SW개발/HW제작 설계서

프로젝트 명 : Smart해상물류를 구현할 최신 Edge Computing을 이용한 자율 운행 자율접안 시뮬레이터 개발 (스마트 항만 자율접안 및 자율운행 및 Edge Computing 시스템 구축) (스마트 항만 물류 자율접안 Edge Computing 시스템구축)

수행 단계별 주요 산출물

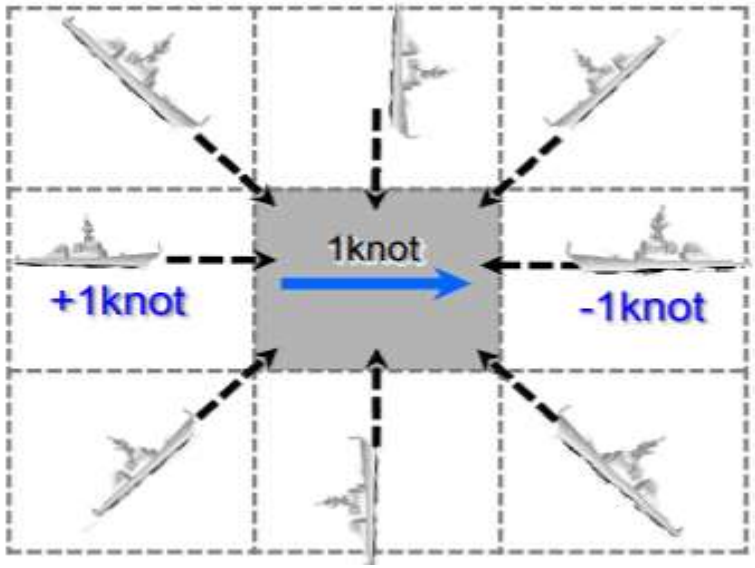
일반

응용 소프트웨어

응용 하드웨어

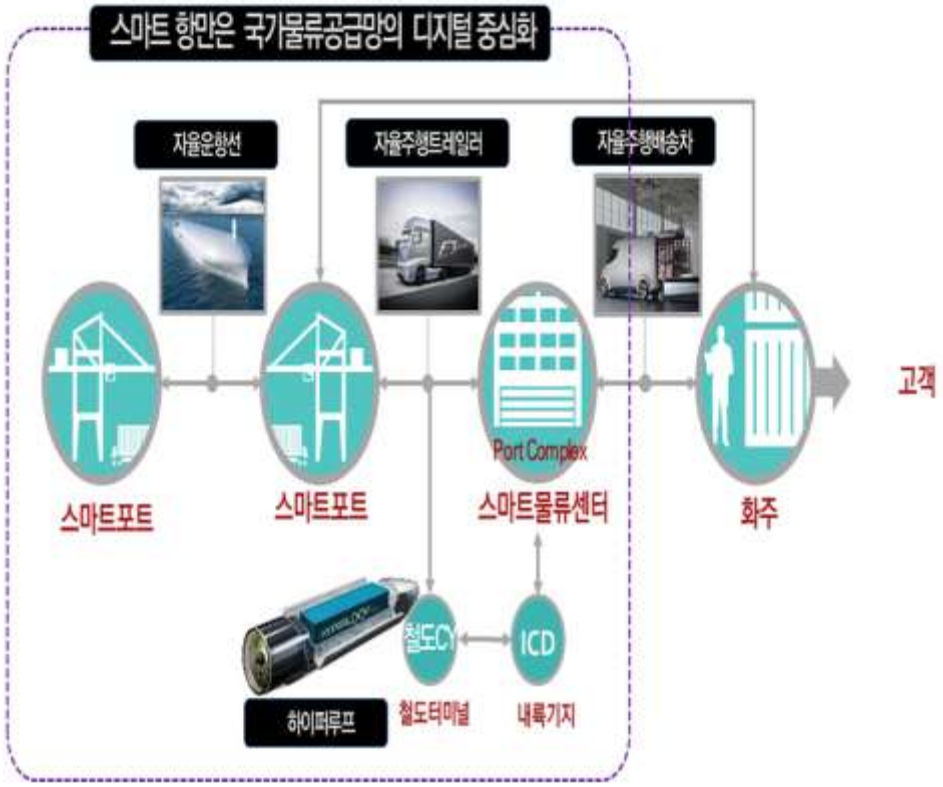
환경분석	시장/ 기술 환경 분석서	○	○	○
	인터뷰 결과서	--	--	--
요구사항 분석	요구사항 정의서	○	○	○
	유즈 케이스 정의서	○	○	○
아키텍쳐 설계	서비스 구성도	○	○	○
	서비스 흐름도	○	○	○
	UI/UX	○	○	--
기능 설계	알고리즘 명세서	○	○	○
	하드웨어 설계도	--	--	○
	기능처리도(기능 흐름도)	○	○	○
개발 /구현	핵심 소스 코드	--	○	○

시장/기술 동향 분석



- 제어·계측공학기술의 발전은 모든 산업분야에서의 자동화 수준을 향상시키는데 지대한 공헌을 하였고, 산업전반에 걸쳐 인간의 직접적 관여를 줄여 편이성을 극 대화하는데 기여하고 있다. 자율 주행 자동차 등이 그 일례이며, 운행 안전성 확보를 위한 기술개발도 지속적 으로 추진되고 있다. 이러한 추세는 육상환경에 국한되지 않고, 해상에서 운용되는 선박 등에 적용하고자 하는 노력도 오래전부터 추진되어 왔고, 관련기술도 상당한 수준에 이르렀다 할 수 있다. 이미 오래전에 무인선 운용 기술이 개발되었던 점을 고려하면 기술의 현장적용 면 에서는 육상보다 훨씬 앞섰다고 할 수 있다. 그러나 해상 에서의 자율운항이 가능한 무인화선박운용기술은 소형 선 및 특수 목적선(군사용)에 한정되어 있고, 상용선박 에 대해서는 파일로트의 피로도를 경감시켜 근무환경을 개선하기 위한 지원기술 정도로 활용되어져 왔다. 끊임 없는 기술개발노력의 결과로 상용선박에도 무인화기술 을 적용하여 운용효율성을 극대화 하려는 시도가 현실 로 다가오고 있다.

시장/기술 동향 분석



-자율 운항선을 통해 스마트 항만의 국가 물류 공급망의 디지털 중심화의 시작

-항만과 항만 외부 물류자원들과의 연결성이 강화되는 커넥티드 스마트 항만은 국가 물류공급망의 디지털 중심으로서 국가 물류비 절감에 기여하게 될 것임.

-위와 같은 커넥티드 스마트 항만을 구축하기 위해 우선 적으로 항만 전체적으로 자율화가능한 환경이 구축되어야 될 것.

-항만을 중심으로 하는 스마트 국가물류공급망의 변화는 먼 미래가 아니라 곧 다가오게 될 현실임.

시장/ 기술 동향 분석

DATA ANALYSIS

선박 자율 주행 기능 및 위치 서비스 필요

선박(클라이언트)과 항만(서버)와의 지속적인 데이터교류 중요

복잡성을 띈 대용량 데이터 저장및 보관 서비스 필요

사고 발생시 신속한 알림 중요

요구사항정의서

요구사항ID	요구사항명	기능ID	기능명	세부사항	예외사항
A01	경로 최적화	A01_B01	장애물 탐지	전방 혹은 측면에 장애물이 있을 경우 탐지 후 알림	장애물이 없을 경우
		A01_B02	조류, 가속도 감지	조류, 가속도 센서를 통해 입력 받은 값으로 최적의 경로 설정	조류가 약할 경우
		A01_B03	모터 방향 설정	입력받은 모든 값을 통합하여 원하는 방향으로 운행 하도록 모터 방향 설정	배가 멈춰있을 경우
A02	자율접안	A02_B01	거리 측정	항만과의 거리를 측정하고 충돌방지를 위하여 항만과 정보를 끊임 없이 공유(C/S)	배가 운행 중일 경우
		A02_B01	모터 방향 설정	조류, 풍향을 고려하고 항만과의 거리를 비교하여 안전하게 접안할 수 있도록 모터 방향 , 세기 설정	배가 운행 중일 경우

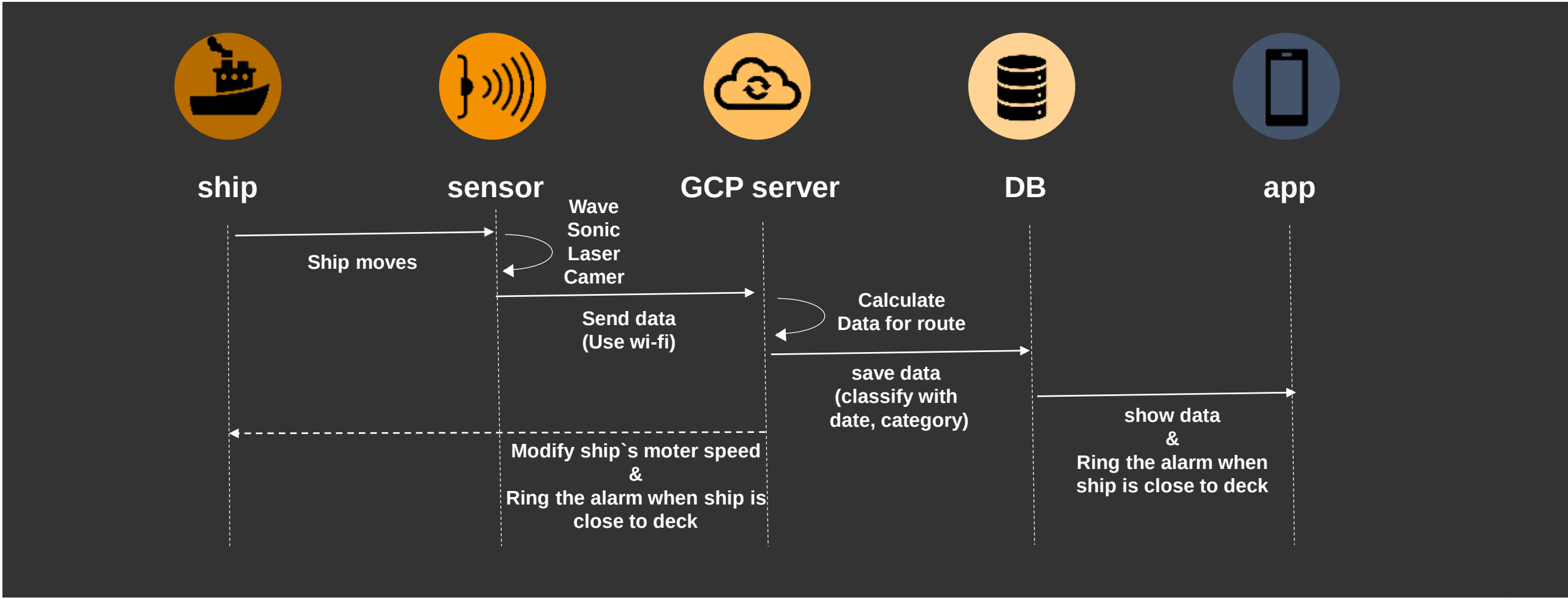
요구사항정의서

구분	기능	설명
S/W	Edge Computing	서버를 분산화하여 중앙집중관리 방식에서 보안의 최적화와 속도 지연문제를 해결하기위한 Edge Computing 기술. GCP(Google Cloud Platform) 사용
	자동/반자동 접안용 C-S개발	배와 항만을 각각 Client, Server로 지정하여 자동/반자동 접안시 서로 데이터를 끊임 없이 주고 받음
	IoT의 스마트패드(도선사 전용, 도선사의 접안신고전화),	배와 항만이 주고 받는 정보를 IoT의 스마트 패드를 통해 볼 수 있고, 이상이 생길 경우 도선사측이 이를 확인 후 비상 사고 방지. 앱 제작 완료
	접안Data(선박의 절대위치, 상대위치) 분석용 AI모의실험(시뮬레이터)	각종 데이터를 주고 받고 이를 자율적으로 처리 하여 배가 자율운행 가능하게 함
	Q-Data Capsule: 분석된 Data의 보안(전자봉인: e-Sealing) 및 양자(Quantum)암호화 통신(RSA 공개키 사용)	서버와 클라이언트 간의 정보가 해킹 당하거나 유출 되지 않도록 보안 . GCP를 사용하여 데이터 보관, 접근 및 보안까지 통합적으로 관리

구분	기능	설명
H/W	가속도센서	유량센서값을 입력받아 제어된 배의 속도와 방향 조절
	유량센서	배의 중심 혹은 방향을 지정할 수 있도록 함
	RF 통신모듈	배와 항만을 각각 Client,Server로 설정하여 서로 데이터를 공유할 수 있도록 함
	초음파 센서,레이져 센서	장애물 혹은 항만과의 거리가 너무 가까워 충돌하지 않도록 배와의 사이 거리를 측정하기 위함 현재 미니 라이더 센서로 대체 가능 여부 확인 중
	라즈베리 파이	RF 통신모듈을 사용하기 위해 라즈베리 파이 이용
	모터	다양한 입력 값들을 바탕으로 배가 원하는 방향으로 운행할 수 있도록 모터 조절

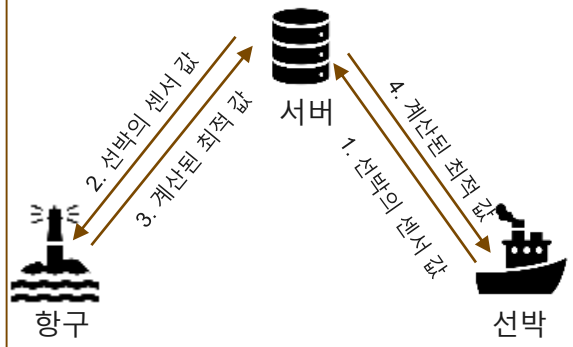
기능 처리도

기능 흐름도


ship
sensor
GCP server
DB
app
Ship moves
Wave
Sonic
Laser
Camer
Send data
(Use wi-fi)
Calculate
Data for route
save data
(classify with
date, category)
show data
&
Ring the alarm when
ship is close to deck
Modify ship`s moter speed
&
Ring the alarm when ship is
close to deck

서비스 구성도

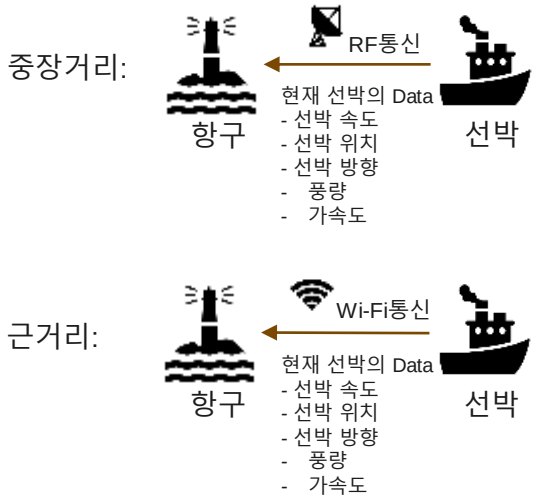
서비스 시나리오, 통신 방식



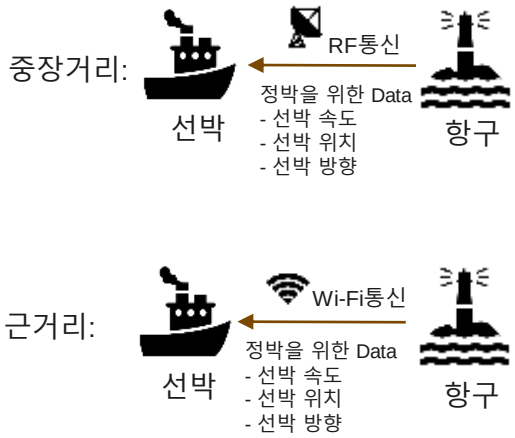
항구와 선박 간 서버를 이용한 통신

1. 선박에서 서버로 센서(유량, 자이로, 가속도, GPS)값 측정 후, 서버로 전송
2. 서버의 값을 항구에서 수신
3. 선박과 항구와의 거리와 선박에서 수신한 데이터를 종합하여 최적값을 계산한 후 서버로 업로드
4. 최적 값을 토대로 선박의 속도와 방향을 수정

항구 -> 선박

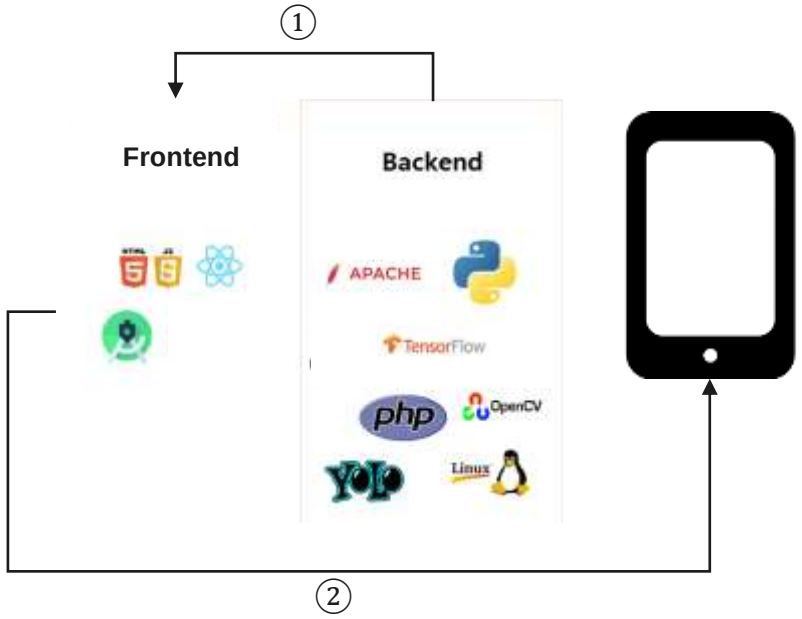


선박 -> 항구



서비스 구성도

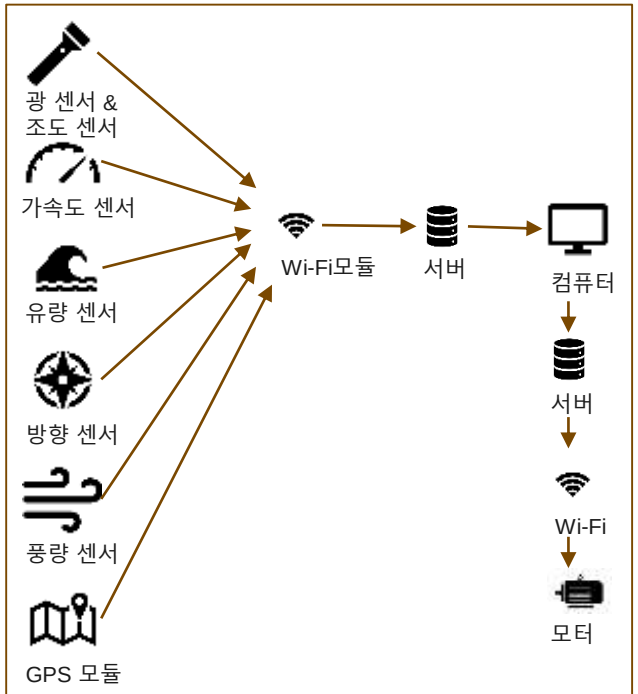
서비스 시나리오



- **Back-end**
  - ① 영상 분석 알고리즘(파이썬, opencv, YOLO)
  - ② 선박 자율주행 알고리즘: 감지된 물체, 선박에 부착된 센서들, 항만의 위치에 따른 이동방향 및 속도 결정
  - ③ 현재 선박의 위치 데이터를 서버, 항만에 전송
- **Front-end**
  - ④ 현재 선박의 위치, 이동속도 웹/앱 페이지에 표시하여 사용자가 가시적으로 확인

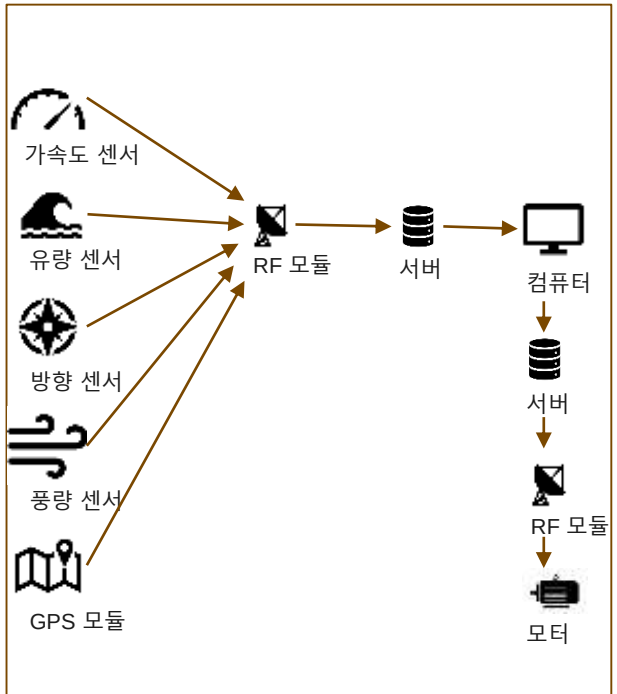
서비스 흐름도

단거리,중장거리 모듈



단거리 센서 및 모듈 흐름도

1. 선박에 탑재된 가속도 센서,유량센서, 방향 센서, 풍량 센서, GPS로 데이터를 수집, 두 개 이상의 광 센서로 항구와 선박의 수평 확인
2. Wi- Fi모듈을 이용하여 서버로 데이터 전송
3. 서버의 값을 네트워크와 연결된 PC를 통해 다운로드 및 조도 센서로 수신한 광 센서의 빛으로 최적 값 계산
4. 계산된 최적 값을 서버로 업로드
5. 최적 값을 서버에서 Wi- Fi 모듈로 수신
6. 수신된 최적 값으로 모터 제어

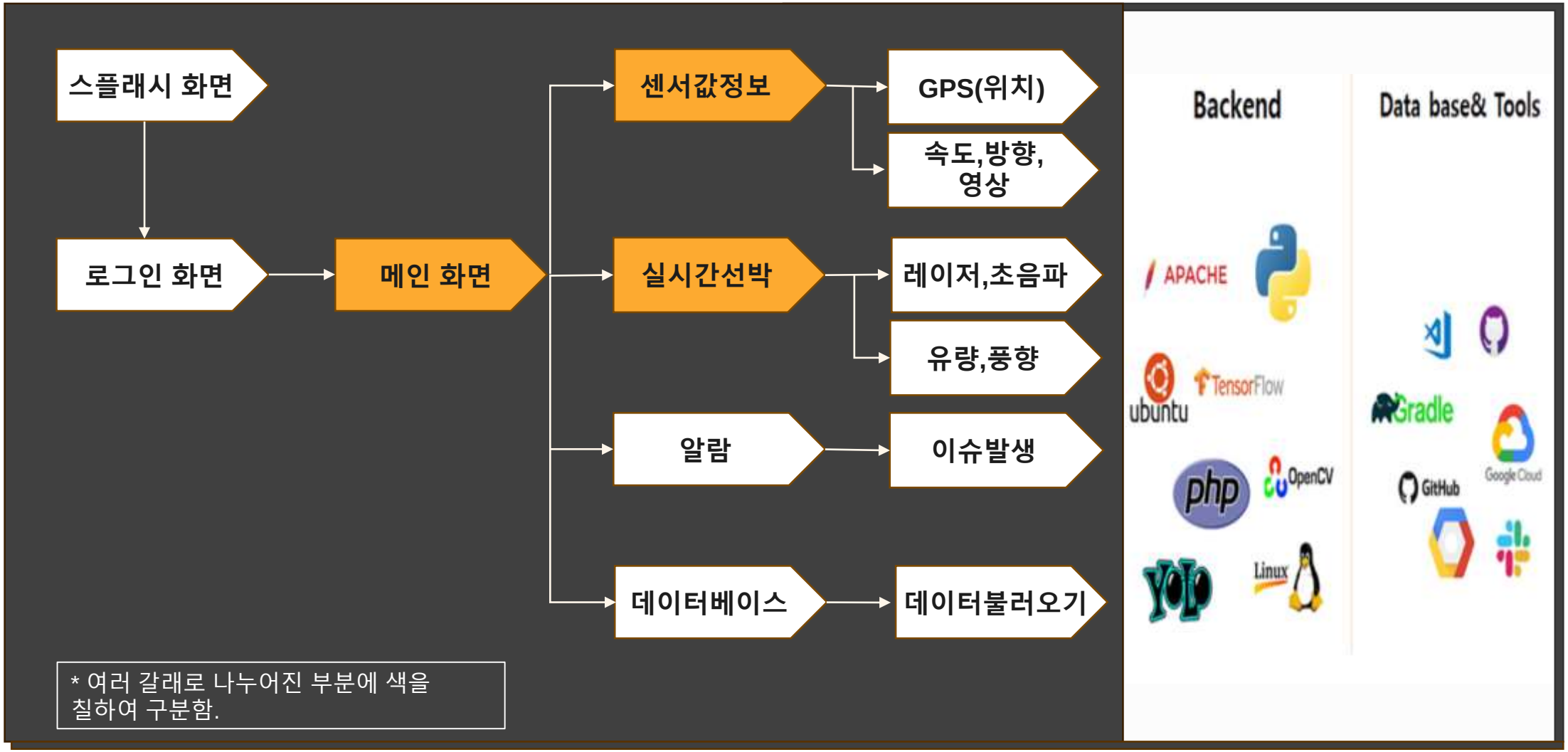


중장거리 센서 및 모듈 흐름도

1. 선박에 탑재된 가속도 센서, 유량 센서, 방향 센서, 풍량 센서, GPS로 데이터를 수집
2. RF 모듈을 이용하여 서버로 데이터 전송
3. 서버의 값을 네트워크와 연결된 PC를 통해 다운로드 후 최적 값을 계산
4. 계산된 최적 값을 서버로 업로드
5. 최적 값을 서버에서 RF모듈로 수신
6. 수신된 최적 값으로 모터 제어

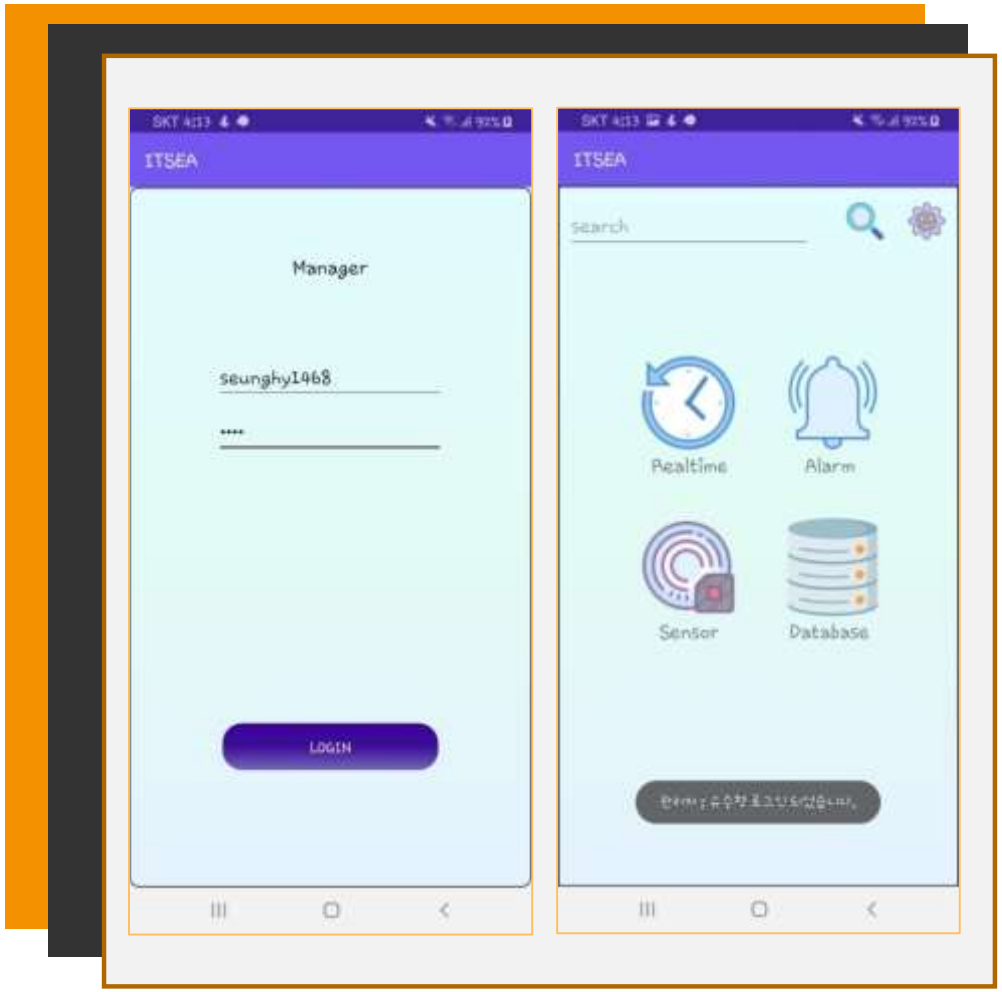
화면구성도

화면구성요소 마인드맵



화면설계서

관리자 화면 설계 및 기능설명



Login

로그인 화면

아이디 혹은 비밀번호가 틀릴 경우, 메인 화면으로 진입하지 못함. 관리자 계정으로만 로그인할 수 있도록 구성.

Main

메인화면

Realtime : 실시간으로 운항 혹은 접안 중인 선박의 카메라, GPS, 레이저, 초음파, 유량, 풍향센서를 이용하여 선박의 위치, 속도, 항구까지의 거리를 측정하여 데이터를 출력함. 또한 실시간 영상 데이터도 포함하여 자율 운항/자율 접안을 확인할 수 있음.

Alarm : 실시간으로 이슈가 발생하면 관리자에게 전달함. 또한 이슈 발생 시 날짜 별, 시간 별로 기록하여 관리자가 직관적으로 데이터를 받아드릴 수 있고 관리하기 편하도록 함.

Sensor : 레이저, 초음파, 유량, 풍향센서 값들을 출력하여 값의 추이를 살펴볼 수 있도록 함.

Database : 모든 데이터를 날짜 별, 시간 별로 기록함.

화면설계서

관리자 화면 세부설명

Realtime

실시간 선박 정보 화면

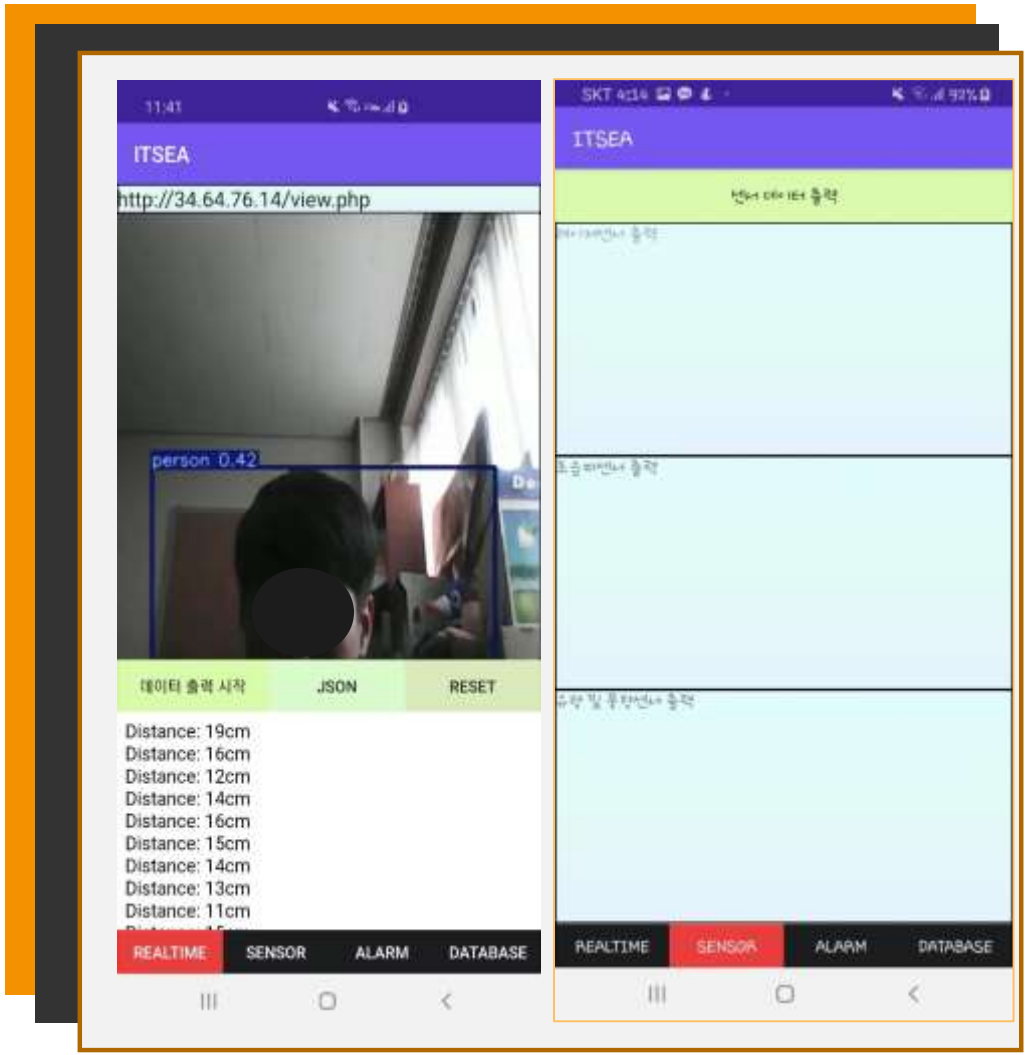
실시간으로 선박의 영상, 속도, 위치, 항구로부터의 거리 데이터를 측정한 값을 확인할 수 있다.
(예시 mp4 파일을 재생한 모습과 직접 설계한 서버에서 데이터를 어플리케이션에서 가져온 모습)

Sensor

센서 값 정보 화면

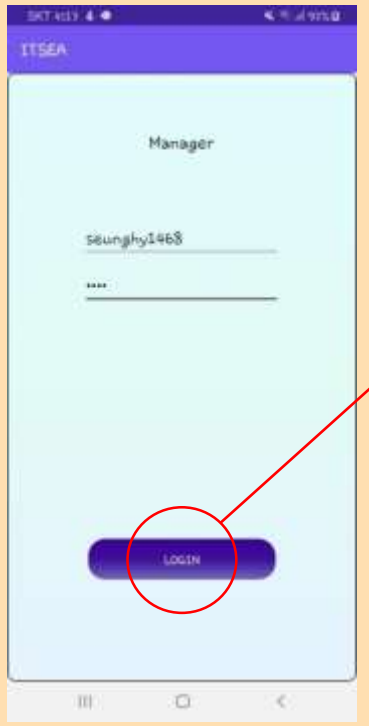
Realtime에서 확인할 수 없는 센서 데이터 값들을 확인할 수 있도록 구성했으며 레이저, 초음파, 유량 및 풍향 센서 값이 출력되도록 구성했다.

이러한 센서 값의 추이 등을 분석하여 자율 운항 및 자율 접안에서 효율적인 모터 속도, 선박이 바라보고 있는 방향 등을 설정할 때 이용할 수 있다.

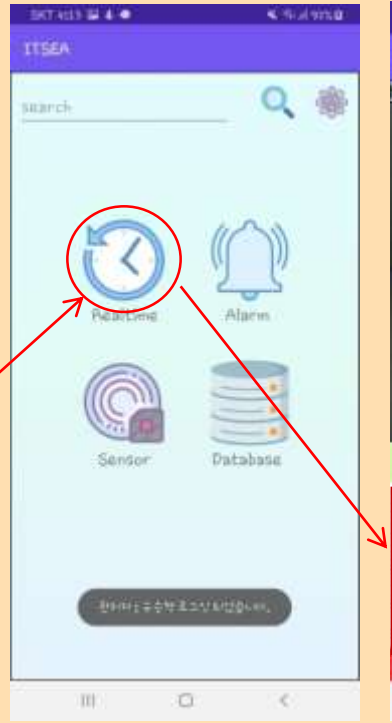


화면설계서

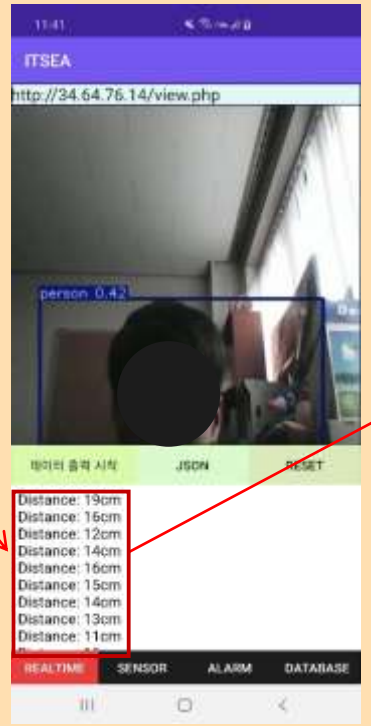
사용 예시 및 입출력 데이터



로그인 화면

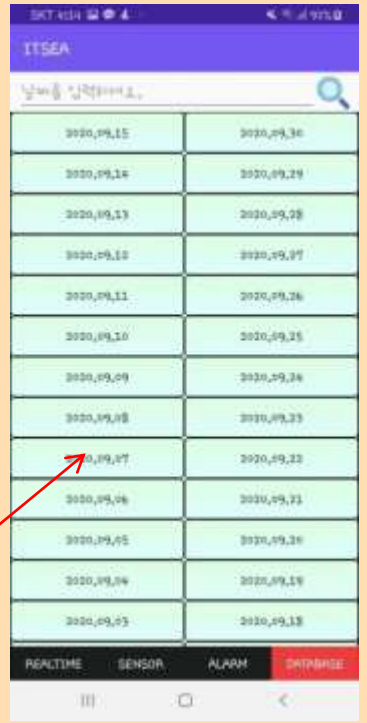


메인 화면



시스템 화면(서버에서 데이터를 받은 모습)

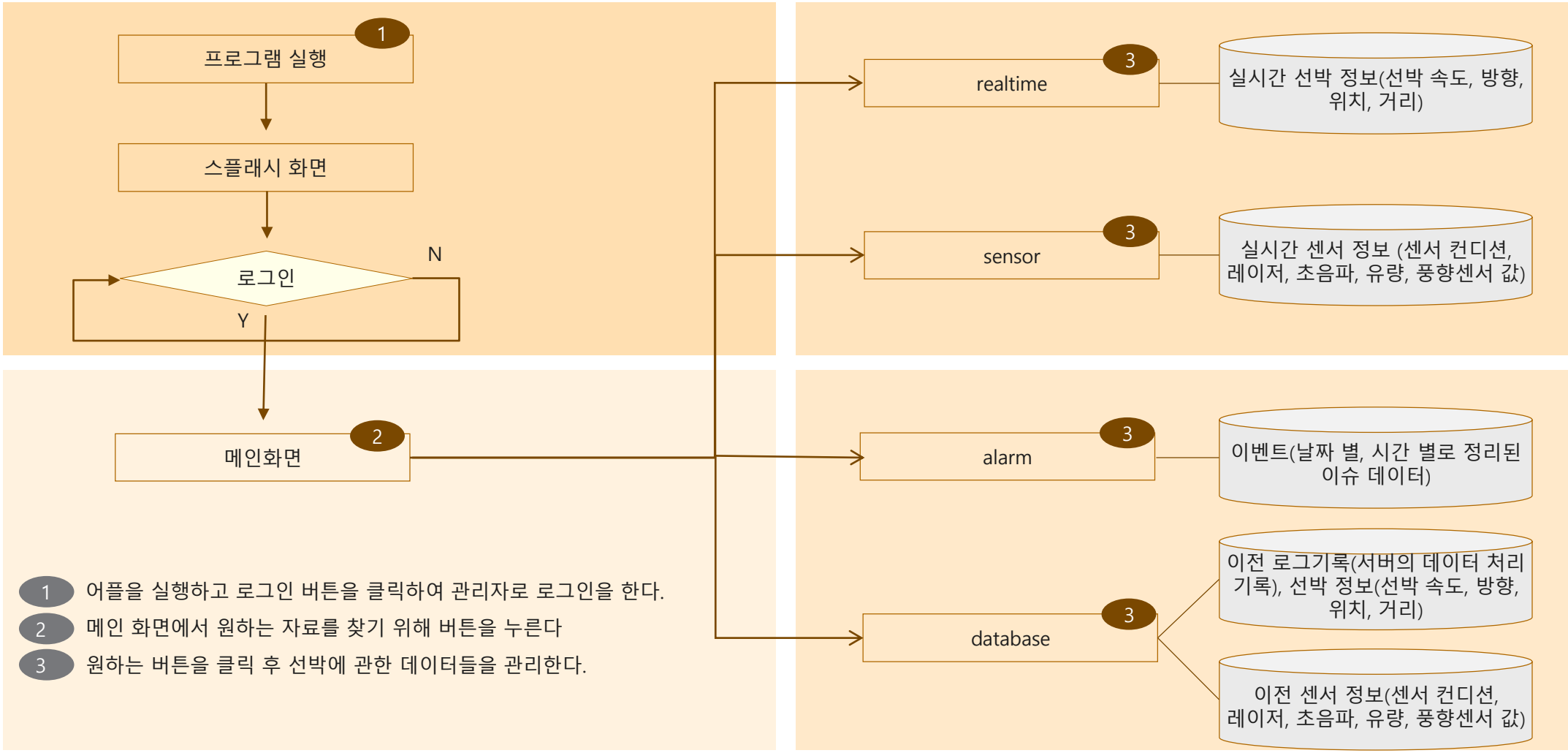
보안관리화면(날짜별로 정리되어있음)



9월 26일에 해당하는 data출력 화면

기능 처리도

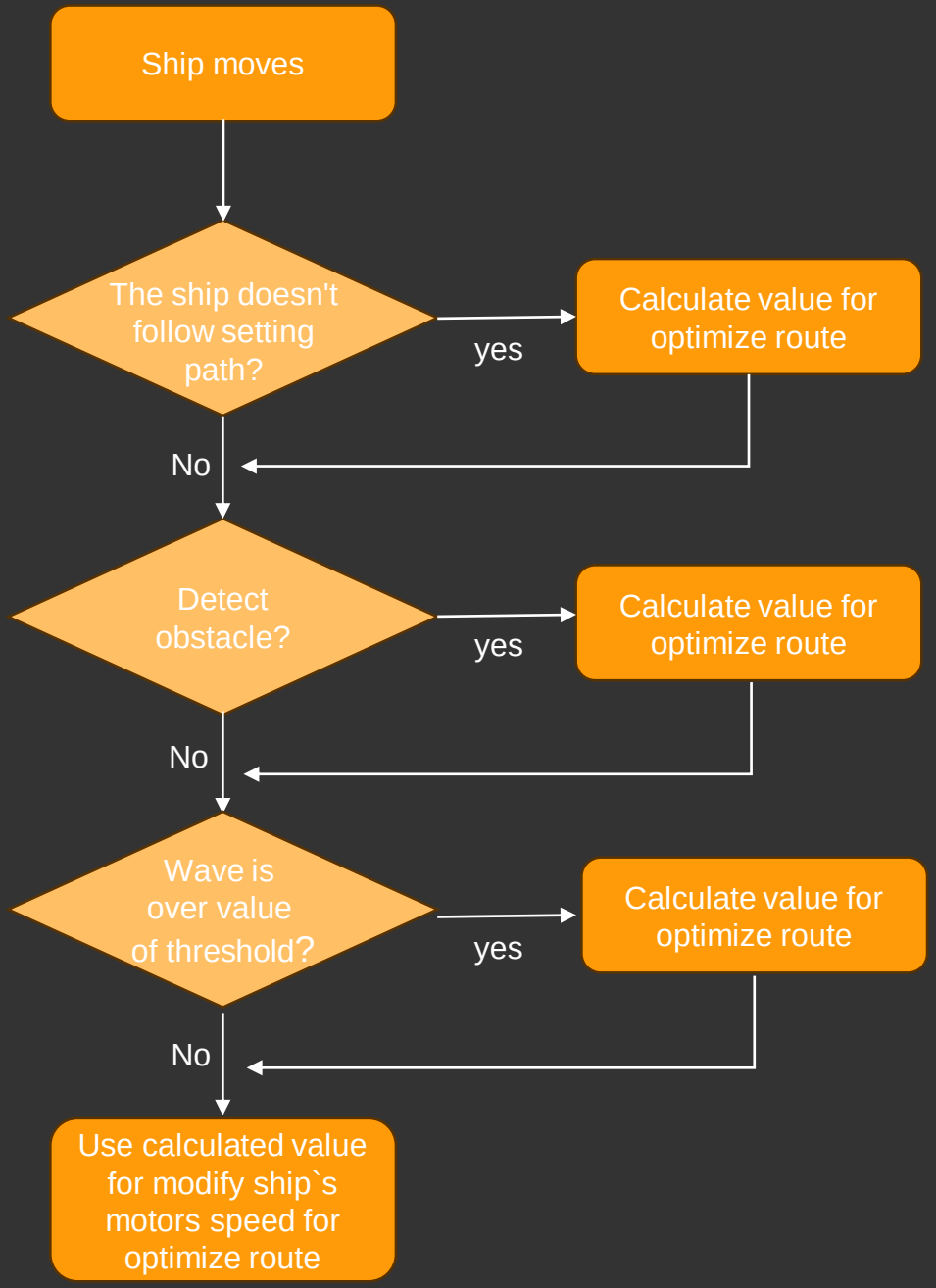
기능처리 구성의 알고리즘



1 어플을 실행하고 로그인 버튼을 클릭하여 관리자로 로그인을 한다.

2 메인 화면에서 원하는 자료를 찾기 위해 버튼을 누른다

3 원하는 버튼을 클릭 후 선박에 관한 데이터들을 관리한다.



알고리즘 명세서

선박 자동운항

선박

선박이 미리 지정된 경로에 따라 움직임

GPS

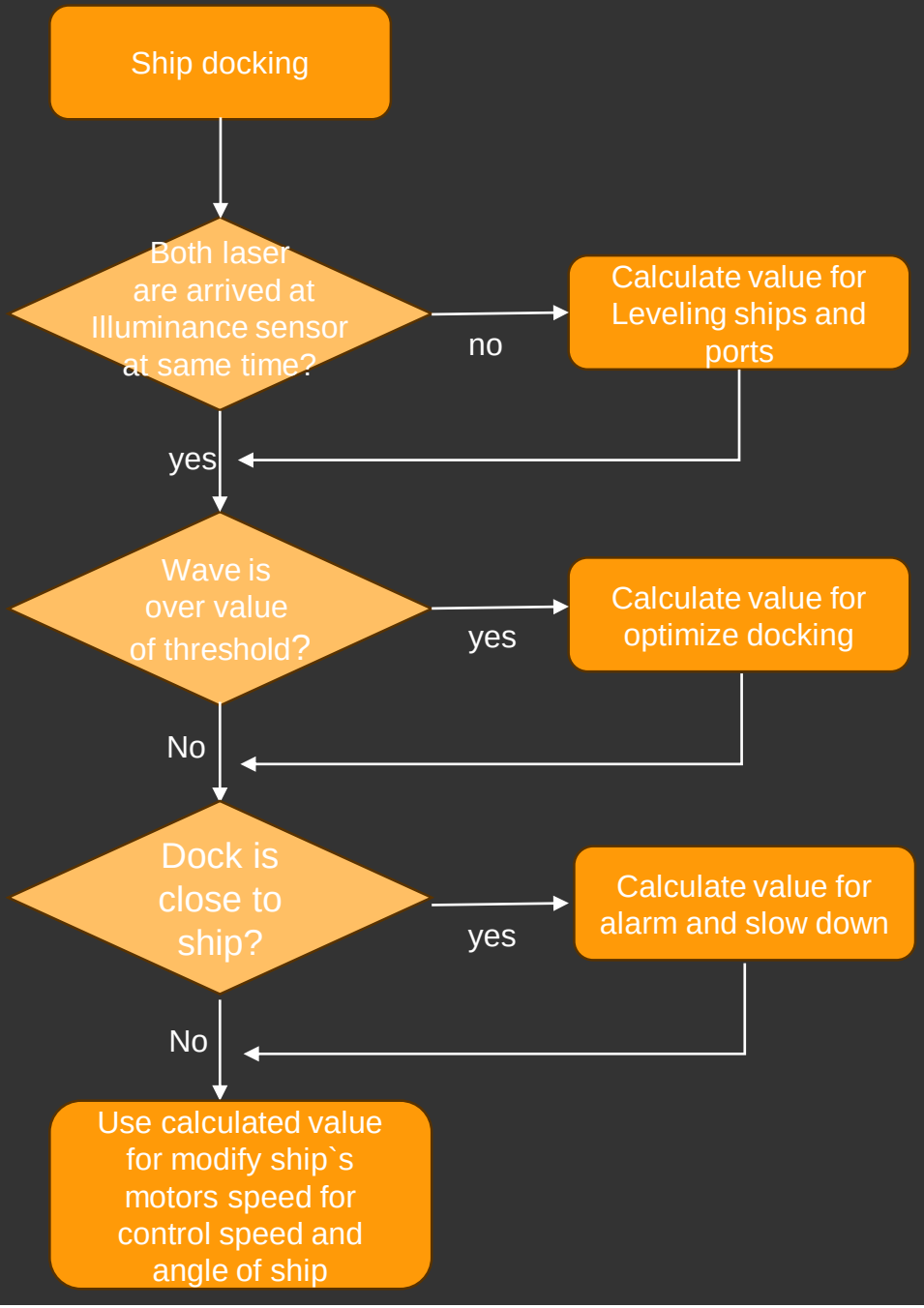
선박이 지정된 경로를 따라가는지 GPS를 통해 체크하고 지정된 경로 이탈 시 값을 수정 .

초음파 센서

초음파 센서를 이용하여 배 앞에 장애물의 존재를 탐지하여 장애물이 있으면 장애물을 피하는 최적의 경로를 계산

유량센서

유량센서를 통해 파량을 측정하며, 파도가 임계 값을 넘을 경우, 해당 값이 배의 경로에 주는 영향을 고려하여 경로를 수정



알고리즘 명세서

선박 자동접안

선박

부두 근처에서 접안을 시작.

레이저

레이저를 통해 배와 항구사이의 수평여부를 판단하여 배의 각도를 수정 .

유량센서

유량센서를 통해 파량을 측정하며, 파도가 임계 값을 넘을 경우, 해당 값이 배의 접안에 주는 영향을 게산.

초음파 센서

초음파 센서를 이용하여 배와 항구 사이의 거리를 파악 하여 가까워 짐에 따른 알람을 울리고 속도를 줄임.

알고리즘 상세 설명서

자율 주행과 자율 접안모드

자율 주행 알고리즘

상황을 설정하여 그에 맞는 항로를 맞춰 이동하는지 확인,
선박이 항로를 벗어나는 경우를 방지하기 위해GPS를
사용. 선박에 부착된 센서들을 통해 유량을 측정, 전방의 물체를
감지한다. 측정된 데이터를 종합하여 외부요소를 극복할 수 있는
방향과 속도 값을 교정하여 선박의 경로이탈을 방지한다.
또한 전방에 물체(타선박/암초)가 감지되면 물체와의
거리 값을 계산해 선박의 움직임을 교정하도록 설계한다.
항구에 일정 이상 가까워진 경우, 접안 알고리즘으로 전환한다.

자율 접안 알고리즘

선박에 부착된 두개 이상의 레이저를 방출한 뒤
이 레이저들이 항구에 부착된 조도센서가 인지하여
감지하여 항구와 수평이 되는 배의 접안각도를 정한다.
배의 속도는 초음파 센서와 레이저 센서를 복합적으로 이용하여
항구와 가까워 짐에 따라 속도를 줄이며 근접할 경우 부저를 통해
알린다. 실제 항만 환경에 착안하여 예인선과 모선을 따로
설계하였고. 예인선이 모선의 접안을 위해 밀며 부두와 모선사이
거리를 계산하여 모터 계산을 한다.

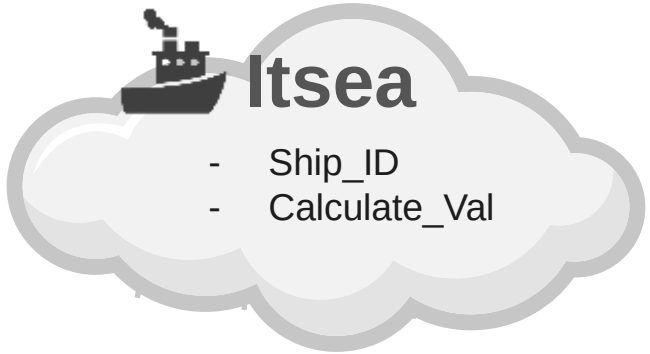
프로그램

목록

기능 분류	기능번호	기능 명
STR	STR-01-01	선박 전진
	STR-01-02	선박 우측으로 회전
	STR-01-03	선박 좌측으로 회전
	STR-01-04	선박 후진
MES	MES-02-01	조도 측정
	MES-02-02	풍향 측정
	MES-02-03	초음파 측정
	MES-02-04	유량 측정
COM	COM-02-01	와이파이 검색
	COM-02-0101	와이파이 검색 - 와이파이 선택
	COM-02-02	데이터 송신
	COM-02-03	데이터 수신
MoV	MOV-03	센서 측정값 기반 선박의 이동 명령

테이블 정의서

GCP mysql



GPS
- Location

sensing_light
- Sensing_time
- Brightness

sensing_sonic
- Distance

sensing_wave
- Wave_val

테이블 정의서

GCP mysql

<itsea>	<itsea>	<GPS>	<Sensing light>	<Sensing light>	<sensing_sonic >	<sensing_wave>
항목명 Ship_ID	항목명 Calculate_Val	항목명 Location	항목명 Sensing_time	항목명 Brightness	항목명 Distance	항목명 Wave_val
Type -> String	Type -> Num	Type -> Num	Type -> Num	Type -> Num	Type -> Num	Type -> Num
필수/선택 -> 필수	필수/선택 -> 필수	필수/선택 -> 필수	필수/선택 -> 필수	필수/선택 -> 필수	필수/선택 -> 필수	필수/선택 -> 필수
활성여부 -> 활성	활성여부 -> 활성	활성여부 -> 활성	활성여부 -> 활성	활성여부 -> 활성	활성여부 -> 활성	활성여부 -> 활성
설명 -> 배를 식별하기 위한 테이블	설명 -> 측정한 센서 값을 계산하여 저장하기 위한 테이블	설명 -> 배의 위치를 파악하고 수정하기 위한 테이블	설명 -> 수평여부를 판단하기 위한 값을 저장하는 테이블	설명 -> 거리 측정을 위한 밝기를 저장하는 테이블	설명 -> 초음파센서를 이용한 거리 측정을 위한 테이블	설명 -> 파도에 의한 오차를 보완하기 위한 값을 저장하는 테이블

핵심소스코드

detect_video.py

```python
import time
from absl import app, flags, logging
from absl.flags import FLAGS
import cv2
import tensorflow as tf
from yolov3_tf2.models import (
    YoloV3, YoloV3Tiny
)
from yolov3_tf2.dataset import transform_images
from yolov3_tf2.utils import draw_outputs


flags.DEFINE_string('classes', './data/coco.names', 'path to classes file')
flags.DEFINE_string('weights', './checkpoints/yolov3.tf',
                    'path to weights file')
flags.DEFINE_boolean('tiny', False, 'yolov3 or yolov3-tiny')
flags.DEFINE_integer('size', 416, 'resize images to')
flags.DEFINE_string('video', './data/video.mp4',
                    'path to video file or number for webcam)')
flags.DEFINE_string('output', None, 'path to output video')
flags.DEFINE_string('output_format', 'XVID', 'codec used in VideoWriter when saving video to file')
flags.DEFINE_integer('num_classes', 80, 'number of classes in the model')
```

```python
def main(_argv):
    physical_devices = tf.config.experimental.list_physical_devices('GPU')
    for physical_device in physical_devices:
        tf.config.experimental.set_memory_growth(physical_device, True)

    if FLAGS.tiny:
        yolo = YoloV3Tiny(classes=FLAGS.num_classes)
    else:
        yolo = YoloV3(classes=FLAGS.num_classes)

    yolo.load_weights(FLAGS.weights)
    logging.info('weights loaded')

    class_names = [c.strip() for c in open(FLAGS.classes).readlines()]
    logging.info('classes loaded')

    times = []

    try:
        vid = cv2.VideoCapture(int(FLAGS.video))
    except:
        vid = cv2.VideoCapture(FLAGS.video)

    out = None
```

```python
    if FLAGS.output:
        # by default VideoCapture returns float instead of int
        width = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = int(vid.get(cv2.CAP_PROP_FPS))
        codec = cv2.VideoWriter_fourcc(*FLAGS.output_format)
        out = cv2.VideoWriter(FLAGS.output, codec, fps, (width, height))

    while True:
        _, img = vid.read()

        if img is None:
            logging.warning("Empty Frame")
            time.sleep(0.1)
            continue

        img_in = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_in = tf.expand_dims(img_in, 0)
        img_in = transform_images(img_in, FLAGS.size)

        t1 = time.time()
        boxes, scores, classes, nums = yolo.predict(img_in)
        t2 = time.time()
        times.append(t2-t1)
        times = times[-20:]

        img = draw_outputs(img, (boxes, scores, classes, nums), class_names)
        img = cv2.putText(img, "Time: {:.2f}ms".format(sum(times)/len(times)*1000), (0, 30),
                          cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 2)
```

핵심소스코드

detect_video.py

```
79          if FLAGS.output:
80              out.write(img)
81          cv2.imshow('output', img)
82          if cv2.waitKey(1) == ord('q'):
83              break
84
85      cv2.destroyAllWindows()
86
87
88  if __name__ == '__main__':
89      try:
90          app.run(main)
91      except SystemExit:
92          pass
```

detect_video.py

자율주행에 있어서 핵심코드. Tensor flow, opencv를 활용한 딥러닝 기반의 Object Detection 프로그램 YOLO V3를 통하여 전방에 존재하는 물체를 감지함.

실시간으로 촬영되는 영상에서 객체를 감지할 때마다 이동경로를 자동으로 수정. detect.py 파일을 실행하여 카메라를 연결해서 실시간으로 물체 감지

핵심소스코드

serial_communicate.py

```
while(True):
    tty.setcbreak(sys.stdin)                # sys.stdin : 모든 대화식 입력에 사용. (콘솔입력에 사용)
    key = ord(sys.stdin.read(1))            # 파일 기술자 fd의 모드를 cbreak로 변경(리눅스 or 맥)
                                            # key captures the key-code
    time.sleep(0.050)                       # ord : 하나의 유니코드 문자를 나타내는 문자열이 주어지면
                                            # 해당 문자의 유니코드 코드 포인트를 나타내는 정수를 돌려준.
                                            # ex) ord('a') --> 정수 97을 반환
    #rotate right if d is pressed
    if key==100: #27 is d key
        ser.write('00aw00030255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00030000;'.encode())

    #rotate left is a is pressed
    elif key==97: #97 is a key
        ser.write('00aw00090255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090000;'.encode())

    #go forward if w is pressed.
    elif key==119: #119 is w key
        ser.write('00aw00030255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00030000;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090000;'.encode())

    elif key==113: #113 is q key
        break
```

```
while(True):
    tty.setcbreak(sys.stdin)                # sys.stdin : 모든 대화식 입력에 사용. (콘솔입력에 사용)
    key = ord(sys.stdin.read(1))            # 파일 기술자 fd의 모드를 cbreak로 변경(리눅스 or 맥)
                                            # key captures the key-code
    time.sleep(0.050)                       # ord : 하나의 유니코드 문자를 나타내는 문자열이 주어지면
                                            # 해당 문자의 유니코드 코드 포인트를 나타내는 정수를 돌려준.
                                            # ex) ord('a') --> 정수 97을 반환
    #rotate right if d is pressed
    if key==100: #27 is d key
        ser.write('00aw00030255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00030000;'.encode())

    #rotate left is a is pressed
    elif key==97: #97 is a key
        ser.write('00aw00090255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090000;'.encode())

    #go forward if w is pressed.
    elif key==119: #119 is w key
        ser.write('00aw00030255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090255;'.encode())
        time.sleep(0.010)
        ser.write('00aw00030000;'.encode())
        time.sleep(0.010)
        ser.write('00aw00090000;'.encode())

    elif key==113: #113 is q key
        break
```

핵심소스코드

serial_communicate.py

```
#rotate right if d is pressed
if key==100: #27 is d key
    ser.write('00aw00030255;'.encode())
    time.sleep(0.010)
    ser.write('00aw00030000;'.encode())

#rotate left is a is pressed
elif key==97: #97 is a key
    ser.write('00aw00090255;'.encode())
    time.sleep(0.010)
    ser.write('00aw00090000;'.encode())

#go forward if w is pressed.
elif key==119: #119 is w key
    ser.write('00aw00030255;'.encode())
    time.sleep(0.010)
    ser.write('00aw00090255;'.encode())
    time.sleep(0.010)
    ser.write('00aw00030000;'.encode())
    time.sleep(0.010)
    ser.write('00aw00090000;'.encode())

elif key==113: #113 is q key
    break

else:
    ser.write('00aw00030000;'.encode())
    time.sleep(0.010)
    ser.write('00aw00090000;'.encode())
```

serial_communicate.py

serial 라이브러리를 import하여 유니코드 ↔ 바이트 변환을 하여 serial 통신을 시행.

선박의 수동 운전을 조작하는 코드로 ‘w’, ‘a’, ‘s’, ‘d’ 키를 활용한 조작코드를 while 무한루프문으로 구현. ‘q’ 입력시 프로그램이 종료됨.

참조-개발 환경 및 설명

구분		항목	적용내역
S/W 개발환경	스마트폰 App 개발	Android Studio (1.2.2)	Android Studio를 이용한 application 개발
		안드로이드 OS(5.0.1)	스마트폰 운영체제
	서버 애플리케이션 개발	PHP(5.3.3)	서버 관리자 웹 페이지 처리 모듈 작성
		MySQL(5.1.73)	자율주행에 필요한 데이터를 저장, 관리하는 데이터베이스
		서버운영체제	Linux Ubuntu, Window CentOS 6.6, GCP(Google Cloud Platform)
		Putty	클라우드 서버 CentOS 원격 접속용 툴 , RSA 키 생성(Putty gen)
	GCP (Google Cloud Platform)	VM instance	가상 서버 설계 저장소, 앱과 서버 연결시 필요한 터미널 구축
		메타데이터 및 SSH	보안을 위한 RSA 키 저장
	자율주행 소프트웨어 개발	Python (3.7)	인공지능형 알고리즘 개발
		MATLAB	신호처리 프로그램 개발
	웹 서버, 데이터 교류 가시화	ATOM, HTML, CSS	서버와 클라이언트의 데이터 교류가 잘 이루어지고 있는지 보여주기 위한 중간 지점구축
	딥러닝, 영상처리	YOLO v3, OpenCV	선박에 부착되어있는 웹캠 영상을 서버로 받아와 물체인식 알고리즘 적용
	NAS 시스템	Portainer, Next Cloud	크기가 큰 영상 혹은 대용량 업데이트 시 빠른 파일 전송을 위한 추가 서버 구축

참조-개발 환경 및 설명

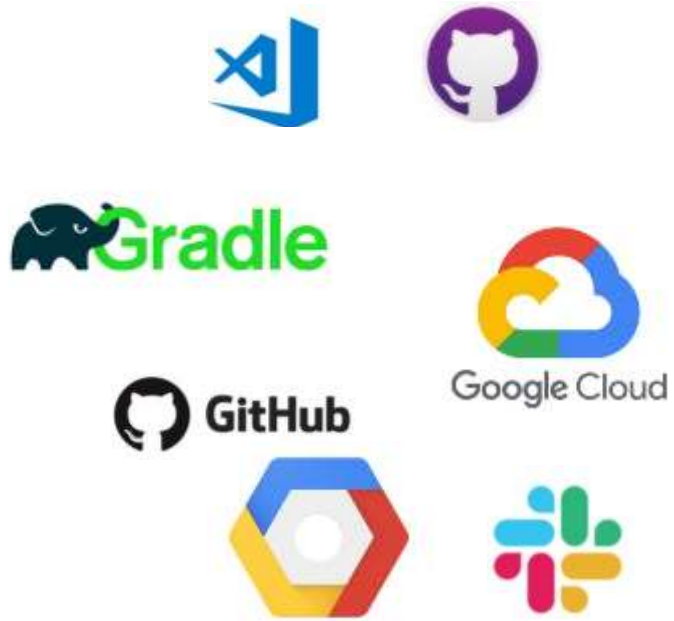
구분		항목	적용내역
H/W 구성장비	Sensor	유량 센서	파도의 세기 측정
		레이저 센서&조도 센서	레이저를 통해 도착지 까지의 근접 거리 측정
		풍향 센서	바람의 세기 측정
		초음파 센서	운행 혹은 접안 중 근처 근접 물체 탐지
	Device & Module	스마트폰 (안드로이드, 아이폰)	사용자에게 서비스를 직접적으로 제공하는 End Device
		아두이노보드(UNO)	센서 및 모터 부착과 작동에 필요, 포트 사용이 많은 구역 Mega 사용
		라즈베리파이	아두이노 제어 , GPS기술 활용 및 항만과의 데이터 교류
	기타	모형 배, 수조	모형 배 제작 및 자율운행 구동 여부 확인 위해 수조를 이용한 시뮬레이션
		방수모터(A2212/13T)	선박을 이동시키기 위한 모터와 프로펠러

Development Environments

Frontend

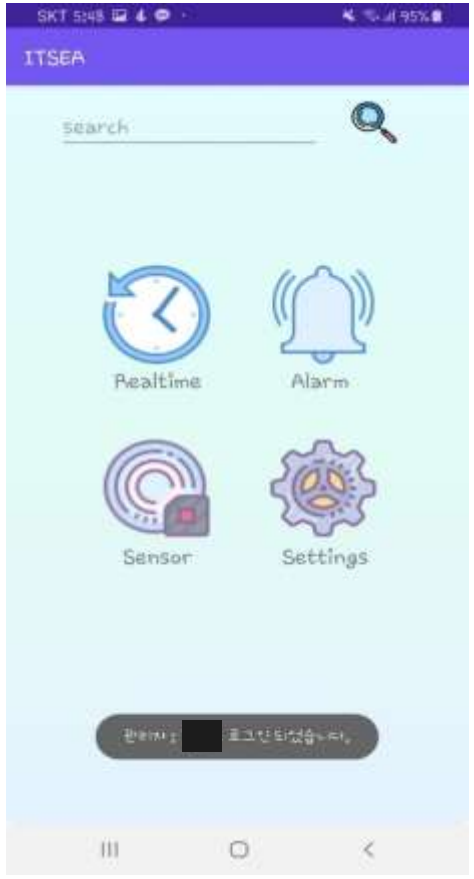
Backend

Data base& Tools



참조 - S/W 기능 실사 사진


ITSEA
Manager
yunmo
LOGIN

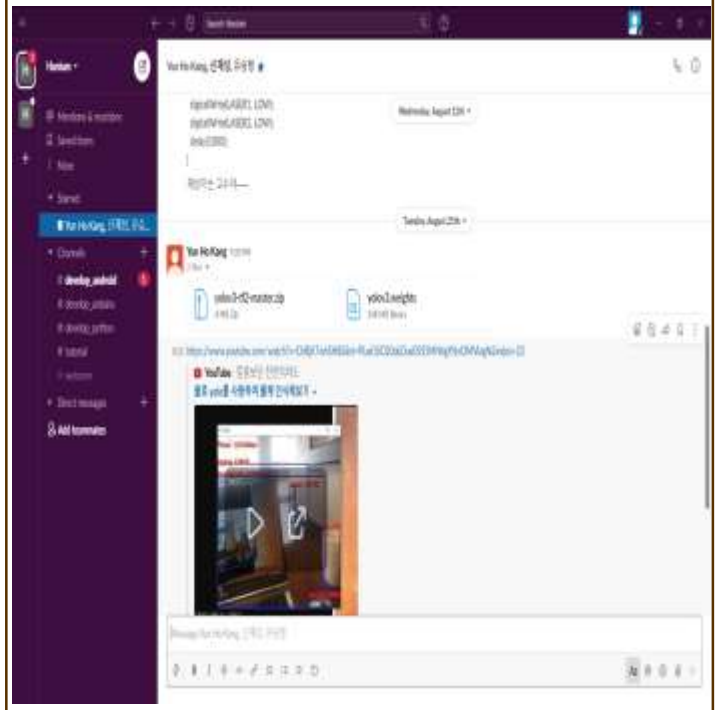

ITSEA
search
Realtime
Alarm
Sensor
Settings


ITSEA
http://34.64.76.14/view.php
person 0.42
JSON
RESET
Distance: 19cm
Distance: 16cm
Distance: 12cm
Distance: 14cm
Distance: 16cm
Distance: 15cm
Distance: 14cm
Distance: 13cm
Distance: 11cm
REALTIME
SENSOR
ALARM
DATABASE

참조 – H/W 기능 실사 사진

참조 – 프로젝트 관리

프로젝트 관리(Slack)



형상 관리

회의록 및 기획 관리

Thank you