안녕하세요 E-3조 발표를 맡게 된 남승록이라고 합니다.

이번주 E그룹에 발표주제는 주방 디스플레이를 통한 서빙 로봇제어및 서빙로봇 작동 구현이였습니다.
처음에 저희조는 너무 뻔한 것을 하려고 했습니다.짜장면집가게 같은거 하려했는데 그런것보다 로봇이 아직 상용화되지 않은곳을 생각해보았습니다. 그러면서 어떤 가게를 해야 차별점이 있을지 참 많은 고민을 하였습니다.
그러던 와중에 피시방이 무인화가 점점 되는것을 이야기를 나누다가 미래에는 PC방에도 음식을 가져다 줄때에는 서빙 로봇이 가져다 주지 않을까?라는 생각을 해서 PC방 서빙 로봇을 구축하자라는 목표를 가지게 되었습니다.

먼저 저희의 컨텐츠와 발표를 하기전에 저희의 역할분담부터 이야기하겠습니다.
우선 저는 저희가 너무 나무만 보지 않게하는것이 중요하다고 생각했기에 전체적인 큰틀을 계속해서 상기시켰으며 주방 디스플레이와 로봇 연동을 했으며 지금 이렇게 발표를 하고 있습니다.
송현님은 MYSQL기반으로 데이터 베이스를 담당하셨고 주문 디스플레이 통계 표현을 구현하셨습니다.
그리고 정말 큰 역할 중 하나인 UI 및 데이터 시각화를 이쁘게 담당해주셨습니다.
재현이는 노드 개발 및 로봇이동을 제어했으며 두 팀원간의 역할을 끊임없이 상기시켜주었으며 테이블 오더와 주방 디스플레이를 연동시키는 역할을 맡았습니다.
이제 저희의 컨텐츠 순서를 말씀드리겠습니다.

발표때 1~8다 읽으며 이 순서대로 진행할것이며 늦었지만 떨어도 너그러운 마음으로 발표를 들어주신다면 감사하겠습니다.

먼저 서빙 로봇이 작동되는것을 시현하기 전에 저희가 이렇게 구현했다는것을 먼저 사진으로 보여드리겠습니다. (사진을 다 보여준후) 역시 백문의 불후일견이라고 직접 보여드리는게 좋겠죠? (보여준후) 저희는 이렇게 서빙 로봇제어 및 서빙로봇 작동을 구현하였습니다.

또한 여기서 터미널의 로그들을 보여주면서
logging부분에서는 debug ,warning,info를 각각 설명하겠습니다. (그리고 피피티 읽기)
1. debug 로그 : 시스템이 각 이미지를 성공적으로 불러왔는지 확인
*Loading image: /home/seungrok/pc_test1/pc_test/image/불닭볶음면.png
*Loading image: /home/seungrok/pc_test1/pc_test/image/짜계치.png

2. Warning
출현 위치: ros2 run pc_test kitchen 및 ros2 run pc_test pc_order 실행 시
* Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
--> 실행 결과에 영향을 미치지는 않으며 환경 설정과 관련된 경고

*ros2 run pc_test pc_order 실행 시 로그중
[WARNING] [order_publisher]: Special menu stock low: 삼겹살 정식 remaining 0
--> Special menu stock low: 삼겹살 정식의 재고가 부족하다는 경고 메시지.
#그닥 치명적이지 않은 경우.

3. info 로그 : 시스템의 상태, 명령 실행, 작업 성공 등을 알리는 일반적인 정보 제공
[INFO] Initial pose set to kitchen successfully.
초기 위치를 주방으로 설정했다는 메시지로, TurtleBot3의 시작 위치를 확인 가능.

[INFO] Received order: {'table_number': 6, 'cart_items': {'짜계치': [3000, 3]}, 'total_price': 9000}
테이블 6번에서 짜계치 3개를 주문받은 로그.

요리 완료 버튼 눌림: 테이블 6
UI에서 "요리 완료" 버튼 클릭 시 출력.

[INFO] Sending navigation goal: (-0.2644, -1.0853)
TurtleBot3가 테이블 6번으로 이동할 목표 좌표를 설정.

[INFO] Goal accepted, waiting for result...
목표가 수락되어 이동을 시작했다는 로그.

[INFO] Goal succeeded!
테이블 6번에 성공적으로 도착.

[INFO] Returning to kitchen...
주방으로 복귀를 시작한다는 메시지.

[INFO] Sending navigation goal: (-2.0712, -0.4722)
주방 좌표로 설정된 목표 위치를 출력.

[INFO] Goal accepted, waiting for result...
복귀 목표가 수락되어 이동 중.

[INFO] Goal succeeded!
TurtleBot3가 주방으로 성공적으로 복귀.

시현영상에서 보신것처럼 저희는 한정판 메뉴를 넣었습니다.
또한 저희는 로봇상태를 확인할수 있습니다. 로봇이 가기전에 대기중이 뜨고 서빙을 할때에는 서빙중이뜨고, 테이블 서빙이 완료되면 서빙이 완료되었다고 뜹니다~ 다른 차별점은 데이터 베이스를 활용한 통계입니다. 월의 총매출 그리고 하루의 매출을 보실수 있습니다. 또한 인기메뉴를 보실수 있습니다~
아까 저희의 시현영상을 잘보셨다면 아까와 다른점을 보실수 있으실거에요. 지금 사진에 있는것이 어떤 값을 수정하기전입니다. 한번 맞춰보십쇼.
저희는 로봇이 각 테이블로 이동 중에 로봇이 영역에 갇혀서 움직이지 못하는 상황이 발생하였습니다. 이를 해결하기 위해 각 테이블의 좌표를 바꾸려고 했으나 다른 방법이 있을까 생각하다보니

turtllebot3_ws의 navigation의 burger.yaml의 global_costmap에 있는 inflation_radius의 값을 줄여서 로봇의 이동이 원활하도록 하였다. inflation_radius라는 값을 고려해서 로봇이 이 값의 영역을 피하려고 경로를 계획하는데 처음의 값으로 했을 때 rviz상의 분홍색 부분의 영역이 넓어서 이를 줄임으로서 로봇의 이동이 훨씬 수월하게 되었다.

이제 데이터베이스는 SQL로 구현하였는데요~

이 데이터베이스를 통해 PC방에서 주문과 서빙을 자동화할 수 있습니다.
특히, 매출 분석 기능을 활용하면 어떤 메뉴가 잘 팔리는지 파악하여 효과적인 운영이 가능합니다.

> "**먼저 orders 테이블을 보시면**, 고객이 음식을 주문할 때 생성되는 데이터를 저장하는 테이블입니다.
> 고객이 주문을 하면, **주문 번호(id), 주문한 테이블 번호(table_number), 메뉴(menu_item), 가격(price), 수량(quantity)** 등이 저장됩니다.
> 또한, timestamp 컬럼을 통해 **언제 주문했는지 기록**할 수 있습니다."

(👉 **ERD에서 orders 테이블을 강조하며 설명**)

"**이 테이블은 매출 분석을 위해 매우 중요한 역할을 합니다.**

예를 들어, 특정 날짜의 매출을 알고 싶다면 orders 테이블에서 해당 날짜의 주문 데이터를 조회하면 됩니다."

> "**다음으로 menu 테이블을 보겠습니다.**
> 이 테이블은 **PC방에서 판매하는 음식의 정보를 저장**합니다.
> 각 메뉴는 menu_id(메뉴 고유 ID)를 가지고 있으며, menu_item(음식 이름)과 price(가격) 정보가 저장됩니다.
> 예를 들어, '해장라면'의 가격이 3000원이면, menu 테이블에 '해장라면'과 3000원이 기록됩니다."

(👉 **ERD에서 menu 테이블을 강조하며 설명**)

"**주문할 때는 menu 테이블에서 가격을 자동으로 가져오게 됩니다.**

즉, orders 테이블에서 주문을 입력할 때, 메뉴의 가격 정보는 이 menu 테이블에서 불러오는 방식입니다.

> **세 번째는 sales 테이블입니다.**
> 이 테이블은 **일별 매출 데이터를 저장**합니다.
> sale_date 컬럼에는 매출 날짜가 저장되며, total_sales 컬럼에는 해당 날짜의 총 매출이 기록됩니다.
> 이 테이블을 활용하면 **월별, 연도별 매출 분석이 가능합니다.**"

(👉 **ERD에서 sales 테이블을 강조하며 설명**)

"예를 들어, 2025년 1월 한 달 동안의 총 매출을 알고 싶다면 sales 테이블을 조회하면 됩니다."

마지막으로 tables 테이블을 보겠습니다.
tables 테이블은 **PC방 내 테이블의 위치 정보를 저장하는 테이블입니다.**
table_id(테이블 번호), x_position, y_position 컬럼이 있으며, 이 좌표 데이터를 사용하여 **서빙 로봇이 특정 테이블로 이동할 수 있습니다.**"

(👉 **ERD에서 tables 테이블을 강조하며 설명**)

"예를 들어, 5번 테이블에 주문이 들어오면, 서빙 로봇은 tables 테이블에서 5번 테이블의 좌표를 가져와 해당 위치로 이동합니다."

2:39

SELECT x_position, y_position FROM tables WHERE table_id = 5;

2:40

"이제 데이터가 어떻게 흐르는지 설명드리겠습니다.

1️⃣ 고객이 PC방에서 음식을 주문하면, `orders` 테이블에 새로운 주문이 추가됩니다.

2️⃣ `menu` 테이블에서 해당 음식의 가격을 불러와 `orders` 테이블에 저장합니다.

3️⃣ 주문이 완료되면, `sales` 테이블에서 해당 날짜의 총 매출이 업데이트됩니다.

4️⃣ 주문이 완료되면, `tables` 테이블에서 좌표 정보를 가져와 서빙 로봇이 이동하게 됩니다."

(👉 **ERD 다이어그램의 연결선을 따라 손으로 가리키며 설명**)

"이렇게 모든 데이터가 하나의 흐름을 따라가면서 시스템이 자동으로 작동하도록 설계되었습니다."

이제 저희가 이 과정을 진행했는지 애기해보겠습니다. 어떻게 보면 전체적인 개요? 또는 목표를 완성하는데 있어 경험했던 시행착오를 보여주는거가 되겠네요. ‘
처음에는 사진과 같이 어떻게 로봇이 전체적인 큰틀을 그려보고 rviz를 켜서 테이블의 번호를 정했고 이 테이블의 번호를 정하고 나니 주방과 각각의 좌표가 필요해 ros2 topic echo /clicked_point을 터미널에 작성하고 rviz에서 publish point를 통해 각각의 위치에 대한 좌표정보를 알게되었습니다. 하지만 이렇게 순조롭게 잘 진행된다면 먼가 이상했죠. 우선 저희가 봉착한 문제점은 로봇과 주방디스플레이를 연동시키는 파일을 한개 더 만들었냐였습니다. 여기서 저희는 비대면때 배운것들이 무엇이 있을까?라는 생각을 하다가 hang-man 구조도를 보게 되었는데요. 여기서 action server가 저희의 kitchen과 유사했으며 이러한 kitchen을 중앙 관제탑처럼 설계를 해야겠다는 생각을 하게 되었습니다. 따라서 고객과 주방간의 통신을 비동기적이고 발행-구독 기반의 통신을 제공하는 토픽으로 결정하였고 이것은 고객(테이블 주문 시스템)에서 발생하는 실시간 데이터(주문 정보)를 주방으로 전송하는데 적합했습니다. 예를 들어 order_topic이라는 토픽을 통해 주문 테이블 번호와 주문한 메뉴를

주방으로 전달하는것이 그 예가 되겠습니다. 또한 주방은 토픽의 구독자로 동작하며, 고객이 발행한 주문 데이터를 실시간으로 받아 처리하는것또한 토픽을 사용했기에 가능했던거 같습니다.
다음은 제가 말로 설명한것을 가시화해서 나타낸 자료입니다.
또한 주방과 로봇간의 통신은 주방에서 주문 데이터를 처리하고,로봇에게 특정 목표 위치 즉 goal pose 로 이동 명령을 전달한 뒤 ,해당 작업의 완료 상태를 주방으로 반환하는 구조이기에 즉 피드백이 필요했기에 액션으로 구현하였습니다. 다음은 제가 말로 설명한것에 대한 가시화입니다.
만약에 전에 배웠던 것을 생각하지 않고 참고하지 않으려고 했다면 정말 먼길을 돌아갈뻔했지만 전에 배웠던것을 바탕으로 응용을 했던부분이 정말로 다행이였던거 같습니다.

5. 메시지 인터페이스는 그냥 피피티 읽고

6. qos는 그냥 피피티 읽기

7. 토픽은 아까도 애기했지만 더 설명하고 싶어서 넣었습니다. 토픽은 비동기적 메시지 전달을 통해 주문 데이터와 위치 명령 데이터를 효율적으로 처리하는것인데 저희는 대표적으로 (위의 피피티 내용 읽기)
*서비스는 그냥 피피티 쳐읽기
* 액션은 아까도 말씀드린것처럼 피드백이 중요한 부분에서 구현을 해야합니다. (그리고 위의 피피티 내용 읽기)