K-Digital Training

[두산로보틱스] 지능형 로보틱스 엔지니어

밤에도 쏠수있어!

필승! **D-3 조(삼인용)**

이강인(팀장), 주진, 최순일, 최재형(탈영병)

이충현 강사님(mentor), 김재권 조교(mentor)

K-Digital Training

목차

국내 사격장 총기 사고

<민간 사격장 사망 사고>

1. 2025년 12월 (인천 송도): 20대 남성이 민간 실탄사격장에서 자신이 사용하던 권총으로 스스로 실탄을 쏴 사망한 사건
2. 2018년 9월 (서울 명동): 30대 남성이 명동의 한 실탄사격장에서 권총으로 극단적 선택

2015년 사격연습장 총기사고

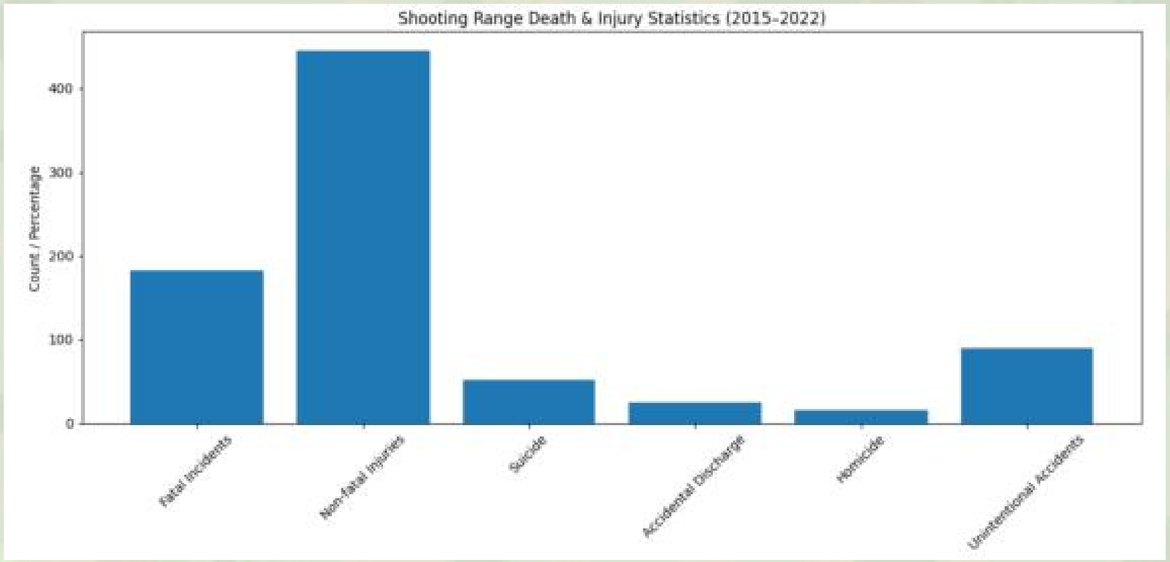
군/경 사격장 총기 사고

1. 2025년 4월 (부산 경찰청): 기동대 소속 20대 순경이 사격 훈련 중 총기 오발 사고로 사망했습니다.

2. 2025년 12월 (파주 JSA): 공동경비구역 사격 훈련장에서 권총 오발 사고로 병사 1명이 부상을 입었습니다.

미국 총기 사고

미국 사격장 내 사고



미국 내 576개 사격 시설에서 발생한 총기 관련 사망 및 부상 사고

<출처 : Henson-Garcia M, et al. Inj Prev 2025;31:253–256. doi:10.1136/ip-2023-045127 253>

국내외 사격장 내 사고 주요원인

→ 안전관리 미흡·관리 부실

사격장 내 사고 예방

사격장 내 총구 방향 확인

탄피 분실 방지

총기 사고 예방

로봇 부사수 효과

현행 안전 요원 2인에서 인력 감소 효과→창업요건 완화가능

엄격한 실시간 통제 가능→안전한 사격 환경 인식 확대 가능

프로젝트 팀 구성 및 역할

팀 구성

대대장 이강인

팀장

PM, 음성명령 인식&YOLO기반 PICKANDMOVE
개발, IR 카메라 뎁스 안정화, 안전 복구 구현

중대장 주진

UI 개발, 노드구성

디텍션 모델 선정, UI 개발 & 통합(음성인식)
개발 & 통합 & 안정화 & 설계 & MINI PROJECT

탈영병 최재형

아두이노 개발, 로봇 좌표

아두이노로 총소리 구현 후 탈영

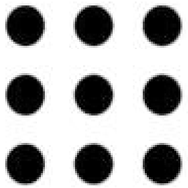
주임원사 최순일

이미지 디텍팅

OPENCV, YOLO-SEGMENT 활용한 이미지 디텍트

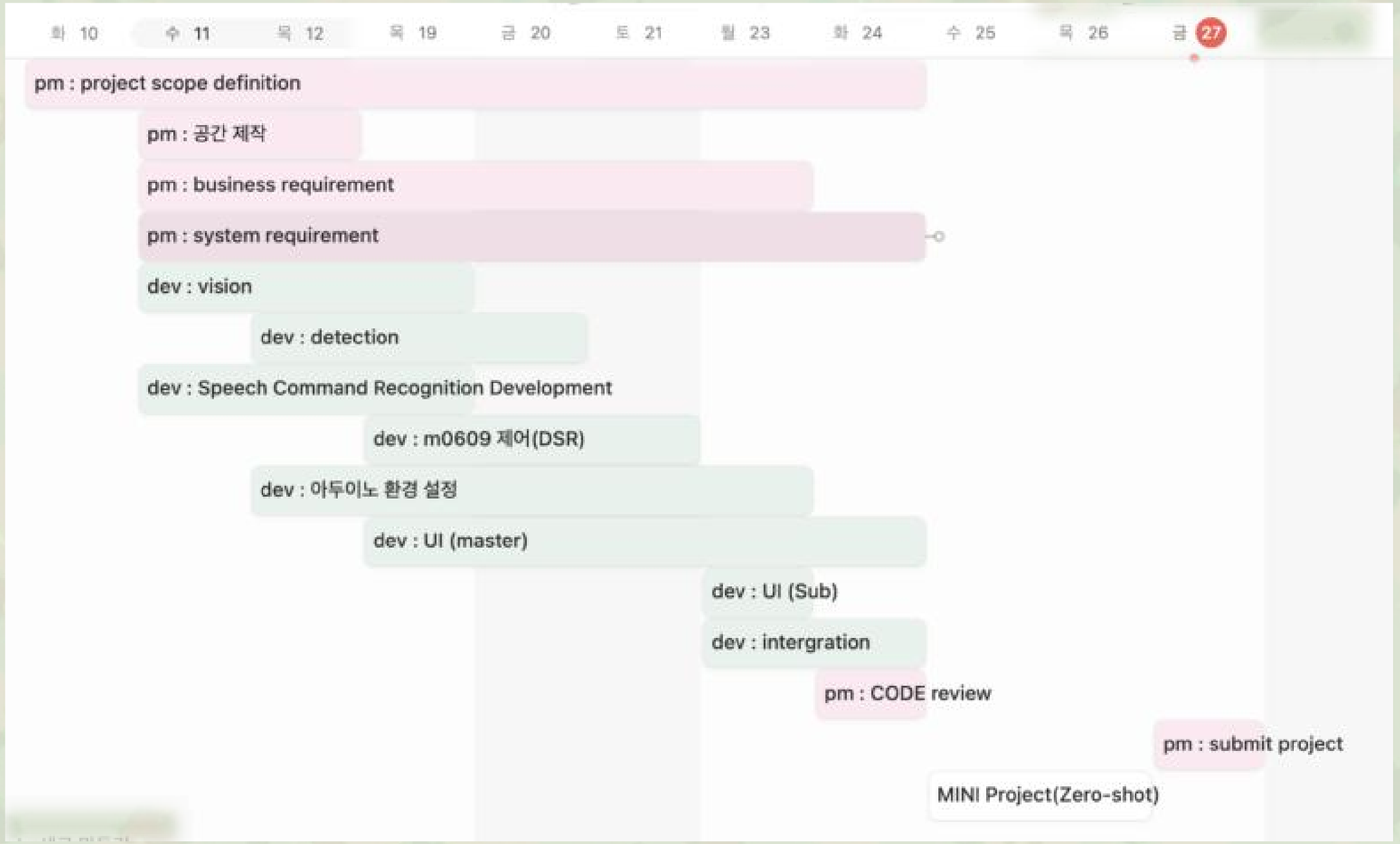
프로젝트 팀 구성 및 역할

환경구성

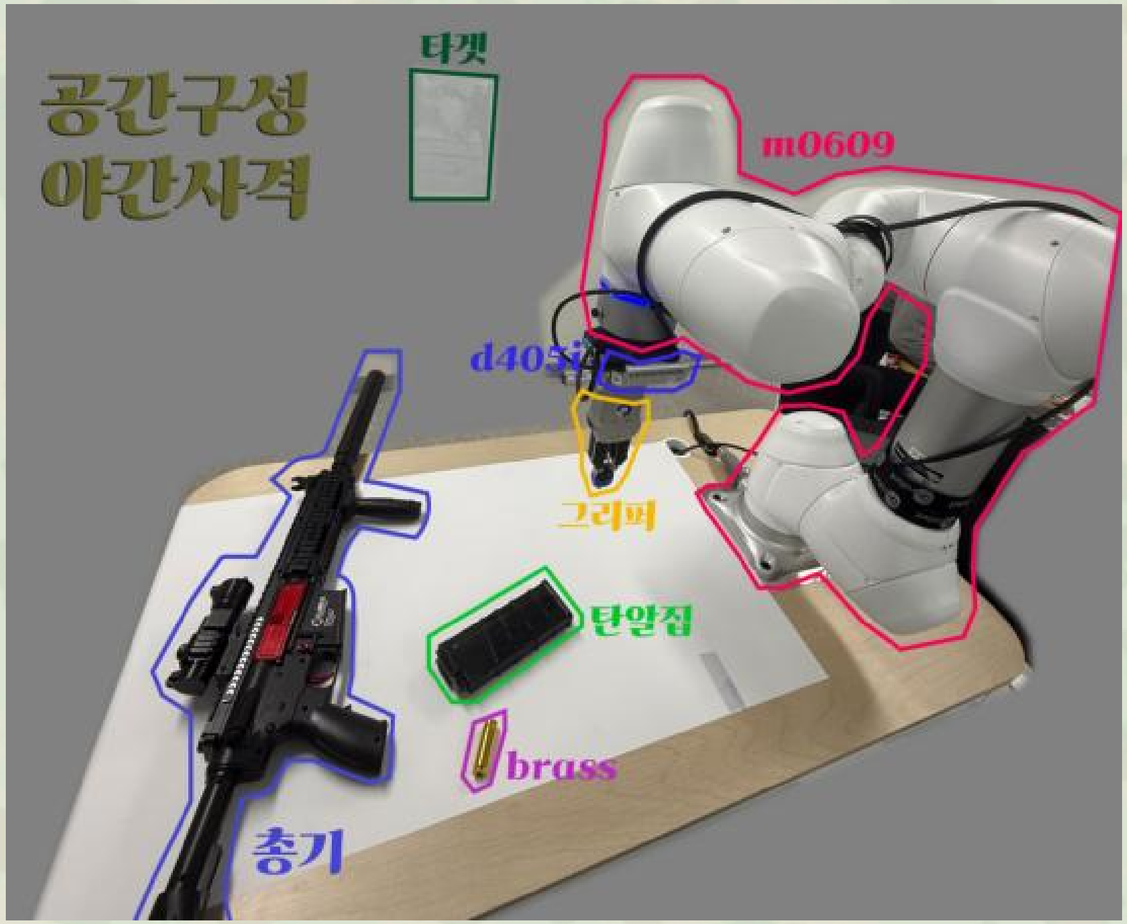
H/W					
	M0609(Doosan)	Controller	Gripper(onrobot)	D435i (intel)	아두이노
S/W					
	Ubuntu 22.04	ROS2 Humble	Python 3.1x	Open CV	Flask
					
	Doosan DRL	JAVA Script	HTML	Notion	cvat

프로젝트 팀 구성 및 역할

Schedule



공간구성



야간사격 공간 및 UI

:UI 내 음성인식으로 탄알집 인계, 및 타겟 탄 개수 집계

Flow Chart

사수입장 / 중대장 명령 "탄알집 인계"

로봇팔 탄알집 인계

중대장 명령 "사격 개시"

사수 7발 사격/ 로봇팔 과녁 응시

기능 고장

사격 종료 / 로봇팔 종료 수신호

중대장 명령 "탄알집 회수"

사수로부터 탄알집 수령 후 수거함에 투척

중대장 명령 "탄피 확인"

탄피확인/ 1차탐지위치이동

if 탄피갯수 < 7

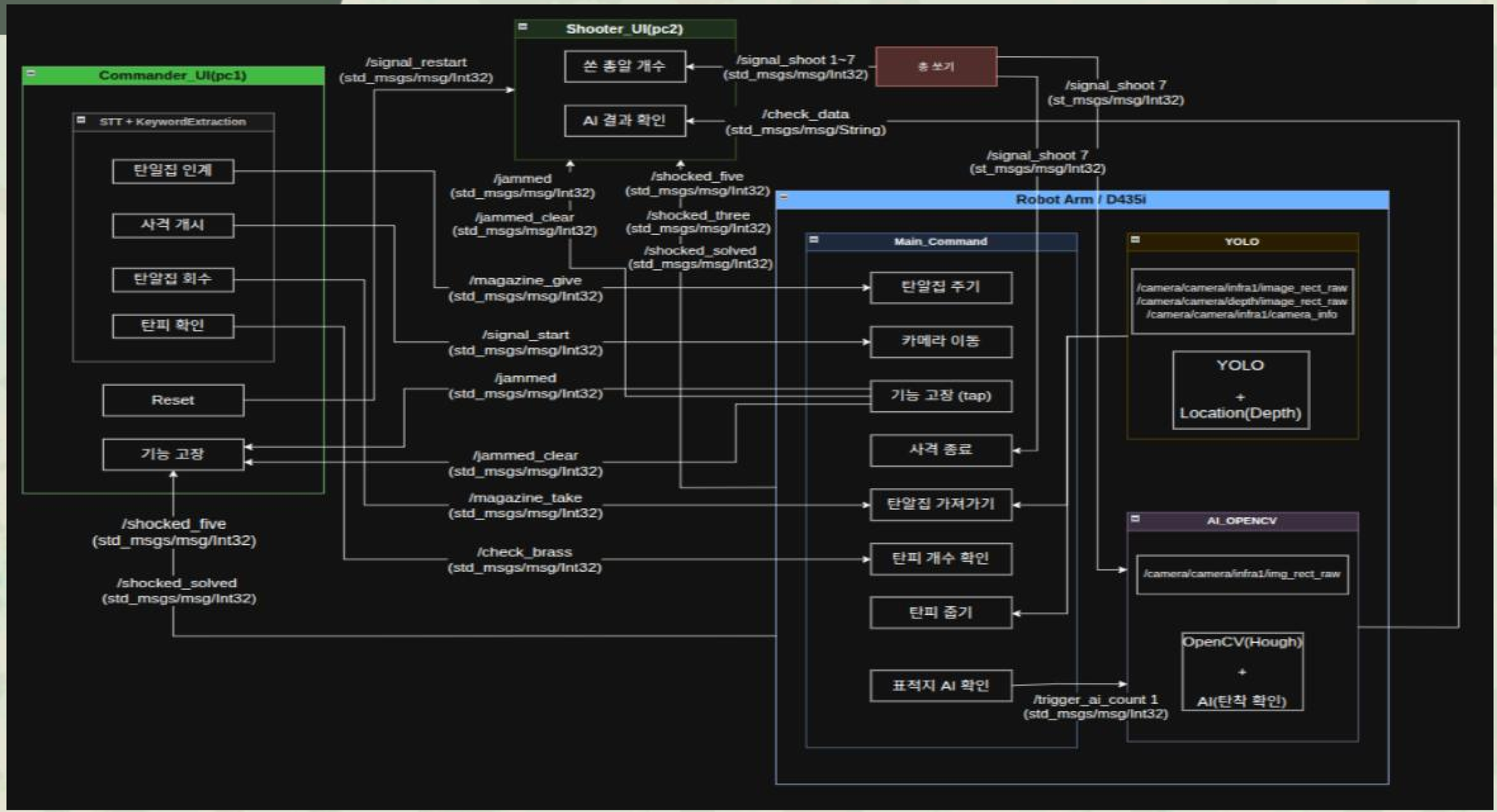
YES

2차 탄피 확인 및 회수

사수결과확인

NO

System Requirement



Node Architecture

PC 1

RobotControlNode
CommanderWebNode
ai_count_node
YOLO_node

/check_data
/signal_restart

/shocked_three
/shocked_five
/shocked_solved
/jammed
/jammed_clear

/signal_start
/magazine_take
/magazine_give
/check_brass

PC 2

shooter_UI

/shocked_three
/shocked_five
/shocked_solved
/jammed
/jammed_clear

TroubleShooting(1)

YOLO 학습(개선 전)

YOLO Labeling
1. 환경 a. index(class) : 0, brass, 1, magazine, 2, bullet b. bullet은 opencv를 이용하였음 c. brass, magazine : 바운딩박스 이용(labelImg) d. yolov8n 사용 2. 결과 a. brass가 모여있을때 하나의 객체로 인식 b. gripper 및 로봇 전원선을 magazine을 인식하는 오류

OBB, Segmentation 프로그램 선정 : cvat

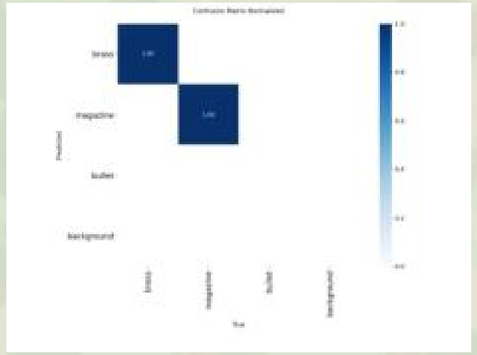
robotflow는 웹기반, cvat는 localhost기반으로 정밀마스크에서 우수한 cvat 선정

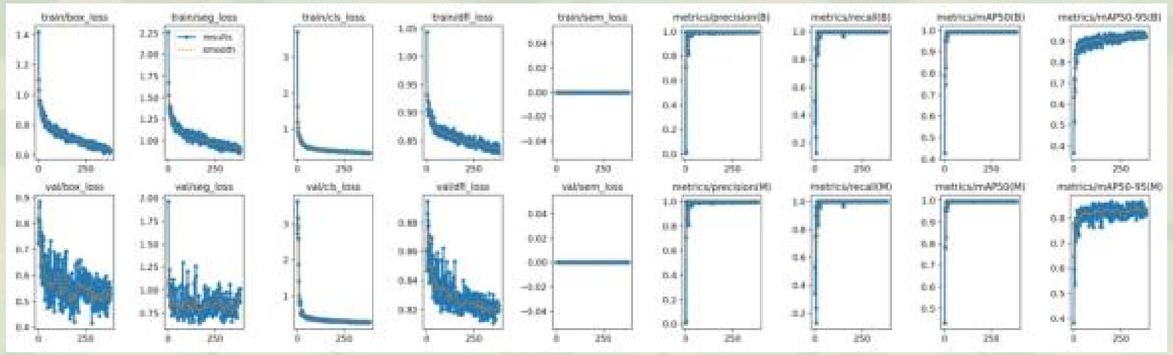
TroubleShooting(1)

YOLO 학습(개선 후)

YOLOv8n-Seg Labeling

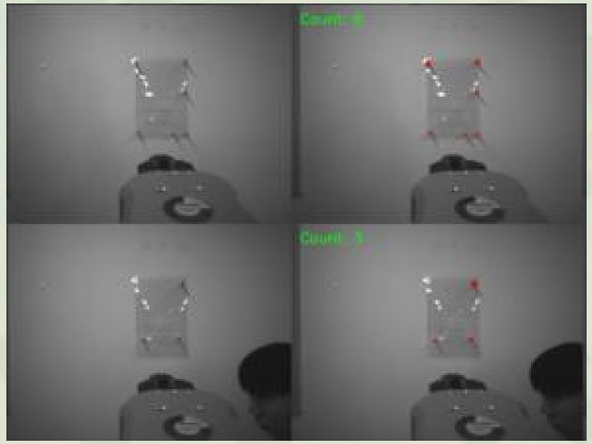
1. 환경
   a. cvat에서 brass는 OBB(reactangle), magazine은 Segmentation으로 annotation함
   b. export시 COCO Dataset -> json파일 -> 파이썬에서 코드분류(import_json.py)
   c. yolov8n-seg
2. 결과
   a. gripper등 네모난 검은 박스를 전부 매거진으로 인식하는 오류 개선
   b. 모여있는 탄피를 각각의 object로 인식





TroubleShooting(1)

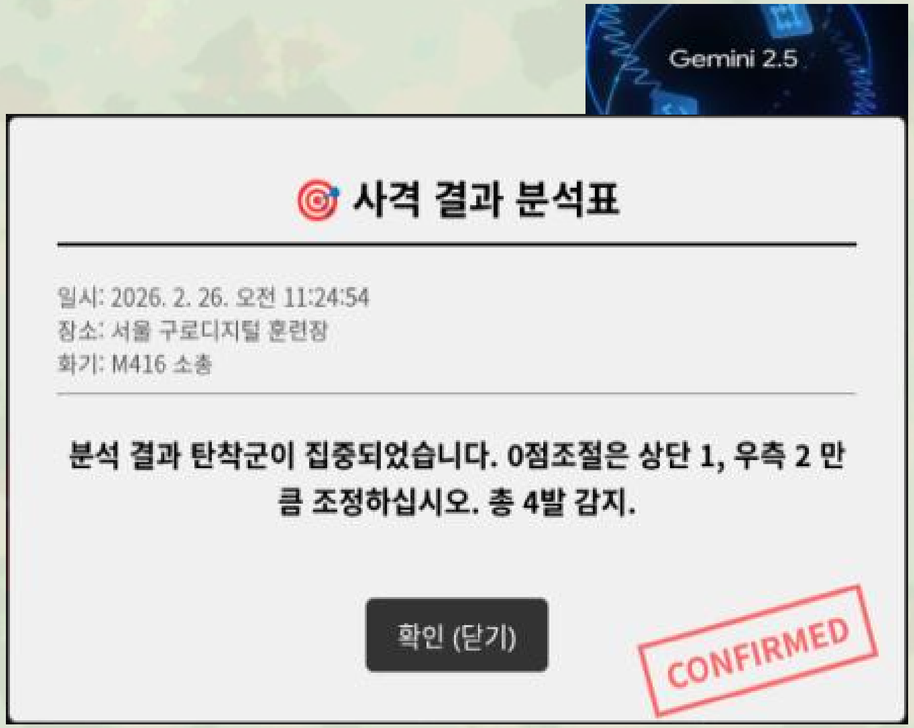
Bullet (OpenCV) 개선 전



OpenCV
1. parameter a. roi 값 지정으로 표적지바운더리 지정 b. GausianBlur(roi,(5,5),0) c. contourArea : 5<area<200 d. circle < 0.2 e. threshold : 80 2. 결과 a. 표적지 외의 이미지에서 발생된 오류로 인해 roi값을 지정하여 표적지의 위치값을 지정 b. 그림자의 겹침에서 오는 탄착군 탐지 오류를 잡기 위해 가우시안블러, 원의 모양, 크기를 따로 조정함 c. 이미지의 흑백 이진화를 위해 threshold 값 조정으로 80 지정

yolov8n에서 약 20%, opencv 파라미터값 조정전 약 50%에서 약 85% 이상의 이미지에서 정확한 탄착군 및 개수를 탐지함.

Bullet 최종 해결책 : VLM 사용



Solution

Vision Launguage Model 활용하여 이미지 직접 분석

VLM 핵심 Code

```
final_message = self.analyze_target_with_gemini(cv_image_bgr)

 def analyze_target_with_gemini(self, img_bgr):
        # 알아낸 최신 모델명(self.active_model)으로 바로 통신!
        self.get_logger().info(f"🧠 [{self.active_model}] 통신 중... (약 3~5초 소요)")

        _, buffer = cv2.imencode('.jpg', img_bgr)
        img_base64 = base64.b64encode(buffer).decode('utf-8')

        prompt_text = """
        당신은 대한민국 국군 사격장의 사격 통제관이자 AI 표적지 분석 전문가입니다.
        주어진 이미지는 흰 벽에 붙어있는 A4 용지에 장난감 너프건 총알들이 박혀 있는 InfraRed 이미지입니다.
        다음 기준에 따라 분석 결과를 한국어로 작성해주세요.

        1. 표적지에 붙어있는 장난감 총알의 개수를 정확히 세어주세요.
        2. 총알들이 한 곳에 모여있는지(집중), 넓게 퍼져있는지(분산) 판단하세요.
        3. 만약 집중되어 있다면, 표적지 정중앙(X텐)을 기준으로 탄착군이 어느 방향(상/하, 좌/우)으로 치우쳐 있는지 파악하고,
           영점 조절을 위해 클리크 수(1~3)를 제안하세요. (예: 하단 1, 우측 2)

        [출력 규칙 - 기존 시스템 호환성을 위해 반드시 아래 3가지 문장 형식 중 하나로만 답변해야 합니다. 다른 부연 설명은 절대 금지합니다.]

        형식 1 (집중 시): "분석 결과 탄착군이 집중되었습니다. 0점조절은 [상/하단] [숫자], [좌/우측] [숫자] 만큼 조정하십시오. 총 [개수]발 감지."
        형식 2 (분산 시): "분석 결과 탄착군이 넓게 분산되었습니다. 호흡 통제 후 재사격을 권장합니다. 총 [개수]발 감지."
        형식 3 (탄착 없음): "분석 결과 탄착이 감지되지 않았습니다."
        """
        url = f"https://generativelanguage.googleapis.com/v1beta/{self.active_model}:generateContent?key={MY_GEMINI_API_KEY}"
```

TroubleShooting(2)

D435i 하드웨어 특성 기반 야간 사격 시스템 최적화

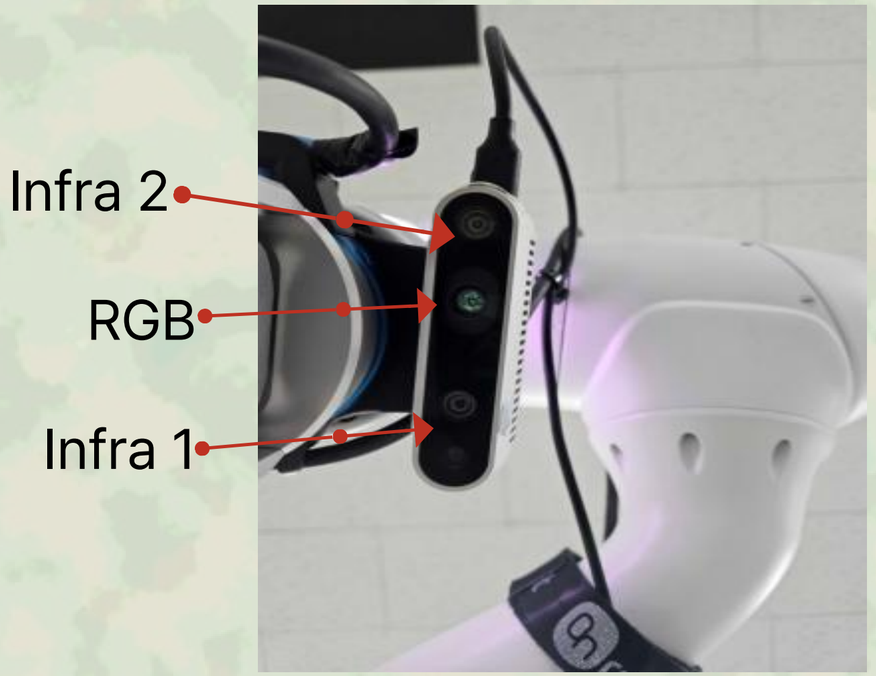
Active Stereo 방식은 살리되, YOLO 를 위해 Laser Point 15% 로 조정

TroubleShooting(2)

D435i 하드웨어 특성 기반 야간 사격 시스템 최적화

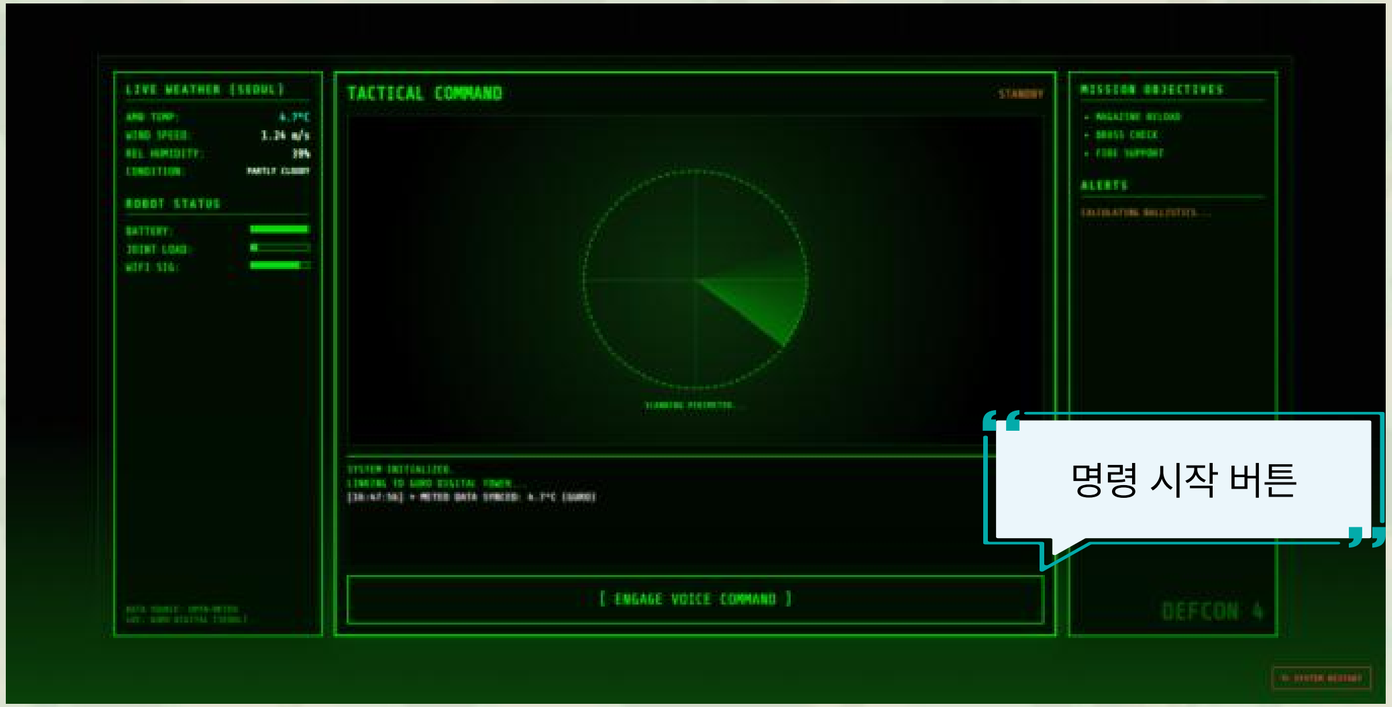
센서 간 물리적 이격(Baseline)으로 인한
3D 좌표 매칭 실패

불필요한 소프트웨어적 정렬
(align_depth_to_color)을 제거



Infra1 원본 이미지 토픽(infra1/image_rect_raw)과 순수 Depth 토픽
(depth/image_rect_raw)을 직접 매칭

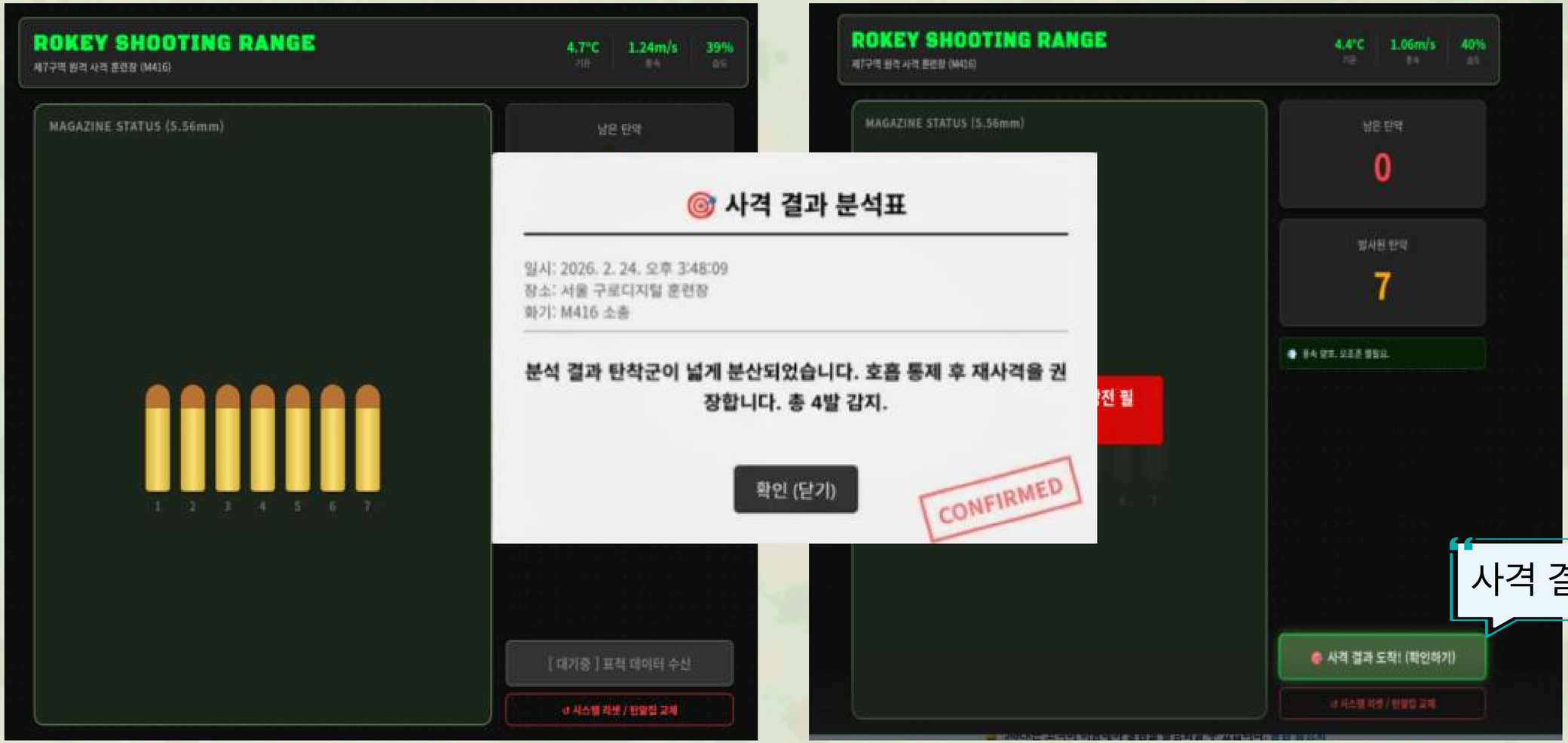
COMMANDER UI 제작



명령

1. 탄알집 인계
2. 사격 개시
3. 탄알집 회수
4. 탄피 확인

SHOOTER UI 제작



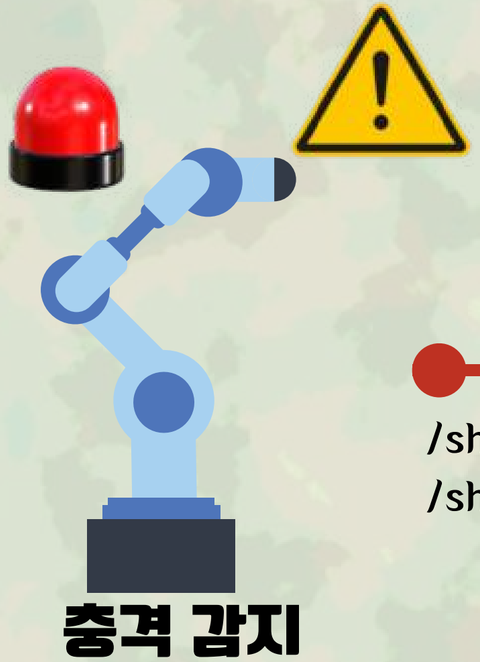
사격 결과 평가

안전복구, 비상정지

안전복구	상태	토픽명	원인
노란불	5	/shocked_five	장비고장

비상정지	상태	토픽명	원인
노란불->적색불	3	/shocked_three	충격

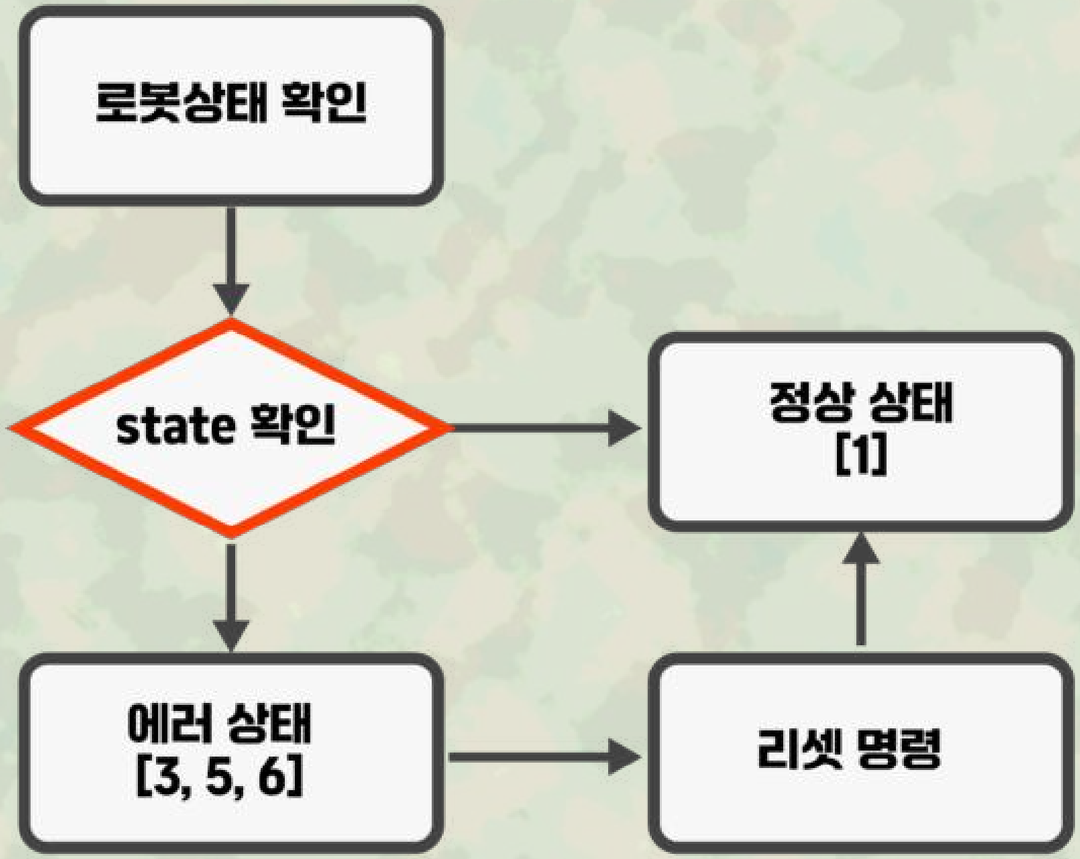
안전복구, 비상정지



충격 감지

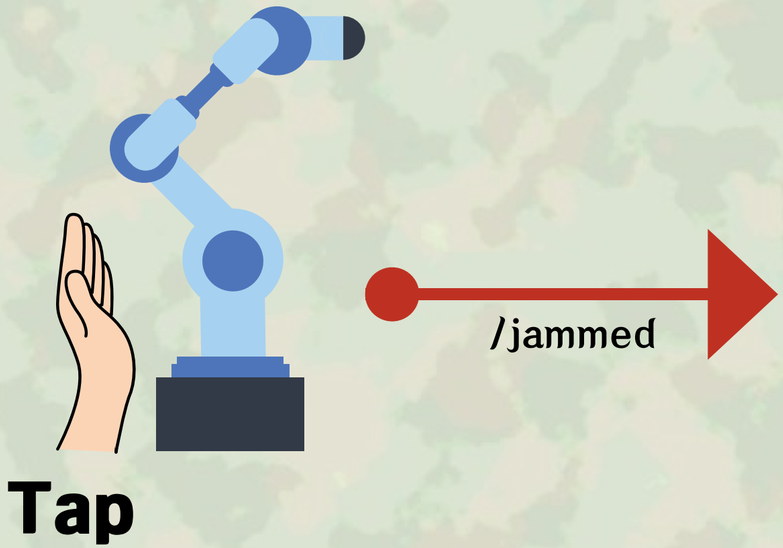
/shocked_five
/shocked_three

안전복구, 비상정지




```python
def check_mission_error(self):
    if not get_robot_state: return
    state = get_robot_state()
    if state in [3, 5, 6]:
        raise RuntimeError(f"ROBOT_ERROR_{state}")

def wait_robot(self):
    if mwait: mwait()
    self.check_mission_error()

def safe_sleep(self, duration):
    start = time.time()
    while time.time() - start < duration:
        self.check_mission_error()
        time.sleep(0.
```

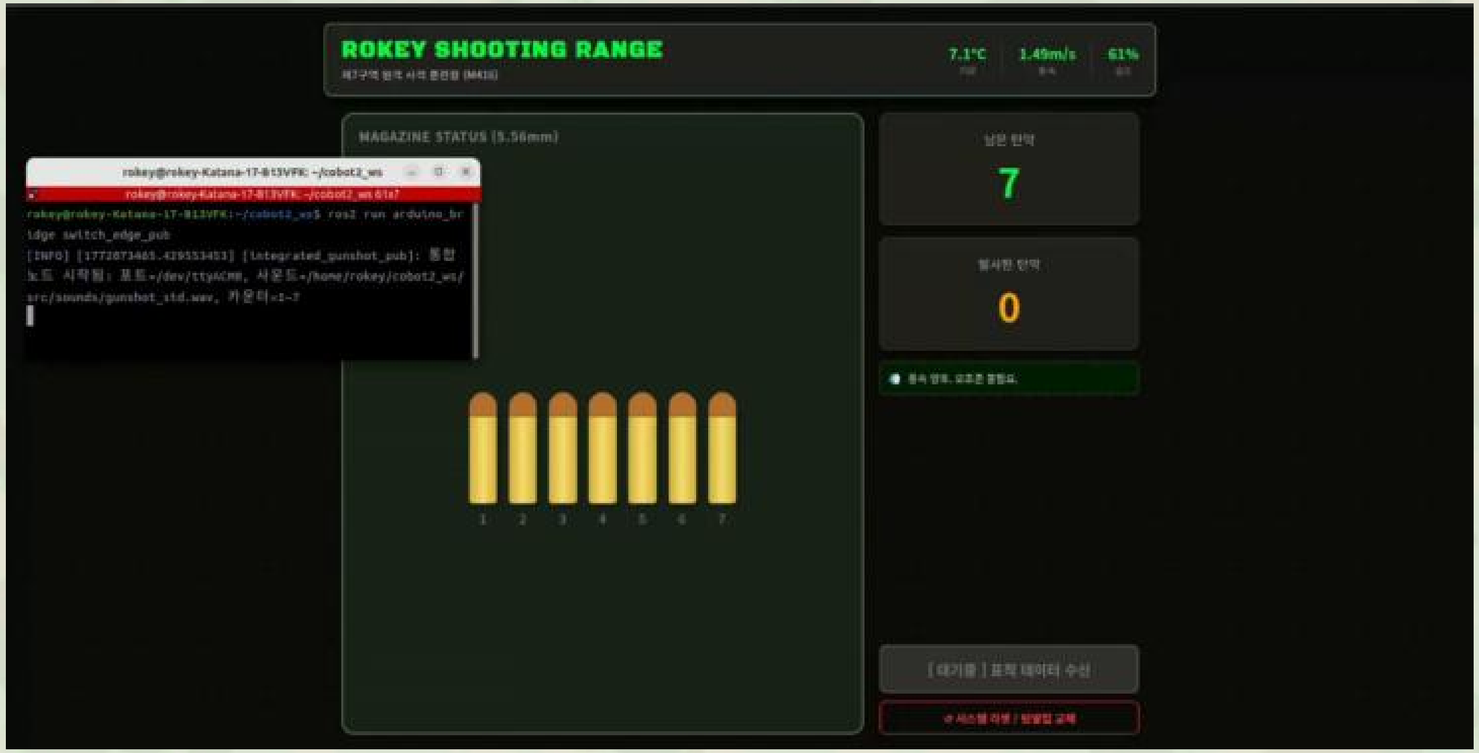
기능고장



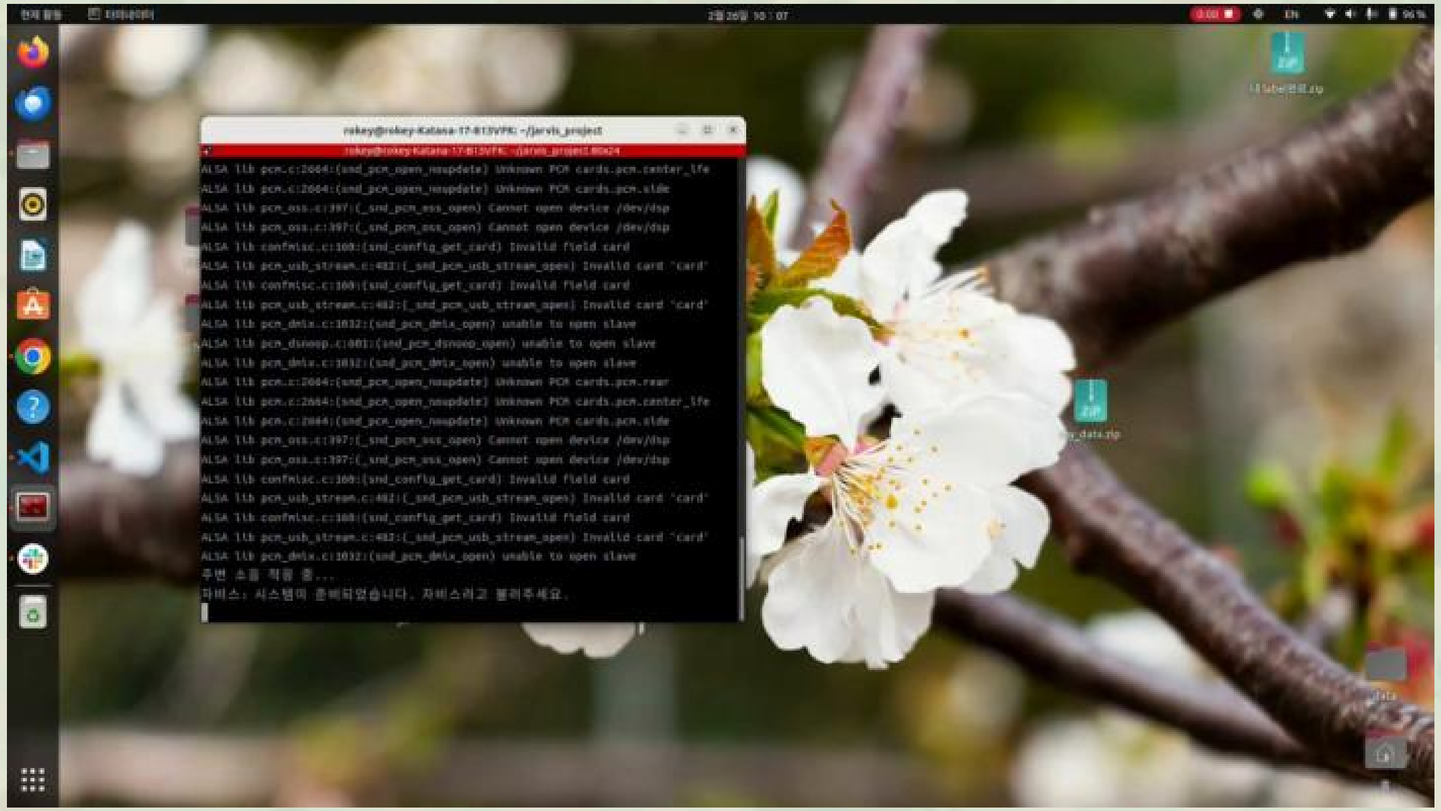
Project 수행경과

아두이노

/signal_shoot 1~7



Jarvis-System Embedded



가동 프로그램

구동에 있어 필요한 환경설정을
"자비스"로 wakeup 후
"준비해"로 환경설정 순서로
실행시킴으로 환경 설정을 완료함

Self-Review

MISSION COMPLETE

각 팀원에 맞는 적절한 업무 배정으로
프로젝트 수행에 있어
과업에 대한 높은 이해도를 바탕으로
robot 제어, IR 이미지 분석, UI 활용으로
프로젝트를 완수함.

GOOD & BAD

Good : 모든 구현을 InfraRed로 시작하여
다른 프로젝트와의 차별성이 존재
Bad : 프로젝트 수행 중 결원 발생으로
(탈영병 존재) 프로젝트 수행에 있어
업무 가중 발생

FUTURE IMPROVEMENT

군 사격장 뿐만 아닌
민간 사격장에서의 시뮬레이션을
통해 민간사격장에서의
위험요소 등을 파악하여
적용한 모델로 개선

실제 로봇 암과 카메라를 활용한 실습 중심 교육을 통해 산업 현장에서 요구되는 직무 역량을 체감할 수 있었던 뜻깊은 프로젝트였음.

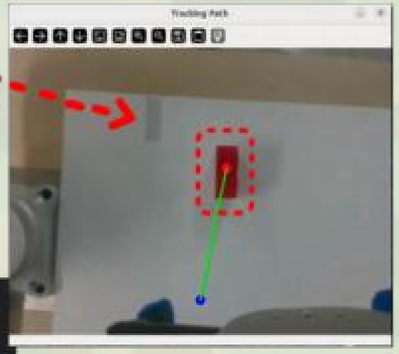
Zero-shot 따라하기

```python
class GeminiZeroShotNode(Node):
    def __init__(self):
        super().__init__('gemini_node', namespace='dsr01')
        self.client = OpenAI(api_key="입력")
        self.bridge = CvBridge()
        self.latest_image = self.latest_depth_image = None

        self.target_item = "the red plastic block"
        self.gripper_point = "the center point of the silver metal gripper base located at the bottom center of the frame"

        self.create_subscription(Image, '/camera/camera/color/image_raw', self.image_callback, 10)
        self.create_subscription(Image, '/camera/camera/aligned_depth_to_color/image_raw', self.depth_callback, 10)

        self.gripper = None
        self.robot_initialized = False
        self.is_analyzing = False

    def init_gripper_onrobot(self):
        try:
            from robot_control.onrobot import RG
            self.gripper = RG("rg2", "192.168.1.1", "502")
        except: pass

    def image_callback(self, msg):
        self.latest_image = self.bridge.imgmsg_to_cv2(msg, "bgr8")
        if self.latest_image is not None:
            cv2.imshow("Zero-Shot Visual Feedback", self.latest_image)
            cv2.waitKey(1)

    def depth_callback(self, msg):
        self.latest_depth_image = self.bridge.imgmsg_to_cv2(msg, "16UC1")

    def run_mission(self):
        if not self.robot_initialized:
            movej([4.47, 7.33, 50.38, -0.04, 122.29, -0.98], vel=30, acc=30)
            self.robot_initialized = True
            return
```

robot 움직임은 초기 pose 만 설정

Open AI 를 임포트한 뒤 TCP 포인트, object

```python
        # 시각화: 물체(빨강), 그리퍼(파랑 - 로키님이 찍은 픽셀 곳), 경로(초록)
        vis_img = self.latest_image.copy()
        cv2.circle(vis_img, (t_px, t_py), 8, (0, 0, 255), -1)
        cv2.circle(vis_img, (g_px, g_py), 8, (255, 0, 0), -1) # 파란 점 위치 확인!
        cv2.line(vis_img, (g_px, g_py), (t_px, t_py), (0, 255, 0), 2)
        cv2.imshow("Tracking Path", vis_img)
        cv2.waitKey(1)

        # [피드백 제어]
        diff_x_px = t_px - g_px
        diff_y_px = t_py - g_py

        z_dist = float(self.latest_depth_image[t_py, t_px])
        if z_dist < 100.0: return

        move_x_mm = (diff_y_px * z_dist / 615.0) * 1.0
        move_y_mm = (diff_x_px * z_dist / 615.0) * 1.0

        self.get_logger().info(f"📏 보정 거리: X {move_x_mm:.1f}mm, Y {move_y_mm:.1f}mm")

        if abs(move_x_mm) < 15.0 and abs(move_y_mm) < 15.0:
            self.get_logger().info("🎯 정렬 완료! 하강 및 집기")
            movel([0, 0, z_dist - 215.0, 0, 0, 0], vel=20, acc=20, ref=1)
            if self.gripper: self.gripper.close_gripper()
            time.sleep(1.0)
            movel([0, 0, -100.0, 0, 0, 0], vel=30, acc=30, ref=1)
        else:
            self.get_logger().info("🔄 파란 점을 빨간 점으로 이동 중...")
            movel([move_x_mm * 0.4, move_y_mm * 0.4, 0, 0, 0, 0], vel=40, acc=40, ref=1)
```



학습없이 targeting

Fianl Demo

Thank You