

<전체 시스템이 작동하기위해 해야할 것>

<BLDC 모터>

1.수동 조작이 블루투스를 통해 4개의 모터를 매끄럽게 제어를 한다. => 어떤 상황이 와도 정지가 최우선적으로 작동해야한다.

2. 4개의 모터의 encoder 값을 맞추고, st를 통해 특정 목적지까지 갈때 각 바퀴에 어느정도의 encoder값이 필요한지 계산해서 그 값을 주었을때 오차가 어느 정도 나는지 여러 경우의 수를 두어서 실험해본다. (실내와 실외에서 둘다 해본다. 슬립이 일어날 수 있게 카페트 깔아서도 실험해보는 등)

=>비상 정지를 앱을 통해 누르고 다시 작동할때 남은 encoder값을 다시 실행할때 오차도 실험

=> 비상 정지는 수동제어모드일때든 자동제어 모드일때 어느 경우에서나 작동이 되어야한다.

=> **그러면** RTOS에서도 비상 정지 값을 담은 변수(전역 변수)가 있다면 그 값을 반영해서 encoder 값이 이루어져야한다.

=> 해보고 이게 오차가 너무 많이 나고 이것을 쓰기에 적합하지 않다고 판단이 들면 바로 localization 하는 성진 이한테 최대한 빨리 알려줘야 localization 할때 모터 encoder를 제외하고 할 수있다.

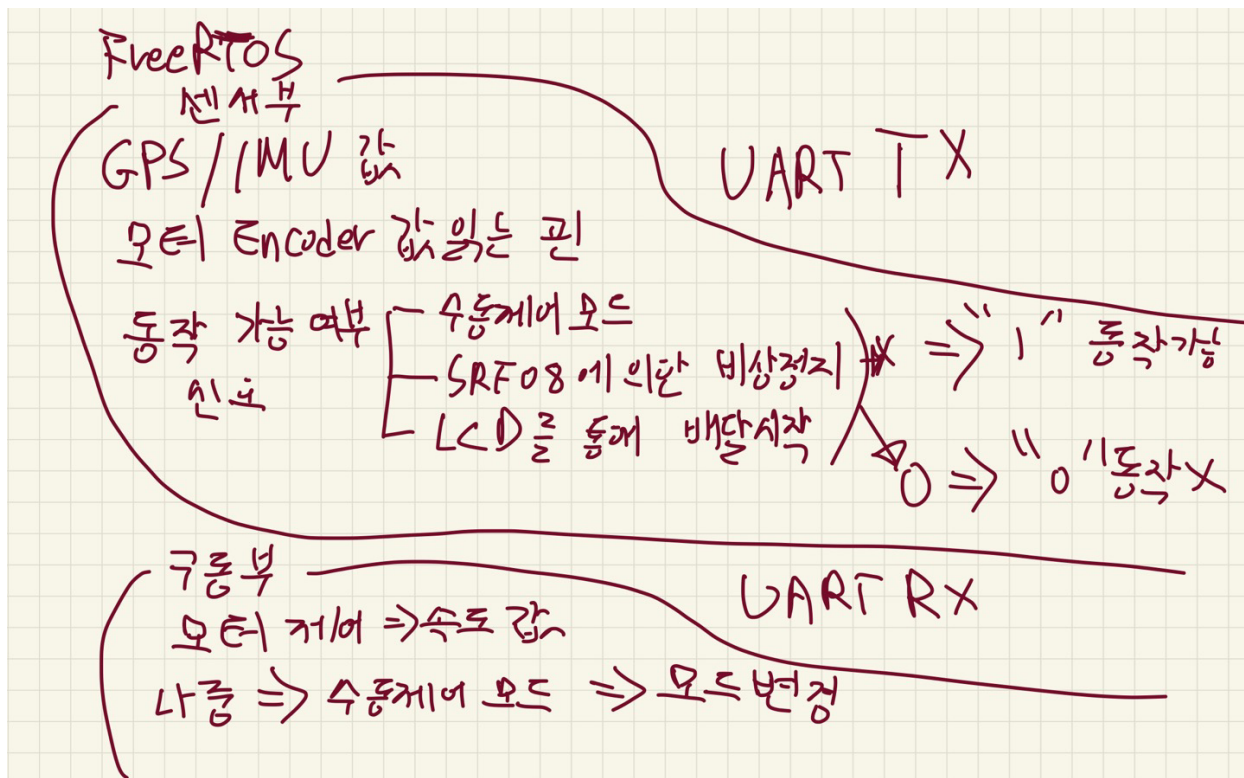
3. srf08에 의해 장애물이 감지되었을때 모터를 비상정지하는 것을 짠다. => 모터 encoder 가 쓸만하다고 위에서 판단되었을 경우 srf08에 의해 멈췄다 다시 남은 encoder 값만큼 다시 움직였을경우 오차가 어느정도되는지 계산 해본다.

=> srf08에 의해 비상 정지를 할때 너무 덜컥하고 멈춰서 음식들이 엎어질 수 있는 경우의 수가 있다 그러니 감지된 거리에 따라 모터의 정지를 달리할 필요가 있다.

➔ 이 일련의 과정은 코드를 짤때 순차적으로 만든 코드에 추가추가해서 이루어져야한다. 따로따로 만들어서 합치는게 아니라 블루투스로 자동제어하는 코드를 만들었으면 그 코드에 추가해서 encoder 값 주었을때 그 위치만큼 가도록 추가 코딩을 하는 식으로 살을 붙여나가야 함

➔그다음 LCD의 기능을 넣어서 코딩 (=> 밑에 전체 구동 메커니즘에 따라)

ST에서의 Free RTOS



UART Tx 부분은 일정한 주기로 실시간으로 계속해서 jetson에게 값을 보내주고있어야한다.

➔ SRF08은 지금 HAL_Delay를 써서 코드가 구현되어있는데 그 부분 Timer로 바뀌서 구현해야한다

=>HAL_Delay는 시스템 정지를 일으키므로 나머지가 동작이 안된다.

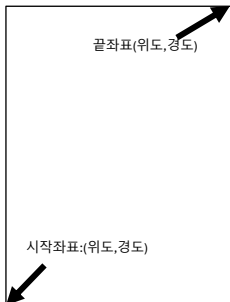
<Localization>

Localization은 jetson을 키는 순간 실시간으로 계속 이루어지고 있어야한다.

=> tmap으로부터 global plan을 받지 않았더라도 실시간으로 localization이 되어야 한다.

1. 우리 학교 면적(7만6천 m^2)의 위도와 경도 좌표를 google 지도를 통해서나 Tmap을 사용하니깐 TMap을 사용해서 시작좌표(위도,경도)~~ 끝 좌표(위도, 경도)를 구하고 그것의 좌표계를 빈 pgm파일에 적용한다.

ex)



캠퍼스 부지 면적은 76,485m²로 옆 학교처럼 캠퍼스가 참으로 아름답다.
또한 옆 학교처럼 건물이 동떨어져있다거나, 경사가 가파르지 않다.

namu.wiki > 인덕대학교
[인덕대학교 - 나무위키](#)

정보오류 수정요청 · 지식스니펫 ⓘ

2. GPS 값을 받아서 만든 빈 pgm 파일에 찍어서 rviz 상에서 찍은 점은 TF만을 표시해서 확인하고, 잘 찍혔는지 토픽으로 그 점을 클릭했을때 subscribe해서 좌표 값을 확인한다.

3.그다음 그 점에 imu를 적용했을때 TF의 방향이 잘 반영되는지 확인한다.

4. EKF를 통해 GNSS와 IMU 그리고 모터 encoder 값을 묶어주어서 localization 잘되는지 확인해준다.

=> 그 전에 BLDC모터의 encoder 값이 사용할만 한지부터 판단이 선행되어서 사용하기엔 너무 부정확하면 제외

통신 - 1. RealtimeDB에 우분투 launch 파일을 통해 값을 넣고 빼는거가 되어야 한다.

2. TMAP이 생성한 경로 좌표를 RealtimeDB를 통해 넣었을때 그것을 우분투에서 값을 빼내서 배열에 담아줘야한다.

3. 빈 pgm 파일에 그 좌표 값을 찍어본다.

4. TMAP에 현재 로봇의 좌표를 얻어와서 출발지로 설정하고 목적지는 카페와 고객이 고른 장소로 설정해서 경로가 생성될 수 있도록해야한다. => 이게 3번 보다 우선이 되어야할지도?

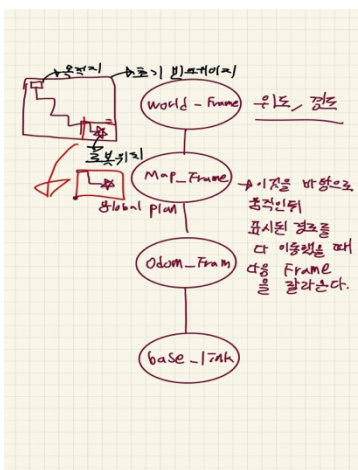
5.주문이 들어왔을때 현재위치에서 카페로 가는 경로 한개 그리고 카페에서 고객에게 가는 경로 한개 그리고 고객에서 원래자리로 돌아가는 경로 1개 총 세개의 경로를 jetson에게 제공해야하며 각각을 주는 시점은 주문이 들어왔을때 현재위치에서 카페로 가는 경로가 실행되며(관리자앱에서 실행을 누르기때문에 LCD 버튼 누를 필요x) 카페에서 배송 시작을 LCD에서 눌렀을때 카페에서 고객에게 가는 경로가 주어져야한다. 마지막으로 고객이 수령 완료 버튼을 눌렀을때 고객에서 원래자리로 돌아가는 경로를 제공해 줄 수 있게 해줘야한다.

<Planner server>

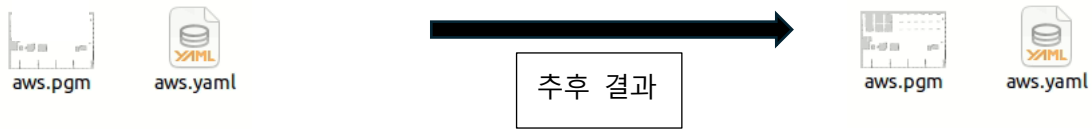
1. TMAP을 통해 얻어온 전체 경로를 어떤 방식으로 잘라서 pgm 파일로 재 설정해줄 것인지 구상

2. planner plugin => 1.configure 함수 - RealtimeDB를 통해 얻어온 전체 경로를 찍어준다.

=> 2. Activate 함수 - 전체 경로를 찍고 이어준 pgm 파일로부터 어느 범위로 어떻게 pgm 파일을 잘라낼 것인지 코드를 작성한다.



3. controller server에서 obstacle layer에서 반영된 벽에 대한 정보를 pgm파일로 저장할 수 있도록 해야한다.



이렇게 작업을 통해 map이 생성되면 나중에는 TMAP을 통해 값을 받지 않고 출발지와 목적지에 대한 좌표 값만 받아서 ROS2 경로 안내를 할 수 있지 않을까 싶다.

⇒ 지금 TMAP으로 해주는 이유는 우리의 작업량이 줄어들고, 미지의 거리에서도 노가다를 많이 안해도되기 때문

4. map_saver에 의해 3번 맵 전체만 저장되어서 map_server에 의해 제공이 되고, 맵 pgm 파일을 자른 것은 planner server에서 이루어지도록 설계함

=> 추후 변경 되면 다시 알려드릴

Ex)



<map_server에 의해 제공되는 것>

<planner server에서 하는 일>

Sensor Fusion은 추후에 파라미터 설정을 해야하니 실험을 통해 어느 정도 범위까지가 센서의 오차가 많이 나지 않아 활용가능한지 알려줘야한다. => 그 거리(m)값에 따라서 local plan 범위를 설정할 수 있다.

⇒ 토픽 구독을 통해 costmap에 반영할 수 있을 것 같다.

노드

/scan => 라이다와 카메라 센서 퓨전을 통한 전방 55도와 사이드 나머지 각도 => 토픽

/imu,/gnss/encoder => ST에 의해 제공 받은 imu와 GPS와 encoder 값 => 토픽

Localization 값을 얻어오는 노드 => 토픽

Localization 노드 와 map server를 묶어서=>그룹 액션

Nav2 => 1. Controller server

2. smoother server

3. planner server => 그룹 액션

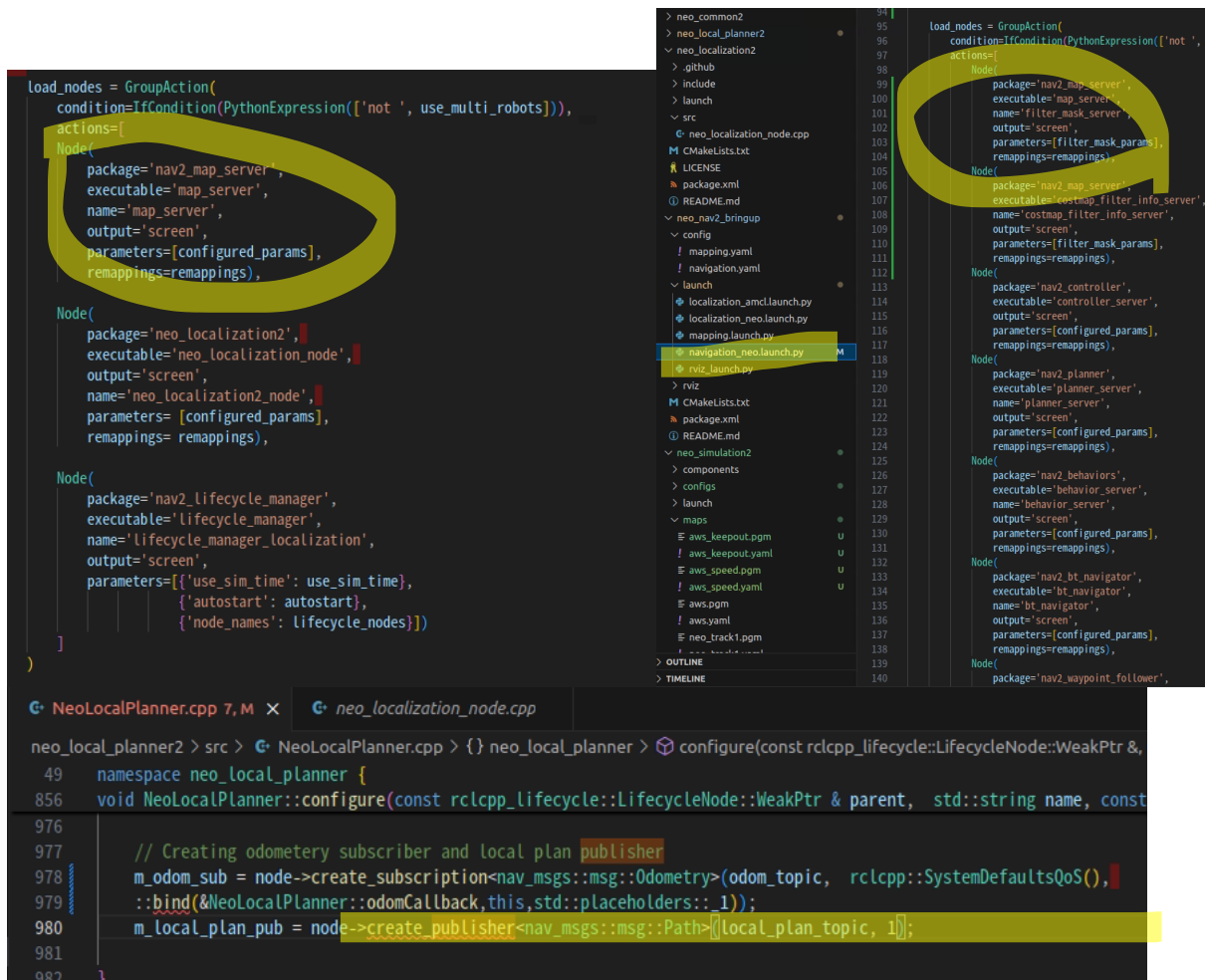
RealtimeDB => TMAP을 통해 전달 받은 전체 경로 =>서비스 3번 정도 주게끔 설정

-> 처음에 잘 안들어올수도 있으므로

➔ 토픽을 통해 노드의 파라미터 값들을 publish 해놓으면 여러개의 필요한 노드에서 subscribe를 통해 값을 얻을 수 있다.

➔ 지금까지 공부해본 바로는 localization의 노드에서 토픽으로 계속해서 내보내고 있고, 동시에 map_server와 localization 노드를 그룹 액션으로 묶어주어서 MAP 상에서 localization 값을 찍어준다.,

➔한개의 노드에서 토픽과 액션 둘다 가능하다.



```
load_nodes = GroupAction(  
    condition=IfCondition(PythonExpression(['not ', use_multi_robots])),  
    actions=[  
        Node(  
            package='nav2_map_server',  
            executable='map_server',  
            name='map_server',  
            output='screen',  
            parameters=[configured_params],  
            remappings=remappings),  
        Node(  
            package='neo_localization2',  
            executable='neo_localization_node',  
            output='screen',  
            name='neo_localization2_node',  
            parameters=[configured_params],  
            remappings=remappings),  
        Node(  
            package='nav2_lifecycle_manager',  
            executable='lifecycle_manager',  
            name='lifecycle_manager_localization',  
            output='screen',  
            parameters=[{'use_sim_time': use_sim_time},  
                        {'autostart': autostart},  
                        {'node_names': lifecycle_nodes}])  
    ]  
)
```

```
neo_local_planner2 > src > NeoLocalPlanner.cpp > { } neo_local_planner > configure(const rclcpp_lifecycle::LifecycleNode::WeakPtr &  
49 namespace neo_local_planner {  
856 void NeoLocalPlanner::configure(const rclcpp_lifecycle::LifecycleNode::WeakPtr & parent, std::string name, const  
976  
977 // Creating odometry subscriber and local plan publisher  
978 m_odom_sub = node->create_subscription<nav_msgs::msg::Odometry>(odom_topic, rclcpp::SystemDefaultsQoS(),  
979 ::bind(&NeoLocalPlanner::odomCallback, this, std::placeholders::_1));  
980 m_local_plan_pub = node->create_publisher<nav_msgs::msg::Path>(local_plan_topic, 1);  
981  
982 }
```

<토픽의 장점>

• 지속적인 정보 제공: 위치 추정은 로봇의 움직임에 따라 계속해서 업데이트되는 정보입니다.

토픽은 이러한 지속적인 데이터 흐름을 처리하는 데 적합

• 다중 구독: 여러 노드가 동시에 위치 추정 정보를 필요한 경우가 있을 수 있다..

토픽은 여러 구독자(subscriber)에게 동시에 정보를 전달할 수 있어 효율적

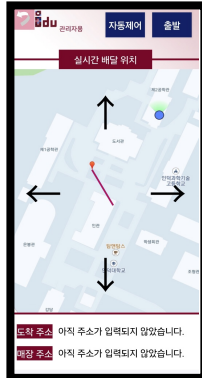
➔ 모든 노드들은 토픽을 기본적으로 하면서 nav2를 할때 필요에 따라 그룹 액션으로 묶어주는 것 같다.

사실상 Thread를 어떻게 묶고 나눠주냐가 더 중요할거 같다.

전제조건: Jetson을 통해 로봇의 현재위치를
지속적으로 입력 받고(localization한 값) 있어야한다.

⇒ 시작 위치를 정해놓아도됨

1. RealtimeDB로부터 주문이 들어옴



2. Jetson 현재 위치(출발지)로부터 카페(도착지)까지의
도보 경로를 TMAP을 통해 global plan을 생성 후
RealtimeDB에 등록

3. Jetson이 RealtimeDB로부터 global plan 값을 얻어온다.

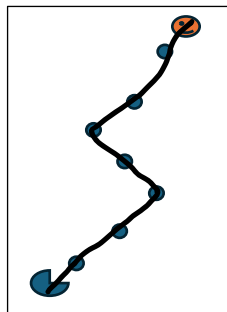
4. ST에 전달

5. LCD에 운행 시작 버튼 활성화

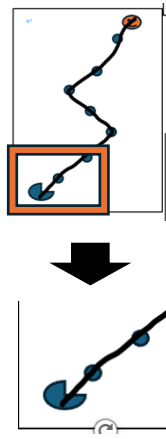
6. LCD 버튼 클릭 시 - 수동 제어 모드X
- srf08에 의한 비상정지 모드X
두가지 조건이 만족 되면 동작 가능 여부 “1” 세팅
=>LCD 화면은 “운행 중” 으로 세팅

7. Jetson에 전달

4. lifecycle manager에게 planner server
와 controller server의 configure 후에
activate 요청



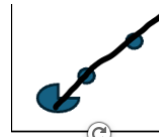
5.우리 학교가 차지하는 면적만큼의 위도와
경로를 반영한 좌표계 적용한 빈 pgm 파일을
map server로부터 얻어온다. Real time DB에서
얻어온 global plan좌표를 빈 pgm 파일에 찍
고, 선으로 연결
=> planner plugin의 configure에서 수행



6. 로봇의 localization된(현재 로봇의 위치) 좌표와 가야하는 경로를 기준으로 일정한 면적만큼 잘라서 다른 pgm 파일에 넣은 뒤 그것을 local plan에서의 실제 global plan으로 설정한다
=> planner plugin의 activate에서 수행

7. lifecycle manager를 통해 controller server의 상태를 configure 후에 activate시켜준다.

8. Controller server에 plugin된 Follow path에 의해 로봇에게 속도 값을 제공하며 주행하고, 실시간으로 장애물 감지 시 costmap에 반영해준다. => Follow path의 activate에서 수행



9. 자른 pgm 파일의 경로를 다 따라가면 또 다시 갱신된 로봇의 localization된 좌표와 가야하는 경로를 기준으로 잘라와서 다른 pgm 파일에 넣은 뒤 그것을 local plan에서의 실제 global plan으로 설정한다.
=> 8번 행동 실행
=> planner plugin의 activate에서 수행



10. 도착지에 도착시 ST에게 신호를 준다.

11. nav2 종료

전제 조건: Srf08을 통해 장애물 감지x

8. Jetson에게 전달 받은 속도 값에 따라 모터 구동

10. 모터 작동이 멈추며, LCD에는 “배송 시작” 또는 “수령 버튼”을 누르는 화면이 출력된다.

11. 매장 직원이 음료수를 넣은뒤 LCD의 “배송 버튼”을 누르면 1번부터 재반복 목적지만 고객 위치로 바뀐다.