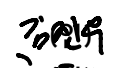



캡스톤디자인I 중간보고서

제 목	국문	유니티 기반 VR 환경에서의 원격 로봇 제어 시스템		
	영문	Remote robot control system in Unity-based VR environment		
진 행 상 황	주요마ilestone	1. 기초 아키텍처 작성(시스템 구성도) 2. 필요한 핵심기능 파악 - 카메라 데이터 송·수신을 위한 데이터 서버 - 컨트롤러 데이터 수집 및 송신 - 원격제어를 위한 데이터 수신 처리 기능. 3. 최소 기능 구동의 요구조건 확인 - 컨트롤러 데이터 수신을 위한 통신 기술 설계 - 카메라 데이터 송·수신을 위한 WebRTC 기술 구축 4. 기능별 필요한 기술 파악 및 필요 리소스 확인 - VR 환경 구축 및 컨트롤러 데이터 수집을 위한 유니티 활용 - 영상 데이터의 실시간 송·수신을 위한 WebRTC 기술 필요 - 컨트롤러 데이터 송·수신에 필요한 통신 기술 필요 5. 설계 및 분석 - 두 Peer 인 유니티와 카메라의 P2P 통신을 위한 Signaling Server 설계 및 구현 - OpenCV를 통해 카메라 데이터 수집 후 WebRTC 기반 송·수신 구조 설계 - 유니티에 연결된 오쿨러스 컨트롤러 데이터 추출 설계 - 추출된 컨트롤러 데이터의 ZMQ 통신 구조 수신 설계 및 구현. 6. E-R 다이어그램 7. 클래스 설계서 및 소스코드 작성		
	진행상황	1. 360카메라와 유니티간 실시간 영상 데이터 통신을 위한 WebRTC 기술 구현 2. 컨트롤러 데이터 송·수신 기능 구현 2. 컨트롤러 데이터를 통한 로봇암 원격제어 구현 3. 유니티 UI를 통한 데이터 생성 방안 설계. 4. WebRTC 기반 영상 데이터 품질 개선 및 저지연 방안 모색. 5. 사용자 친화적 UI 구현 방법 모색.		
산출물	요구사항 정의서(별첨 1), 중간보고서(별첨 2)			
팀 구성원	학년	학 번	이 름	연락처(전화번호/이메일)
	4	20181586	김진우	01085784990/20181586@edu.hanbat.ac.kr
	4	20181588	김현기	01046543431/20181588@edu.hanbat.ac.kr
	4	20181612	김자용	01034124397//20181612@edu.hanbat.ac.kr
컴퓨터공학과의 프로젝트 관리규정에 따라 다음과 같이 요구사항 정의서와 중간보고서를 제출합니다 <div style="text-align: right;"> 2023년 04월 26일 책임자 : 김진우  지도교수 : 최창범  </div>				

[별첨1]

프로젝트명 : 유니티 기반 VR 환경에서의 원격 로봇 제어 시스템 개발

소프트웨어 요구사항 정의서

Version 1.0

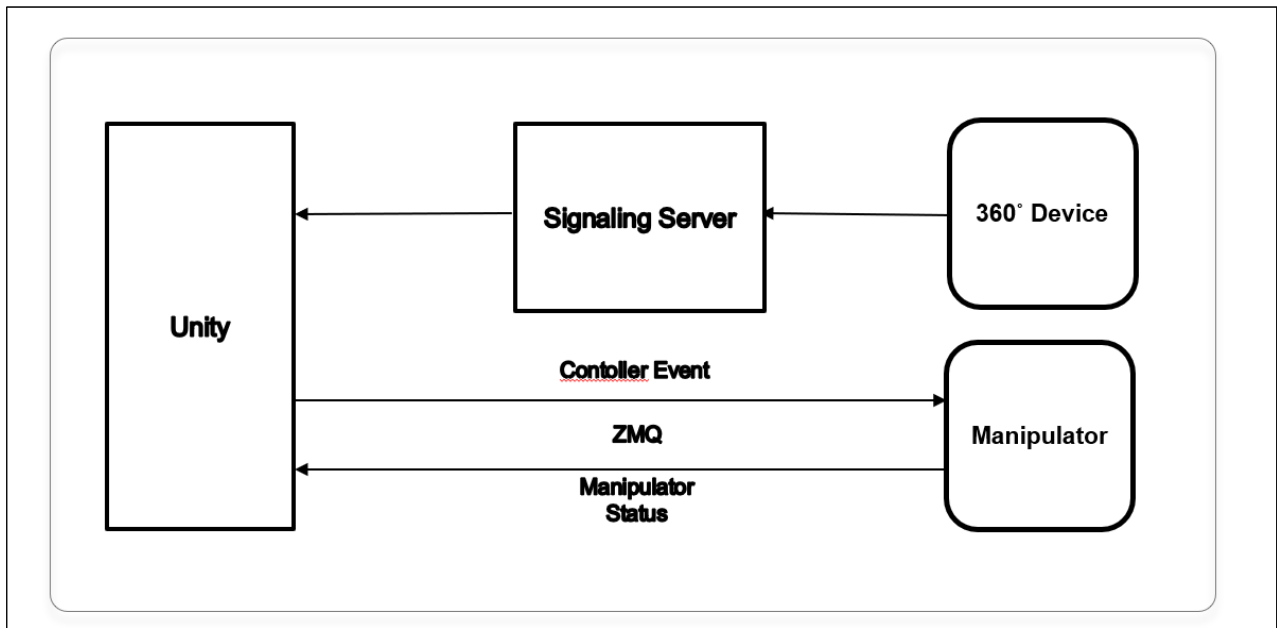
개발 팀원 명(팀리더):김진우
김현기
김자용

대표 연락처:010-8578-4990
e-mail: 20181586@edu.hanbat.ac.kr

목차

1. 개요
2. 시스템 장비 구성요구사항
3. 기능 요구사항
4. 성능 요구사항
5. 인터페이스 요구사항
6. 데이터 요구사항
7. 테스트 요구사항
8. 보안 요구사항
9. 품질 요구사항
10. 제약 사항
11. 프로젝트 관리 요구사항

1. 시스템 개요



2. 시스템 장비 구성요구사항

요구사항 고유번호		ECR-001		
요구사항 명칭		Mirobot Arm		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	<ul style="list-style-type: none"> - 로봇 암 		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : Mirobot Arm - 장비 수량 : 1개 - 장비 기능 : 머니플레이터 - 장비 성능 및 특징 : 로봇 원격 제어 실사용을 위해 로봇암을 활용하여 테스트 - 자원 제약사항 : 외부 카메라 사용 		

요구사항 고유번호		ECR-002		
요구사항 명칭		Oculus Quest2		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	<ul style="list-style-type: none"> - VR 장비 		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : Oculus Quest2 - 장비 수량 : 1개 - 장비 기능 : VR 환경 구축 - 장비 성능 및 특징 : 360도 카메라를 활용하여 VR 환경에서 원격으로 로봇을 제어 		

요구사항 고유번호		ECR-003		
요구사항 명칭		Server		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	- 통신 서버		
	세부 내용	- 장비 품목 : WebRTC, ZMQ - 장비 수량 : 1개 - 장비 기능 : 실시간 스트리밍과 데이터 교환을 위한 웹 서버 - 장비 성능 및 특징 : 원격으로 로봇을 제어하기 위해 실시간으로 카메라 데이터, 스틱 데이터 송수신		

요구사항 고유번호		ECR-004		
요구사항 명칭		Ricoh Theta V		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	- 카메라		
	세부 내용	- 장비 품목 : Ricoh Theta V - 장비 수량 : 1개 - 장비 기능 : 카메라 영상 촬영 - 장비 성능 및 특징 : 360도 영상을 실시간으로 촬영		

3. 기능 요구사항

요구사항 고유번호		SFR-001		
요구사항 명칭		카메라 데이터 수집		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	360도 카메라로 촬영한 데이터 수집		
	세부 내용	- Ricoh Theta V 카메라로 촬영한 데이터를 OpenCV를 사용하여 카메라 데이터 수집 - 사용언어 : Python		

요구사항 고유번호		SFR-002		
요구사항 명칭		영상 전처리 기능		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	수집한 카메라 데이터 전처리		
	세부 내용	- 사용자 맞춤 영상을 제공할 수 있도록 데이터 전처리 - 사용언어 : Python		

요구사항 고유번호		SFR-003		
요구사항 명칭		Signaling Server 구축		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	스트리밍 하기 위한 서버 구축		
	세부 내용	- 각 peer 장치간의 P2P 연결을 관리하기 위해 Signaling Server 구축 - P2P 연결을 통해 실시간성 보장 - 사용언어 : Python		

요구사항 고유번호		SFR-004		
요구사항 명칭		스트리밍 데이터 송·수신		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	카메라 데이터 송·수신		
	세부 내용	- WebRTC 기반 카메라 데이터 송·수신 - 사용언어 : Python		

요구사항 고유번호		SFR-005		
요구사항 명칭		Controller Data 수집		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	Oculus Quest2 Controller Data 수집		
	세부 내용	<ul style="list-style-type: none"> - 로봇을 원격 제어하기 위해 Oculus Quest2 Controller Data를 수집 및 처리 - 사용언어 : C# 		

요구사항 고유번호		SFR-006		
요구사항 명칭		Controller Data 송·수신		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	Controller Data 송·수신		
	세부 내용	<ul style="list-style-type: none"> - 수집한 Controller Data 송신 및 송신한 Controller Data 머니플레이터 서버에서 수신 - 사용언어 : Python, C# 		

요구사항 고유번호		SFR-007		
요구사항 명칭		영상 가시화		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	수신받은 스트리밍 데이터 Unity 환경에서 가시화		
	세부 내용	<ul style="list-style-type: none"> - Signaling Server를 통해 수신받은 스트리밍 데이터를 Unity 환경에서 가시화 - 사용언어 : Python, C# 		

요구사항 고유번호		SFR-008		
요구사항 명칭		원격 제어		
요구사항 분류		기능 요구사항	응락수준	필수
요구사항 상세설명	정의	머니플레이터 원격 제어		
	세부 내용	<ul style="list-style-type: none"> - 수신받은 Controller Data를 활용하여 머니플레이터에서 Unity 환경에서 원격 제어 - 사용언어 : C# 		

4. 성능 요구사항

요구사항 고유번호		PER-001		
요구사항 명칭		스트리밍 실시간성		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	원격 로봇 제어를 위한 스트리밍 실시간성		
	세부 내용	<ul style="list-style-type: none"> - 원격으로 로봇을 제어하기 위해 스트리밍 데이터를 송·수신 - 실시간성을 보장하기 위해 WebRTC 기반에 P2P 통신 - 실시간성을 보장하기 위해 딜레이는 1초 이내로 한다. 		

요구사항 고유번호		PER-002		
요구사항 명칭		스트리밍 데이터 전처리		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	스트림이 데이터 전처리		
	세부 내용	<ul style="list-style-type: none"> - 우선적으로 사용자에게 친화적인 높은 해상도의 스트리밍 데이터 제공 - 네트워크 환경 및 주변 상황에 따라 사용자가 직접 해상도 조절 - 실시간성이 떨어지면 실시간성을 유지하기 위한 해상도 조절 		

5. 인터페이스 요구사항

요구사항 고유번호		SIR-001		
요구사항 명칭		사용자 친화적 인터페이스		
요구사항 분류		사용자 인터페이스	응락수준	필수
요구사항 상세설명	정의	직관적인 인터페이스 구현		
	세부 내용	<ul style="list-style-type: none"> - Unity 환경에서 로봇을 제어하기 위해 사용자 직관적인 인터페이스 구현 - Unity 환경에서 사용자 직관적인 해상도 조절 인터페이스 구현 		

6. 데이터 요구사항

요구사항 고유번호		DAR-001		
요구사항 명칭		데이터 관리		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세설명		<ul style="list-style-type: none"> - 프로토타입을 활용하여 개발자가 정의하기 어려운 사용자의 구체적인 요구사항 및 데이터 수집 - 데이터는 Oculus Quest2 Controller Data 수집 - 네트워크 환경에 따라 다른 스트리밍 속도 데이터 수집 - 사용자의 데이터 수집은 공업설계입문 수업에서 학생들 대상으로 데이터 수집 		

7. 테스트 요구사항

요구사항 고유번호		TER-001		
요구사항 명칭		테스트 방안		
요구사항 분류		테스트	응락수준	필수
요구사항 상세설명		<ul style="list-style-type: none"> - 시스템은 축소화된 버전의 프로토타입을 단기간에 개발하며 사용자 요구사항 데이터를 수집하여 모든 부가 기능을 구현하는 것을 목표로 한다. - Signaling Server와 ZMQ 통신 방식을 활용하여 머니플레이터와 Unity환경, 카메라 데이터와 Unity 환경에서 연동을 기초로 하며 추가적으로 실시간성을 보장하는 것을 목표로 한다. - 최종 테스트는 부가적인 기능을 완료하고 공업설계입문 수업에서 학생들을 대상으로 테스트 하여 학생들의 의견을 기준으로 완성 여부를 정한다. 		

8. 보안 요구사항

요구사항 고유번호		SER-001		
요구사항 명칭		서버 보안		
요구사항 분류		보안	응락수준	필수
요구사항 상세설명		<ul style="list-style-type: none"> - 클라이언트 간에 Private IP 주소 유출을 막기 위해 STUN Server를 활용하여 Public IP 주소를 활용하여 통신한다. - IP Address, SDP Pattern 등 개인정보 유출을 막기 위해 해당 정보를 소스코드에 직접 하드코딩하지 않는다 		

9. 품질 요구사항

요구사항 고유번호		QUR-001		
요구사항 명칭		WebRTC		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	P2P 방식을 활용한 스트리밍		
	세부 내용	<ul style="list-style-type: none"> - 최적의 원격 로봇 제어를 위해 P2P 방식을 활용하여 WebRTC 통신 기술 사용. - 실시간성 보장 		

요구사항 고유번호		QUR-001		
요구사항 명칭		WebRTC		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	P2P 방식을 활용한 스트리밍		
	세부 내용	<ul style="list-style-type: none"> - 최적의 원격 로봇 제어를 위해 P2P 방식을 활용하여 WebRTC 통신 기술 사용. - 실시간성 보장 		

10. 제약 사항

요구사항 고유번호		COR-001		
요구사항 명칭		유니티 기반 VR 환경에서의 원격 로봇 제어 시스템		
요구사항 분류		제약 사항	응락수준	필수
요구사항 상세설명		<ul style="list-style-type: none"> - 유니티 기반 VR 환경에서의 원격 로봇 제어를 위해서는 네트워크 환경이 구축되어 있어야 하며 네트워크 환경이 구축되어 있지 않으면 원격 제어의 조작의 한계가 있음. - 360도 카메라 데이터는 용량이 커 최적의 실시간성을 보장하기에 한계가 있음. 		

11. 프로젝트 관리 요구사항

요구사항 고유번호		PMR-001		
요구사항 명칭		프로젝트 관리		
요구사항 분류		프로젝트 관리	응락수준	필수
요구사항 상세설명		<ul style="list-style-type: none"> - Unity 환경과 머니플레이터간의 통신을 위해 서버의 최적화를 최우선 목표로 한다. - 서버는 항상 동작하며 실시간성 향상을 목표로 한다. - 팀원은 각 서버, Unity , 머니플레이터 중 맡은 역할을 수행한다. - 모든 팀원은 기존의 프로토타입 완성도를 높이고자 매주 부득이한 경우를 제외하고 대면으로 회의를 진행한다. - 프로젝트의 완성도를 높이고자 해당 전문 교수님께 자문을 구한다. 		

[별첨2]

중간보고서

1. 요구사항 정의서에 명시된 기능에 대하여 현재까지 분석, 설계, 구현(소스코드 작성) 및 테스트한 내용을 기술하시오.

- 시스템(하드웨어, 시스템)구성도, 또는 소프트웨어 아키텍처

- Unity : 사용자에게 360도 카메라를 통한 시야 제공 및 연동된 오쿨러스 컨트롤러를 통한 제어 기능 제공.
- WebRTC : 카메라 데이터의 데이터를 실시간 처리 및 출력으로 유니티에게 데이터를 제공.
- ZMQ : 유니티를 통해 제공 받은 오쿨러스 컨트롤러 데이터를 머니플레이터에 실시간 전송 및 제어 기능 제공.

- 기능별 상세 요구사항(또는 유스케이스)

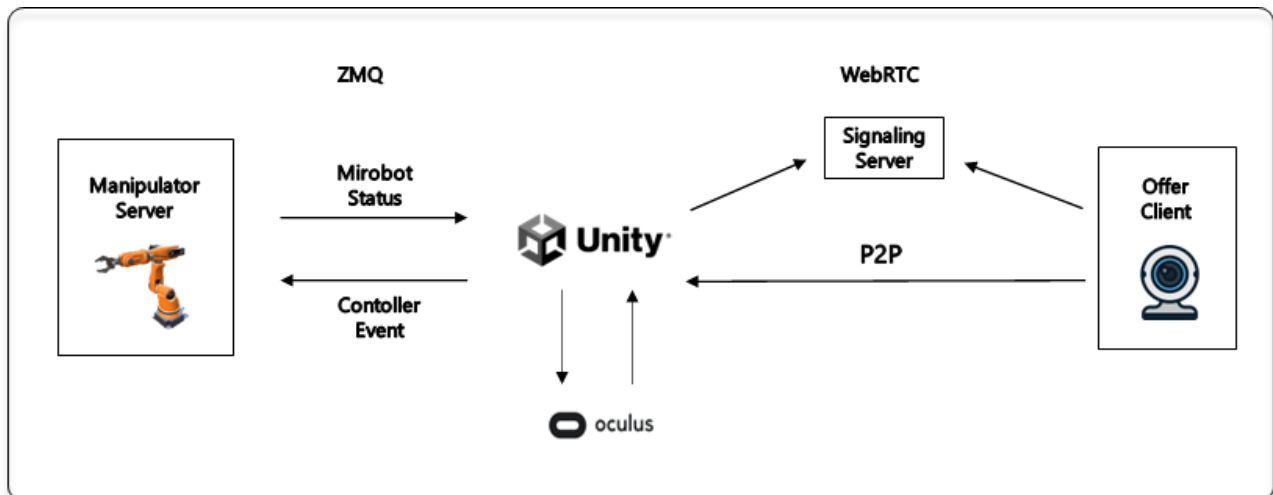
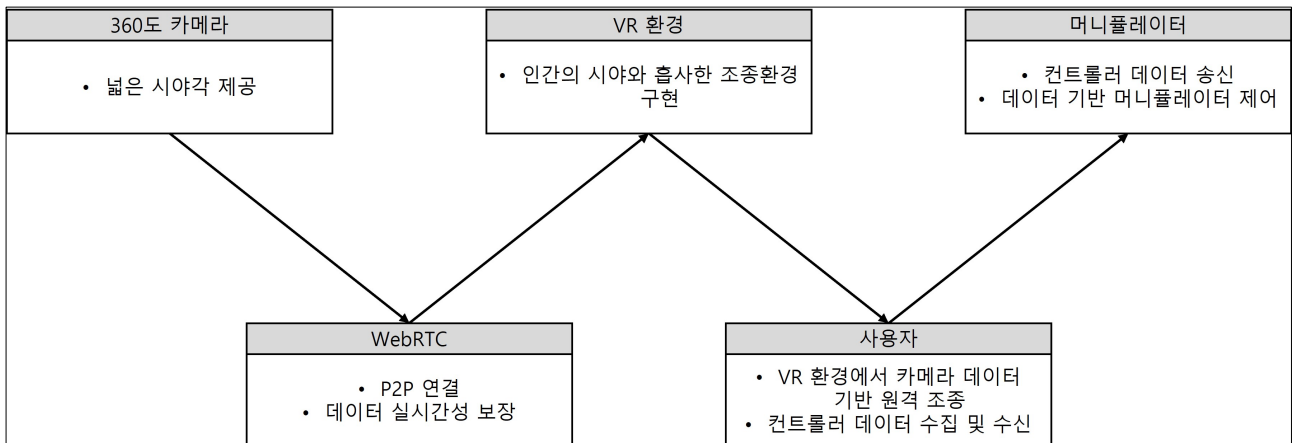


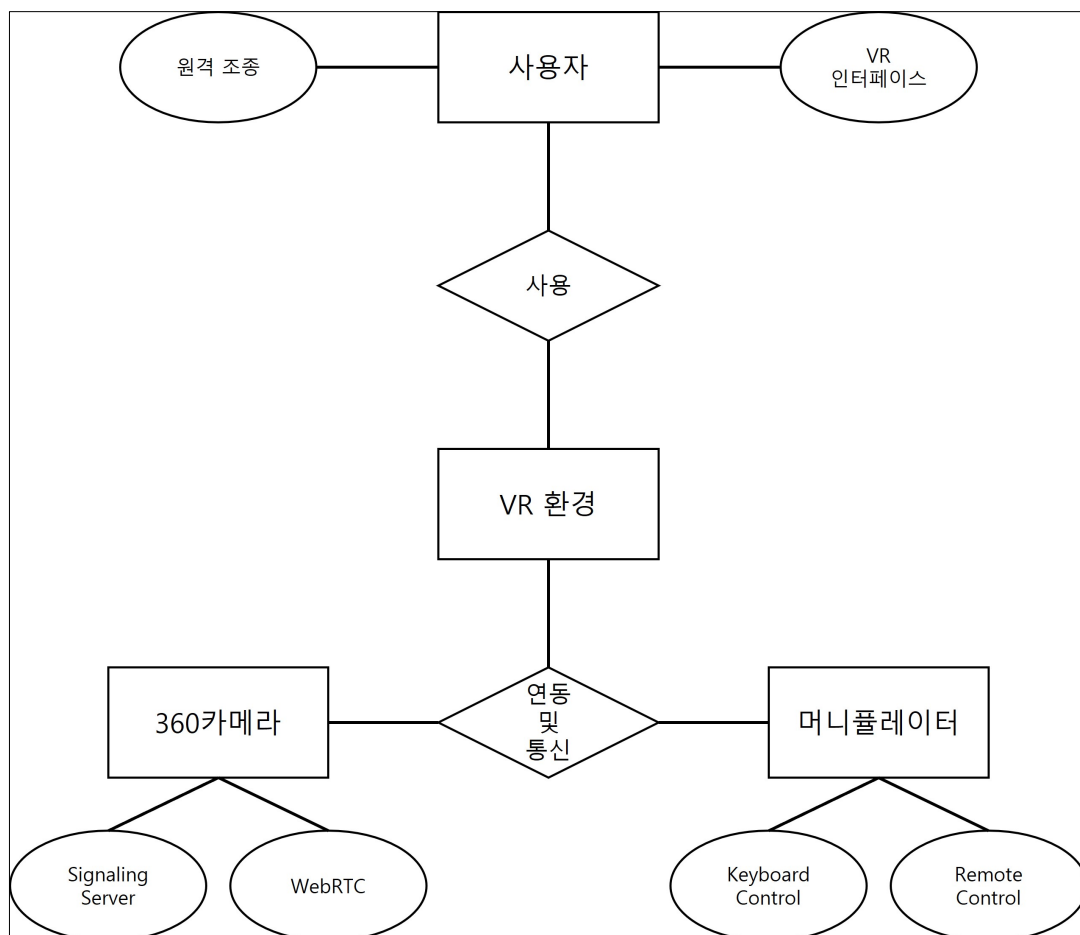
그림 2 시스템
구조도

기술	설명
WebRTC	<ul style="list-style-type: none">- WebRTC 기반 연결- WebRTC 기반 실시간 360도 카메라 데이터 교환
사용자	<ul style="list-style-type: none">- Unity Engine 기반 WebRTC 데이터 시각화 제공- Oculus Controller 데이터 전송
ZMQ	<ul style="list-style-type: none">- Unity 데이터 처리를 통한 로봇 암 간 통신 환경 구축- 로봇 암 조종

- 설계 모델(클래스 다이어그램, 클래스 및 모듈 명세서)



- E-R 다이어그램/DB 설계 모델(테이블 구조)



- 테스트 계획서, 테스트 케이스 기술서 등등.

- 프로젝트 최종 기능 구현을 위해 팀원 간 역할분담(유니티, Webrtc, ZMQ 로봇제어)을 하여, 360도 카메라 데이터를 VR 환경(오큘러스 퀘스트)에서 실시간 스트리밍하며 오큘러스 퀘스트의 스틱을 활용하여 VR 환경에서 원격으로 현실에 있는 로봇을 제어할 수 있도록 가능하게 해주는 시스템을 만드는 것이 목표이다.

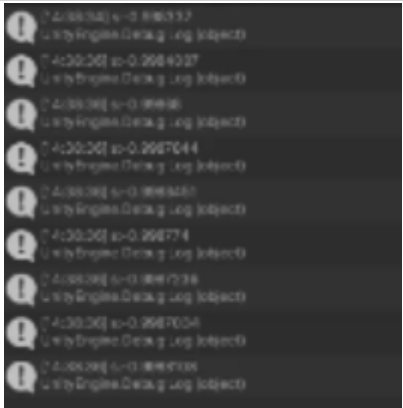
- 현재까지 구현한 소스코드 및 캡처

촬영



● 해당 화면은 WebRTC 및 ZMQ 통신 기술과 유닛을 통해 실시간 조작 하는 이미지이다.

컨트롤러 데이터 출력 및 전송



● 해당 사진은 위의 찍은 사진의 컨트롤러 데이터의 정보와 전송됨을 나타내는 사진이다.

Unity Controller 소스코드

```
{
    clientnew PublisherSocket();
    client.Bind("tcp://*:11012");
}

async void manual_stick_data()
{
    if(leftController.isValid)
    {
        Vector2 leftStick;
        if(leftController.TryGetFeatureValue(CommonUsages.primary2DAxis, out leftStick))
        {
            if(leftStick[1] > 0.998)
            {
                Debug.Log("w: "+leftStick[1]);
                client.SendFrame("w");
            }

            if(leftStick[1] < -0.998)
            {
                Debug.Log("s: "+leftStick[1]);
                client.SendFrame("s");
            }

            if(leftStick[0] > 0.9)
            {
                Debug.Log("d: "+leftStick[0]);
                client.SendFrame("d");
            }

            if(leftStick[0] < -0.9)
            {
                Debug.Log("a: "+leftStick[0]);
                client.SendFrame("a");
            }
        }

        if(rightController.isValid)
        {
            // Get right stick position
            Vector2 rightStick;
            if(rightController.TryGetFeatureValue(CommonUsages.primary2DAxis, out rightStick))
            {
                if(rightStick[1] > 0.998)
                {
                    client.SendFrame("up");
                }

                if(rightStick[1] < -0.998)
                {
                    client.SendFrame("down");
                }

                if(rightStick[0] > 0.998)
                {
                    client.SendFrame("left");
                }

                if(rightStick[0] < -0.998)
                {
                    client.SendFrame("right");
                }
            }
        }

        await Task.CompletedTask;
    }
}
```

- 해당 코드는 Unity에서 ZMQ-Pub을 사용하여 수집된 컨트롤러의 데이터를 전송하는 코드이다.

WebRTC - Server 소스코드

```
# 전송하는 카메라 데이터 저장
@app.route('/offer', methods=['POST'])
def offer():
    """
    Offer에서 제안하는 SDP 메시지 내용으로 선언 및 연결
    """
    if request.form["type"] == "offer":
        data["offer"] = {"id":request.form['id'], "type":request.form['type'], "sdp":request.form['sdp']}
        return Response(status=200)

    else:
        return Response(status=400)

@app.route('/answer', methods=['POST'])
def answer():
    """
    Answer에서 제안하는 SDP 메시지 내용으로 선언 및 연결
    """
    req=request.form.to_dict()
    req_keylist(req.keys())
    req_key=req_key[0]
    req_result=json.loads(req_key)

    if req_result['type'] == 'Answer':
        data["answer"] = {"id":req_result['id'], "type":req_result['type'], "sdp":req_result['sdp']}
        return Response(status=200)
    else:
        return Response(status=400)

@app.route('/get_offer')
def get_offer():
    """
    Offer SDP 데이터 처리
    """
    # 카메라 데이터 확인
    if "offer" in data:
        # 카메라 데이터 json객체로 저장
        jjson.dumps(data["offer"])
        # json객체로 저장한 데이터 삭제
        del data["offer"]
        # 상태 코드 및 json파일 반환
        return Response(j, status=200, mimetype='application/json')
    else:
        return Response(status=503)

@app.route('/get_answer')
def get_answer():
    """
    Answer SDP 데이터 처리
    """
    if "answer" in data:
        jjson.dumps(data["answer"])
        del data["answer"]
        return Response(j, status=200, mimetype='application/json')
    else:
        return Response(status=503)
```

- 해당 코드는 유니티에 전송된 데이터를 송/수신 하는 서버 코드이다.

Unity - WebRTC 소스코드

```
private void Update()
{

    peerConnection.OnDataChannel =(channel_1)=>
    {
        Debug.Log("Data channel created by remote party!");
        // Assign the channel object to the dataChannel variable
        channel =channel_1;
        // Set up the data channel event handlers
        channel.OnOpen +=()=>>
        {
            Debug.Log("Data channel open!");
        };
        channel.OnClose +=()=>>
        {
            Debug.Log("Data channel closed!");
        };
        channel.OnMessage +=(byte[]bytes)=>
        {

            var message =System.Text.Encoding.UTF8.GetString(bytes);

            Texture2D texture =new Texture2D(2,2);

            texture.LoadImage(Convert.FromBase64String(System.Text.Encoding.UTF8.GetString(bytes)));
            cuberenderer.material.mainTexture =texture;
            cuberenderer.enabled =true;

        };
    };
}
```

- 해당 코드는 Unity에서 Webrtc를 통해 수신된 데이터를 처리하고 송신하는 코드이다.

머니플레이터 원격 제어 소스코드

```
class Remotemanager(MirobotManager):
    def __init__(self, arm) -> None:
        super().__init__(arm)
        print("ZMQ recieve Initialized")

        # Asyncio ZMQ Object
        self.contextzmq.asyncio.Context()
        asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())

        # Default ZMQ Object
        # self.context = zmq.Context()

        self.socket = self.context.socket(zmq.SUB)
        self.socket.setsockopt_string(zmq.SUBSCRIBE, "")
        self.socket.connect(f"tcp://{HOST}:{PORT}")
        print(f"zmq subscribed at {HOST}:{PORT}....")

        ### ZMQ Response
        # self.context = zmq.Context()
        # self.socket = self.context.socket(zmq.REP)
        # self.socket.bind(f"tcp://{HOST}:{PORT}")
        # print(f"Server listening on port {PORT}...")
    async def _remoteRecv(self):
        while True:
            self.message = await self.socket.recv_string()
            print(f"Received message: {self.message}")

            await self.AxisControl(self.message)
```

- 해당 코드는 수신된 컨트롤러 데이터를 처리하여 머니플레이터를 제어하는 코드이다.

2. 프로젝트 수행을 위해 적용된 추진전략, 수행 방법의 결과를 작성하고, 만일 적용과정에서 문제점이 도출되었다면 그 문제를 분석하고 해결방안을 기술하시오.

- 문제해결을 위해 적용한 방법(또는 기법) 결과, (문제점, 해결방안)

[문제점]
- 게임엔진 특유의 프레임에 따른 데이터 송/수신 반복 및 전송 시간 문제
[해결방안]
- 프레임에 따른 데이터 송수신 반복이 존재하여 해당 문제를 해결하고자 비동기 및 코루틴을 사용해 데이터 송수신 문제를 해결. 코루틴 및 비동기를 동시에 사용하여 데이터의 송수신 속도를 제어할 수 있었음.
- 전송시간에 따른 문제는 마찬가지로 코루틴 과 비동기를 사용함으로써 1~3초사이에는 데이터를 송신하게만듬.

- 팀원의 책임 및 역할 수행에 대한 결과, (문제점, 해결방안)

[문제점]
- 게임 엔진과 서버간의 통신 시 송/수신 속도에 대한 문제점
[해결방안]
- 지연 속도를 줄이기 위해 ZMQ PUB-SUB 패턴 사용
- 서버 및 게임 엔진 속도 향상을 위해 비동기 방식 사용

[문제점]
- 정밀한 로봇암 원격 제어의 제약사항이 생기는 문제점
[해결방안]
- 컨트롤러 데이터 수집
- 로봇암의 정확한 동작 수치 파악 및 추출

[문제점]
- 프레임마다 추출하는 360도 카메라 데이터 처리로 인해 느려지는 속도에 대한 문제점
[해결방안]
- 360도 카메라 데이터로 인해 속도가 느려져 원시 데이터 전송
- WebRTC 방식을 통해 클라이언트 간의 직접 데이터를 송/수신하여 실시간성 보장

- 프로젝트 일정계획에 맞추지 못한 경우의 문제점, 해결 방안

[문제점]
- 현실감 있는 UI를 제공하지 못한다면 사용자에게 불편함을 제공하고 몰입감있는 게임엔진을 사용한다는 의미가 존재하지 않을 수 있는 문제점
[해결방안]
- 현재 UI문제만 해결한다면 전체적인 프로젝트 일정계획은 어느 정도 해결되어감. UI 설계 및 구축은 공학설계 입문 수업을 통해 사용자에게 현실감있는 UI는 어떻게 구성되어야 하는지 분석해볼 예정.

- 요구사항 변경관리의 결과, (문제점, 해결 방안) 등등.

[문제점]
- 서로 역할분담을 하는데 같은 조원에게 어떤 부분이 필요한지 알기 어려움
[해결방안]
- 서로 맡은 부분 외에 같은 조원의 역할을 이해하고 자신에게 필요한 것과 상대방에게 필요한 것을 요구할 수 있을 수준의 공부가 필요함

캡스톤 디자인 I 중간보고서 채점표

평가도구	평 가 항 목	평 가 점 수				
		1	2	3	4	5
중간 보고서 및 실행 결과	1. 요구사항 정의서(기능, 성능, 인터페이스 등)가 구체적으로 작성되었는가?		✓			
	2. 요구분석, 설계 산출물(모델, 프로토타입 등)의 내용이 충실한가?		✓			
	3. 설계 및 구현 문제를 위해 적용한 이론, 문제해결 방법이 제시되었으며 그 적용이 적합한가?				✓	
	4. 구현된 소프트웨어(또는 이와 동등한 하드웨어 시스템)가 버그없이 실행되었는가?				✓	
	5. 구현된 소프트웨어(또는 이와 동등한 하드웨어 시스템)의 성능 요구사항은 충족되었는가?				✓	
도구활용	6. 설계 및 구현을 위해 도구가 적절히 활용되었는가?				✓	
	7. 도구의 활용수준(능숙도)은 프로젝트 수행에 적합한가?		✓			
팀원의 업무 및 역할	8. 팀원의 업무분담에 따른 역할 및 협력이 충실히 이루어졌는가? (평가자에 의한 질의)				✓	
	9. 프로젝트 중간 진척상황에 대해 팀원이 충분히 인지하고 있는가?(평가자에 의한 질의)				✓	
합계		30				
<p>*검토 의견(최종완료 때까지 보완해야할 점에 대해 작성 요망)</p> <p>- 구체적인 미흡으로 진행 상황에 개한 경우 적음</p> <p>- 팀원 협조도 높음에서 요구사항은 명확하고 체계적</p> <p>내용으로 소개 및 관련 내용 수정 필요</p>						
심사위원(소속):		김민준		(이름) 김민준		