[두산로보틱스] 지능형 로보틱스 엔지니어

Auto Weld(용접협동로봇)

TEAM B-3조 가디구디

최정호, 이세현, 이하빈, 홍진규

[멘토] 김민수

K-Digital Training

목 차

K-Digital Training

01 프로젝트 개요

1

주제 선정 배경

수작업 용접의 한계

2

프로젝트 내용

프로젝트 구현,
훈련내용과의 연관성

3

활용 장비 및 개발환경

활용 장비 및 개발환경

ROS 2™

4

프로젝트 구조

이미지 → 좌표화

↓

스케일링/저장

↓

자동 용접 제어

5

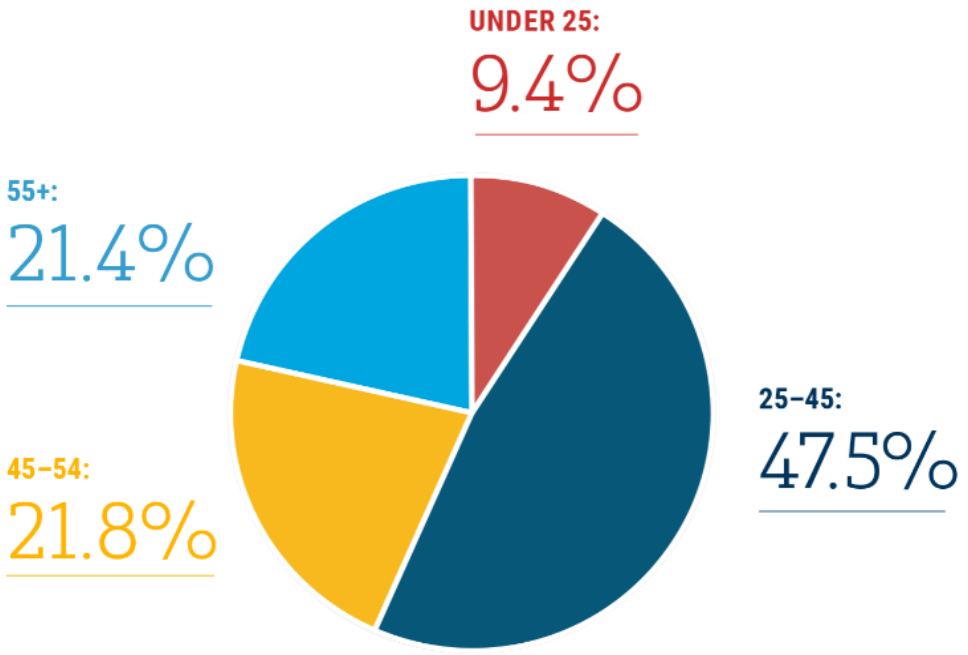
활용방안 및 기대 효과

기대 효과와
비즈니스 실무 활용성

주제 선정 배경

- **심각한 용접 인력 부족**
  - 미국의 경우 2025~2029년까지 매년 8만명 이상의 신규 용접 인력이 필요, 누적 32만명 공백 예상

- **생산성↑**
  - 중국 제조업체가 DOBOT CRA cobot 도입 후 MIG 공정 생산성 30%↑



현재 전 세계의 용접 인력의 연령대 분포

https://weldingworkforcedata.com/
https://www.fanucamerica.com/news-resources/articles/why-welding-robots-and-cobots-are-becoming-essential
https://www.dobot-robots.com/insights/news/how-robot-welders-transform-manufacturing.html?utm.com

주제 선정 배경

- 작업자 안전

구분	위험 요소	급성 영향	만성, 장기 영향
물리적	아크 광선	각막 손상 ,피부 화상, 수포	피부암, 백내장
	소음	두통, 집중력 저하	소음성 난청, 이명
화학적	금속 산화 미세입자	금속열 증후군, 호흡 곤란	폐암, 신장암
	유해가스 (O3, CO)	두통, 급성 중독	만성 기관지염
기타	진동	손 저림, 혈류 장애	손-팔 진동 증후군

용접 협동로봇을 통해 작업인력 수요 해결, 생산 효율성 증대 및 작업자의 안전 증대를 목표

01 프로젝트 개요

프로젝트 내용

다양한 유형의 용접기능 지원: 규칙적인 2D, 불규칙적인 2D, 규칙적인 3D 용접 기능 지원

수학적 궤적 분석: 3점 캡처를 통해 원의 중심 및 반지름을 계산하여 궤도 형상 추정

ROS2 노드 및 토픽 통신 구현: 실시간 로봇 위치(posx, posj) 수신 및 표시

Doosan API를 활용한 로봇 이동 제어: `movej(posj(...))` 명령으로 홈 위치 이동 기능 구현

Tkinter 기반 GUI: 실시간 좌표 확인, 캡처, 복사, CSV 저장, 궤적 계산 기능 포함

https://weldingworkforcedata.com/

프로젝트 개요

활용 장비 및 개발환경

Doosan M0609

python™

ROS 2™

Tkinter

Matplotlib

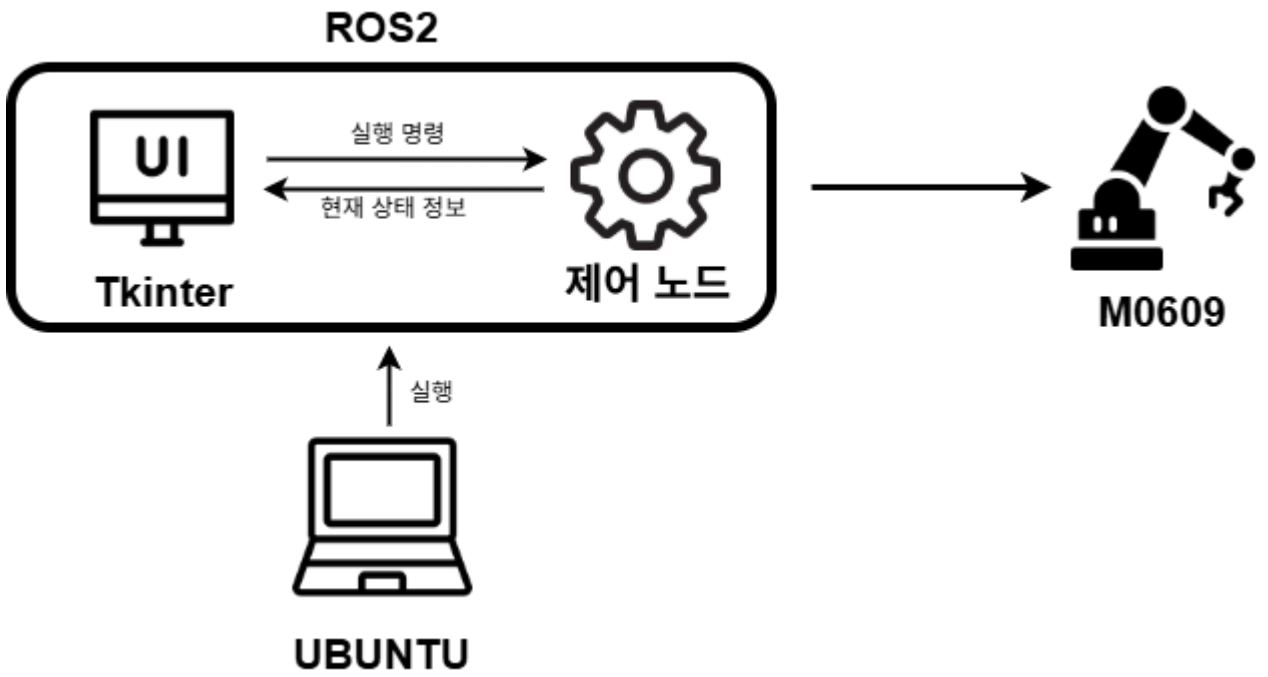
Visual Studio Code

https://weldingworkforcedata.com/

프로젝트 개요

프로젝트 구조

- Tkinter UI와 제어노드 분리
- 제어노드에서 DR명령어 실행하여 용접로봇(M0609) 동작



01 프로젝트 개요

프로젝트 산출물의 기대효과 및 비즈니스 실무 활용성

- 정확한 원호 궤적 제어로 작업 품질 향상
- 실시간 위치 모니터링으로 안전성 강화
- 자동화 작업 효율성 증가 및 작업 시간 단축
- 산업 현장 협동로봇 적용 가능성 확대
- 비즈니스 현장 자동화 및 스마트 팩토리 구현에 기여
- 안전사양을 통한 작업자 보호

https://weldingworkforcedata.com/

프로젝트 팀 구성 및 역할

담당 업무와 역할

B-3	역할	담당 업무
최정호	팀장	규칙적 2D 용접 경로 구현
이하빈	팀원	불규칙적 2D 용접 경로 구현
이세현	팀원	규칙적 3D 용접 경로 구현
홍진규	팀원	충돌 감지 후 즉시 정지(안전기능)

03 프로젝트 수행 절차 및 방법

구분	기간	활동	비고
사전 기획	5/14(수) ~ 5/15(목)	프로젝트 기획 및 주제 선정	프로젝트 주제선정
정보 수집	5/15(목) ~ 5/16(금)	관련 정보 수집	기존의 용접로봇 참고 작동구현 유튜브 시청
1차 개발	5/17(토) ~ 5/18(일)	작동구현방식 코딩	
중간 점검	5/19(월) ~ 5/20(화)	파트별 코딩 테스트	팀원간 중간코딩 점검
최종 개발	5/20(화) ~ 5/21(수)	각 코드간 시스템 설계	최적화, 오류 수정
총 개발기간	5/14(수) ~ 5/21(목)(총 8일)	조원들의 적극적인 참여와 활동을 통해 획기적인 솔루션 도출	

프로젝트 수행 경과

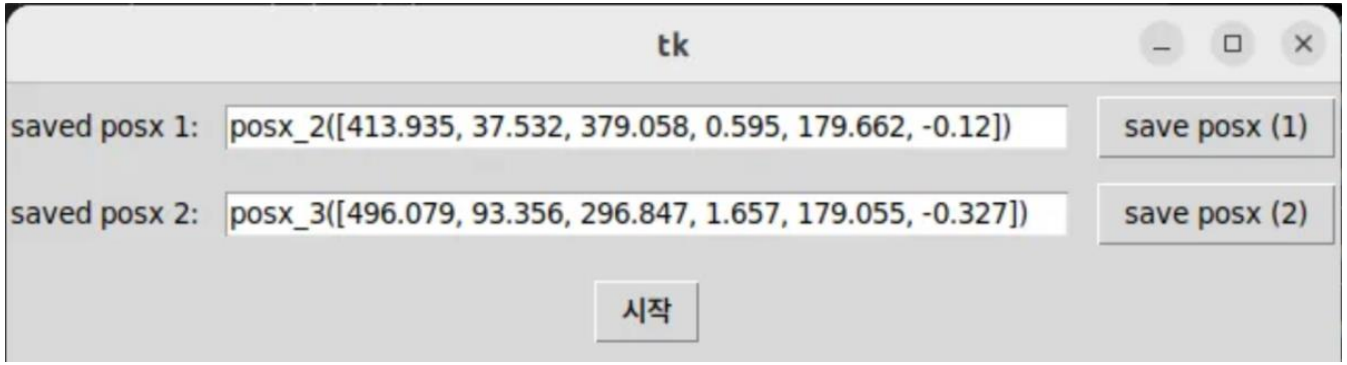
개발한 코드

- 규칙적인 2D 용접 기능
- 불규칙적인 2D 용접 기능
- 규칙적인 3D 용접
- **Welding Safety for Human-Robot Collaboration(안전기능)**

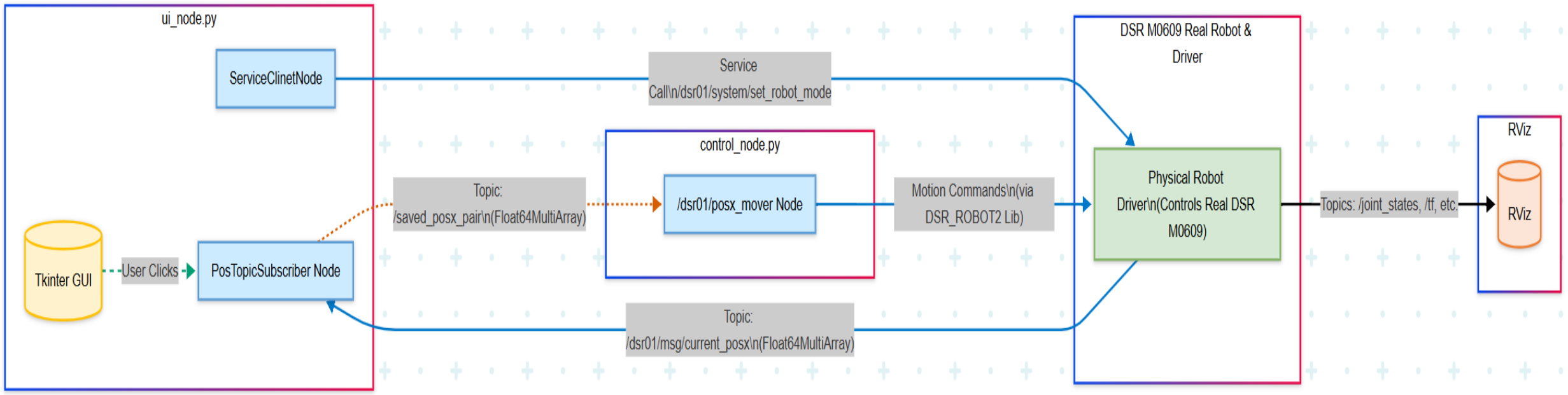
프로젝트 수행 경과

규칙적인 2D 용접 기능

- UI노드와 제어노드로 분리
- UI노드에서 직접 교시로 시작점과 도착점에 대한 좌표를 제어노드로 publish
- 제어노드에서 시작점과 도착점에 대한 좌표를 전달받으면, 선형이동 시작



노드 통신 구조


ui_node.py
ServiceClinetNode
Service Call\n/dsr01/system/set_robot_mode
DSR M0609 Real Robot & Driver
control_node.py
Topic: /saved_posx_pair\n(Float64MultiArray)
/dsr01/posx_mover Node
Motion Commands\n(via DSR_ROBOT2 Lib)
Physical Robot Driver\n(Controls Real DSR M0609)
Topics: /joint_states, /tf, etc.
RViz
RViz
Tkinter GUI
User Clicks
PosTopicSubscriber Node
Topic: /dsr01/msg/current_posx\n(Float64MultiArray)

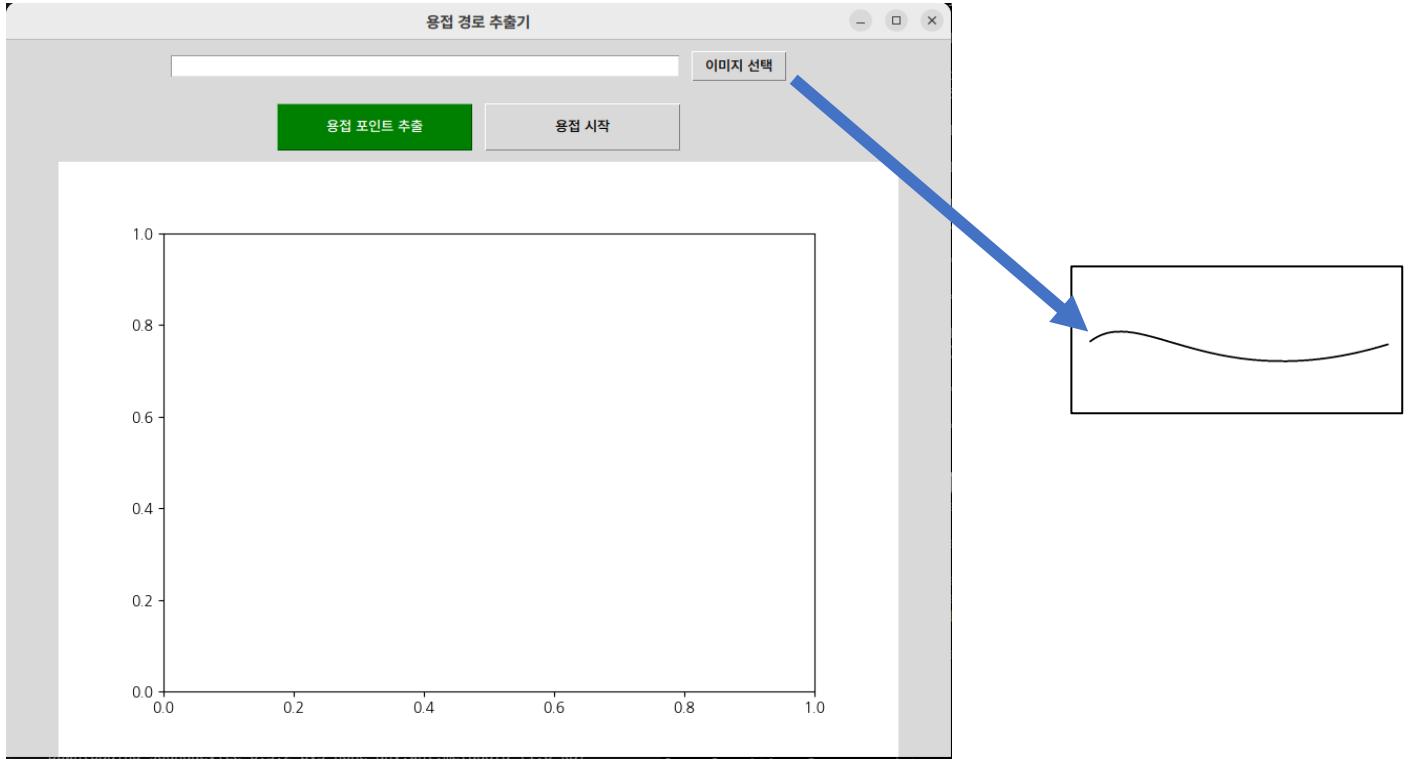
```python
# 메인 실행 함수
def main(args=None):
    global posx_2, posx_3, received

    rclpy.init(args=args)  # ROS2 초기화
    node = rclpy.create_node("posx_mover", namespace=ROBOT_ID)
    DR_init.__dsr__node = node  # 노드 설정 등록

    # 두산로봇 제어 함수 불러오기
    from DSR_ROBOT2 import set_tool, set_tcp, movej, movel
    from DR_common2 import posj

    JReady = [0, 0, 90, 0, 90, 0]  # 홈 위치 (기본 대기 자세)

    # 로봇 툴/TCP 설정
    set_tool("Tool Weight_2FG")  # 사용 중인 그리퍼 툴명
    set_tcp("2FG_TCP")           # 툴 기준 좌표계

    # 좌표 데이터 구독 시작
    node.create_subscription(
        Float64MultiArray,   # 메시지 타입
        "/saved_posx_pair",  # 구독할 토픽명
        callback,            # 콜백 함수 지정
        10                   # 큐 크기
    )
    print("[INFO] Waiting for posx data...")

    # 좌표 수신 완료될 때까지 대기 (비동기 이벤트 대기 방식)
    while rclpy.ok() and not received:
        rclpy.spin_once(node, timeout_sec=0.1)

    # 수신 완료되면 모션 시퀀스 실행
    if received and posx_2 and posx_3:
        print("[INFO] Starting motion sequence...")

        while rclpy.ok() and received:
            print("→ movej JReady")
            movej(JReady, vel=VELOCITY, acc=ACC)  # 홈 위치로 이동

            print("→ movel posx_2")
            movel(posx_2, vel=VELOCITY, acc=ACC)  # posx_2로 선형 이동

            print("→ movel posx_3")
            movel(posx_3, vel=VELOCITY, acc=ACC)  # posx_3로 선형 이동

            received = False  # 한 번만 실행하고 종료
    else:
        print("[ERROR] No posx data received. Exiting.")

    # ROS2 종료
    rclpy.shutdown()

# 메인 함수 호출
if __name__ == "__main__":
    main()
```

프로젝트 수행 경과

불규칙적인 2D 용접 기능

- 용접할 부분의 도면이나, 센서 등으로 용접로봇이 이동할 경로 정보를 얻어올 수 있다는 시나리오를 가정

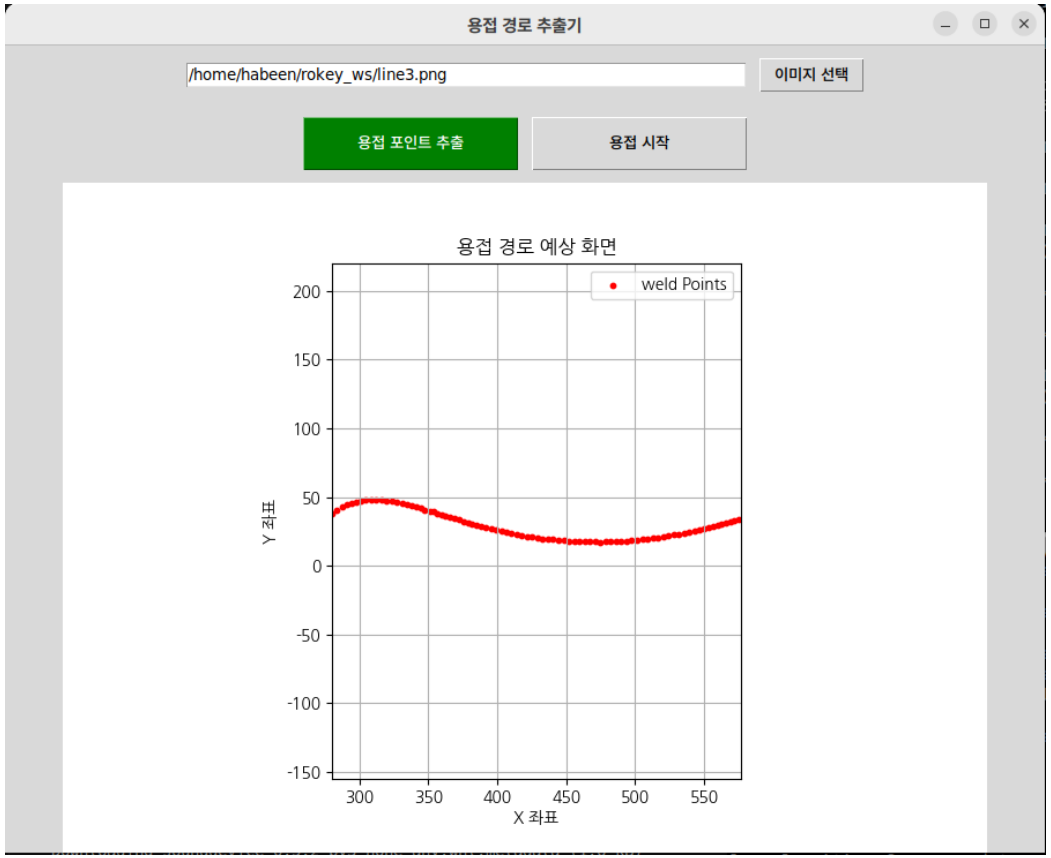
- 다음 과정으로 작업 수행
  - 이미지 선택
  - 용접 포인트 추출
  - 좌표 스케일 변환
  - CSV 저장 및 시각화
  - CSV 기반 좌표로 용접 실행



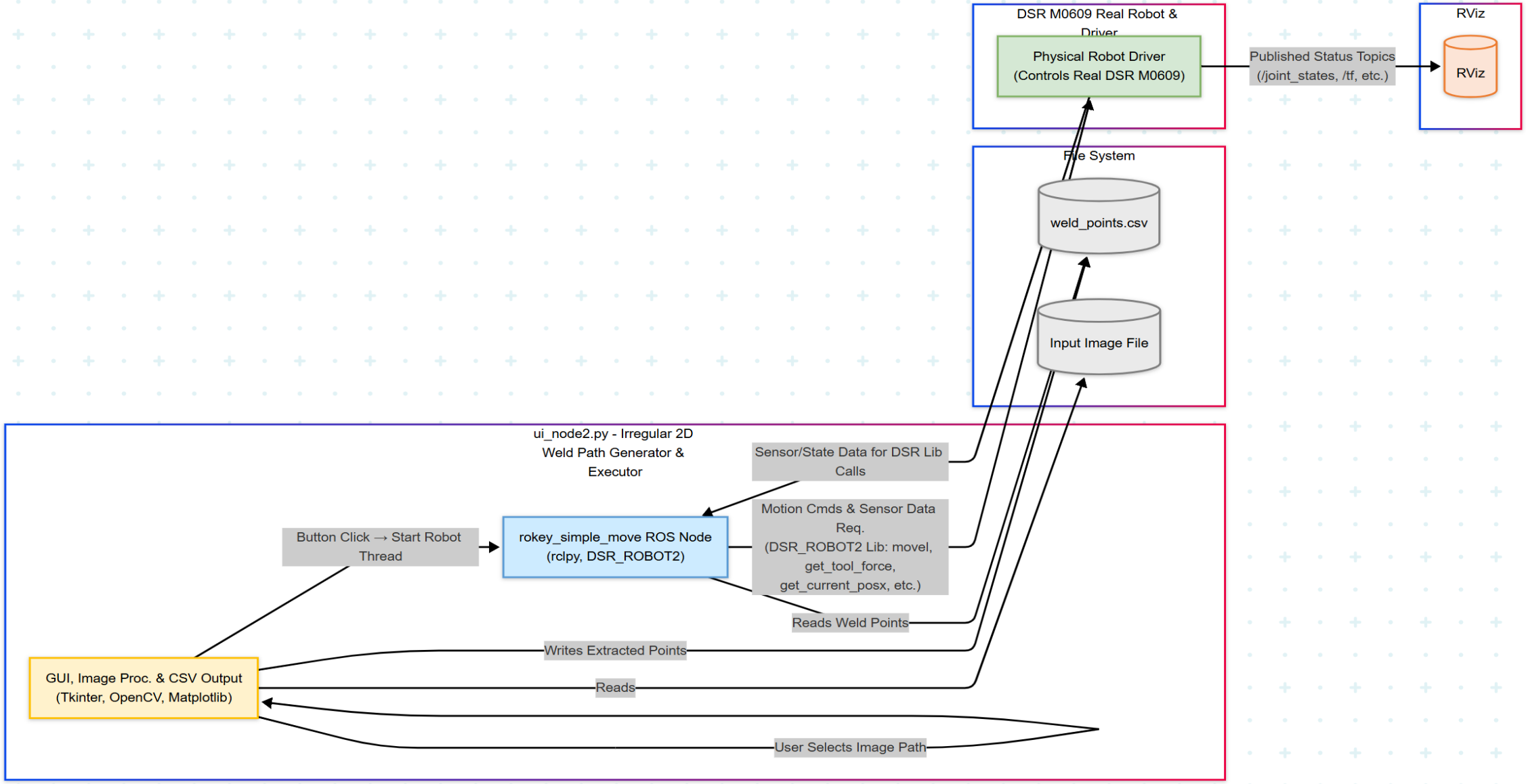
프로젝트 수행 경과

불규칙적인 2D 용접 기능

- 경로 이미지 파일을 업로드하면, 일정한 간격으로 용접로봇이 이동할 포인트들을 추출
- 해당 포인트가 로봇의 작동범위의 최대 최소값을 벗어나지 않도록 스케일링
- 스케일링된 좌표들을 csv 파일에 저장하여, 해당 좌표를 순차적으로 읽어 경로 이동



노드 통신 구조


DSR M0609 Real Robot & Driver
Physical Robot Driver
(Controls Real DSR M0609)
Published Status Topics
(/joint_states, /tf, etc.)
RViz
RViz
File System
weld_points.csv
Input Image File
ui_node2.py - Irregular 2D Weld Path Generator & Executor
Sensor/State Data for DSR Lib Calls
Motion Cmds & Sensor Data Req.
(DSR_ROBOT2 Lib: movel, get_tool_force, get_current_posx, etc.)
Button Click → Start Robot Thread
rokey_simple_move ROS Node
(rclpy, DSR_ROBOT2)
Reads Weld Points
Writes Extracted Points
Reads
User Selects Image Path
GUI, Image Proc. & CSV Output
(Tkinter, OpenCV, Matplotlib)

```python
# 핵심 요약 : GUI 기반 이미지 처리 + 용접 경로 좌표 추출 + 로봇 실행

# 1. 이미지에서 용접 경로의 선을 추출하는 함수
def extract_points(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    # 이미지를 흑백으로_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)  # 이진화
    skeleton = cv2.ximgproc.thinning(binary)  # 뼈대 추출 알고리즘 적용
    points = np.column_stack(np.where(skeleton > 0))[:, [1, 0]]  # 좌표 추출 및 x, y 순서 변경
    return points

# 2. 추출 픽셀 좌표를 로봇의 작업 공간 좌표로 스케일 조정
def scale_points(points):
    x_min, x_max = np.min(points[:,0]), np.max(points[:,0])  # x 좌표 범위 계산
    y_min, y_max = np.min(points[:,1]), np.max(points[:,1])  # y 좌표 범위 계산
    scale_x = (577 - 280) / (x_max - x_min)  # X축 스케일 비율
    scale_y = (220 + 155) / (y_max - y_min)  # Y축 스케일 비율
    scale = min(scale_x, scale_y)  # 둘 중 더 작은 값 사용 (비율 유지)
    points[:,0] = (points[:,0] - x_min) * scale + 280  # X 좌표 변환
    points[:,1] = (points[:,1] - y_min) * scale - 155  # Y 좌표 변환
    return points

# 3. 좌표 추출부터 시각화, CSV 저장까지 전체 흐름 담당
def start_welding():
    path = entry_path.get()  # 이미지 경로 호출
    points = extract_points(path)  # 경로 좌표 추출
    scaled = scale_points(points)  # 스케일 조정
    sampled = scaled[np.argsort(scaled[:,0])][::10]  # X축 기준 정렬 후 간격 추출
    with open('weld_points.csv', 'w', newline='', encoding='utf-8-sig') as f:
        csv.writer(f).writerows(sampled)  # CSV로 저장
    ax.clear()  # 기존 그래프 초기화
    ax.scatter(sampled[:,0], sampled[:,1], c='red', s=10)  # 산점도 시각화
    canvas.draw()  # 캔버스 업데이트
```

```python

# 5. 저장된 좌표를 따라 협동로봇을 자동 이동시키는 함수
def execute_robot():
    import rclpy
    import DR_init
    from DSR_ROBOT2 import movej, movel, set_tool, set_tcp

    rclpy.init()  # ROS2 초기화
    DR_init.__dsr__id = DR_init.__dsr__model = 'dsr01'  # 로봇 ID 설정
    DR_init.__dsr__node = rclpy.create_node('move_node', namespace='dsr01')  # ROS2 노드 생성
    set_tool("Tool Weight_2FG")  # 툴 설정
    set_tcp("2FG_TCP")  # TCP 설정

    # 좌표 파일 로드
    with open('weld_points.csv', newline='', encoding='utf-8-sig') as f:
        data = list(csv.reader(f))[1:]  # 헤더 제외하고 읽기

    movej([0,0,90,0,90,0], 30, 30)  # 초기 자세로 이동

    for x, y in data:
        pos = [float(x), float(y), 17.0, 151.75, 179.00, 151.09]  # z, rx, ry, rz 고정값 포함
        movel(pos, 60, 60)  # 선형 이동 수행

    movej([0,0,90,0,90,0], 30, 30)  # 종료 후 복귀 자세
    rclpy.shutdown()  # ROS 종료

```

프로젝트 수행 경과

규칙적인 3D 용접

point 3개 캡처 ← **point1(끝점), point2(중간점), point3(시작점)**

↓

원호의 중심 및 반지름 계산

↓

좌표 CSV 저장

↓

CSV 좌표 불러오기

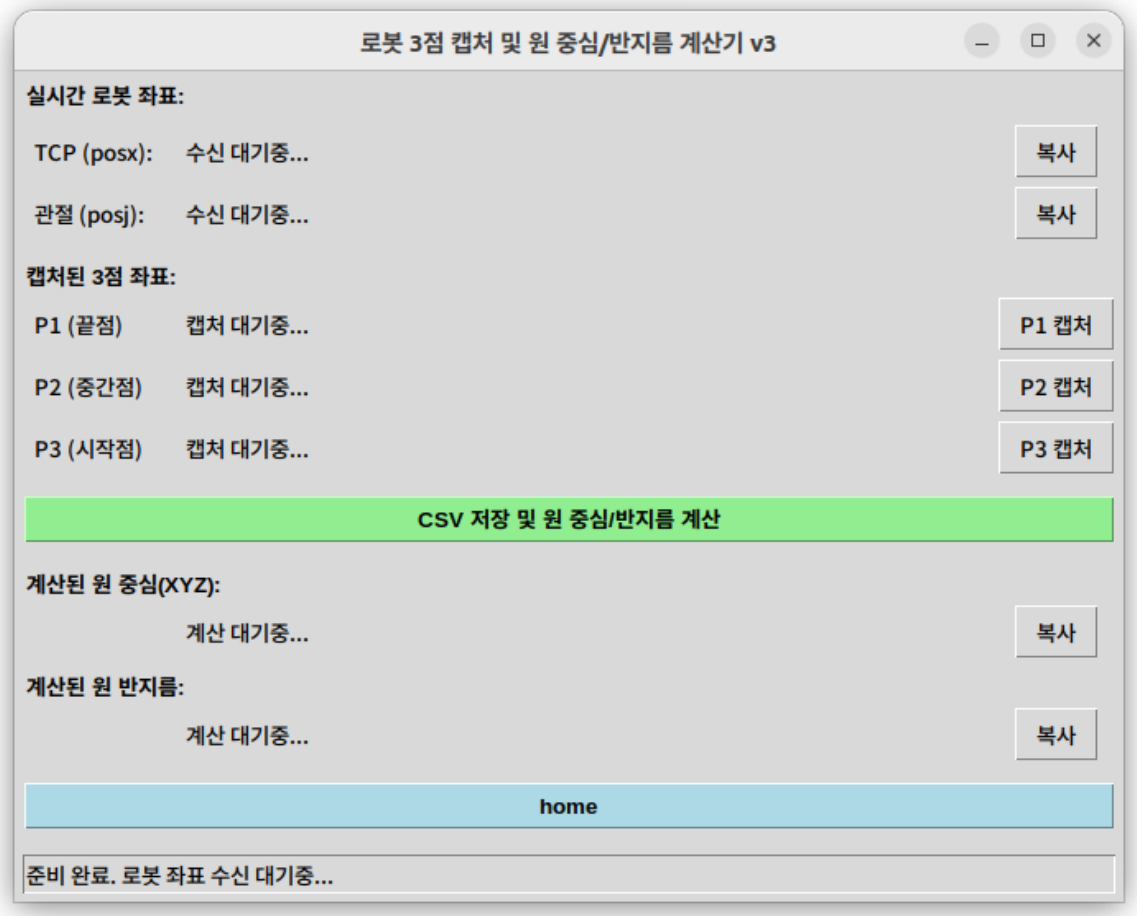
↓

CSV 기반 로봇 이동 실행

프로젝트 수행 경과

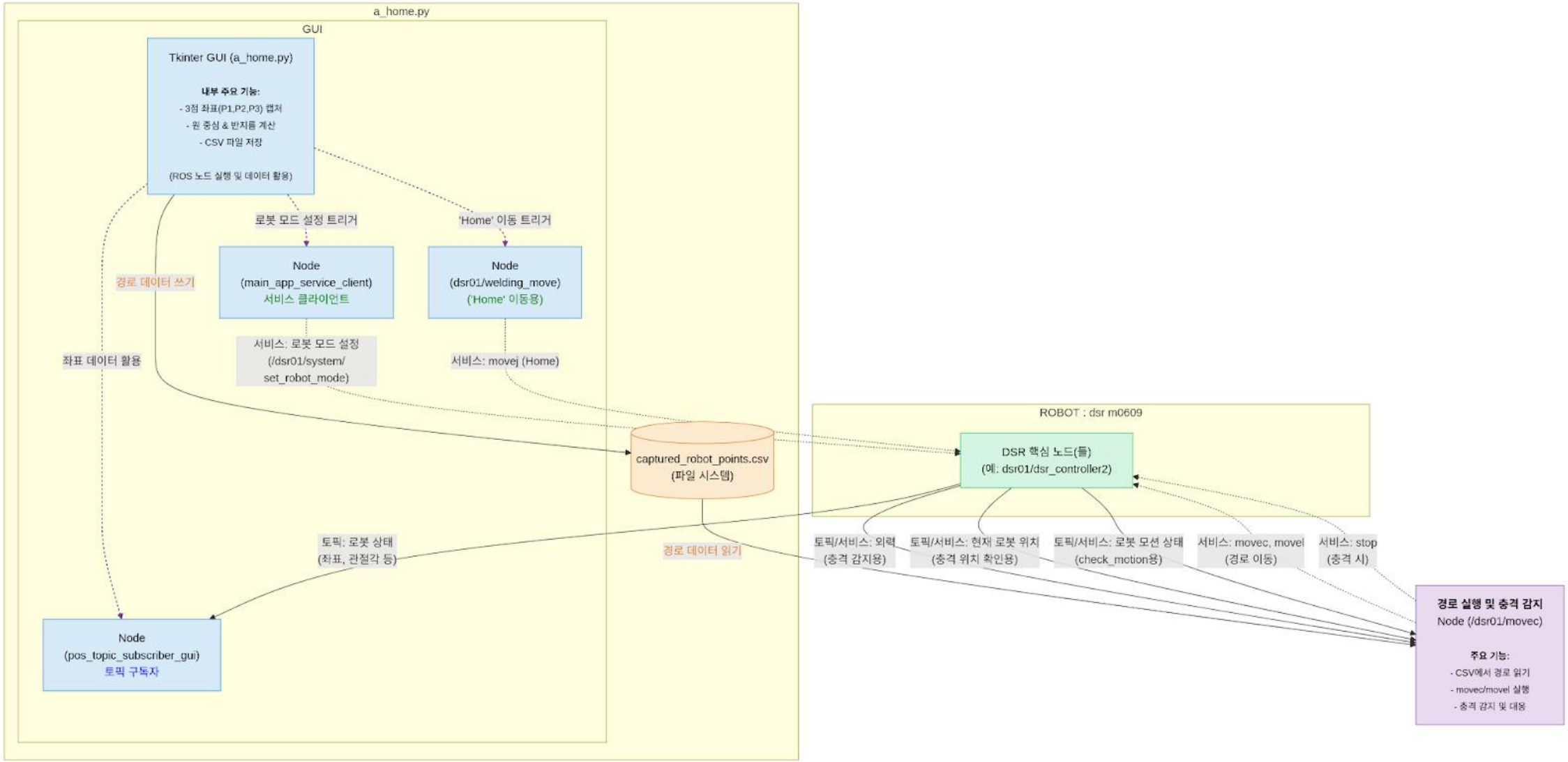
규칙적인 3D 용접

- GUI 실행으로 실시간 로봇 좌표 확인
- 용접 접합부 위치 직접교시를 위해 GUI 실행과 동시에 수동모드 활성화
- 캡처 버튼을 통해 좌표값 기록
- 버튼을 통해 좌표값 CSV 저장 및 원 중심/ 반지름 계산, 홈 정렬 기능
- 개별적 movec 노드를 통해 동작 (P3 -> P1 사이클 타임 단축)



<a_home gui tkinter>

노드 통신 구조


a_home.py
GUI
Tkinter GUI (a_home.py)
내부 주요 기능:
- 3점 좌표(P1,P2,P3) 캡처
- 원 중심 & 반지름 계산
- CSV 파일 저장
(ROS 노드 실행 및 데이터 활용)
로봇 모드 설정 트리거
'Home' 이동 트리거
경로 데이터 쓰기
Node
(main_app_service_client)
서비스 클라이언트
Node
(dsr01/welding_move)
('Home' 이동용)
좌표 데이터 활용
서비스: 로봇 모드 설정
(/dsr01/system/
set_robot_mode)
서비스: movej (Home)
captured_robot_points.csv
(파일 시스템)
ROBOT : dsr m0609
DSR 핵심 노드(들)
(예: dsr01/dsr_controller2)
토픽: 로봇 상태
(좌표, 관절각 등)
경로 데이터 읽기
토픽/서비스: 외력
(충격 감지용)
토픽/서비스: 현재 로봇 위치
(충격 위치 확인용)
토픽/서비스: 로봇 모션 상태
(check_motion용)
서비스: movec, movel
(경로 이동)
서비스: stop
(충격 시)
Node
(pos_topic_subscriber_gui)
토픽 구독자
경로 실행 및 충격 감지
Node (/dsr01/movec)
주요 기능:
- CSV에서 경로 읽기
- movec/movel 실행
- 충격 감지 및 대응

<a_home.py 일부 코드>

```
  def set_initial_robot_mode(self): # 로봇 모드 설정 함수
      if self.service_client_node:
          self.update_status("초기 로봇 모드 설정 시도 (예: 수동 모드 0)...")
          future = self.service_client_node.send_robot_mode_request(0)
          if future:
              # 변경점: 비동기 결과 처리를 위해 콜백 추가
              future.add_done_callback(self.robot_mode_set_callback)
              self.update_status("로봇 모드 설정 요청 전송됨. 응답 대기 중...")
          else:
              self.update_status("로봇 모드 설정 서비스 호출에 즉시
실패했습니다.")
      else:
          self.update_status("서비스 클라이언트 노드가 없어 로봇 모드를 설정할 수
없습니다.")
```

```
# get_current_posx() => 단순 반환 함수로 GUI에 수동 업데이트 필요.
# StringVar() =>    값 저장 + 위젯 자동 연결, 실시간 반영 가능
# 따라서 tkinter의 내장 함수 StringVar을 통해 실시간 좌표를 기록
  def capture_point(self, point_index_to_capture): # 변경 없음
      current_posx_string = self.live_posx_display_var.get()
      self.update_status(f"{self.point_labels_text[point_index_to_capture].split(' ')[0]} 캡처 시도...")
      try:
          if not current_posx_string or not current_posx_string.startswith("posx(") or not current_posx_string.endswith(")"):
              raise ValueError("실시간 좌표 문자열 형식이 올바르지 않습니다.")
          coord_list_as_string = current_posx_string[len("posx("):-1].strip('[]')
          if not coord_list_as_string or coord_list_as_string == "수신 대기중...":
              raise ValueError("수신된 유효한 좌표값이 없습니다.")
          coords_as_float = [float(c.strip()) for c in coord_list_as_string.split(',')]
          if len(coords_as_float) != 6:
              raise ValueError(f"6개의 좌표값(posx)이 필요하지만 {len(coords_as_float)}개를 얻었습니다.")
          self.captured_points_posx[point_index_to_capture] = coords_as_float
          display_text = f"posx({[round(c, 3) for c in coords_as_float]})"
          self.captured_point_display_vars[point_index_to_capture].set(display_text)
          self.update_status(f"{self.point_labels_text[point_index_to_capture].split(' ')[0]} 캡처 완료.")
      except ValueError as ve:
          self.update_status(f"좌표 파싱 오류: {ve} (원본: '{current_posx_string}')")
          messagebox.showerror("캡처 오류", f"좌표 파싱 중 오류 발생:\n{ve}\n원본 문자열: '{current_posx_string}'")
      except Exception as e:
          self.update_status(f"캡처 중 알 수 없는 오류: {e} (원본: '{current_posx_string}')")
          messagebox.showerror("캡처 오류", f"알 수 없는 오류 발생:\n{e}")
```

<a_home.py csv 코드>

```
# === 원의 중심 및 반지름 계산 함수 (GUI용) ===
# 변경점: 함수 이름 변경, 반지름 계산 및 반환 추가
def calculate_circle_center_and_radius_for_gui(p1_xyz, p2_xyz, p3_xyz, status_update_func=print): # 이름 변경
    P1 = np.array(p1_xyz, dtype=float)
    P2 = np.array(p2_xyz, dtype=float)
    P3 = np.array(p3_xyz, dtype=float)

    if np.allclose(P1, P2) or np.allclose(P1, P3) or np.allclose(P2, P3):
        status_update_func("오류: 두 개 이상의 점이 거의 동일합니다. 중심/반지름 계산 불가.")
        return None, None # << 변경: 반지름에 대해서도 None 반환

    v_ab = P2 - P1
    v_bc = P3 - P2
    normal_plane = np.cross(v_ab, v_bc)

    if np.linalg.norm(normal_plane) < 1e-7:
        status_update_func("오류: 세 점이 거의 한 직선 위에 있어 중심/반지름 정의 불가.")
        return None, None # << 변경: 반지름에 대해서도 None 반환

    M1 = (P1 + P2) / 2.0
    M2 = (P2 + P3) / 2.0
    n1 = np.cross(normal_plane, v_ab)
    n2 = np.cross(normal_plane, v_bc)

    if np.linalg.norm(n1) < 1e-7 or np.linalg.norm(n2) < 1e-7:
        status_update_func("오류: 수직이등분선 방향 벡터 계산 실패.")
        return None, None # << 변경: 반지름에 대해서도 None 반환

    A_mat = np.column_stack((n1, -n2))
    b_vec = M2 - M1

    try:
        params, _, _, _ = np.linalg.lstsq(A_mat, b_vec, rcond=None)
        t = params[0]
        center = M1 + t * n1

        # >>> 추가된 부분: 반지름 계산 <<<
        radius = np.linalg.norm(center - P1)

        r2 = np.linalg.norm(center - P2)
        r3 = np.linalg.norm(center - P3)
        if not (np.isclose(radius, r2, atol=1e-3) and np.isclose(radius, r3, atol=1e-3)):
            status_update_func(
                f"경고: 중심-각 점 거리 (반지름) 약간 불일치. R1:{radius:.4f}, R2:{r2:.4f}, R3:{r3:.4f}"
            )
        return center.tolist(), radius # << 변경: 중심과 함께 반지름 반환
    except np.linalg.LinAlgError as e:
        status_update_func(f"선형대수 오류: {e}. 점들이 한 직선 위일 가능성.")
        return None, None # << 변경: 반지름에 대해서도 None 반환
    except Exception as e_gen:
        status_update_func(f"중심/반지름 계산 중 일반 오류: {e_gen}")
        return None, None # << 변경: 반지름에 대해서도 None 반환
```

<movec.py 실행 코드>

```python
# CSV 파일에서 좌표 불러오기 (p1: 도착점, p2: 경유점, p3: 시작점)
p1_data, p2_data, p3_data = None, None, None
csv_file_path = "/home/fred/ws_doosan/3_point(center)/captured_robot_points.csv" # 경로를

with open(csv_file_path, mode="r", newline="") as file: # 변수 사용
    reader = csv.reader(file)
    rows = list(reader)
    if len(rows) >= 4:
        try:
            p1_data = list(map(float, rows[1][1:]))
            p2_data = list(map(float, rows[2][1:]))
            p3_data = list(map(float, rows[3][1:]))
            print(p1_data)
            print(p2_data)
            print(p3_data)
        except ValueError:
            if node:
                node.get_logger().error("CSV 파일의 좌표 형식이 잘못되었습니다.")
            else:
                print("CSV 파일의 좌표 형식이 잘못되었습니다.")
            sys.exit(1) # 오류 발생 시 즉시 종료
    else:
        if node:
            node.get_logger().error("CSV에 최소 3개의 포인트가 필요합니다.")
        else:
            print("CSV에 최소 3개의 포인트가 필요합니다.")
        sys.exit(1) # 오류 발생 시 즉시 종료

# posx 객체로 변환
p1_posx = posx(*p1_data)
p2_posx = posx(*p2_data)
p3_posx = posx(*p3_data)
```

```python
        # posx 객체로 변환
        p1_posx = posx(*p1_data)
        p2_posx = posx(*p2_data)
        p3_posx = posx(*p3_data)


        # 1. 시작점(P3)으로 직선 이동
        print("시작점(P3)으로 이동 중...")
        movel(p3_posx, VELOCITY, ACC)
        sleep(1)
        # 2. 원호 경로 이동 (P2 경유 → P1 도착)
        # 실제 경로는: 현재 위치(P3) -> P2 -> P1
        print("movec 실행 중 (현재위치 → P2 경유 → P1 도착)...")
        movec(p2_posx, p1_posx, VELOCITY, ACC)
        print(check_motion())
        print("완료")
        # movec 변수 중 ori=DR_MV_ORI_RADIAL 를 통해 원주구속자세 기능을 ROS에선 지원하지않아
        # xz,yz 평면에서의 원호를 그리기 힘듬. xy평면에 대해서 작동예정.
        sleep(1)
        print("movec 동작 완료.")


    except Exception as e:
        if node:
            node.get_logger().error(f"로봇 동작 중 예외 발생: {e}")
        else:
            print(f"로봇 동작 중 예외 발생: {e}")
    finally:
        if node:
            node.destroy_node()
            print("ROS2 노드 종료.")
        rclpy.shutdown()
        print("ROS2 rclpy 종료.")


if __name__ == '__main__':
    main()
```

프로젝트 수행 경과

* 결과 제시 ④ 모델 평가 및 개선

Welding Safety for Human-Robot Collaboration(안전기능)

- 충돌 대응 기능 통합
  - 경고음 재생 + 팝업 알림(정지위치좌표출력)

Monitor_impact()에 통합

->충돌 시 작업자 인지 및 로봇 피해 방지

동시 실현

<soundplay code.py>

```python
import tkinter as tk                    # 팝업 GUI를 위한 tkinter 불러오기
from tkinter import messagebox          # 메시지 박스(경고창) 모듈
import sounddevice as sd                # 소리 재생용 모듈
import numpy as np                      # 사인파 생성을 위한 수치 계산 모듈
import threading                        # 경고음 중복 방지를 위한 스레드 처리

#  경고음 재생 상태를 저장하는 플래그 (중복 재생 방지용)
warning_sound_playing = False

#  경고음을 사인파로 생성하여 재생하는 함수
def play_sine_beep():
    global warning_sound_playing
    if warning_sound_playing:
        return  # 이미 재생 중이면 중복 방지

    warning_sound_playing = True
    try:
        fs = 44100              # 샘플레이트 (Hz)
        duration = 0.5          # 지속 시간 (초)
        frequency = 1000.0      # 사운드 주파수 (Hz)
        t = np.linspace(0, duration, int(fs * duration), endpoint=False)
        wave = np.sin(2 * np.pi * frequency * t).astype(np.float32)
        sd.play(wave, samplerate=fs)  # 사인파 재생
        sd.wait()                     # 재생이 끝날 때까지 대기
    finally:
        warning_sound_playing = False  # 플래그 초기화
```

<Tk code.py>

```python
#  팝업 메시지를 띄우고, 동시에 경고음을 재생하는 함수
def show_impact_popup(pos):
    # 사운드는 백그라운드 스레드로 실행 (팝업 차단 방지)
    threading.Thread(target=play_sine_beep, daemon=True).start()

    # 팝업 창 생성 및 메시지 표시
    popup = tk.Tk()
    popup.withdraw()  # 메인 윈도우 숨기기 (경고창만 띄우기)
    msg = f"⚠️ 충격이 감지되었습니다!\n위치: {pos}"  # 메시지 내용
    messagebox.showwarning("충격 감지", msg)  # 경고창 띄우기
    popup.destroy()  # 팝업 창 닫기
```

rokey@rokey-550XBE-350XBE: ~/ros2_ws

rokey@rokey-550XBE-350XBE: ~/ros2_ws 163x16

용접 경로 추출기

이미지 선택

용접 포인트 추출

용접 시작

충격 감지

충격이 감지되었습니다!
위치: ([399.4097900390625,
25.724714279174805,
54.64097595214844,
151.01669311523438,
179.0076446533203,
150.38064575195312], 2)

OK

rokey@rokey-550XBE-350XBE: ~/ros2_ws 81x15

35: 이동 -> [399.25577, 25.85358, 54.56, 151.75, 179.0, 151.09]
충격 감지됨! 위치: ([399.4097900390625, 25.724714279174805, 54.64097595214844,
151.01669311523438, 179.0076446533203, 150.38064575195312], 2)

BE:~/ros2_ws$ ros2 run rokey test_safe
call last):
2_ws/install/rokey/lib/rokey/test_safe", line 33, in <modu
_point('rokey==0.0.0', 'console_scripts', 'test_safe')())
2_ws/install/rokey/lib/rokey/test_safe", line 25, in impor
return next(matches).load()
StopIteration
[ros2run]: Process exited with failure 1
rokey@rokey-550XBE-350XBE:~/ros2_ws$

05 K-Digital Training

자체 평가

평가 의견과 느낀 점

프로젝트 결과물에 대한 완성도 평가

9/10

잘한 부분과 아쉬운 점

잘한 부분 : 구상했던 내용을 전부 구현, 프로젝트 결과물을 GUI로 표현
아쉬운 점 : 원주방향 모드 미지원으로, XY 평면상의 원호 궤적으로 계획을 전환했다. Tkinter와 스레드 간 충돌로 통합 구현에 제약이 있었다.

보완할 점

보완할 점 : 용접 결과를 기반으로 로봇이 스스로 학습하고 재현할 수 있는 구조가 필요하다.

느낀 점이나 경험한 성과

느낀점 : 코드 구조의 차이로 어려움이 있었지만, 소통과 조율을 통해 각자의 강점을 융합하며 협업과 팀워크의 중요성을 실감했다.

협동로봇 기반 용접 후 품질 검사 (Machine Vision 적용)

- 구성 요소
- **로봇팔 + 카메라 (산업용 or 3D)**
- **비전 소프트웨어 (OpenCV + AI 모델)**
- **딥러닝 기반 결함 판단 모델 (CNN)**
- **판정 기준 알고리즘 (길이, 너비, 비드 일관성 등)**

검사 항목	기준	문제 예시
비드 폭	±0.5 mm	비정상적으로 좁거나 넓은 비드
비드 높이	±0.2 mm	비드가 너무 높아진 경우
비정상 아크 흔적	있음/없음	스패터 과다, 비정상 패턴
크랙, 핀홀	없음	용접 결함

출처

사진출처 :
파이썬 : https://www.python.org/
OPENCV :https://opencv.org/
ROS2 : https://www.ros.org/
Doosan M0609 :https://www.doosanrobotics.com/kr/product-solutions/product/m-series/m0609/
Visual Studio Code : https://code.visualstudio.com/
자료 출처 :
https://www.osha.com/blog/how-to-avoid-welding-hazards?utmcom
https://www.3m.com/blog/en_US/safety-now/science-of-safety/quality-products/respiratory-protection/understanding-the-hazards-of-welding-in-construction-and-the-ppe-available/?utm.com
https://www.cdc.gov/niosh/welding/about/index.html?utm_com