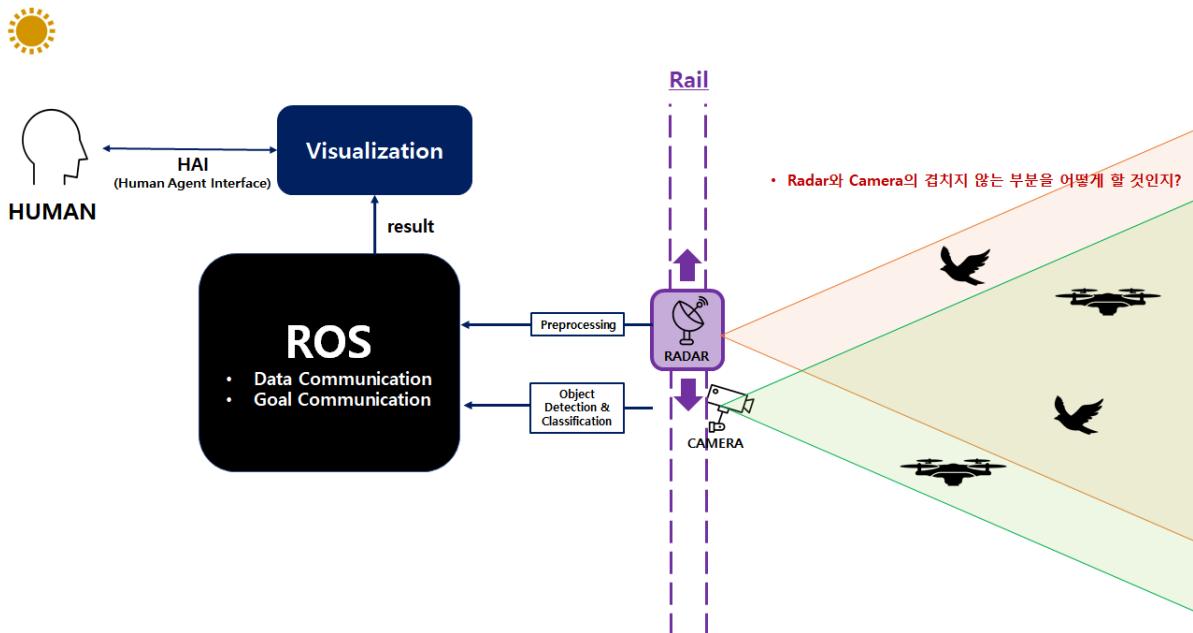


# ROS Design

Created By	inbae Kang
Last Edited	@Dec 04, 2019 1:33 PM
Property	
Tags	Document ROS

## 시나리오



Rail에 고정된 RADAR는 양 끝을 움직이면서 SAR 이미지를 만들어낸다.

동시에 CAMERA는 사진을 찍는다.

두 가지 이미지는 각각 학습된 모델을 통해서 Object Detection한다.

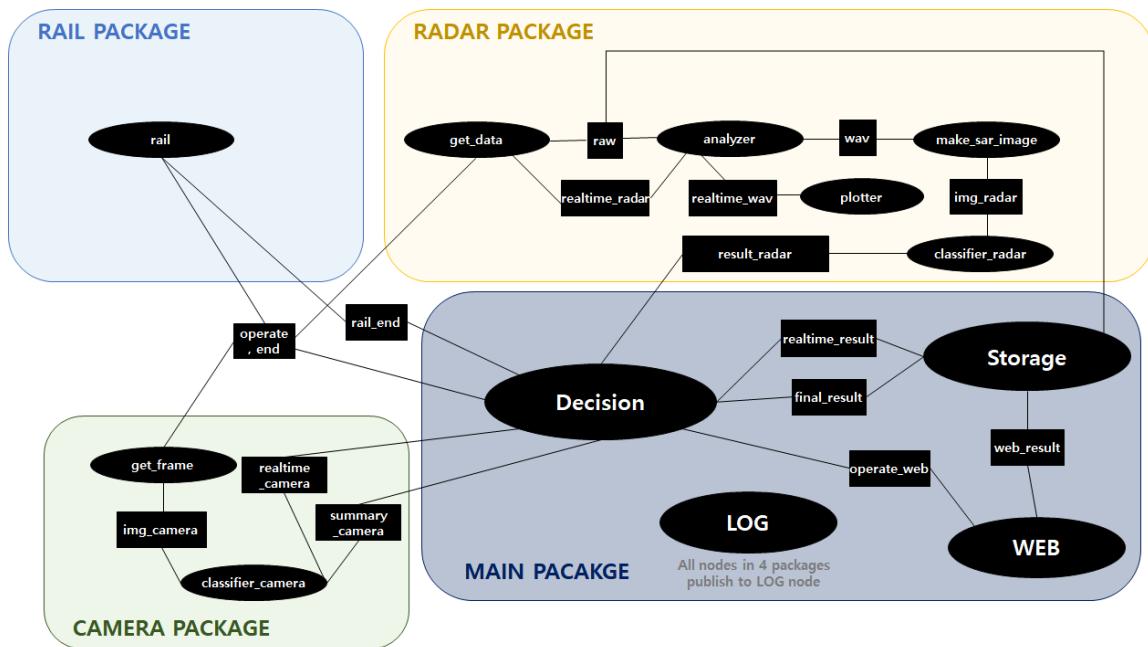
두 가지 정보를 적절하게 활용하여 UAV, 자동차, 사람을 판별한다.

도출된 결과는 확인할 수 있도록 웹 서비스로 제공한다.

이 과정 중에 모든 데이터 교환은 ROS의 Node 및 Topic으로 정의되어 진행된다.

## ROS 구성

- ROS 구성도 (노드 구성도)



- ROS Node (레일 노드, 레이더 노드, 등) 정의 (생성 시기 / 생성 조건 / 토픽 / 입력 출력 등)
  - 각 노드의 생성 시기 : 기본적으로 모든 노드를 Static하게 생성하고 시작.
  - 생성 조건 : 없음
  - 토픽 # 메시지 타입
    - operate** : 초기화, 시작을 알리는 토픽 **# main.msg/operate**
      - `init` : RAIL 노드의 초기 시작 위치 조정. 카메라와 레이더 노드도 필요한 초기화 작업이 필요하면 수행.
      - `start` : RAIL의 초기화 작업이 끝나고 `rail_end` 토픽으로부터 "`init_finish`" 메시지를 전달 받으면 한 사이클을 시작하기 위해서 각 노드에게 알림.
    - end** : RAIL이 끝에 도달하면 한 사이클을 끝내기 위해서 각 노드에게 알림. **# main.msg/operate**
    - rail\_end** : rail의 초기화 및 cycle 종료를 알리기 위한 토픽. **# std\_msgs/String**

- **init\_finish** : rail이 시작하기 전 상태로 초기화가 완료되었다는 메시지
- **cycle\_finish** : rail이 끝에 도달하였다는 메시지
- **logs** : 각 패키지의 로그를 전달 받기 위함.
- **raw** : 얻은 raw data를 analyzer 노드가 IFFT 처리, binary text를 wav파일로 변환하도록
- **wav** : make\_sar\_image 노드가 변환된 wav파일로 SAR이미지를 만들도록
- **img\_radar** : classifier\_radar 노드가 SAR이미지를 통해서 분류하도록
- **img\_camera** : classifier\_camera 노드가 분류하도록
- **result\_camera, result\_radar** : Decision노드가 RADAR와 CAMERA를 통해서 받은 데이터를 바탕으로 결과를 도출할 수 있도록
- **realtime\_result** : 실시간 데이터를 전달하기 위한 토픽
- **final\_result** : 레이더와 카메라에서 받은 최종 정보를 전달하기 위한 토픽
- ROS Message Type 정의 (노드에서 사용되는 메시지 정의)

#### msg type에 명시된 primitive로 더 구체적으로 구현 예정

- radar/railstop
- radar/railstart
- radar/raw
- radar/wav
- radar/img
- radar/result
- camera/img
- camera/result
- main/operate
- main/result
- main/result\_web
- main/realtme

main

## Document

### … Message Definition

## 리허설 (처음 노드 구성부터 결과값을 얻기 까지의 진행을 기술)

1. 메인 패키지에서 ROSCORE가 실행됨. 나머지 레일/레이더 노드와 카메라 노드에서 실행 환경을 갖추기 위한 Shell Script를 실행한다. 메인 라즈베리파이는 localhost로 카메라와 레이더 라즈베리파이는 메인 라즈베리파이/PC의 ip주소로 ROS\_MASTER\_URI를 설정하여 ROS상에서 통신할 수 있도록 설정.
2. 카메라, 레이더 라즈베리 파이에서도 각 노드가 실행되고(<package\_name>.sh이름의 shell script로 실행), main package에서는 순서대로 log, storage, web node 가 실행되고, 마지막으로 decision노드가 실행되고, init 메시지를 operate topic에 보낸다.
3. rail 및 camera 패키지는 각각의 초기화를 마치고, rail node는 rail의 초기화가 끝났다는 메시지를 rail\_end 토픽에 메시지를 보낸다. 메시지를 받은 decision노드는 init phase에서 start phase로 넘어간다.
4. Decision노드가 operate 토픽에 start 메시지를 보낸다.
5. 메시지를 받은 세 노드는
  1. 레일노드는 레일을 74초에 걸쳐 움직이도록 한다.
  2. 레이더노드는 binary 데이터 수집을 시작한다.
  3. Camera 노드는 데이터 수집을 시작. 데이터를 수집하면서 realtime\_camera 토픽을 통해서 Decision노드는 실시간 데이터를 받아서 storage노드에 저장하고 web노드에게 해당 파일의 경로 및 결과 값을 result\_web 토픽에 전달하여 보여 줄 수 있도록 한다.
6. 레일이 끝에 도달하면, rail\_end 토픽에 한 주기가 끝났다고 message를 publish하고 메시지를 받은 decision노드는 end phase로 넘어간다.
7. 해당 토픽을 subscribe하고 있는 decision node는 end 토픽에 한 주기가 끝났다는 메시지를 보내어 카메라 및 레이더가 데이터 수집을 중단하도록 알린다.
8. radar 패키지에서는 지금까지 쌓인 binary data를 raw 토픽에 publish함. storage 노드는 해당 raw파일을 저장한다.

9. IFFT 및 wav 데이터로 변환 후에 make\_sar\_image 노드가 받을 수 있도록 publish 함.
10. SAR Image를 만들고 Decision노드에게 전달
11. Camera 패키지에서는 지금까지의 사진들의 summary를 만들어서 summary\_camera에 메시지를 보낸다.
12. RADAR와 CAMERA는 각각 result\_radar, summary\_camera 토픽에 publish하여 Decision노드가 받아서 storage노드에게 전달하고, storage노드에서는 파일로 저장한 후에 경로를 web노드에게 전달하여 보여줄 수 있도록 한다.

## English Version

1. ROSCORE is run on the main package. The rest of the rail / radar nodes and camera nodes run a shell script to set up the execution environment. Main Raspberry Pi is localhost and Camera and Radar Raspberry Pi are configured to communicate on ROS by setting ROS\_MASTER\_URI to the IP address of main Raspberry Pi / PC.
2. Each node is executed in camera and radar Raspberry Pi (by shell script named <package\_name>.sh), log, storage, web nodes are executed in order in main package, finally decision node is executed. Send an init message to the operate topic.
3. The rail and camera packages complete their initialization, and the rail node sends a message to the rail\_end topic indicating that the rail has finished initializing. The decision node receiving the message goes from the init phase to the start phase.
4. The Decision node sends a start message to the operate topic.
5. The three nodes that received the message
  1. The rail node moves the rail over 74 seconds.
  2. The radar node starts collecting binary data.
  3. The Camera node starts collecting data. While collecting data, the Decision node receives real-time data through the realtime\_camera topic, stores it in the storage node, and passes the file path and result value to the web node to show the result\_web topic.

6. When the rail reaches the end, it posts a message to the rail\_end topic that the cycle is over and the decision node that receives the message goes to the end phase.
7. The decision node subscribed to that topic sends a message to the end topic telling the camera and radar to stop collecting data.
8. The radar package publishes the binary data accumulated so far to the raw topic. The storage node stores the raw file.
9. Publish for make\_sar\_image node to receive after conversion to IFFT and wav data.
10. Create a SAR image and pass it to the decision node
11. The Camera package creates a summary of the photos so far and sends a message to the summary\_camera.
12. RADAR and CAMERA publish to the result\_radar and summary\_camera topics, respectively, so that the Decision node receives it and delivers it to the storage node.

## 실제 환경 구성

### 하드웨어 구성

- 카메라 라즈베리파이
- 레이더/레일 라즈베리파이
- 메인 라즈베리파이

### 네트워크

- **ROS**에서 여러 기기 연결하기 링크 참조.
- 각각의 실행하기 전 환경을 shell script로 만들어 두고, ssh를 통해서 실행하는 것으로 계획.
- 하나의 공유기를 통해서 로컬 네트워크로 내부 고정아이피를 통해서 통신.

