

# 협동 2 프로젝트 결과 보고서

F - 4 조

김현아

장석환

홍유진

# 목차

01 팀 소개

02 문제점 & 해결 방안

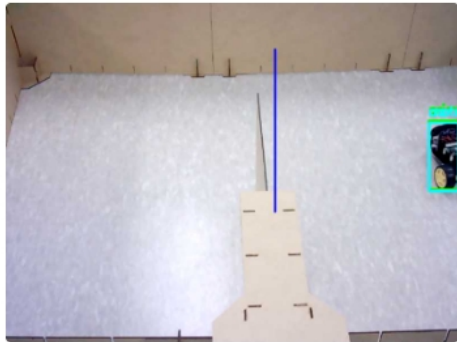
03 결과

# 01 팀 소개

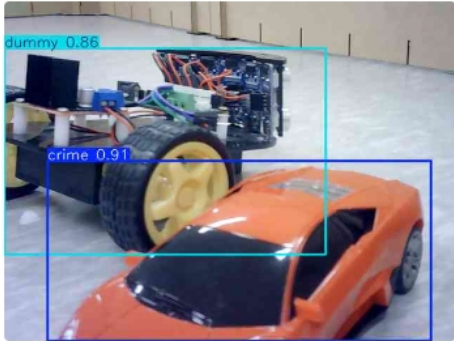
- AI 비전 감시 시스템 구축

## Criminal Detection System

### YOLO Node Image



### AMR Webcam



### AMR Current Location

Rviz2 View

### Detected Object & Crime Search

crime

Search

ct ID	Confidence	X	Y	Timestar
finid	crime-1	0.4055522680282593	595.5	141
finid	crime-2	0.3582836985588074	631.5	94

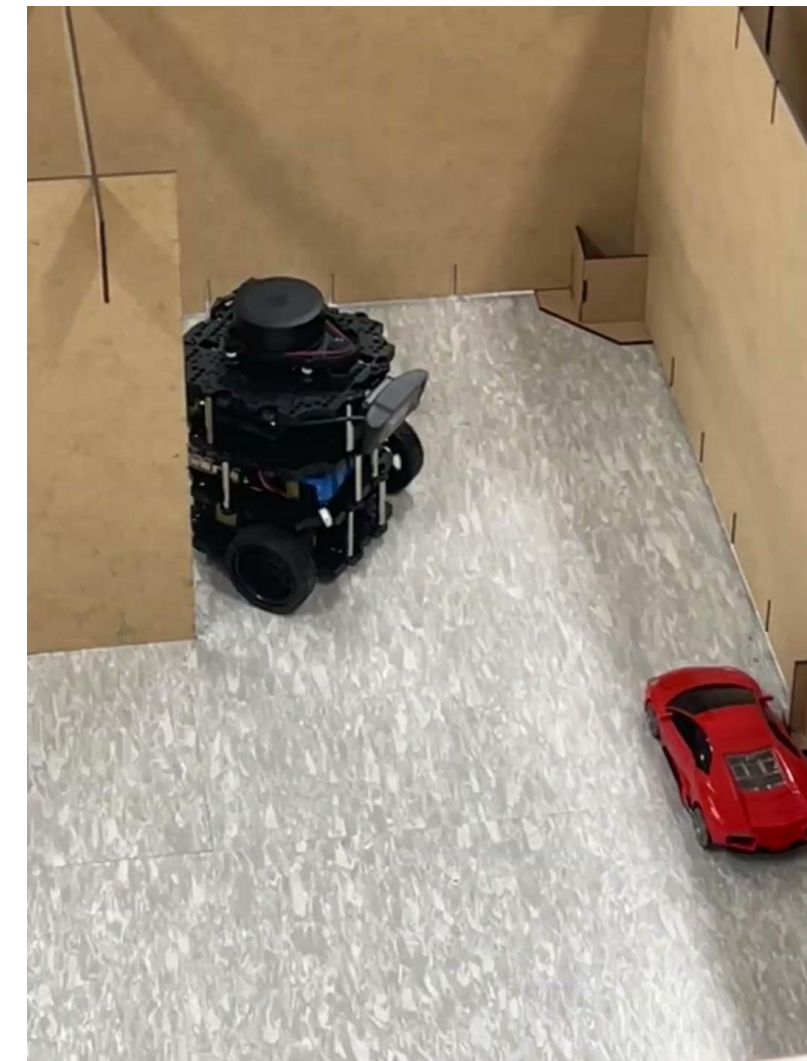
Previous Page 1 Next

### Notification Status

Object ID: 1, Time: 2024-11-11 15:14:34  
Object ID: 2, Time: 2024-11-11 15:14:35  
Object ID: 3, Time: 2024-11-11 15:14:37  
Object ID: 4, Time: 2024-11-11 15:17:38  
Object ID: 7, Time: 2024-11-11 15:20:31  
Object ID: 8, Time: 2024-11-11 15:20:57

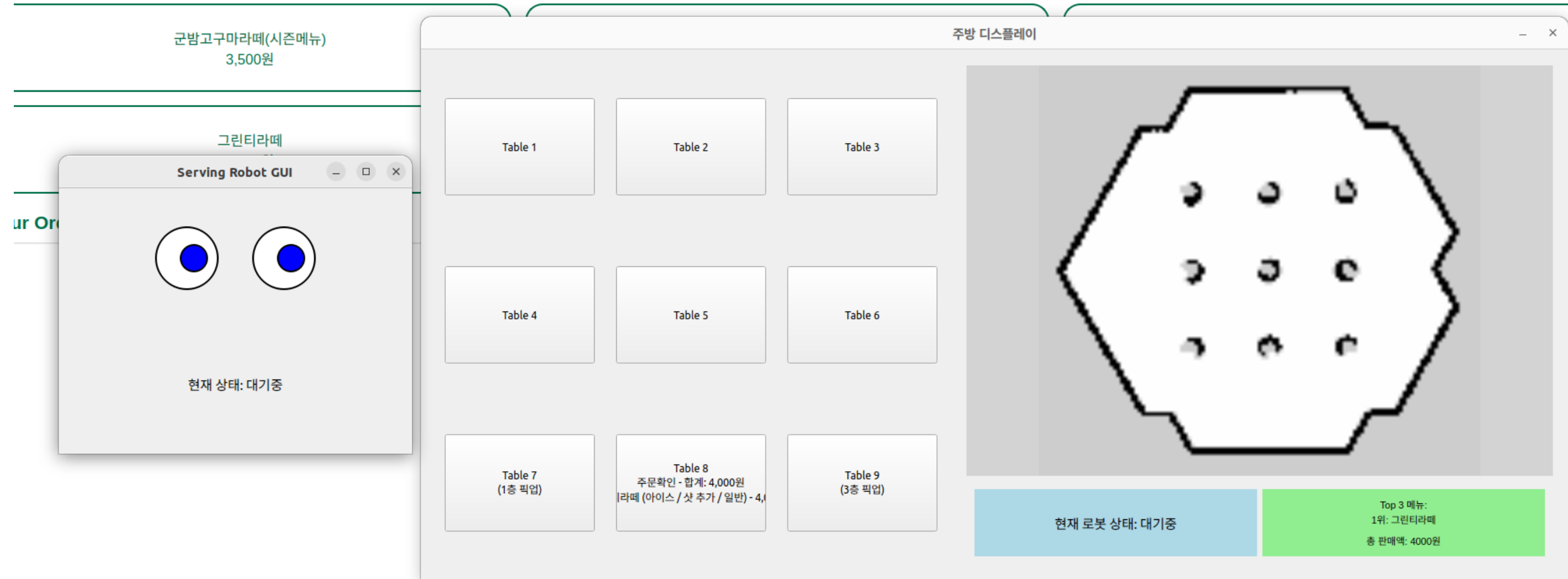
### AMR Current State

AMR 상태: 초기 위치



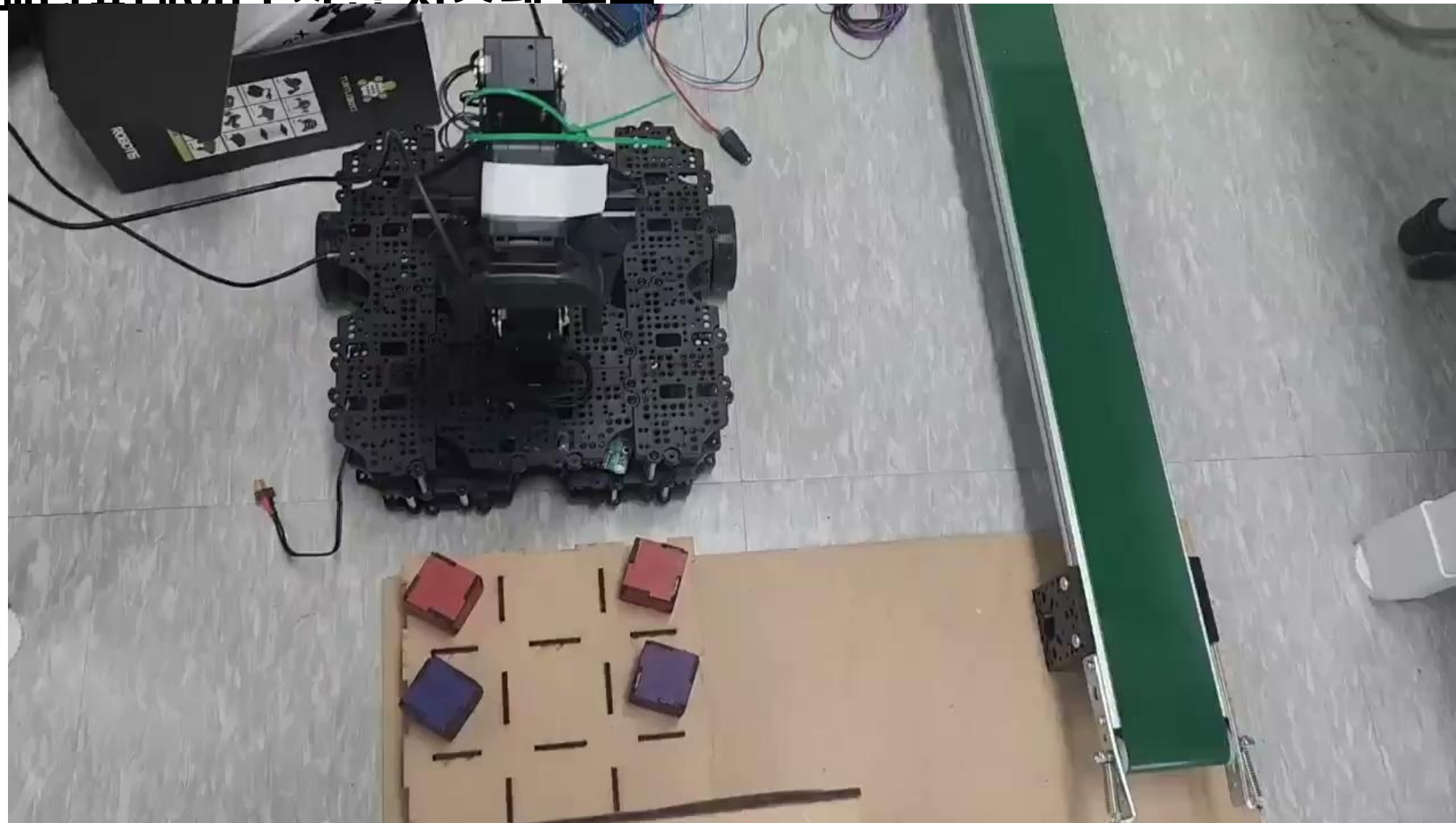
# 01 팀 소개

- 서빙로봇 개발 시뮬레이션



# 01 팀 소개

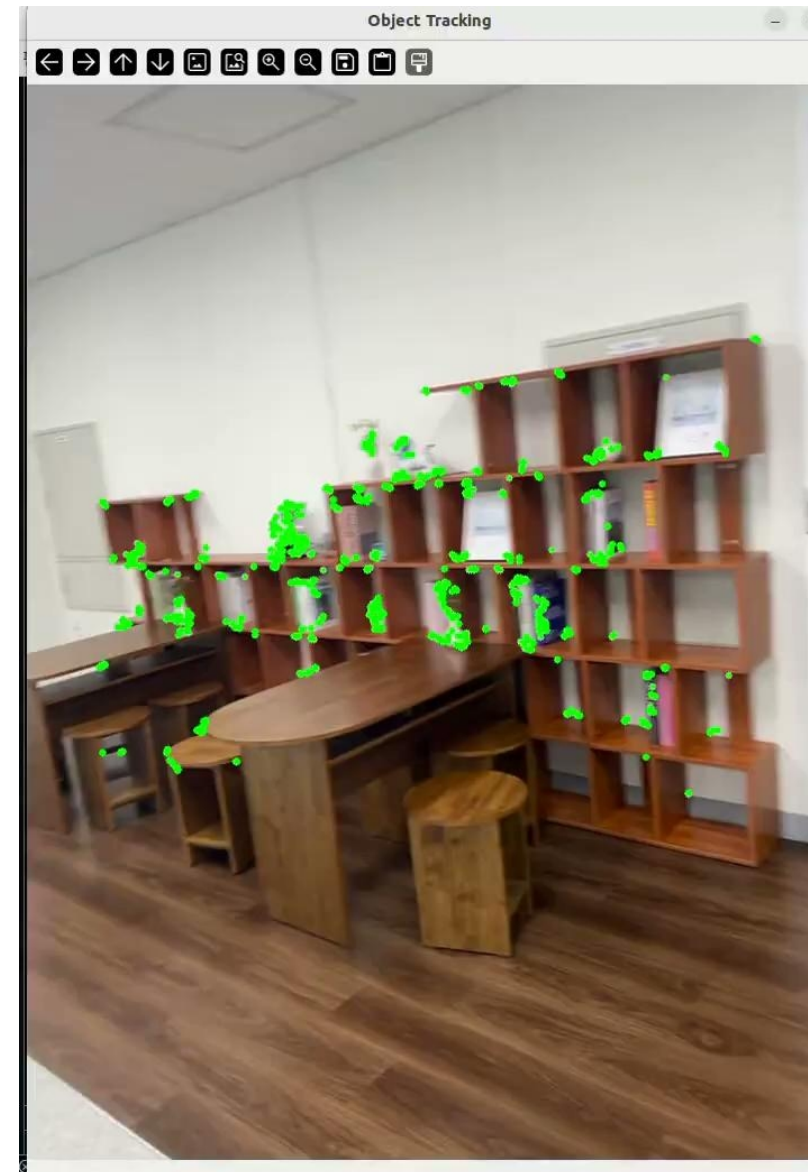
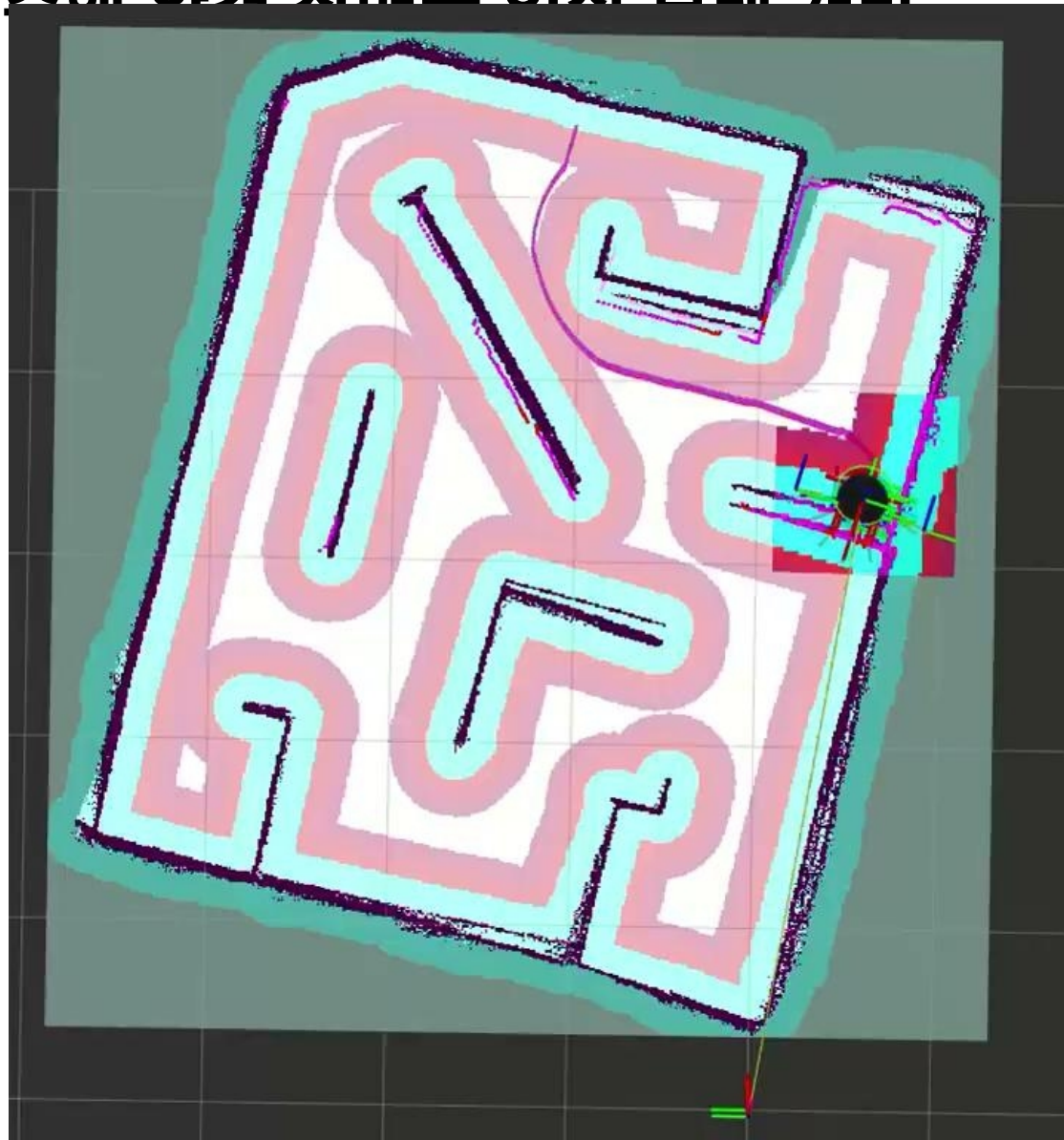
- 산업용 매니퓰레이터 제어 시스템 그라운드





# 01 팀 소개

- 로봇 조해 하거 자메르 이시 그데 기바



## 02 문제점

컵 시작 좌표 고정 문제

컵을 잡을 때 그리퍼의 흔들림으로 기본 시작 위치 변경

다중 컵 문제

그리퍼 사용 시 컵 2 개 이상이 동시에 잡히는 현상 발생

좌표 계산 문제

한 점을 기준으로 모든 점의 좌표를 계산해 오차 발생

## 03 해결 방안

- 컵 시작 좌표 고정 문제



- 나사를 사용해 처음 컵의 위치 고정
- 그리퍼 작업 중 흔들림 최소화를 통해 좌표 오차 방지



## 03 해결 방안

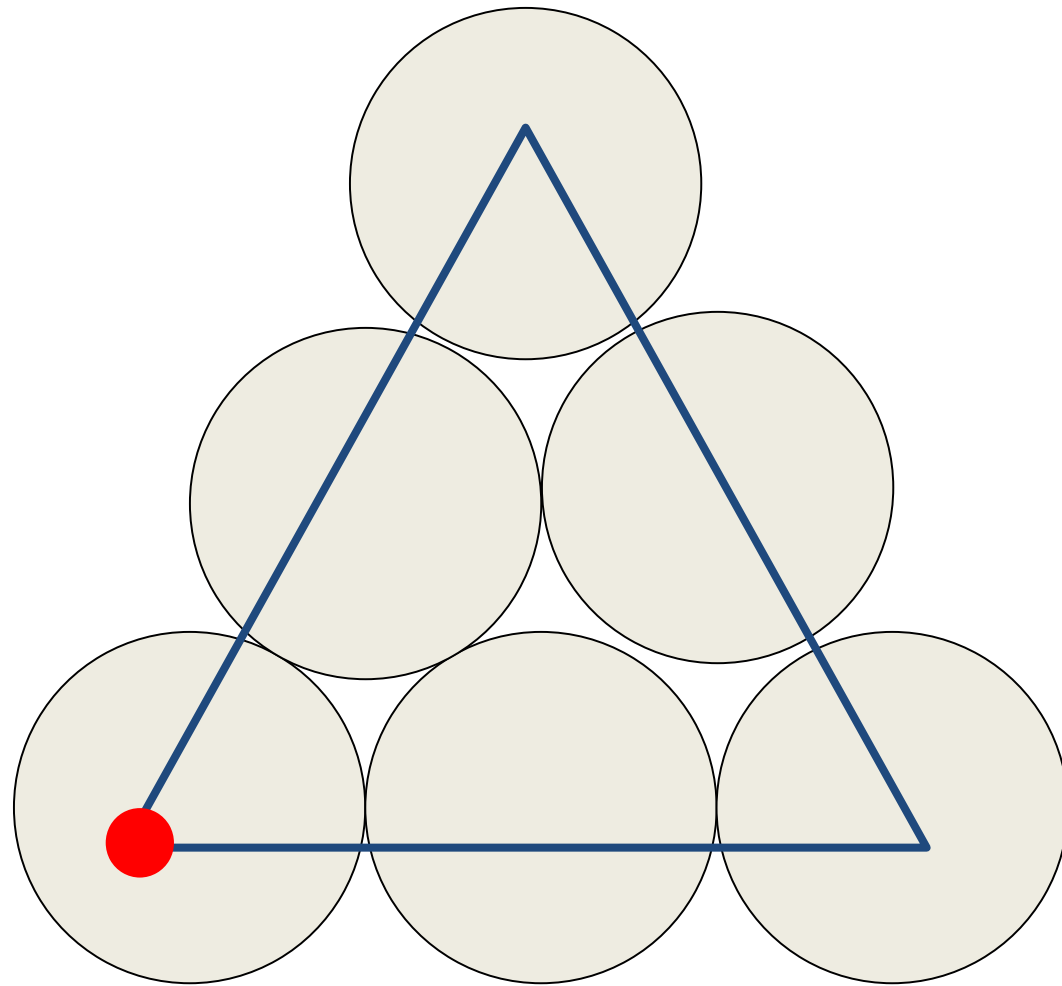
- 다중 컵 문제

<input checked="" type="checkbox"/> #0	RG2-0 ▼																
IN <table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr><tr><td>1</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr></table>	1	2	3	4	5	6	7	8	1	0	-	-	-	-	-	-	Width (56 mm, 5 N)
1	2	3	4	5	6	7	8										
1	0	-	-	-	-	-	-										
<input checked="" type="checkbox"/> #1	RG2-0 ▼																
IN <table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr><tr><td>0</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr></table>	1	2	3	4	5	6	7	8	0	1	-	-	-	-	-	-	Width (70 mm, 40 N)
1	2	3	4	5	6	7	8										
0	1	-	-	-	-	-	-										

- 그리퍼의 힘을 23N → 5N 으로 변경
- 하나의 컵만 정확히 잡도록 개선

## 03 해결 방안

- 좌표 계산 문제



```
def calculate_triangle_and_midpoints(x1, y1, side_length):  
  
    x2 = x1 + side_length * math.cos(math.radians(120))  
    y2 = y1 + side_length * math.sin(math.radians(120))  
  
    x3 = x1 + side_length * math.cos(math.radians(240))  
    y3 = y1 + side_length * math.sin(math.radians(240))  
  
    m1 = ((x1 + x2) / 2, (y1 + y2) / 2) # 첫 번째 변의 중간점  
    m2 = ((x2 + x3) / 2, (y2 + y3) / 2) # 두 번째 변의 중간점  
    m3 = ((x3 + x1) / 2, (y3 + y1) / 2) # 세 번째 변의 중간점  
  
    return {  
        "Vertex 1": (round(x1, 2), round(y1, 2)),  
        "Vertex 2": (round(x2, 2), round(y2, 2)),  
        "Vertex 3": (round(x3, 2), round(y3, 2)),  
        "Midpoint 1-2": (round(m1[0], 2), round(m1[1], 2)),  
        "Midpoint 2-3": (round(m2[0], 2), round(m2[1], 2)),  
        "Midpoint 3-1": (round(m3[0], 2), round(m3[1], 2))  
    }
```

## 03 해결 방안

- 좌표 계산 문제

```
cup1 = posx(272.375, -197.656, 200.4, 2.998, -179.319, 4.023) #11개짜리 컵 높이
cup=[cup1]

for i in range (9):
    prev_cup = posx(*cup[i])
    cup.append(trans(prev_cup, zdown1))

cpos1_1 = posx(600.224, -81.234, 180.17, 89.978, -179.381, 90.876)
cpos1_2 = posx(598.293, -4.537, 180.72, 54.443, -179.26, 55.234)
cpos1_3 = posx(600.129, 72.453, 180.519, 63.009, -179.275, 63.303)
cpos1_4 = posx(530.565, 40.163, 180.017, 55.163, -179.382, 55.7)
cpos1_5 = posx(530.751, -42.885, 180.583, 54.077, -179.437, 54.73)
cpos1_6 = posx(464.756, -4.085, 180.966, 49.847, -179.684, 50.417)
```

- 1 층 전체를 좌표값으로 설정
- 계산 과정을 단순화하여 오차 최소화

## 04 결과

### ● 좌표 계산

```
zdown = posx(0, -12, 0, 0, 0)
```

```
cup1 = posx(272.375, -197.656, 200.4, 2.998, -179.319, 4.023) #11개짜리 컵 높이  
cup=[cup1]
```

```
for i in range(9):  
    prev_cup = posx(*cup[i])  
    cup.append(trans(prev_cup, zdown1))
```

```
cpos1_1 = posx(600.224, -81.234, 180.17, 89.978, -179.381, 90.876)  
cpos1_2 = posx(598.293, -4.537, 180.72, 54.443, -179.26, 55.234)  
cpos1_3 = posx(600.129, 72.453, 180.519, 63.009, -179.275, 63.303)  
cpos1_4 = posx(530.565, 40.163, 180.017, 55.163, -179.382, 55.7)  
cpos1_5 = posx(530.751, -42.885, 180.583, 54.077, -179.437, 54.73)  
cpos1_6 = posx(464.756, -4.085, 180.966, 49.847, -179.684, 50.417)  
cpos2_1 = [(cpos1_1[0]+cpos1_4[0])/2, (cpos1_1[1]+cpos1_2[1])/2, 274.966, 49.847, -179.684, 50.417]  
cpos2_2 = [(cpos1_2[0]+cpos1_4[0])/2, (cpos1_2[1]+cpos1_3[1])/2, 274.966, 49.847, -179.684, 50.417]  
cpos2_3 = [(cpos1_4[0]+cpos1_6[0])/2, -4.085, 274.966, 49.847, -179.684, 50.417]  
cpos3 = [554.264, -4.537, 295.815, 96.932, -179.649, 97.426]  
cpos = [cpos1_1, cpos1_2, cpos1_3, cpos1_4, cpos1_5, cpos1_6, cpos2_1, cpos2_2, cpos2_3, cpos3]  
last_cup = [281.457, -177.982, 115.014, 59.331, 93.271, 90.613]  
put_last_cup = [570.118, 15.017, 330.672, 61.703, 94.896, -89.788]
```

한번 옮길때마다  
Z 축으로 -12 하강

1 층 전체 좌표값 설정  
2 층 중간위치 계산

## 04 결과

### ● 컵 옮기기

```
def move_cup(start_pos, end_pos):  
    release()  
    movel(start_pos, vel=VELOCITY, acc=ACC)  
    movel(zdown, vel = VELOCITY, acc = ACC, mod = DR_MV_MOD_REL)  
    grip()  
    movel(zup, vel = VELOCITY, acc = ACC, mod = DR_MV_MOD_REL)  
    movel(xup, vel = VELOCITY, acc = ACC, mod = DR_MV_MOD_REL)  
    movel(end_pos, vel=VELOCITY, acc=ACC)  
    movel(zdown, vel = VELOCITY, acc = ACC, mod = DR_MV_MOD_REL)  
  
    task_compliance_ctrl(stx=[500, 500, 500, 100, 100, 100])  
    set_desired_force(fd=[0, 0, -30, 0, 0, 0], dir=[0, 0, 1, 0, 0, 0], mod=DR_FC_MOD_REL)  
    while not check_force_condition(DR_AXIS_Z, max=5):  
        pass  
    release_compliance_ctrl()  
  
    release()  
    movel(end_pos, vel=VELOCITY, acc=ACC)
```

# 그리퍼 열림  
# 잡을 컵의 상단으로 이동  
# 컵을 잡을 위치로 이동  
# GRIP!!  
# 컵을 다른 컵들로부터 빼기 위해 z축으로 들어올림  
# 컵이 이동하다 처음 위치에 존재하는 컵들을 치지 않기 위해 앞으로 이동  
# 컵을 놓을 위치의 상단으로 이동  
# 시간 단축 위해 일정값 아래로 이동  
  
# z방향으로 -30N의 힘으로 내리며 force condition 체크  
# 5N이상의 힘이 작용될 경우 종료  
  
# RELEASE!!!  
# 놓은 컵이 흔들리지 않기 위해 놓은 위치의 상단으로 이동



## 04 결과

- 컵 회전

```
def turn_cup(start_pos, end_pos):
    release()
    movel(start_pos, vel=100, acc=ACC) #컵을 집을 위치에 도달(옆으로)
    movel(zdown2, vel = 100, acc = ACC, mod = DR_MV_MOD_REL) #z를 내려서 컵 집기
    grip()
    movel(zup, vel = 100, acc = ACC, mod = DR_MV_MOD_REL) #z up
    movej(turn, vel= 100, acc =ACC, mod = DR_MV_MOD_REL) #180도 회전(90도로 2번)
    movej(turn, vel= 100, acc =ACC, mod = DR_MV_MOD_REL)
    movel(zup2, vel = 100, acc = ACC, mod = DR_MV_MOD_REL) #꼭대기 컵에 부딪히지 않게 하기 위해 z up
    movel(end_pos, vel=VELOCITY, acc=ACC) #end_pos로 이동

    task_compliance_ctrl(stx=[500, 500, 500, 100, 100, 100]) #돌아간 컵을 제대로 놓기 위한 force compliance
    set_desired_force(fd=[0, 0, -30, 0, 0, 0], dir=[0, 0, 1, 0, 0, 0], mod=DR_FC_MOD_REL)
    while not check_force_condition(DR_AXIS_Z, max=5):
        pass
    release_compliance_ctrl()
    release()
    movel(down, vel = VELOCITY, acc = ACC, mod = DR_MV_MOD_REL) #완성된 탐에 충돌을 방지하기 위해 x, y down
```

## 04 결과

- 전체 동작

```
while rclpy.ok():
    release()
    print("I'm Ready")
    movej(JReady, vel=VELOCITY, acc=ACC)

    for i in range(10):
        movej(JReady, vel=VELOCITY, acc=ACC)
        print(f"{i+1}번 컵 옮기는 중")
        move_cup(cup[i], cpos[i])

    movej(JReady, vel=VELOCITY, acc=ACC)

    print("11번 컵 옮기는 중")
    turn_cup(last_cup, put_last_cup)
    movej(JReady, vel=VELOCITY, acc=ACC)
    print("프로그램 종료")

rclpy.shutdown()
```

move\_cup 으로 10 번째 컵까지  
쌓기

turn\_cup 으로 11 번째 컵 회전 후  
쌓기

## 04 결과

1

### 단순한 로직의 중요성

때로는 복잡한 계산이나 추론 과정을 줄이는 것이 문제 해결의 실마리가 될 수 있다

2

### 환경에 맞춘 세심한 조정

작업 환경 ( 초기 좌표 , 그리퍼 힘 등 ) 에 맞춘 수치 조정이 필수적이다

3

### 소통의 중요성

서로의 강점과 약점을 적극적으로 공유하고 소통할 때 어려운 문제도 팀의 힘으로 해결할 수 있다  
있다

## 04 결과

- 시연

