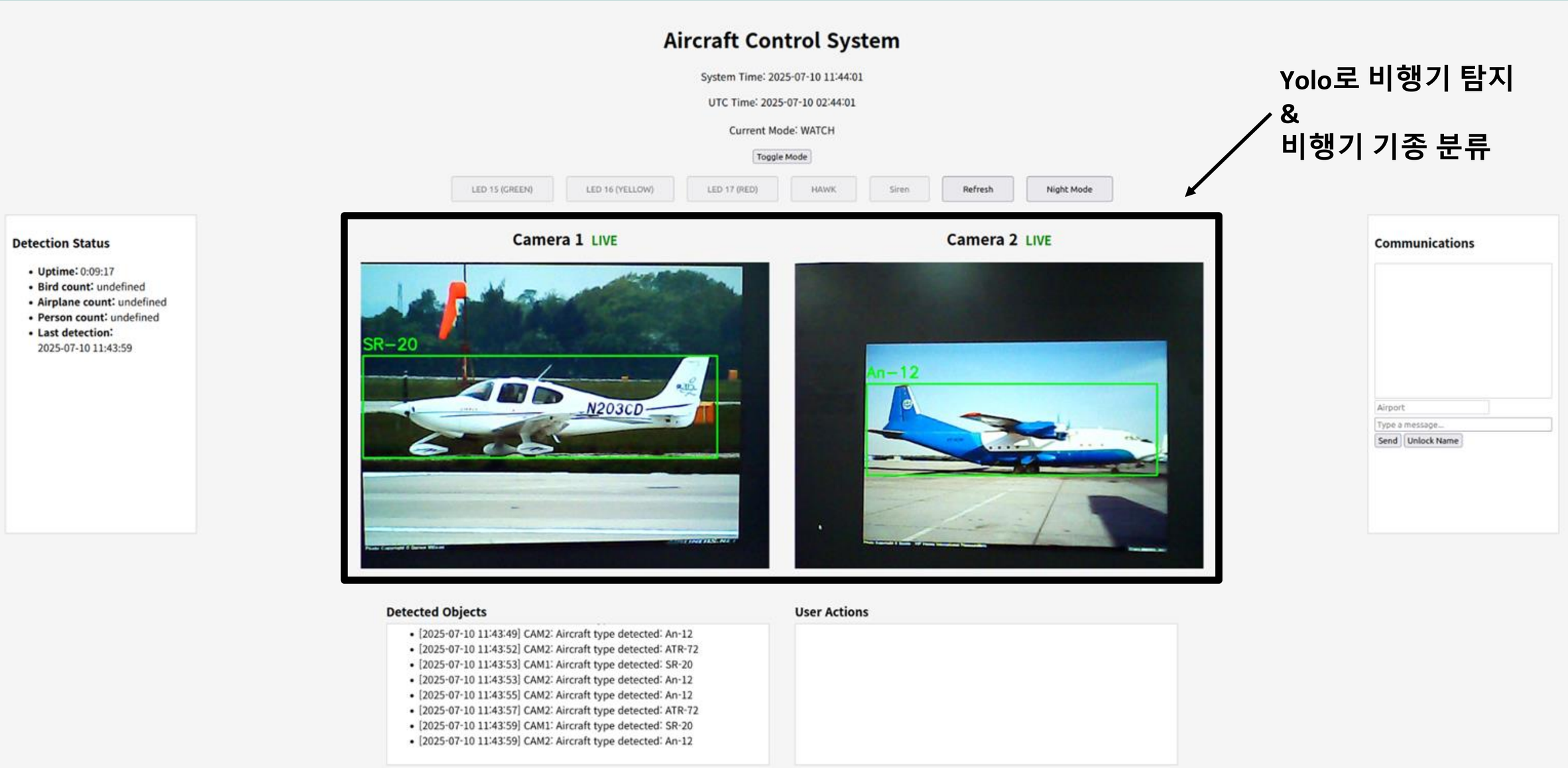
```python
from time import sleep
from actions.buzzer_hw import buzzer

def play_eagle_sound():
    print("EAGLE SOUND")

    # (1) 저음에서 고음으로 부드럽게 상승
    for freq in range(500, 1700, 60):
        buzzer.frequency = freq
        buzzer.value = 0.9
        sleep(0.03)
    buzzer.value = 0
    sleep(0.08)
```

- trigger_siren() :
  사람 + 비행기 동시 탐지 → 사이렌 소리 출력

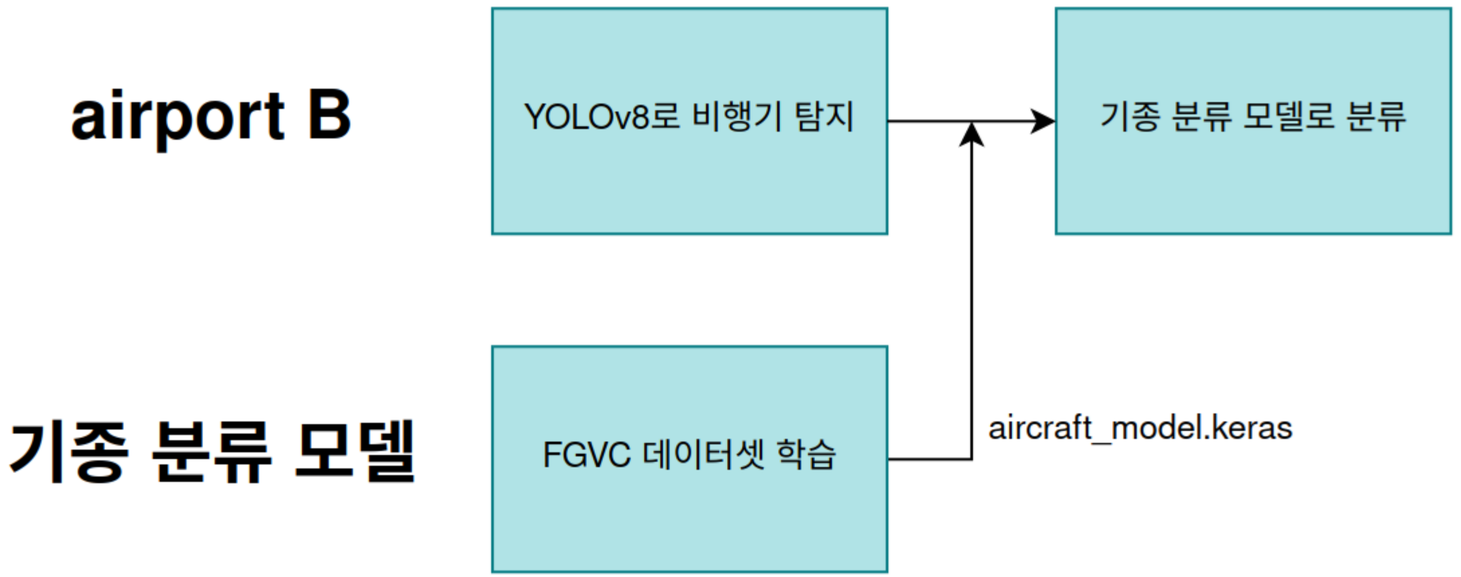
- play_eagle_sound() :
  새 무리 5마리 이상 탐지 → 독수리 소리 출력

GPIO 핀: PWMOutputDevice 사용해 주파수/볼륨 제어

03 | 시스템 구성 - airport B



03 | 시스템 구성 - airport B



03 | 시스템 구성 - airport B

- 기종 분류 모델 학습

- FGVC Aircraft 데이터셋 기반으로 100개 항공기 기종 분류 모델 구축
- DenseNet201 모델 기반 전이학습 및 fine-tuning 적용
- 고해상도 이미지 전처리 및 데이터 증강
- Train Accuracy 97%, Validation Accuracy 약 66%
- YOLO로 탐지된 항공기 crop 이미지를 분류기에 입력해 실시간 분류

```
209/209 ━━━━━━━━━━━━━━━━━━━━ 140s 604ms/step - accuracy: 0.0412 - loss: 4.9004 - val_accuracy: 0.1677 - val_loss: 3.9299
Epoch 2/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 111s 533ms/step - accuracy: 0.1984 - loss: 3.5077 - val_accuracy: 0.3066 - val_loss: 3.0807
Epoch 3/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 109s 522ms/step - accuracy: 0.3390 - loss: 2.8459 - val_accuracy: 0.3630 - val_loss: 2.7340
Epoch 4/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 109s 524ms/step - accuracy: 0.4414 - loss: 2.4226 - val_accuracy: 0.3810 - val_loss: 2.6683
Epoch 5/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 108s 516ms/step - accuracy: 0.5010 - loss: 2.1314 - val_accuracy: 0.4326 - val_loss: 2.4856
Epoch 6/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 108s 515ms/step - accuracy: 0.5337 - loss: 1.9949 - val_accuracy: 0.4578 - val_loss: 2.3403
Epoch 7/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 107s 514ms/step - accuracy: 0.5824 - loss: 1.8499 - val_accuracy: 0.4575 - val_loss: 2.3001
Epoch 8/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 107s 513ms/step - accuracy: 0.6264 - loss: 1.7261 - val_accuracy: 0.4704 - val_loss: 2.3180
Epoch 9/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 108s 515ms/step - accuracy: 0.6670 - loss: 1.5737 - val_accuracy: 0.4806 - val_loss: 2.2853
Epoch 10/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 108s 517ms/step - accuracy: 0.6967 - loss: 1.4825 - val_accuracy: 0.4908 - val_loss: 2.2806
Epoch 11/40
209/209 ━━━━━━━━━━━━━━━━━━━━ 107s 514ms/step - accuracy: 0.6959 - loss: 1.4648 - val_accuracy: 0.4791 - val_loss: 2.2892
```

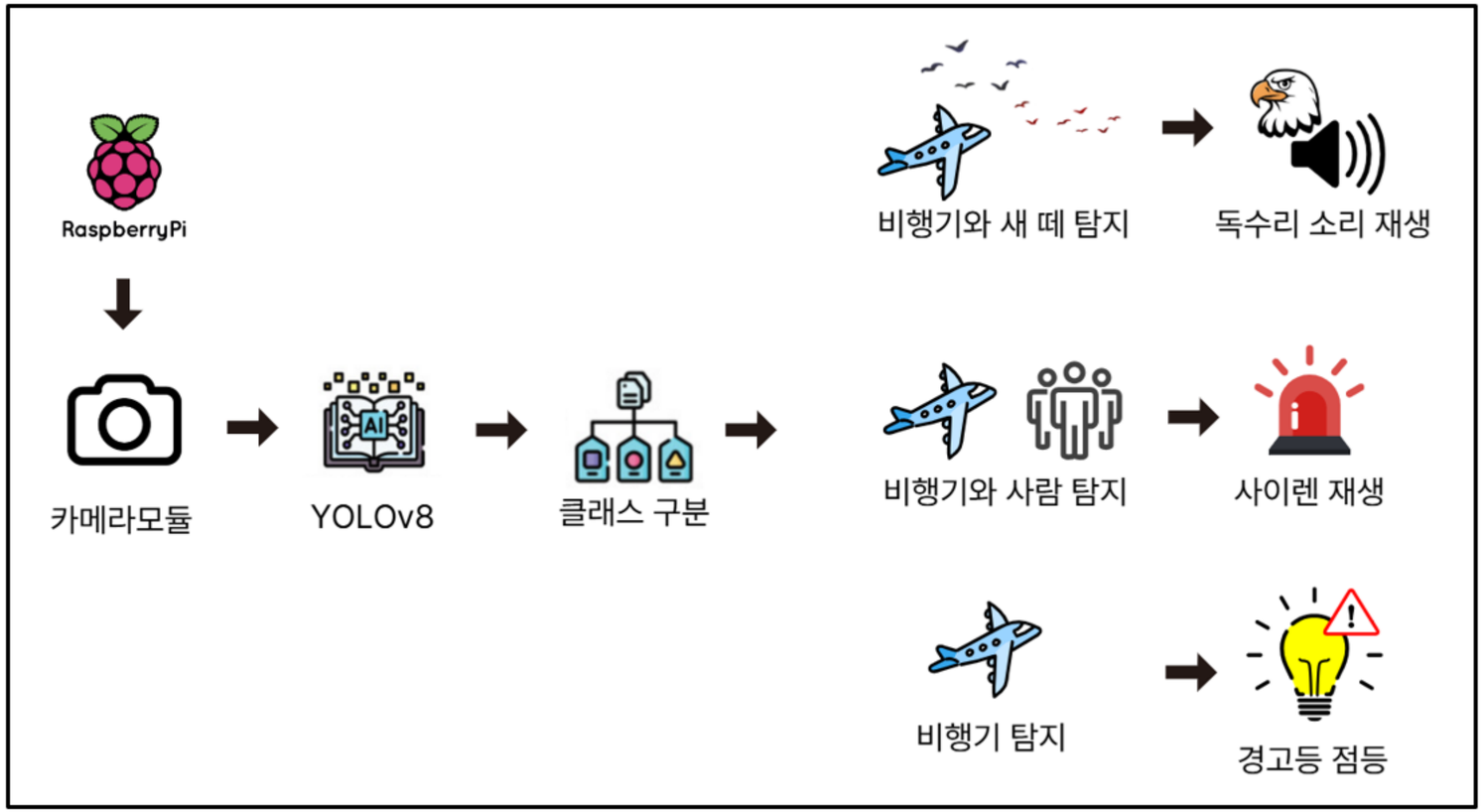
03 | 시스템 구성 - airport B

- 기종 분류 모델 학습

707-320	0.52	0.39	0.45	33
727-200	0.71	0.52	0.60	33
737-200	0.47	0.50	0.49	34
737-300	0.23	0.09	0.13	33
737-400	0.42	0.24	0.31	33
737-500	0.37	0.47	0.42	34
737-600	0.42	0.76	0.54	33
737-700	0.33	0.48	0.39	33
737-800	0.39	0.26	0.32	34
737-900	0.68	0.58	0.62	33
747-100	0.42	0.42	0.42	33
747-200	0.24	0.44	0.31	34
747-300	0.50	0.21	0.30	33
747-400	0.44	0.42	0.43	33
757-200	0.54	0.41	0.47	34
757-300	0.83	0.61	0.70	33
767-200	0.41	0.64	0.50	33
767-300	0.21	0.21	0.21	34
767-400	0.52	0.82	0.64	33
777-200	0.42	0.30	0.35	33
777-300	0.92	0.32	0.48	34
A300B4	0.41	0.58	0.48	33
A310	0.24	0.58	0.34	33
A318	0.79	0.79	0.79	34
A319	0.52	0.70	0.60	33
A320	0.40	0.42	0.41	33
A321	0.43	0.35	0.39	34
A330-300	1.00	0.12	0.22	33
A340-200	0.45	0.56	0.50	34
A340-300	0.55	0.36	0.44	33
A340-500	0.77	0.61	0.68	33
A340-600	0.66	0.62	0.64	34
A380	0.91	0.64	0.75	33
ATR-42	0.57	0.70	0.63	33
ATR-72	0.75	0.62	0.68	34
An-12	0.97	0.88	0.92	33
BAE 146-200	0.62	0.85	0.72	33
BAE 146-300	0.85	0.32	0.47	34
BAE-125	0.75	0.82	0.78	33
Beechcraft 1900	0.91	0.91	0.91	33
Boeing 717	0.49	0.62	0.55	34
C-130	0.76	0.94	0.84	33
C-47	0.42	0.42	0.42	33
CRJ-200	0.63	0.71	0.67	34
CRJ-700	0.72	0.55	0.62	33
CRJ-900	0.68	0.70	0.69	33
Cessna 172	0.96	0.79	0.87	34
Cessna 208	0.76	0.94	0.84	33
Cessna 525	0.96	0.73	0.83	33
Cessna 560	0.73	0.88	0.80	34
Challenger 600	0.77	0.91	0.83	33
DC-10	0.62	0.55	0.58	33
DC-3	0.43	0.53	0.47	34
DC-6	0.64	0.76	0.69	33
DC-8	0.63	0.36	0.46	33
DC-9-30	0.53	0.53	0.53	34
DH-82	0.91	0.94	0.93	33
DHC-1	1.00	0.85	0.92	33
DHC-6	0.88	0.68	0.77	34
DHC-8-100	0.74	0.79	0.76	33
DHC-8-300	0.78	0.55	0.64	33
DR-400	0.94	0.91	0.93	34
Dornier 328	0.79	0.94	0.86	33
E-170	0.58	0.79	0.67	33
E-190	0.46	0.53	0.49	34
E-195	0.64	0.55	0.59	33
EMB-120	0.74	0.88	0.81	33
ERJ 135	0.68	0.62	0.65	34
ERJ 145	0.58	0.42	0.49	33
Embraer Legacy 600	0.74	0.76	0.75	33
Eurofighter Typhoon	0.87	0.97	0.92	34
F-16A/B	0.91	0.91	0.91	33
F/A-18	0.97	0.85	0.90	33
Falcon 2000	0.70	0.82	0.76	34
Falcon 900	0.96	0.70	0.81	33
Fokker 100	0.84	0.48	0.62	33
Fokker 50	1.00	0.74	0.85	34
Fokker 70	0.64	0.91	0.75	33
Global Express	0.63	0.79	0.70	33
Gulfstream IV	0.62	0.71	0.66	34
Gulfstream V	0.75	0.45	0.57	33
Hawk T1	0.97	0.94	0.95	33
Il-76	0.97	0.94	0.96	34
L-1011	0.60	0.82	0.69	33
MD-11	0.58	0.64	0.61	33
MD-80	0.40	0.12	0.18	34
MD-87	0.42	0.58	0.49	33
MD-90	0.52	0.39	0.45	33
Metroliner	0.81	0.88	0.85	34
Model B200	1.00	0.85	0.92	33
PA-28	0.71	0.82	0.76	33
SR-20	1.00	0.97	0.99	34
Saab 2000	0.76	0.88	0.82	33
Saab 340	0.74	0.85	0.79	33
Spitfire	0.93	0.82	0.88	34
Tornado	0.82	1.00	0.90	33
Tu-134	0.79	0.70	0.74	33
Tu-154	0.85	0.68	0.75	34
Yak-42	0.75	0.82	0.78	33
accuracy			0.64	3333
macro avg	0.67	0.64	0.64	3333
weighted avg	0.67	0.64	0.64	3333

04 | Use Case

Use Case



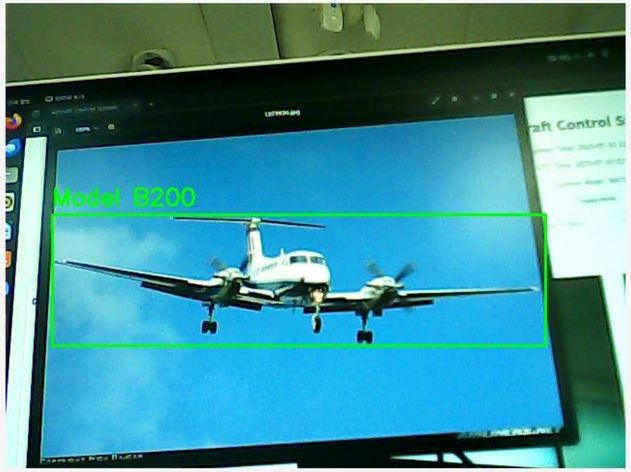
04 | Use Case

Use Case

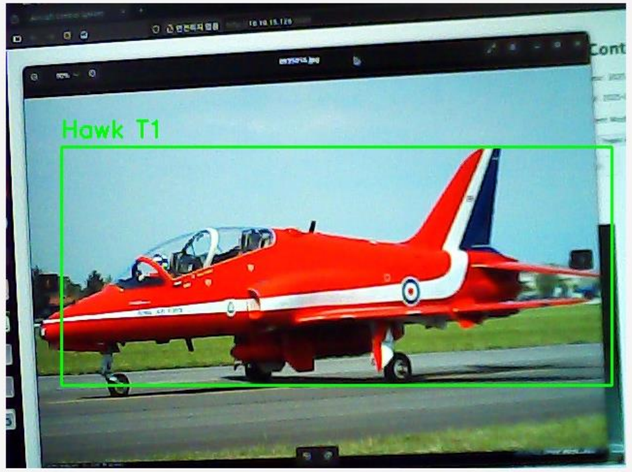
Use Case	설명
객체 탐지	YOLOv8로 프레임에서 비행기, 버스, 사람, 새 탐지
탐지 결과 처리	조건에 따라 LED 색상, 8x8 led(화살표), 사이렌, 독수리 소리 출력
LED 제어	비행기: 빨간 LED, 버스: 초록 LED, 비행기+버스: 노란 LED
8x8 Matrix 표시	비행기 탐지 시 8x8 Matrix에 화살표 패턴 표시
사이렌 출력	비행기 + 사람 동시 탐지 시 경고 사이렌 출력
독수리 소리 출력	새 무리 5마리 이상 탐지 시 독수리 소리 발생
원격 접속 & 제어	웹 브라우저로 영상 모니터링 + 상태 확인 + API로 직접 제어
Control Center 연동	공항별 상태 실시간 확인, 채팅, 로그 공유
수동 테스트 모드	관리자 모드에서 LED/사이렌 수동 on/off 가능

04 | 시연 결과 - 비행기 기종 분류

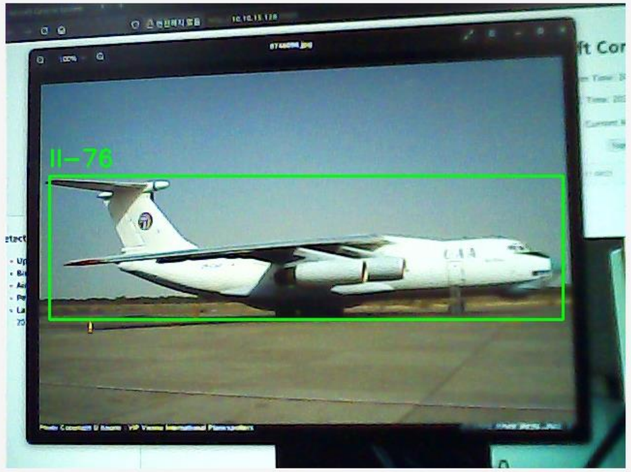
Camera 2 LIVE



Camera 2 LIVE



Camera 2 LIVE



Camera 2 LIVE

04 | 시연 영상

시연 영상

05 | 트러블 슈팅 및 고찰

트러블 슈팅

문제 1.

GPIO 핀 제어를 Python 가상환경(Venv) 안에서 실행하려 했으나, 하드웨어 접근 권한 문제 발생

문제 2.

8x8 매트릭스 LED 깜빡이는 문제 발생

해결

가상환경 밖(시스템 Python)에서 gpiozero 라이브러리로 실행하여 정상 제어 가능

해결

display_arrow() 함수의 refresh rate 조절하여 해결

05 | 트러블 슈팅 및 고찰

트러블 슈팅

문제 3.

단일 부저로 다중 사운드 출력을 하는 과정에서 충돌발생

문제 4.

버스+비행기 동시 감지 시 중복 반응

해결

buzzer_hw.py모듈을 통해 부저 사용 모듈 동작 병행 처리

해결

상태 flag(status)를 두어 중복 처리 방지

05 | 트러블 슈팅 및 고찰

트러블 슈팅

문제 5.

비행기 기종 분류 학습 정확도 낮음
→ 초기 모델은 학습 정확도 대비 검증 정확도가 낮아 과적합 및 일반화 문제 발생

해결

전이학습, 이미지 augmentation 다양화, fine-tuning 구간 조정, learning rate 조절 등을 통해 성능 개선

학습률 감소 및 후반부 레이어만 unfreeze 하여 과적합 최소화

최종적으로 val_accuracy 약 66%, loss 안정화 달성

05 | 트러블 슈팅 및 고찰

고찰

시스템 통합 설계 능력의 중요성	하드웨어 간의 모듈 간 인터페이스를 설계하고 통합하는 과정은 실제 설계 직무에서 다양한 설비 간 신호 흐름, 동작 조건, 예외처리를 고려하는 시스템 설계와 매우 유사함
상태 관리 및 로깅 설계	감지 상태와 로깅 시스템을 통해 시스템 동작 기록 및 오류 추적이 가능하게 하여 실제 설계에서 센서 로그, 비정상 이벤트 트래킹 구조 설계와 밀접한 연관이 있음
협업/역할 분담의 중요성	관제센터 관리, 기능 개발, 항공기 기종 분류 등 역할을 나눠 맡아 진행하면서 개발 효율성과 완성도를 높였고, 각 파트 간 연계와 소통의 중요성을 경험
시스템 타이밍 제어	YOLO 탐지 이후 Classification까지 순차적 처리 필요 → 기반 제어 로직/ 타이밍 조율 회로 설계 필요 동기화, 레지스터 스테이지 분리, FSM 상태 관리 등의 RTL 설계가 중요
인공지능 학습 모델 연산 병목	CNN 모델은 Convolution 연산이 매우 많아 연산 병목 발생 → 연산을 병렬로 처리할 수 있도록 Convolution 엔진 설계 필요 (ex. RTL 기반 MAC array 설계)
데이터 이동 병목	Feature map 및 weight의 이동이 많아 외부 메모리 접근이 잦고 지연 발생 → 효율적인 데이터 흐름 제어를 위한 버퍼 제어기와 메모리 인터페이스 설계 필요 (ex. RTL 기반 Buffer/ Memory Controller)

06 | 결론

성과

- 라즈베리파이 기반의 완전한 온디바이스 실시간 객체 탐지 시스템 구축
- 경고반응 : LED, Dot Matrix, Buzzer(사이렌, 독수리 소리)
- Flash 기반의 웹 인터페이스 : 실시간 카메라 스트리밍, 상태 API 및 각종 제어 API, 객체 감지 통계 시각화, 경고 시스템 수동 제어, 로그 저장 및 표시 등
- 100개 클래스 항공기 기종 분류 모델 연동

06 | 결론

확장 가능성 및 개선사항

- TTS 를 상황에 맞게 다양화 하여 사용자 친화적 시스템 구축
- 비행기 기종 분류 모델 정확도 향상
- 비행기 외에도 드론, 헬기 범용성 확장

THANK YOU
Q & A