

# Serving Robot System with Integrated Kitchen and Table Displays for Restaurant Automation

# OUTLINE

- 
1. Introduction
  2. System Design
  3. Implementation
  4. Result
  5. Conclusion & Future Work

# Introduction

## Project Background



- 레스토랑 운영의 핵심 : 빠르고 정확한 서비스
- 주문 관리 및 서빙 과정에서의 효율성은 고객 만족도와 직접적으로 연결
- 기존 방식의 한계: 주문 정보 전달의 지연 및 혼란 발생 가능

# | Introduction

## Project Objective

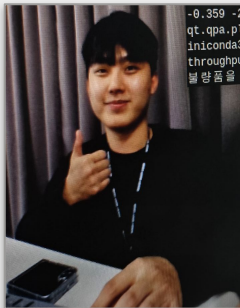
- 서빙 로봇과 주방/테이블 디스플레이 시스템을 통합하여 식당 운영 자동화 시스템 구축
- 효율성 향상  
주문 관리의 디지털화  
로봇을 활용한 자동화 서빙
- 고객 경험 개선  
테이블 디스플레이를 통한 주문 상태 실시간 제공
- 시뮬레이션 환경 검증  
실제 환경 적용 가능성을 평가하기 위한 테스트베드 설계

# Introduction

## Members



윤민식  
DB 담당



정승환  
터틀봇 담당



김승중  
주방 디스플레이  
담당



장준하  
테이블 오더 담당

# | System Design

## Technical Components

TurtleBot



ROS2

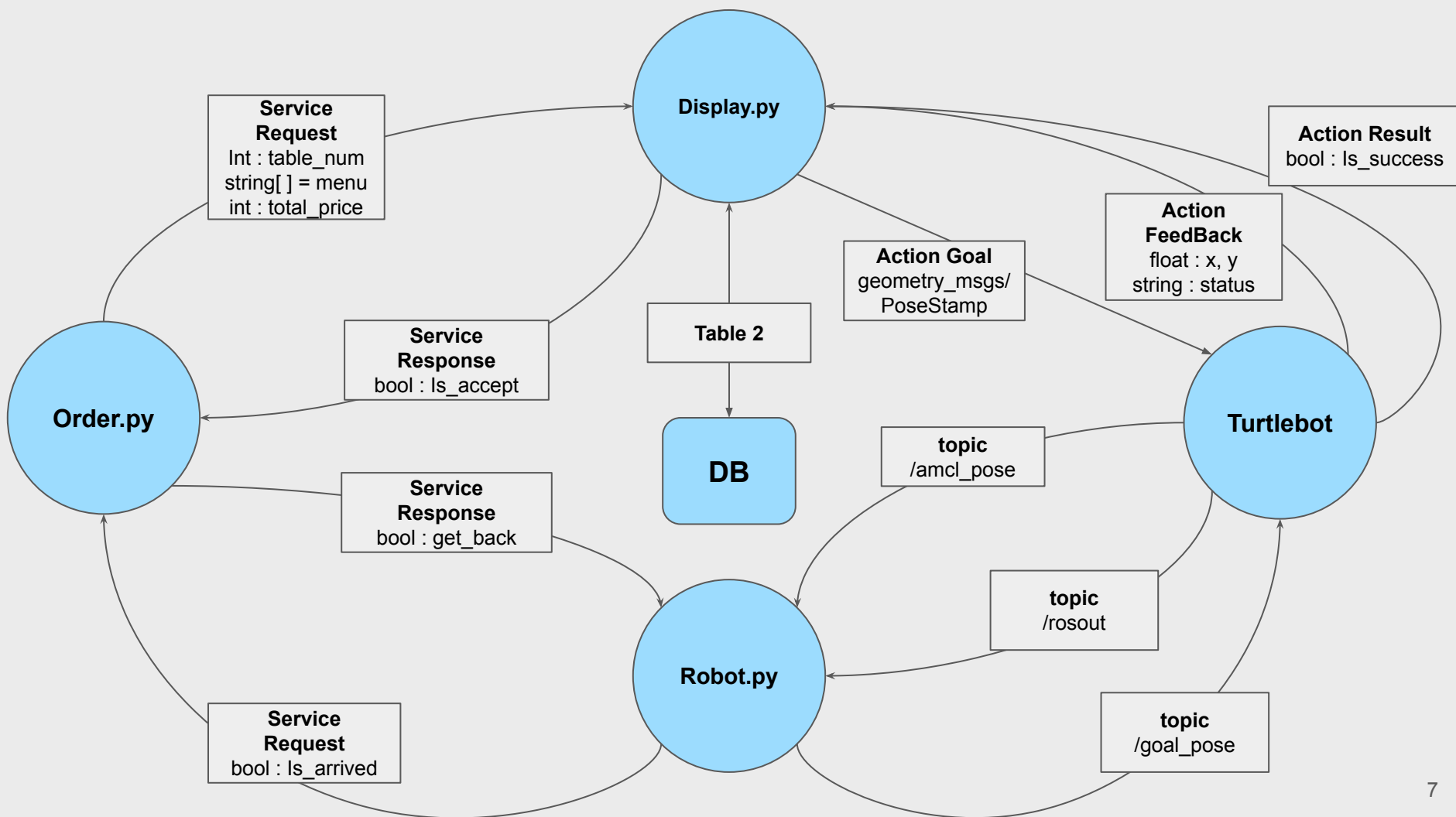


PyQt5



SQLite





# Implementation

## Kitchen Display

주방 디스플레이					
테이블 3	테이블 6	테이블 2	테이블 9	테이블 1	추가 주문
쫄지마라탕 파인샤베트 제로콜라 참이슬 참이슬 참이슬 참이슬 참이슬		통돈가스김치우동 환타포도		라면 라면 제로사이다 제로사이다 제로사이다	테이블 8: 치즈라볶이, 치즈라볶이 테이블 7: 과일화채, 참이슬오리지널, 참
무구리 무구리 참이슬 참이슬	참이슬 황도 얼큰수제비		참이슬 참이슬 참이슬 참이슬 참이슬 참이슬	캘리	
출발	출발	출발	출발	출발	
정산하기					



# Implementation

## Kitchen Display Code Review

조리 완료된 메뉴, 조리 완료 칸으로 이동 & 출발 버튼  
활성화

```
def move_to_completed(self, item, index):  
    """  
    클릭된 메뉴를 완료된 메뉴로 이동  
    """  
    _, pending_menu_list, completed_menu_list, start_button = self.table_widgets[index]  
    completed_menu_list.addItem(item.text()) # 완료된 메뉴로 이동  
    pending_menu_list.removeItem(pending_menu_list.row(item)) # 조리할 메뉴에서 삭제  
  
    # 모든 메뉴가 완료된 경우 버튼 활성화 및 색깔 변경  
    if pending_menu_list.count() == 0:  
        start_button.setEnabled(True)  
        start_button.setStyleSheet("background-color: #4CAF50; color: white; font-size: 18px; padding: 10px; border-radius: 5px;")  
    else:  
        start_button.setEnabled(False)  
        start_button.setStyleSheet("background-color: #FF5722; color: white; font-size: 18px; padding: 10px; border-radius: 5px;")
```

# Implementation

## Kitchen Display Code Review

출발 버튼 눌렀을 때 왼쪽으로 1칸 shift

```
def start_robot(self, index):  
    """  
    로봇 출발 처리  
    """  
    # 완료된 메뉴 초기화  
    table_label, pending_menu_list, completed_menu_list, start_button = self.table_widgets[index]  
    table_number = table_label.text().replace("테이블 ", "").strip()  
  
    completed_menu_list.clear()  
    table_label.setText("")  
    pending_menu_list.clear()  
  
    # 버튼 비활성화 및 색상 초기화  
    start_button.setEnabled(False)  
    start_button.setStyleSheet("background-color: #FF5722; color: white; font-size: 18px; padding:  
    | | | | | "border-radius: 5px;")
```

```
# 기존 테이블 내용 왼쪽으로 Shift  
for i in range(index, len(self.table_widgets) - 1):  
    next_label, next_pending, next_completed, next_button = self.table_widgets[i + 1]  
    current_label, current_pending, current_completed, current_button = self.table_widgets[i]  
  
    # 테이블 정보 이동  
    current_label.setText(next_label.text())  
    current_pending.clear()  
    current_completed.clear()  
  
    for j in range(next_pending.count()):  
        current_pending.addItem(next_pending.item(j).text())  
    for j in range(next_completed.count()):  
        current_completed.addItem(next_completed.item(j).text())  
  
    # 버튼 활성화 상태와 스타일 이동  
    current_button.setEnabled(next_button.isEnabled())  
    current_button.setStyleSheet(next_button.styleSheet())  
  
    # 다음 테이블 데이터 초기화  
    next_label.setText("")  
    next_pending.clear()  
    next_completed.clear()  
    next_button.setEnabled(False)  
    next_button.setStyleSheet("background-color: #FF5722; color: white; font-size: 18px; paddi  
    | | | | | "border-radius: 5px;")
```

# Implementation

## Kitchen Display Code Review

### 추가 주문 목록 shift

```
# 추가 주문에서 첫 번째 주문을 맨 오른쪽으로 이동
if self.extra_orders_list.count() > 0:
    next_order_text = self.extra_orders_list.takeItem(0).text() # 첫 번째 추가 주문 텍스트를 가져옴
    last_label, last_pending, _, last_button = self.table_widgets[-1] # 마지막 테이블 정보 가져옴

    # 테이블 번호와 메뉴 추출
    table_info, menu_info = next_order_text.split(":") # 테이블 정보와 메뉴 정보를 분리
    table_label = table_info.strip() # 테이블 번호
    last_label.setText(table_label) # 마지막 테이블의 레이블에 테이블 번호 설정

    # 메뉴 항목을 나누어서 추가
    menu_items = menu_info.split(",") # 메뉴 항목을 쉼표로 분리
    for menu_item in menu_items:
        menu_name = menu_item.split(" ")[0].strip() # 메뉴 이름만 추출
        last_pending.addItem(menu_name) # 메뉴 이름을 마지막 테이블의 '대기 메뉴' 목록에 추가

    # 추가된 마지막 테이블의 버튼 비활성화 및 색상 초기화
    last_button.setEnabled(False)
    last_button.setStyleSheet("background-color: #FF5722; color: white; font-size: 18px; padding: 5px; border-radius: 5px;")

if table_number:
    self.node.send_goal(int(table_number))
```

# Implementation

## Table Display

부리부리대마왕포차

NO.3

메인메뉴

사이드메뉴

음료

주류

직원 호출

메인메뉴

얼큰수제비 15,000원

짜빠구리범벅세트 13,000원

치즈라볶이범벅세트 13,000원

짜빠구리 10,000원

결제 확인

정바구리의 상품과 수량을 확인하셨습니다?

결제 완료

주문이 접수되었습니다!

OK

취소

결제하기

합계: 144,000원

메뉴	수량	현재 가격
1 무구리	- 2 +	70,000원
2 풀지마라탕	- 1 +	21,000원
3 파인사베트	- 1 +	6,000원
4 제로콜라	- 1 +	2,000원
5 참이슬	- 10 +	45,000원

총 금액: 144,000원

전체 제거

주문하기

# Implementation

## Table Display Code Review

카테고리 클릭시 해당 위치로 이동

```
def scroll_to_category(self, category):
    """카테고리 버튼 클릭 시 해당 위치로 스크롤"""
    if category in self.category_positions:
        target_widget = self.category_positions[category]
        # QScrollArea의 verticalScrollBar를 사용하여 스크롤 이동
        self.menu_scroll.verticalScrollBar().setValue(target_widget.y())
```

장바구니에 메뉴 추가

```
def add_to_cart(self, name, price):
    """메뉴를 장바구니에 추가"""
    if name in self.cart:
        self.cart[name]['quantity'] += 1
    else:
        self.cart[name] = {'price': price, 'quantity': 1}
    self.update_cart()
```

# Implementation

## Table Display Code Review

### 장바구니 업데이트

```
def update_cart(self):  
    """장바구니 테이블 업데이트"""  
    self.cart_table.setRowCount(len(self.cart))  
    total = 0  
    for row, (name, data) in enumerate(self.cart.items()):  
        total += data['price'] * data['quantity']
```

```
# 테이블 이름  
self.cart_table.setItem(row, 0, QTableWidgetItem(name))  
  
# 줄의 고정 크기(양쪽) (줄의 폭 + 증가/감소 버튼)  
quantity_widget = QWidget()  
quantity_layout = QHBoxLayout(quantity_widget)  
quantity_layout.setContentsMargins(0, 0, 0, 0)  
  
# 감소 버튼  
decrease_button = QPushButton("-")  
decrease_button.setFixedSize(30, 30)  
decrease_button.setStyleSheet("font-size: 14px;")  
decrease_button.clicked.connect(lambda _, n=name: self.decrease_quantity(n))  
quantity_layout.addWidget(decrease_button)  
  
# 중앙 표시  
quantity_label = QLabel(str(data['quantity']))  
quantity_label.setAlignment(Qt.AlignCenter)  
quantity_layout.addWidget(quantity_label)  
  
# 증가 버튼  
increase_button = QPushButton("+")  
increase_button.setFixedSize(30, 30)  
increase_button.setStyleSheet("font-size: 14px;")  
increase_button.clicked.connect(lambda _, n=name: self.increase_quantity(n))  
quantity_layout.addWidget(increase_button)
```

```
# 현재 가격  
self.cart_table.setItem(row, 2, QTableWidgetItem(f"{data['price'] * data['quantity']:,}원"))  
  
self.total_label.setText(f"총 금액: {total:,}원")
```

# Implementation

## Table Display Code Review

### 결제

```
def process_payment(self, popup):
    table_num = 3
    menu_list = [(name, details['price'] * details['quantity']) for name, details
    total_price = sum(item['price'] * item['quantity'] for item in self.cart.values)

    order_request = (table_num, menu_list, total_price)

    def send_request():
        global is_accept
        self.ros2_client.send_order(order_request)
        time.sleep(1)
        if is_accept:
            msg_box = QMessageBox(QMessageBox.Information, "결제 완료", "주문이 접수되었습니다.")
            msg_box.exec_()
            self.cart.clear()
            self.update_cart()
        else:
            QMessageBox.warning(self, "결제 실패", "주문에 실패했습니다.")

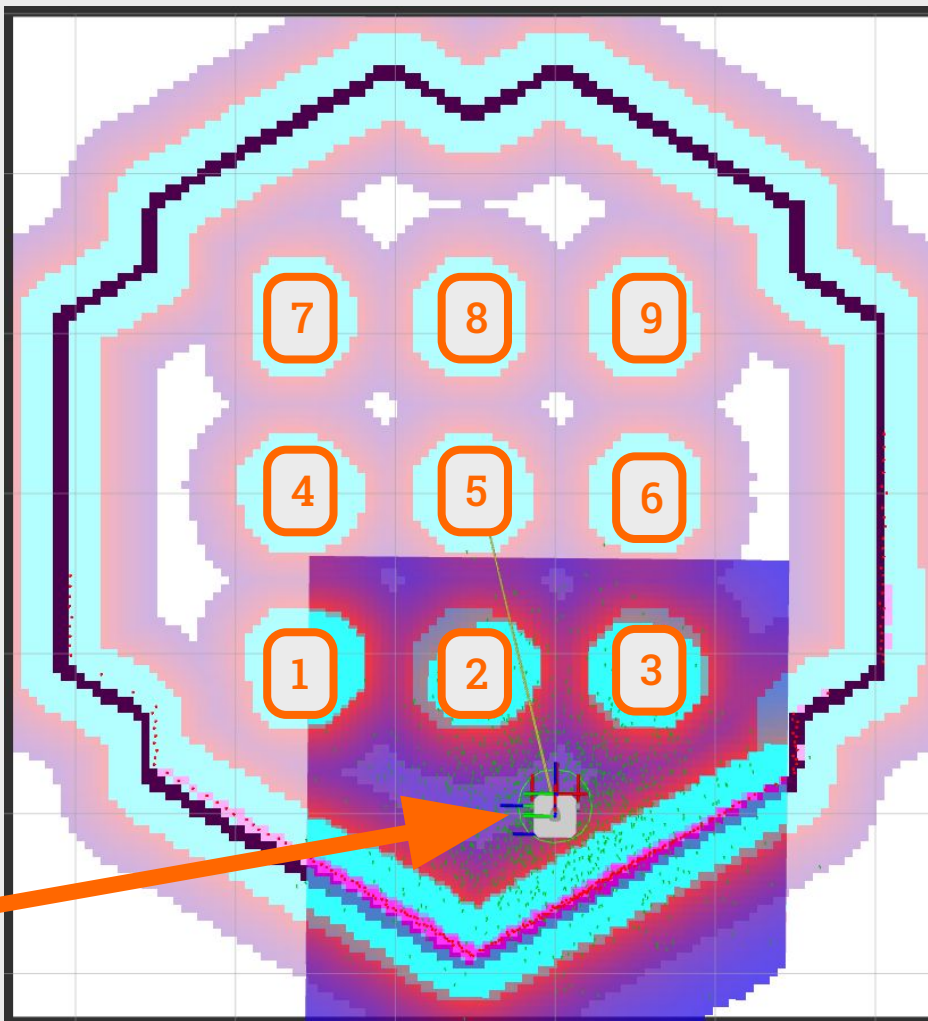
    threading.Thread(target=send_request, daemon=True).start()
```



# Implementation

TurtleBot

초기  
위치





# Implementation

## TurtleBot Code Review

목표 지점에 도달했을 때

```
# Goal succeeded
self.sub_current_status = self.create_subscription(
    Log, 'rosout', self.goal_status_callback, 10
)
```

```
def goal_status_callback(self, msg):
    if "Goal succeeded" in msg.msg:
        self.get_logger().info("목표 지점 도달")

        if abs(self.x) < 0.5 and abs(self.y) < 0.5:
            self.get_logger().info("arrived initial pose")
        else:
            self.get_logger().info("arrived table")
            self.send_goal_status(True)
    else:
        self.get_logger().error('Service call failed: goal_status_callback')
```

# Implementation

## TurtleBot Code Review

### 테이블에 도착

```
def send_goal_status(self, is_arrived):
    self.req.is_arrived = is_arrived
    self.future = self.client.call_async(self.req)
    self.future.add_done_callback(self.response_callback)

def response_callback(self, future):
    if future.result():
        response = future.result()
        if response.get_back():
            self.get_logger().info("go to initial pose")
            self.to_initial_pose()
        else:
            self.get_logger().error('Service call failed: response_callback')
```

### 초기 위치로 복귀

```
def to_initial_pose(self):
    self.get_logger().info("이동 중 . . . ")
    self.initial_pose.header.stamp = self.get_clock().now().to_msg()
    self.pub_initial_pose.publish(self.initial_pose)
```

# | System Design

## Data Flow Diagram

**Table 1. <메뉴, 가격, 원가>**
























**Table 2. <테이블 번호, 주문 내역, 총액, 시간, 주문  
번호>**

**Table 3. <테이블 번호, 테이블 좌표>**

**Table 4. <출발시간, 도착시간>**

# Implementation

## SQLite

Name	Type	Schema
▼  Tables (5)		
▼  image_file		CREATE TABLE image_file( name text not null, img blob not null )
 name	text	"name" text NOT NULL
 img	blob	"img" blob NOT NULL
▼  menu		CREATE TABLE menu( name text primary key not null, category text not null, price integer not null, cost_price integer not null )
 name	text	"name" text NOT NULL
 category	text	"category" text NOT NULL
 price	integer	"price" integer NOT NULL
 cost_price	integer	"cost_price" integer NOT NULL
▼  menu_order		CREATE TABLE menu_order( order_number integer primary key not null, order_time text not null, menu_list text not null , table_number INTEGER)
 order_number	integer	"order_number" integer NOT NULL
 order_time	text	"order_time" text NOT NULL
 menu_list	text	"menu_list" text NOT NULL
 table_number	INTEGER	"table_number" INTEGER
▼  moving		CREATE TABLE moving( start_time text not null, end_time text not null )
 start_time	text	"start_time" text NOT NULL
 end_time	text	"end_time" text NOT NULL
▼  pose		CREATE TABLE pose( table_number integer not null, pose_x real not null, pose_y real not null , ori_z REAL, ori_w REAL)
 table_number	integer	"table_number" integer NOT NULL
 pose_x	real	"pose_x" real NOT NULL
 pose_y	real	"pose_y" real NOT NULL
 ori_z	REAL	"ori_z" REAL
 ori_w	REAL	"ori_w" REAL

# Implementation

## SQLite

Table: menu\_order

	_number	order_time	menu_list	table_number
		Filter	Filter	Filter
1	2411291	241129 011249	치즈라볶이, 치즈라볶이, 짜빠구리, 풀지마라탕	3
2	2411292	241129 095529	풀지마라탕, 풀지마라탕, 풀지마라탕, 치즈라볶이, 볶...	3
3	2411293	241129 100855	치즈라볶이, 치즈라볶이, 치즈라볶이, 치즈라볶이	3
4	2411294	241129 101312	치즈라볶이, 치즈라볶이	3
5	2411295	241129 101641	얼큰수제비, 짜빠구리, 볶음세트, 불짬뽕탕, 어묵탕	3
6	2411296	241129 102448	치즈라볶이, 치즈라볶이, 짜빠구리, 짜빠구리	3
7	2411297	241129 102649	치즈라볶이, 풀지마라탕, 참이슬, 참이슬	3
8	2411298	241129 102840	치즈라볶이, 짜빠구리, 콜라, 새로, 새로	3
9	2411299	241129 150007	풀지마라탕, 풀지마라탕, 풀지마라탕, 풀지마라탕, 풀지...	3
10	2412011	241201 002138	얼큰수제비, 새로, 참이슬, 오리지널	3
11	2412012	241201 010502	얼큰수제비, 짜빠구리, 볶음세트, 어묵탕	3
12	2412021	241202 100124	치즈라볶이, 치즈라볶이	3
13	2412022	241202 100128	풀지마라탕	3
14	2412023	241202 100310	풀지마라탕, 치즈라볶이	3
15	2412024	241202 100329	풀지마라탕, 얼큰수제비	3

Table: moving

	start_time	end_time
	Filter	Filter
1	2024-12-02 10:40:20	2024-12-02 10:40:33.329077
2	2024-12-02 10:56:00	2024-12-02 10:56:25.805070
3	2024-12-02 10:56:48	2024-12-02 10:57:14.070564
4	2024-12-02 10:59:37	2024-12-02 11:00:21.241394
5	2024-12-02 11:05:14	2024-12-02 11:05:46.987753
6	2024-12-02 11:06:08	2024-12-02 11:06:50.063457


Table: pose

	table_number	pose_x	pose_y	ori_z	ori_w
	Filter	Filter	Filter	Filter	Filter
1	1	1.277	1.605	0.999	0.026
2	2	1.299	0.489	-0.999	0.024
3	3	1.312	-0.488	-0.666	0.746
4	4	2.481	1.654	-0.993	0.117
5	5	2.456	0.515	-0.999	0.018
6	6	2.41	-0.623	-0.999	0.01
7	7	3.65	1.614	-0.999	0.059
8	8	3.913	0.512	0.999	0.023
9	9	3.755	-0.668	-0.999	0.008

Table: image\_file

	name	img
	Filter	Filter
1	배달크림파스타	BLOB
2	모듬수제비	BLOB
3	치킨	BLOB
4	순살치킨	BLOB
5	참이슬	BLOB
6	크리시	BLOB
7	황도	BLOB
8	채로콜라	BLOB
9	원장술밥	BLOB
10	편의점인쇄물	BLOB
11	감자튀김	BLOB
12	편다포도	BLOB
13	짜빠구리	BLOB
14	자율행동	BLOB
15	매운오명탕	BLOB
16	과일차	BLOB
17	어묵탕	BLOB
18	닭발	BLOB
19	오명탕	BLOB

Mode: Image



# Implementation

## SQLite Code Review

### 주문 정보를 데이터에 삽입하기

```
"""주문 정보 삽입"""
global today_count, today_day, cursor, conn
menu_string = ",".join(menu_list)
today_count += 1
order_number = int(today_day+str(today_count))
table_number = request.table_num
order_time = datetime.now().strftime("%y%m%d %H%M%S")

query = f"INSERT INTO menu_order VALUES ({order_number},'{order_time}','{menu_string}',{table_number})"
cursor.execute(query)
conn.commit()
self.get_logger().info("order info insert")
"""===== """
```

# Implementation

## SQLite Code Review

### 데이터를 활용한 통계

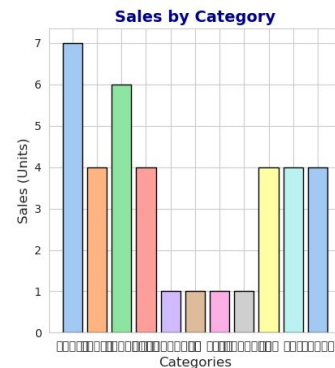
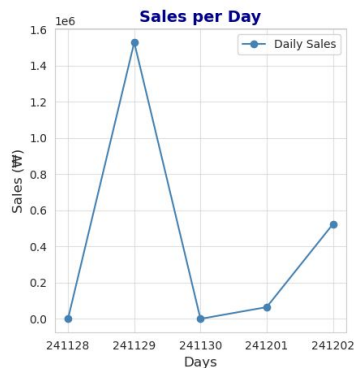
```
def cal_per_day(self):  
    global cursor  
  
    total_prices = []  
    today_dates = []  
  
    for i in range(5):  
        today = (datetime.now() + timedelta(days=-i)).strftime("%y%m%d")  
        geury = f"SELECT * FROM menu_order WHERE order_number LIKE '{today}%'"  
        cursor.execute(geury)  
        orders = cursor.fetchall()  
        today_dates.append(today)  
        if len(orders) == 0:  
            total_prices.append(0)
```

```
        else:  
            sum = 0  
            for order in orders:  
                menu_list = order[2]  
                menu_list = menu_list.split(',')  
  
                for menu in menu_list:  
                    geury = f"SELECT price FROM menu WHERE name='{menu}'"  
                    cursor.execute(geury)  
                    price = cursor.fetchone()[0]  
                    sum += price  
            total_prices.append(sum)  
  
    total_prices.reverse()  
    today_dates.reverse()  
  
    return total_prices, today_dates
```

# Implementation

## Statistics

정산 내역			
	테이블	메뉴	가격
1	3	무구리	70000
2	3	무구리	70000
3	3	폴지마라탕	21000
4	3	파인샤베트	6000
5	3	제로콜라	2000
6	3	참이슬	45000
7	3	참이슬	45000
8	3	참이슬	45000
9	3	참이슬	45000
10	3	참이슬	45000
11	3	참이슬	45000
12	3	참이슬	45000
13	3	참이슬	45000
14	3	참이슬	45000
15	3	참이슬	45000
16	6	얼큰수제비	11000
17	6	참이슬	4500
18	6	황도	7000
19	2	통통가...김...	19000
20	2	환타...도	2000
21	9	참이슬	31500
22	9	참이슬	31500
23	9	참이슬	31500
24	9	참이슬	31500
25	9	참이슬	31500
26	9	참이슬	31500
총계: 1010500 원			
그래프 보기			





# Implementation

.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():

    return LaunchDescription(
        [Node(
            package="serving_node",
            executable="order"
        ),
        Node(
            package="serving_node",
            executable="display"
        ),
        Node(
            package="serving_node",
            executable="robot"
        ),
    ]
)
```

# | Implementation

## Logging

현재 로봇의 포지션 (debug)

```
def amcl_pose_callback(self,msg):  
    pose_msg = msg  
    self.x = pose_msg.pose.pose.position.x  
    self.y = pose_msg.pose.pose.position.y  
    self.get_logger().debug(f"{self.x},{self.y}")
```

# Implementation

## Logging

### 로봇의 이동 상태 (info)

```
if "Goal succeeded" in msg.msg:
    self.get_logger().info("arrived goal")

    if abs(self.x) < 0.5 and abs(self.y) < 0.5:
        self.get_logger().info("arrived initial pose")
    else:
        self.get_logger().info("arrived table")
        self.send_goal_status(True)
```

```
def response_callback(self, future):
    if future.result():
        response = future.result()
        if response.get_back:
            self.get_logger().info("go to initial pose")
            self.to_initial_pose()
```

### 주문 정보 확인 (info)

```
def process_order_callback(self, request, response):
    self.get_logger().info(f"Received order: Table {request.table_num},
                           menu: {request.menu}, Total: {request.total_price}")
```

```
query = f"INSERT INTO menu_order VALUES ({order_id}, {request.menu})"
cursor.execute(query)
conn.commit()
self.get_logger().info("order info insert")
"""====="""
```

# Implementation

## Logging

### 서버의 on/off (warn)

```
while not self.client.wait_for_service(timeout_sec = 1.0):  
    self.get_logger().warn('Waiting for order...')
```

```
while not self.client.wait_for_service(timeout_sec=1.0):  
    self.get_logger().warn('Waiting for Table Order Serve
```

### 통신의 상태 (error)

```
f future.result():  
    response = future.result()  
    if response.get_back:  
        self.get_logger().info("go to initial pose")  
        self.to_initial_pose()  
    else:  
        self.get_logger().error('Service call failed: response
```

# | Implementation

## Qos

### TurtleBot Position Qos

```
qos_profile_pose = QoSProfile(  
    history=QoSHistoryPolicy.KEEP_LAST, depth=10,  
    reliability=QoSReliabilityPolicy.BEST_EFFORT,  
    durability=QoSDurabilityPolicy.VOLATILE,  
)
```

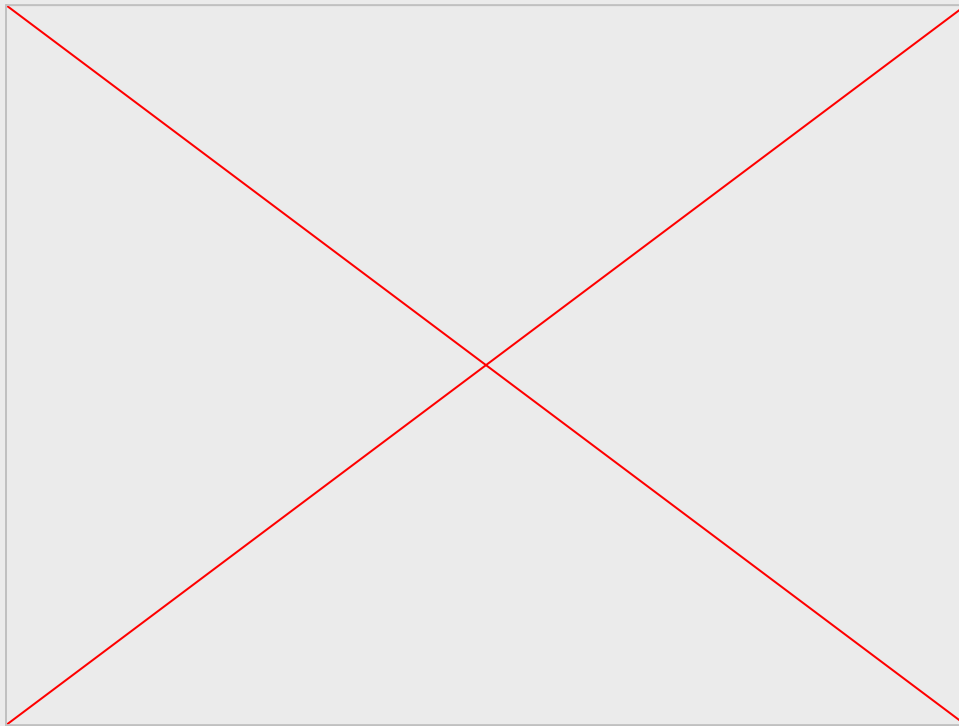
### Publish position Qos

```
qos_profile_pub_pose = QoSProfile(  
    history=QoSHistoryPolicy.KEEP_LAST, depth=10,  
    reliability=QoSReliabilityPolicy.RELIABLE,  
    durability=QoSDurabilityPolicy.VOLATILE  
)
```

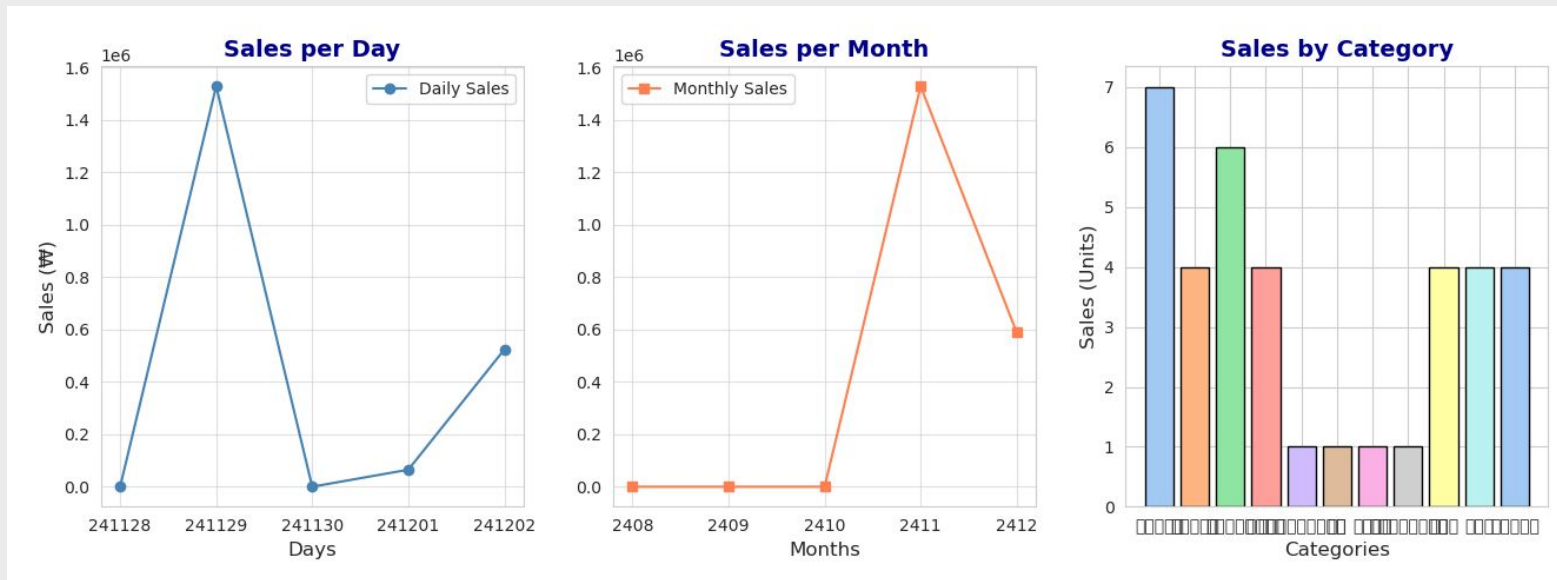
### Order Service Qos

```
qos_order = QoSProfile(  
    history=QoSHistoryPolicy.KEEP_ALL,  
    reliability=QoSReliabilityPolicy.RELIABLE,  
    durability=QoSDurabilityPolicy.VOLATILE,  
)
```

# | Result



# Result



# | Conclusion & Future Work

## Conclusion

- 테이블과 주방 디스플레이를 통해 로봇을 제어하는 로직을 완성
- 시뮬레이션으로 구현

## Future

- 실제 터틀봇에 반영하여, 현실에서 서비스될 수 있도록 구현하는 것이 목표



# Conclusion

Git으로 협력하기 ? [https://github.com/yms0606/turtlebot3\\_servingRobot](https://github.com/yms0606/turtlebot3_servingRobot)

yms0606 / turtlebot3\_servingRobot

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

turtlebot3\_servingRobot Public

main 6 Branches 0 Tags

ksj5398 Merge pull request #56 from yms0606/kimseungjoong 8f5abd2 · 45 minutes ago 103 Commits

src	Merge pull request #56 from yms0606/kimseungjoong	45 minutes ago
.gitignore	modify ignore file	3 days ago
README.md	junha JANG	5 days ago
ServingRobotDB.db	hihi	3 hours ago
insert_data.py	update DB	3 days ago
test.py	DB create, package create	5 days ago

README

2024.11.26 minsik YOON

2024.11.26 seungjoong KIM

2024.11.26 seunghwan JEONG

2024.11.26 junha JANG

Commits

main

Commits on Dec 2, 2024

Merge pull request #56 from yms0606/kimseungjoong	Verified	8f5abd2	<>
ksj5398 authored 47 minutes ago			
to teacher		f2a5ffa	<>
ksj5398 committed 48 minutes ago			
Merge pull request #55 from yms0606/yms	Verified	418eebd	<>
yms0606 authored 3 hours ago			
qos update2		7c8579c	<>
yms0606 committed 3 hours ago			
qos update2		dd723fc	<>
yms0606 committed 3 hours ago			
Merge pull request #53 from yms0606/kimseungjoong	Verified	86ed99e	<>
ksj5398 authored 3 hours ago			
hihi		62f9944	<>
ksj5398 committed 3 hours ago			
Merge pull request #52 from yms0606/kimseungjoong	Verified	f28e961	<>
ksj5398 authored 4 hours ago			
ddd		be926b2	<>
ksj5398 committed 5 hours ago			
Merge pull request #51 from yms0606/yms	Verified	1c3196b	<>
yms0606 authored 14 hours ago			
qos update		e2efdc6	<>
yms0606 committed 14 hours ago			

# Q&A

---

감사합니다