

박물관 안내로봇

안광필, 김민석, 유연중, 오준원, 이윤성



Index

01

주제선정 이유 (프로젝트 방향성)



02

개발환경 - SerBot



03

IMU 각도 계산 , UI (PyQt5)



04

구동 - Zigbee, Lidar , 맵핑



05

오디오 (google assistant, TTS, PYaudio)



06

Photo - Zone , Mask - detect (opencv, pytorch yolov5)

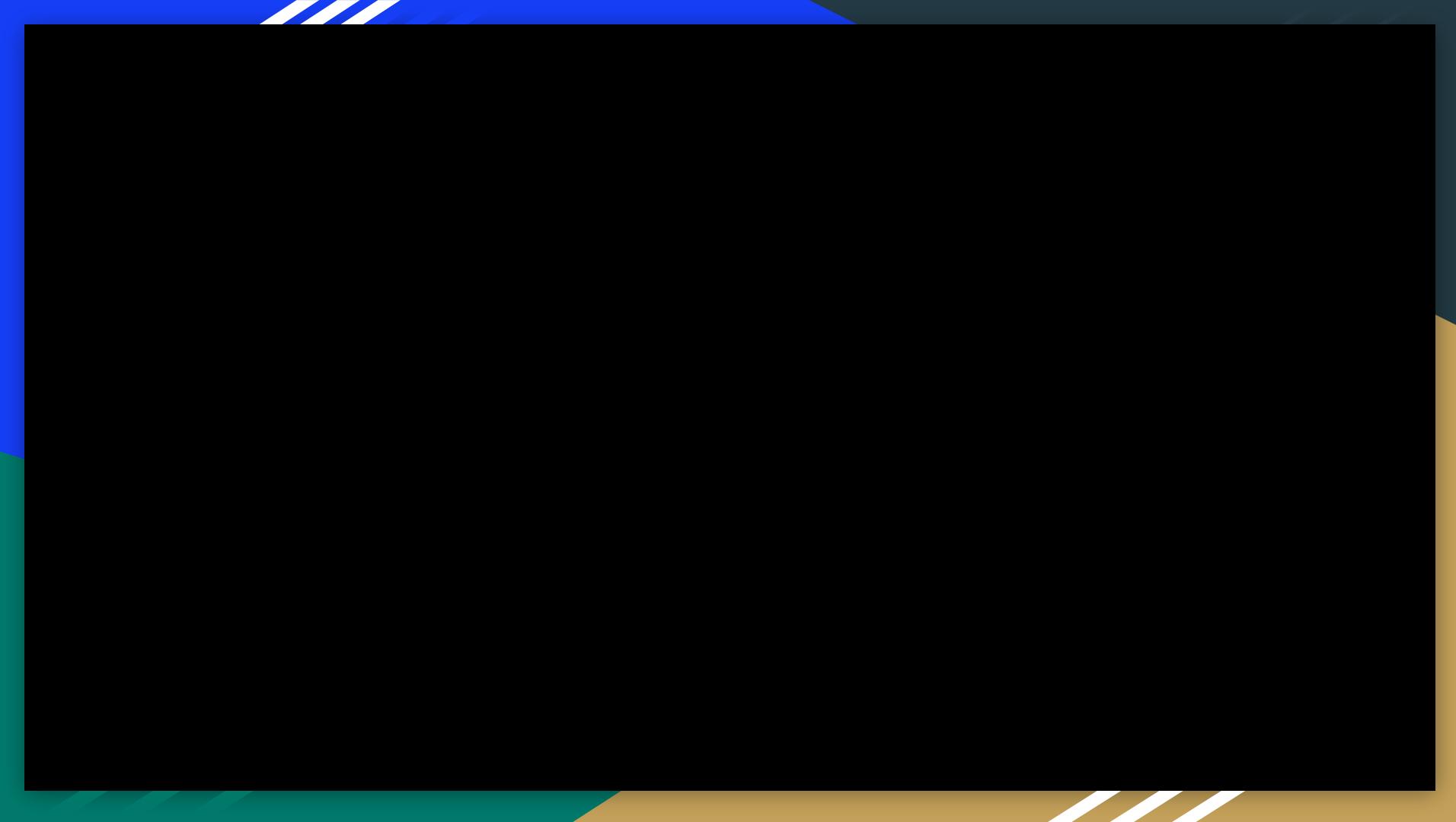


07

마무리 (아쉬운점)



주제선정이유



왕릉 발굴 50돌·갱위강국 1500돌 '무령왕의 해' 기린다

 노형석 기자 [+구독](#)

[f](#) [t](#) [talk](#) [o](#) [★](#) [□](#) [가+](#)

문화재청-공주시 25일 선포식 등 함께
백제문화권 핵심유적 학술조사연구



1971년 7월 무령왕릉 발굴 장면 국립문화재연구소 제공

공주 무령왕릉 발굴 50돌.. 무령왕 동상 선다

[△ 정영순 기자 7000ys@dailycc.net](#) | [○ 승인 2020.12.16 13:17](#) | [💬 댓글 0](#)

| 갱위강국 선포 1500주년 기념.. 내년 9월 공산성 앞에 건립



백제 25대 무령왕 동상 조감도(공주시 제공)

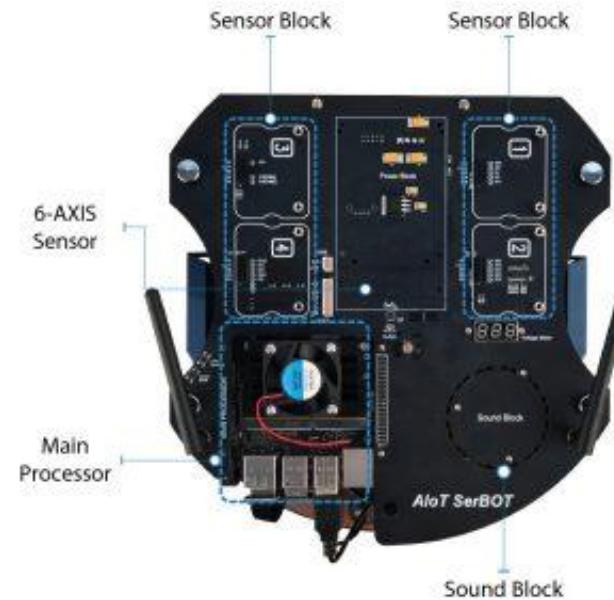


← 지도에서 목적지를 클릭하세요



안내시작 🔍

개발환경



Serbot 목표 위치까지로 이동

Zigbee IMU 센서 - 절대 방향, Serbot 방향 제어

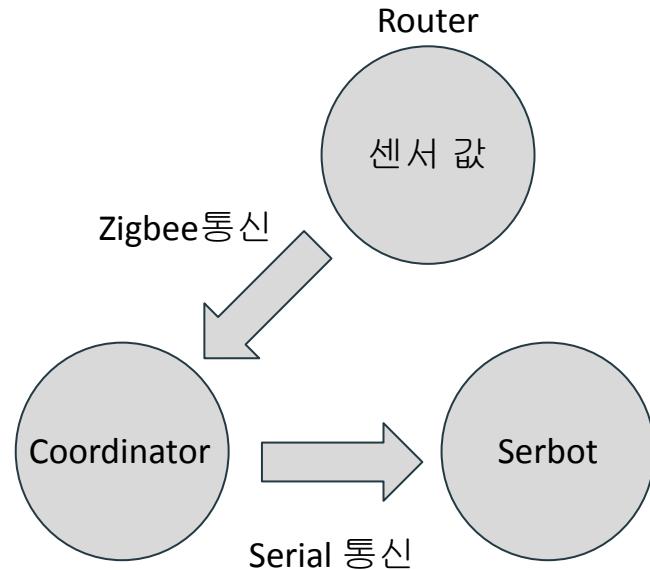
Zigbee PIR 센서 - 움직임 감지, 손님 입장 확인

2차원 배열 - 맵 구현, 특정 좌표 지정

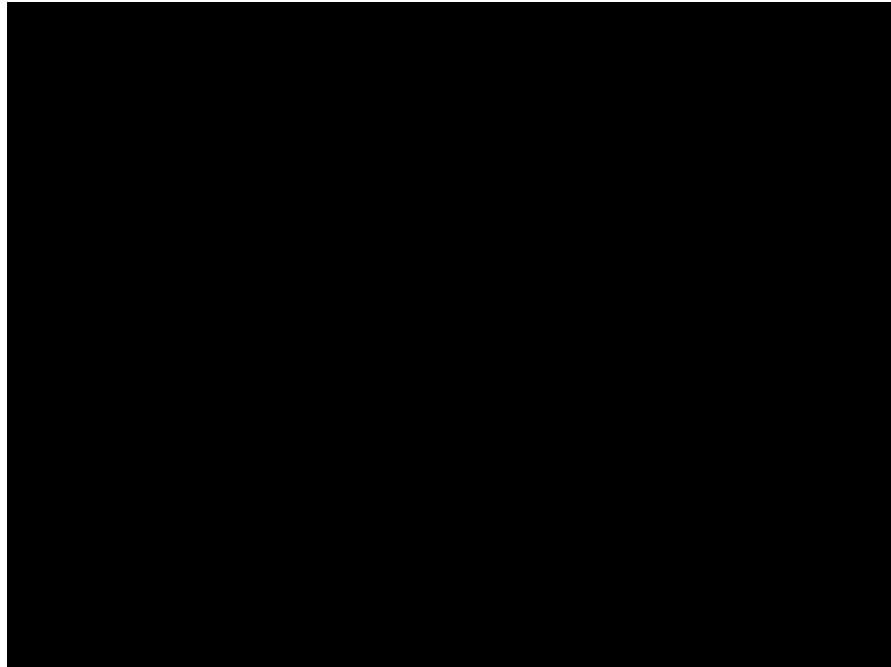
LIDAR - 장애물 회피 및 좌표 재설정

IMU, PIR 센서 값 전달

- Zigbee 통신 Coordinator 와 Router
- Serbot과 Xnode의 Serial 통신



IMU, PIR 센서 값 전달



IMU 센서 활용

- 쓰레드로 실시간 IMU 값 측정하여 각도 계산
- 목표 방향에서 5도 이상 벗어날 시 회전하여 조정
- 5도에서 2도 사이는 바퀴의 속도 제어로 방향 조정



2차원 배열 맵핑

- Serbot 위치 : 1
- 목표 위치 : 2
- 중간 지점 : 3
- 장애물 : 4

2	0	0	3	0
4	4	4	0	4
1	0	0	3	0

2차원 배열 맵핑

- 목적지와 출발지의 0번째 인덱스 비교
- 거치지 않는 중간 지점 0으로 변경

2	0	0	3	0	0	0
4	4	4	0	4	4	4
1	0	0	3	0	0	0
4	4	4	0	4	4	4
0	0	0	3	0	0	0

2차원 배열 맵핑

- 목적지와 중간지점들의 0번째 인덱스 비교
- 제일 가까운 중간지점으로 이동

2	0	0	3	0	0	0
4	4	4	0	4	4	4
1	0	0	3	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0

2차원 배열 맵핑

- 통과한 중간지점은 0으로 변경
- 제일 가까운 중간지점으로 이동

2	0	0	3	0	0	0
4	4	4	0	4	4	4
0	0	0	1	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0

2차원 배열 맵핑

- 현재위치 1번째 인덱스에 목적지 유무 확인
- 확인 후 목적지로 이동

2	0	0	1	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0

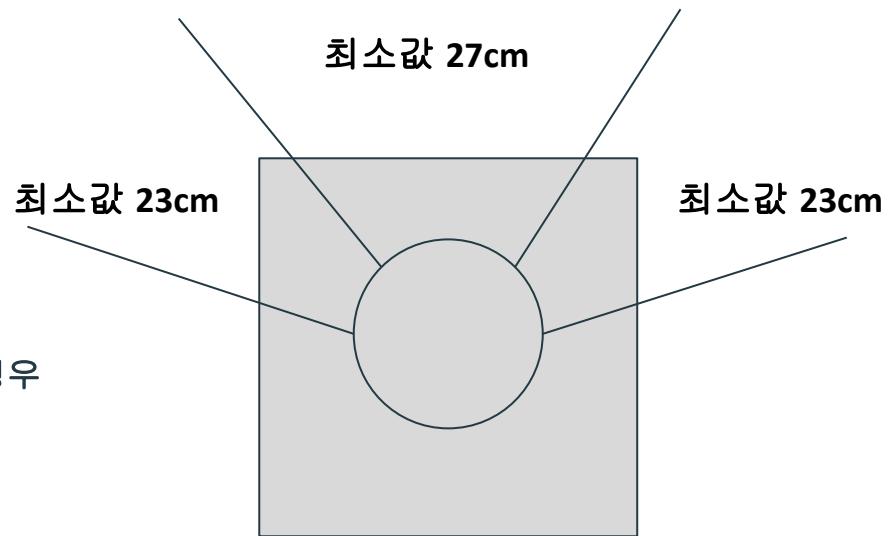
2차원 배열 맵핑

- 좌표 하나당 대략적인 이동시간 측정
- 오차가 생긴 현재위치를 재설정 하는 지점

1	0	0	0	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0
4	4	4	0	4	4	4
0	0	0	0	0	0	0

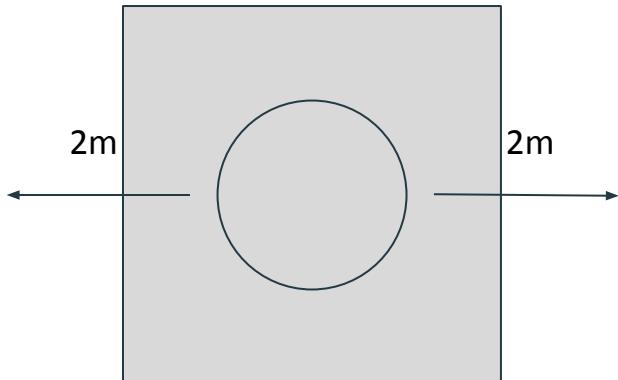
LIDAR - 장애물 회피

- 쓰레드로 실시간 라이더 값 측정
- 330 ~ 30도의 최소값이 27이하인 경우 정지
- 30 ~ 70도와 290 ~ 330도의 최소값이 23이하인 경우
좌 우로 이동하여 부딪힘 방지



LIDAR - 현재 좌표 재설정

- 중간지점 주변에 있을 때 중간지점의 특징인
좌우 벽과의 거리가 2m 이상일 경우
현재 위치에서 벗어나 중간지점으로 위치 설정
- 라이더 값에 오류가 있을 수 있으니 라이더 값의
일정범위의 평균값 그리고 3번 연속 2m 이상 값을
가져왔을 경우에만 적용



LIDAR - 현재 좌표 재설정



ZigBee - 네트워크망 구성

코디네이터와 라우터간 데이터 송수신

```
from pop import xnode
```

```
NI = "SBCood"  
CE = 0x01      # 0x00 or 0x01  
ID = 0x70
```

```
xnode.atcmd('NI', NI)  
xnode.atcmd('CE', CE)  
xnode.atcmd('ID', ID)
```

```
if CE == 0x00:  
    xnode.atcmd('JV', 0x01)
```

```
xnode.atcmd('WR')
```

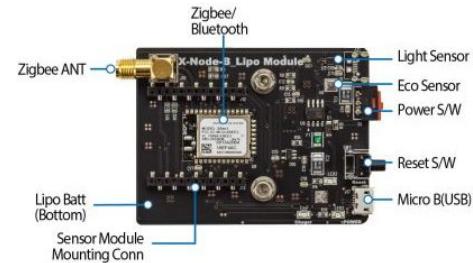
```
from pop import xnode
```

```
NI = "Rout00"  
CE = 0x00      # 0x00 or 0x01  
ID = 0x70
```

```
xnode.atcmd('NI', NI)  
xnode.atcmd('CE', CE)  
xnode.atcmd('ID', ID)
```

```
if CE == 0x00:  
    xnode.atcmd('JV', 0x01)
```

```
xnode.atcmd('WR')
```



IMU를 이용한 각도측정 (바라보는 방향)

1. serBot(서봇)에 IMU(9-axis)를 장착한 후, 서봇을 회전시켜 값을 측정함

(IMU값의 quaternion값 중 1번째와 4번째 값만 이용)



$$\text{quat}[0] : -\sin\left(\frac{x}{2}\right)$$

$$\text{quat}[3] : \cos\left(\frac{x}{2}\right)$$

IMU를 이용한 각도측정 (바라보는 방향)

2. 한바퀴에서만 바로 각도를 알아내기 위해

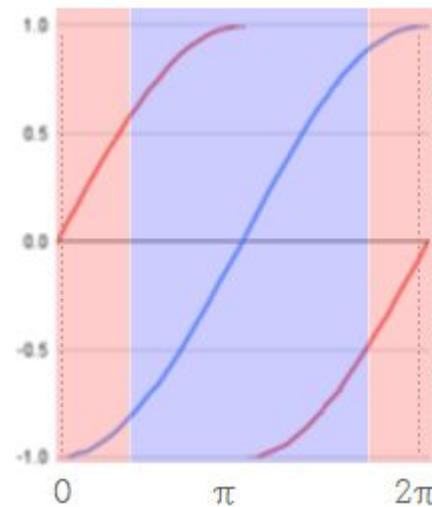
sin부분이 음수인 부분은 -1을 곱하고

대칭인 sin부분을 분리하여 구별시킴

cos부분 값의 변화량이 적은부분(양 끝쪽)은

미세한 변화를 감지하기 어려우므로

sin부분으로 대체하여 사용함.

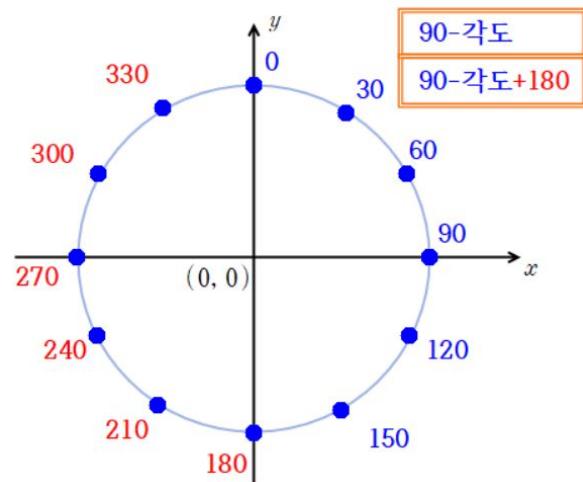


IMU를 이용한 각도측정 (바라보는 방향)

3. 서봇에 맞는 각도로 변환.

(정면 0도, 시계방향으로 360까지)

```
if moveEnd[0]==moveStart[0]:  
    if moveEnd[1] > moveStart[1]:  
        moveAngle = 0  
    else:  
        moveAngle = 180  
    else:  
        angle = int(round(np.arctan((moveEnd[1]-moveStart[1])  
            /(moveEnd[0]-moveStart[0]))*180/np.pi,1))  
        moveAngle = 90-angle if angle>180 else 270-angle
```



화면 출력 GUI - PyQT5

실내지도 그리기

1. numpy 2차 배열로 좌표를 만들고
matplotlib으로 장애물, 목적지 등의 위치를 지정하여 표시
2. 그림파일로 변환하여 저장 - 지도의 배경 이미지로 사용

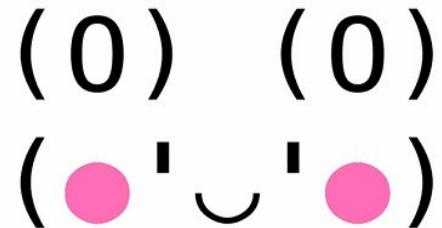


인터페이스 구상

첫화면 - 귀엽고 친근한 이모티콘 이미지로
손님을 반갑게 맞이함

인사와 안내문구를 음성으로 출력

화면전체를 투명한 버튼으로 덮어서
어딜 누르더라도 본격적인 안내화면으로 이어짐



안내를 원하시면 화면을 클릭해주세요

인터페이스 구상

화면 좌측 - 실내 지도를 표시하고

특정 지점마다 투명버튼을 만들어

화면클릭시 이동경로를 표시



← 지도에서 목적지를 클릭하세요



안내시작 ►

화면 우측 - 안내문구와 목적지 선택시 이해를 돋는 그림 출력

안내시작 - 안내음성과 함께 경로를 따라 서봇이 이동하며, 주행중에 음악이 재생되고

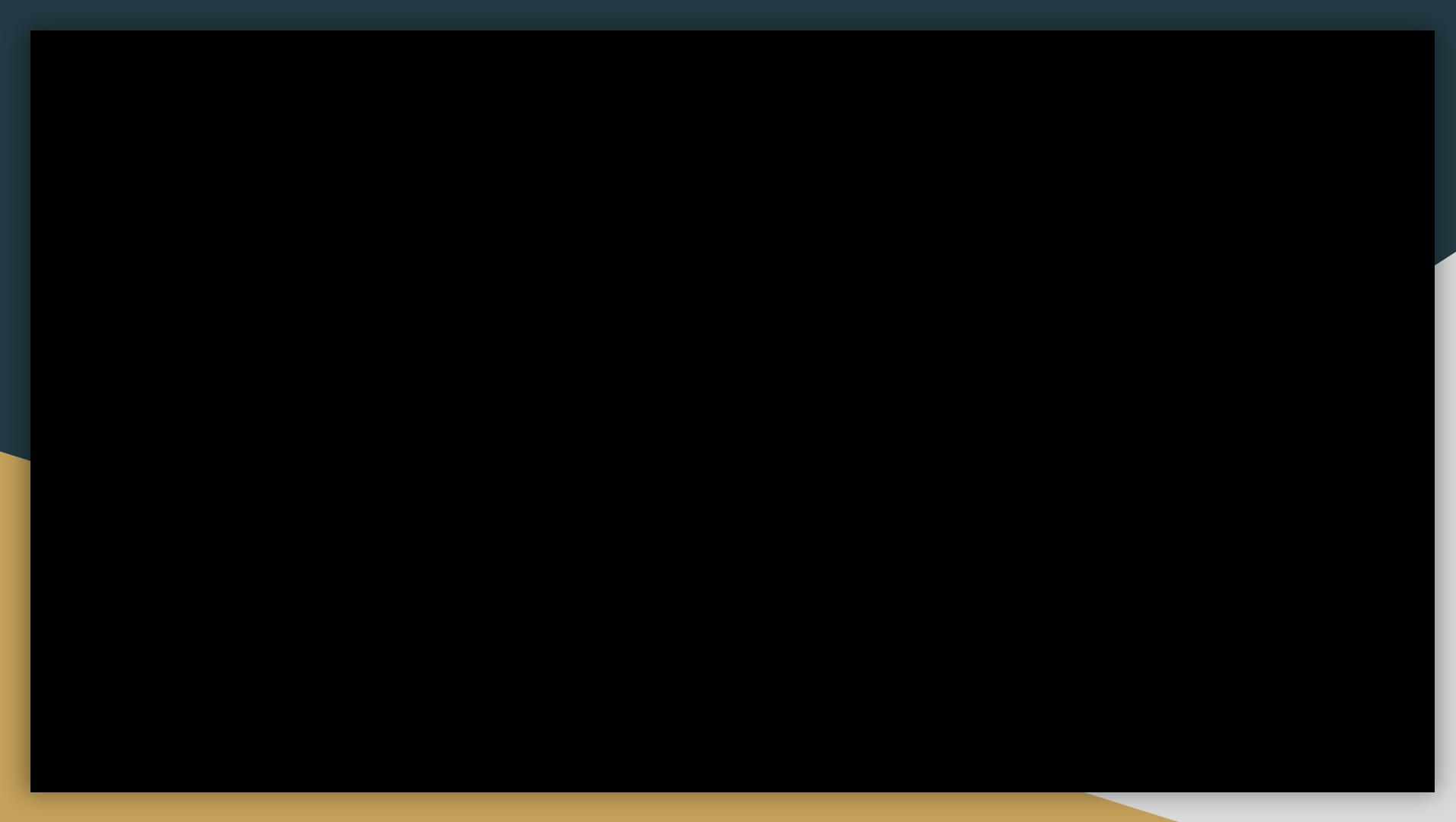
진행정도에 따른 진행도(bar)가 표시됨. 특정지점에 도착시 설명이 추가로 음성출력

화면 출력GUI - PyQt5

경로 계산(최단경로)

1. 경유지 위치지정 (각 통로)
2. 목적지를 버튼클릭으로 좌표값을 입력받음
3. 현재위치(시작점)에서 가까운 경유지1와 목적지에서 가까운 경유지2를 찾음
4. 현재위치 – 경유지1 – 경유지2 – 목적지를 연결
 - 경유지1,2가 같고, 현재위치와 목적지까지의 거리가 더 짧다면 바로 연결
5. matplotlib을 이용하여 대략적인 실내 지도를 그리고,
원하는 위치의 좌표를 알아내어 그래프로 노선을 그림
6. 실내지도 그림파일 위에 노선을 그려 최종 경로표시가 완성되고 화면에 출력





3. Audio



Google Assistant

- 구글 gTTS로 음성인식 및 텍스트 합성
- 눈블로킹을 이용한 이동시 음악재생 기능
- 사용자 장치 액션으로 음성명령으로 제어 가능

코드 소개

```
from pop import Pilot
from gtts import gTTS
from popAssist import *
import subprocess
import pyaudio
import wave
import time
```

gtts로 만든 안내음성을 mp3파일로 저장

음악은 pyaudio를 사용함으로 wave를 사용

```
def userAction(text):
    global filename
    global stream

    action = False
    print(text)
```

← 사용자 장치 정의로 다양한 액션활동 사용가능

```
if text.find("동") != -1 or text.find("대") != -1 or text.find("향") != -1 or text.find("로") != -1:  
    text = "금동대향로로 안내를 시작합니다"  
    filename = "start.mp3"  
    tts = gTTS(text, lang='ko')  
    tts.save(filename)
```

음성인식 기능 사용시 동, 대, 향 정도의 낱말만 인식이 되어도 “금동대향로”로 인식하고
안내를 시작합니다.

안내 텍스트 음성은 미리 mp3파일로 저장해두고 즉각적으로 출력하도록 설정했습니다.

```
def callback(in_data, frame_count, time_info, status):
    data = w.readframes(frame_count)
    return (data, pyaudio.paContinue)

w = wave.open("12.wav", "rb")
p = pyaudio.PyAudio()

stream = p.open(format=p.get_format_from_width(w.getsampwidth()),
                 channels=w.getnchannels(),
                 rate=w.getframerate(),
                 output=True,
                 stream_callback=callback)
```

-논블로킹 모드로 로봇이 동작하면서도 배경음악 사용

-구글 어시스턴트 가능

textList[0] = "무령왕 금제관식은 공주시 무령왕릉에서 출토된 백제 시대의 금으로 만든 왕관 꾸미개 한쌍이다. 1974년 7월 9일 대한민국의 국보 제154호로 지정되었다."

textList[1] = "무령왕 금귀걸이는 충청남도 공주시 무령왕릉에서 출토된 백제시대의 금 귀고리 한 쌍으로 국립공주박물관에 소장되어 있다. 1974년 7월 9일 대한민국의 국보 제156호로 지정되었다."

textList[2] = """백제 금동대향로는 백제에서 만들어진 금동 향로이다. 1993년 12월 12일 부여군 능산리 절터의 목곽 수로 안에서 발견되었으며 국보 제287호로 지정되었다."""

textList[3] = "금제뒤꽂이는 충청남도 공주시 금성동 송산리 고분군에 위치한 무령왕릉에서 1971년 여러 유물들과 함께 출토된 무령왕의 금제 뒤꽂이 한점이 발견되었다. 현재 국립공주박물관에 소장 중이며, 대한민국 국보 제159호로 지정되어 있다"

textList[4] = """백제의 제25대 국왕이자 건길지. 삼국사기에 따르면 키가 8척이고 눈매가 그림과 같았으며 인망이 두터웠다고 한다. 키가 8척이면 한척으로도 190cm라는 뜻인데, 무령왕릉에서는 무령왕의 유해로 추정되는 뼛조

텍스트는 리스트로 만들어 미리

mp3파일로 저장해서 사용

향후 과제

- 마이크의 잡음값을 낮춰서 더 정확한 음성인식 구현
- 구글 Assist push talk 등 다양한 응용프로그램 사용
- soundMeter 등 소음값을 측정한 여러가지 액션 기능
- 사용자 편의를 조금 더 생각하는 오디오

서봇에서 사용된 음악파일

멘델스 존 = 봄의 노래

Good Evening Narvik

베토벤 = 베토벤 바이러스

오나라 (대장금OST)

에릭사티 = 짐노페디 1번

Stepping on the Rainy Street

슈베르트 = 송어

레드제플린 - Stairway to Heaven(Piano Cover)

Departure

출처:



Photo - Zone

1 개발 환경 구성
| Hardware , Software

2 Photo zone
| OpenCv, Pytorch 과정

3 코드설명
| 영상 및 사진

개발 환경구성

- **Hardware**

- Nvidia Jetson Xavier AGX Board
- Raspberry Pi Camera

- **Software**

- jetpack 4.4
- python 3.6
- torch 1.7.0
- torchvision 0.8.0
- opencv

SerBot 기념사진 촬영(포토존)

OpenCV 를 활용한 사진촬영 및 합성

포토존에서 개인 또는 단체사진 촬영

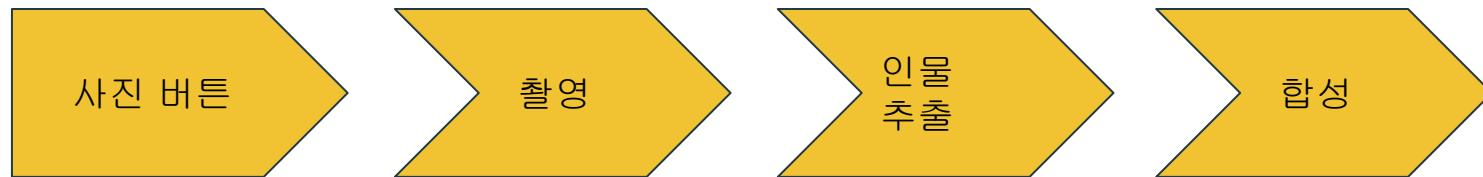
사진촬영용 Serbot이 포토존에서 사진 촬영 대기

촬영버튼을 누르면 잠시 뒤에 촬영

촬영 후 인물만 추출하여 왕릉 배경에 합성



과정 설명





원본 사진



인물 추출



인물 + 배경
합성결과



코드 설명

```
def __init__(self):
    super().__init__()

    self.setWindowTitle("picture")

    self.pictureBt = QPushButton("촬영",self)
    self.pictureBt.move(0,0)
    self.pictureBt.resize(1280,720)
    self.pictureBt.clicked.connect(self.capture)
    #self.exitBt.setFont(QFont('MesloLGS NF', 55))
    #self.pictureBt.setStyleSheet("background:transparent; border:0px")

    self.exitBt = QPushButton("X",self)
    self.exitBt.move(1180,0)
    self.exitBt.resize(100,100)
    self.exitBt.clicked.connect(self.onExitBtClicked)
    self.exitBt.setStyleSheet("background:transparent; border:0px; color:#F0F0F0")
    #self.exitBt.setFont(QFont('MesloLGS NF', 15))
    self.exitBt.raise_() # 버튼 높이 위로가기
```

UI 화면 코드

```
def capture(self):
    #
    time.sleep(5000)
    cam = Camera(width=1024,height=768)
    cv2.imwrite("pic{}.png".format(count), cam.value)
    print("Complete! count = {}".format(count))
```

카메라 촬영

torchvision을 이용한 인물 추출 과정

```
def deepLabV3(self):
    if count < 0: # 파일이 없는 경우
        print("파일이 없습니다")
        return

    input_image = Image.open("pic{}.png".format(count))
    preprocess = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])

    input_tensor = preprocess(input_image)
    input_batch = input_tensor.unsqueeze(0) #create a mini-batch as expected by the model

    input_batch = input_batch.to('cuda')
    self.model.to('cuda')

    with torch.no_grad():
        output = self.model(input_batch)[‘out’][0]
    output_predictions = output.argmax(0)

    palette = torch.tensor([2 ** 25 - 1, 2 ** 15 - 1, 2 ** 21 - 1])
    colors = torch.as_tensor([i for i in range(21)])[:, None] * palette
    colors = (colors % 255).numpy().astype("uint8")

    r = Image.fromarray(output_predictions.byte().cpu().numpy()).resize(input_image.size)
    r.putpalette(colors) #

    r = ImageQt(r)

    r.save("makeMask.png", 'PNG')

    pixmap = QPixmap.fromImage(r) #
    self.frame.setPixmap(pixmap) #

    self.exitBt.raise_()
    self.combine()
```



합성 후 저장, 화면에 띄움

```
def combine(self):
    src = cv2.imread('pic{}.png'.format(count)) # 원본사진
    mask = cv2.imread('makeMask.png') # 마스크 이미지 - imgcut.py
    dst = cv2.imread('tomb.png') # 배경1
    dst2 = cv2.imread('crown.png') # 배경2

    # 사이즈 변경
    src = cv2.resize(src, (1024, 576))
    mask = cv2.resize(mask, (1024, 576))
    dst = cv2.resize(dst, (1024, 576))
    dst2 = cv2.resize(dst2, (1024, 576))

    cv2.copyTo(src, mask, dst)

    frame = cv2.copyTo(src, mask, dst)

    cv2.imwrite('completed/pic_{}.jpg'.format(count), frame)
    print('complete')

    cv2.imshow('dst', dst)

    self.countUp() # 파일이름 구분용 카운트 Up

    cv2.waitKey(5000) # 5초 대기
    cv2.destroyAllWindows()
```

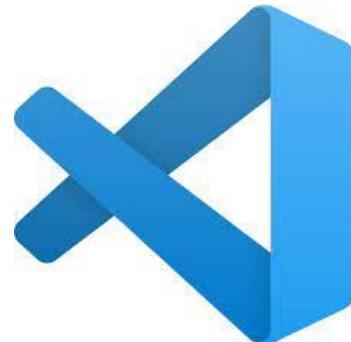
Mask-detection

- 1** 개발 환경
| Serbot
- 2** 개발 환경 구성
| Hardware , Software
- 3** Photo zone
| 모델 및 데이터셋
- 4** 결과
| 영상 및 사진
- 5** 느낀점
| refactory

개발환경



ubuntu



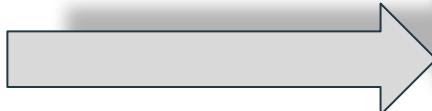
개발 환경구성

- **Hardware**

- Nvidia Jetson Xavier AGX Board
- Raspberry Pi Camera

- **Software**

- jetpack 4.4
- python 3.6
- torch 1.7.0
- torchvision 0.8.0



YOLOv5

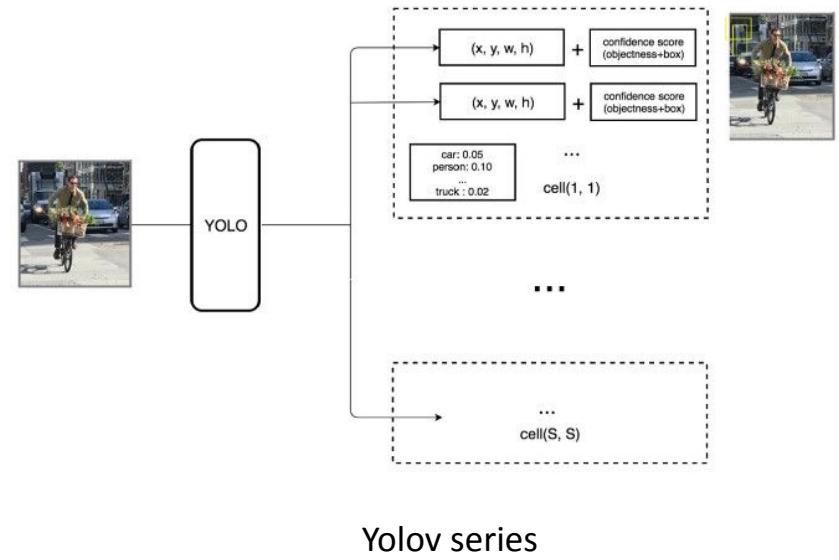
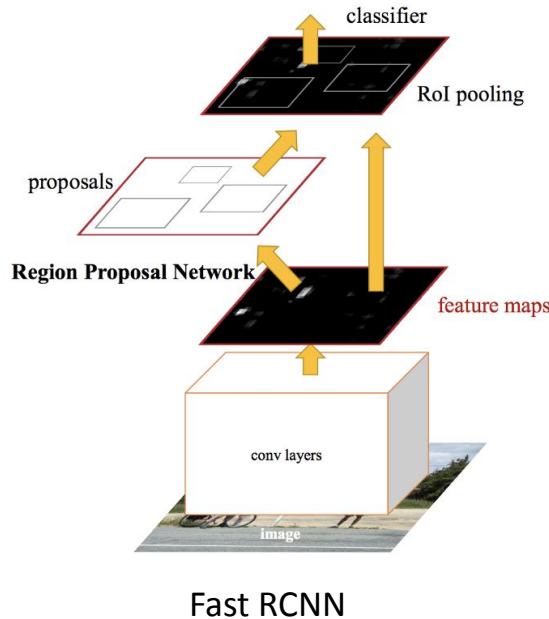


Download on the
App Store

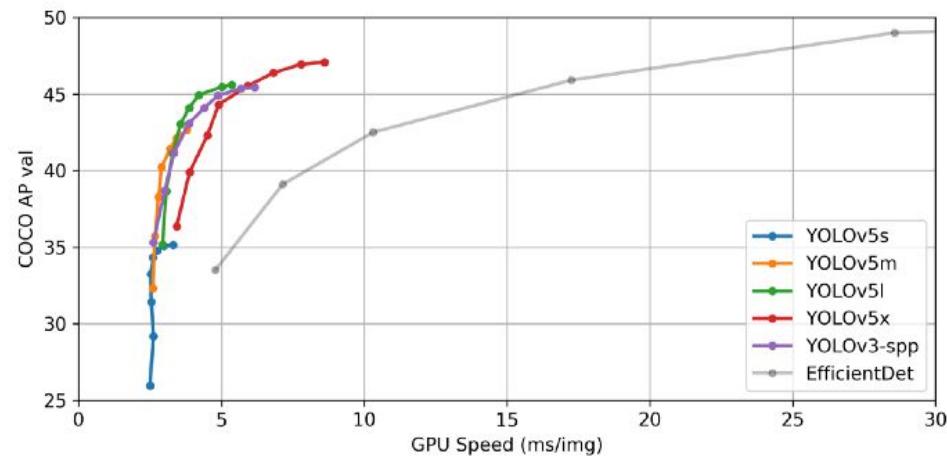
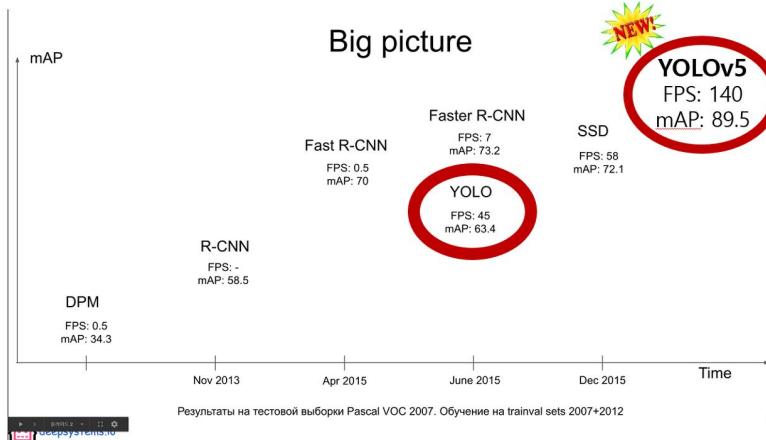


Coming Soon on
Google Play

YOLOv5 | yolov5를 선정한 이유



YOLOv5 | yolov5를 선정한 이유



YOLOv5 | yolov5 커스텀 훈련방법

1. Dataset 준비

1. <https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset.git>
2. <https://github.com/UniversalDataTool/coronavirus-mask-image-dataset.git>
3. <https://github.com/iAmEthanMai/mask-detection-dataset.git>



YOLOv5 | yolov5 커스텀 훈련방법

2.requirement.txt를 통한필요 환경설정

```
numpy==1.19
scipy==1.4.1
cudatoolkit==10.2.89
opencv-python
torch==1.7
torchvision==0.8.0
matplotlib
pycocotools
tqdm
pillow
tensorboard
pyyaml
```



Serbot 환경(리눅스)에서는
numpy 1.19.3 이하로만 설치해야함

주목해야 할 중요한 점은
PyTorch 버전 \geq 1.5, Python 버전 3.6 및
CUDA 버전 10.2가 필요함

YOLOv5 | yolov5 커스텀 훈련방법

3.Yaml 파일을 통하여 클래스명 파일위치 설정

```
import yaml

with open('/content/yolov5_mask/data.yaml', 'r') as f:
    data = yaml.load(f)

print(data)

data['train'] = '/content/yolov5_mask/'
data['test'] = '/content/yolov5_mask/'
data['val'] = '/content/yolov5_mask/'

with open('/content/yolov5_mask/data.yaml', 'w') as f:
    yaml.dump(data, f)

print(data)
```

YOLOv5 | yolov5 커스텀 훈련방법

4. 훈련시작

```
python train.py --img 640 --batch 8 --epochs 30 --data  
./data/mask-detection.yaml --cfg ./models/yolov5s.yaml  
--weights '' --device 0
```

			
Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
14 MB _{FP16} 2.2 ms _{V100} 36.8 mAP _{COCO}	41 MB _{FP16} 2.9 ms _{V100} 44.5 mAP _{COCO}	90 MB _{FP16} 3.8 ms _{V100} 48.1 mAP _{COCO}	168 MB _{FP16} 6.0 ms _{V100} 50.1 mAP _{COCO}

– img: 이미지 사이즈 설정

– batch: 연산한번에 들어가는 크기를 말함

– epochs: 학습 데이터 모델을 통과한 횟수

– data: 3단계와 같이 파라미터들을 설정하는 yaml 파일 설정

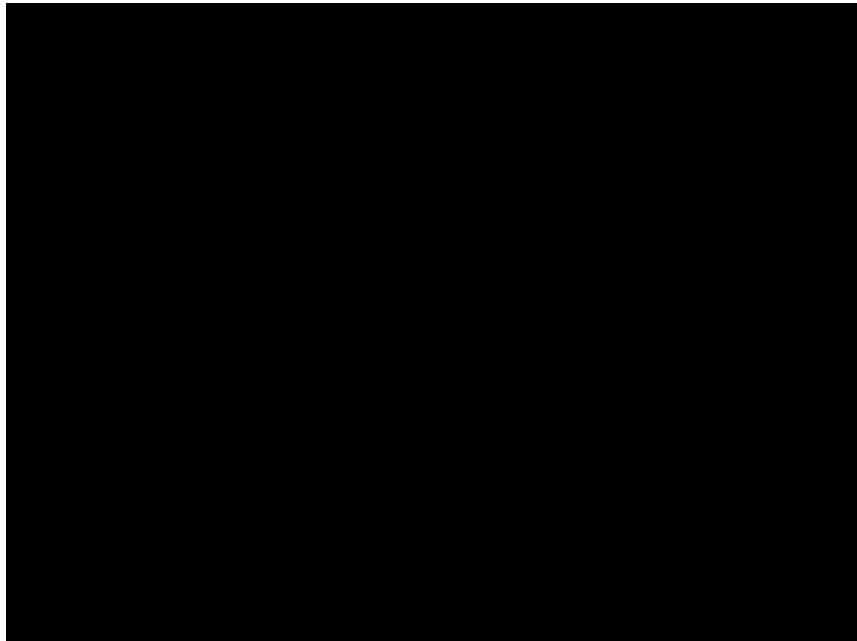
– cfg: yolov5 안에서 여러가지 모델중 설정하는 방법 (config)

– weights: 전이학습 할 모델설정(가중치)

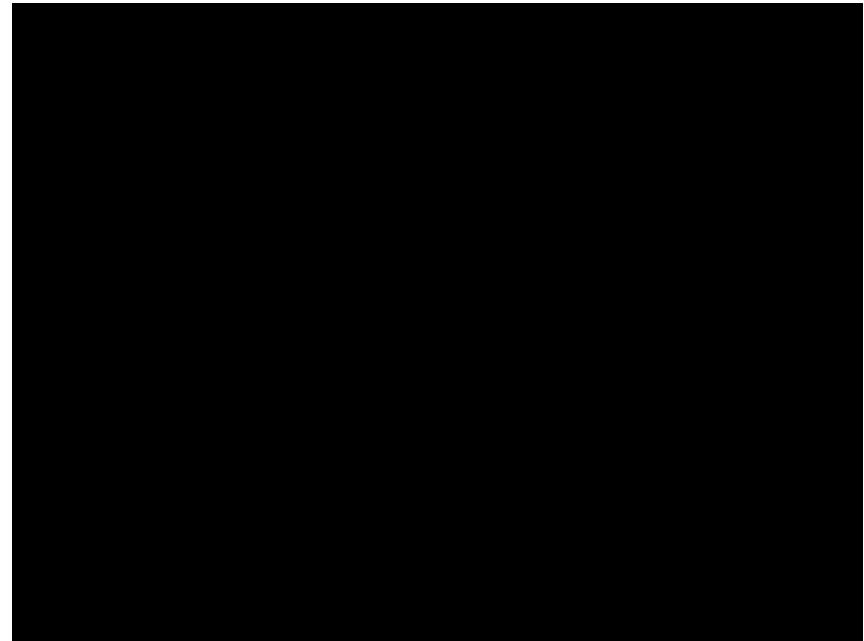
– device: gpu 혹은 cpu 중 무엇을 선택해서 훈련을 진행할지 선택

적용 모습

| colab을 통해 영상에 적용 (예제)



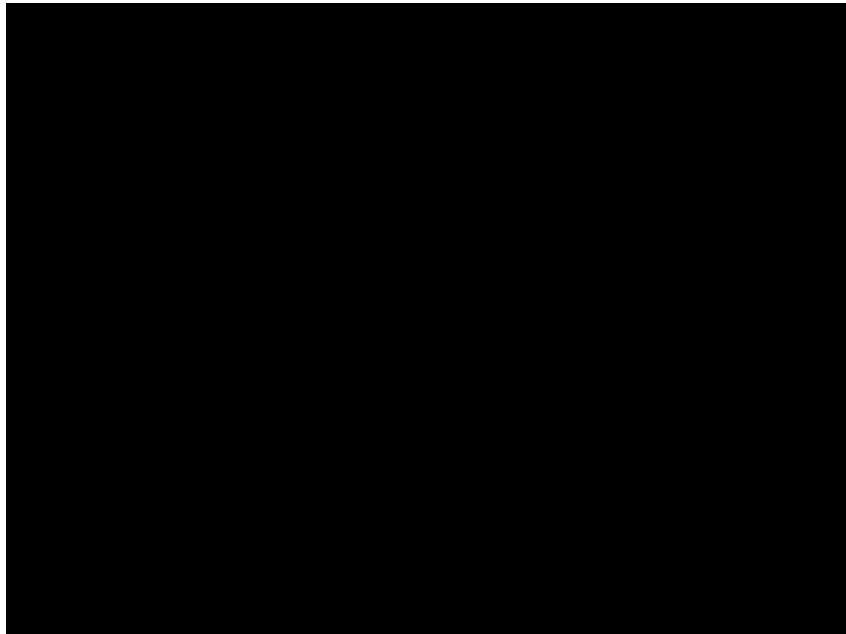
[원본]



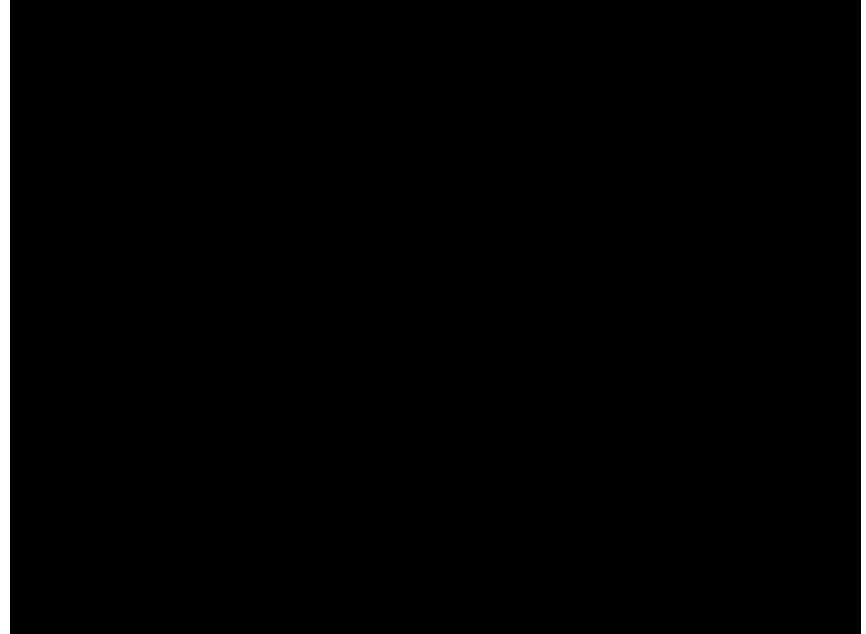
[적용한 후 영상]

적용 모습

| Serbot 적용



[원본]



[적용한 후 영상]

개선방안 | 완성도

[목표]

1. 구동시에 마스크를 안쓴 인원에게
다가가 경고할수있는 기능
2. 어느 각도에서도 마스크를 착용
여부를 확인 및 높은 정확도

[현재 상태]

1. 구동시에 마스크를 안쓴 인원에게 다가가
경고할수있는 기능

>>>> 구현 불가
2. 어느 각도에서도 마스크를 착용 여부를 확인
및 높은 정확도

>>>> 더 많은 학습훈련 필요

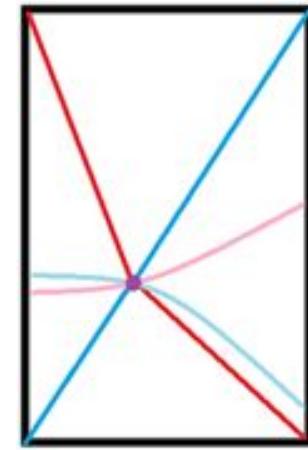
아쉬운 점 - ZigBee (위치측정)

zigbee의 신호세기를 이용한 현재 위치 측정

- 실내(직사각형)의 각 꼭지점에 zigbee를 설치한 후, 신호 세기를 측정
- 신호 세기는 측정 장소까지의 거리와 비례한다고 예상하여
마주보는 대각선끼리의 비율을 이용하여 현재 위치(좌표)를 추정함

한계점

- 신호를 전달하는 과정에서 **장애물**의 영향이 큼
- zigbee 기기마다 **신호세기가 다름.**
- 배터리의 영향도 큰데, 오랜 시간 **전원 공급의 한계**가 있음
- 측정 결과 거리에 따라 신호 세기의 변화량이 미미하여 **정확한 위치 측정이 어려움**



아쉬운 점 - PyQt, ZigBee, gAssist, LiDAR

실시간 화면출력의 제한

- 노선을 따라 서봇이미지가 이동하는 모습을 출력하고 싶었지만
실시간으로 PyQt의 이미지 이동이 원활히 작동되지 않음

serBot → zigbee 송신

- serBot에서 zigbee신호를 수신하는 것은 가능했지만, 송신하는 부분이 미흡함

구글 어시스턴트, LiDAR 사용불가

- gAssistant가 코드 합친 후 작동되지 않음
- LiDAR가 간혹 동작이 되지 않을 때가 있음



아쉬운 점 - CAM

Photo -zone, Mask -detection

- 가까울수록 가운데 물체가 크고 넓게 보이는 현상이 있음
- 배경이 복잡한 경우 다른 물체도 같이 인식되는 경우가 많음
- 원인을 알수없는 카메라 연결끊김이 자주 발생함
- Yolo 사용이 원활하지 않음

**THANK
YOU!**



취업하자! 취업하자!



대한상공회의소
충남인력개발원