

미래내일청년

음성대화 ChatGPT 로봇

목차 소개

1. 프로젝트 목표 및 기대효과
2. 팀원 소개
3. 프로젝트 수행일정
4. 프로젝트 초기 예산안
5. 프로젝트 진행과정, 코드리뷰
6. 문제점과 해결과정
7. 프로젝트 결과
8. 향후 도전 과제

프로젝트 목표

터틀봇으로 뭘
할 수 있을까?



TurtleBot3 + ROS2 + LLM

주요 기능

- 음성 인식
- 음성 대화
- 감정 분석
- 로봇 이동



프로젝트 향후 기대 효과

EY한영 2024년 경제 전망 설문조사

귀사의 AI도입 현황은 어떠합니까?



■ 전사적 도입 및 적용 중 ■ 일부 도입 및 활용 중
■ 도입할 계획 있음 ■ 도입 계획 없음

출처 : <https://www.ciokorea.com/news/325632>

다양한 분야에서 자동화와 인공지능을 활용한 혁신적인 솔루션 수요 증가, 로봇 기술의 고도화로 로봇과 사용자와의 상호작용을 강화하는 기능 필수적

음성인식 챗봇 : 사회적 약자나 특정 요구를 가진 사용자들에게 실질적인 도움을 주어 사회적 기여를 수 있습니다.

예)

취업 및 면접 지원 로봇 (AI 면접관)
요리 보조 로봇 (레시피 제공, 타이머 기능 등)
스마트 상점에서 고객 지원 로봇
치매 환자와의 상호작용 및 지원 로봇




팀원 소개 및 역할 분담



팀 장	최용한	프로젝트 일정 관리, ppt제작, 프로젝트 기획 및 설계, 시스템 통합
팀 원1	박도엽	ros2 노드(class) 설계 및 코드 작성, launch파일 작성, chatGPT연동
팀 원2	정태양	라즈베리파이 마이크/스피커 설치 및 연동, 개발환경 설정, 하드웨어 구성
팀 원3	유재영	openai api 모델 연동, 음성 <-> 텍스트 변환 및 텍스트 생성 코드 작성

프로젝트 수행일정



구분	추진 내용	추진 일정			
		1주차	2주차	3주차	4주차
도입	프로젝트 계획 및 검토				
계획	역할 분담 및 협업 톨 셋팅				
실행	필요 자재품, openAI 결제				
	ROS2 노드 설계, openAI api 적용				
	결과 보고서, ppt 초안 작성				
최종 마무리	결과 보고서, 발표 ppt 최종 검토				
오프라인 미팅계획	매 주 1~2회 대면 회의 진행				



프로젝트 초기 예산안

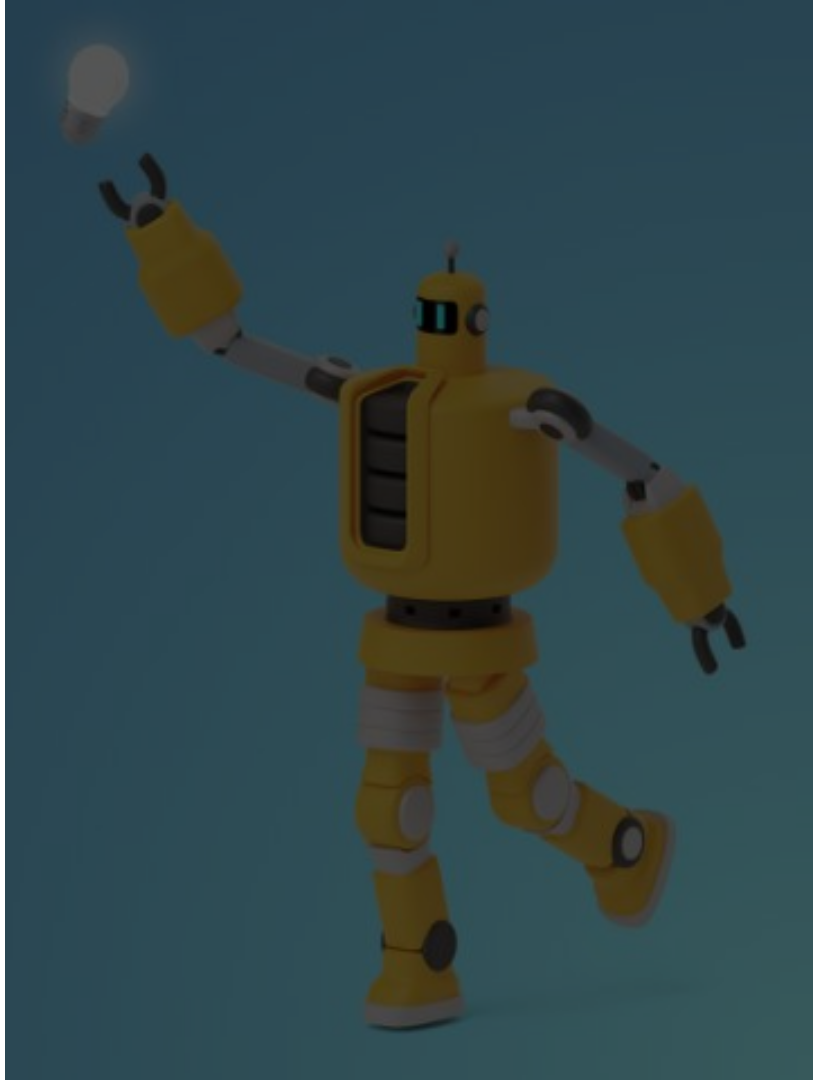
- OpenAI 모델 (chatgpt4o api, whisper, tts) 사용료 : 100,000 내외 (70\$)
- [Raspberry Pi] 라즈베리파이 공식 7인치 터치스크린 (Raspberry Pi Touch Display) : 93,500
- [Raspberry Pi] 라즈베리파이 5 FPC 디스플레이 케이블 200mm - 22P(0.5mm) to 15P(1mm) : 1,540
- [KEYES] 테스트[CH254] 소켓 점퍼 케이블 40P (칼라) (M/F) 20cm : 935
- 하이크세미 마이크로 SD카드 NEO LUX 256GB : 33,000
- 컴스 회의실용 원터치 콘덴서 마이크 : 33,240
- [Raspberry Pi] 라즈베리파이 5 (Raspberry Pi 5) 8GB + 가이드북 : 117,700

사용 기술

ROS2는 로봇 애플리케이션 개발, 통합, 배포에 최적화된 오픈 소스 프레임워크로, 실시간 성능, 분산 시스템 등 하드웨어와 소프트웨어 환경에 대응할 수 있어 로봇 프로젝트에서 많이 사용된다.

ROS2 노드를 구현하여 로봇과의 상호작용(음성)을 통하여 음성 대화/이동 기능을 수행 할 수 있게 하고 패키지 관리와 빌드를 진행 하였습니다.

ROS 2TM



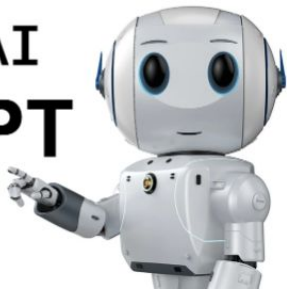
사용 기술

OpenAI는 인공지능 (AI)의 개발과 연구를 주도하는 기관으로, 저희 팀은 프로젝트 진행을 위해 **OpenAI API**를 사용하였습니다. **API**는 개발자들이 **OpenAI**가 만든 인공지능 모델(**GPT**, **DALL-E**, **Codex** 등)을 자신의 애플리케이션, 제품, 서비스에 통합할 수 있도록 제공하는 인터페이스로, 아래와 같은 3가지 **API** 사용

- 사용된 API 모델

- 1) **GPT-4** : 자연어 생성, 대화형 AI 작업에 매우 유용한 최신 모델
- 2) **Whisper** : 음성 인식과 텍스트 전환을 지원하는 모델입니다.
음성 데이터를 텍스트로 변환하는데 활용
- 3) **TTS** : 텍스트를 음성데이터로 변환, 56개 언어 지원, 6가지 목소리 출력 형태

 OpenAI
ChatGPT



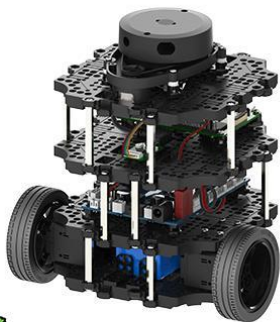
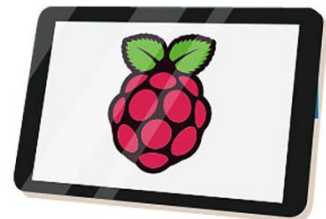
사용 기술

- 1) **ROS2** - 최신 로봇 운영체제 플랫폼으로, 실시간 통신 및 멀티로봇 시스템 개발 지원
- 2) **OpenAI api (chatgpt4o, whisper, tts)** - 사용자의 음성을 텍스트로 변환, 텍스트를 사용자의 음성으로 변환하는 기술, 사용자의 입력에 대한 답변을 생성하고 이를 감정분석 하는 기능도 지원함
- 3) **Python** - 노드 개발과 로봇 제어를 위한 프로그래밍 언어
- 4) **C++** - 노드 개발과 로봇 제어를 위한 프로그래밍 언어
- 5) **TurtleBot3** - 기본 하드웨어로 사용한 로봇
- 6) **Raspberrypi** - 싱글보드 SBC
- 7) **Linux Ubuntu** - Ubuntu 22.04, ROS2 개발환경



프로젝트 구현

TURTLEBOT3 Burger



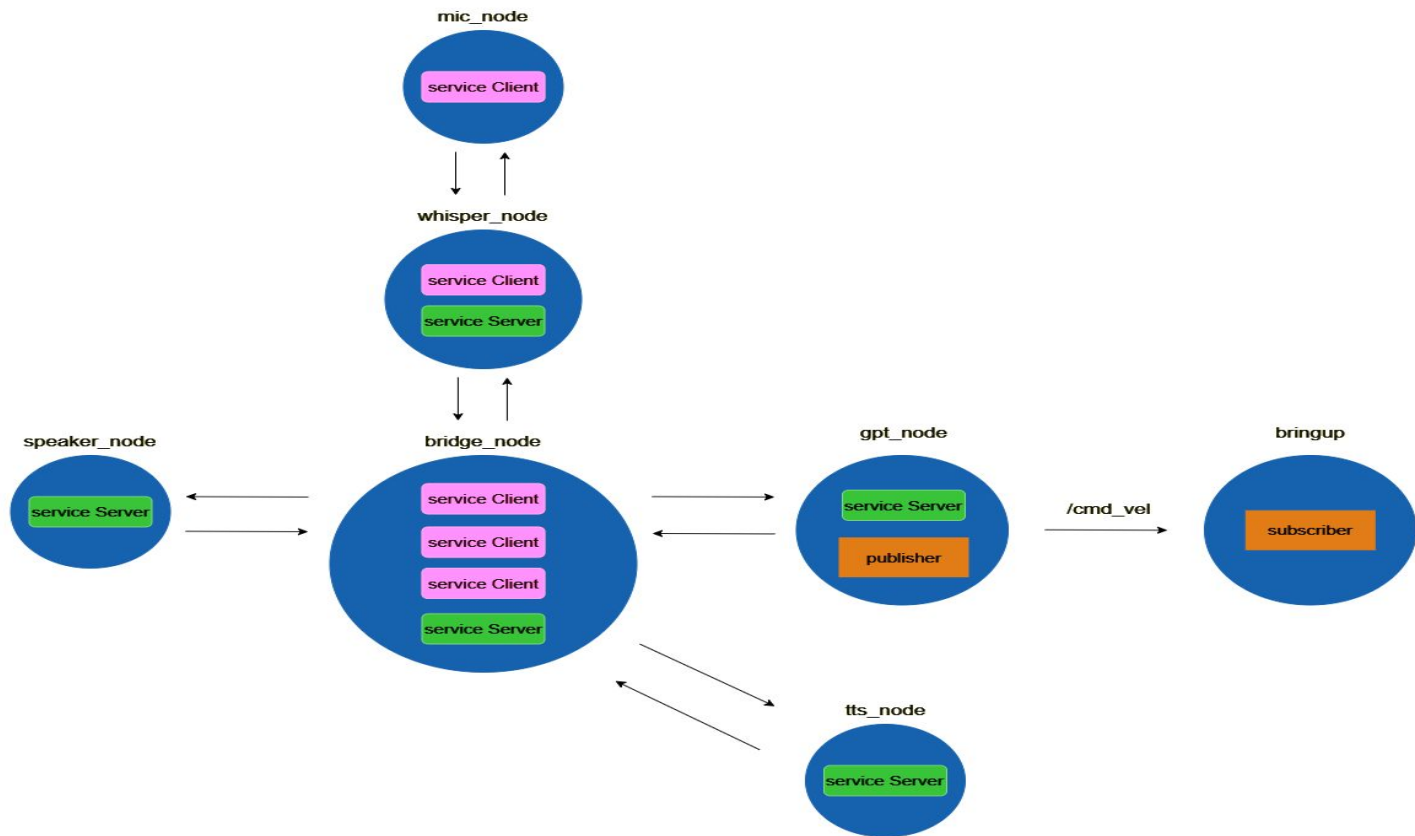
Raspberry Pi® 4 (2GB)



요구사항 분석

요구사항명	기능명	설명
하드웨어	TurtleBot3	물리적 이동 및 제어를 위해 사용
	마이크	사용자의 음성 입력을 수신하기 위해 사용
	스피커	음성 응답 출력을 위해 사용
소프트웨어	음성인식 기능	사용자의 음성 입력을 마이크로 수신해야 한다. 음성 입력을 Whisper API를 이용하여 텍스트로 변환해야 한다.
	자연어처리 기능	텍스트로 변환한 음성 데이터를 GPT API에 전송하여 적절한 답변을 생성한다.
	음성출력 기능	TTS API를 이용하여 텍스트를 음성 데이터로 변환한다. 스피커를 통해 음성으로 변환된 응답을 사용자에게 제공한다.
	로봇움직임 제어	음성 명령을 미리 학습시킨다. 음성 명령에 대한 이동 동작을 로봇이 수행한다.

프로젝트 설계



코드리뷰 - Whisper

```
void Whisper::openai_whisper()
{
    openai::start();
    auto transcription = openai::audio().transcribe(R"(
        {
            "file": "/home/ubuntu/robot_ws/src/FutureTomorrowExperience_R0S2/openai_cpp_pkg/audio/question.mp3",
            "model": "whisper-1",
            "prompt": "question"
        }
    )"_json);

    _question_msg = transcription["text"];
}
```

whisper 코드 : 사용자 음성 -> Text로 변환

코드리뷰

```
class Bridge : public rclcpp::Node {
private:
    rclcpp::Service<openai_interface::srv::QaInterface>::_SharedPtr _service_whisper; //whisper에 request에 response
    rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr _client_gpt;
    rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr _client_tts;
    rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr _client_speaker;

    std::string _gpt_question;
    std::string _gpt_answer;
    std::string _tts_answer;
    //bool _status;

    void send_request_gpt();
    void response_callback_gpt_thread(rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr future);
    void response_callback_gpt(rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr future);
    void send_request_tts();
    void response_callback_tts(rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr future);
    void send_request_speaker();
    void response_callback_speaker(rclcpp::Client<openai_interface::srv::QaInterface>::_SharedPtr future);
    void callback_whisper(const openai_interface::srv::QaInterface::Request::SharedPtr request,
        openai_interface::srv::QaInterface::Response::SharedPtr response);
```

bridge 코드 : openAI 노드들을 연결 해주는 역할

코드리뷰 - GPT

```
void Gpt::gpt(const std::string str)
{
    nlohmann::json jsonObj = R"(
    {
        "model": "gpt-4o",
        "max_tokens": 1000,
        "temperature": 0
    }
    )"_json;

    jsonObj["messages"].push_back({
        {"role", "user"},
        {"content", str}
    });

    openai::start();
    auto chat = openai::chat().create(jsonObj); // API 호출의 반환 값 확인 필요

    // API 결과를 JSON으로 파싱
    try {
        std::string str = chat.dump();
        nlohmann::json json_result = nlohmann::json::parse(str); // chat이 JSON 문자열인지 확인 필요
        std::string content = json_result["choices"][0]["message"]["content"];
        std::cout << "content: " << content << std::endl;
        _answer = content;
    } catch (const std::exception& e) {
        std::cerr << "JSON 파싱 에러: " << e.what() << std::endl;
    }
}
```


코드리뷰 - TTS

```
def tts_answer(self):  
    print("tts start22")  
    self.client = OpenAI()  
    speech_file_path = Path("/home/ubuntu/robot_ws/src/FutureTomorrowExperience_R0S2/result_audio/answer.wav")  
    response = self.client.audio.speech.create(  
        model="tts-1",  
        voice="alloy",  
        input=self.answer  
    )  
    response.stream_to_file(speech_file_path)  
    print("audio done")
```

tts 코드 : GPT답변 (Text) -> 음성으로 변환

코드리뷰 - 파인튜닝

```
{ "role": "user", "content": "뒤로 가", { "role": "assistant", "content": "category: movement, parameters: {x: -0.1}" } }  
{ "role": "user", "content": "뒤로 이동", { "role": "assistant", "content": "category: movement, parameters: {x: -0.1}" } }  
{ "role": "user", "content": "후진", { "role": "assistant", "content": "category: movement, parameters: {x: -0.1}" } }  
{ "role": "user", "content": "앞으로 가", { "role": "assistant", "content": "category: movement, parameters: {x: 0.1}" } }  
{ "role": "user", "content": "앞으로 이동", { "role": "assistant", "content": "category: movement, parameters: {x: 0.1}" } }  
{ "role": "user", "content": "전진", { "role": "assistant", "content": "category: movement, parameters: {x: 0.1}" } }
```

MODEL

ft:gpt-3.5-turbo-0125:personal::A0HYXmGw

○ Status

✔ Succeeded Click to copy

ⓘ Job ID

ftjob-kvhbNGpp1wHPchUBEXkK9TnW

📦 Base model

gpt-3.5-turbo-0125

📦 Output model

ft:gpt-3.5-turbo-0125:personal::A0HYXmGw

🕒 Created at

2024년 8월 26일 오전 9:00

코드리뷰 - 파인튜닝

```
client = OpenAI()

try :
    response = client.files.create(
        file=open("../data/mydata.jsonl", "rb"),
        purpose="fine-tune"
    )

    file_id = response.id

    print(f"Uploaded File ID: {file_id}")
    response1 = client.fine_tuning.jobs.create(
        training_file=file_id,
        model="gpt-3.5-turbo"
    )

    fine_tune_id = response1.id
    client.fine_tuning.jobs.list(limit=10)
```

```
while True :
    response2 = client.fine_tuning.jobs.retrieve(fine_tune_id)
    print(response2.status, "\n")

    if response2.status == "succeeded":
        fine_tuned_model_id = response2.fine_tuned_model
        print(fine_tuned_model_id, "\n")
        break

    elif response2.status == "failed":
        print("Fine-tuning is not yet complete.")
        break

    time.sleep(10)
```

코드리뷰 - 로봇 이동

```
std::string keyword = "movement";
std::string str = _answer;

if (str.find(keyword) != std::string::npos) {
    std::cout << "The string contains '" << keyword << "'," << std::endl;

    size_t x_pos = str.find("x:S");
    if (x_pos != std::string::npos) {
        size_t start = x_pos + 2;
        size_t end = str.find("}", start);

        std::string x_value_str = str.substr(start, end - start);
        x_value_str.erase(0, x_value_str.find_first_not_of(" \t"));
        x_value_str.erase(x_value_str.find_last_not_of(" \t") + 1);

        _x_val = std::stof(x_value_str);
        std::cout << "Extracted value of x: " << _x_val << std::endl;
    }
}
```

```
void Gpt::publish_turtlesim_msg()
{
    auto msg = geometry_msgs::msg::Twist();
    msg.linear.x = _x_val;

    _pub->publish(msg);
}
```


문제점과 해결 과정

1. openai api 연동 이슈

- 문제 : openai api key를 설정해야 한다.

```
Traceback (most recent call last):
  File "/home/jty6109/robot_ws/src/FutureTomorrowExperience_ROS2/test_pkg/src/opanai_test.py", line 4, in <module>
    client = OpenAI()
  File "/home/jty6109/.local/lib/python3.10/site-packages/openai/_client.py", line 105, in __init__
    raise OpenAIError(
openai.OpenAIError: The api_key client option must be set either by passing api_key to the client or by setting the OPENAI_API_KEY environment variable
```

- 해결 : 환경변수에 openai key 설정

```
export OPENAI_API_KEY="openai api key"
```

- 결과

TurtleBot 3에 대해 설명해줘

TurtleBot 3는 로봇 운영 체제 ROS(Robot Operating System)를 기반으로 한 교육용 및 연구용 로봇 플랫폼입니다. TurtleBot 시리즈는 오픈 소스를 기반으로 하여 로봇 공학을 배우고 연구하는 데 필요한 다양한 기능을 제공하고 있습니다. TurtleBot 3는 이전 버전들에 비해 더욱 향상된 성능과 확장성을 갖추고 있습니다.

주요 특징:

1. ****모듈형 디자인****: TurtleBot 3는 모듈형 설계로 다양한 부품을 쉽게 교체하고 업그레이드할 수 있습니다. 특히, 사용자가 필요에 따라 로봇의 센서, 구동 장치 등을 변경하거나 추가할 수 있습니다.
2. ****작고 가벼움****: 이전 버전들과 비교하여 소형화되고 가벼워졌습니다. 덕분에 좁은 공간에서도 활용이 가능하며 휴대성이 높습니다.
3. ****다양한 모델****: TurtleBot 3는 모델별로 다양한 스펙을 제공하며, 대표적인 모델로는 Burger와 Waffle/Waffle Pi가 있습니다. Burger는 작고 가벼운 반면, Waffle은 더 많은 센서와 강력한 성능을 자랑합니다.
4. ****ROS 지원****: TurtleBot 3는 ROS와 완벽하게 호환되어, 다양한 ROS 패키지를 활용하여 복잡한 로봇 제어 및 데이터 처리 작업을 수행할 수 있습니다. 이를 통해 SLAM (Simultaneous Localization and Mapping), 내비게이션, 객체 인식 등 다양한 로봇 기능을 구현할 수 있습니다.
5. ****확장성****: 3D 프린터를 이용해 사용자 맞춤형 부품을 제작할 수 있으며, 다양한 오픈 소스 하드웨어와 소프트웨어를 결합하여 로봇을 확장할 수 있습니다.

문제점과 해결 과정

```
#include <cstdlib>
#include <sys/stat.h>
```

```
const char* filePath = "/home/ubuntu/robot_ws/src/FutureTomorrowExperience_ROS2/result_audio/answer.wav";
const char* command = "ffmpeg -i /home/ubuntu/robot_ws/src/FutureTomorrowExperience_ROS2/result_audio/answer.wav -acodec pcm_s16le -ar 44100 -ac 2 answer2.wav";

if (chmod(filePath, S_IRUSR | S_IWUSR | S_IXUSR) == -1) {
    std::cerr << "Error changing file permissions: " << strerror(errno) << std::endl;
} else {
    std::cout << "Permissions changed successfully for " << filePath << std::endl;
}

int result = system(command);
RCLCPP_INFO(get_logger(), "%d", result);
```

```
Permissions changed successfully for /home/ubuntu/robot_ws/src/FutureTomorrowExperience_ROS2/result_audio/answer.wav
Failed to open sound file "/home/ubuntu/robot_ws/src/FutureTomorrowExperience_ROS2/result_audio/answer.wav" (format not supported)
Error loading audio file!
```

2. 음성파일 재생 문제점

- 문제 : 저장된 오디오 파일 실행권한 X
tts노드 저장한 음성파일이, sfml 라이브러리가 재생할 수 없는 포맷이었음
- 해결 : 실행권한 부여, 포맷 변경

프로젝트 결과

실험 결과

TurtleBot 3에 대해 설명해줘
TurtleBot 3는 교육과 연구 목적으로 많이 사용되는 소형 모바일 로봇 플랫폼입니다. OpenRobotics와 ROBOTIS에서 개발한 TurtleBot 시리즈 중 하나로, 특히 ROS(Robot Operating System)와의 호환성이 높아 다양한 로봇 프로젝트에 활용됩니다.

주요 특징은 다음과 같습니다:

1. ****모듈식 디자인****: TurtleBot 3는 모듈식으로 설계되어 있어 사용자가 쉽게 조립하고 부품을 교체할 수 있습니다. 이를 통해 로봇 하드웨어를 자신의 필요에 맞게 커스터마이징 할 수 있습니다.

2. ****ROS 지원****: ROS의 다양한 패키지와 함께 사용될 수 있어, 로봇 제어 및 시뮬레이션에 매우 유리합니다. Gazebo와 같은 시뮬레이션 툴과도 잘 통합됩니다.

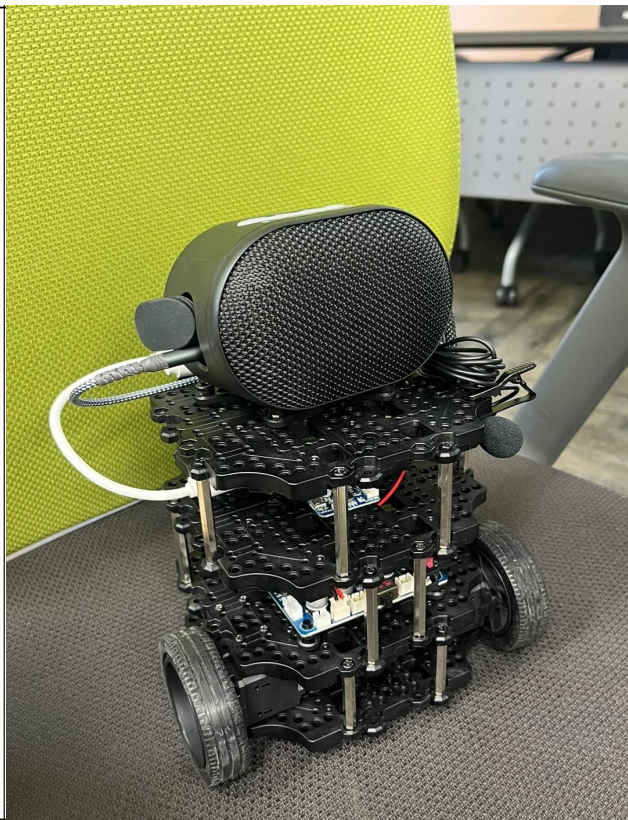
3. ****오픈소스****: 하드웨어 및 소프트웨어 모두 오픈소스라서 연구 목적이나 교육 목적으로 사용하기에 적합합니다. 다양한 커뮤니티와 협력하여 문제를 해결하고 개선할 수 있습니다.

4. ****다양한 모델****: TurtleBot 3에는 "Burger"와 "Waffle"이라는 두 가지 모델이 있습니다.

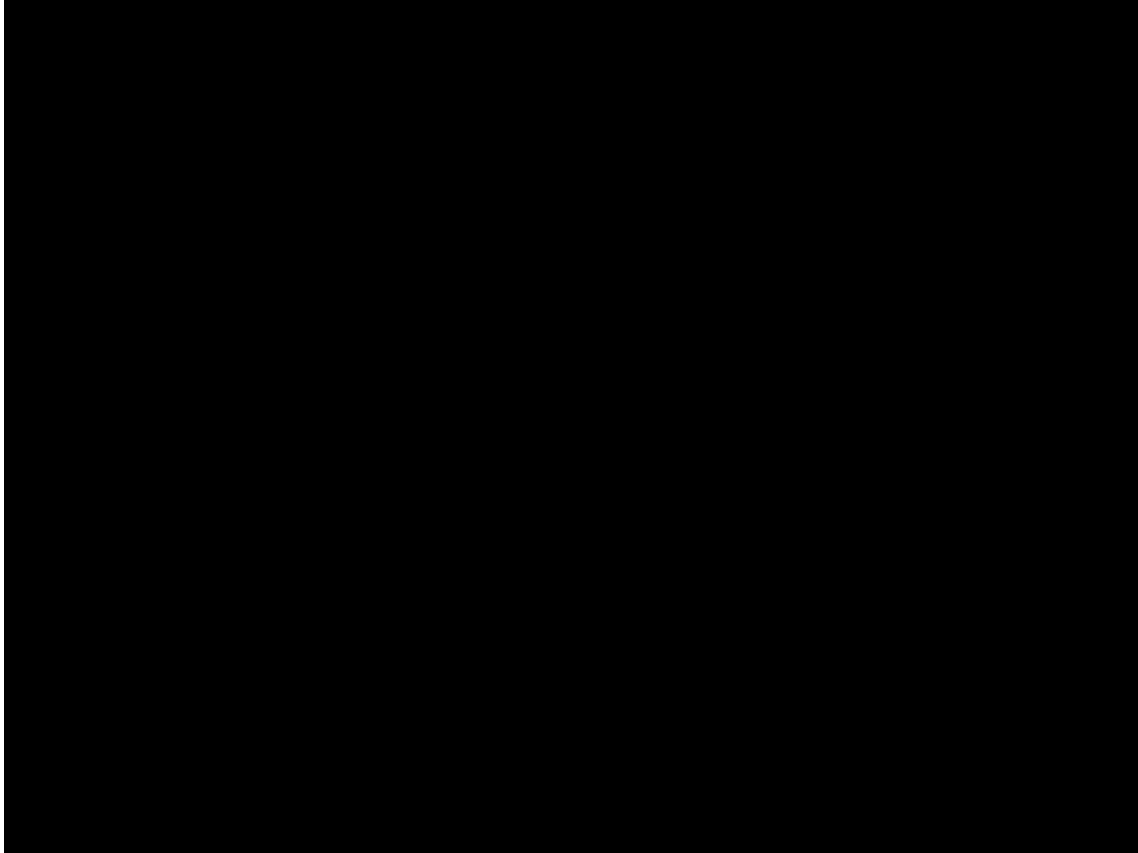
- ****Burger****: 보다 저렴하고 소형화된 모델로, 교육 및 기본적인 연구용으로 적합합니다.

- ****Waffle****: 더 큰 크기와 더 강력한 성능을 가지

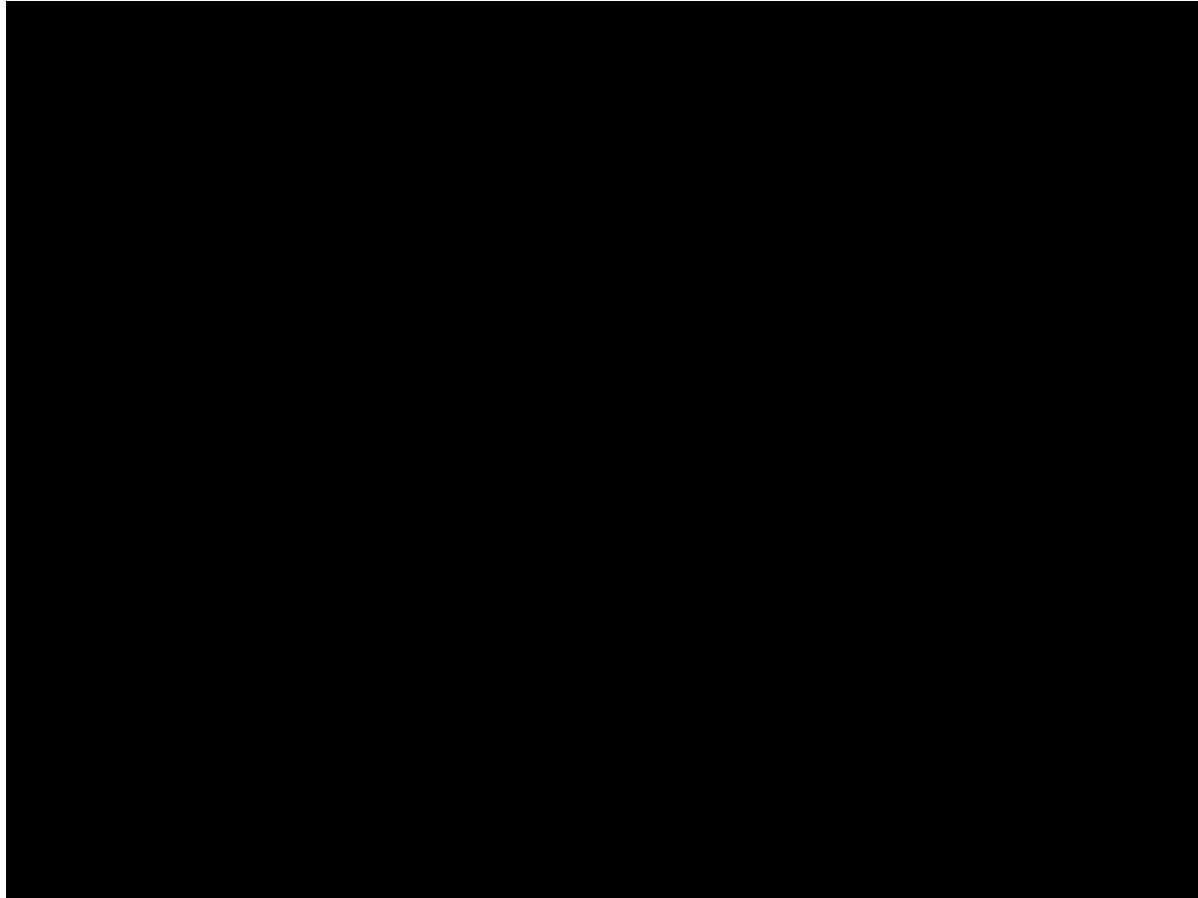
로봇



프로젝트 결과 - 음성대화 시연영상



프로젝트 결과 - 로봇이동 시연영상

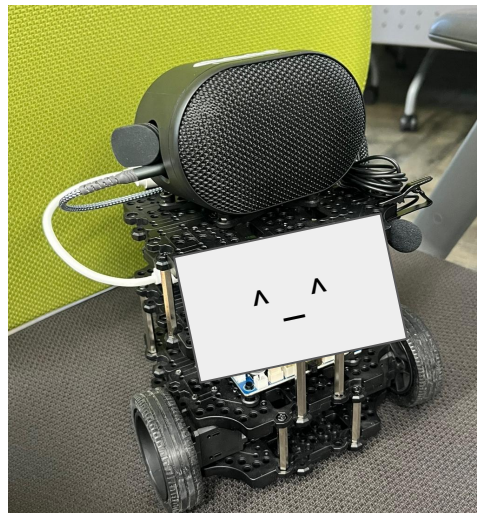


향후 도전과제

- **Response 문제 해결**
- 감정분석
- SLAM, Navigation 이용 구역 인식
- 특정 구역에서 특정 gpt모델 적용

```
[INFO] [1724291069.815598176] [bdirge_node]: 응답 메시지: answer  
recorded complete
```

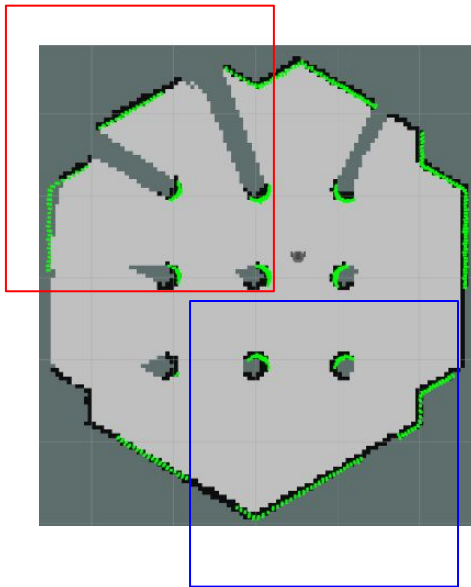
```
[INFO] [1724291046.245882836] [whisper_node]: test  
[INFO] [1724291046.246109700] [whisper_node]: test2  
[INFO] [1724291047.013185049] [whisper_node]: 응답 메  
시지:
```



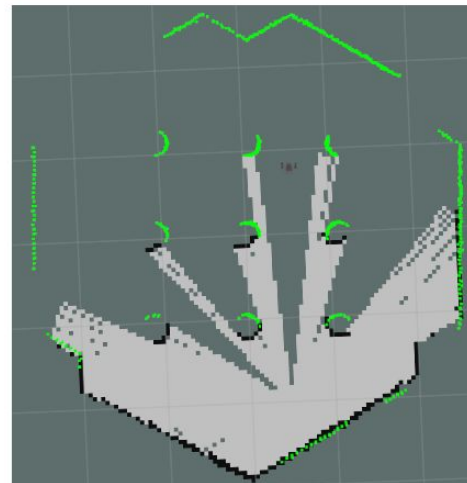
향후 도전과제

- Response 문제 해결
- 감정분석
- SLAM, Navigation 이용 구역 인식
- 특정 구역에서 특정 gpt모델 적용

주방 -> 요리보조 모델



방 -> 학습보조 모델



Q & A



감사합니다

