

캡스톤디자인 II 중간보고서(표지)

프로젝트명 : 학교 내 자율주행 로봇 배달 서비스
캡스톤 디자인II, 중간보고서

Version 1.0

개발 팀원 명(팀리더): 김수미
이동현
송찬호

대표 연락처: 010-8235-9619
e-mail: ymg12347@gmail.com

캡스톤 디자인 II 중간보고서 내용

1. 요구사항 정의서에 명시된 기능에 대하여 현재까지 진척된 결과 및 그 내용을 기술하시오.

- 소스코드 일부(코드 작성지침의 준수 확인용) 및 ver1.0 실행파일(환경설치 문서 포함)

- (변경된) 요구정의서 - [별첨 1]

- 시스템(하드웨어, 시스템)구성도, 또는 소프트웨어 아키텍처

- 기능별 상세 요구사항(또는 유스케이스)

- 설계 모델(클래스 다이어그램, 클래스 및 모듈 명세서)

- UI 분석/설계 모델/프로토타입

- E-R 다이어그램/DB 설계 모델(테이블 구조)

- 테스트 계획서, 테스트 케이스 기술서 등등

1-1. ROSBridge를 활용한 서버와 ROS 간의 통신

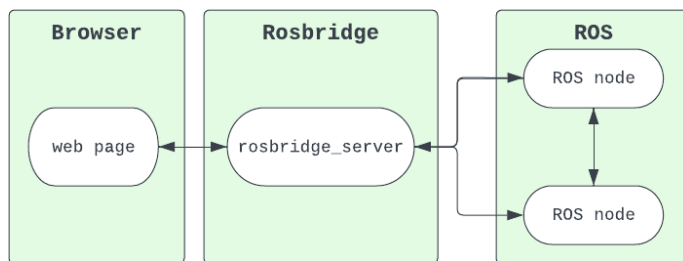


그림1. RosBridge 구조

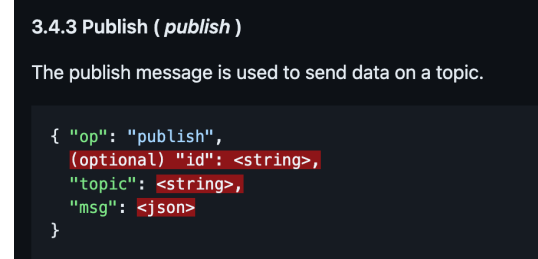


그림2. RosBridge publish 명령 JSON 형식

ROSBridge는 로봇 운영 체제(ROS)와 다양한 외부 시스템 간의 원활한 통신을 가능하게 하는 소프트웨어 패키지이다. 이 패키지는 비-ROS 시스템이 ROS와 상호작용할 수 있도록 지원하며, 특히 웹 애플리케이션과 ROS 간의 양방향 통신을 구현하는 데 유용하다. ROSBridge는 웹 소켓을 통해 데이터의 송수신을 처리하며, 이를 통해 실시간으로 정보를 주고받을 수 있다. 데이터 송수신 시에는 명령에 맞는 JSON 형식을 반드시 준수해야 한다.

우리 프로젝트에서는 서버와 ROS 간의 효과적인 통신을 위해 ROSBridge를 활용하였다. 이를 통해 서버와 ROS 간의 데이터 교환을 원활하게 처리할 수 있었으며, 시스템의 상호작용을 개선할 수 있었다.

1-2. 서버를 통한 로봇 네비게이션 명령 전송

```
private static WebSocketMessage<String> createRobotNavigationMessage(Location location) {
    JSONObject position = new JSONObject();
    position.put("x", location.getPositionX());
    position.put("y", location.getPositionY());
    position.put("z", location.getPositionZ());

    JSONObject orientation = new JSONObject();
    orientation.put("x", location.getOrientationX());
    orientation.put("y", location.getOrientationY());
    orientation.put("z", location.getOrientationZ());
    orientation.put("w", location.getOrientationW());

    JSONObject pose = new JSONObject();
    pose.put("position", position);
    pose.put("orientation", orientation);

    JSONObject header = new JSONObject();
    header.put("frame_id", "map");

    JSONObject msg = new JSONObject();
    msg.put("op", "publish");
    msg.put("topic", "/move_base_simple/goal");
    msg.put("msg", new JSONObject().put("header", header).put("pose", pose));
    return new TextMessage(msg.toString());
}
```

그림3. 네비게이션 명령 웹소켓 JSON 메세지 구현 코드

서버는 로봇에게 네비게이션 명령을 전송하기 위해 move_base_simple/goal토픽을 이용하였다. 이 토픽은 geometry_msgs/PoseStamped메시지 형식을 사용하며, 이를 통해 로봇에게 목표 지점을 지정하고 해당 목표 지점에서의 네비게이션 명령을 전송하여 자동 네비게이션을 수행할 수 있도록 한다.

서버는 RosBridge와 WebSocket을 통해 로봇과 연결되어 있으며, geometry_msgs/PoseStamped메시지 형식에 맞게 로봇의 목표 지점과 헤더 정보를 설정하여 통신을 수행한다. 이후, 서버는 move_base_simple/goal토픽에 publish 명령 메시지를 전달하여 로봇에게 네비게이션 지시를 내린다.

1-3. 네비게이션 목표 상태 추적

actionlib_msgs/GoalStatus Message

File: actionlib_msgs/GoalStatus.msg

Raw Message Definition

```
GoalID goal_id
uint8 status
uint8 PENDING = 0 # The goal has yet to be processed by the action server
uint8 ACTIVE = 1 # The goal is currently being processed by the action server
uint8 PREEMPTED = 2 # The goal received a cancel request after it started executing
# and has since completed its execution (Terminal State)
uint8 SUCCEEDED = 3 # The goal was achieved successfully by the action server (Terminal State)
uint8 ABORTED = 4 # The goal was aborted during execution by the action server due
# to some failure (Terminal State)
uint8 REJECTED = 5 # The goal was rejected by the action server without being processed,
# because the goal was unattainable or invalid (Terminal State)
uint8 PREEMPTING = 6 # The goal received a cancel request after it started executing
# and has not yet completed execution
uint8 RECALLING = 7 # The goal received a cancel request before it started executing,
# but the action server has not yet confirmed that the goal is canceled
uint8 RECALLED = 8 # The goal received a cancel request before it started executing
# and was successfully cancelled (Terminal State)
uint8 LOST = 9 # An action client can determine that a goal is LOST. This should not be
# sent over the wire by an action server

#Allow for the user to associate a string with GoalStatus for debugging
string text
```

그림4. actionlib_msgs/GoalStatusArray메시지 형식

네비게이션 목표의 상태 추적을 위해 /move_base/status토픽을 구독하였다. 이 토픽은 actionlib_msgs/GoalStatusArray메시지 형식을 사용하며, 로봇의 네비게이션 목표 상태를 실시간으로 모니터링할 수 있도록 한다. 상태 코드 1은 새로운 네비게이션 목표가 설정되었음을 나타내며, 상태 코드 3은 목표에 도착했음을 의미한다. 로봇은 동시에 하나의 네비게이션 목표만 수행할 수 있으므로,

상태 코드 1이 나타날 경우, 현재 진행 중인 네비게이션으로 간주된다.

1-4. 네비게이션 목표 식별 개선

```
if (!statusList.isEmpty()) {
    for (int i = 0; i < statusList.size(); i++) {
        JSONObject statusObject = (JSONObject)statusList.get(i);
        Long status = (Long)statusObject.get("status");
        Log.info(status.toString());

        // status가 1이면, 새로 생긴 네비게이션이므로, 해당 goalId를 파싱
        if (status == 1 && currentGoalId == null) {
            JSONObject goalIdObject = (JSONObject)statusObject.get("goal_id");
            currentGoalId = goalIdObject.get("id").toString();
            Log.info("Goal ID with status 1 : " + currentGoalId);
        }

        JSONObject goalIdObject = (JSONObject)statusObject.get("goal_id");
        String goalId = goalIdObject.get("id").toString();
        Log.info("Parsing Goal Id : " + goalId);
        Log.info("Current Goal Id : " + currentGoalId);

        // 해당 goalId인 네비게이션이고, 도착했다면,
        if (status == 3 && goalId.equals(currentGoalId)) {
```

그림4. 네비게이션 상태 추적 및 목표 식별 개선 구현 코드

/move_base/status토픽만을 구독하여 네비게이션 목표의 상태를 확인할 때, 이전 네비게이션의 기록이 남아 있어 현재 진행 중인 네비게이션 작업을 파악하는 데 어려움이 발생할 수 있다. 이러한 이슈를 해결하기 위해 /move_base/goal토픽을 추가로 구독하여 move_base_msgs/MoveBaseActionGoal메시지에서 goal_id를 추출하는 방식으로 현재 네비게이션 목표를 식별할 수 있도록 구현하였다.

이 방법을 통해 로봇의 네비게이션 목표를 보다 명확하게 식별하고 관리할 수 있으며, goal_id를 활용하여 현재 진행 중인 네비게이션 작업을 정확히 파악할 수 있게 되었다. 이를 통해 네비게이션 목표의 상태 추적과 관리의 정확성을 높일 수 있었다.

1-5. 로봇의 실시간 위치 표시 구현

Odometry Localization



AMCL Map Localization

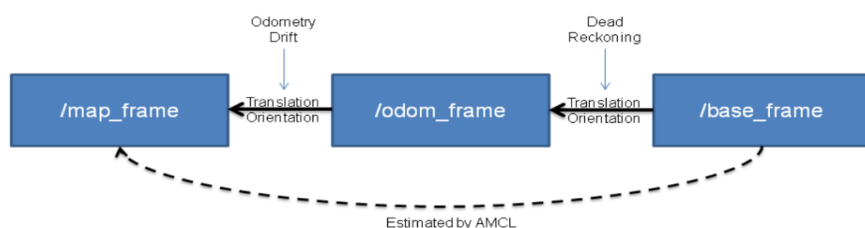


그림5. odom과 amcl에 관한 공식문서

로봇의 실시간 위치를 정확하게 표시하기 위해, 로봇의 위치 정보를 얻을 수 있는 두 가지 주요 토픽, /odom과 /amcl_pose를 비교하였다.

/amcl_pose토픽은 로봇의 절대 위치를 제공하며, Adaptive Monte Carlo Localization (AMCL) 알고리즘을 통해 로봇의 기본 프레임을 전역 프레임과 비교하여 로봇의 절대적인 위치를 추정한다. AMCL은 로봇의 위치를 전역 좌표계에 대해 직접적으로 제공하며, 이 정보는 로봇의 정확한 절대 위치를 파악하는데 유용하다.

반면, /odom토픽은 로봇의 상대 위치를 제공하며, 로봇의 odometry 프레임과 전역 프레임 간의 변환을 게시한다. 이 변환은 로봇의 현재 위치와 이전 위치 간의 상대적인 변화를 나타내며, Dead Reckoning 방식으로 인해 누적 오차(드리프트)가 발생할 수 있다.

이러한 이유로 /amcl_pose토픽이 로봇의 현재 위치를 전역 좌표계에 대해 직접적으로 제공하므로, 맵 상의 절대 위치를 얻기 위해 /amcl_pose토픽을 사용하는 것이 적합하다고 결정하였다.

1-6. 로봇 위치 데이터의 실시간 프론트엔드 전송

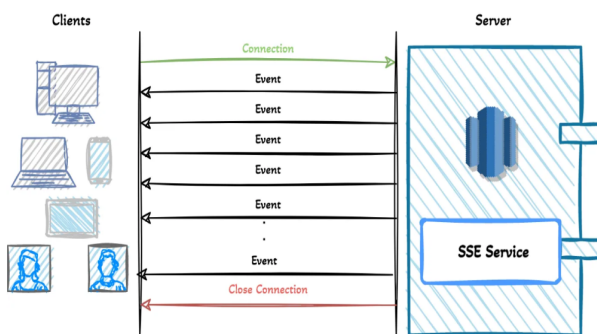


그림6. SSE(Server-Sent Events) 구조

```
public void updatePosition(JSONObject position) { 1 usage  ▲ suri-pu-bi
    this.currentPosition = position;
    sendPosition(position);
}

public void sendPosition(JSONObject position) { 1 usage  ▲ suri-pu-bi
    for (SseEmitter emitter : emitters) {
        CompletableFuture.runAsync(() -> {
            try {
                emitter.send(SseEmitter.event()
                    .name( eventName: "positionUpdate")
                    .data(position.toString()));
            } catch (IOException e) {
                handleEmitterError(emitter);
            }
        });
    }
}
```

그림7. SSE(Server-Sent Events) 구현 코드

/amcl_pose토픽을 구독하여 로봇의 위치를 실시간으로 얻을 수 있다는 점을 확인하였다. 그러나 프론트엔드에서 맵 상에 로봇의 위치를 실시간으로 표시하기 위해서는, 서버에서 수신한 위치 데이터를 프론트엔드로 실시간으로 전달할 필요가 있었다.

로봇의 위치 데이터를 실시간으로 프론트엔드에 전달하기 위해, 서버와 클라이언트 간의 데이터 전송 방식을 검토하였다. Polling 방식은 주기적인 요청으로 인해 불필요한 네트워크 트래픽과 서버 부하를 초래할 수 있다는 점을 확인하였다. 양방향 통신을 지원하는 WebSocket은 실시간 데이터 전송에 적합하지만, 우리 프로젝트에서는 단방향 데이터 전송만이 필요하므로 Server-Sent Events (SSE)를 선택하였다. SSE는 서버에서 클라이언트로의 단방향 데이터 전송을 효율적으로 처리할 수 있으며, 지속적인 업데이트를 실시간으로 제공하는 데 효과적이다.

따라서, 클라이언트와 서버 간의 데이터 전송 효율성을 고려하여 SSE를 설정하였다.

1-7. 로봇 위치 표시를 위한 맵 데이터 처리 및 전송

초기에는 스프링 서버에서 /map토픽을 구독하여 로봇의 맵 데이터를 가져오려고 시도하였다. 그러나 맵 메타데이터가 너무 커서 오류가 발생하였고, 이로 인해 직접 구독 방식이 비효율적이라는 문제에 직면하였다. 이 문제를 해결하기 위해 ROS에서 커스텀 토픽을 생성하고, /map토픽을 구독하여 nav_msgs/OccupancyGrid메시지에서 필요한 정보를 추출하는 방법을 채택하였다.

```
def send_data_to_rosbridge(self, occupancy_grid):

    json_str = json.dumps(self.extract_info(occupancy_grid))
    rospy.loginfo(json_str)
    compressed_data = zlib.compress(json_str.encode())
    encoded_data = base64.b64encode(compressed_data).decode('utf-8')

    data = {
        "op": "publish",
        "topic": "/send/map_data",
        "msg": {
            "data": encoded_data
        }
    }

    try:
        # send Json String data to rosbridge
        self.ws.send(json.dumps(data))
        rospy.loginfo(f"publish topic to rosbridge : {data}")

    except websocket.WebSocketException as e:
        rospy.logerr(f"Failed to send data: {e}")

# message type : OccupancyGrid
def extract_info(self, occupancy_grid):
    return {
        "width": occupancy_grid.info.width,
        "height": occupancy_grid.info.height,
        "resolution": occupancy_grid.info.resolution,
        "origin": {
            "position": {
                "x": occupancy_grid.info.origin.position.x,
                "y": occupancy_grid.info.origin.position.y
            },
        },
        "data": list(occupancy_grid.data)
    }
```

그림8. 매테이터 전송 ROS 구현 코드

nav_msgs/MapMetaData Message

File: `nav_msgs/MapMetaData.msg`

Raw Message Definition

```
# This hold basic information about the characterists of the OccupancyGrid

# The time at which the map was loaded
time map_load_time
# The map resolution [m/cell]
float32 resolution
# Map width [cells]
uint32 width
# Map height [cells]
uint32 height
# The origin of the map [m, m, rad]. This is the real-world pose of the
# cell (0,0) in the map.
geometry_msgs/Pose origin
```

Compact Message Definition

```
time map_load_time
float32 resolution
uint32 width
uint32 height
geometry_msgs/Pose origin
```

autogenerated on Wed, 02 Mar 2022 00:06:54

그림9. nav_msgs/MapMetaData 메시지 형식

추출한 맵 데이터는 압축된 형식으로 변환 후 RosBridge를 통해 전송하였다. 이 방식으로 스프링 서버는 커스텀 토픽을 구독하여 맵 데이터를 수신하고, 원본 맵 데이터를 복원 및 파싱하여 처리할 수 있도록 구현하였다.

맵 데이터는 Server-Sent Events (SSE)를 통해 클라이언트로 전송되었으며, 이를 통해 프론트엔드는 맵 데이터를 받아서 화면에 맵을 그릴 수 있도록 하였다. 이러한 과정은 로봇의 위치를 효과적으로 표시하기 위한 기반을 제공하며, 실시간으로 업데이트되는 맵 데이터를 클라이언트에게 제공하는 데 중요한 역할을 한다.

1-8. 로봇 위치 표시를 위한 프론트엔드 구현

```
// 서버로부터 'positionUpdate' 이벤트를 받았을 때 실행
eventSource.addEventListener('positionUpdate', function(event) {
    try {

        const positionData = JSON.parse(event.data);
        const x = positionData.x;
        const y = positionData.y;
        console.log(`Position X: ${x}, Y: ${y}`);

        // 맵 좌표를 캔버스 좌표로 변환
        const canvasX = (-y - mapOriginY) / mapResolution;
        const canvasY = (-x - mapOriginX) / mapResolution;
        console.log(`Canvas X: ${canvasX}, Y: ${canvasY}`)

        // 로봇 위치에 레드점 찍기
        if (positionData && x !== undefined && y !== undefined) {
            ctx.fillStyle = 'red';
            ctx.beginPath();
            ctx.arc(canvasX, canvasY, 5, 0, Math.PI * 2); // radius 5
            ctx.fill();
        }
    } catch (e) {
        console.error('Error parsing or displaying position data:', e);
    }
})
```

그림10. 프론트엔드 테스트 코드

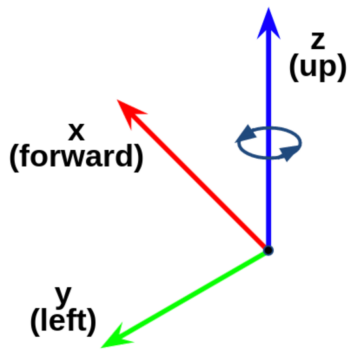


그림11. ROS 로봇 좌표계

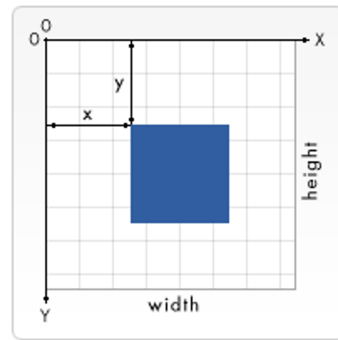
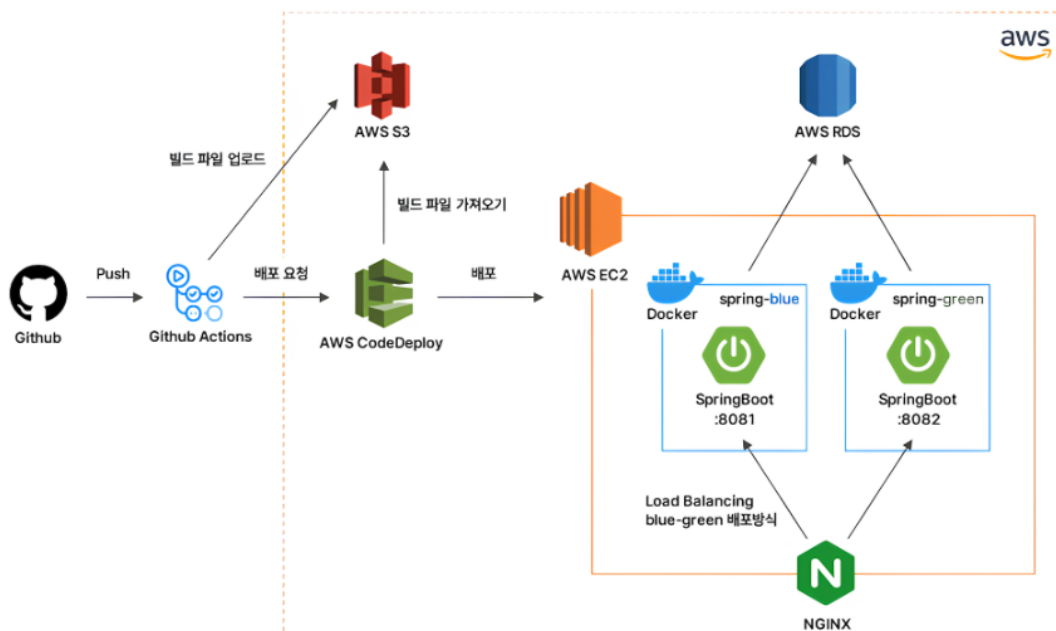


그림12. 프론트엔드 캔버스 좌표계

프론트엔드에서 로봇의 위치를 표시하기 위해, 서버로부터 'positionUpdate' 이벤트를 수신하는 코드가 작성되었다. 이 코드는 JSON 데이터를 파싱하여 로봇의 x와 y 좌표를 추출하고, 이를 맵의 해상도(resolution)를 이용해 캔버스 좌표로 변환하였다.

좌표 변환 과정에서 로봇 좌표계와 캔버스 좌표계 간의 차이로 인한 문제를 해결하기 위해, x와 y 좌표를 각각 -y와 -x로 변환하였으며, 원점의 차이를 반영하기 위해 -marginX와 -marginY 값을 적용하였다. 이를 통해 로봇의 위치가 맵에 정확히 표시되도록 구현하였다.

1-9. 서버 아키텍처



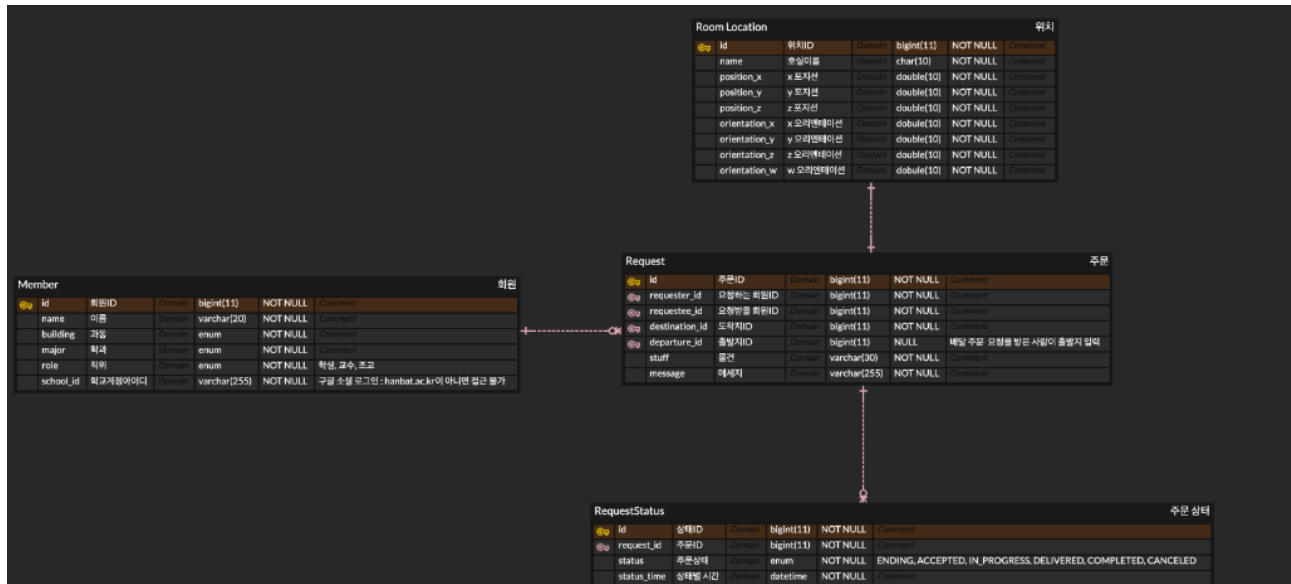
이 아키텍처는 Spring Boot 애플리케이션을 블루-그린(Blue-Green) 배포 방식으로 관리하는 파이프라인을 설명하며, 현재 모든 구성 요소가 구현된 상태이다. 애플리케이션의 소스 코드는 GitHub에 저장되며, 개발자가 코드를 Push하면 GitHub Actions가 트리거되어 자동으로 빌드와 배포 요청을 시작한다. GitHub Actions는 소스 코드를 빌드하여 결과 파일을 AWS S3에 업로드한다. 이후 AWS CodeDeploy가 배포 요청을 받아 AWS EC2 인스턴스로 빌드 파일을 가져온다. EC2 인스턴스는 Docker를 사용하여 Spring Boot 애플리케이션을 구동한다.

배포 방식으로는 블루-그린 배포가 사용된다. 두 개의 Docker 컨테이너(spring-blue와 spring-green)가 서로 다른 포트(8081, 8082)에서 Spring Boot 애플리케이션을 실행하며, NGINX를 통해 로드 밸런싱을

수행한다. 새로운 버전의 애플리케이션이 성공적으로 배포되면 트래픽이 새로운 컨테이너로 전환되며, 이를 통해 다운타임 없이 업데이트를 관리할 수 있다.

또한, 애플리케이션은 AWS RDS를 통해 데이터베이스와 연결되어 있어 데이터 관리를 중앙에서 통합적으로 수행한다. 이 아키텍처는 지속적 통합 및 배포(CI/CD)를 구현하여 빠르고 안정적인 애플리케이션 배포를 가능하게 한다.

1-10. ERD 다이어그램



이 ERD 다이어그램은 로봇 배달 시스템의 데이터베이스 구조를 설명하며, Member, Room Location, Request, RequestStatus라는 네 개의 주요 테이블로 나누어져 있다.

먼저, Member테이블은 회원 정보를 관리하기 위해 따로 분리되었다. 이 테이블은 회원 ID, 이름, 학과, 직위, 학교 계정 아이디 등의 정보를 저장하며, 사용자 인증 및 기본적인 신원 관리를 담당한다. 이를 별도의 테이블로 두는 이유는 회원 정보가 다양한 요청과 상태와 독립적으로 관리되어야 하기 때문이다.








Room Location테이블은 특정 위치 정보를 별도로 관리한다. 위치 정보는 로봇의 배달 경로 설정에 중요한 역할을 하므로, 이를 별도의 테이블로 유지하여 위치 데이터의 독립성과 관리 용이성을 높였다. 각 위치에 대한 상세한 좌표와 오리엔테이션 정보는 변경 가능성이 적은 고정된 데이터로서 회원 정보나 배달 요청과 구분된다.

Request테이블은 배달 요청과 관련된 정보를 관리하기 위해 분리되었다. 이 테이블에는 주문 ID, 요청자와 수락자의 회원 ID, 출발지 및 도착지 ID, 배달할 물건, 요청 메시지 등이 포함된다. 배달 요청은 회원 정보나 위치 정보와는 별도로, 각각의 요청에 대한 상세한 정보를 기록하고 관리하는 것이 중요하기 때문에 독립된 테이블로 분리하였다.

마지막으로, RequestStatus테이블은 각 배달 요청의 상태를 추적하기 위해 분리되었다. 상태 관리가 중요하기 때문에, 각 요청의 진행 상황을 독립적으로 기록할 수 있도록 하였다. 이는 배달 요청이 생성된 후 여러 상태를 거칠 수 있기 때문에, 상태 변화를 효율적으로 관리하고 추적하기 위해 필요하다.

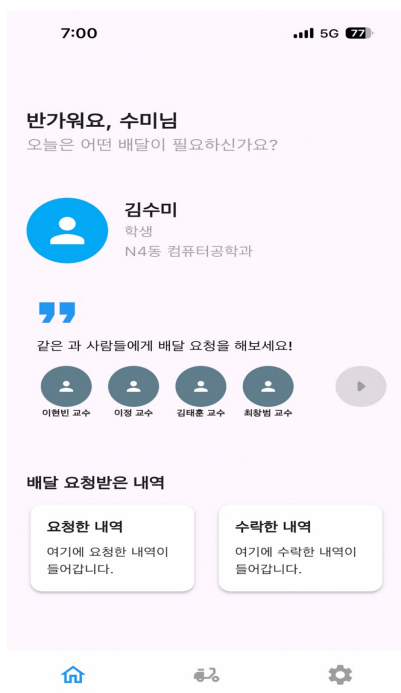
1-11. API 명세서

API 명세서 ...

| <input type="checkbox"/> | 권한 | 메서드 | Aa 요청주소 | 설명 |
|-------------------------------------|----|------|---|--|
| <input type="checkbox"/> | | POST | /user/school-account/login | 학교 계정 로그인 및 유저 등록 |
| <input checked="" type="checkbox"/> | | POST | /user/info/register  | 유저 정보 등록  |
| <input checked="" type="checkbox"/> | | GET | /user/info/get | 유저 정보 조회 |
| <input checked="" type="checkbox"/> | | GET | /people/same-major/list  | 같은 학과 사람들 조회 |
| <input checked="" type="checkbox"/> | | GET | /people/list/?name={name}&major={major}  | 이름과 학과 검색 시 다른 사람들 조회 |
| <input checked="" type="checkbox"/> | | POST | /delivery/request/send | 배달 요청 전송 |
| <input checked="" type="checkbox"/> | | POST | /delivery/request/accepted | 배달 요청 수락 → 로봇에게 명령 |
| <input checked="" type="checkbox"/> | | GET | /delivery/today/request?date={date} | 오늘 날짜의 배달 요청 내역 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/today/response?date={date} | 오늘 날짜의 배달 수락 내역 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/request/info?requestId={requestId} | 배달 요청 내역 상세 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/response/info?responseId={responseId} | 배달 수락 내역 상세 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/1week/list  | 이번주 배달 내역 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/1month/list  | 한달 배달 내역 조회 |
| <input checked="" type="checkbox"/> | | GET | /delivery/2month/list  | 세달 배달 내역 조회 |
| <input type="checkbox"/> | | | | |

이 API 명세서는 로봇 배달 시스템의 다양한 기능을 지원하는 API 엔드포인트들을 정의하고 있다. 총 15개의 엔드포인트가 있으며, 사용자 인증 및 정보 관리, 배달 요청 및 응답 처리, 배달 내역 조회 등의 기능을 제공한다. 현재, 전체 API 중 약 절반 정도가 구현된 상태이다.

1-12. 앱 UI 및 앱 설명



- 같은 과 사람들을 확인한 후, 보내고 싶은 사람에게 배달 요청하기 버튼을 누른다. 교수님, 조교님만 뜨도록 되어있다.
- 다른 사람에게 요청하고 싶을 경우, 화살표버튼을 클릭하고, 검색해 사람을 찾을 수 있다.
- 배달을 요청할 때, 수신자, 물건, 메시지, 출발지를 입력하여 보낸다.
- 배달 요청 수신자는 요청내역을 확인 후, 도착지를 입력하고, 수락 또는 거절 버튼을 누른다.
- 배달 수락되면, 로봇은 해당 출발지 위치로 네비게이션하여 이동하고, 물건을 받은 뒤에 도착지 위치로 이동해 물건을 전달한다.
- 물건이 전달되면 요청자와 요청 수신자 둘다 알림을 받는다.
- 홈화면에서 오늘 배달 요청한 내역과 수락한 내역을 확인할 수 있고, 상세 조회가 가능하다.
- 배달 화면에서는 현재 진행되고있는 배달 상황, 예상 위치 등을 확인할 수 있고, 완료된 배달 내역들만 1주, 1달, 2달로 필터링해 확인할 수 있다.

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)
- 팀원의 책임 및 역할 수행에 대한 결과, (문제점, 해결방안)
- 프로젝트 일정계획에 맞추지 못한 경우의 문제점, 해결 방안
- 요구사항 변경관리의 결과, (문제점, 해결 방안) 등등.

2-1. 문제해결을 위해 적용한 방법 및 결과

1) 맵 데이터 전송 오류

맵 데이터 전송 시, /map토픽에서 제공하는 nav_msgs/OccupancyGrid메시지가 너무 커서 서버에서 처리하기 어려운 오류가 발생하였다. 이 문제를 해결하기 위해, 먼저 ROS에서 커스텀 토픽을 생성하여 맵 데이터 중 필요한 정보만을 추출하였다. 이후, 추출된 데이터를 압축하여 전송함으로써 데이터 크기를 줄이고 네트워크 대역폭과 서버 자원을 절약하였다. 압축된 데이터를 RosBridge를 통해 스프링 서버로 전달하여, 데이터 전송의 효율성을 크게 개선하였다. 이 방법을 통해 데이터 전송 오류를 해결하고, 스프링 서버가 원활하게 데이터를 처리할 수 있게 되었다.

2) 로봇 위치 데이터 실시간 전송의 비효율성

로봇의 위치 데이터를 실시간으로 클라이언트에 전달하는 과정에서, 기존 방식으로는 데이터 전송의 지연이 발생하고, 클라이언트와 서버 간의 연결 유지가 비효율적이었다. 이를 해결하기 위해 Server-Sent Events (SSE)를 도입하였다. SSE는 서버에서 클라이언트로 단방향으로 데이터를 지속적으로 전송할 수 있게 해주며, 클라이언트는 단일 연결을 통해 실시간 업데이트를 수신할 수 있다. 이 방식으로 데이터 전송의 지연을 줄이고, 클라이언트가 최신 위치 정보를 즉시 받아볼 수 있게 되어, 실시간 데이터 전송의 효율성을 높였다.

2-2. 요구사항 변경관리의 결과

1) 원래 설계의 상태별 테이블 구조의 비효율성

초기 설계에서는 상태별로 각각의 테이블을 생성하는 방식이었으나, 이는 데이터베이스 구조를 복잡하게 하고 관리의 비효율성을 초래할 수 있었다. 이를 해결하기 위해 데이터베이스 구조를 간소화하여, 회원 정보는 Member테이블로, 위치 정보는 Room Location테이블로, 배달 요청 정보는 Request테이블로, 요청 상태는 RequestStatus테이블로 분리하였다. 이 접근 방식은 데이터의 독립성과 관리 용이성을 높이고, 시스템의 확장성과 유지보수를 용이하게 하며, 데이터 통합과 쿼리 성능을 향상시키는 데 기여하였다.

[별첨1]

프로젝트명 : 학교 내 자율주행 로봇 배달 서비스

소프트웨어 요구사항 정의서

Version 1.0

개발 팀원 명(팀리더): 김수미
이동현
송찬호

대표 연락처 : 010-8235-9619

e-mail: ymg12347@gmail.com

목차

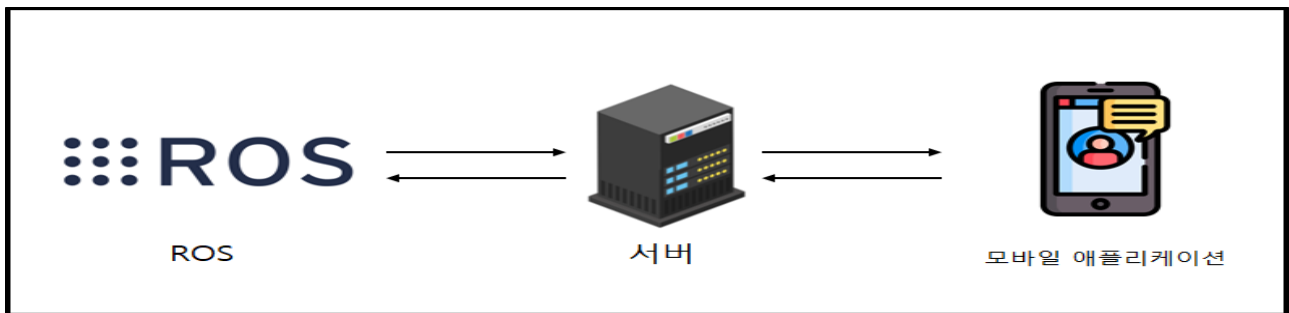
1. 개요
2. 시스템 장비 구성요구사항
3. 기능 요구사항
4. 성능 요구사항
5. 인터페이스 요구사항
6. 데이터 요구사항
7. 테스트 요구사항
8. 보안 요구사항
9. 품질 요구사항
10. 제약 사항
11. 프로젝트 관리 요구사항

1. 시스템 개요

- ROS 기반 자율주행 로봇 하드웨어 구조

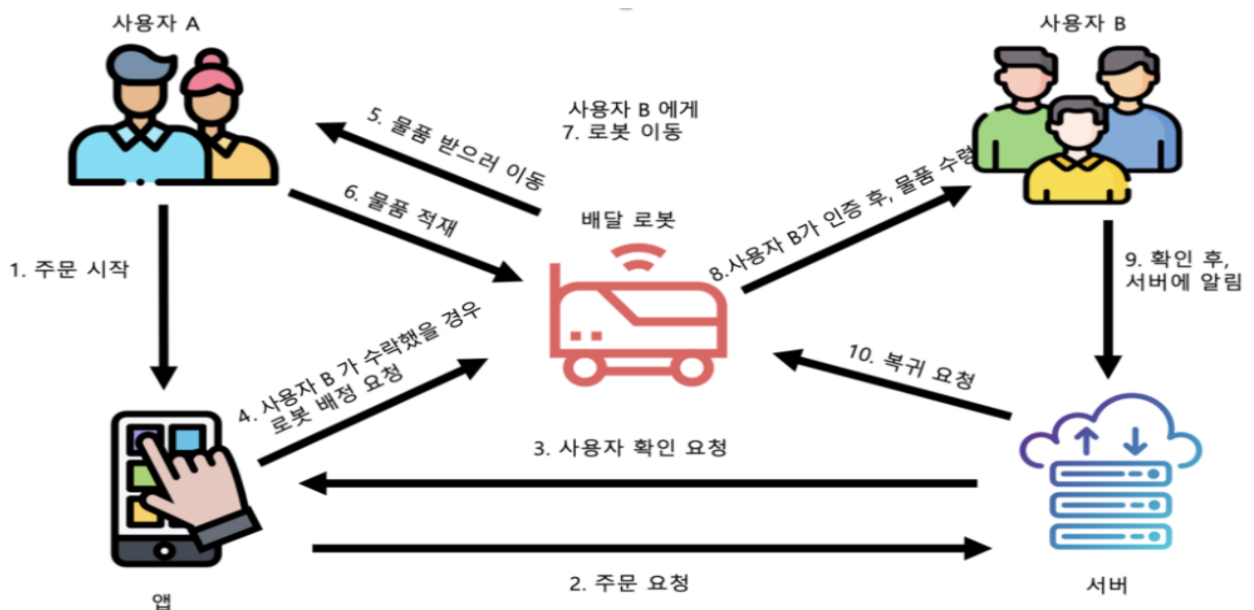


- 자율주행 배달 서비스 파이프라인



- 전체적인 시스템에 대한 시나리오

- 어플리케이션을 통해 주문을 하면 서버를 통해 로봇과 통신하여 로봇이 해당 목표지점까지 배달을 완료.



2. 시스템 장비 구성요구사항

| | | |
|-----------|-------|--|
| 요구사항 고유번호 | | ECR-001 |
| 요구사항 명칭 | | NVIDIA Jetson Orin Nano |
| 요구사항 분류 | | 시스템 장비 구성 |
| 상세 설명 | 정의 | NVIDIA Jetson Orin Nano 개발자 키트에는 SD 카드 슬롯, 참조 캐리어 보드, 사전 조립된 방열판/팬, 19V DC 전원 공급 장치, M.2-Key E 기반 무선 네트워킹 모듈이 탑재된 특수 NVIDIA Orin Nano 8GB 모듈이 포함되어 있습니다. 부팅 가능한 마이크로SD 카드 슬롯 외에도 고속 저장을 위해 캐리어 밑면에 2개의 M.2 Key-M NVMe 소켓이 제공됩니다. |
| | 세부 내용 | <p>GPU - 32개의 Tensor 코어가 있는 NVIDIA Ampere 아키텍처</p> <p>CPU - 6코어 Arm Cortex-A78AE v8.2 64비트 CPU 1.5 MB L2 + 4 MB L3</p> <p>메모리 - 8 GB 128비트 LPDDR5 초당 68 GB</p> <p>저장장치 - 마이크로SD 슬롯을 통한 외부 M.2 Key M을 통한 외부 NVMe</p> <p>전력 - 7 W ~ 15 W</p> <p>카메라 - 2x MIPI CSI-2 22핀 카메라 커넥터</p> <p>M.2 Key M - 3세대 x4 PCIe</p> <p>M.2 Key M - 3세대 x2 PCIe</p> <p>M.2 Key E - PCIe (x1), USB 2.0, UART, I2S, I2C</p> <p>USB - A 타입: 4x USB 3.2 2세대 C타입: 디버그 및 디바이스 모드용1x</p> <p>네트워킹 - 1x GbE 커넥터</p> <p>디스플레이 - 디스플레이포트 1.2 (+MST)</p> <p>마이크로SD 슬롯 - 최대 SDR104 모드 UHS-1 카드</p> <p>기타 - 40핀 확장 헤더(UART, SPI, I2S, I2C, GPIO)12핀 버튼 헤더 4핀 팬 헤더 DC 전원 잭</p> <p>크기 - 100 mm x 79 mm x 21 mm(피트, 캐리어 보드, 모듈 및 열 솔루션 포함 높이)</p> |
| 관련 요구사항 | | . |
| 요구사항 출처 | | https://developer.nvidia.com/ko-kr/blog/develop-ai-powered-robots-smart-vision-systems-and-more-with-nvidia-jetson-orin-nano-developer-kit/ |

| | | |
|-------------------|----------|--|
| 요구사항 고유번호 | | ECR-002 |
| 요구사항 명칭 | | RGB-D Camera |
| 요구사항 분류 | | 시스템 장비 구성 |
| 요구사항 상세 설명 | 정의 | 컬러 이미지와 깊이 정보를 동시에 캡처할 수 있는 카메라입니다. "RGB"는 빨간색, 초록색 및 파란색 채널의 색상 정보를 나타내며, "D"는 각 픽셀에서 카메라까지의 거리를 나타냅니다. |
| | 세부 내용 | <p>소프트웨어 지원: RGB-D 카메라와 함께 사용할 수 있는 소프트웨어 및 SDK(Software Development Kit)가 필요합니다.</p> <p>인터페이스 및 호환성: RGB-D 카메라는 일반적으로 USB 또는 기타 인터페이스를 통해 컴퓨터 또는 장치에 연결됩니다.</p> <p>시야: 수평 54 도, 수직 45 도</p> <p>깊이 이미지 크기: VGA(640x480)</p> <p>공간 X/Y 해상도 @ 0.5m: 1mm</p> <p>깊이 Z 해상도 @ 0.5m: <1mm</p> <p>최대 이미지 처리량: (QVGA) 60fps, (VGA) 30fps</p> <p>작동 범위: 0.35-3m</p> <p>컬러 이미지 크기: 1280x960</p> <p>오디오: 내장 마이크-마이크 2 개, 디지털 입력-4 시간</p> <p>데이터 인터페이스: USB 2.0, USB 3.0</p> <p>전력 소비: 2.25W</p> <p>전원 공급 장치: USB 2.0, USB 3.0</p> <p>치수: 18x2.5x3.5cm</p> <p>무게: 0.5lb</p> <p>운영 환경: 실내</p> |
| 관련 요구사항 | | . |
| 요구사항 출처 | | https://item.gmarket.co.kr/Item?goodscode=3641652410&GoodsSale=Y&jaehuid=200001169&NaPm=ct%3Dlvpzvltc%7CCi%3Dd9db57a79c4f2e2c04e12e6068aa3cd395cd9ff7%7Ctr%3DsIsI%7Csn%3D24%7Chk%3D6f65e6c6604d0f6fe2a4fd2e00b2451546b805c9 |

| | | |
|-------------------|----------|---|
| 요구사항 고유번호 | | ECR-003 |
| 요구사항 명칭 | | LIDAR Sensor |
| 요구사항 분류 | | 시스템 장비 구성 |
| 요구사항 상세 설명 | 정의 | 레이저를 사용하여 대상으로 광을 발사하고, 반사된 광을 측정하여 정확한 거리 및 공간 정보를 제공하는 광학 원격 감지 기술 센서. |
| | 세부 내용 | <p>소프트웨어 및 통합 지원: 센서와 함께 제공되는 소프트웨어 및 API(Application Programming Interface)가 있어야 합니다. 이는 센서 데이터를 효과적으로 처리하고 다른 시스템에 통합하는 데 필요합니다.</p> <p>360도 전방향 스캐닝 거리 측정</p> <p>작은 거리 오차, 안정적인 성능 및 높은 정확도</p> <p>거리 측정은 30m 이상</p> <p>강력한 주변광 간섭에 대한 저항성</p> <p>안정적인 성능을 위한 산업용 등급 무브러시 모터 구동</p> <p>레이저 출력이 1등급 레이저 안전 기준을 충족</p> <p>5-12Hz의 적응형 스캐닝 주파수 (사용자 정의 지원)</p> <p>광자성 융합 기술을 이용한 무선 통신, 무선 전원 공급</p> <p>측정 주파수 최대 20kHz (사용자 정의 지원)</p> |
| 관련 요구사항 | | . |
| 요구사항 출처 | | https://www.devicemart.co.kr/goods/view?no=12533064&market=nave&NaPm=ct%3Dlvq09sm8%7Cci%3De5aed2bff1cf27dda7b6aaadfcece354173694ae%7Ctr%3Dslct%7Csn%3D876973%7Chk%3Ded998dff474801c47654c9bd5222ae9433afdb |

3. 기능 요구사항

| | | |
|------------|-------|---|
| 요구사항 고유번호 | | SFR-001 |
| 요구사항 명칭 | | 사용자 배달 서비스 요청 |
| 요구사항 분류 | | 기능 |
| 요구사항 상세 설명 | 정의 | 모바일 애플리케이션을 통해 배달 서비스를 이용하기 위해 사용자는 양식을 작성 및 요청 |
| | 세부 내용 | - 작성된 정보를 서버로 전송 |
| 산출정보 | | . |
| 관련 요구사항 | | . |
| 요구사항 출처 | | . |

| | | |
|------------|-------|--|
| 요구사항 고유번호 | | SFR-002 |
| 요구사항 명칭 | | SLAM & NAVIGATION |
| 요구사항 분류 | | 기능 |
| 요구사항 상세 설명 | 정의 | 로봇이 외부 센서(LIDAR, 카메라)를 사용하여 주변 환경을 감지하고 이를 기반으로 자신의 위치를 추정하며, 동시에 환경의 구조를 매핑합니다. SLAM에서 생성된 지도를 사용하여 로봇이 목표 지점까지 경로를 계획하고 이동합니다. |
| | 세부 내용 | - SLAM 통한 실내 맵 확보 - 서버를 통해 전송받은 정보로 로봇에게 NAVIGATION 좌표 지정 - 로봇이 배달 작업을 마친 후 서버에게 결과 반환 및 사용자에게 알림 |
| 산출정보 | | . |
| 관련 요구사항 | | . |
| 요구사항 출처 | | . |

| | | |
|------------|-------|--|
| 요구사항 고유번호 | | SFR-003 |
| 요구사항 명칭 | | 배달 결과 확인 |
| 요구사항 분류 | | 기능 |
| 요구사항 상세 설명 | 정의 | 실시간으로 YOLO를 사용하여 호실 간판 객체를 탐지합니다. 객체의 영역에 있는 간판 이미지 정보를 텍스트 정보로 변환합니다. |
| | 세부 내용 | - 로봇이 배달을 통해 최종적으로 도착한 호실이 맞는지 호실 객체 탐지 수행 - 탐지된 영역에 대해 OCR을 통해 텍스트 검출 - 서버를 통해 전송받은 정보와 검출한 텍스트 정보 대조 및 판별 후 서버에 배달 결과 반환 |
| 산출정보 | | . |
| 관련 요구사항 | | . |
| 요구사항 출처 | | . |

4. 성능 요구사항

| | | |
|---------------|----------|---|
| 요구사항 고유번호 | | PER-001 |
| 요구사항 명칭 | | 통신 지연 시간 최소화 |
| 요구사항 분류 | | 성능 |
| 요구사항 상세 설명 | 정의 | 데이터를 송수신하는 데 걸리는 시간을 최대한 단축하여 시스템의 실시간성과 반응성을 향상시킨다. |
| | 세부 내용 | - 모바일 애플리케이션, 서버, 로봇이 배달 서비스 과정에서 단계마다 필요 정보를 주고받기 때문에 통신 지연 시간이 길어질수록 응답속도도 늦어져 좋지 못한 사용자 경험을 만든다. |
| 산출정보 | | . |
| 관련 요구사항 | | . |
| 요구사항 출처 | | . |

5. 인터페이스 요구사항

| | | | | |
|------------|-------|---|------|-----|
| 요구사항 고유번호 | | SIR-001 | | |
| 요구사항 명칭 | | 입력창 | | |
| 요구사항 분류 | | 사용자 인터페이스 | 응락수준 | 필 수 |
| 요구사항 상세 설명 | 정의 | 호실 입력창 구현 | | |
| | 세부 내용 | <ul style="list-style-type: none"> - 사용자가 목표로 하는 해당 호실을 입력하는 인터페이스 구현 - 사용자의 현재 위치와 해당 목표 위치를 구분하여 인터페이스를 표시하여 구현 | | |

| | | | | |
|------------|-------|---|------|-----|
| 요구사항 고유번호 | | SIR-001 | | |
| 요구사항 명칭 | | 알림창 | | |
| 요구사항 분류 | | 사용자 인터페이스 | 응락수준 | 필 수 |
| 요구사항 상세 설명 | 정의 | 도착 시, 알림창 구현 | | |
| | 세부 내용 | <ul style="list-style-type: none"> - 로봇이 사용자에게 도착했다는 알림창을 구현하여 보내는 사람과 받는 사람에게 알림이 가도록 구현 - 알림을 받고 확인을 하면 다시 되돌아가는 인터페이스 구현 | | |

6. 데이터 요구사항

| | | | |
|------------|--|------|-----|
| 요구사항 고유번호 | DAR-001 | | |
| 요구사항 명칭 | 데이터셋 구축 | | |
| 요구사항 분류 | 데이터 | 응락수준 | 필 수 |
| 요구사항 상세 설명 | <ul style="list-style-type: none"> - SLAM을 통한 MAP 데이터 구축하여 yaml파일로 저장 - YOLO 작업을 위한 복도와 여러 이미지셋 데이터셋 수집 후 저장 | | |

| | | | |
|------------|--|------|-----|
| 요구사항 고유번호 | DAR-001 | | |
| 요구사항 명칭 | 전처리된 데이터 | | |
| 요구사항 분류 | 데이터 | 응락수준 | 필 수 |
| 요구사항 상세 설명 | <ul style="list-style-type: none"> - 저장된 MAP 데이터를 조정하여 보다 세밀한 MAP 구축 - 이미지데이터셋에 여러 albumentations 적용하여 새로운 데이터셋 저장 | | |

7. 테스트 요구사항

| | | | |
|------------|--|------|-----|
| 요구사항 고유번호 | TER-001 | | |
| 요구사항 명칭 | 성능 테스트 | | |
| 요구사항 분류 | 테스트 | 응락수준 | 필 수 |
| 요구사항 상세 설명 | <ul style="list-style-type: none">- YOLOv8과 SLAM 중 자율 주행을 할 때, 장애물 회피 능력과 해당 목표지점까지 정확도 성능을 비교하여 평가- 기존의 파라미터값과 인공지능 기술을 앙상블 적용하여 성능을 비교하여 평가 | | |

8. 보안 요구사항

| | |
|------------|---|
| 요구사항 고유번호 | SER-001 |
| 요구사항 명칭 | 인증 및 권한 보안 |
| 요구사항 분류 | 보안 |
| 요구사항 상세 설명 | <ul style="list-style-type: none">- 앱에서 필요 없는 기능의 권한이나 개인정보를 요청하지 않도록 함- 학교 구글 계정으로 로그인을 진행하고, 사용자 인증을 수행하여 학교 내 사용자만 앱에 접근할 수 있도록 함 |

| | |
|------------|---|
| 요구사항 고유번호 | SER-002 |
| 요구사항 명칭 | 사용자 인터페이스 보안 |
| 요구사항 분류 | 보안 |
| 요구사항 상세 설명 | <ul style="list-style-type: none">- 시스템 화면에는 권한이나 인증 절차 없이 개인정보가 노출되지 않아야함- 사용자의 인증 상태에 따라 시스템의 화면에 보여지는 정보가 동적으로 변경되어야함 |

| | |
|------------|--|
| 요구사항 고유번호 | SER-003 |
| 요구사항 명칭 | 데이터 보안 |
| 요구사항 분류 | 보안 |
| 요구사항 상세 설명 | <ul style="list-style-type: none">- 데이터베이스 접속은 관리자만 접속 가능해야하며, AWS IAM 통해 데이터베이스 접속 권한을 제어함- 코드를 공유할 때, DB 설정 파일이 같이 올라가지 않도록 해야함 |

9. 품질 요구사항

| | | |
|------------|-------|---|
| 요구사항 고유번호 | | QUR-001 |
| 요구사항 명칭 | | 신뢰성 |
| 요구사항 분류 | | 품질 |
| 요구사항 상세 설명 | 정의 | 품질관리 |
| | 세부 내용 | <ul style="list-style-type: none"> - 로봇이 배달하는 과정에서 장애물 탐지, 호실 찾기 등 자율주행이 원활하게 되어야함 - 배달 물품의 보안을 잘 유지시킬 필요가 있음 |

| | | |
|------------|-------|---|
| 요구사항 고유번호 | | QUR-002 |
| 요구사항 명칭 | | 사용성 |
| 요구사항 분류 | | 품질 |
| 요구사항 상세 설명 | 정의 | 품질관리 |
| | 세부 내용 | <ul style="list-style-type: none"> - 사용자가 시스템을 쉽게 운용할 수 있도록 직관적인 UI/UX 디자인이 필요함 - 로봇으로부터 물품을 수령하거나 로봇에게 물품을 전달할 때, 조작이 어렵지 않아야함 |

| | | |
|------------|-------|--|
| 요구사항 고유번호 | | QUR-003 |
| 요구사항 명칭 | | 유지관리성 |
| 요구사항 분류 | | 품질 |
| 요구사항 상세 설명 | 정의 | 품질관리 |
| | 세부 내용 | <ul style="list-style-type: none"> - 시스템에 문제가 발생할 경우, 문제를 신속하게 해결하기 위한 유지보수 및 복구 방안이 있어야함 |

10. 제약 사항

| | | |
|------------|-------|---|
| 요구사항 고유번호 | | COR-001 |
| 요구사항 명칭 | | 시스템 개발 제약사항 |
| 요구사항 분류 | | 제약사항 |
| 요구사항 상세 설명 | 세부 내용 | <ul style="list-style-type: none"> - 로봇 프로그래밍을 위해서는 ROS 환경을 이용해야하고, 해당 환경에서 특정 프로그래밍 언어(python, C++)를 사용해야함 - 서버 개발을 위해 프레임워크 Spring boot를 사용할 수 있어야하고, 프로그래밍 언어 Java를 다룰 수 있어야함 - 앱 개발을 위해 프레임워크 Flutter를 사용할 수 있어야하고, 프로그래밍 언어 Dart를 다룰 수 있어야함 - 애자일 방법론을 준수해야함 |

| | | |
|------------|-------|---|
| 요구사항 고유번호 | | COR-002 |
| 요구사항 명칭 | | 설계 및 구현 제약사항 |
| 요구사항 분류 | | 제약사항 |
| 요구사항 상세 설명 | 세부 내용 | <ul style="list-style-type: none"> - Jetson nano로는 Yolo, OCR, Slam을 모두 수행하기에 연산량이 부족할 가능성이 있으므로 Jetson Orin Nano을 로봇의 구성품으로 사용할 필요가 있음 |

| | | |
|------------|-------|--|
| 요구사항 고유번호 | | COR-003 |
| 요구사항 명칭 | | 데이터 제약사항 |
| 요구사항 분류 | | 제약사항 |
| 요구사항 상세 설명 | 세부 내용 | <ul style="list-style-type: none"> - 사용자의 개인 정보나 기타 민감한 데이터를 안전하게 보호해야하므로 데이터 암호화, 접근 제어, 백업 및 회복 정책 등을 준수해야함 |

11. 프로젝트 관리 요구사항

| | |
|------------|--|
| 요구사항 고유번호 | PMR-001 |
| 요구사항 명칭 | 프로젝트 수행 조직 |
| 요구사항 분류 | 프로젝트 관리 |
| 요구사항 세부 내용 | <ul style="list-style-type: none"> - 로봇 개발 (송찬호) <ul style="list-style-type: none"> • 로봇 하드웨어 및 소프트웨어 개발 • 로봇 자율주행 알고리즘 적용 - 백엔드 개발 (김수미) <ul style="list-style-type: none"> • 데이터베이스 설계 및 관리 • API 엔드포인트 개발 - 프론트엔드 개발 (이동현) <ul style="list-style-type: none"> • 사용자 인터페이스 개발 • 사용자 입력 기능 구현 |

| | |
|------------|---|
| 요구사항 고유번호 | PMR-001 |
| 요구사항 명칭 | 프로젝트 일정계획 |
| 요구사항 분류 | 프로젝트 관리 |
| 요구사항 세부 내용 | <ul style="list-style-type: none"> - 프로젝트 계획 단계에서는 요구사항 분석 및 명세화, 시스템 설계 및 구조화, 로봇 ROS 개발 환경 설정, 서버 및 앱 개발 환경 설정이 필요하다. 이 단계에서는 요구사항 명세서, 아키텍처 설계서, 프로젝트 일정서 제출한다. - 프로젝트 개발 단계에서는 백엔드 서비스 및 API 개발, 프론트엔드 화면 개발, 자율주행 로봇 하드웨어 및 소프트웨어 개발, 로봇과 서버/서버와 앱 통신 개발이 필수적이다. 이에 따라 개발한 코드 산출물을 제출한다. - 프로젝트 테스트 단계에서는 실제로 로봇 배달 시스템 운영해보며 테스트하고, 문제점을 해결해야한다. |