

# 서비스(음식배달,정찰경찰 등) 로봇 및 관제 시스템 개발

진민혁, 서연우

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# 목차

## 1. 설계

- a. gui 상세 설계도
- b. 노드 설계

## 2. 구현

- a. DB Scheme
- b. setup.py
- c. menu.json과 gui창 연동
- d. 테이블 오더 화면
- e. 인터페이스 패키지
- f. 주방 화면
- g. 토픽 통신
- h. 액션 통신
- i. Logging (로그 레벨 설정)
- j. db Log에 저장된 피드백값
- k. 로봇의 액션 통신
- l. rqt\_graph
- m. QoS 옵션 설정

# gui 상세 설계도

figma틀로 gui 디자인 설계

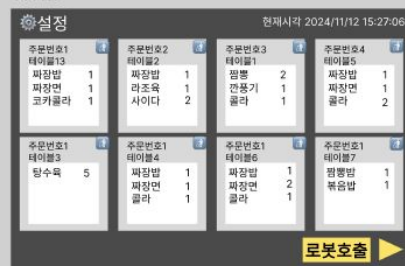
장바구니 팝업



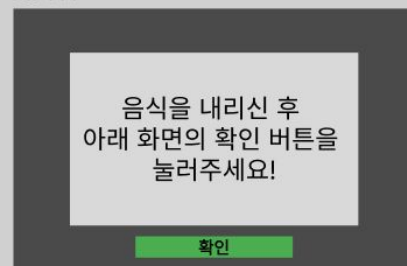
면 페이지



주방 화면



로봇 페이지



주문목록 확인



밥 페이지



주방 화면2

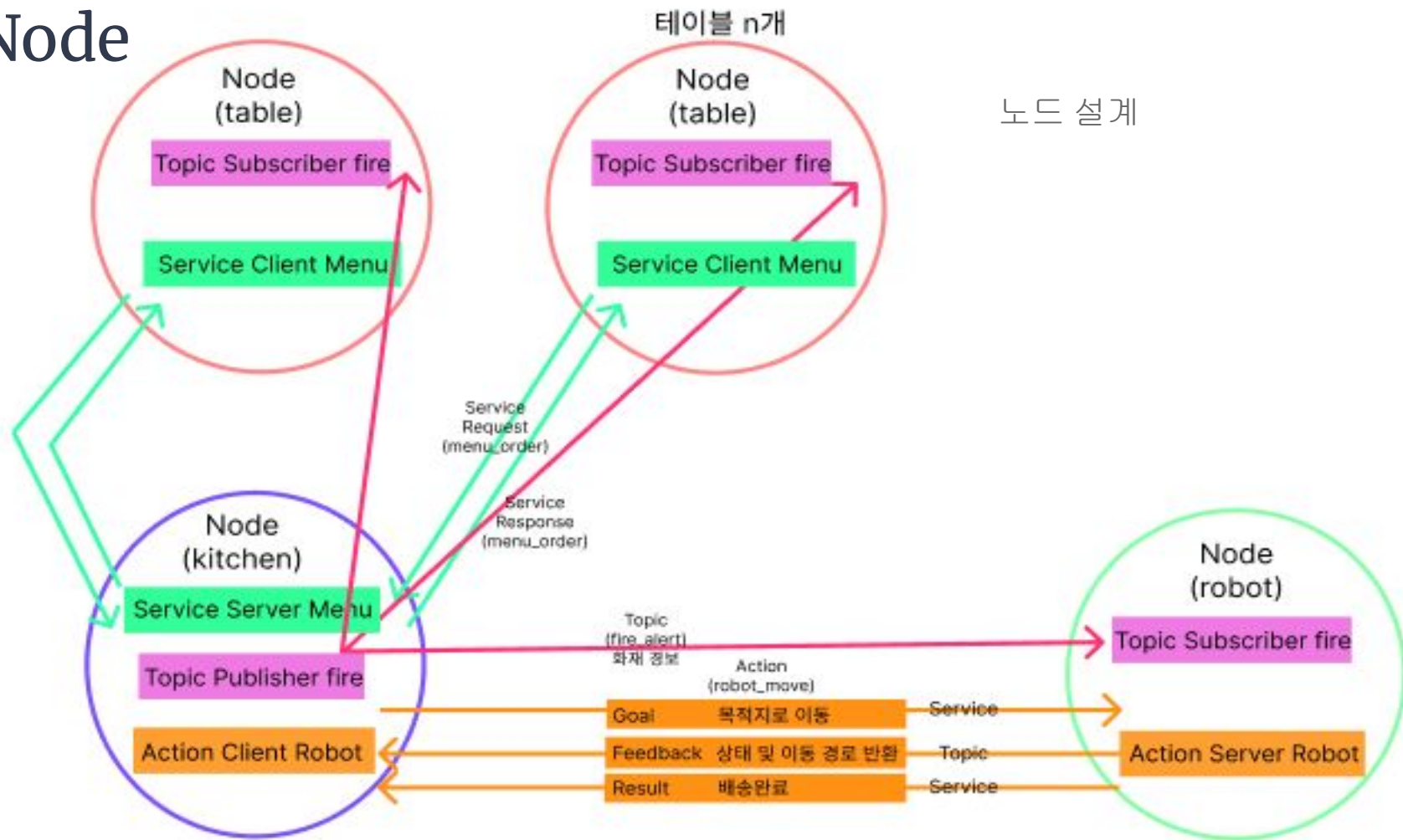


결제 페이지

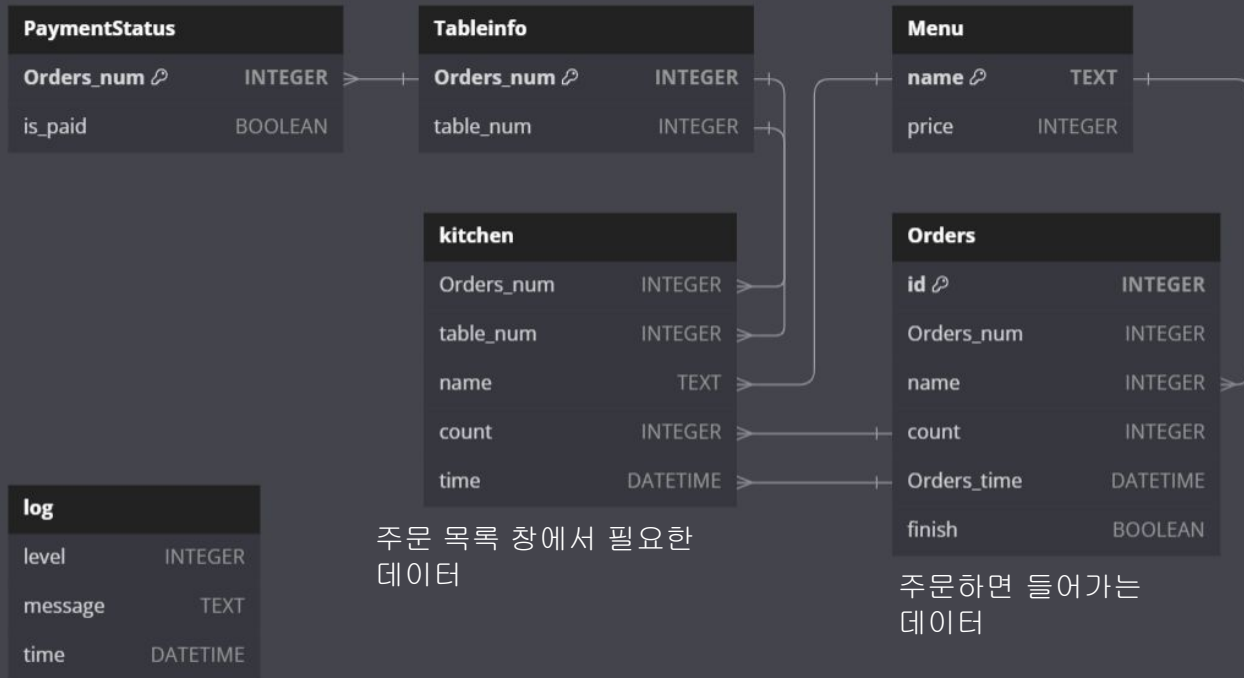


# Node

노드 설계



# DB Scheme

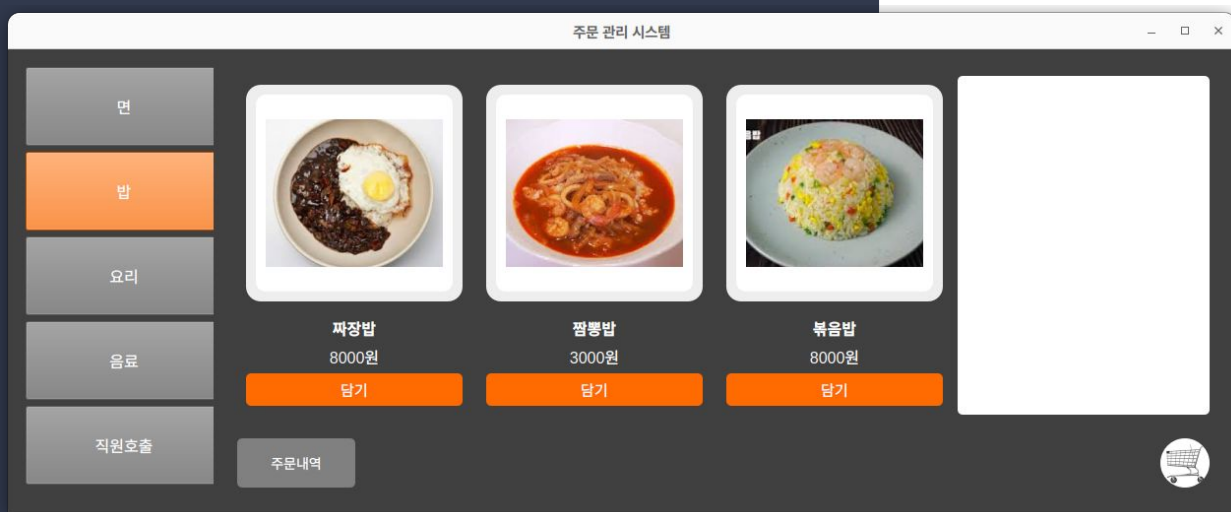


# setup.py

콘솔 스크립트 이름과 호출 함수 기입

```
15     maintainer="ynu",
16     maintainer_email="ynu@todo.todo",
17     description="TODO: Package description",
18     license="TODO: License declaration",
19     tests_require=["pytest"],
20     entry_points={
21         "console_scripts": [
22             "kitchen = table_order.kitchen_gui_main:main",
23             "table = table_order.table:main",
24             "robot = table_order.robot_gui:main"
25         ],
26     },
27 )
```

# menu.json과 gui창 연동

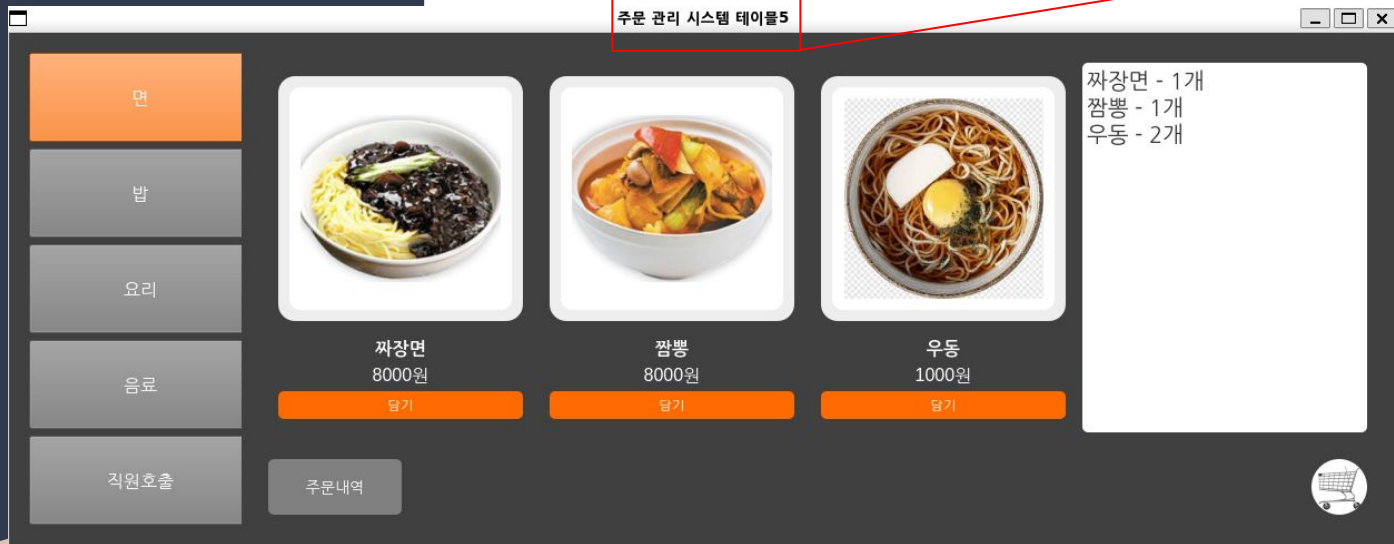


```
src > table_order > table_order > ( ) menus.json > ...  
1  {  
2    "면_짜장면": 8000,  
3    "면_짬뽕": 8000,  
4    "밥_짜장밥": 8000,  
5    "밥_짬뽕밥": 3000,  
6    "밥_볶음밥": 8000,  
7    "요리_탕수육": 8000,  
8    "요리_깐풍기": 8000,  
9    "요리_라조육": 8000,  
10   "음료_코카콜라": 2000,  
11   "음료_사이다": 2000,  
12   "음료_제로콜라": 2500,  
13   "직원호출_물": 0,  
14   "직원호출_물티슈": 0  
15 }
```

메모장으로 수정 가능

# 테이블 오더 화면

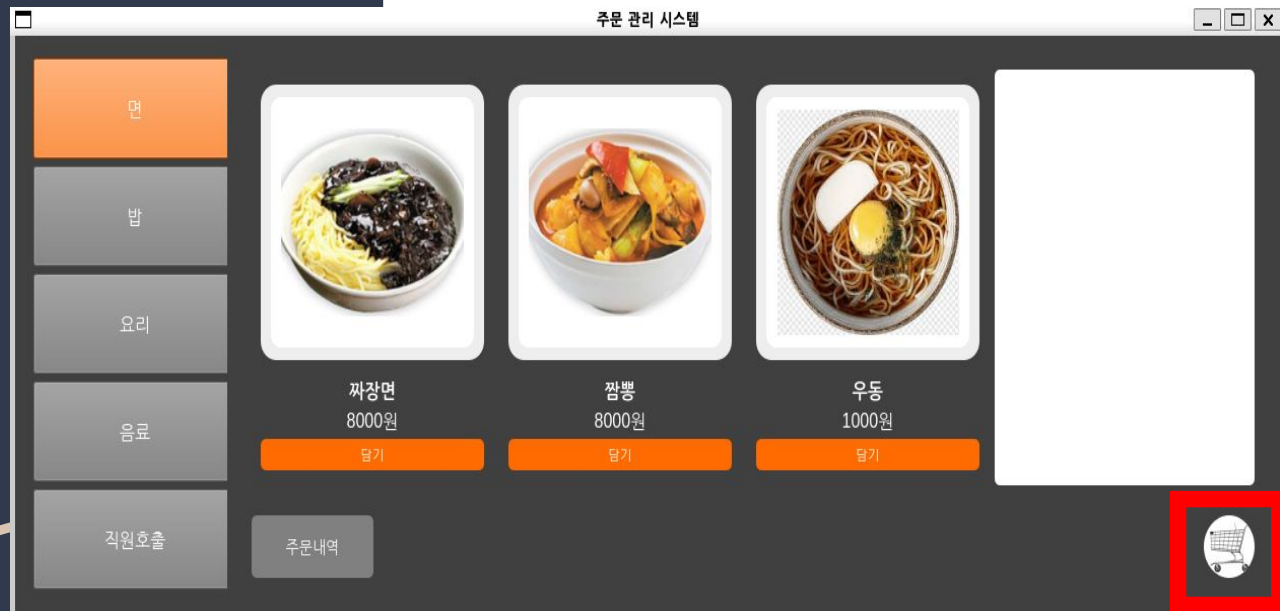
## 주문 관리 시스템 테이블5



1. 각 테이블의 번호(예: 1~9) 설정 가능
2. '담기' 버튼 클릭 흰 박스에 출력됨



# 테이블 오더 화면



장바구니 버튼 클릭 시 주문서에 주문한 메뉴별 수량  
계산되어 출력됨

# 인터페이스 패키지

## 서비스 통신

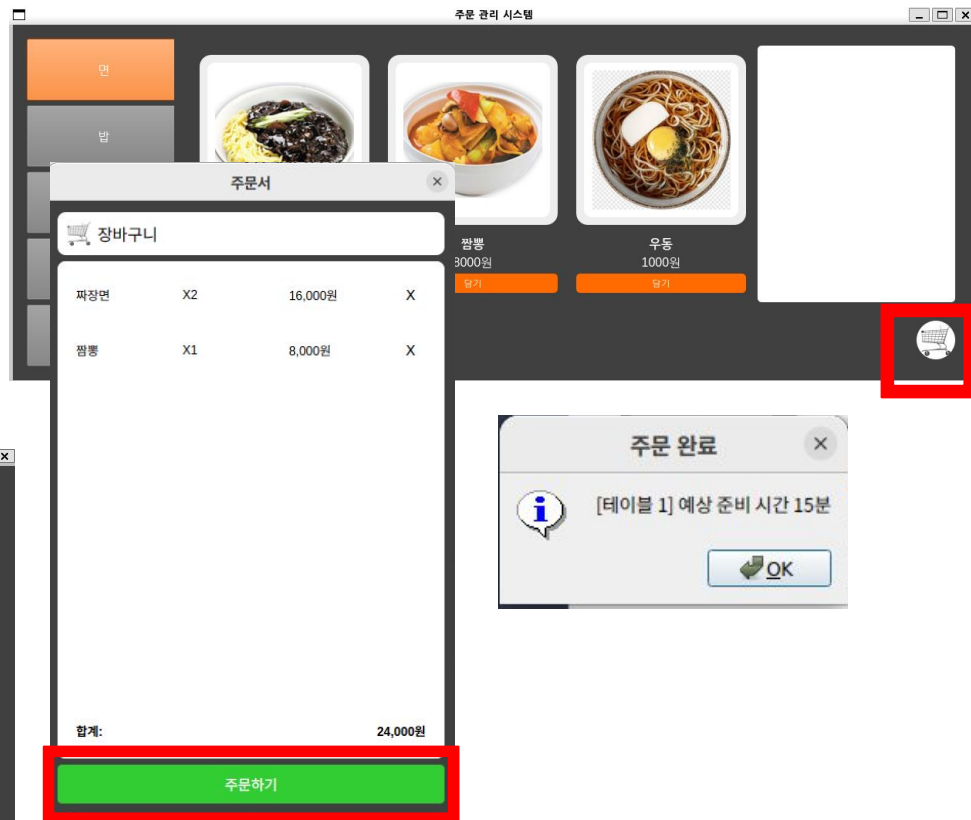
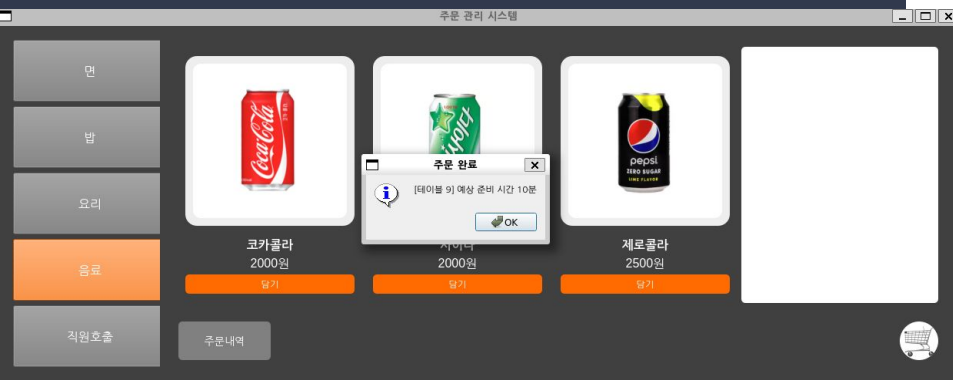
```
ynu@ynu-ROKEY:~/cr_ws$ ros2 interface package table_msgs  
table_msgs/srv/MenuOrder  
ynu@ynu-ROKEY:~/cr_ws$
```

```
ynu@ynu-ROKEY:~/cr_ws/src/table_msgs$ tree  
.  
├── CMakeLists.txt  
├── action  
├── include  
│   └── table_msgs  
├── msg  
├── package.xml  
├── src  
├── srv  
│   └── MenuOrder.srv
```

```
GNU nano 6.2 srv/MenuOrder.srv  
# Request  
string data  
---  
# Response  
int8 minute
```

# 테이블 오더 화면

## 서비스 통신



1. 주문하기 버튼 클릭 시 서비스 요청값으로 메뉴 수량 전달
2. 서비스 콜백으로 메뉴 하나당 5분으로 계산하여 응답값 반환
3. 지정한 테이블 번호 값으로 db의 Orders Table에 주문내역이 데이터로 들어감

# 테이블 오더 화면

## 서비스 통신

id	orders_...	name	count	orders_time	finish
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
63	26	팜뽕	1	2024-11-18 01:18:01	1
64	26	탕수육	1	2024-11-18 01:18:01	1
65	26	깡통기	1	2024-11-18 01:18:01	1
66	26	라조육	1	2024-11-18 01:18:01	1
67	26	코카콜라	1	2024-11-18 01:18:01	1
68	27	탕수육	1	2024-11-18 01:18:06	1
69	27	깡통기	1	2024-11-18 01:18:06	1
70	27	라조육	1	2024-11-18 01:18:06	1
71	28	물	8	2024-11-18 01:18:13	1

1. 주문하기 버튼 클릭 시 서비스 요청값으로 메뉴 수량 전달
2. 서비스 콜백으로 메뉴 하나당 5분으로 계산하여 응답값 반환
3. 지정한 테이블 번호 값으로 db의 Orders Table에 주문내역이 데이터로 들어감

# 테이블 오더 화면

주문 내역

주문 시간: 2024-11-18 09:50:46

짜장면	X2	16,000원
짬뽕	X1	8,000원

전체 합계:

24,000원

닫기

주문 관리 시스템


전

반

요리


음료

직원 호출



짜장면  
8000원


담기



짬뽕  
8000원

담기

주문내역



주문내역 버튼 클릭 시 주문했던 시간 및 총액이 출력됨

# 주방 화면

id	orders_...	name	count	orders_time	finish
87	87	36 짜장면	2	2024-11-18 12:15:05	0
88	88	36 짬뽕	1	2024-11-18 12:15:05	0

```
kitchen node init end
thread_node_kitchen start
spin start
get_menu_order start
{'table': '3', 'menu': {'짜장면': 2, '짬뽕': 1}}
cal: {'짜장면': 2, '짬뽕': 1}
[INFO] [1731899705.570638086] [kitchen_navi]: [테이블 3] : 메뉴 3개 준비시간 15분 정도 소요됩니다.
get_menu_order end
{'짜장면': 2, '짬뽕': 1}
```

주문 관리 시스템

2024년 11월 18일 12:16:48

⚙️ 설정

주문번호11

T6

짬뽕 1

우동 1

주문 완료

주문번호12

T6

짜장면 1

짬뽕 6

주문 완료

주문번호19

T7

짜장면 1

짬뽕 1

주문 완료

주문번호21

T7

짬뽕 1

짜장면 1

주문 완료

주문번호22

T7

짜장면 1

짬뽕 1

주문 완료

주문번호23

T7

짬뽕 1

짜장면 4

주문 완료

주문번호25

T7

짜장면 1

짬뽕 1

탕수육 1

간짜개 1

라조육 7

주문 완료

주문번호30

T9

짬뽕 1

짜장면 1

주문 완료

주문번호31

T9

짬뽕 1

짜장면 1

주문 완료

주문번호32

T9

짬뽕 1

짜장면 1

주문 완료

주문번호33

T9

짬뽕 1

짜장면 3

주문 완료

주문번호36

T3

짜장면 2

짬뽕 1

주문 완료

1. db를 통해 주문내역 데이터를 받아옴
2. 주문번호와 완료여부(Boolean)를 받아와 주문내역을 띄움

# 주방 화면

주문 관리 시스템

2024년 11월 18일 12:20:14

⚙️ 설정

주문번호11

T6

빤뽕 1

우동 1

주문 완료

주문번호12

T6

짜장면 1

빤뽕 6

주문 완료

주문번호19

T7

짜장면 1

빤뽕 1

주문 완료

주문번호21

T7

빤뽕 1

짜장면 1

주문 완료

주문번호22

T7

짜장면 1

주문 완료

주문번호23

T7

짜장면 1

주문 완료

주문번호25

T7

짜장면 1

빤뽕 1

탕수육 1

간퐁기 1

라조육 7

주문 완료

주문번호30

T9

빤뽕 1

짜장면 1

주문 완료

주문번호31

T9

빤뽕 1

짜장면 1

주문 완료

주문번호32

T9

빤뽕 1

짜장면 1

주문 완료

주문번호33

T9

빤뽕 1

짜장면 3

주문 완료

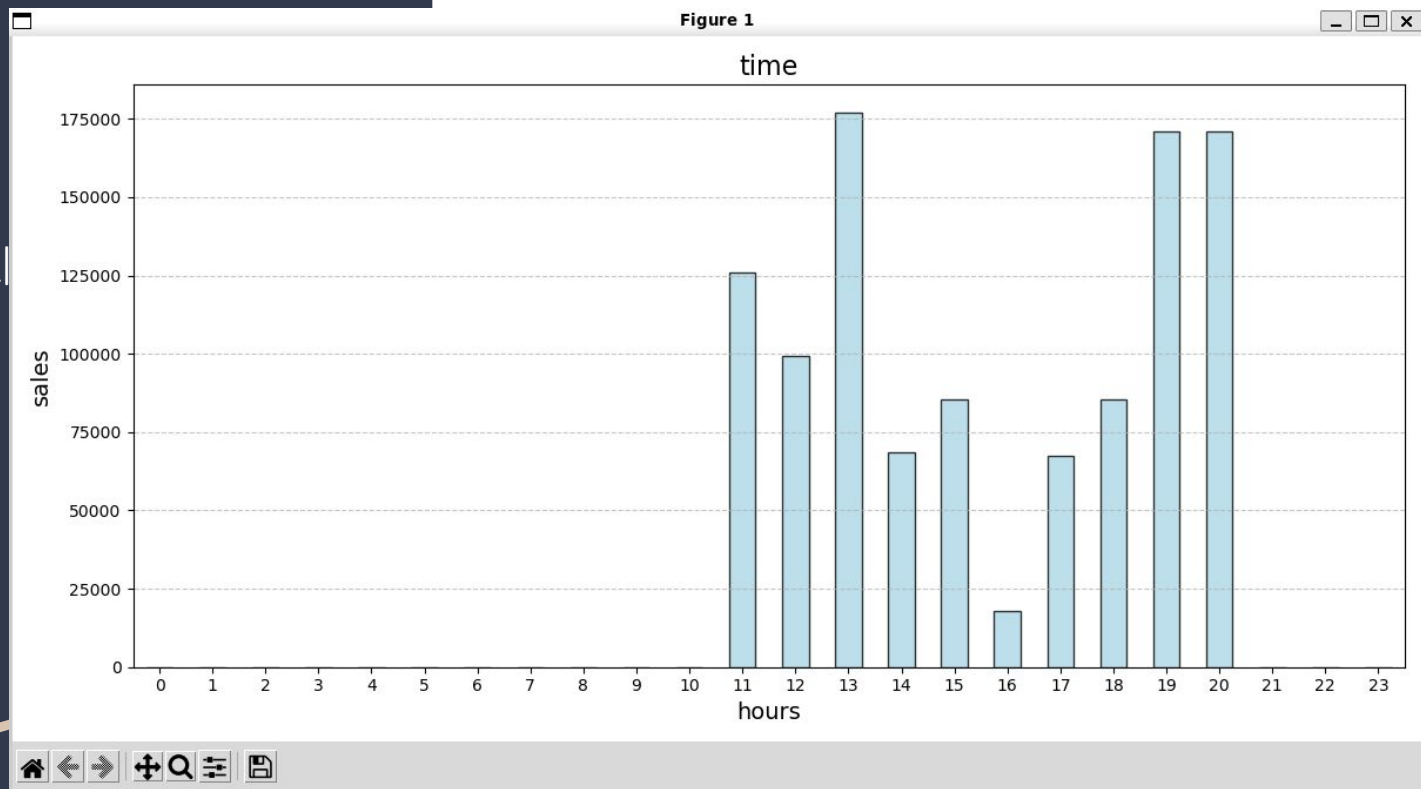
87	87	36	짜장면	2	2024-11-18 12:15:05	1
88	88	36	빤뽕	1	2024-11-18 12:15:05	1

주문 완료 클릭 시

db의 Orders Table의 완료여부(finish)값이 1로 바뀌면서 주방 화면에서 사라짐

# 주방 화면

관리자 설정 페이지



관리자 설정 창에서 db로 읽어온 시간대 별 매출 통계 확인 가능



# 토픽 통신

실시간으로 주방 노드에서 테이블 노드에 화재 발생  
토픽 발행 -> 'OK' 누르면 False 값으로 바꿔 thread 중지

주문 관리 시스템

화재발생

주방

```
[INFO] [1731897675.451384347] [table_node]: recieved: [fire 1731897676.4451997]
[INFO] [1731897675.961768482] [table_node]: recieved: [fire 1731897676.9508598]
[INFO] [1731897676.474192676] [table_node]: recieved: [fire 1731897677.461367]
```

주문 관리 시스템

```
[info][2024-11-17 23:30:09]목적지가 수락되었습니다.
[info][2024-11-17 23:30:09]목적지가 수락되었습니다.
[warning][2024-11-17 23:30:10]배송을 실패했습니다. (6)
[info][2024-11-17 23:30:10]목적지가 수락되었습니다.
[warning][2024-11-17 23:30:10]배송을 실패했습니다. (6)
[warning][2024-11-17 23:30:10]배송을 실패했습니다. (6)
[info][2024-11-17 23:30:10]목적지가 수락되었습니다.
[info][2024-11-17 23:30:41]주방에 도착했습니다.
[info][2024-11-17 23:37:54]Service /set_initial_pose not
available, waiting again...
[info][2024-11-17 23:47:06]목적지가 수락되었습니다.
[info][2024-11-17 23:47:06]목적지에 도착했습니다.
[info][2024-11-17 23:47:06]주방에 도착했습니다.
[info][2024-11-18 00:37:17]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 00:37:17]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 00:37:18]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:03:36]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:18:33]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:38:14]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:38:14]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:38:15]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:39:15]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 01:55:23]화재발생
[info][2024-11-18 01:58:01]화재발생
[info][2024-11-18 01:59:58]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 02:00:19]화재발생
[info][2024-11-18 02:00:31]목적지가 수락되었습니다.
[info][2024-11-18 02:00:49]목적지에 도착했습니다.
[info][2024-11-18 02:00:49]1번 테이블에 도착했습니다.
[info][2024-11-18 02:23:24]화재발생
[info][2024-11-18 02:24:10]화재발생
[info][2024-11-18 02:24:15]화재발생
[info][2024-11-18 10:12:22]Service /set_initial_pose not
available, waiting again...
[info][2024-11-18 10:49:12]화재발생
[info][2024-11-18 10:49:44]화재발생
```

화재발생



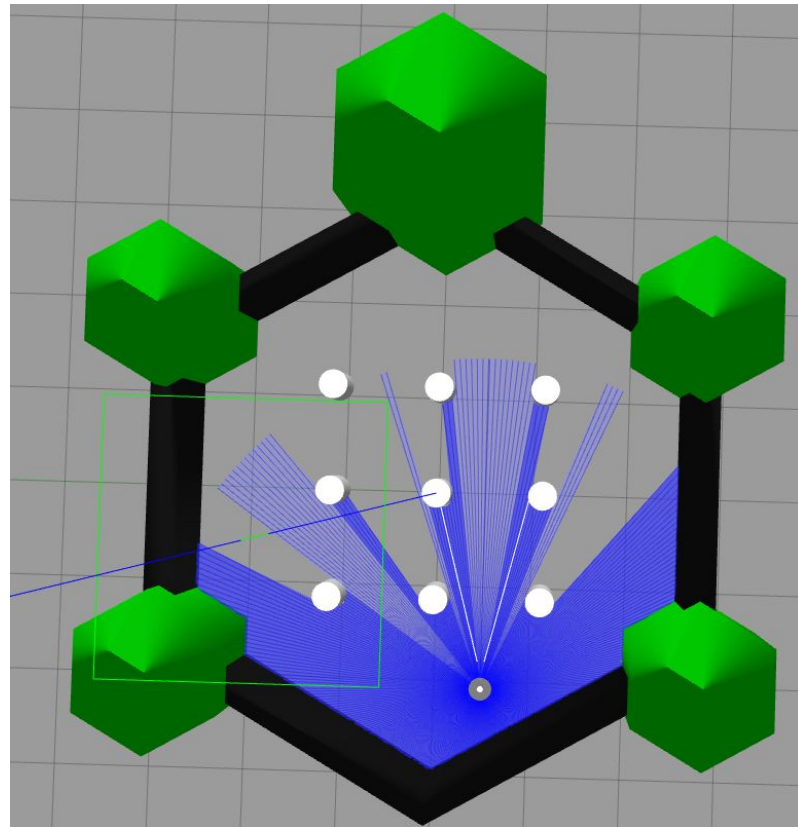
화재발생

테이블에 알립니다.

OK

화재경보

주방 노드 실행 시, gazebo / rviz 위치 주방으로  
초기화



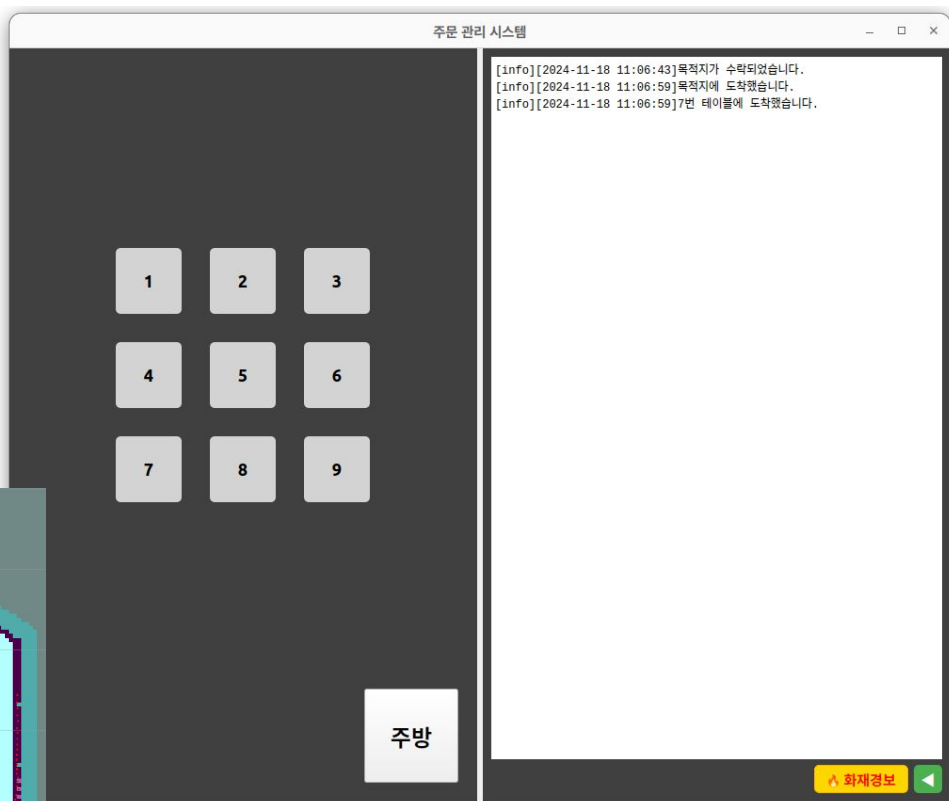
# 액션 통신

1. 주방에서 테이블 번호나 주방 버튼을 누르면, gazebo에 명령을 내려 특정 테이블/주방으로 이동하게 요청함  
→ 액션 요청 시, “목적지가 수락되었습니다” 메시지 알림
2. 이동 위치 값(진행 상태)을 피드백으로 실시간 받아와 db의 Log Table에 데이터 저장
3. 액션 결과를 “n번 테이블/주방에 도착했습니다.” 메시지로 목적지 도착 알림

# 액션 통신

```
self.goal_poses = [  
    [1.2, 0.8], # table1  
    [1.2, -0.2], # table2  
    [1.2, -1.5], # table3  
    [0.0, 0.8], # table4  
    [0.0, -0.2], # table5  
    [0.0, -1.5], # table6  
    [-1.2, 0.9], # table7  
    [-1.2, -0.2], # table8  
    [-1.2, -1.5], # table9  
    [-2.0, -0.5], # kitchen  
]
```

1. 서빙 로봇에게 7번 테이블로 명령
2. 서빙 로봇에게 주방으로 돌아오라고 명령



```
[info][2024-11-18 11:06:43]목적지가 수락되었습니다.  
[info][2024-11-18 11:06:59]목적지에 도착했습니다.  
[info][2024-11-18 11:06:59]7번 테이블에 도착했습니다.  
[info][2024-11-18 11:08:32]목적지가 수락되었습니다.  
[info][2024-11-18 11:08:48]목적지에 도착했습니다.  
[info][2024-11-18 11:08:48]주방에 도착했습니다.
```

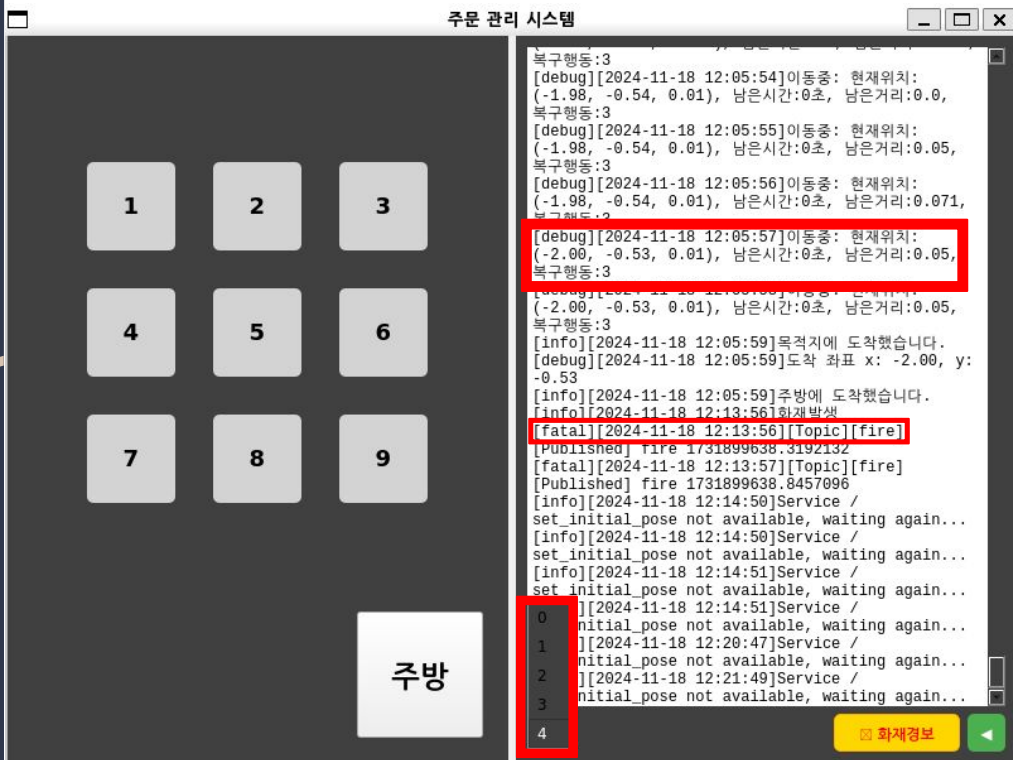
# Logging (로그 레벨 설정)

1. read\_log() 값을 0~4까지 토글로 변경 가능
2. log\_level을 log\_level = {0: "info", 1: "warning", 2: "error", 3: "debug", 4: "fatal"}로 설정
3. 선택한 level의 log 정보 gui로 읽어옴

(모든 log 값은 db에 저장)

```
level_dict = {
    0: "info",
    1: "warning",
    2: "error",
    3: "debug",
    4: "fatal",
}
global combo_log_level
while True:
    log_level_from_combo = 4
    if type(combo_log_level) == type(QComboBox()):
        log_level_from_combo = int(combo_log_level.currentText())
    log_data_all = self.db_conn.read_log(log_level_from_combo)
    log_readable = []
    for msg in log_data_all:
        log_readable.append(f"[{level_dict[msg[0]]}] [{msg[2]]} {msg[1]}")
```

```
[info][2024-11-18 11:21:52]목적지가 수락되었습니다.
[warning][2024-11-18 11:21:52]배송을 실패했습니다. (6)
[info][2024-11-18 11:21:54]목적지가 수락되었습니다.
[warning][2024-11-18 11:21:54]배송을 실패했습니다. (6)
[info][2024-11-18 11:22:15]목적지에 도착했습니다.
[info][2024-11-18 11:22:15]1번 테이블에 도착했습니다.
```





# db Log에 저장된 피드백값

TABLES
> Log
> Menu
> Orders
> PaymentStatus
> Tableinfo
> sqlite_sequence

	id	level	message	time
	Filter...	Filter...	Filter...	Filter...
1	36371	0	목적지가 수락되었습니다.	2024-11-18 11:06:43
2	36372	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:0.0, 복구행동:0	2024-11-18 11:06:43
3	36373	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:0.0, 복구행동:0	2024-11-18 11:06:43
4	36374	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:0.0, 복구행동:0	2024-11-18 11:06:43
5	36375	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
6	36376	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
7	36377	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
8	36378	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
9	36379	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
10	36380	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
11	36381	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
12	36382	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
13	36383	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
14	36384	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
15	36385	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
16	36386	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
17	36387	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
18	36388	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
19	36389	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
20	36390	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
21	36391	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
22	36392	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
23	36393	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
24	36394	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
25	36395	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:0초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
26	36396	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:131초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
27	36397	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:131초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
28	36398	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:131초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
29	36399	3	이동중: 현재위치:(-1.98, -0.5, 0.01), 남은시간:131초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
30	36400	3	이동중: 현재위치:(-1.97, -0.5, 0.01), 남은시간:66초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43
3,146	36401	3	이동중: 현재위치:(-1.97, -0.5, 0.01), 남은시간:66초, 남은거리:1.5, 복구행동:0	2024-11-18 11:06:43

# 로봇의 액션 통신

[메뉴 다 가져갔을 시, 주방으로 되돌아가는 시스템]

1. 로봇 노드에서 '확인' 버튼으로 목적지(주방) 요청
2. 위치 피드백값 db Log Table에 저장
3. 주방의 액션 통신으로 7번 테이블로 이동했던 gazebo 로봇이 다시 주방으로 돌아감  
→ 주방 내비 화면에서 액션 응답값을 내비게이션 로그 창으로 확인 가능

7번 테이블로 이동했던

로봇



주방으로 되돌아가는

로봇

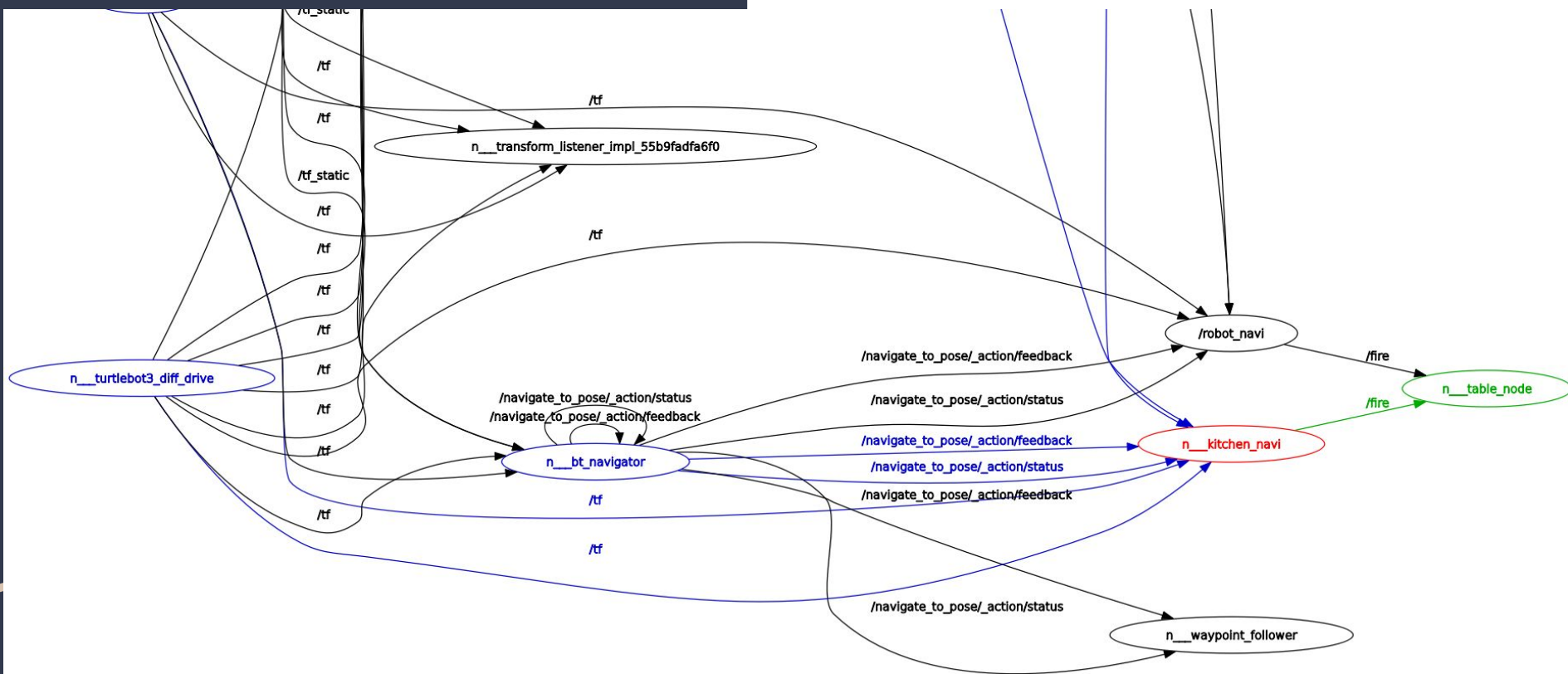


로봇 화면

음식을 내리신 후  
아래 화면의 확인 버튼을  
눌러주세요!

확인

# rqt\_graph





# QoS 옵션 설정

## 토픽

```
self.subscription = self.create_subscription(
    String,
    "fire",
    self.listener_callback,
    qos_profile=QoSProfile(
        reliability=QoSReliabilityPolicy.BEST_EFFORT, # 속도 중점
        reliability=QoSReliabilityPolicy.RELIABLE, # 유실 방지
        history=QoSHistoryPolicy.KEEP_LAST, # 최신 값만 저장
        history=QoSHistoryPolicy.KEEP_ALL, # 전부 저장해놓고 나중에라도 다시 전송
        depth=100,
        durability=QoSDurabilityPolicy.TRANSIENT_LOCAL, # 구독 전 메시지 유지
        durability=QoSDurabilityPolicy.VOLATILE, # 구독 전 데이터 유지 안함
    ),
    callback_group=self.callback_group,
)
```

```
# Service: Server ; Menu 선언
self.menu_order_server = self.create_service(
    MenuOrder, # srv 타입
    "menu_order", # 서비스명
    self.get_menu_order, # 콜백함수 (서비스 클라이언트-서비스 요청 있을 때마다)
    callback_group=self.callback_group, # 멀티 스레드 병렬 콜백함수 실행
    qos_profile=QoSProfile(
        reliability=QoSReliabilityPolicy.BEST_EFFORT, # 속도 중점
        reliability=QoSReliabilityPolicy.RELIABLE, # 유실 방지
        history=QoSHistoryPolicy.KEEP_LAST, # 최신 값만 저장
        history=QoSHistoryPolicy.KEEP_ALL, # 전부 저장해놓고 나중에라도 다시 전송
        depth=5,
        durability=QoSDurabilityPolicy.TRANSIENT_LOCAL, # 구독 전 메시지 유지
        durability=QoSDurabilityPolicy.VOLATILE, # 구독 전 데이터 유지 안함
    ),
)
```

# QoS 옵션 설정

## 액션

```
self.navigate_to_pose_action_client = ActionClient(  
    self,  
    NavigateToPose,  
    "navigate_to_pose",  
    goal_service_qos_profile=QoSProfile(  
        # reliability=QoSReliabilityPolicy.BEST_EFFORT, # 속도 중점  
        reliability=QoSReliabilityPolicy.RELIABLE, # 유실 방지  
        # history=QoSHistoryPolicy.KEEP_LAST, # 최신 값만 저장  
        history=QoSHistoryPolicy.KEEP_ALL, # 전부 저장해놓고 나중에라도 다시 전송  
        # depth=5,  
        durability=QoSDurabilityPolicy.TRANSIENT_LOCAL, # 구독 전 메시지 유지  
        # durability=QoSDurabilityPolicy.VOLATILE, # 구독 전 데이터 유지 안함  
    ),  
    feedback_sub_qos_profile=QoSProfile(  
        reliability=QoSReliabilityPolicy.BEST_EFFORT, # 속도 중점  
        # reliability=QoSReliabilityPolicy.RELIABLE, # 유실 방지  
        history=QoSHistoryPolicy.KEEP_LAST, # 최신 값만 저장  
        # history=QoSHistoryPolicy.KEEP_ALL, # 전부 저장해놓고 나중에라도 다시 전송  
        depth=100,  
        # durability=QoSDurabilityPolicy.TRANSIENT_LOCAL, # 구독 전 메시지 유지  
        durability=QoSDurabilityPolicy.VOLATILE, # 구독 전 데이터 유지 안함  
    ),  
    result_service_qos_profile=QoSProfile(  
        # reliability=QoSReliabilityPolicy.BEST_EFFORT, # 속도 중점  
        reliability=QoSReliabilityPolicy.RELIABLE, # 유실 방지  
        # history=QoSHistoryPolicy.KEEP_LAST, # 최신 값만 저장  
        history=QoSHistoryPolicy.KEEP_ALL, # 전부 저장해놓고 나중에라도 다시 전송  
        # depth=5,  
        durability=QoSDurabilityPolicy.TRANSIENT_LOCAL, # 구독 전 메시지 유지  
        # durability=QoSDurabilityPolicy.VOLATILE, # 구독 전 데이터 유지 안함  
    ),  
)
```

# 시연 순서

[실행]

gazebo, rviz

kitchen

table, robot

화재경보

내비 이동 (주방, 로봇)

테이블 오더 및 예상시간

주방 주문완료 (db 확인)

관리자 설정 페이지 → 통계 확인