**포팅메뉴얼**

**C101 로봇은 퇴근하지마**

**팀장: 서영은**

**팀원: 고태연, 권오준, 권우현, 김성훈, 신현학**

**[Gazebo 프로젝트]**

1. **개요**

이 문서는 GAZEBO\_SIM 프로젝트의 사전 설치 라이브러리, 파일 구조, 각 노드의 역할, 실행 명령어, 그리고 커스텀 메시지 사용 및 필수 설치 항목에 대해 상세히 설명합니다.

1. **사전 설치 라이브러리**

프로젝트에서 사용되는 주요 라이브러리는 다음과 같습니다:

1. **libgeographic-dev**

* gps\_common 패키지는 ROS 1용으로 개발되었으며, ROS 2용 공식 버전은 제공되지 않습니다.
* UTM 좌표 변환용 C++ 라이브러리

1. **networkx**

* 그래프 구조 파이썬 라이브러리

1. **ros-humble-gazebo-\***

* GAZEBO 시뮬레이션 관련 라이브러리

1. **ros-humble-velodyne\***

* Velodyne 관련 라이브러리

1. **mrpt-apps**

* Velodyne 관련 라이브러리

1. **Eigen3**
2. **pcl**

* Point Cloud 관련 라이브러리

1. **파일 구조**

프로젝트의 파일 구조는 아래와 같습니다

|  |
| --- |
| …/SIMULATE/GAZEBO\_SIM  └── ssafy\_ws  ├── src  │ ├── ~~map\_maker\_pkg~~  │ │ └── map\_maker.cpp  │ ├── robot\_control\_pkg  │ │ ├── joystick\_control.cpp  │ │ ├── key\_publisher.cpp  │ │ ├── middle\_teleop\_node.cpp  │ │ └── robot\_patrol.cpp  │ ├── robot\_custom\_inerfaces  │ │ ├── msg  │ │ │ └── Status.msg  │ │ └── srv  │ │ ├── Estop.srv  │ │ ├── Homing.srv  │ │ ├── Manual.srv  │ │ ├── Navigate.srv  │ │ ├── Patrol.srv  │ │ └── Waiting.srv  │ ├── robot\_planning\_pkg  │ │ └── global\_path\_planner.py  │ ├── robot\_status\_publisher  │ │ └── one\_robot\_status\_pub.cpp  │ ├── robot\_vision\_pkg  │ │ └── velodyne\_detection.cpp  │ ├── robot\_ai\_pkg  │ │ └── ai\_process.cpp  │ ├── gps\_hz\_pkg  │ │ └── gps.py  │ ├── total\_launch\_pkg  │ │ └── launch  │ │ └── ssafy\_robot\_launch.py  │ ├── rosbridge\_suite  │ ├── turtlebot3\_description  │ ├── turtlebot3\_simulations  │ └── velodyne\_simulator  └── maps  ├── global\_map\_maker.py  ├── global\_map.json  ├── json\_viewer.py  ├── path\_to\_png.py  └── robot\_plot.py |

1. **노드 설명**
2. **robot\_control\_pkg**

* joystick\_control.cpp

/joy 토픽을 수신하여 /cmd\_vel로 명령을 발행하며, 조이스틱을 통해 로봇 제어를 구현합니다.

* key\_publisher.cpp

키보드 입력을 string 타입의 명령으로 변환하여 해당 토픽을 발행합니다.

* middle\_teleop\_node.cpp

string 타입의 명령을 수신하여 /cmd\_vel을 발행, 원격 조작에 사용됩니다.

* robot\_patrol.cpp

Pure-pursuit 알고리즘을 이용해 로봇이 지정된 경로를 따라 이동하도록 제어합니다.

1. **robot\_custom\_inerfaces**

* msg
* Status.msg

로봇 상태 정보를 담는 커스텀 메시지

* Srv
* Estop.srv : 로봇 정지 서비스
* Homing.srv: 로봇 귀환 서비스
* Manual.srv: 메뉴얼 조작 서비스
* Navigate.srv: 네비게이션 서비스
* Patrol.srv: 순찰 서비스
* Waiting.srv: 대기 서비스

1. **robot\_planning\_pkg**

* global\_path\_planner.py

A\* 알고리즘을 이용해 목적지까지의 최단 경로를 계산합니다. 순찰(Patrol) 모드일 경우, 순찰 경로까지의 Approach\_path와 Global\_path를 함께 계산합니다.

1. **robot\_status\_publisher**

* one\_robot\_status\_pub.cpp

로봇의 상태를 발행하며, 서비스 호출을 통해 로봇 상태 변경 및 다른 노드 실행을 수행합니다.

1. **robot\_vision\_pkg**

* velodyne\_detection.cpp

3D 라이다 센서를 이용해 전방 물체를 탐지하고, 정지 명령을 발행합니다.

1. **total\_launch\_pkg**

* launch/ssafy\_robot\_launch.py

로봇별 파라미터를 입력받아 전체 노드를 실행하는 런치 파일입니다.

1. **기타 패키지**

* rosbridge\_suite

ROS와 웹소켓을 통해 백엔드를 연결합니다.

* turtlebot3\_description

터틀봇3 모델을 GAZEBO에서 실행하기 위한 패키지입니다.

* turtlebot3\_simulations

GAZEBO에서 벨로다인 시뮬레이터를 실행하기 위한 패키지입니다.

1. **Maps**

* global\_map\_maker.py

로봇이 이동 가능한 경로를 Graph 형태의 JSON 파일로 저장합니다.

* global\_map.json

노드 간 엣지 구조를 나타내는 JSON 파일입니다.

* json\_viewer.py

matplotlib을 이용하여 JSON 파일을 시각화합니다.

* path\_to\_png.py

경로 정보를 PNG 파일로 변환합니다.

* robot\_plot.py

로봇의 현재 위치, 목표점, 및 진행 방향을 시각화합니다.

1. **모델 경로 설정**

|  |
| --- |
| export ROS\_DOMAIN\_ID=30  export TURTLEBOT3\_MODEL=burger  export GAZEBO\_MODEL\_PATH=$GAZEBO\_MODEL\_PATH:/home/ubuntu/S12P11C101/SIMULATE/GAZEBO\_SIM/ssafy\_ws/src/turtlebot3\_simulations/turtlebot3\_gazebo/models/ |

위 내용에서 경로를 자신의 위치로 맞춰서 ~/.bashrc에 추가하세요.

 **예시**

|  |
| --- |
| echo 'export ROS\_DOMAIN\_ID=30' >> ~/.bashrc  echo 'export TURTLEBOT3\_MODEL=burger' >> ~/.bashrc  echo 'export GAZEBO\_MODEL\_PATH=$GAZEBO\_MODEL\_PATH:/home/ubuntu/S12P11C101/SIMULATE/GAZEBO\_SIM/ssafy\_ws/src/turtlebot3\_simulations/turtlebot3\_gazebo/models/' >> ~/.bashrc  source ~/.bashrc |

1. **실행 명령어**
2. **가제보 실행**

모델을 불러오는데 시간이 꽤 걸립니다.

|  |
| --- |
| ros2 launch turtlebot3\_gazebo ssafy\_office.launch.py |

1. **런치 파일 실행 (로봇별)**

|  |
| --- |
| ros2 launch total\_launch\_pkg ssafy\_robot\_launch.py robot\_name:=ssafy robot\_number:=1 |

|  |
| --- |
| ros2 launch total\_launch\_pkg ssafy\_robot\_launch.py robot\_name:=samsung robot\_number:=2 |

|  |
| --- |
| ros2 launch total\_launch\_pkg ssafy\_robot\_launch.py robot\_name:=gwangju robot\_number:=3 |

1. **키보드 입력 (메뉴얼 모드 변경 후)**

|  |
| --- |
| ros2 run robot\_control\_pkg key\_publisher --ros-args -p robot\_name:=ssafy -p robot\_number:=1 |

1. **서비스 호출 명령어**

* 호밍서비스

|  |
| --- |
| ros2 service call /robot\_1/homing robot\_custom\_interfaces/srv/Homing |

* 네비게이트 서비스

|  |
| --- |
| ros2 service call /robot\_1/navigate robot\_custom\_interfaces/srv/Navigate "{goal: {x: 304401.2780476108, y: 3892837.505477577, theta: 0.0}}" |

* 패트롤 서비스

|  |
| --- |
| ros2 service call /robot\_1/patrol robot\_custom\_interfaces/srv/Patrol "{goals: [{x: 304401.2780476108, y: 3892837.505477577, theta: 0.0}, {x: 304417.68651993107, y: 3892838.6313477717, theta: 0.0}, {x: 304417.91822496144, y: 3892850.079599839, theta: 0.0}]}" |

* Estop 서비스

|  |
| --- |
| ros2 service call /robot\_1/stop robot\_custom\_interfaces/srv/Estop |

* Temp\_stop 서비스

|  |
| --- |
| ros2 service call /robot\_1/temp\_stop robot\_custom\_interfaces/srv/Estop |

* Resume 서비스

|  |
| --- |
| ros2 service call /robot\_1/resume robot\_custom\_interfaces/srv/Estop |

* Waiting 서비스

|  |
| --- |
| ros2 service call /robot\_1/waiting robot\_custom\_interfaces/srv/Waiting |

* Manual 서비스 (모드 변경)

|  |
| --- |
| ros2 service call /robot\_1/manual robot\_custom\_interfaces/srv/Manual |

1. **여러 대의 로봇 동시 주행 예시**

* 예시 1:

|  |
| --- |
| ros2 service call /robot\_2/patrol robot\_custom\_interfaces/srv/Patrol "{goals: [{x: 304396.57, y: 3892844.46, theta: 0.0}, {x: 304401.15, y: 3892849.86, theta: 0.0}]}" |

* 예시 2:

|  |
| --- |
| ros2 service call /robot\_1/patrol robot\_custom\_interfaces/srv/Patrol "{goals: [{x: 304417.70, y: 3892839.59, theta: 0.0}, {x: 304417.77, y: 3892845.46, theta: 0.0}, {x: 304408.34, y: 3892851.10, theta: 0.0}]}" |

* 예시 3:

|  |
| --- |
| ros2 service call /robot\_3/patrol robot\_custom\_interfaces/srv/Patrol "{goals: [{x: 304401.85, y: 3892835.85, theta: 0.0}, {x: 304417.43, y: 3892837.16, theta: 0.0}]}" |

1. **커스텀 메시지 사용 및 필수 설치 항목**

**1)** **커스텀 메시지 사용**

GAZEBO\_SIM 프로젝트 내 ssafy\_ws 폴더 안에 위치한 robot\_custom\_interfaces 패키지에서 커스텀 메시지를 정의합니다.

CMakeLists.txt 수정 사항

* 패키지 찾기

robot\_custom\_interfaces 패키지를 찾기 위해 아래 코드를 추가합니다.

|  |
| --- |
| find\_package(robot\_custom\_interfaces REQUIRED) |

* 실행 파일 추가 및 의존성 설정

C++ 파일을 작성 후 실행하기 위해 아래와 같이 설정합니다.

|  |
| --- |
| add\_executable(robot\_status\_pub src/robot\_status\_pub.cpp)  ament\_target\_dependencies(robot\_status\_pub rclcpp std\_msgs robot\_custom\_interfaces)  install(TARGETS  robot\_status\_pub  DESTINATION lib/${PROJECT\_NAME}) |

package.xml 수정 사항

* 의존성 추가

package.xml 파일에 아래 내용을 추가하여 robot\_custom\_interfaces 패키지에 대한 의존성을 명시합니다.

|  |
| --- |
| <depend>robot\_custom\_interfaces</depend> |

**2) 필수 설치 항목**

프로젝트 실행을 위해 다음 명령어들을 사용하여 필수 라이브러리 및 패키지를 설치합니다.

ssafy\_ws/ 디렉토리에서 아래명령어 수행

|  |
| --- |
| rosdep update  rosdep install --from-paths src --ignore-src -r -y |

|  |
| --- |
| sudo pip3 install scikit-learn  sudo pip3 install networkx  sudo pip3 install catkin\_pkg  sudo pip3 install "numpy<1.25.0" |

|  |
| --- |
| sudo apt-get update  sudo apt-get install mrpt-apps  sudo apt-get install libeigen3-dev  sudo apt-get install ros-humble-geographic-msgs  sudo apt-get install libgeographic-dev  sudo apt-get install libpcl-dev  sudo apt-get install ros-hubmle-pcl-conversions  sudo apt-get install ros-hubmle-pcl-ros |

**[Issac Sim 프로젝트]**

1. **Isaac Sim 시뮬레이터 설치 매뉴얼**
2. **Isaac Sim (ver 4.2.0)**

공식 홈페이지 Linux version 다운

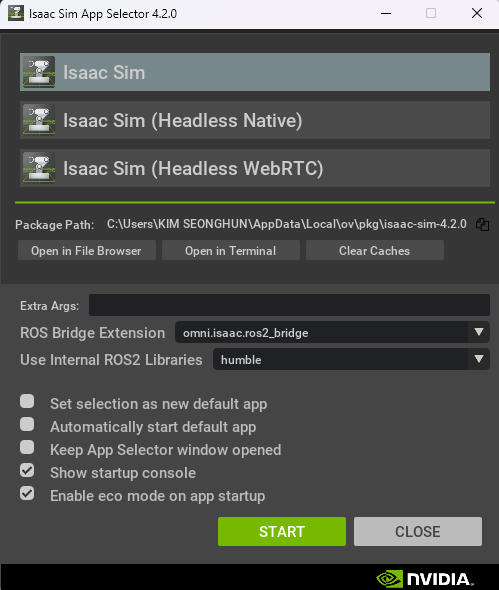
|  |
| --- |
| <https://download.isaacsim.omniverse.nvidia.com/isaac-sim-standalone%404.2.0-rc.18%2Brelease.16044.3b2ed111.gl.linux-x86_64.release.zip> |

1. **Terminal 창에 입력**

|  |
| --- |
| mkdir ~/isaacsim  cd ~/Downloads  unzip "isaac-sim-standalone@4.5.0-rc.36+release.19112.f59b3005.gl.linux-x86\_64.release.zip" -d ~/isaacsim  cd ~/isaacsim  ./post\_install.sh  ./isaac-sim.selector.sh |

이후 Terminal에 ./isaac-sim.sh. 입력하여 프로그램 구동하면 Isaac sim 실행가능

아래와 같은 창이 나타나면 옵션 두개를 바꿔줘야 한다.



1. **ROS Bridge Extension 에 omni.isaac.ros2\_bridge 선택**
2. **Use Internal ROS2 Libraries 에 humble 선택**
3. **파일 선택하여 오픈**

Isaac Sim 프로그램이 켜지면 File -> open ->

/S12P11C101/SIMULATE/GAZEBO\_SIM/Isaac\_sim\_usd/250217(7-Isaac\_bot.usd) 선택

1. **재생 버튼 클릭**

로딩이 완료 된 후 왼쪽 바의 재생 버튼으로 구동 준비 완료

1. **Isaac Sim ROS 2 topic 설명**
2. **/Isaacbot/velodyne**

Isaac sim 상의 로봇 Isaacbot의 Velodybe 3d Lidar 데이터 발행

1. **/Isaacbot/imu**

Isaac sim 상의 로봇 Isaacbot의 imu 데이터 발행

1. **/Isaacbot/cmd\_vel**

Isaac sim 상의 로봇 Isaacbot을 구동하는 데이터 구독

1. **/Isaacbot/front/img\_compressed**

Isaac sim 상의 로봇 Isaacbot의 전방 카메라 이미지 데이터 발행

1. **/Isaacbot/rear/img\_compressed**

Isaac sim 상의 로봇 Isaacbot의 후 방 카메라 이미지 데이터 발행

1. **Isaac sim의 서버로의 연결과 구동 전 topic 변환을 위한 과정 (각각의 terminal 창에 입력)**
2. **rosbridge 구동**

|  |
| --- |
| ros2 launch rosbridge\_server post\_ launch.xml |

1. **ssh 연결**

|  |
| --- |
| sudo ssh -i /home/ssafy/c101/I12C101T.pem -R \*:10001:localhost:10001 ubuntu@52.79.51.253 |

1. **영상 converting 패키지 설치**

|  |
| --- |
| sudo apt install ros-humble-compressed-image-transport |

1. **Front 카메라 이미지 변환**

|  |
| --- |
| ros2 run image\_transport republish raw --ros-args --remap in:=/Isaacbot/front/rgb --remap out:=/Isaacbot/front/img\_compressed |

1. **Rear 카메라 이미지 변환**

|  |
| --- |
| ros2 run image\_transport republish raw --ros-args --remap in:=/Isaacbot/rear/rgb --remap out:=/Isaacbot/rear/img\_compressed |

**[온디바이스 얼굴 인식 프로젝트]**

1. **주요 개발환경**

**jetpack version == 6.0  
CUDA == 12.2.140  
cuDNN == 8.9.4.25  
python == 3.10.12**

1. **핵심 라이브러리**

|  |  |
| --- | --- |
| **Package** | **Version** |
| dlib face-recognition face-recognition-models  numpy  pandas  pillow  pip  pygame  scikit-learn  torch  torchvision  ultralytics  ultralytics-thop | 19.24.2  1.3.0  0.3.0  1.26.1  2.2.3  11.1.0  25.0.1  2.6.1  1.6.1  2.3.0  0.18.0a0+6043bc2  8.3.74  2.0.14 |

1. **젯슨 오린 나노 CUDA 환경 세팅 매뉴얼**
2. **python == 3.10.12**

sudo apt update

sudo apt install -y software-properties-common

sudo add-apt-repository ppa:deadsnakes/ppa -y

sudo apt update

PPA 저장소 추가

sudo apt install -y python3.10

파이썬 3.10.12 버전 설치

python3 --version

버전 확인

1. **opencv == 4.10.0**

sudo apt update

sudo apt install -y build-essential cmake git unzip pkg-config \\

libjpeg-dev libpng-dev libtiff-dev \\

libavcodec-dev libavformat-dev libswscale-dev \\

libgtk-3-dev libcanberra-gtk\* \\

python3-dev python3-numpy python3-pip \\

libtbb2 libtbb-dev libdc1394-22-dev \\

libv4l-dev v4l-utils \\

libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \\

libavresample-dev libvorbis-dev libxine2-dev \\

libfaac-dev libmp3lame-dev libtheora-dev \\

libopencore-amrnb-dev libopencore-amrwb-dev \\

libopenblas-dev libatlas-base-dev libblas-dev \\

liblapack-dev libeigen3-dev gfortran \\

libhdf5-dev protobuf-compiler \\

libprotobuf-dev libgoogle-glog-dev libgflags-dev

필수 패키지 설치

cd ~

wget -O opencv.zip <https://github.com/opencv/opencv/archive/4.10.0.zip>

wget -O opencv\_contrib.zip <https://github.com/opencv/opencv\_contrib/archive/4.10.0.zip>

unzip opencv.zip

unzip opencv\_contrib.zip

소스 코드 다운로드

cd ~/opencv-4.10.0

mkdir build

cd build

빌드 디렉토리 생성

cmake -D CMAKE\_BUILD\_TYPE=RELEASE \\

-D CMAKE\_INSTALL\_PREFIX=/usr/local \\

-D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-4.10.0/modules \\

-D WITH\_CUDA=ON \\

-D WITH\_CUDNN=ON \\

-D OPENCV\_DNN\_CUDA=ON \\

-D CUDA\_ARCH\_BIN=8.7 \\

-D WITH\_GSTREAMER=ON \\

-D WITH\_V4L=ON \\

-D BUILD\_opencv\_python3=ON \\

-D BUILD\_TESTS=OFF \\

-D BUILD\_PERF\_TESTS=OFF \\

-D BUILD\_EXAMPLES=OFF ..

Cmake 구성

nproc # 시스템의 코어 수 확인

make -j$(nproc)

sudo make install

sudo ldconfig

빌드 및 설치

python3 -c "import cv2; print(cv2.getBuildInformation())"

설치 확인

<https://forums.developer.nvidia.com/t/pytorch-for-jetson/72048>

nvidia jetson 호환성 공식 페이지

1. **pytorch == 2.3.0**
2. **torchvision == 0.18**

pip install torch-2.3.0-cp310-cp310-linux\_aarch64.whl

pip install torchvision-0.18.0a0+6043bc2-cp310-cp310-linux\_aarch64.whl

1. **dlib == 19.24.2**

sudo apt update

sudo apt install -y build-essential cmake git \\

libgtk-3-dev libboost-all-dev \\

python3-dev python3-pip python3-numpy

필수 패키지 설치

cd ~

git clone <https://github.com/davisking/dlib.git>

cd dlib

 dlib 소스 코드 다운로드

mkdir build

cd build

cmake .. -DDLIB\_USE\_CUDA=ON -DUSE\_AVX\_INSTRUCTIONS=ON

cmake 빌드 설정

cmake --build . --config Release -- -j$(nproc)

sudo make install

sudo ldconfig

dlib 빌드 및 설치

cd ..

python3 setup.py install --yes USE\_CUDA

 파이썬 바인딩 설치

import dlib

print(dlib.DLIB\_USE\_CUDA)

print(dlib.cuda.get\_num\_devices())

설치 확인

1. **numpy == 1.26.1**

pip install numpy==1.26.1

넘파이 다운그레이드

1. **YOLOv8 ultralytics == 8.3.74**

pip install ultralytics==8.3.74 --no-deps

호환성 해치지 않게 —no-deps 옵션 활성화

1. **sklean == 1.6.1**

pip install scikit-learn==1.6.1

scikit\_learn 설치

1. **face\_recognition == 1.3.0**

pip install face\_recogonition

face\_recognition 설치

1. **pygame == 2.6.1**

pip install pygame

pygame 설치

**[Frontend - ROBOCOP 관리자 페이지]**

1. **주요 개발환경**

**VS Code == 1.97.2**

**Vue == 3.3.11**

**Node.js == 20.18.2**

**JavaScript == 10.8.2**

1. **핵심 라이브러리**

|  |  |
| --- | --- |
| Package | Version |
| axios  chart.js  vue-chart-3  echarts  vue-echarts  pinia  three  vue-router  tailwind-css  vite  @vitejs/plugin-vue | 1.7.9  3.9.1  3.1.8  5.6.0  6.6.8  2.3.1  0.173.0  4.5.0  3.4.17  5.0.10 |

1. **빌드하기**

Npm i

Npm run build

[Backend – ROBOCOP 관리자 페이지]

1. **필요 라이브러리 (requirements)**
2. **Python Version: 3.10.12**

motor==3.1.1

pymongo[srv]==4.5.0

dnspython>=2.5.0

fastapi==0.109.0

starlette>=0.35.0,<0.36.0

alembic==1.13.1

1. **FastAPI & Web Dependencies**

uvicorn==0.27.0

httptools>=0.5.0

uvloop>=0.17.0

python-dotenv==1.0.0

pydantic==2.5.3

pydantic-settings==2.1.0

pydantic-core==2.14.6

python-multipart==0.0.6

python-jose[cryptography]==3.3.0

passlib[bcrypt]==1.7.4

bcrypt==4.0.1

email-validator==2.1.0.post1

beanie==1.25.0

1. **Utility Libraries**

anyio==4.8.0

click==8.1.8

h11==0.14.0

typing\_extensions==4.12.2

1. **HTTP & Network**

aiohttp==3.11.11

aiohappyeyeballs==2.4.4

aiosignal==1.3.2

async-timeout==5.0.1

certifi>=2024.2.2

idna==3.10

urllib3==1.26.5

roslibpy==1.8.1

opencv-python==4.11.0.86

av==13.1.0

aiortc==1.10.1

websockets==14.2

hangul-romanize==0.1.0

python-dotenv==1.0.0

redis==5.0.1

websockets

1. **테스트 의존성**

pytest==7.4.3

pytest-asyncio==0.21.1

httpx==0.25.1

pytest-cov==4.1.0

pytest-mock==3.12.0

aioresponses==0.7.4

freezegun==1.2.2

1. **EC2 서버 Nginx 설정**

server {

    listen 443 ssl;

    server\_name robocopbackendssafy.duckdns.org;

    ssl\_certificate /etc/letsencrypt/live/robocopbackendssafy.duckdns.org/fullchain.pem;

    ssl\_certificate\_key /etc/letsencrypt/live/robocopbackendssafy.duckdns.org/privkey.pem;

    root /var/www/html;

    index index.html;

    location /dev/api {

    proxy\_pass http://localhost:8900/api;

    proxy\_http\_version 1.1;

    proxy\_set\_header Upgrade $http\_upgrade;

    proxy\_set\_header Connection "upgrade";

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    }

    location /dev/docs {

    proxy\_pass http://localhost:8900/docs;

    proxy\_http\_version 1.1;

    proxy\_set\_header Upgrade $http\_upgrade;

    proxy\_set\_header Connection "upgrade";

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    }

    location /dev/ws/ {

        proxy\_pass http://localhost:8900/ws/;

        proxy\_http\_version 1.1;

        proxy\_set\_header Upgrade $http\_upgrade;

        proxy\_set\_header Connection "upgrade";

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        proxy\_set\_header X-Forwarded-Proto $scheme;

        # WebSocket 타임아웃 설정 증가

        proxy\_read\_timeout 86400;    # 24시간

        proxy\_send\_timeout 86400;    # 24시간

        proxy\_connect\_timeout 60s;   # 연결 타임아웃

        # WebSocket 버퍼 설정

        proxy\_buffer\_size 64k;

        proxy\_buffers 8 32k;

        proxy\_busy\_buffers\_size 128k;

        # WebSocket 연결 유지

        proxy\_set\_header Connection "Upgrade";

    }

    location / {

        try\_files $uri $uri/ /index.html;

    }

    # FastAPI 문서

    location /docs {

        proxy\_pass http://localhost:8000/docs;

        proxy\_http\_version 1.1;

        proxy\_set\_header Upgrade $http\_upgrade;

        proxy\_set\_header Connection "upgrade";

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

    }

    location /openapi.json {

        proxy\_pass http://localhost:8000/openapi.json;

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

    }

    # FastAPI API 엔드포인트

    location /api {

        proxy\_pass http://localhost:8000/api;

        proxy\_http\_version 1.1;

        proxy\_set\_header Upgrade $http\_upgrade;

        proxy\_set\_header Connection "upgrade";

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

    proxy\_connect\_timeout 300;

    proxy\_send\_timeout 300;

    proxy\_read\_timeout 300;

    fastcgi\_send\_timeout 300;

    fastcgi\_read\_timeout 300;

    }

    # 미디어 파일 서빙

    location /media/ {

        alias /var/www/media/;

    expires 30d;

    add\_header Cache-Control "public,, no-transform";

    access\_log off;

        autoindex on;

        try\_files $uri $uri/ =404;

        add\_header 'Access-Control-Allow-Origin' '\*';

        add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';

        add\_header 'Access-Control-Allow-Headers' '\*';

    }

    location /ws/ {

        proxy\_pass http://localhost:8000/ws/;

        proxy\_http\_version 1.1;

        proxy\_set\_header Upgrade $http\_upgrade;

        proxy\_set\_header Connection "upgrade";

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        proxy\_set\_header X-Forwarded-Proto $scheme;

        # WebSocket 타임아웃 설정 증가

        proxy\_read\_timeout 86400;    # 24시간

        proxy\_send\_timeout 86400;    # 24시간

        proxy\_connect\_timeout 60s;   # 연결 타임아웃

        # WebSocket 버퍼 설정

        proxy\_buffer\_size 64k;

        proxy\_buffers 8 32k;

        proxy\_busy\_buffers\_size 128k;

        # WebSocket 연결 유지

        proxy\_set\_header Connection "Upgrade";

    }

    location /sse {

        proxy\_pass http://localhost:8000/sse;

        proxy\_http\_version 1.1;

    proxy\_set\_header Connection 'keep-alive';

        proxy\_set\_header X-Accel-Buffering no;

        proxy\_set\_header Cache-Control no-cache;

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

    proxy\_set\_header X-Forwarded-Proto $scheme;

        proxy\_buffering off;

        proxy\_cache off;

    proxy\_read\_timeout 86400s;

    proxy\_send\_timeout 86400s;

    keepalive\_timeout 86400s;

    }

    location /api/v1/robots/sse {

        # Allow specific HTTP methods

        if ($request\_method !~ ^(GET|OPTIONS)$) {

           return 405;

        }

        proxy\_pass http://localhost:8000/api/v1/robots/sse;

        proxy\_http\_version 1.1;

    proxy\_set\_header Connection 'keep-alive';

        proxy\_set\_header X-Accel-Buffering no;

        proxy\_set\_header Cache-Control no-cache;

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

    proxy\_set\_header X-Forwarded-Proto $scheme;

        proxy\_buffering off;

        proxy\_cache off;

    proxy\_read\_timeout 86400s;

    proxy\_send\_timeout 86400s;

    keepalive\_timeout 86400s;

    }

    location /api/v1/lidar/sse {

        # Allow specific HTTP methods

        if ($request\_method !~ ^(GET|OPTIONS)$) {

           return 405;

        }

        proxy\_pass http://localhost:8000/api/v1/lidar/sse;

        proxy\_http\_version 1.1;

    proxy\_set\_header Connection 'keep-alive';

        proxy\_set\_header X-Accel-Buffering no;

        proxy\_set\_header Cache-Control no-cache;

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

    proxy\_set\_header X-Forwarded-Proto $scheme;

        proxy\_buffering off;

        proxy\_cache off;

    proxy\_read\_timeout 86400s;

    proxy\_send\_timeout 86400s;

    keepalive\_timeout 86400s;

    }

# 미디어 서버 API 엔드포인트

    location /api/upload {

    client\_max\_body\_size 10M;  # 파일 업로드 크기 제한

    proxy\_pass http://localhost:8100/api/upload;

    proxy\_http\_version 1.1;

    proxy\_set\_header Upgrade $http\_upgrade;

    proxy\_set\_header Connection "upgrade";

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

    proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        proxy\_set\_header X-Forwarded-Proto $scheme;

    }

}

# HTTP to HTTPS redirect

server {

    listen 80;

    server\_name robocopbackendssafy.duckdns.org;

    return 301 https://$server\_name$request\_uri;

}

1. **Docker File**

FROM python:3.10.12 AS builder

ENV PYTHONUNBUFFERED=1 \

    PYTHONDONTWRITEBYTECODE=1

WORKDIR /BACKEND\_WAS

# 시스템 라이브러리 설치

RUN apt-get update && apt-get install -y \

    libgl1-mesa-glx \

    libglib2.0-0 \

    && rm -rf /var/lib/apt/lists/\*

# requirements.txt 복사 및 의존성 설치

COPY requirements.txt .

COPY .env .env

RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사

COPY . .

# 실행 명령어

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]

1. **Docker-compose**

version: '3.8'

services:

  # ---------------------------------------

  # (A) WAS ????

  # ---------------------------------------

  was:

    build: .

    image: robocop-was:latest

    container\_name: robocop-was

    deploy:

      resources:

        limits:

          memory: 16G  # ?? ??

        reservations:

          memory: 10G  # ??? ??

    volumes:

      - ./app:/BACKEND\_WAS/app

      - ./storage:/BACKEND\_WAS/storage

      - /var/www/media:/BACKEND\_WAS/media

    env\_file:

      - .env

    depends\_on:

      - mongodb

      - redis

    network\_mode: host

    environment:

      - RMW\_IMPLEMENTATION=rmw\_fastrtps\_cpp

      - TZ=Asia/Seoul

    # WAS ?? ???? ENTRYPOINT? ???? ??? ?

  # ---------------------------------------

  # (B) MongoDB

  # ---------------------------------------

  mongodb:

    image: mongo

    container\_name: mongodb

    volumes:

      - mongodb\_data:/data/db

    network\_mode: host

    environment:

      - TZ=Asia/Seoul

  # ---------------------------------------

  # (C) Redis

  # ---------------------------------------

  redis:

    image: redis

    container\_name: redis

    volumes:

      - redis\_data:/data

    command: redis-server --save 60 1

    network\_mode: host

    environment:

      - TZ=Asia/Seoul