

---

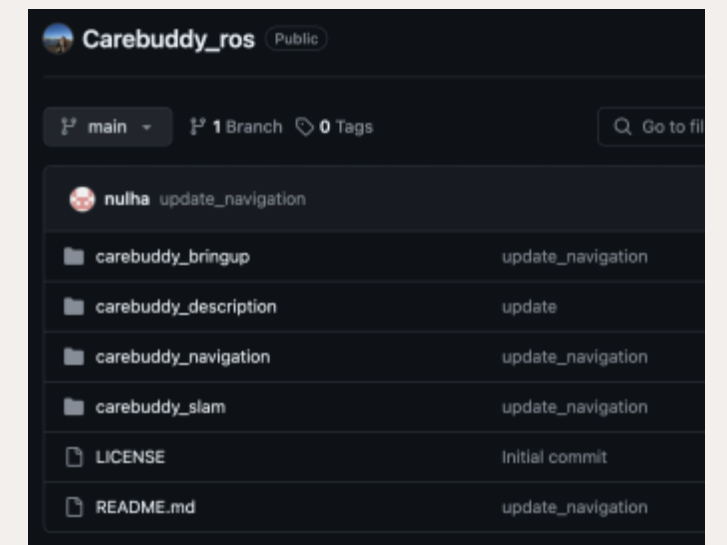
# 캡스톤디자인(2) 진행 상황

MOTUS+ER

---

# 01 간략한 진행 상황

모터 제어  
문제



## 문제 : 제어의 문제

- 기본 TT 모터로 PWM 제어 시도
- cmd\_vel 값을 받아서 동작하지만, 엔코더 센서가 없어서 세밀한 조정이 힘들

## 문제 : 모터 수량 문제

- 엔코더 모터 수량이 얼마 없어서 5/20~5/31 사이에 모터가 도착 예정

## 시도 : 로봇 구매

- 엔코더 모터가 탑재된 로봇을 추가로 구매함
- omo r1 mini

## 추가 계획

- 졸업 작품 : omo 로봇으로 구현
- 이후 엔코더 모터가 배송오면
  - PID 설계
  - CareBuddy ros 패키지 빌드
  - 음성인식 구현

## 02 Segmentation 및 UI

관련된 모든 파일들  
(launch, msgs, cpp)을 수정



msg를 통해서 기존의 액션 서버와 새로 만든 "클라이언트 코드"가 통신을 하게 되는데, optional 변수를 설정해도 완전한 값을 가져오지 못함

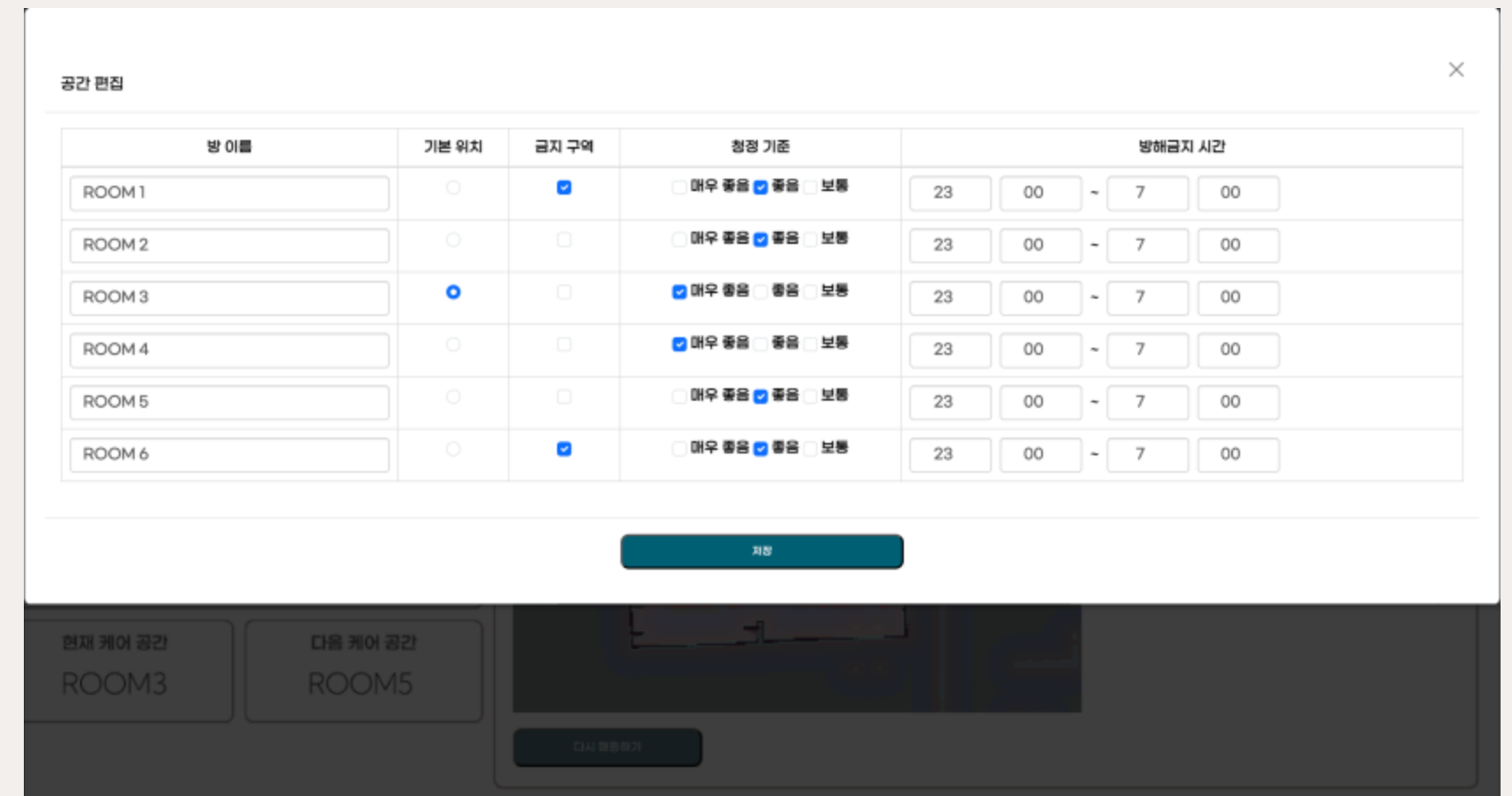


min, max 기준의 방 크기를 limit으로 "클라이언트 코드"에 두는 것이 시도한 것 중 가장 결과가 좋았음



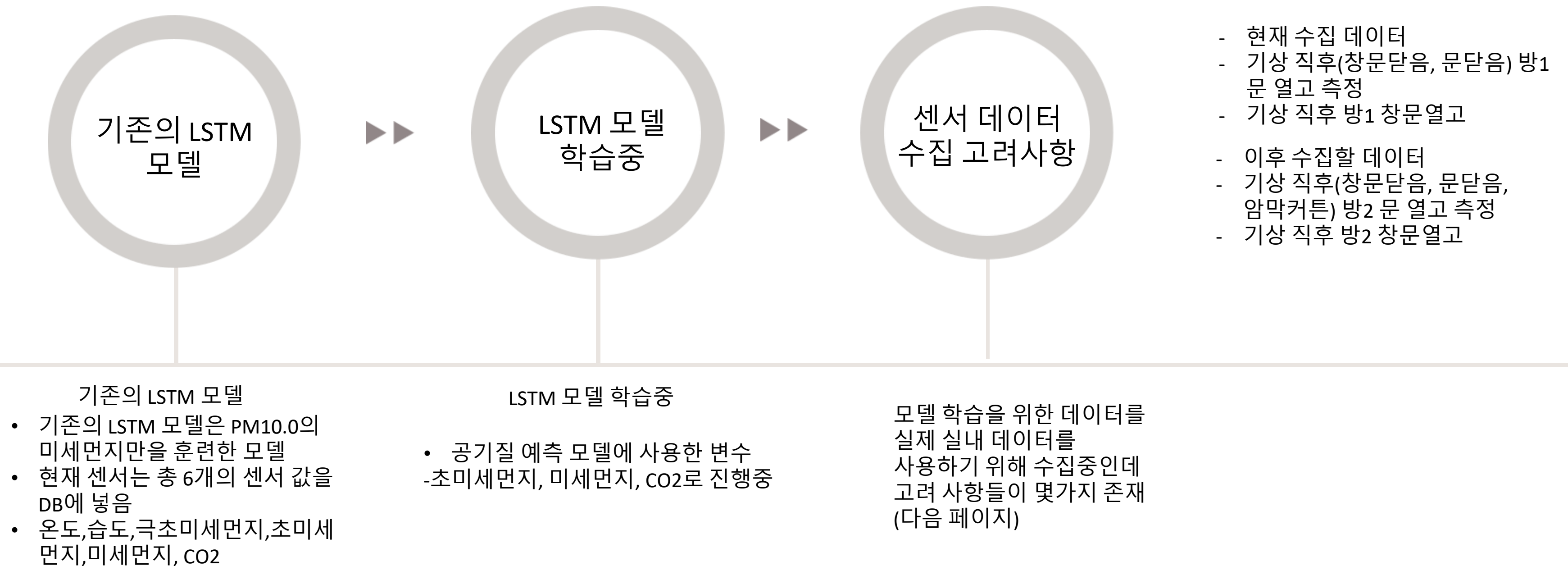
분할된 공간 선택

UI에서 사용자가 사용할 공간만 선택하도록 할 예정



공간 편집

## 03 LSTM 모델 작업



기상 직후의 방1

```
Temp: 80.6 F / 27.0 C    Humidity: 44%
PMS 7003 dust data
PM 1.0 : 6
PM 2.5 : 16
PM 10.0 : 17
{'co2': 1732}
```

30분후 방1 (문만 열음)

```
Temp: 78.8 F / 26.0 C    Humidity: 38%
PMS 7003 dust data
PM 1.0 : 5
PM 2.5 : 8
PM 10.0 : 8
{'co2': 846}
```

기상 후 방문을 열고서 의 방1  
CO2 수치가 지속 적으로 줄어드는것을 확인가능

```
final_result
27
44
7
17
17
1526
final_result
27
43
6
16
16
1523
final_result
27
43
7
16
17
1522
```

측정 1시간후 방1 (문만 열음),

27	40	5	14	17	718
27	40	5	12	16	716
27	40	5	12	15	715
27	40	7	12	16	714
27	40	5	11	16	713
27	40	6	11	18	714
27	40	5	11	18	713
27	40	5	12	17	613
26	38	7	13	18	614
26	38	7	13	19	631
26	38	5	12	17	650
26	38	5	11	16	687
26	38	5	10	15	727
26	38	5	10	15	745
26	38	4	9	14	752
26	38	4	9	14	758
26	38	5	11	15	765
26	38	5	13	16	768

그래서 교수님 의견을  
구하고 싶은 부분이

창문의 열림, 문의  
열림을 즉 환기등을  
고려하지 않고 그냥  
계속해서 데이터를 모을  
것인가

아니면 창문과 문을  
1시간 가량 열어놓고서  
최저치부터 시작해서  
이후 일정시간동안의  
변화를 이용해서 학습을  
할것인가

기상직후 창문과 문을  
열어놓고서 매우 좋음의  
수치까지 떨어지는데  
얼마나 걸릴것인가도  
넣을 것이냐?

# 04 Carebuddy\_ros 패키지 구현

Carebuddy\_ros 패키지를 직접 구현하였다. github에 올려두었고 git clone 하여 사용 가능한 상태이다. 각각의 패키지는 다음과 같다.

## Carebuddy\_bringup

- 로봇과 로봇과 연결된 센서들(라이다)을 활성화하며 ros와 연결하는 등의 역할이다.
- 라즈베리파이에서 roslaunch carebuddy\_bringup carebuddy\_robot.launch를 통해 로봇과 라이다의 ros 통신을 활성화한다.

## Carebuddy\_description

- 로봇의 물리적 구조와 모습을 정의한다.
- URDF 파일을 사용하여 로봇의 크기, 형태, 관절, 센서 위치 등을 설명한다.
- 시뮬레이션 및 시각화 도구에서 로봇 모델을 정확하게 재현하는 데 사용된다.

## Carebuddy\_slam

- mapping(gmapping을 사용)을 실하는 패키지이다.
- gmapping 패키지를 include하여 로봇의 현재 상태 정보와 라이다 센서에서 발행되는 tf 토픽을 통합하여 환경의 지도를 생성한다.
- 여기서 얻은 지도는 Carebuddy\_navigation에서 사용된다.

## Carebuddy\_navigation

- 지도를 로드하고, 라이다 센서를 통해 얻은 정보와 로봇의 상태 정보를 통해 현재 로봇의 위치를 지도상에서 파악한다.
- amcl(위치추정 알고리즘 패키지)를 include 하여 로봇의 정확한 위치를 지속적으로 추정할 수 있도록 한다.
- 목적지를 설정하면 move\_base\_goal 토픽이 발행되어 목적지로의 경로를 계획하고, 이동 중 장애물을 회피하는 등의 로컬 경로 계획 알고리즘을 사용하여 안전하게 목적지에 도달하도록 한다.



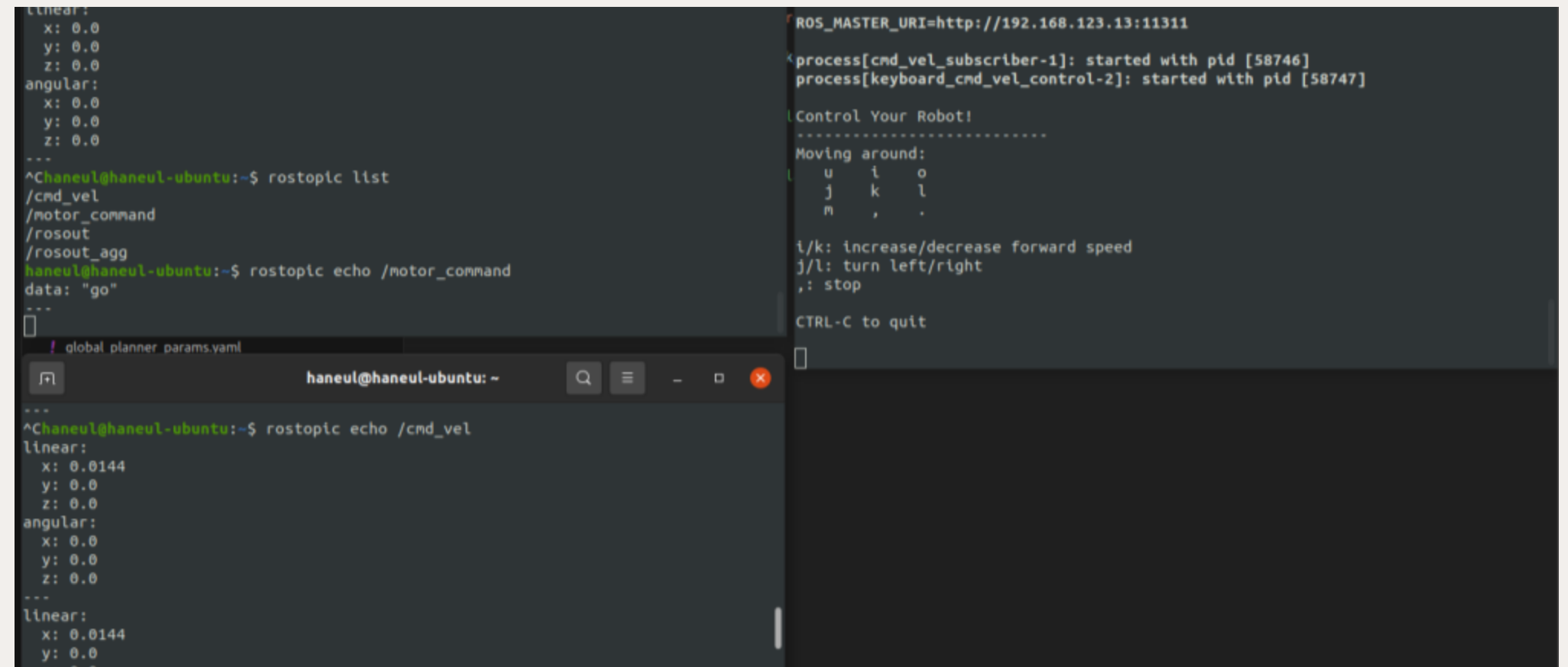
# 05 로봇 원격 구동

ros에서 navigation할 때 사용되는 cmd\_vel(로봇의 선속도, 각속도) 토픽 발행과 pwm 제어(현재 carebuddy의 모터 제어 방식)가 연결되도록 하였다.

cmd\_vel 에서 선속도는 0.0144, 각속도는 0.411로 고정하였다. (현재 carebuddy의 rpm과 바퀴간 간격을 고려하여 계산하였다. carebuddy의 속도가 변하는 일은 없다고 가정하였다.)

## <작동 순서>

navigation에서 목적지 명령(경로 계획 알고리즘 실행)  
→ cmd\_vel 값 발행 →  
motor\_cammand('go','stop','left','right','stop' 등)  
발행 → 로봇 구동



```
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
^Chaneul@haneul-ubuntu:~$ rostopic list
/cmd_vel
/motor_command
/rosout
/rosout_agg
haneul@haneul-ubuntu:~$ rostopic echo /motor_command
data: "go"
---
^Chaneul@haneul-ubuntu:~$ rostopic echo /cmd_vel
linear:
  x: 0.0144
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0144
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---

ROS_MASTER_URI=http://192.168.123.13:11311
process[cmd_vel_subscriber-1]: started with pid [58746]
process[keyboard_cmd_vel_control-2]: started with pid [58747]

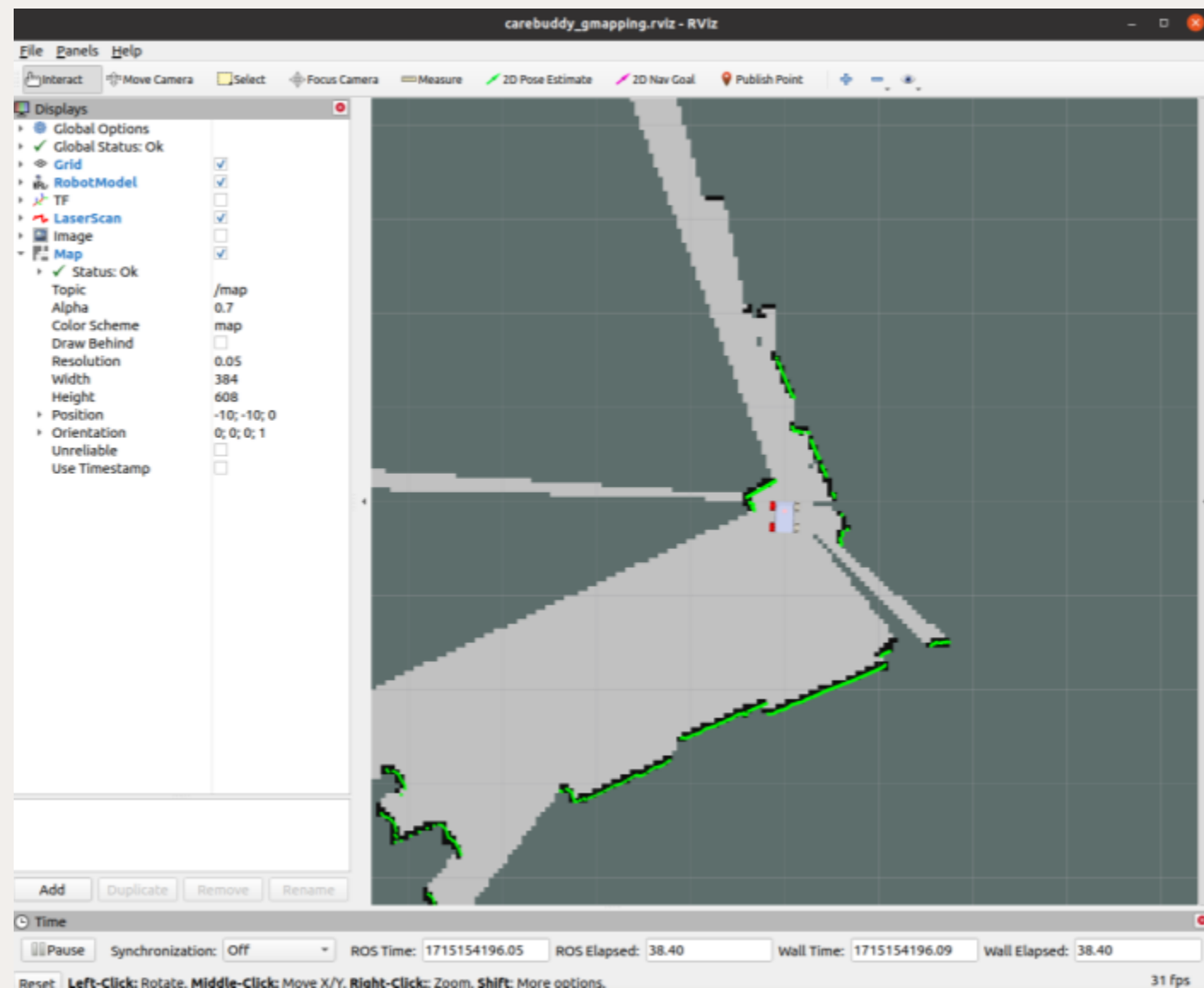
Control Your Robot!
-----
Moving around:
  u   i   o
  j   k   l
  m   ,   .

i/k: increase/decrease forward speed
j/l: turn left/right
,: stop
CTRL-C to quit
```

cmd\_vel 값이 변경되면 (왼쪽 아래) 그에 따라 motor에 명령을 주게 된다. (왼쪽 위)  
cmd\_vel 값은 계속 무한으로 발행되기 때문에 (0.1초에 한번 정도씩) 이전과의 값이 달라질 때만  
motor가 움직이도록 하였다.

# 06 Carebuddy\_slam

gmapping 패키지를 통한 mapping을 진행한다.



## 라즈베리파이에서

```
roslaunch carebuddy_bringup carebuddy_robot.launch (로봇과 라이다 활성화)
```

## 우분투에서

```
roslaunch carebuddy_slam carebuddy_slam.launch (로봇 모델과 gmapping 실행)
```

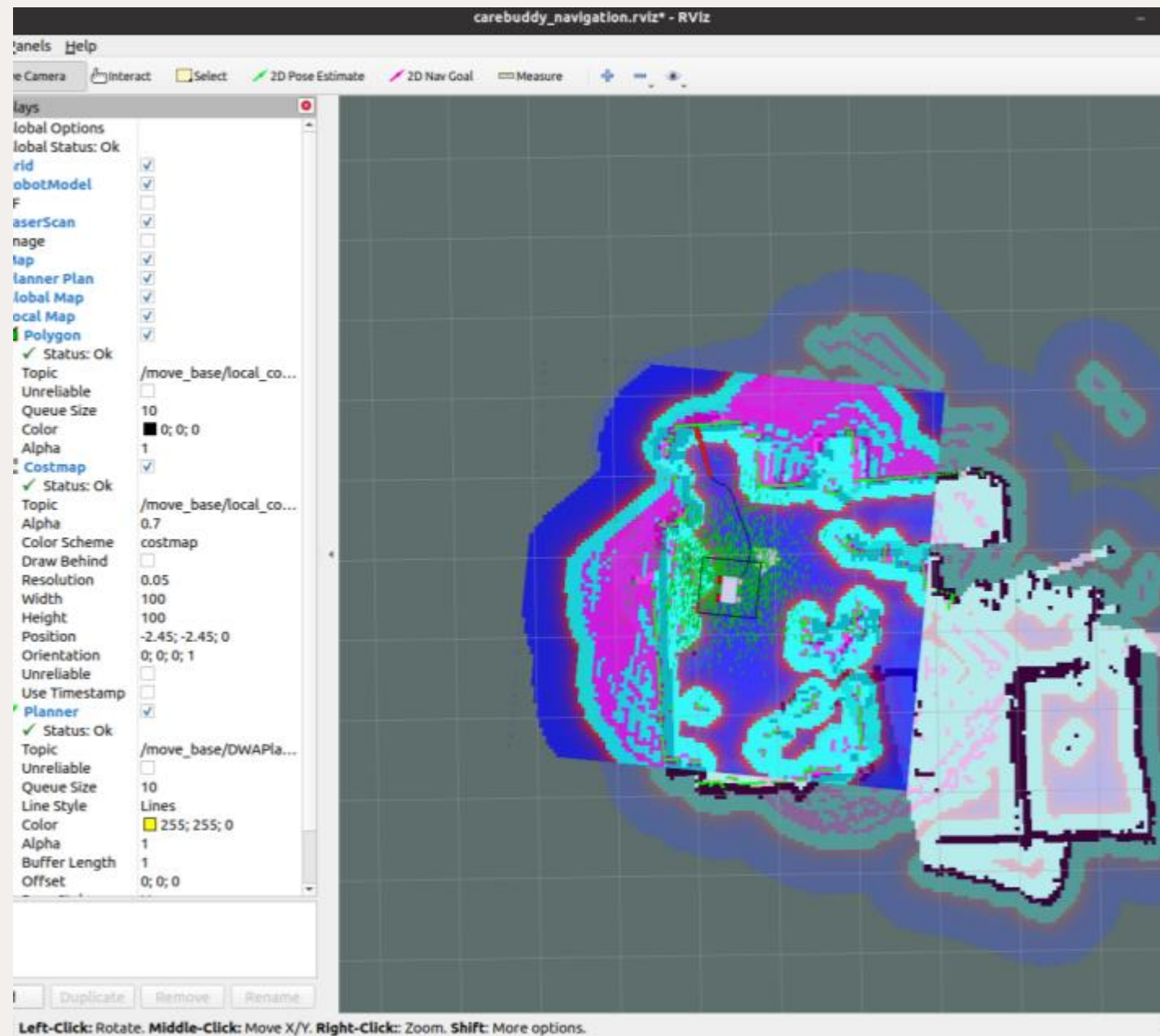
```
roslaunch carebuddy_slam carebuddy_slam_rviz.launch (rviz에 지도와 로봇 시각화)
```

현재는 mapping을 키보드로 하지만 turtlebot에서 시뮬레이션 해본 자동매핑 알고리즘을 통해 자동으로 mapping 할 예정이다.



# 07 Carebuddy\_navigation

amcl과 move\_base 패키지를 통해 navigation을 진행한다.



라즈베리파이에서

```
roslaunch carebuddy_bringup carebuddy_robot.launch (로봇과 라이다 활성화)
```

우분투에서

```
roslaunch carebuddy_navigation carebuddy_navigation.launch
```

```
roslaunch carebuddy_navigation carebuddy_navigation_rviz.launch
```

carebuddy는 속도가 항상 일정해야하기 때문에 속도(cmd\_vel)를 수정해야하고 , 참고했던 로봇 패키지와는 다른 부분이 많기 때문에 수정할 부분이 남아있다.

# 08 계획

5/13 - 5/19

- omo 로봇을 이용하여 자동 매핑
- omo 로봇에 맞게 웹 코드 수정

5/20 - 5/26

- 최종 UI 수정
- 전체 시스템 통합
- 졸업 작품 제출 영상용 PPT 제작 및 영상 제출

5/27 - 5/29

- 최종 마무리 및 Demo 준비

6/1 ~

- 도착한 엔코더 모터를 이용하여 PID 제어 코드를 통해 모터 제어
- carebuddy\_ros 로봇 패키지 완성하기
- gpt api와 구글 stt, tts api를 사용하여 음성으로 소통하는 기능 구현

[

감사합니다

]

Motus+er