Apollo와 Simulator 연동

1. Apollo 설치

* Apollo에는 버전이 존재하며 해당 버전에따라 통신하는 방법이 다르다.
* Apollo 3.0 : ROS, Apollo 3.5 : Cyber RT, Apollo 5.0 : Cyber RT
* Cyber RT는 병렬처리 및 통신적인 부분에 있어 ROS보다 좋지만 추구하는 방법과 다르므로 진행하지 않음.

|  |  |
| --- | --- |
|  |  |
| Apollo 3.0 | Apollo 5.0 |

* Apollo-3.0 설치 전, LG simulator 설치를 진행한다.(설치 링크 : <https://github.com/lgsvl/simulator/blob/master/Docs/docs/build-instructions.md>)
* 설치는 편한 설치 및 연동성을 위해 original Apollo가 아닌 LG simulator GitHub에 있는 apollo-3.0을 설치한다.(설치 링크 : <https://github.com/lgsvl/apollo-3.0>)
* 해당 매뉴얼대로 설치 및 테스트하면 Apollo 관련 모든 설치는 끝난다.
* Docker를 이용하여 설치하므로 Docker 밖에서 설정하는 것은 매우 제한적일 수 있다.
* Docker는 환경 구성을 더 편리하게 하도록 만든 리눅스 컨테이너로 생각하면 된다.
* Docker가 외부로 통신하기 위해서는 Docker 내부 포트와 Ubuntu 포트를 연결하여 외부에서 Docker로 직접적인 통신이 가능하도록 설정한다.   
  (./apollo/docker/script/dev\_start.sh 내부 265번째 줄의 내용을 아래와 같이 수정하면 통신이 가능하다. ROS\_IP에 해당 컴퓨터의 IP를 넣으면 된다. 또한 Ubuntu의 방화벽도 수정한다. 명령어 : sudo ufw allow from *컴퓨터\_IP*, sudo ufw allow from *시뮬레이터\_IP*)

|  |  |
| --- | --- |
|  |  |
| Docker/script/dev\_start.sh | Ubuntu 방화벽 상태 확인 |

1. ROS and Protobuf 연동

* Apollo-3.0은 모델에 입력하는 데이터를 위해 Google에서 만든 포맷인 Protobuf를 사용한다.
* 따라서 ROS bridge를 통해 시뮬레이터 데이터를 받아온 뒤, ROS to Protobuf를 직접적으로 변환하는 것이 없어 Dictionary를 이용하여 ROS to Dictionary & Dictionary toProtobuf로 변경하여 데이터를 입력한다.

1. Simulator 연동

* Protobuf의 형태와 동일한 ROS Message를 생성한 뒤, matlab에서 rosgenmsg(*folderpath*)의 명령어를 통해 생성하며 된다. (해당 참조 링크 : <https://kr.mathworks.com/help/ros/ref/rosgenmsg.html;jsessionid=195bc154f5849ab29cfdbf84c4e7>)
* 생성된 ROS message를 기반으로 송수신 해야하는 데이터를 연결하여 통신을 시작한다.

1. 실행 순서
   1. Apollo 실행
      1. Apollo 폴더로 이동.
      2. ./docker/script/dev\_start.sh 실행
      3. ./docker/script/dev\_into.sh 실행하여 Apollo docker로 이동

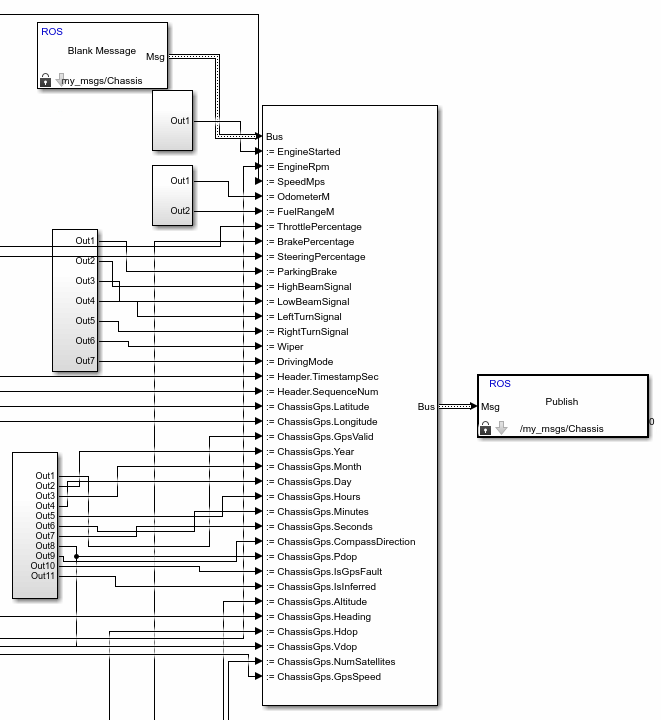
|  |
| --- |
|  |
| Dreamview 실행 및 차량, Map 선택 시킨 모습 |

* + 1. ./script/bootstrap.sh를 통하여 Dreamview(Apollo visualizer) 실행 및 크롬창에서<http://localhost:8888> 주소로 접속하면 Apollo Dreamview 동작 가능.(왼쪽 위에 선택을 통해 차량과 지도 선택)
    2. ./script/rosbridge.sh를 통해 ROS Bridge를 실행시킨다.
  1. ROS – Protobuf 통신 열기
     1. 압축되어있는 파일(apollo\_backup\_1217.zip)을 풀어 기존 apollo 폴더에 덮어씌운다.
     2. 추가적인 터미널을 열어 시뮬레이터에서 받은 ROS 데이터를 Protobuf 데이터로 변환하는 파이썬 파일을 실행한다.(Ctrl + Shift + T 누르면 터미널창에서 새로운 창이 생성)
     3. Docker 안에서 생성되어야하므로 명령어(./docker/script/dev\_into.sh)를 통해 Apollo Docker 내부로 진입한다.
     4. ros\_pkg 폴더로 이동한 뒤, my\_msgs(ROS 데이터를 Protobuf로 변환시키기 위한 중간 매개체)의 설치를 위해 명령어(./setup\_my\_msgs.sh)를 입력한다.
     5. 생성한 ROS 메시지인 my\_msgs를 적용하기위해 source devel/setup.bash를 명령어창 입력하여 적용한다.
     6. cd .. 명령어를 통하여 ros\_pkg 폴더를 벗어난다.
     7. 파이썬 파일 start\_message\_to\_pb.py를 실행시켜 통신할 데이터를 선택한다.

|  |
| --- |
|  |
| start\_message\_to\_pb.py 실행 모습 |

* + 1. 파이썬 파일 부분에서 Control 데이터 생성에 필요한 chassis(0번), pose(6번), obstacles(11번), routing\_request(17번)을 실행시키고 Control 데이터 및 미래Trajectory를 시뮬레이터에 전송하기위해 control(1번), planning(13번)을 실행시킨다.
  1. 시뮬레이터 통신
     1. Prescan에서 ROS 관련 message를 만든다.

Chassis



* + 1. 모든 데이터 연결 후 실행(재생)시키면 연동은 완료된다.