스마트해상물류
ICT멘토링

SW개발 / HW개발
제작 설계서

[22_HP061] 영상 기반 항만근로자 안전 모니터링 시스템

2022.11.01

팀명 ESL
팀원 이상원, 이현수, 김시현, 천수빈
멘토 현창호

수행 단계별 주요 산출물

※ 아래 표에 맞춰 산출물 작성

※ ○ 필수, △ 선택

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	○	○	○

환경 분석

| 시장/기술 동향 분석

정부의 2025년 항만 작업자 사고 발생 확률 30% 감축 계획에 따른 4차 산업혁명 기술을 통한 안전 플랫폼 구축 예정이 있을 만큼 항만 안전 중요도는 굉장히 높아지고 있다.

| 시장/기술 동향 분석

스마트 해상물류 체계 구축 (4차 산업혁명 기술 접목) :

정부 2025년까지 IoT/AI 등 4차 산업혁명 기술을 선박과 항만 등에 접목시키는 “스마트 해상물류 체계 구축전략” 발표하여 해수부는 2025년까지 스마트 해상물류 기반을 마련하고 2030년까지 스마트 해상물류 실현한다는 계획.

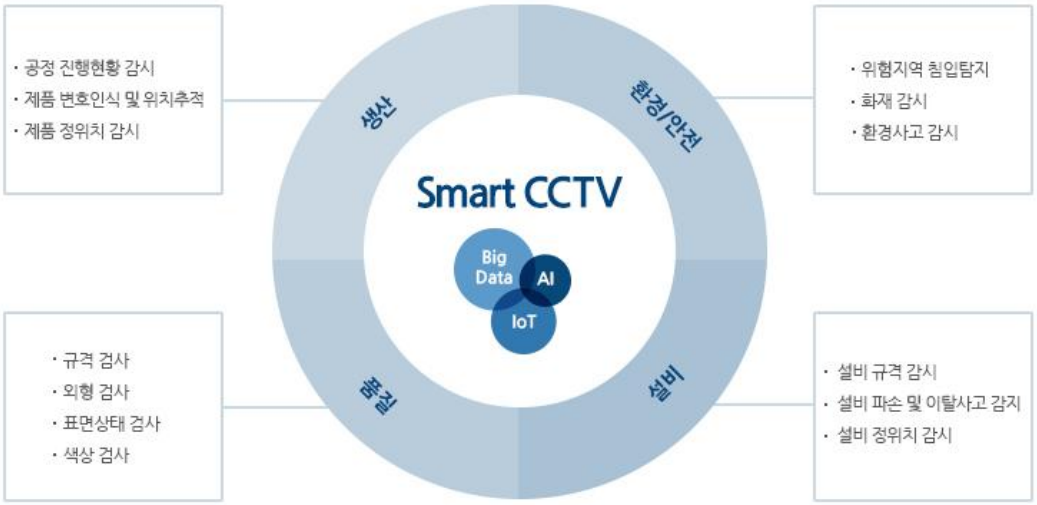
"2025년까지 스마트 해운물류 구축"...확산전략 수립

해운산업팀 | 승인 2021.04.07 16:52 | 댓글 0

비상경제중앙대책본부 회의 장면

| 시장/기술 동향 분석

국내·외 인공지능 기술을 융합한 cctv에 대한 수요가 증가하고 있다. 포스코 ict 의 smart cctv의 경우 cctv에 AI 기술을 적용해 실시간 영상분석 솔루션을 통해 위험지역 침입탐지, 화재감시, 환경사고 감시등의 cctv를 제공하고 있다.



요구사항 분석

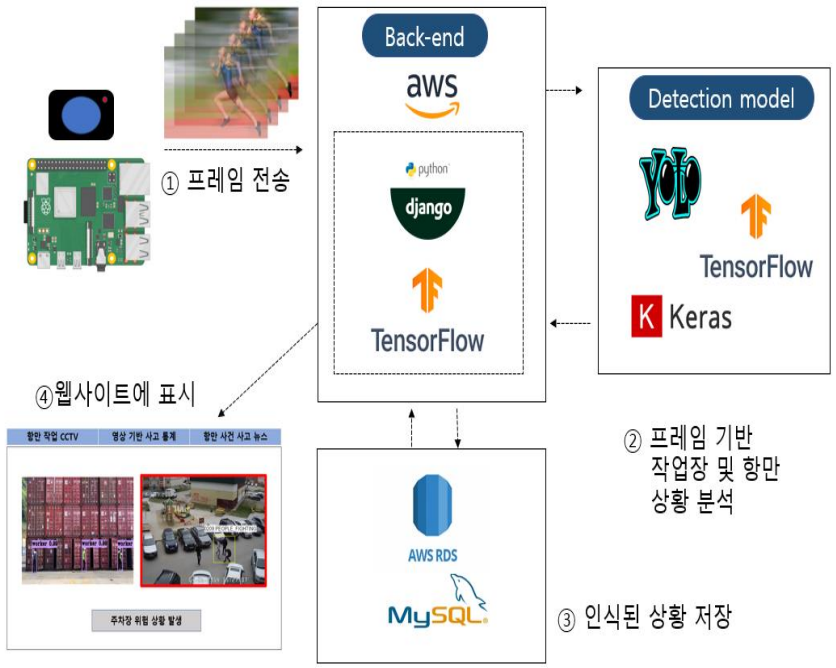
| 요구사항 정의서

구분	기능	설명
S/W	웹 - 기본 페이지	현재 주요 항만의 날씨 상황, 통계 요약본, 연결된 카매라의 대수 및 종류, 지도를 통한 위치 시각화를 확인할 수 있다.
	영상 처리 –화재 상황 탐지	프레임 실시간 분석 중 항만에서 화재 사고 발생시 바로 상황을 알림하도록 한다.
	영상 처리 –안전 수칙 준수 탐지	CCTV에서 전송된 프레임을 실시간으로 분석하여 안전수칙을 잘 준수하는지 살피고 N명 이상 근무지에서 N명 미만 근무 작업모 미착용에 대하여 실시간으로 상황을 감지 알림을 한다.
	영상 처리 – 실신 감지	CCTV에서 전송된 프레임을 실시간으로 분석하여 작업현장에서 작업자가 실신을 하는 사고가 발생할 경우 바로 알림을 줄 수 있도록 한다.
	웹 – 영상 결과 제공 및 녹화 및 캡쳐	로컬 CCTV에서 클라우드 서버로 프레임을 전송하여 항만에서의 사고 발생 영상을 스트리밍 할 수 있도록 한다. 또한 영상을 녹화 및 캡쳐 기능을 통해 사고 내용을 기록할 수 있도록 한다.

구분	기능	설명
S/W	웹 – 정책 설정	각 카메라 마다 필요한 정책 설정(화재 탐지, 안전 수칙, 실신 탐지)을 할 수 있게 한다.
	웹 - 사고 통계치 정보 제공	각 카메라 마다 필요한 정책 설정(화재 탐지, 안전 수칙, 실신 탐지)을 할 수 있게 한다.
	웹 - 항만 사건 사고 뉴스 제공	항만의 사건 사고에 대한 뉴스를 서비스 받음으로써 사건 사고 예방을 위한 교육 등이 빠르게 이뤄질 수 있는 계기를 마련할 수 있다.
	앱 – 사고 정보 제공	항만에 사고가 발생했을 시 앱을 통해서 사고 정보를 받을 수 있도록 한다.
H/W	로컬에서 서버로 영상 전송	보드를 통해서 촬영된 영상을 웹서버로 프레임 단위로 보내준다.

아키텍쳐 설계

| 서비스 구성도 - 서비스 시나리오

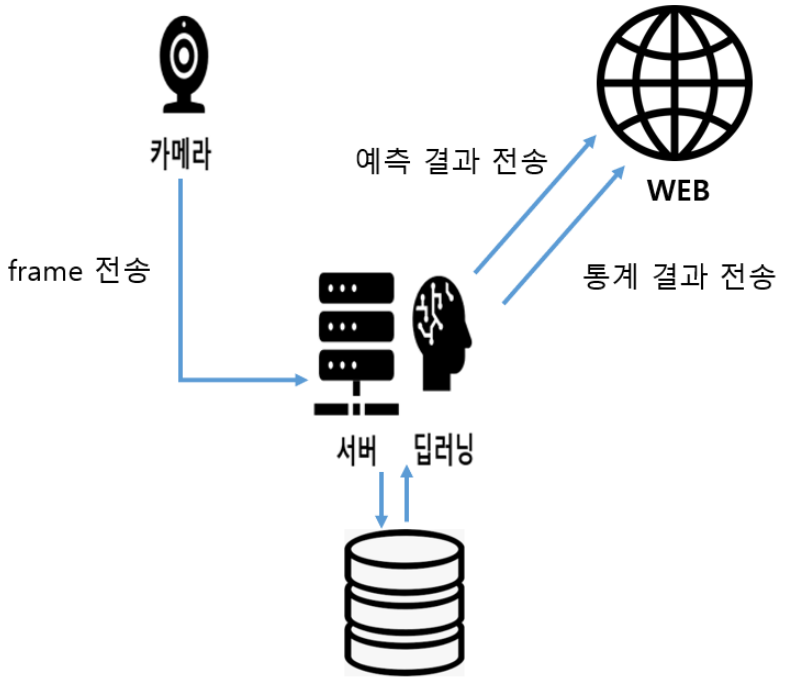


-영상 획득 및 분류(Camera)
- 라즈베리파이의 카메라를 통해 프레임을 계속해서 서버로 보내준다.

-Back-end
- 라즈베리 파이 카메라로 부터 받아온 프레임을 바로 디텍션 모델로 넘겨서 상황 판단을 할 수 있도록 한다.
- 사용자에게 cctv 상황을 파악할 수 있도록 한다.
- 만약 cctv가 위치한 공간에서 문제가 발생했을 경우 웹페이지에 알림을 줄 수 있도록 한다.
- 사고 상황에 대한 통계 정보등을 제공할 수 있도록 한다.

-Detection model
- 라즈베리 파이로 부터 전달받은 프레임을 yolo알고리즘을 통해 현재 안전 수칙이 안지켜지는 상황, 화재등을 연산한다.
- 프레임들을 일정 프레임 만큼 컨테이너등에 저장하여 일정 길이의 프레임이 어떤 상황을 가지고 있는 지를 계산하여 쓰러짐과 같은 이상행동을 연산한다.

| 서비스 흐름도



<frame 서버 전송>

1) 라즈베리파이 카메라를 통해 frame을 찍어 서버로 frame 데이터를 전송한다.
2) 서버에서는 frame을 담아서 연산한다.

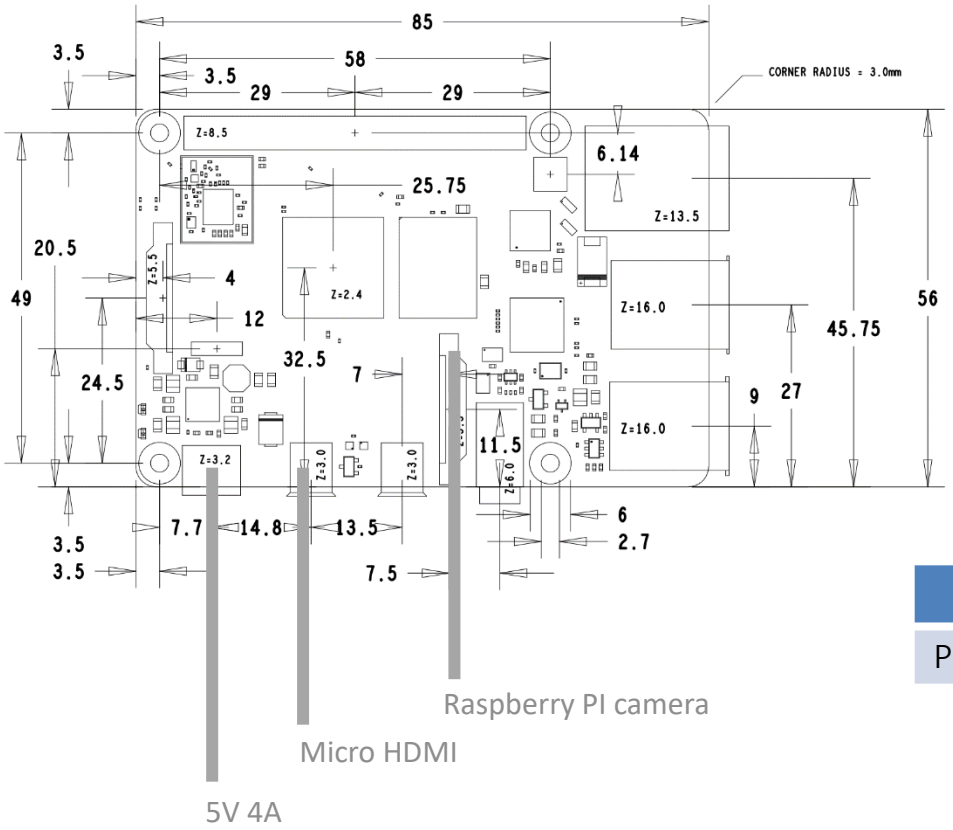
<Detectiom 모델로 예측 제공>

1) 서버에 전달된 frame을 yolo로 연산한다.
2) 일정 프레임이 쌓일때까지 queue 저장소에 데이터를 저장한다.
3) 일정 수 만큼 queue에 쌓이면 해당 queue를 이상행동 연산 알고리즘을 통해 계산한다.
4) 해당 예측된 결과를 WEB에 알려준다.

<통계 결과 제공>

1) 예측된 사고 결과를 DB에 저장한다.
2) WEB에서 통계 결과를 요청할 경우 DB에서 정보를 꺼내서 WEB에 시각화 해준다.

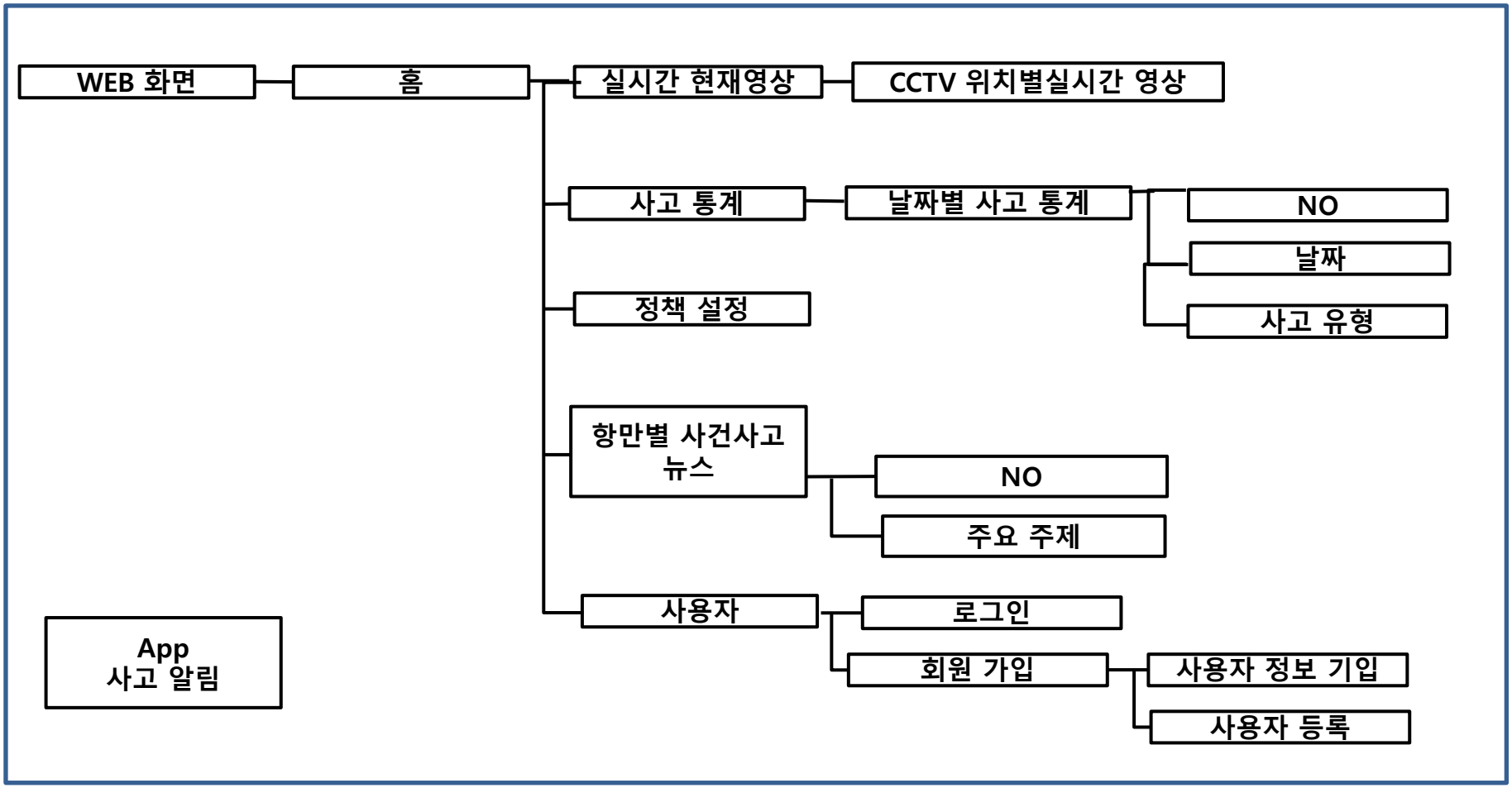
| 하드웨어/센서 구성도



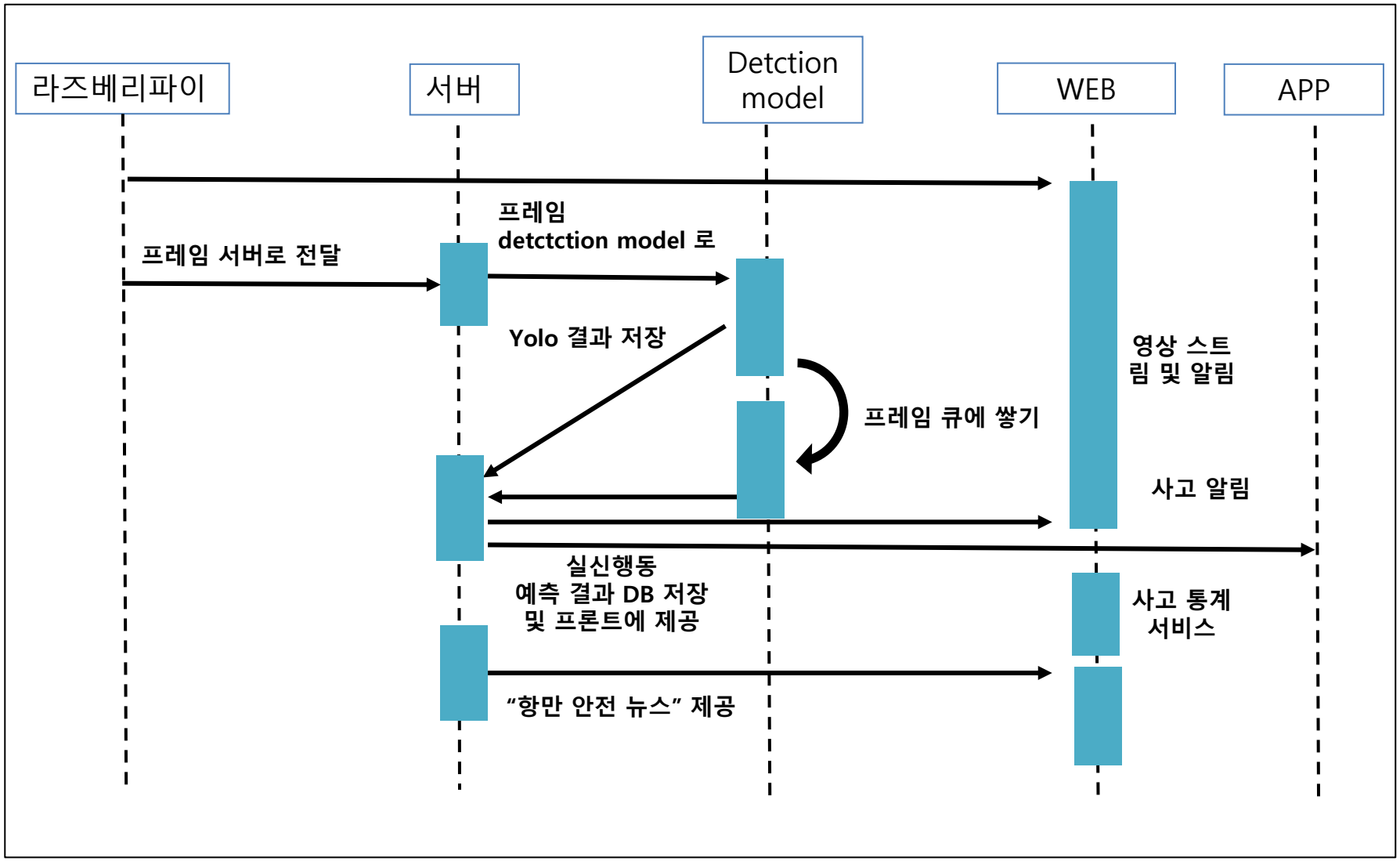
연결	연결 단자	설명
Pi camera	CSI Port	카메라를 위한 연결

기능 설계

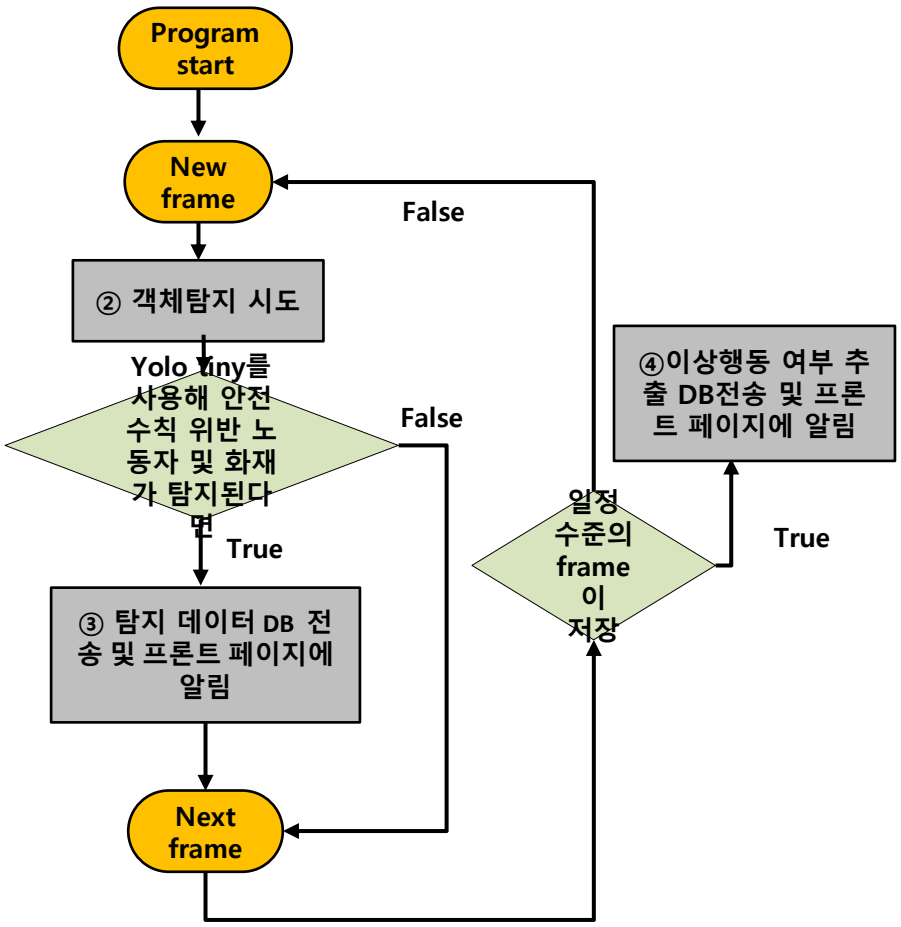
| 메뉴 구성도


WEB 화면
홈
실시간 현재영상
CCTV 위치별실시간 영상
사고 통계
날짜별 사고 통계
NO
날짜
사고 유형
정책 설정
항만별 사건사고 뉴스
NO
주요 주제
사용자
로그인
회원 가입
사용자 정보 기입
사용자 등록
App
사고 알림

| 기능 처리도(기능 흐름도)


라즈베리파이
서버
Detction model
WEB
APP
프레임 서버로 전달
프레임 detctction model 로
Yolo 결과 저장
프레임 큐에 쌓기
영상 스트림 및 알림
사고 알림
실신행동 예측 결과 DB 저장 및 프론트에 제공
사고 통계 서비스
"항만 안전 뉴스" 제공

| 알고리즘 명세서



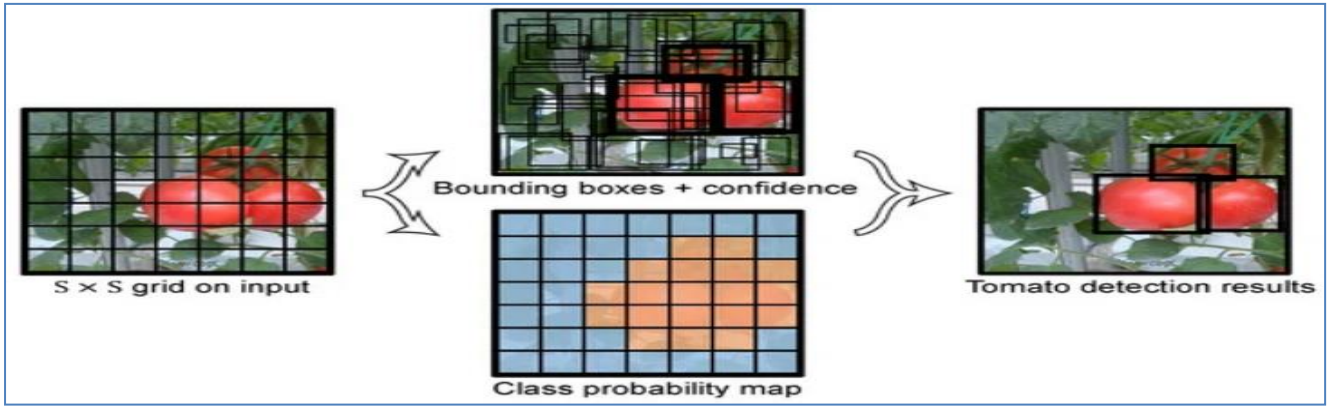
O 인공지능 탐지 알고리즘 시나리오

1. 라즈베리파이 카메라로부터 프레임을 받는다.
2. 프레임을 서버로 전송한다.
3. 서버에서 yolo 로 어떠한 객체가 탐지되는지 확인을 한다.
4. 일정 프레임 만큼 frame을 queue 저장소에 쌓는다.
5. queue 저장소를 이상행동 추출 알고리즘에 넣어서 어떤 행동 상황인지를 파악한다.
6. 위의 과정을 계속해서 프레임을 받아 수행한다.

o 작업자 및 화재 탐지등 – YOLO Tiny 알고리즘

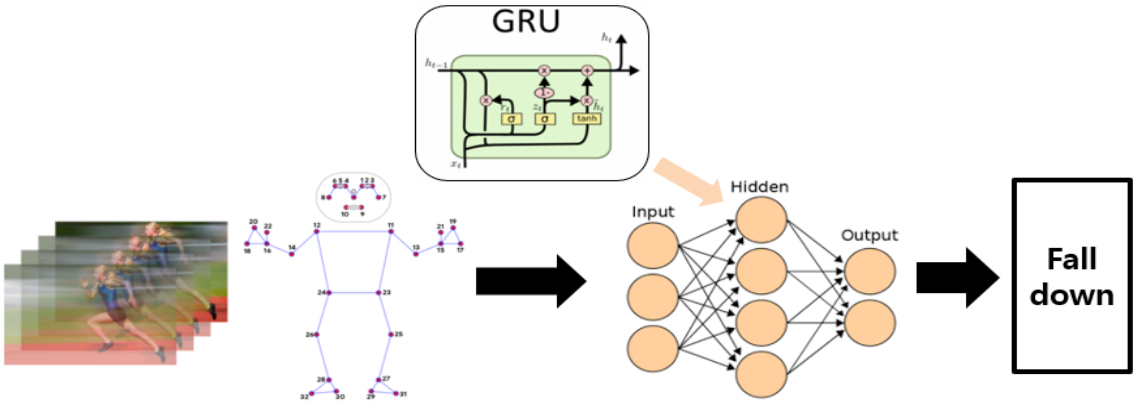
1. 입력이미지를 S x S의 Grid cells로 나눈다.
2. 미리 설정된 개수의 boundary boxes를 예측한다.
3. Conditional class probabilities를 예측한다.
4. 임계값 이상의 박스만 표기한다.

Yolo Tiny 버전의 경우 Yolo보다 정확도가 낮지만 속도 개선의 효과를 볼 수 있어 Yolo Tiny 버전을 사용했다.

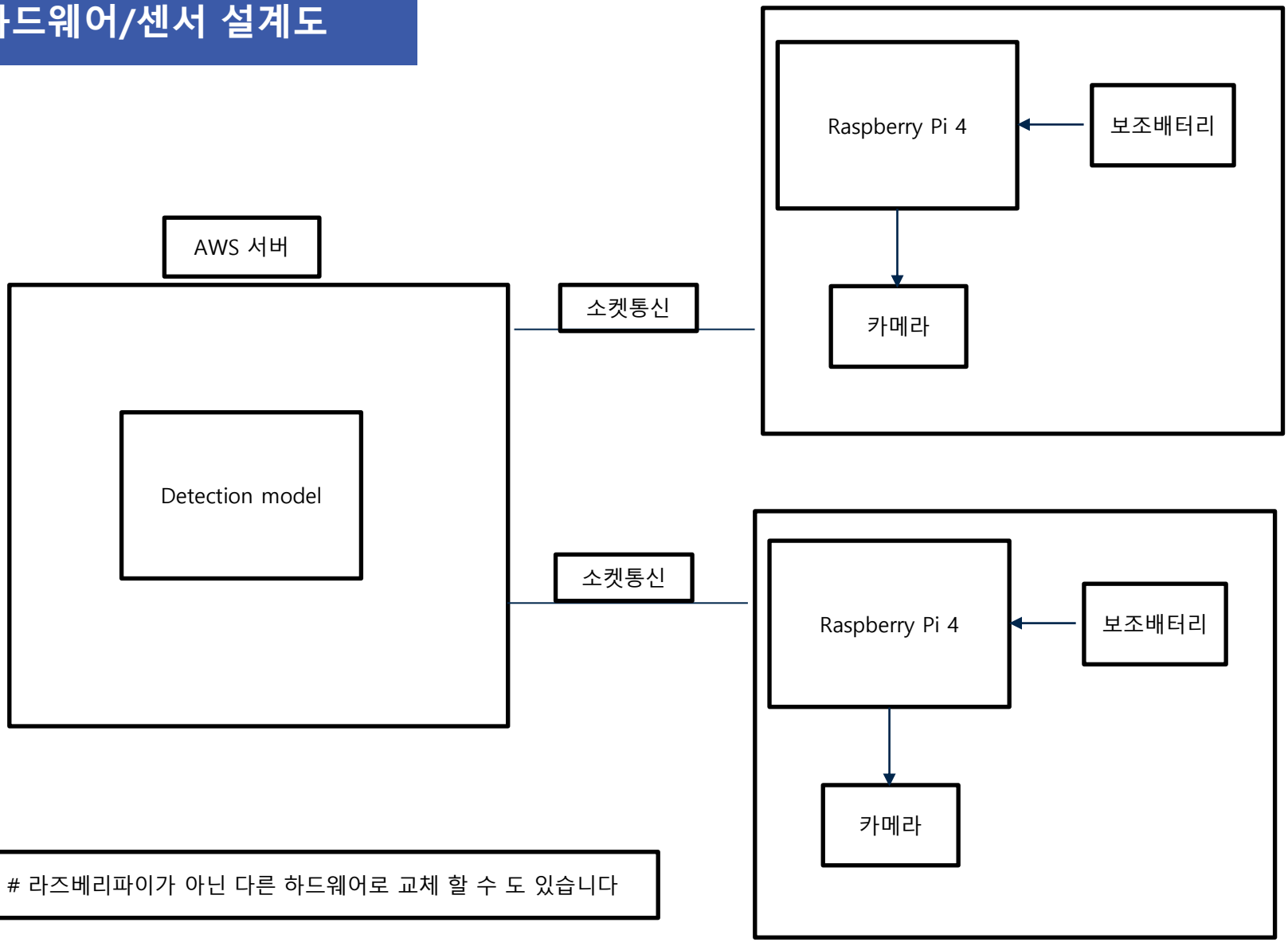


o 이상행동 탐지– Action Recognition

1. 입력 프레임을 Mediapipe 를 통해 pose estimation을 하여 skeleton 좌표를 추출한다.
2. Skeleton 좌표 정보들을 저장소에 100frame 만큼 쌓는다.
3. GRU 알고리즘을 통해 학습된 모델에 저장소에 쌓인 프레임을 넣고 자세를 추정한다.



| 하드웨어/센서 설계도



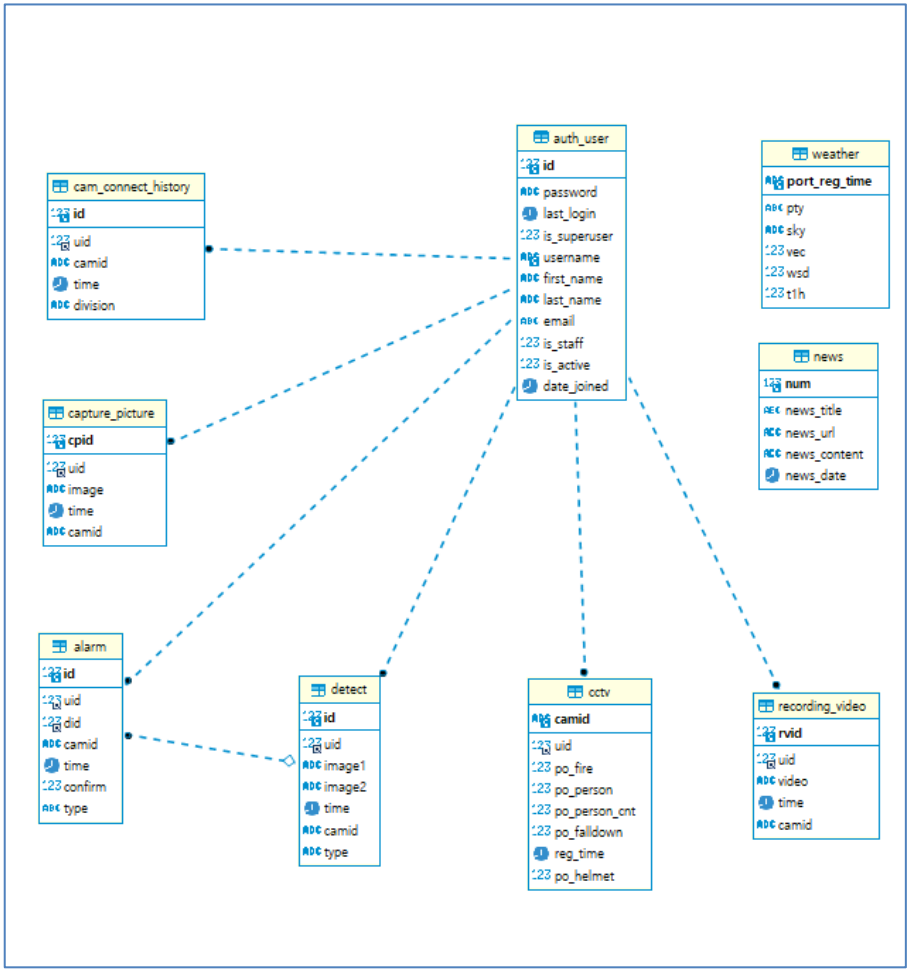
라즈베리파이가 아닌 다른 하드웨어로 교체 할 수 도 있습니다

개발 / 구현

| 프로그램 - 목록

기능 분류	기능번호	기능 명
LOG	LOG-01-01	회원가입
	LOG-01-02	로그인
MYP	MYP-01-01	사용자 저장 이미지 캡처 조회
	MYP-01-02	사용자 녹화 영상 조회
	MYP-02-01	사고상황 감지 기록 조회
	MYP-02-02	안전 수칙 준수 감지 기록 조회
	MYP-02-03	이상행동 감지 기록 조회
AIS	AIS-01-01	실시간 스트리밍 보기
	AIS-02-01	사고상황 감지
	AIS-02-02	안전 수칙 준수 감지
	AIS-02-03	이상행동 감지
NEW	NEW-01	항만 관련 뉴스 제공
STA	STA-01	사고상황 감지 날짜별, 시간별 통계 제공

| 테이블 정의서 - ERD



해당 프로젝트를 제작하면서 사용한 엔티티 들의 모음

auth_user : 유저정보 저장

cam_connect_history : cctv 연결 기록

Cctv : cctv 정보 저장

capture_picture : 사용자 저장 사진 저장

recording_video : 사용자 저장 동영상 저장

detect : 탐지 저장

alarm : 알람 저장

weather : 날씨 정보 저장

news : 뉴스 관련 정보 저장

| 핵심소스코드(1)

```
class Client:
    def __init__(self, username, client_socket, rpid):
        self.cnt = 0
        self.username = username
        self.connections = {}
        self.threads = {}


        self.cnt+=1
        conn = Frame(client_socket, username,rpid)
        self.connections[rpid]=conn
        print("========================1")
        live_detect_thread = Thread(target=self.thread_func, args=(rpid,conn,))#Thread(target=conn.detect_live)
        print("========================2")
        live_detect_thread.start()
        print("========================3")
        self.threads[rpid] = live_detect_thread
        print("========================4")
        cam_connetct_history = CamConnectHistory()
        user = AuthUser.objects.get(username=self.username)
        ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
        cam_connetct_history.uid = user
        cam_connetct_history.camid = rpid
        cam_connetct_history.time = timestamp
        cam_connetct_history.division = 'CONNECT'
        cam_connetct_history.save()


    def thread_func(self, rpid, conn):
        conn.detect_live()
        self.disconnect_socket(rpid)

    def add_client(self, client_socket, rpid):
        print('add client')
        self.cnt += 1
        conn = Frame(client_socket, self.username, rpid)
        self.connections[rpid] = conn
        live_detect_thread = Thread(target=self.thread_func, args=(rpid,conn,))#Thread(target=conn.detect_live)
        live_detect_thread.start()
        self.threads[rpid] = live_detect_thread

        cam_connetct_history = CamConnectHistory()
        user = AuthUser.objects.get(username=self.username)
        ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
        cam_connetct_history.uid = user
```

본 프로젝트에서 웹서버에 사용자의 복수개의 라즈베리파이가 연결되어 서비스를 이용할 수 있음. 이때 라즈베리파이는 소켓 통신을 통해 연결됨.

복수의 사용자에 대해 각각의 사용자가 복수의 라즈베리파이를 연결해 cctv 서비스를 이용하기 위해 라즈베리파이를 연결하는 객체를 싱글톤으로 만들어 관리함.

| 핵심소스코드(2)

```
def detect_live(self):
    while True:
        if self.check == True:
            return
        # 설정한 데이터의 크기보다 버퍼에 저장된 데이터의 크기가 작은 경우
        while len(self.data_buffer) < self.data_size:
            # 데이터 수신
            try:
                self.data_buffer += self.client_socket.recv(4096)
            except:
                return
            '''
            temp = self.client_socket.recv(4096)
            if temp=='disconnect':
                print(1)
                self.disconnet()
            self.data_buffer += temp
            '''

        self.client_socket.sendall("10".encode())
        # 버퍼의 저장된 데이터 분할
        packed_data_size = self.data_buffer[:self.data_size]
        self.data_buffer = self.data_buffer[self.data_size:]
        # struct.unpack : 변환된 바이트 객체를 원래의 데이터로 변환
        frame_size = struct.unpack(">L", packed_data_size)[0]
        # 프레임 데이터의 크기보다 버퍼에 저장된 데이터의 크기가 작은 경우
        while len(self.data_buffer) < frame_size:
            # 데이터 수신
            try:
                self.data_buffer += self.client_socket.recv(4096)
            except:
                return
        # 프레임 데이터 분할
        frame_data = self.data_buffer[:frame_size]
        self.data_buffer = self.data_buffer[frame_size:]
        # print("수신 프레임 크기 : {} bytes".format(frame_size))
        # loads : 직렬화된 데이터를 역직렬화
        # 역직렬화(de-serialization) : 직렬화된 파일이나 바이트 객체를 원래의 데이터로 복원하는 것
        frame = pickle.loads(frame_data)

        # imdecode : 이미지(프레임) 디코딩
        frame = cv2.imdecode(frame, cv2.IMREAD_COLOR)
```

연결된 라즈베리파이로부터 소켓통신(TCP)을 통해 이미지 프레임을 가져오는 코드임.

버퍼에 저장된 데이터를 분할하고 변환된 바이트 객체를 원래의 데이터로 변환 후 직렬화된 바이트 객체를 원래의 데이터로 복원함. 이렇게 얻어진 이미 프레임 한장을 가지고 AI 인공지능 모델과 YOLO모델에 적용함. 또 실시간 스트리밍을 위해서 이미지 프레임을 반환하는 함수임.

| 핵심소스코드(3)

```
class FireDetector():
    def __init__(self, username):
        self.username = username
        self.net = cv2.dnn.readNet("cctv/data/custom-yolov4-tiny-detector_best.weights",
                                   "cctv/data/yolov3_custom.cfg")
        self.classes = ["fire"]
        # YOLO 네트워크 불러오기
        self.layer_names = self.net.getLayerNames()

        self.output_layers = [self.layer_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]
        # 클래스의 갯수만큼 랜덤 RGB 배열을 생성
        self.colors = np.random.uniform(0, 255, size=(len(self.classes), 3))

        # 탐지 시간
        self.detect_fire_time = time.time()
```

FireDetector 객체에서 화재 감지 모델을 불러와 각 프레임에 대해 화재감지를 진행한다.

| 핵심소스코드(4)

```python
def detect_fire(self, frame, size, score_threshold, nms_threshold, camid):
    if time.time()-self.detect_fire_time<10:
        return;
    copy_frame = frame.copy()
    # 이미지의 높이, 너비, 채널 받아오기
    height, width, channels = frame.shape

    # 네트워크에 넣기 위한 전처리
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (size, size), (0, 0, 0), True, crop=False)

    # 전처리된 blob 네트워크에 입력
    self.net.setInput(blob)

    # 결과 받아오기
    outs = self.net.forward(self.output_layers)

    # 각각의 데이터를 저장할 빈 리스트
    class_ids = []
    confidences = []
    boxes = []
    # printq
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if not (class_id == 0 or class_id == 15 or class_id == 16):
                continue
            if confidence > 0.1:
                # 탐지된 객체의 너비, 높이 및 중앙 좌표값 찾기
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # 객체의 사각형 테두리 중 좌상단 좌표값 찾기
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    # Non Maximum Suppression (겹쳐있는 박스 중 confidence 가 가장 높은 박스를 선택)
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=score_threshold, nms_threshold=nms_threshold)
```

FireDetector 객체에서 화재 감지 모델을 불러와 각 프레임에 대해 화재감지를 진행한다.

| 핵심소스코드(5)

```
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        class_name = self.classes[class_ids[i]]
        label = f"{class_name} {confidences[i]:.2f}"
        color = self.colors[class_ids[i]]

        # 사각형 테두리 그리기 및 텍스트 쓰기
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.rectangle(frame, (x - 1, y), (x + len(class_name) * 13 + 65, y - 25), color, -1)
        cv2.putText(frame, label, (x, y - 8), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 0), 2)

        if class_name=='fire':
            detectModel = Detect()

            ret1, frame1 = cv2.imencode('.jpg', frame)
            ret2, frame2 = cv2.imencode('.jpg', copy_frame)
            user = AuthUser.objects.get(username=self.username)
            ts = time.time()
            timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

            detectModel.uid = user
            file1 = ContentFile(frame1)
            file2 = ContentFile(frame2)
            file1.name = 'fire_'+timestamp + '_1' + '.jpg'
            file2.name = 'fire_'+timestamp + '_2' + '.jpg'
            detectModel.image1 = file1
            detectModel.image2 = file2
            detectModel.time = timestamp
            detectModel.camid = camid
            detectModel.type='FIRE'
            detectModel.save()

            alarm = Alarm()
            alarm.uid=user
            alarm.did=detectModel
            alarm.camid=camid
            alarm.time=timestamp
            alarm.confirm=0
            alarm.type='FIRE'
            alarm.save()

            self.detect_fire_time=time.time()
            print('detect fire')
            break

return frame
```

FireDetector 객체에서 화재 감지 모델을 불러와 각 프레임에 대해 화재감지를 진행한다.

| 핵심소스코드(6)

```
class HelmetDetector():
    def __init__(self, username):
        self.username = username
        self.net = cv2.dnn.readNet("cctv/data/yolov4-tiny.weights",
                                   "cctv/data/yolov4-tiny.cfg")
        self.net_helmet = cv2.dnn.readNet("cctv/data/yolov3-obj_2400.weights",
                                          "cctv/data/yolov3-obj.cfg")
        self.classes = ["person"]
        self.classes_helmet = ["helmet"]

        # YOLO 네트워크 불러오기
        self.layer_names = self.net.getLayerNames()
        self.layer_names_helmet = self.net_helmet.getLayerNames()

        self.output_layers = [self.layer_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]
        self.output_layers_helmet = [self.layer_names_helmet[i[0] - 1] for i in
                                     self.net_helmet.getUnconnectedOutLayers()]

        # 클래스의 갯수만큼 랜덤 RGB 배열을 생성
        self.colors = np.random.uniform(0, 255, size=(len(self.classes), 3))
        self.colors_helmet = np.random.uniform(0, 255, size=(len(self.classes_helmet), 3))

        # 탐지 시간
        self.detect_helmet_time = time.time()
```

HelmetDetector 객체에서 안전모 감지 모델을 불러와 각 프레임에 대해 안전모 감지를 진행한다.

| 핵심소스코드(7)

```python
def detect_helmet(self, frame, size, score_threshold, nms_threshold, camid):
    if time.time() - self.detect_helmet_time < 10:
        return;
    copy_frame = frame.copy()
    draw_frame = frame.copy()
    height, width, channels = frame.shape

    blob = cv2.dnn.blobFromImage(frame, 0.00392, (size, size), (0, 0, 0), True, crop=False)
    self.net.setInput(blob)
    outs = self.net.forward(self.output_layers)
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if not (class_id == 0 or class_id == 15 or class_id == 16):
                continue
            if confidence > 0.1:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=score_threshold, nms_threshold=nms_threshold)
```

HelmetDetector 객체에서 안전모 감지 모델을 불러와 각 프레임에 대해 안전모 감지를 진행한다.

| 핵심소스코드(8)

```
detect_person_cnt = 0
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        class_name = self.classes[class_ids[i]]
        label = f"{class_name} {confidences[i]:.2f}"
        color = self.colors[class_ids[i]]
        cv2.rectangle(draw_frame, (x, y), (x + w, y + h), color, 2)
        cv2.rectangle(draw_frame, (x - 1, y), (x + len(class_name) * 13 + 65, y - 25), color, -1)
        cv2.putText(draw_frame, label, (x, y - 8), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 0), 2)
        if class_name == 'person':
            detect_person_cnt += 1

blob = cv2.dnn.blobFromImage(frame, 0.00392, (size, size), (0, 0, 0), True, crop=False)
self.net_helmet.setInput(blob)
outs = self.net_helmet.forward(self.output_layers_helmet)
class_ids = []
confidences = []
boxes = []
```

HelmetDetector 객체에서 안전모 감지 모델을 불러와 각 프레임에 대해 안전모 감지를 진행한다.

| 핵심소스코드(9)

```
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if not (class_id == 0 or class_id == 15 or class_id == 16):
                continue
            if confidence > 0.1:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=score_threshold, nms_threshold=nms_threshold)
    detect_helmet_cnt = 0
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            class_name = self.classes_helmet[class_ids[i]]
            label = f"{class_name} {confidences[i]:.2f}"
            color = self.colors_helmet[class_ids[i]]
            cv2.rectangle(draw_frame, (x, y), (x + w, y + h), color, 2)
            cv2.rectangle(draw_frame, (x - 1, y), (x + len(class_name) * 13 + 65, y - 25), color, -1)
            cv2.putText(draw_frame, label, (x, y - 8), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 0), 2)
            if class_name == 'helmet':
                detect_helmet_cnt += 1
```

HelmetDetector 객체에서 안전모 감지 모델을 불러와 각 프레임에 대해 안전모 감지를 진행한다.

| 핵심소스코드(10)

```
if time.time() - self.detect_helmet_time > 10 and detect_person_cnt > 0 and detect_person_cnt > detect_helmet_cnt:
    detectModel = Detect()

    ret1, frame1 = cv2.imencode('.jpg', draw_frame)
    ret2, frame2 = cv2.imencode('.jpg', copy_frame)
    user = AuthUser.objects.get(username=self.username)
    ts = time.time()
    timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

    detectModel.uid = user
    file1 = ContentFile(frame1)
    file2 = ContentFile(frame2)
    file1.name = 'helmet_' + timestamp + '_1' + '.jpg'
    file2.name = 'helmet_' + timestamp + '_2' + '.jpg'
    detectModel.image1 = file1
    detectModel.image2 = file2
    detectModel.time = timestamp
    detectModel.camid = camid
    detectModel.type = 'HELMET'
    detectModel.save()

    alarm = Alarm()
    alarm.uid = user
    alarm.did = detectModel
    alarm.camid = camid
    alarm.time = timestamp
    alarm.confirm = 0
    alarm.type = 'HELMET'
    alarm.save()

    self.detect_helmet_time = time.time()
    print('detect HELMET'+str(detect_person_cnt)+' '+str(detect_helmet_cnt))

return frame
```

HelmetDetector 객체에서 안전모 감지 모델을 불러와 각 프레임에 대해 안전모 감지를 진행한다.

| 핵심소스코드(11)

```python
class PersonDetector():
    def __init__(self, username):
        self.username = username
        self.net = cv2.dnn.readNet("cctv/data/yolov4-tiny.weights",
                                   "cctv/data/yolov4-tiny.cfg")
        self.classes = ["person"]
        # YOLO 네트워크 불러오기
        self.layer_names = self.net.getLayerNames()

        self.output_layers = [self.layer_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]
        # 클래스의 갯수만큼 랜덤 RGB 배열을 생성
        self.colors = np.random.uniform(0, 255, size=(len(self.classes), 3))

        # 탐지 시간
        self.detect_person_time = time.time()

    def detect_person(self, frame, size, score_threshold, nms_threshold, camid, policy_person_cnt):
        if time.time()-self.detect_person_time<10:
            return;
        copy_frame = frame.copy()
        # 이미지의 높이, 너비, 채널 받아오기
        height, width, channels = frame.shape

        # 네트워크에 넣기 위한 전처리
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (size, size), (0, 0, 0), True, crop=False)

        # 전처리된 blob 네트워크에 입력
        self.net.setInput(blob)

        # 결과 받아오기
        outs = self.net.forward(self.output_layers)

        # 각각의 데이터를 저장할 빈 리스트
        class_ids = []
        confidences = []
        boxes = []
```

PersonDetector 객체에서 사람 감지 모델을 불러와 각 프레임에 대해 사람 감지를 진행한다.

| 핵심소스코드(12)

```
# printq
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if not (class_id == 0 or class_id == 15 or class_id == 16):
            continue
        if confidence > 0.1:
            # 탐지된 객체의 너비, 높이 및 중앙 좌표값 찾기
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            # 객체의 사각형 테두리 중 좌상단 좌표값 찾기
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

# Non Maximum Suppression (겹쳐있는 박스 중 confidence 가 가장 높은 박스를 선택)
indexes = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=score_threshold, nms_threshold=nms_threshold)
```

PersonDetector 객체에서 사람 감지 모델을 불러와 각 프레임에 대해 사람 감지를 진행한다.

| 핵심소스코드(13)

```
detect_person_cnt=0
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        class_name = self.classes[class_ids[i]]
        label = f"{class_name} {confidences[i]:.2f}"
        color = self.colors[class_ids[i]]

        # 사각형 테두리 그리기 및 텍스트 쓰기
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.rectangle(frame, (x - 1, y), (x + len(class_name) * 13 + 65, y - 25), color, -1)
        cv2.putText(frame, label, (x, y - 8), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 0), 2)

        if class_name=='person':
            detect_person_cnt+=1
```

PersonDetector 객체에서 사람 감지 모델을 불러와 각 프레임에 대해 사람 감지를 진행한다.

| 핵심소스코드(14)

```
if time.time()-self.detect_person_time>10 and detect_person_cnt>0 and detect_person_cnt<policy_person_cnt:
    detectModel = Detect()

    ret1, frame1 = cv2.imencode('.jpg', frame)
    ret2, frame2 = cv2.imencode('.jpg', copy_frame)
    user = AuthUser.objects.get(username=self.username)
    ts = time.time()
    timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

    detectModel.uid = user
    file1 = ContentFile(frame1)
    file2 = ContentFile(frame2)
    file1.name = 'person_' + timestamp + '_1' + '.jpg'
    file2.name = 'person_' + timestamp + '_2' + '.jpg'
    detectModel.image1 = file1
    detectModel.image2 = file2
    detectModel.time = timestamp
    detectModel.camid = camid
    detectModel.type = 'PERSON'
    detectModel.save()

    alarm = Alarm()
    alarm.uid = user
    alarm.did = detectModel
    alarm.camid = camid
    alarm.time = timestamp
    alarm.confirm = 0
    alarm.type = 'PERSON'
    alarm.save()

    self.detect_person_time = time.time()
    print('detect person')


return frame
```

PersonDetector 객체에서 사람 감지 모델을 불러와 각 프레임에 대해 사람 감지를 진행한다.

| 핵심소스코드(15)

```
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose

class FallDownDetector():
    def __init__(self, username):
        self.username = username

        self.model = tf.keras.models.load_model('cctv/data/human_skeleton_act-99-0.92.hdf5')
        self.pose = mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)

        self.sequence = []

        self.threshold = 0.8

        # 탐지 시간
        self.detect_falldown_time = time.time()

    def detect_falldown(self, frame, camid):
        copy_frame = frame.copy()
        # Make detections
        image, results = self.mediapipe_detection(frame, self.pose)
        # print(results)

        # 2. Prediction logic
        keypoints = self.extract_keypoints(results)
        #         sequence.insert(0,keypoints)
        #         sequence = sequence[:30]
        self.sequence.append(keypoints)
        self.sequence = self.sequence[-100:]
```

FallDownDetector 클래스에서 실신 감지에 필요한 모델을 불러와 프레임에 대해 실신을 감지한다.

| 핵심소스코드(16)

```
text = 'no'
if len(self.sequence) == 100:
    res = self.model.predict(np.expand_dims(self.sequence, axis=0))[0]
    # text= LABEL_INT_DICT[np.argmax(res)]
    text = str(np.argmax(res))
    if text=='1' and time.time()-self.detect_falldown_time>10:
        detectModel = Detect()

        ret1, frame1 = cv2.imencode('.jpg', frame)
        ret2, frame2 = cv2.imencode('.jpg', copy_frame)
        user = AuthUser.objects.get(username=self.username)
        ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

        detectModel.uid = user
        file1 = ContentFile(frame1)
        file2 = ContentFile(frame2)
        file1.name = 'falldown_' + timestamp + '_1' + '.jpg'
        file2.name = 'falldown_' + timestamp + '_2' + '.jpg'
        detectModel.image1 = file1
        detectModel.image2 = file2
        detectModel.time = timestamp
        detectModel.camid = camid
        detectModel.type = 'PERSON'
        detectModel.save()

        alarm = Alarm()
        alarm.uid = user
        alarm.did = detectModel
        alarm.camid = camid
        alarm.time = timestamp
        alarm.confirm = 0
        alarm.type = 'FALLDOWN'
        alarm.save()

        self.detect_falldown_time = time.time()

print(text)
```

FallDownDetector 클래스에서 실신 감지에 필요한 모델을 불러와 프레임에 대해 실신을 감지한다.

| 핵심소스코드(17)

```
    # 3. Viz logic
    self.draw_styled_landmarks(image, results)

    # cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
    cv2.putText(image, ' val =  ' + (text), (3, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    return image



def extract_keypoints(self, results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in
                     results.pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33 * 4)
    return pose

def mediapipe_detection(self, image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False  # Image is no longer writeable

    results = model.process(image)  # Make prediction
    image.flags.writeable = True  # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)  # COLOR COVERSION RGB 2 BGR

    return image, results

def draw_styled_landmarks(self, image, results):
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80, 22, 10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80, 44, 121), thickness=2, circle_radius=2)
                              )
```

FallDownDetector 클래스에서 실신 감지에 필요한 모델을 불러와 프레임에 대해 실신을 감지한다.

| 참조- 개발 환경 및 설명

구분		상세내용
S/W 개발환경	OS	Linux(라즈베리파이, 클라우드 환경), Windows10(로컬 개발)
	개발환경(IDE)	Pycharm , Visual Studio Code
	개발도구	MySQL, MySQL Workbench(8.0)
	개발언어	Python , javascript
	기타사항	
H/W 구성장비	디바이스	라즈베리파이
	센서	라즈베리파이 카메라
	통신	소켓통신
	언어	Python
	기타사항	새로운 장비를 통해 CCTV 역할을 시도할 수 있음
프로젝트 관리환경	형상관리	Git lab , Git hub
	의사소통관리	카카오톡, Zoom, Discord
	기타사항	

| 참조-S/W 기능 실사 사진

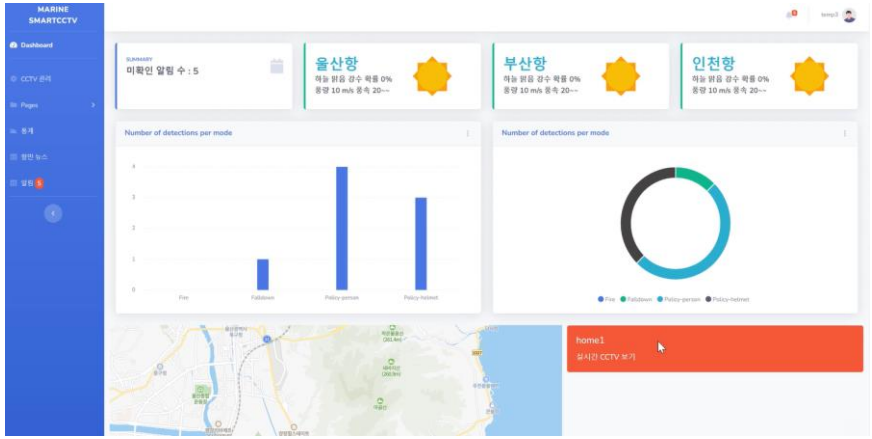


그림 1. 웹 페이지

웹의 전반적인 페이지
지도를 통해 cctv의 위치를 확인할 수 있으며 , 옆에 바를 통해 현재 CCTV 위치의 상태(파란색 이상 없음, 회색 미연결, 빨간색 사고)를 알 수 있다.

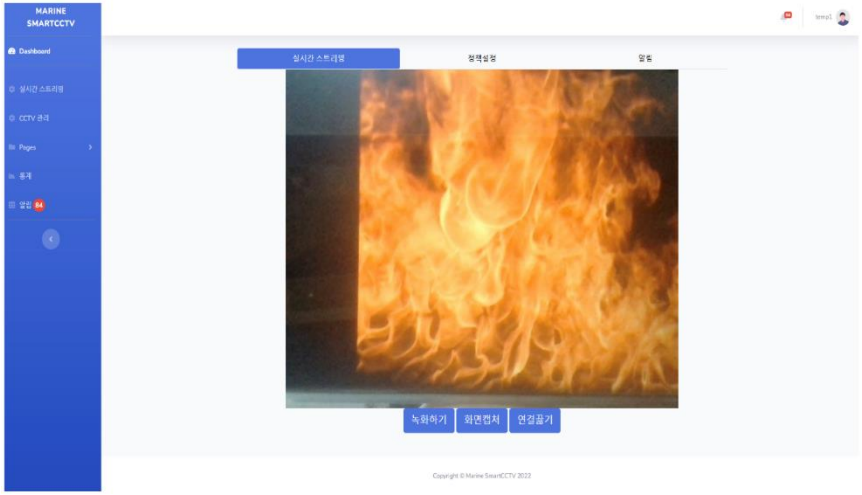


그림 2. 화재 탐지

화재를 탐지하는 것을 테스트 한 장면이다.

| 참조-S/W 기능 실사 사진

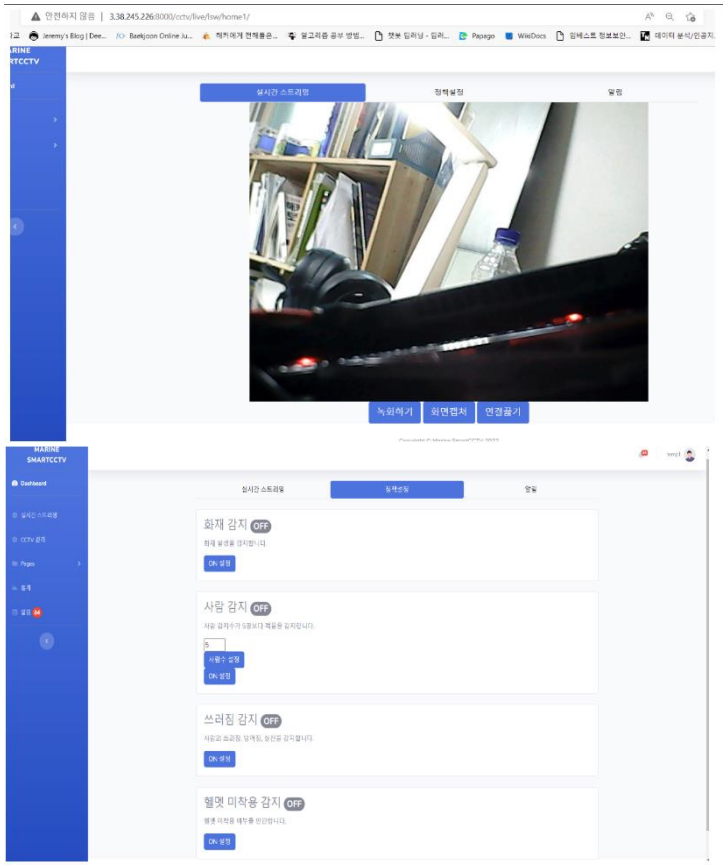


그림 3.

클라우드 서버를 통해 로컬에서의 영상을 받는 사진, 영상 송출 페이지에서 녹화 이미지 캡쳐등을 할 수 있다.

그림 4.

웹에서 정책 설정을 통해 CCTV에서 어떠한 사고를 탐지할 지 결정할 수 있다.사진

| 참조-S/W 기능 실사 사진

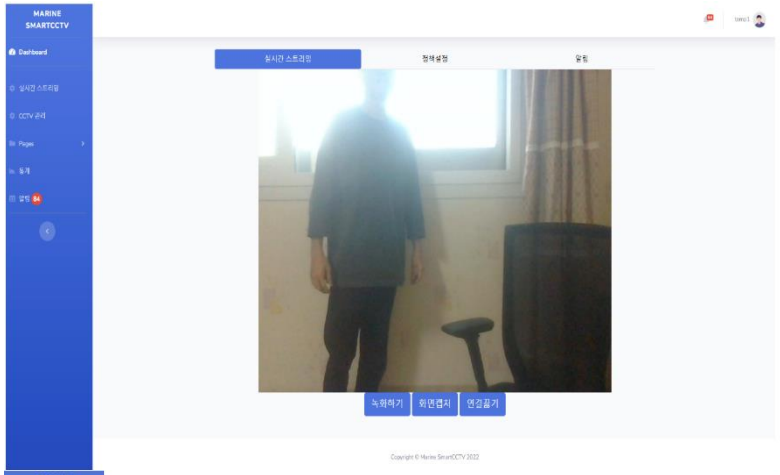


그림 5. 안전수칙 위반 탐지

작업자가 안전모 미착용 및 n명 이상 근무지에 n명 미만 출입등을 탐지하는 사진이다.

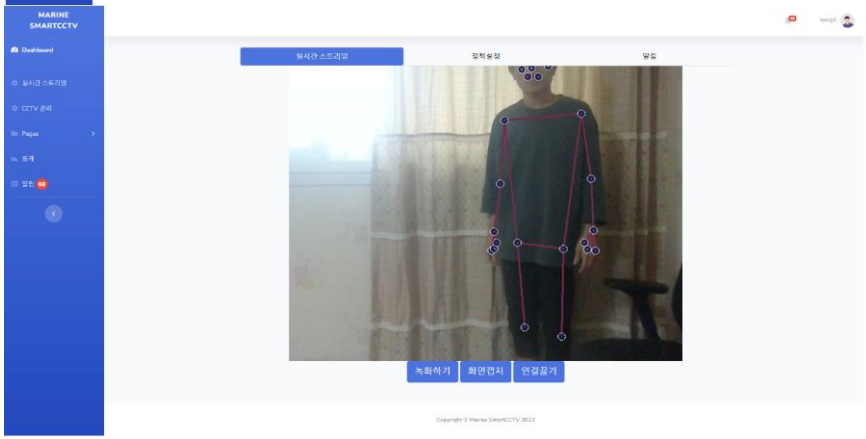


그림 6. 실신 탐지

작업자가 쓰러졌는 지 여부를 pose estimatio을 통해 탐지를 해낸다.

| 참조-S/W 기능 실사 사진

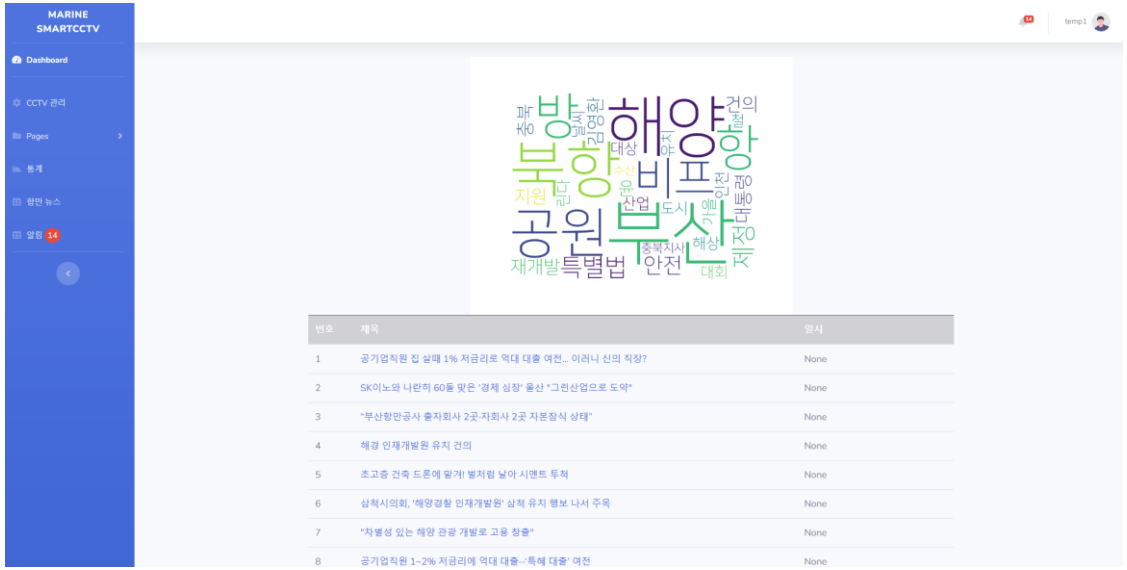


그림 7. 뉴스 크롤링

항만 정보 및 항만 사고 관련 뉴스의 정보를 크롤링 한 모습이다.

```
[뉴스 제목]
['여수광양항만공사, 광양항 해양산업클러스터 입주
, 체선완화 방안 이행합의서 체결', '부산항만공사
치장 활용해 물류난 해소…1년간 14만TEU 처리', '부
공사, 초중생용 비대면 해운·항만 교육 키트 제작'
 재정비 추진', '부산해수청, 항만건설공사 하도급

[뉴스 링크]
['https://n.news.naver.com/mnews/article/421/00
?sid=100', 'https://n.news.naver.com/mnews/arti
0005137998?sid=102', 'https://n.news.naver.com/
ticle/421/0006294407?sid=102', 'https://n.news.
m/mnews/article/001/0013391978?sid=102', 'https

[뉴스 내용]
['[\n\n\n\n\n제4차 광양항 해양산업클러스터 입주
26일 밝혔다. 2020년 국내 최초 해양산업클러스터
```

| 참조-S/W 기능 실사 사진

그림 8. 통계

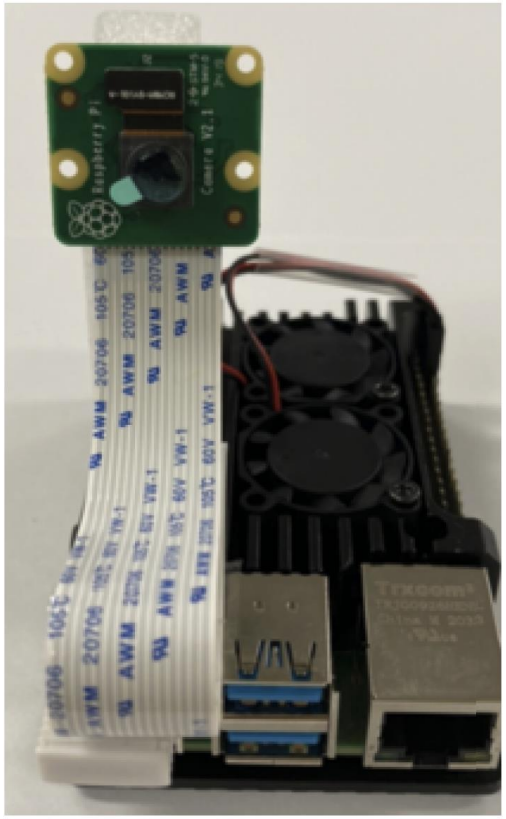
사고상황 감지 날짜별, 시간별 통계 제공.

| 참조-S/W 기능 실사 사진

그림 9. 앱 알림 기능

앱을 통해 사고 내용을 확인할 수 있다.

| 참조- H/W 기능 실사사진



```
전송 프레임 크기 : 59711 bytes
b'10'
전송 프레임 크기 : 59834 bytes
b'10'
전송 프레임 크기 : 59529 bytes
b'10'
전송 프레임 크기 : 59439 bytes
b'10'
전송 프레임 크기 : 59172 bytes
b'10'
전송 프레임 크기 : 59105 bytes
b'10'
전송 프레임 크기 : 59602 bytes
b'10'
전송 프레임 크기 : 59751 bytes
b'10'
전송 프레임 크기 : 59577 bytes
b'10'
전송 프레임 크기 : 59534 bytes
b'10'
전송 프레임 크기 : 59648 bytes
b'10'
전송 프레임 크기 : 59497 bytes
b'10'
전송 프레임 크기 : 59551 bytes
```

그림 10. 영상 프레임 전달

영상 프레임을 전달할 라즈베리 파이와 라즈베리 파이에서 서버로 영상을 전달하는 장면이다.

Thank you