C심화과정 프로젝트 (카메라를 활용한 로봇 시뮬레이션) 결과보고서

1. 서론

1.1 요구사항

카메라 calibration 이용하여 Scale 값을 추출 후 가상의 작업대 공간의 물체를 UNIVERSAL ROBOTS UR5e 제어하여 가상의 도형의 중심좌표로 이동시키고 다른 작업대 공간으로 이동하는 시뮬레이션 시스템

1.2 과제설계의 목표

C++ MFC를 이용하여 사용자 UI 및 로봇 제어프로그램 설계 및 구현 (Client)
Delfoi Premium 4.1 프로그램을 이용하여 가상로봇 시뮬레이션 (Server)

1.3 현실적 제한 요건

1) 카메라 측정에서 환경적인 요인 문제
2) 정확한 좌표값에 대한 보정작업이 필요
   ex) 원하는 위치의 좌표값과 실제 동작 시 움직여간 위치값을 보정필요
3) 다른 작업대 위치 좌표는 임의 값으로 세팅

2. 본론

2.1 문제 정의

로봇 시스템에 범용적인 기구학 적용하여 위치제어 시스템 및 MFC를 활용하여 도형의 좌표값 이랑 동일하게 로봇의 위치가 적용되는지 보기 위한 시스템 개발

2.2 개념설계

- 계발 환경

구분	개발 환경
운영체제	Windows 10
언어	C++ / Python
로봇 시뮬레이터	Delfoi Premium 4.1
개발툴	Microsoft Visual Studio 2017
장비	웹캠

- 목표 시스템 입력 및 출력

1) MFC

- 스케일 버튼 클릭시 스케일 값 측정 및 intricsics.yml파일 저장
- 도형중심좌표 검출 버튼 클릭시 도형중심 좌표 검출 및 centerScalePoint.yml파일 저장
- 서버연결 버튼 클릭시 Delfoi 서버로 접속
- 도형이동위치 버튼 클릭시 조인트 (J1 ~ J6) Delfoi로 각도 전달
- 도형좌표 Listbox 도형중심 좌표 표기 (스케일 값을 곱한 실제 좌표)
- 목표좌표 Listbox 목표좌표 표기 (도형이 있는 위치의 좌표)
- 조인트각도 Listbox 조인트 각도 표기 및 델포이 연결상태 표시

2.3 설계 제작 과정

- 기능 요구사항 (기능 리스트)

유형	기능		
식별자	MFC_스케일 버튼	요구사항 명	스케일 측정
개요	스케일 값 측정 및 저장		
상세설명	1. 스케일 값 측정 2. 값 저장		

유형	기능		
식별자	MFC_도형중심좌표검출버튼	요구사항 명	중심좌표검출
개요	도형 중심좌표 검출 및 실제 좌표계 변환		
상세설명	1. 도형 외각선 검출 및 중심점 검출 2. 도형 중심좌표 검출 3. 중심좌표 실제 좌표계 계산 4. 계산된 좌표계 값 저장		

유형	기능		
식별자	MFC_서버연결 버튼	요구사항 명	서버연결
개요	Delfoi 서버 연결		
상세설명	1. IP 입력 2. 서버 연결		

유형	기능		
식별자	MFC_도형좌표 List Box	요구사항 명	도형좌표 표기
개요	도형 실제 좌표 표기		
상세설명	1. 1번도형 좌표위치 표기 2. 2번도형 좌표위치 표기 3. 3번도형 좌표위치 표기		

유형	기능		
식별자	MFC_목표좌표 List Box	요구사항 명	목표 좌표 표기
개요	목표좌표 표기		
상세설명	1. 목표좌표		

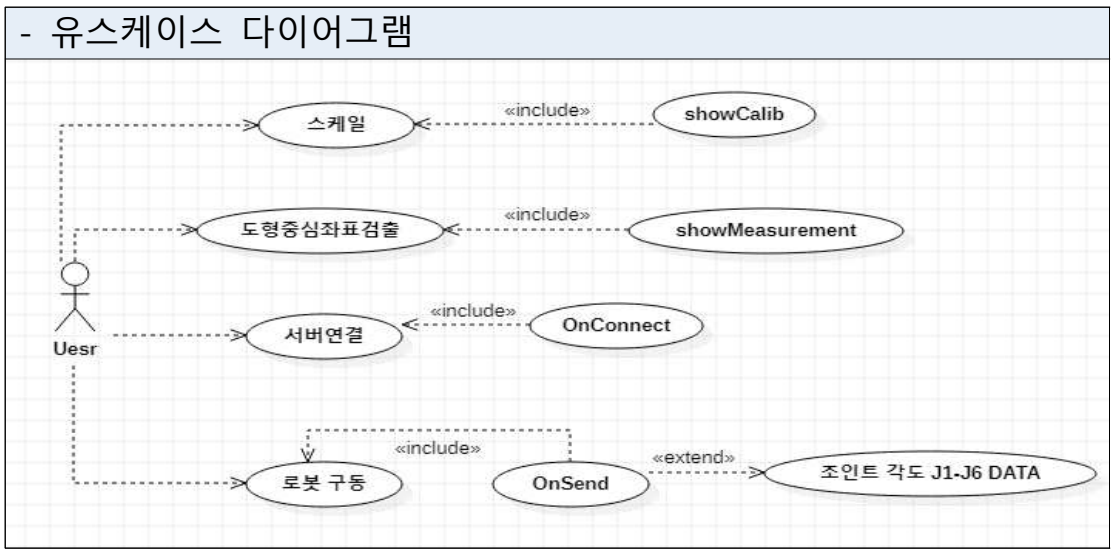
유형	기능		
식별자	MFC_조인트 각도 List Box	요구사항 명	조인트 각도 출력
개요	조인트 각도 및 Delfoi 연결 여부 표기		
상세설명	1. 조인트 각도 표기 2. Delfoi 여부 표기		

유형	기능		
식별자	MFC_도형1위치로이동 버튼	요구사항 명	조인트 각도 전달
개요	도형1 위치로 이동 버튼		
상세설명	1. Delfoi로 도형1위치 좌표 계산된 조인트 각도 전달		

유형	기능		
식별자	MFC_도형2위치로이동 버튼	요구사항 명	조인트 각도 전달
개요	도형2 위치로 이동 버튼		
상세설명	1. Delfoi로 도형2위치 좌표 계산된 조인트 각도 전달		

유형	기능		
식별자	MFC_도형3위치로이동 버튼	요구사항 명	조인트 각도 전달
개요	도형3 위치로 이동 버튼		
상세설명	1. Delfoi로 도형3위치 좌표 계산된 조인트 각도 전달		

- 유스케이스 다이어그램



- 유스케이스 명세

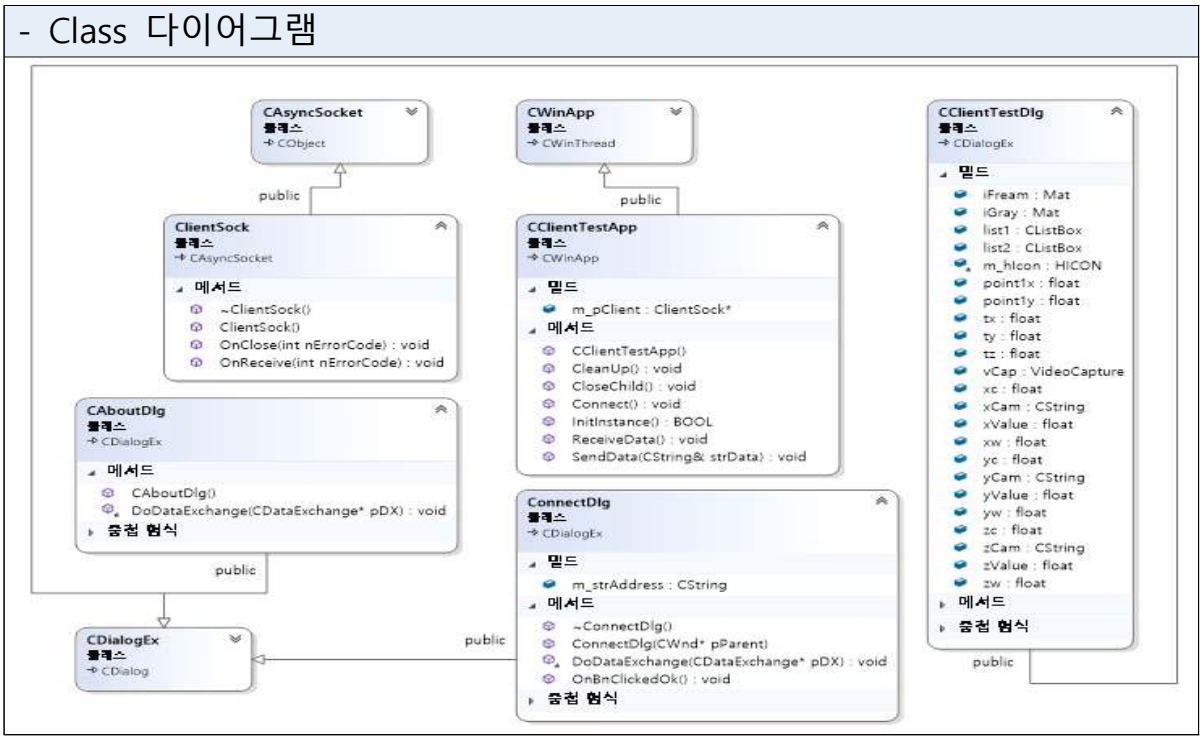
Use case Name	스케일
Use case Description	이 Use case는 사용자가 스케일 버튼 클릭 시 호출
Primary Acrtor	사용자
Goal	사용자가 스케일 버튼 클릭 시 동작 수행
Basic Flow	1. 사용자가 스케일 버튼 클릭 2. 영상 프레임 출력 3. 영상 프레임에 스케일 값 표기 4. 영상 프레임 종료시 스케일 값 저장

Use case Name	도형중심좌표검출
Use case Description	이 Use case는 사용자가 도형중심좌표검출 버튼 클릭 시 호출
Primary Acrtor	사용자
Goal	사용자가 도형중심좌표검출 버튼 클릭 시 동작 수행
Basic Flow	1. 사용자가 도형중심좌표검출 버튼 클릭 2. 영상 프레임 출력 3. 영상 프레임에 도형중심좌표 값 표기 4. 영상 프레임 종료시 도형중심좌표검출 값 저장

Use case Name	서버연결
Use case Description	이 Use case는 사용자가 서버연결 버튼 클릭 시 호출
Primary Acrtor	사용자
Goal	사용자가 서버연결 버튼 클릭 시 동작 수행
Basic Flow	1. 사용자가 서버연결 버튼 클릭 2. IP 입력 3. 연결 버튼 클릭 시 서버 연결

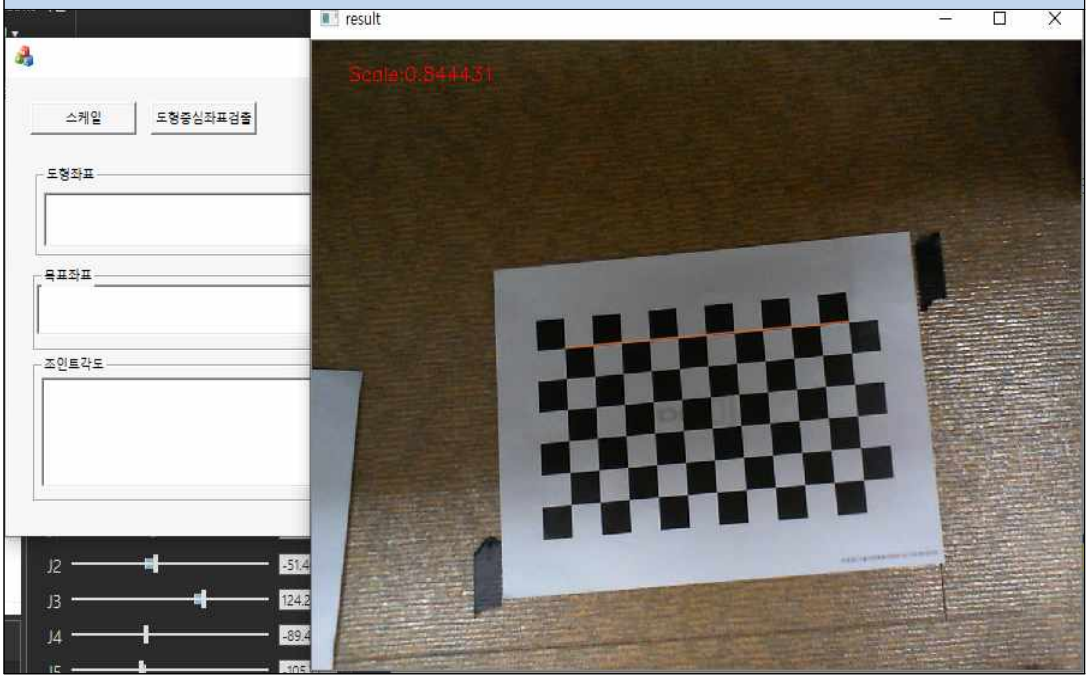
Use case Name	로봇 구동(도형위치버튼)
Use case Description	이 Use case는 사용자가 로봇 구동 버튼 클릭 시 호출
Primary Acrtor	사용자
Goal	사용자가 로봇 구동 버튼 클릭 시 동작 수행
Basic Flow	1. Delfoi로 (각 J01 ~ J06 각도) 값 전달

- Class 다이어그램

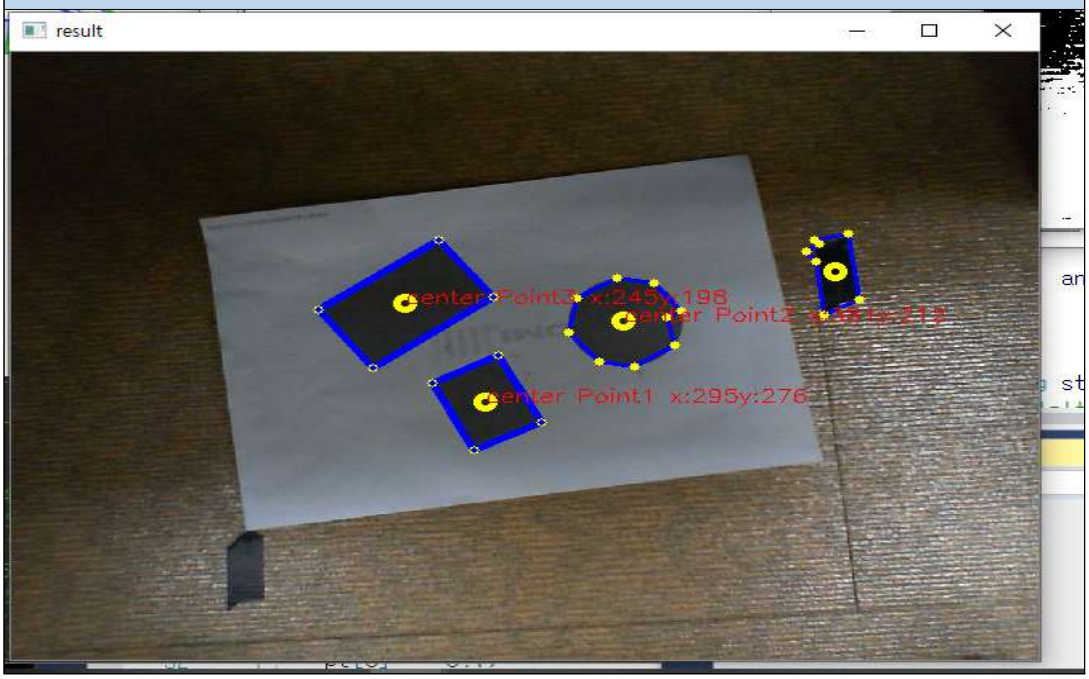


- 사용자 인터페이스

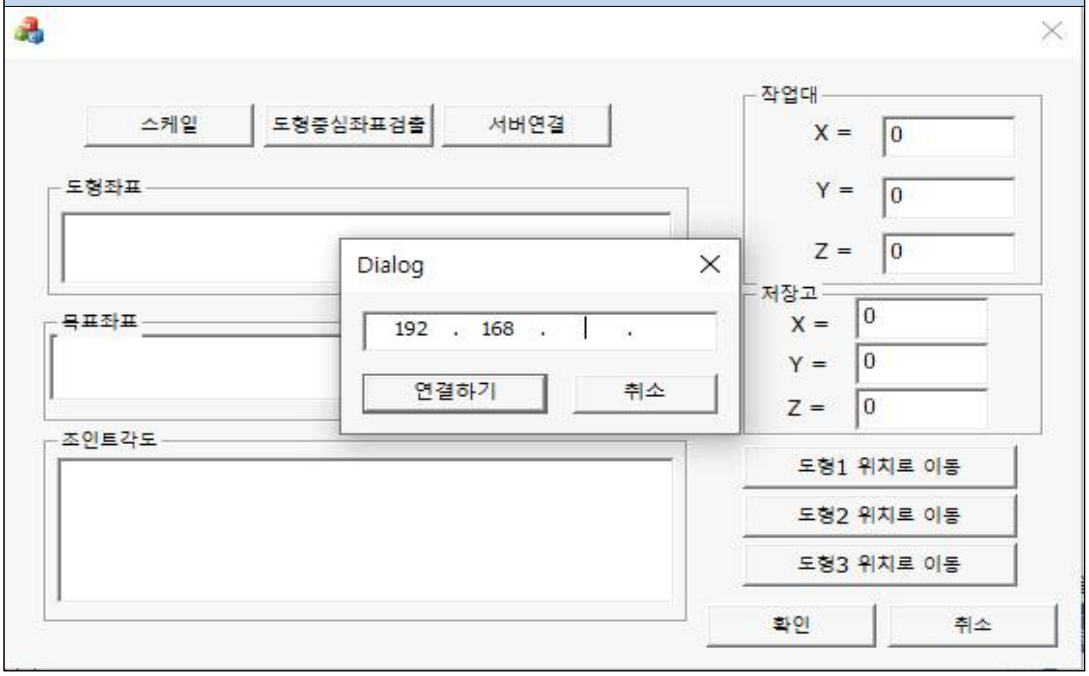
- 스케일 버튼 클릭시



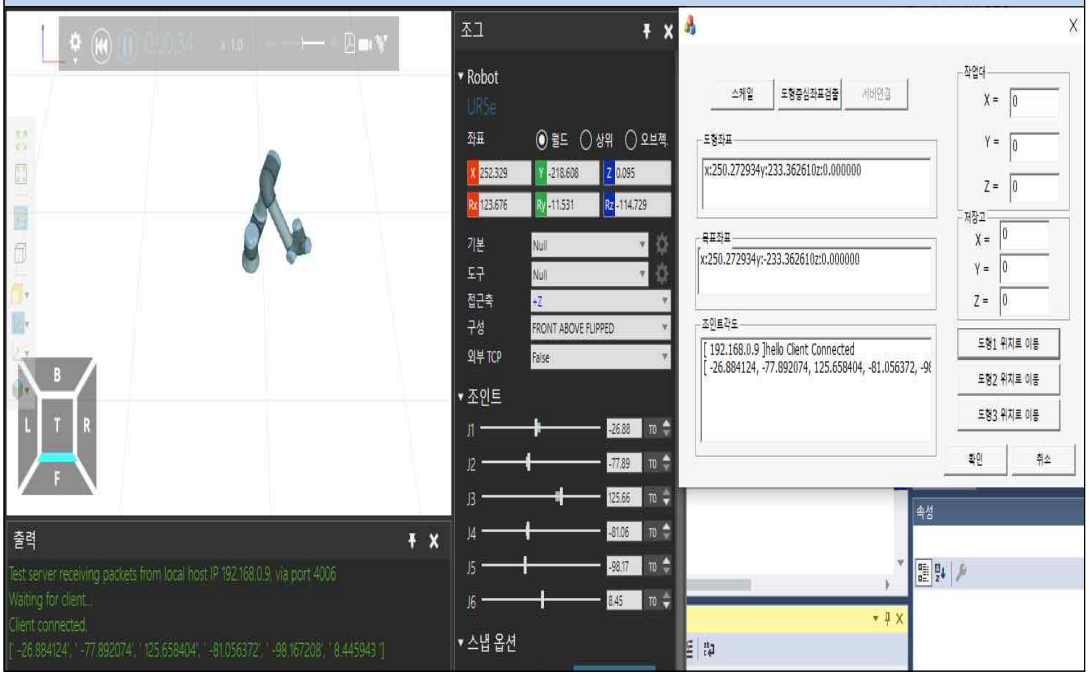
- 도형중심좌표 검출 버튼 클릭시



- 서버연결 버튼 클릭시



- 로봇구동 버튼 클릭시 (도형 1번 클릭시 1번위치로 이동)



- 프로그램 사용방법

1. MFC 및 Delfoi Premium 4.1 를 이용하여 로봇 제어사용방법

1) Delfoi Premium 4.1 실행 하여 로봇중에 "UNIVERSAL ROBOTS UR5e" 선택
2) 오프셋 값 설정 (오프셋 설정 화면 보고 기록하기)
3) 파이선 스크립트 추가 (서버)
4) Delfoi Premium 4.1 프로그램 실행
5) MFC 실행 (클라이언트)

* 참고 (저장고 구역에 물체를 놓아야하는 위치는 임의로 정해두었습니다)

5-1) 서버연결 버튼 클릭 "본인 컴퓨터 IP입력"연결시도
5-2) IP검색 방법 CMD창에서 ipconfig 입력 해서 IP확인체크 필요
5-3) 작업대 X Y Z 좌표 입력 / 저장고 X Y Z 좌표 입력
5-4) 로봇 구동 버튼 실행
5-5) 각 List box에서 사용자가 목표로하는 위치값 및 조인트 각도, 도형의 위치 값 확인가능

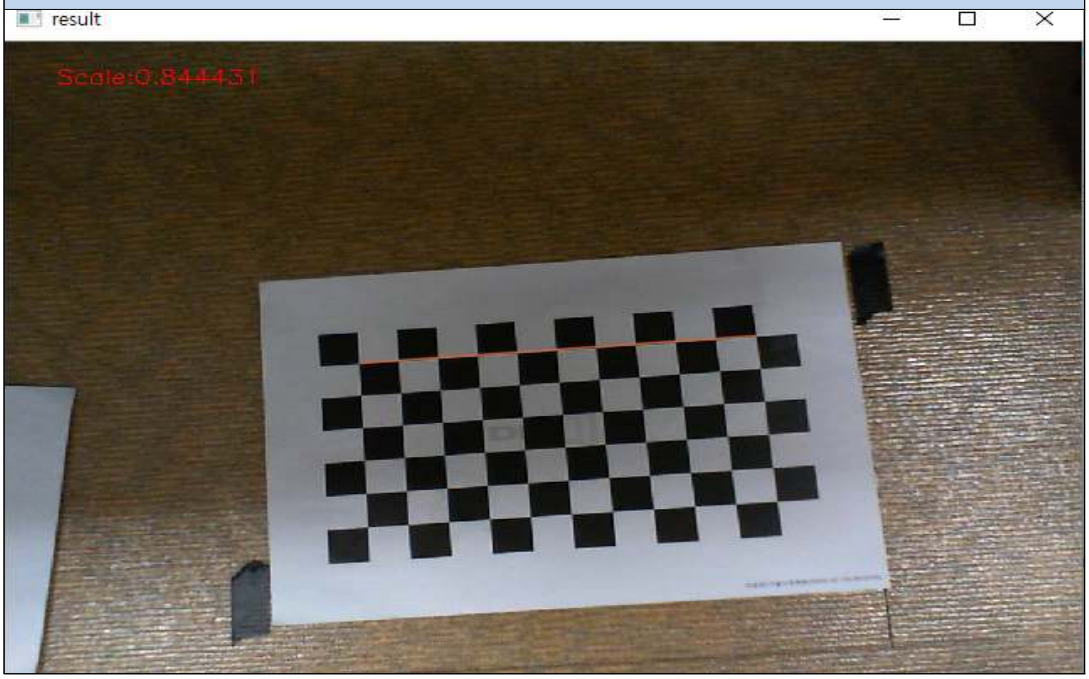
- 주요 함수 기능 (스케일 값 추출, 외각선 검출, 중심점 좌표 검출, 목표지점)

[스케일 값 구하는 식]

- (0,0) 좌표 x y / (10,10) 좌표 x y 추출
- 두 점사이의 거리를 구하는 공식을 이용하여 d를 구함
- (0,0) ~ (10,10) 까지의 실제 거리 200mm
- D = 실제거리 = 200 / d(두점사이의 거리)
- D = 실제 한칸의 거리 값이 추출 및 저장

```
// 코너 x1 , x2 값 추출
int x1 = corners[0].x;
int y1 = corners[0].y;
int x2 = corners[10].x;
int y2 = corners[10].y;

pt1.x = x1;
pt1.y = y1;
pt2.x = x2;
pt2.y = y2;

// x1,y1 x2,y2 사이의 거리 구하기 픽셀상
float d = sqrt(pow((x1 - x2), 2) + pow((y1 - y2), 2));

// 0.0 ~ 10.10 까지 실거리 측정하면 200MM

// 스케일 10칸에 한 실제 거리값
float a = 200; // 실거리 측정값

// 실거리 측정값 / 픽셀상의 거리 나누어서 스케일 값 구하기
D = a / d;

// 스케일 값을 이용하여 거리 측정 = 스케일 (D1)
D1 = d * D;
```

- 스케일 값 0.84431 측정 결과



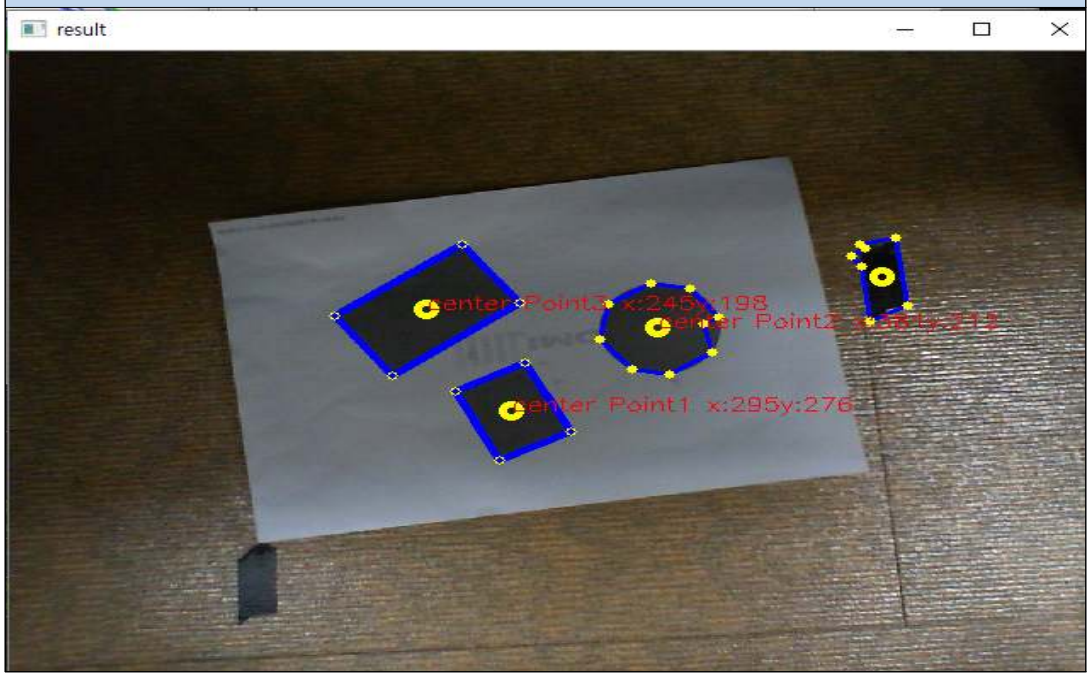
[중심점 좌표 검출 및 월드 좌표계 변환 방법]

- 센터 중심 좌표를 vector<Point> center1 저장
- center1.push_back(center) 값을 push_back 저장
- 스케일 값을 가져와서 각 도형의 중심좌표 X Y값에 곱해서 (실제 좌표계로 변환)
- 실제 좌표계 값을 파일로 저장

vector<Point> center1;
center1.push_back(center);

```
// 영상이 출력될때 처음 백터에 쓰레기 값이 들어가는 경우가 발생하여 딱 3개 잡은경우만 백터에 넣는 방식
if (contours.size() > 3) {
    pt1 = center1[0];
    pt2 = center1[1];
    pt3 = center1[2];

    // 스케일 값 가져오기
    FileStorage fsFrontRead("intricsics.yml", cv::FileStorage::READ);
    float scaleReuslt;
    fsFrontRead["Scale"] >> scaleReuslt;
    fsFrontRead.release();

    //text = "scale x:" + to_string(scaleReuslt);
    //putText(dst, text, Point(100, 100), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255), 1);
    text = "center Point1 x:" + to_string(pt1.x) + "y:" + to_string(pt1.y);
    putText(dst, text, pt1, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255), 1);
    text = "center Point2 x:" + to_string(pt2.x) + "y:" + to_string(pt2.y);
    putText(dst, text, pt2, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255), 1);
    text = "center Point3 x:" + to_string(pt3.x) + "y:" + to_string(pt3.y);
    putText(dst, text, pt3, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255), 1);

    // 픽셀상의 중심 좌표를 실제 xy 값으로 변환 데이터 저장
    FileStorage fsStorage;
    fsStorage.open("ceterScalePoint.yml", FileStorage::WRITE);
    fsStorage << "Point1x" << pt1.x * scaleReuslt;
    fsStorage << "Point1y" << pt1.y * scaleReuslt;
    fsStorage << "Point2x" << pt2.x * scaleReuslt;
    fsStorage << "Point2y" << pt2.y * scaleReuslt;
    fsStorage << "Point3x" << pt3.x * scaleReuslt;
    fsStorage << "Point3y" << pt3.y * scaleReuslt;

    fsStorage.release();
```

- 중심 좌표 측정 결과



[원하는 위치의 목표 값을 구하기위한 식]

MATLAB을 이용하여 월드 좌표계 와 카메라 좌표 사이의 변환 행렬

- x축 으로 180도 회전 변환

$$\begin{bmatrix} Xw \\ Yw \\ Zw \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & \cos\theta & -\sin\theta & ty \\ 0 & \sin\theta & \cos\theta & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Xc \\ Yc \\ Zc \\ 1 \end{bmatrix}$$

다음과 같이 식을 풀면 밑에 식이완성됩니다.

```
xw = tx + xc;
yw = ty + yc * cos(180 * CV_PI / 180) - zc * sin(180 * CV_PI / 180);
zw = tz + zc * cos(180 * CV_PI / 180) + yc * sin(180 * CV_PI / 180);
```

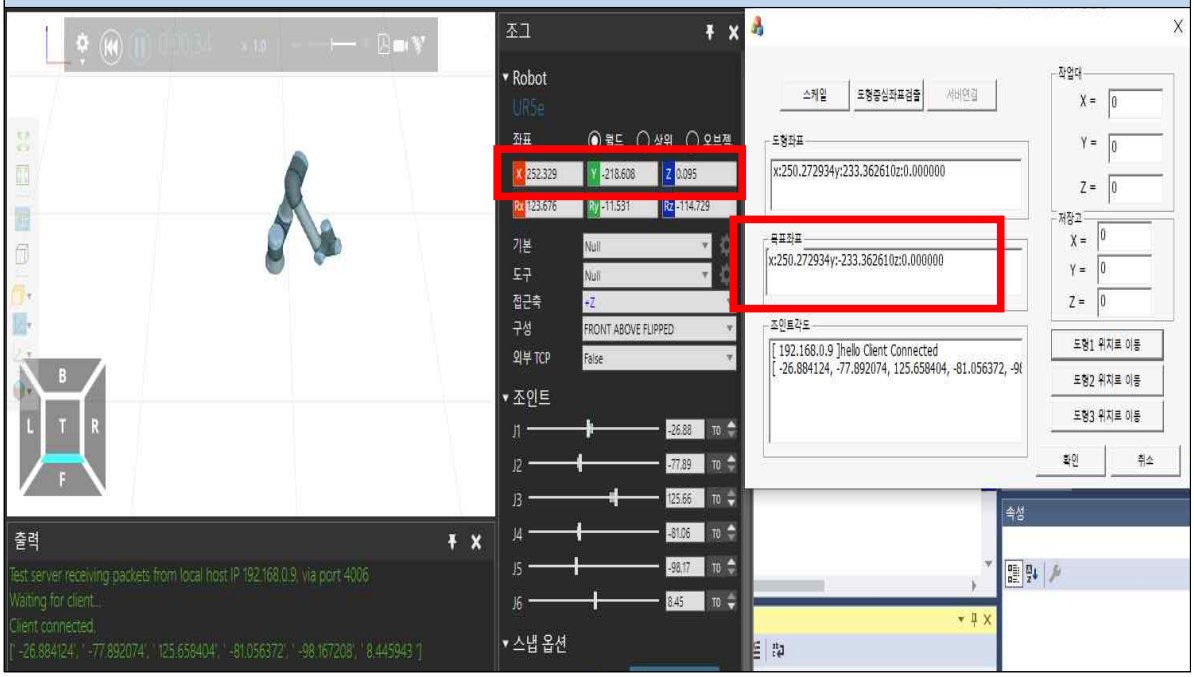
Xw, Yw, Zw (목표로 하는 위치좌표값)

Tx, Ty, Tz (작업대 위치, 저장고 위치 좌표값)

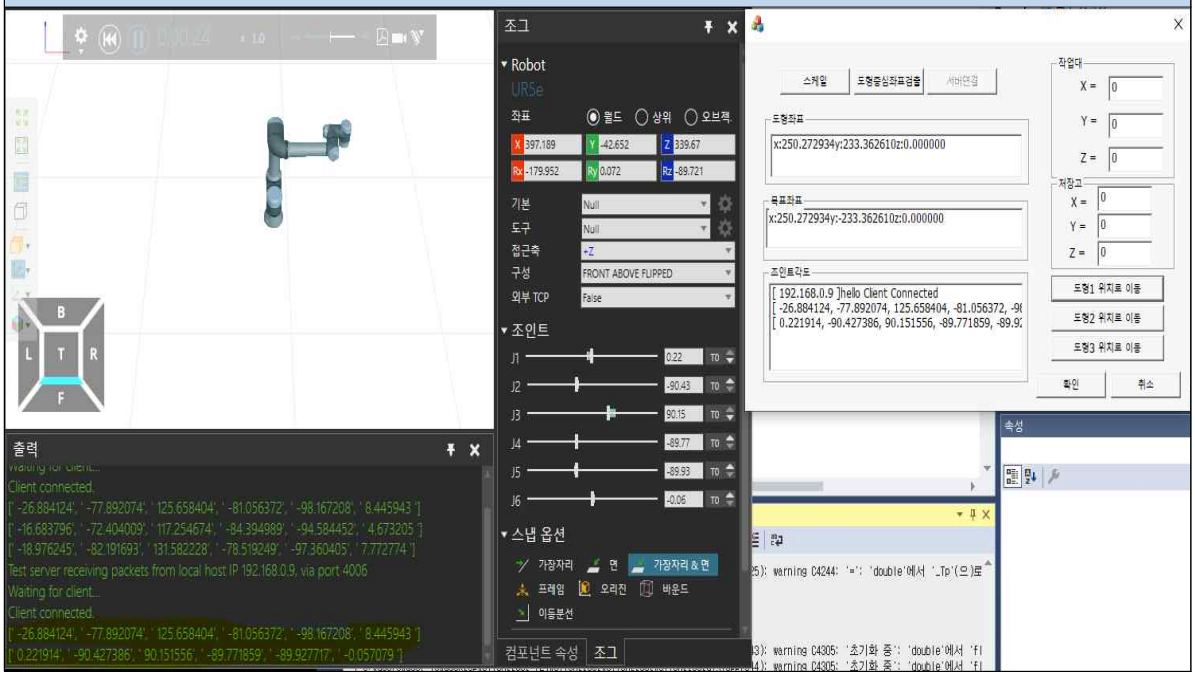
Xc, Yc, Zc (물체 도형 중심 좌표)

자코비안식을 이용하여 각도를 구하게 되고 이 각도로 원하는 위치로 이동

- 작업대 도형1번 좌표 위치로 이용하는 사진 (도형1번)

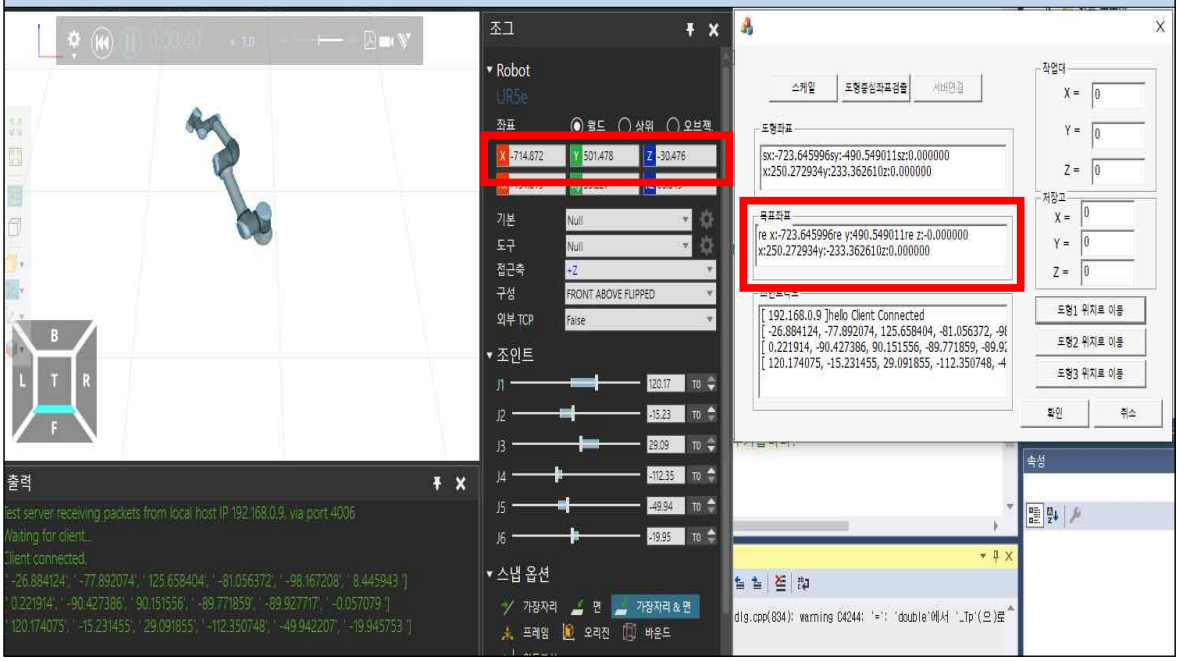


- 물체를 찍고 원래 자세로 돌아가는 값 사진 (도형1번)

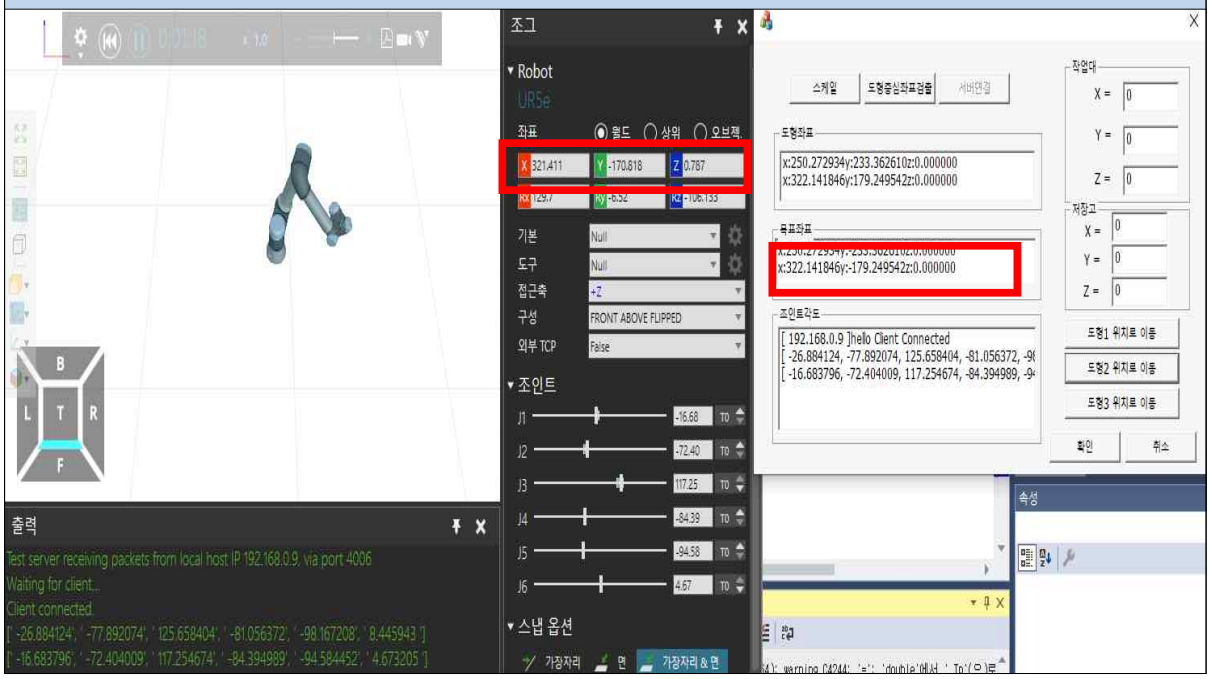


- 저장고 위치 원하는 임의 좌표 x -723 y 490.549011 z 0.0

- 저장고 위치에서 원하는 좌표에 물체 놓는 사진 (도형1번)

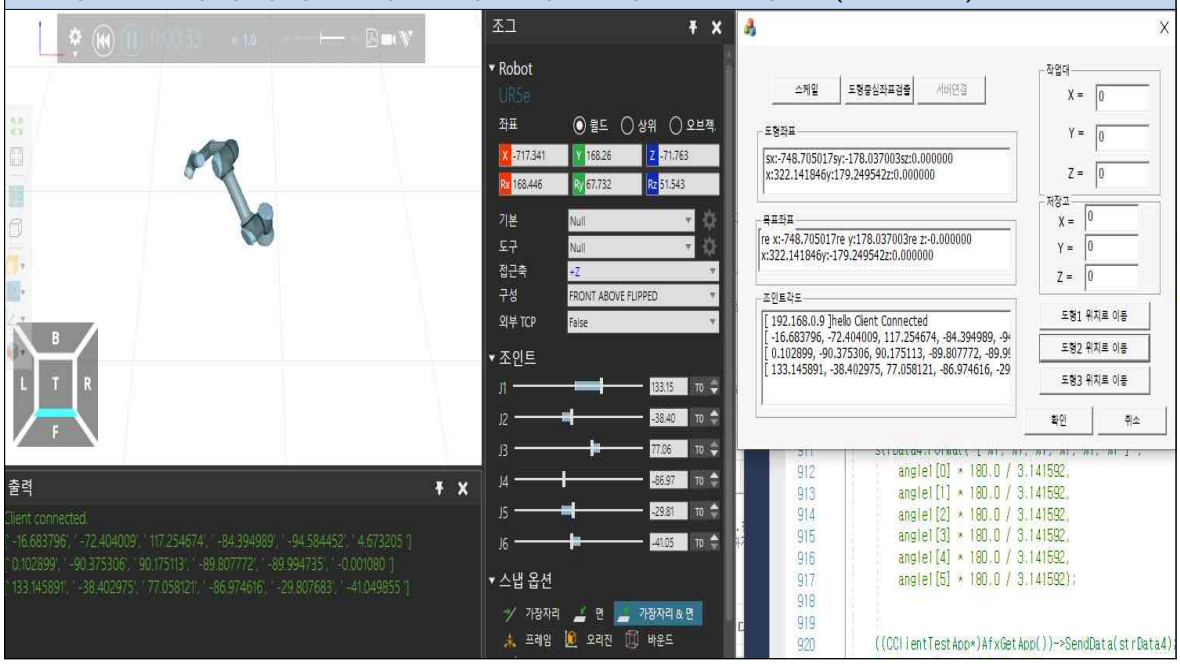


- 작업대 도형2번 좌표 위치로 이용하는 사진 (도형2번)

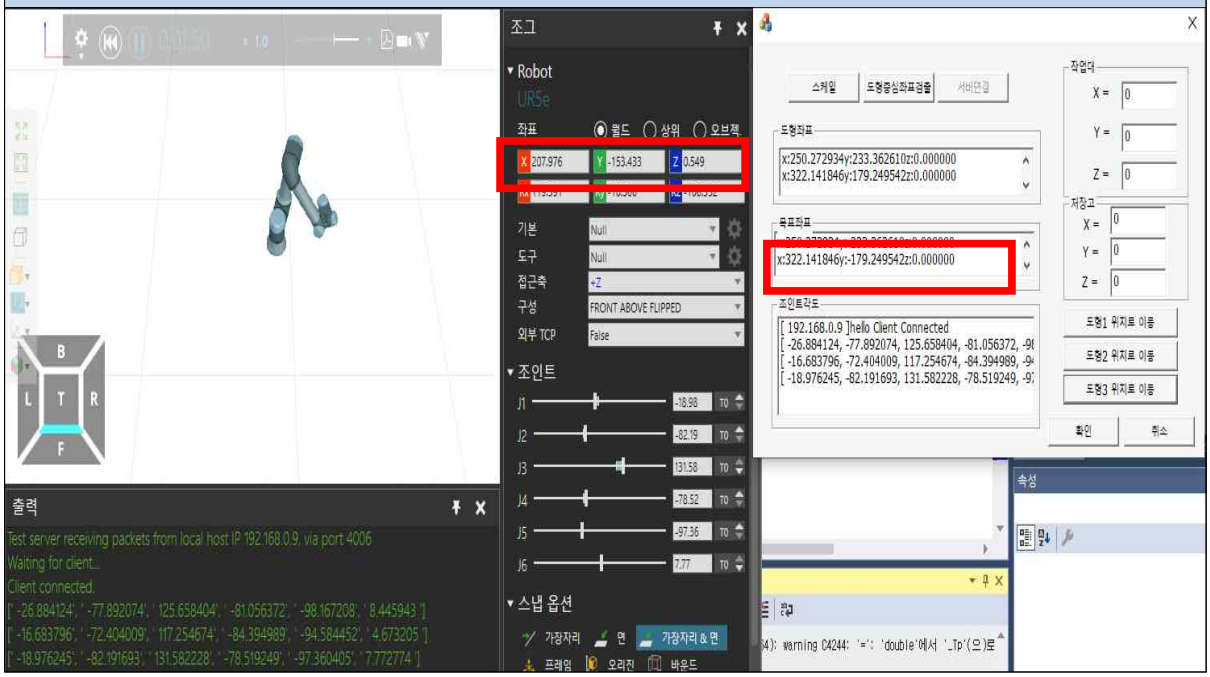


- 저장고 위치 원하는 임의 좌표 2번째 x -748.705 y 178.03700 z 0.0

- 저장고 위치에서 원하는 좌표에 물체 놓는 사진 (도형2번)

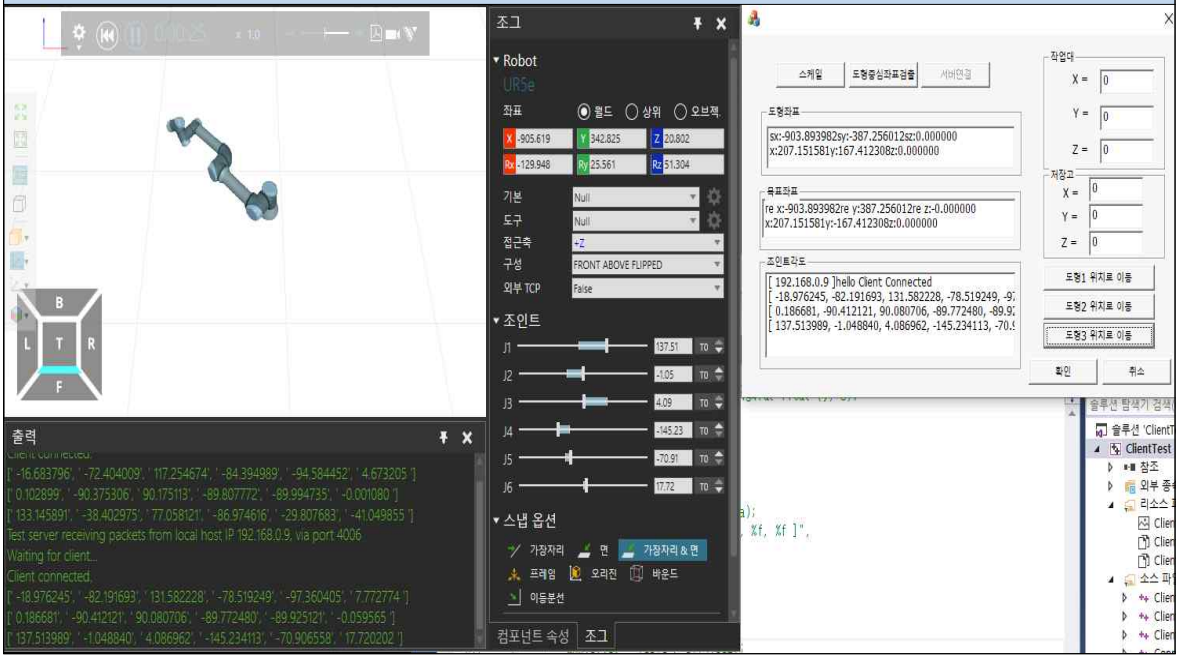


- 작업대 도형3번 좌표 위치로 이용하는 사진 (도형3번)



- 저장고 위치 원하는 임의 좌표 2번째 x -903 y 387 z 0.0

- 저장고 위치에서 원하는 좌표에 물체 놓는 사진 (도형3번)



3. 결론

3.1 설계보완점 및 목표구현 정도

1) 목표 구현 정도

스케일 측정 : 100 %

도형중심점, 중심좌표, 월드좌표계 변환 : 100 %

회전행렬 및 로봇중심 좌표 변환 : 100%

원하는 좌표 위치로 로봇움직임 : 100%

2) 보완 할점 : 정확한 위치로 로봇 움직임 필요(오차 차이가 조금 있음)

3.2 작품 개발과정에서 활용한 공학적 도구

1. MATLAB R2019B
2. Delfoi Premium 4.1

3.3 향후 개선사항

1. 로봇의 정확한 위치를 위한 보간 작업 및 피드백 알고리즘 고안 필요