SW개발/HW제작 설계서

프로젝트 명 : 객체 인식을 통한 보행자 안전 보호 알림 장치

2024. 07. 16

(팀명) 엘리펀트

수행 단계별 주요 산출물

단계	산출물	일반 ·모바일 APP ·Web 등	응용 소프트웨어 ·빅데이터 ·인공지능 ·블록체인 등	응용 하드웨어 ·IoT ·로봇 ·드론 등
환경 분석	시장/기술 환경 분석서	△	△	△
	설문조사 결과서	△	△	△
	인터뷰 결과서	△	△	△
요구사항 분석	요구사항 정의서	○	○	○
	유즈케이스 정의서	▲	▲	▲
아키텍처 설계	서비스 구성도(시스템 구성도)	○	○	○
	서비스 흐름도(데이터 흐름도)	△	○	△
	UI/UX 정의서	△	△	△
	하드웨어/센서 구성도	-	-	○
기능 설계	메뉴 구성도	○	○	○
	화면 설계서	○	○	△
	엔티티 관계도	○	○	△
	기능 처리도(기능 흐름도)	○	○	○
	알고리즘 명세서/설명서	△	○	○
	데이터 수집처리 정의서	-	○	-
	하드웨어 설계도	-	-	○
개발 / 구현	프로그램 목록	○	○	○
	테이블 정의서	○	○	△
	핵심 소스코드	○	○	○

※ ○ 필수, △ 선택

| 요구사항 정의서

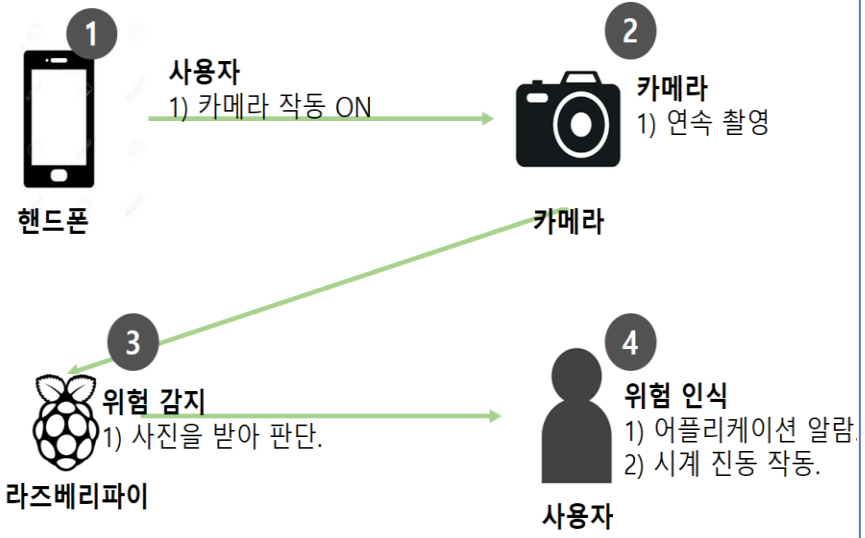
요구사항 ID	요구사항명	기능 ID	기능명	세부사항	예외사항
CAM01	카메라 관리	CAM01_FN01	카메라 자동 작동	보행자가 어플리케이션을 실행하면 후방카메라가 자동으로 작동하여 실시간 영상을 촬영해야 한다.	*어플리케이션 실행 실패 시 카메라 작동 안 됨*
			영상 실시간 전송	후방카메라에서 촬영한 영상을 어플리케이션으로 실시간 전송해야 한다.	*네트워크 불안정 시 전송 지연 발생 가능*
APP01	어플리케이션 관리	APP01-FN01	AI 영상 분석	어플리케이션은 후방카메라로부터 전송된 영상을 실시간으로 분석하여 후방에서 접근하는 차량을 인식해야 한다.	*영상 품질 저하 시 인식률 감소*
			경고 신호 생성	차량이 일정 거리 이내로 접근하면 어플리케이션은 경고 신호를 생성해야 한다.	*차량 접근 속도 예측 오류 발생 가능*
			경고 신호 전송	어플리케이션에서 생성된 경고 신호를 시계로 실시간 전송해야 한다.	*블루투스 연결 실패 시 신호 전송 불가*
WATCH01	시계 관리	WATCH01_FN01	진동 및 소리 경고	시계는 진동과 소리로 보행자에게 경고를 제공해야 한다.	*시계 배터리 부족 시 경고 제공 불가*

| 요구사항 정의서

구분	기능	설명
S/W	영상 분석 및 차량 인식	어플리케이션은 후방카메라로부터 전송된 영상을 실시간으로 분석하여 후방에서 접근하는 차량을 인식해야 한다.
	경고 신호 생성	차량이 일정 거리 이내로 접근하면 어플리케이션은 경고 신호를 생성해야 한다.
	경고 신호 전송	어플리케이션에서 생성된 경고 신호를 시계로 실시간 전송해야 한다.

구분	기능	설명
H/W	후방카메라 작동	보행자가 어플리케이션을 실행하면 후방카메라가 자동으로 작동하여 실시간 영상을 촬영해야 한다.
	후방카메라 영상 전송	후방카메라에서 촬영한 영상을 어플리케이션으로 실시간 전송해야 한다.
	시계 경고 제공	시계는 진동과 소리로 보행자에게 경고를 제공해야 한다.

| 서비스 구성도 - 서비스 시나리오



사용자 위험 감지

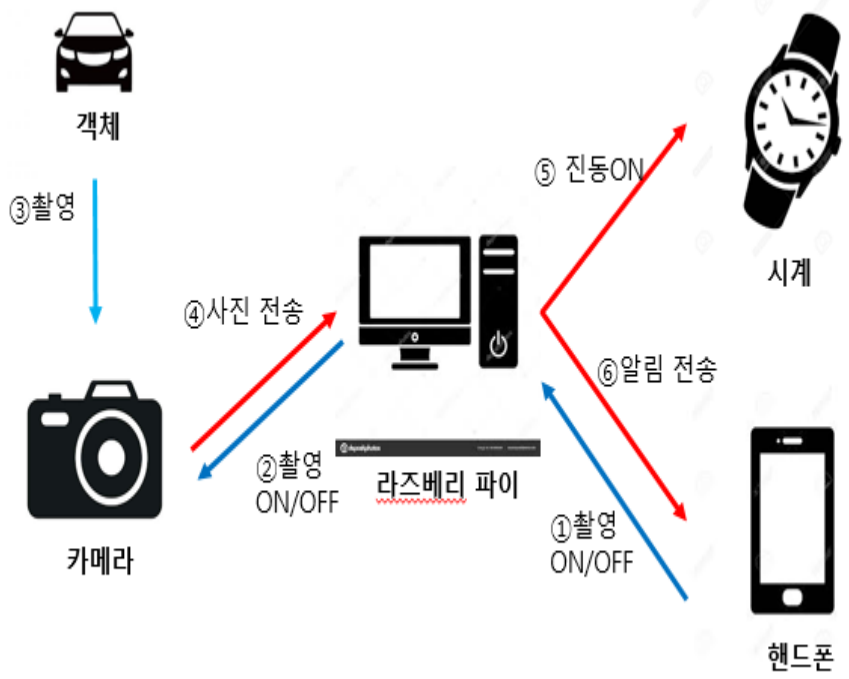
① 사용자가 어플리케이션으로 카메라 기능을 켠다.

② 촬영하며 라즈베리파이에 사진 전송.

③ 위험 상황인지 인식, 위험 상황이라면 시계와 어플리케이션으로 정보 전달.

④ 시계 진동, 어플리케이션에 알람.

| 서비스 흐름도



- 사용자의 사각지대 위험 상황 인식

사용자가 어플리케이션으로 카메라 기능을 키면 라즈베리 파이로 통신 통신이 간다. …… ①, ②

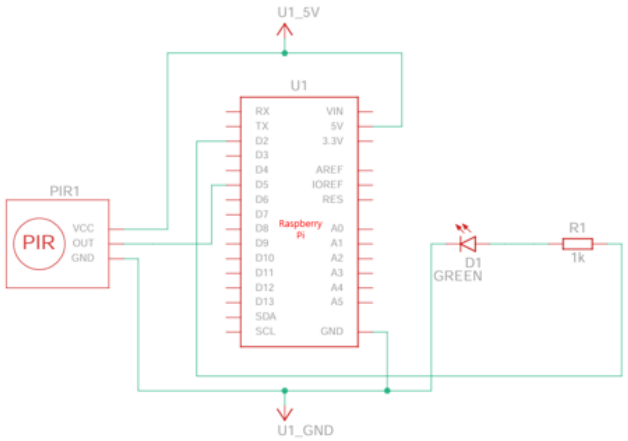
2) 카메라는 초마다 촬영한다. …… ③

3) 촬영된 이미지를 라즈베리 파이에 전달한다. …… ④

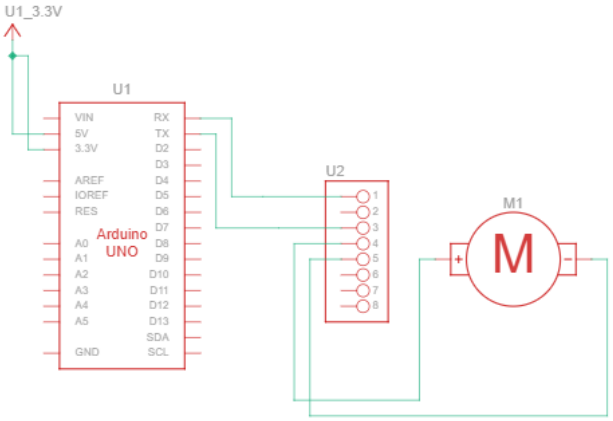
4) 위험 상황인지 모델링 된 자료와 비교한다. 위험 상황이라면 시계와 어플리케이션으로 정보 전달. …… ⑤, ⑥

5) 시계 진동, 핸드폰 알람.

| 하드웨어/센서 구성도

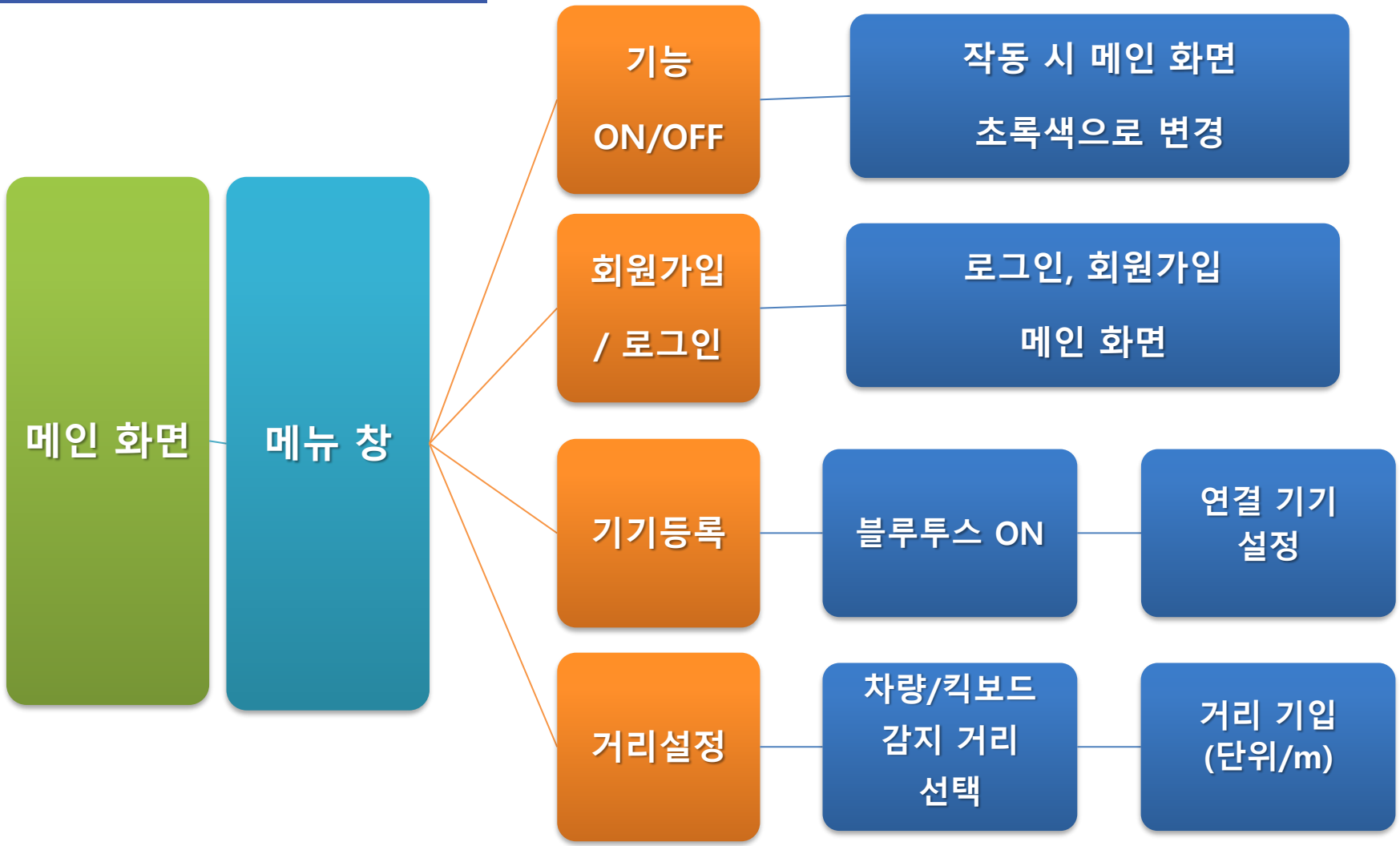


라즈베리 파이		
센서종류	연결 핀	설명
후방카메라	VCC	5V 핀에 연결
	GND	GND 핀에 연결
	DATA_OUT	CSI 포트에 연결
	TRIG	GPIO 핀 (예: GPIO 17)



아두이노		
센서종류	연결 핀	설명
블루투스 모듈	VCC	3.3V 핀에 연결
	GND	GND 핀에 연결
	TXD	디지털 핀 (예: D2)
	RXD	디지털 핀 (예: D3)
진동 모터	VCC	디지털 핀 (예: D4) (PWM 가능 핀)
	GND	GND 핀에 연결

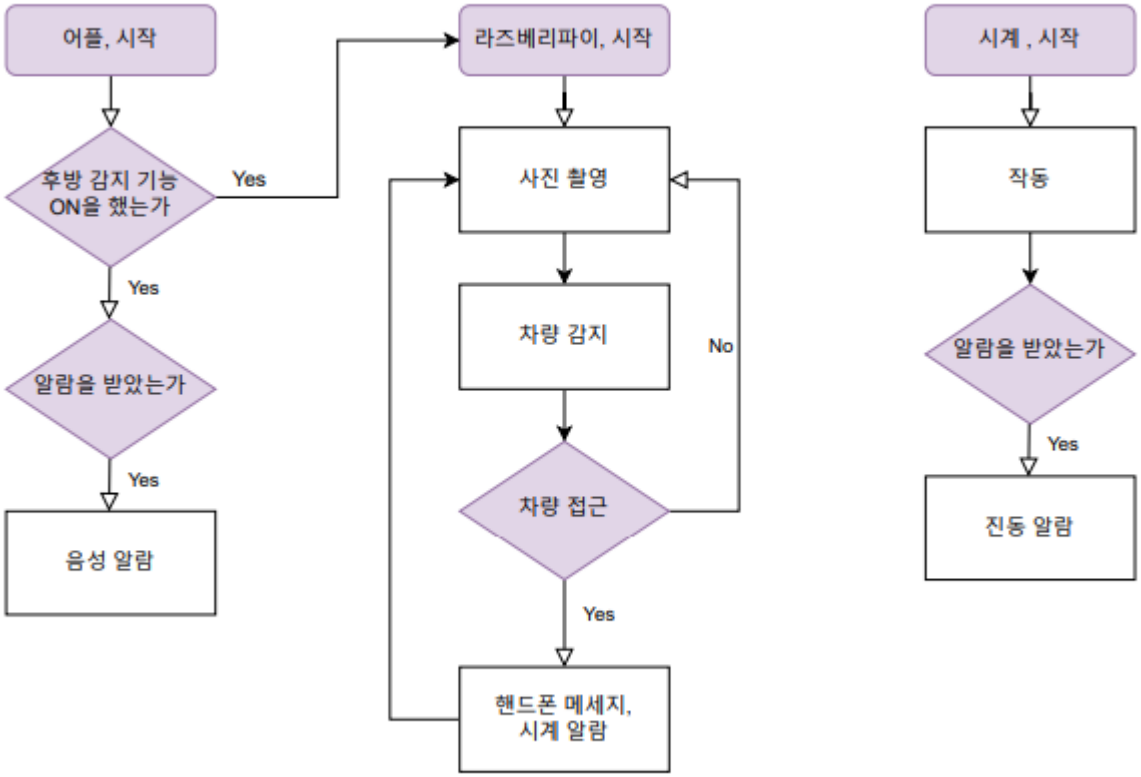
| 메뉴 구성도


메인 화면
메뉴 창
기능 ON/OFF
작동 시 메인 화면 초록색으로 변경
회원가입 / 로그인
로그인, 회원가입 메인 화면
기기등록
블루투스 ON
연결 기기 설정
거리설정
차량/킥보드 감지 거리 선택
거리 기입 (단위/m)

| 기능 처리도(기능 흐름도)

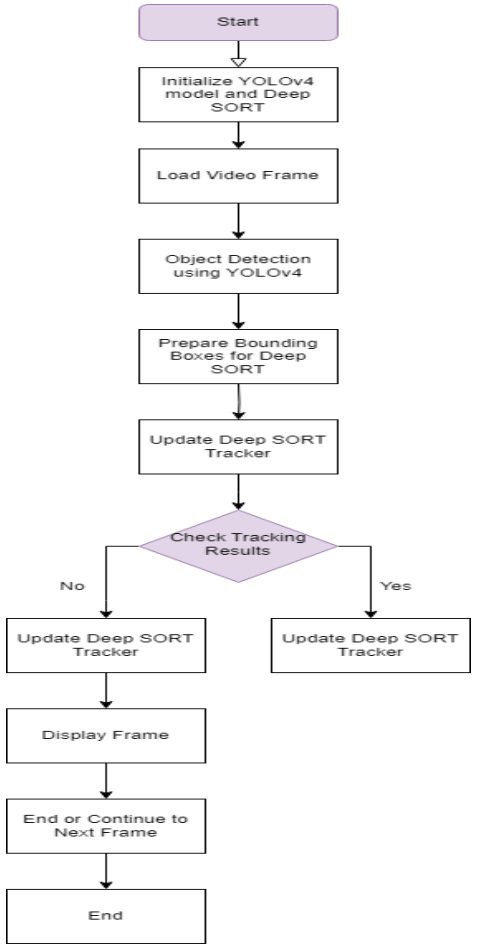
기능 흐름도

어플, 라즈베리 파이, 시계 연계도



| 알고리즘 명세서

객체 연속 추적 알고리즘 (Deep sort)



ㅇ 알고리즘

① 사전 훈련된 YOLOv4 객체 검출 모델과 Deep SORT 추적 알고리즘을 초기화합니다.

② 비디오 파일이나 실시간 스트림에서 프레임을 로드합니다.

③ YOLOv4 모델을 사용하여 현재 프레임에서 객체를 검출합니다. 객체의 바운딩 박스, 클래스, 신뢰도 점수를 반환합니다.

④ YOLOv4에서 검출된 바운딩 박스를 Deep SORT 추적기에 입력하기 위해 준비합니다. 바운딩 박스 좌표와 신뢰도 점수를 추출합니다.

⑤ 준비된 바운딩 박스를 사용하여 Deep SORT 추적기를 업데이트합니다. 추적기의 상태를 최신 프레임에 맞게 업데이트합니다.

⑥ Deep SORT 추적기로부터 객체의 추적 상태를 확인합니다. 추적된 객체가 있는지 확인합니다.

⑦ 추적된 객체가 있는 경우, 해당 객체의 바운딩 박스를 프레임에 그립니다. 각 객체의 ID를 표시합니다.

⑧ 현재 프레임을 화면에 표시합니다. 추적된 객체의 바운딩 박스와 ID가 포함됩니다.

⑨ 더 이상의 프레임이 없으면 종료하고, 그렇지 않으면 다음 프레임을 처리합니다.

| 알고리즘 상세 설명서

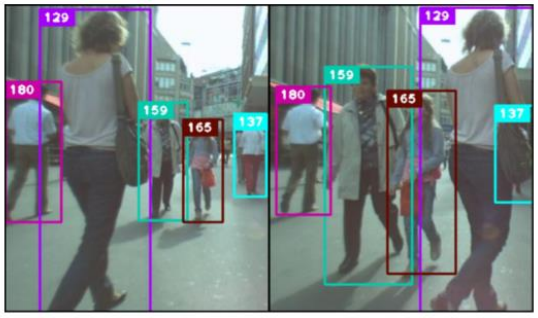
- 객체 추적 – Deep SORT

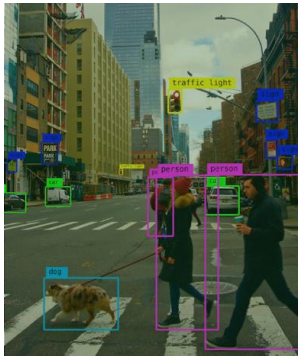
Deep SORT(Deep Simple Online and Realtime Tracking),

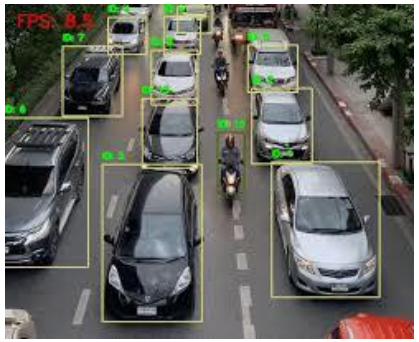
SORT(Simple Online and Realtime Tracking)를 기반으로 한 객체 추적 알고리즘.

SORT는 프레임 간 객체의 위치를 예측하고, 이 위치를 기반으로 객체를 추적하는 간단하고 효율적인 방법. Deep SORT는 여기서 더 나아가, 객체의 시각적 특징을 기반으로 객체를 더 정확하게 추적, 이는 특히 다중 객체 추적에 유리하며, 객체 재식별(Re-Identification, ReID)을 통해 일관된 추적 성능을 유지.

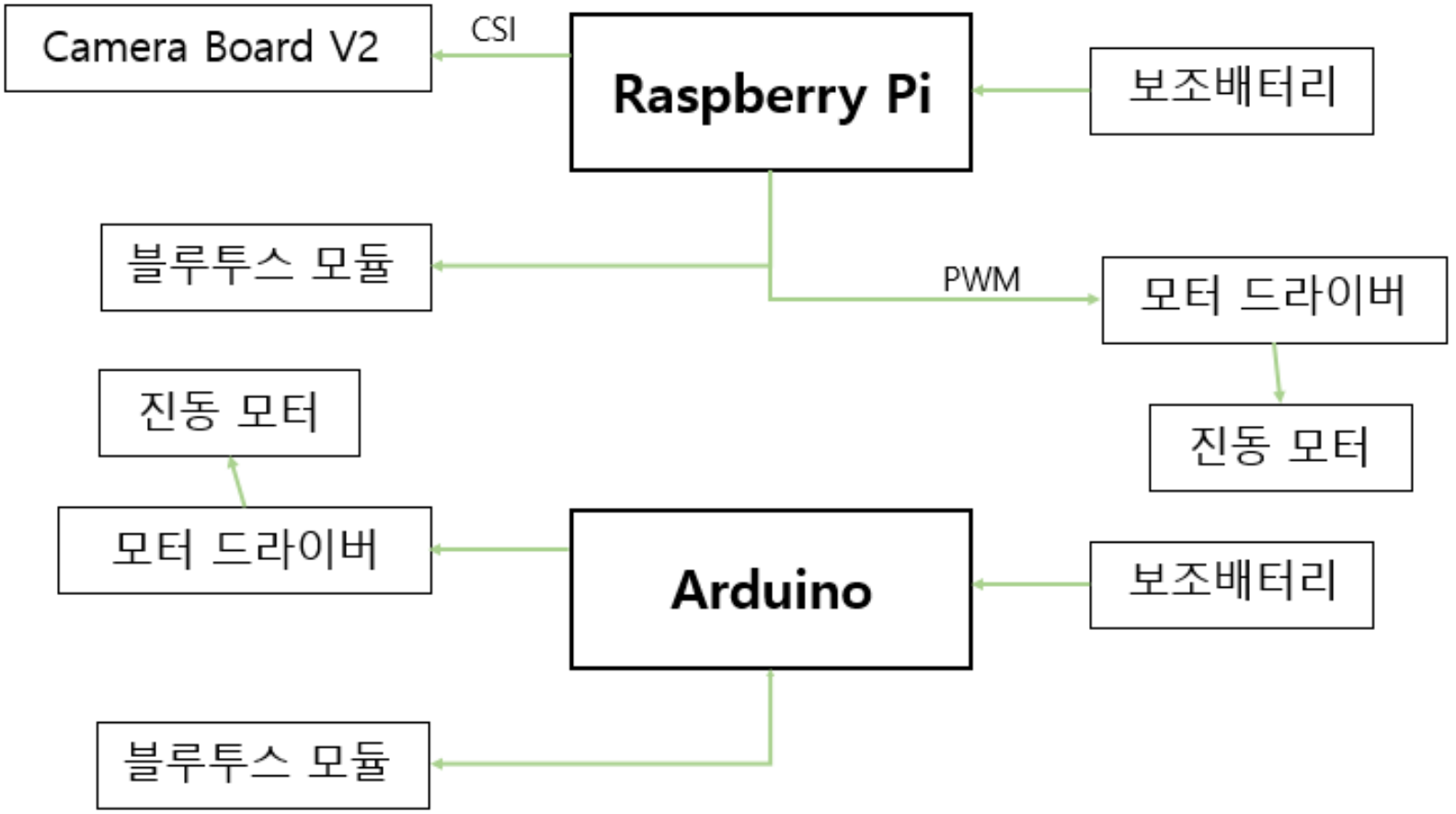
이를 통해, 각 객체에 번호(라벨)을 부여하여 차량, 킥보드 등 여러 탐지 객체가 있어도 후방에서 오는 보행자를 향해 오는 차량만 식별 가능.







| 하드웨어 설계도


Camera Board V2
CSI
Raspberry Pi
보조배터리
블루투스 모듈
PWM
모터 드라이버
진동 모터
진동 모터
모터 드라이버
Arduino
보조배터리
블루투스 모듈

| 프로그램 - 목록

기능 분류	기능번호	기능 명
라즈베리 파이 연동 (LOG)	LOG-01-01	카메라 연결 설정 및 YOLO 모델 설치 및 설정
	LOG-01-02	YOLO 데이터 다운로드 및 준비
	LOG-01-03	라즈베리 파이에 인터넷 연결 설정
객체 검지 (OBJ)	OBJ-01-01	YOLO를 이용한 객체 검지 기능 활성화
	OBJ-01-02	YOLO를 이용한 객체 분류 및 추적 기능
결과 전송 (SND)	SND-01-01	검출된 객체 데이터를 인터넷을 통해 전송 및 저장
	SND-01-02	YOLO 검출 결과를 로컬에 저장
결과 처리 (RES)	RES-01-01	검출된 객체를 화면에 출력
	RES-01-02	특정 객체 감지 시 진동 알림 설정

| 핵심소스코드 / 초기세팅

```python
import cv2
import numpy as np
from picamera import PiCamera
from picamera.array import PiRGBArray
from time import sleep

# YOLO 설정 파일과 가중치 파일 경로
model_configuration = '/home/pi/yolo/yolov3.cfg'
model_weights = '/home/pi/yolo/yolov3.weights'
class_file = '/home/pi/yolo/coco.names'

# 클래스 이름 로드
with open(class_file, 'r') as f:
    classes = [line.strip() for line in f.readlines()]

# 네트워크 설정
net = cv2.dnn.readNetFromDarknet(model_configuration, model_weights)
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# 카메라 초기화
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
raw_capture = PiRGBArray(camera, size=(640, 480))
```

1. 이미지 캡처, 객체 검지, 바운딩 박스 그리기 등의 기능 수행
2. 라즈베리 파이의 카메라 모듈 제어

3. 핵심> YOLO 모델 가져올 경로 (라즈베리 파이에 YOLO파일 저장)

4. 가져온 클래스 이름으로 파일 리스트로 저장

5. Darknet형식 YOLO모델 로드, 레이어를 가져와 저장
Darknet 형식이란, 모델 가중치 파일, 구성 파일, 클래스 이름 파일을 가져와 YOLO모델을 로드하고 사용

6. 라즈베리 파이 카메라 해상도, 프레임 속도, RGB 여부 설정

YOLO에서 학습시킨 데이터 모델을 가져오고,
라즈베리 파이와 연동된 카메라 초기 설정

| 핵심소스코드 / 블롭검지

7. While True를 이용해 카메라가 계속 촬영하게 세팅

```
try:
    while True:
        # 이미지 캡처
        raw_capture.truncate(0)  # 버퍼를 초기화
        camera.capture(raw_capture, format="bgr")
        image = raw_capture.array
```

8. 촬영한 이미지에서 특정 객체를 찾아 박스 처리

```
# 이미지를 블롭(blob) 형식으로 변환하여 네트워크 입력으로 사용
blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
```

학습시킨 데이터가 현재 이미지에 존재하는지 판단을 진행

핵심 > OpenCV에서 촬영한 이미지를 블롭 형식으로 변환하여 YOLO 네트워크에 적합한 형식으로 변환하여 객체 검지를 진행해야 합니다.

이후에는 찾은 객체의 데이터 값을 추출하여 데이터에 따른 모터 제어를 추가 할 예정

Thank you