직접 실행되는 python코드(최희우 담당)

```python
import RPi.GPIO as GPIO
import time
import cv2
import torch
import numpy as np
from picamera2 import Picamera2
from models.common import DetectMultiBackend

# GPIO 핀 설정: 초음파 센서의 TRIG(출력) 및 ECHO(입력) 핀 번호
TRIG_PIN = 24
ECHO_PIN = 23

# GPIO 모드 및 핀 초기화
GPIO.setmode(GPIO.BCM)  # GPIO 핀 모드를 BCM으로 설정
GPIO.setup(TRIG_PIN, GPIO.OUT)  # TRIG 핀을 출력으로 설정
GPIO.setup(ECHO_PIN, GPIO.IN)  # ECHO 핀을 입력으로 설정

# YOLOv5 모델 로드: 사전에 학습된 모기 탐지 모델 경로 및 장치 설정
model_path = '/home/user/Downloads/best.pt'  # 학습된 모델 파일 경로
device = 'cpu'  # 모델을 실행할 장치 설정 (예: CPU)
model = DetectMultiBackend(model_path, device=device)  # YOLOv5 모델 초기화

# Picamera2 초기화 및 설정
picam2 = Picamera2()  # Picamera2 객체 생성
picam2.configure(picam2.create_preview_configuration(main={"size": (640, 480)}))  # 해상도 및 설정 구성
picam2.start()  # 카메라 스트림 시작

def initialize_command_file():
    """명령 파일 초기화: /tmp/command.txt 파일을 비움."""
    try:
        with open('/tmp/command.txt', 'w') as f:
            f.write('')  # 파일 내용을 비움
        print("명령 파일 초기화 완료")
    except Exception as e:
        print(f"명령 파일 초기화 오류: {e}")

def measure_distance():
    """
    초음파 센서를 사용해 거리를 측정.
    TRIG 핀으로 초음파 신호를 송출하고 ECHO 핀으로 반사 신호를 수신하여 거리 계산.
    """
    GPIO.output(TRIG_PIN, False)  # TRIG 핀을 LOW로 설정 (초기화)
    time.sleep(0.1)  # 잠시 대기 (안정화)

    # 초음파 신호 송출
    GPIO.output(TRIG_PIN, True)  # TRIG 핀을 HIGH로 설정
    time.sleep(0.00001)  # 10마이크로초 동안 신호 유지
    GPIO.output(TRIG_PIN, False)  # TRIG 핀을 다시 LOW로 설정

    # ECHO 핀 신호를 수신하여 시간 측정
    pulse_start = time.time()
    pulse_end = time.time()
```

```
53
54      # ECHO 핀이 HIGH가 될 때까지 대기 (신호 시작 시간 기록)
55      while GPIO.input(ECHO_PIN) == 0:
56          pulse_start = time.time()
57          if pulse_start - pulse_end > 0.02:  # 타임아웃 방지
58              return 999  # 범위를 초과한 거리 반환
59
60      # ECHO 핀이 LOW가 될 때까지 대기 (신호 종료 시간 기록)
61      while GPIO.input(ECHO_PIN) == 1:
62          pulse_end = time.time()
63          if pulse_end - pulse_start > 0.02:  # 타임아웃 방지
64              return 999  # 범위를 초과한 거리 반환
65
66      # 초음파 이동 시간 계산
67      pulse_duration = pulse_end - pulse_start
68      distance = pulse_duration * 17150  # 거리(cm) 계산 (음속 343m/s 기준)
69      return round(distance, 2)  # 소수점 2자리로 반올림
70
71  def preprocess_frame(frame):
72      """
73      YOLOv5 모델 입력을 위한 영상 전처리.
74      - BGR -> RGB 색상 변환
75      - HWC -> CHW 형태로 변경
76      - 정규화 (0~1 범위)
77      """
78      frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  # BGR 이미지를 RGB로 변환
79      frame = np.transpose(frame, (2, 0, 1))  # HWC -> CHW 형태로 변경
80      frame = np.ascontiguousarray(frame, dtype=np.float32) / 255.0  # 정규화
81      return torch.from_numpy(frame).unsqueeze(0).to(device)  # 텐서 변환 후 추가 차원
82
83  def detect_mosquito():
84      """
85      모기 탐지 함수.
86      - Picamera2로 이미지를 캡처하고 YOLOv5 모델로 모기 여부를 판단.
87      """
88      frame = picam2.capture_array()  # 카메라로 이미지 캡처
89      frame_bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)  # RGB 이미지를 BGR로 변환
90      input_tensor = preprocess_frame(frame_bgr)  # YOLO 모델 입력 형태로 전처리
91      results = model(input_tensor)  # YOLO 모델로 객체 감지
92
93      conf_thres = 0.25  # 신뢰도 임계값
94      detections = []  # 감지 결과 저장
95
96      if isinstance(results[0], torch.Tensor):  # 결과가 텐서인지 확인
97          results_np = results[0].detach().cpu().numpy()  # 텐서를 NumPy 배열로 변환
98          for det in results_np:
99              if det.ndim == 2:  # 감지된 객체가 있는지 확인
100                 for inner_det in det:
101                     confidence = inner_det[4]  # 신뢰도 값
102                     if len(inner_det) >= 6 and confidence >= conf_thres:  # 조건 충족 시
103                         detections.append(inner_det)
```

```python

    # 모기가 감지되었는지 확인
    for detection in detections:
        x1, y1, x2, y2, conf, cls = detection[:6]
        label = model.names[int(cls)]  # 클래스 이름
        if label == 'mosquito':  # 'mosquito' 클래스로 탐지 시
            return True
    return False

def write_command_to_file(command):
    """
    명령을 /tmp/command.txt 파일에 기록.
    """
    try:
        print(f"명령 기록 중: {command}")  # 디버깅 출력
        with open('/tmp/command.txt', 'w') as f:
            f.write(command + '\n')  # 명령 파일에 쓰기
        time.sleep(3)  # 명령 처리 대기
        initialize_command_file()  # 명령 파일 초기화
    except Exception as e:
        print(f"명령 파일 쓰기 오류: {e}")

try:
    initialize_command_file()  # 초기화

    while True:
        distance = measure_distance()  # 거리 측정
        print(f"Measured distance: {distance} cm")

        if 18 < distance <= 40:  # 18~40cm 범위 내에서 모기 탐지 시도
            print("40cm 이내로 감지됨. 카메라 작동 시작...")
            if detect_mosquito():
                print("모기 감지됨! C 프로그램에 명령 전달: spray_attractant")
                write_command_to_file("spray_attractant")  # 유인제 분사 명령
                time.sleep(3)
            else:
                print("모기 없음.")

        if distance <= 18:  # 18cm 이내에서 문 닫기 및 살충제 분사
            print("18cm 이내로 감지됨. C 프로그램에 명령 전달: close_door_and_spray_pesticide")
            write_command_to_file("close_door_and_spray_pesticide")
            time.sleep(9)

        time.sleep(1)  # 센서 측정 주기

except KeyboardInterrupt:
    print("프로그램 종료")
finally:
    picam2.stop()  # 카메라 중지
    GPIO.cleanup()  # GPIO 리소스 해제

```

직접 실행하는 c코드(장수인 담당)

```c
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

// GPIO 핀 번호 정의
#define SERVO_PIN1_1 0   // 유인제 분사용 서보모터1
#define SERVO_PIN1_2 24  // 유인제 분사용 서보모터2
#define SERVO_PIN2 6     // 문 닫기/열기용 서보모터
#define SERVO_PIN3_1 28  // 살충제 분사용 서보모터1
#define SERVO_PIN3_2 29  // 살충제 분사용 서보모터2

// 마지막으로 처리된 명령 저장
char lastCommand[256] = "";

// 서보모터의 각도를 설정하는 함수
void setServoAngle(int pin, int angle) {
    // 서보모터 각도에 따른 듀티 사이클 계산
    int dutyCycle = 5 + (angle * 20) / 180; // 0도=5, 180도=25
    printf("핀 %d, 각도 %d도, 듀티 사이클 %d\n", pin, angle, dutyCycle); // 디버깅 출력
    softPwmWrite(pin, dutyCycle); // 듀티 사이클 설정
    delay(500); // 0.5초 대기
}

// 서보모터를 초기화하거나 멈추는 함수
void resetServo(int pin) {
    setServoAngle(pin, 0); // 각도를 0도로 설정 (초기 위치)
    delay(500); // 동작 완료를 위해 대기
    softPwmWrite(pin, 0); // PWM 신호 중단
    printf("핀 %d 서보모터 정지\n", pin);
}

// 유인제 분사 함수
void sprayAttractant() {
    printf("유인제 분사\n");
    // 두 개의 유인제 분사용 서보모터를 작동
    setServoAngle(SERVO_PIN1_1, 90); // 90도로 이동
    setServoAngle(SERVO_PIN1_2, 90);
    delay(1000); // 1초 대기
    // 서보모터를 초기 위치로 복원
    resetServo(SERVO_PIN1_1);
    resetServo(SERVO_PIN1_2);
}

// 문 열기 함수
void openDoor() {
    printf("문 열기\n");
    setServoAngle(SERVO_PIN2, 0); // 0도로 이동 (문 열기)
    softPwmWrite(SERVO_PIN2, 0); // PWM 신호 중단
}
```

```c

// 문 닫기 및 살충제 분사 함수
void closeDoorAndSprayPesticide() {
    printf("문 닫기\n");
    setServoAngle(SERVO_PIN2, 180); // 180도로 이동 (문 닫기)
    delay(3000); // 3초 대기
    softPwmWrite(SERVO_PIN2, 0); // PWM 신호 중단

    printf("살충제 분사\n");
    // 두 개의 살충제 분사용 서보모터를 작동
    setServoAngle(SERVO_PIN3_1, 90); // 90도로 이동
    setServoAngle(SERVO_PIN3_2, 90);
    delay(1000); // 1초 대기
    // 서보모터를 초기 위치로 복원
    resetServo(SERVO_PIN3_1);
    resetServo(SERVO_PIN3_2);
}

int main() {
    // WiringPi 초기화
    if (wiringPiSetup() == -1) {
        printf("WiringPi 초기화 실패!\n");
        return -1; // 오류 종료
    }

    // 서보모터 초기화 (PWM 생성)
    if (softPwmCreate(SERVO_PIN1_1, 0, 200) != 0 ||
        softPwmCreate(SERVO_PIN1_2, 0, 200) != 0 ||
        softPwmCreate(SERVO_PIN2, 0, 200) != 0 ||
        softPwmCreate(SERVO_PIN3_1, 0, 200) != 0 ||
        softPwmCreate(SERVO_PIN3_2, 0, 200) != 0) {
        printf("소프트웨어 PWM 초기화 실패!\n");
        return -1; // 오류 종료
    }

    printf("서보모터 제어 프로그램 시작\n");

    // 무한 반복 루프 (명령 대기)
    while (1) {
        FILE *file = fopen("/tmp/command.txt", "r"); // 명령 파일 열기
        if (file) {
            char command[256] = {0};
            // 파일에서 명령 읽기
            if (fscanf(file, "%s", command) == 1) {
                printf("명령 읽음: %s\n", command); // 디버깅 출력
                strcpy(lastCommand, command);

                // 유인제 분사 명령 처리
                if (strcmp(command, "spray_attractant") == 0) {
                    sprayAttractant();
                }
                // 문 닫기 및 살충제 분사 명령 처리
                else if (strcmp(command, "close_door_and_spray_pesticide") == 0) {
                    closeDoorAndSprayPesticide();
                    delay(3000); // 명령 처리 후 대기
                    openDoor(); // 문 열기
                }
            }
            fclose(file); // 파일 닫기
        }
        usleep(500000); // 0.5초 대기
    }

    return 0; // 프로그램 종료
}

```

1. 이 코드 모두 라즈베리파이 안에서 돌렸던 코드라 윈도우 환경에서 오류가 뜬다는 점 참고해 주시기 바랍니다.

2. 박수빈은 딥러닝 학습시키는 부분 코드 담당이라 직접 실행하는 부분에는 담당이 없다는 점 양해부탁드립니다.