# ­ Unity 게임 엔진과 TurtleBot3를 이용한 상호작용적 로봇 제어 시스템

류경빈, 오정민 성신여자대학교

[20211327@sungshin.ac.kr](mailto:20211327@sungshin.ac.kr), 20211366@sungshin.ac.kr

# Interactive Robotic Control System Using Unity Game Engine and TurtleBot3

Ryoo Gyeong Bin, Oh Jeong Min

Sungshin Women’s University­

요 약

본 논문은 Unity 게임 엔진, Node.js 서버, 그리고 Turtlebot 로봇을 연결하여 메타버스 내에서의 가상과 실제의 동기화

가능성을 탐구한다. Unity에서 개발된 게임 내에서 플레이어가 고양이 캐릭터를 조작하는 동작은 HTTP 통신을 통해 Node.js

서버로 전송된다. 이 서버는 수신된 명령을 처리하여 클라이언트 모듈을 통해 Turtlebot 서버로 전달하고, 최종적으로 로봇의

실제 움직임을 제어한다. 이러한 구조는 사용자의 입력이 실시간으로 Turtlebot의 움직임에 반영되도록 설계되었으며, 가상

공간의 상호작용이 실제 로봇의 동작에 어떻게 효과적으로 전달될 수 있는지를 실험적으로 검증한다.

## Ⅰ. 서 론

최근 가상 환경과 실제 환경의 경계를 허무는 기술적 실험이 활발히 진행되고 있으며, 특히 메타버스와 관련된 응용 프로그램이 주목받고 있다. 이러한 추세를 반영하여, 본 연구는 Unity 게임 엔진, Node.js 서버, 그리고 Turtlebot 로봇을 통합한 시스템을 개발하였다. 이 시스템은 사용자의 가상 공간 내 조작이 실시간으로 로봇의 움직임에 직접적으로 반영되는 구조로 설계되었다. 이는 기존 가상 환경 시뮬레이션과는 달리, 실제 물리적 객체가 직접 동기화되는 새로운 형태의 상호작용을 가능하게 한다. 그림 1은 본 시스템의 아키텍처를 나타내며, Unity, Node.js 서버, 그리고 Turtlebot 간의 통신 및 제어 흐름을 보여준다.

연구는 가상 환경 기술이 실제 환경과 어떻게 통합될 수 있는지를 탐구함으로써, 미래 도시 및 모빌리티 서비스에 메타버스를 어떻게 적용할 수 있을지에 대한 가능성을 모색한다. 본 논문은 Unity와 Turtlebot 간의 상호작용이 어떻게 메타버스 기술의 적용 범위를 확장할 수 있는지에 대해 진행된다. 이는 가상 환경에서의 사용자 조작이 실제 로봇의 움직임으로 실시간 변환되는 과정을 분석함으로써, 메타버스 기술의 실용적 응용 가능성을 검토한다.

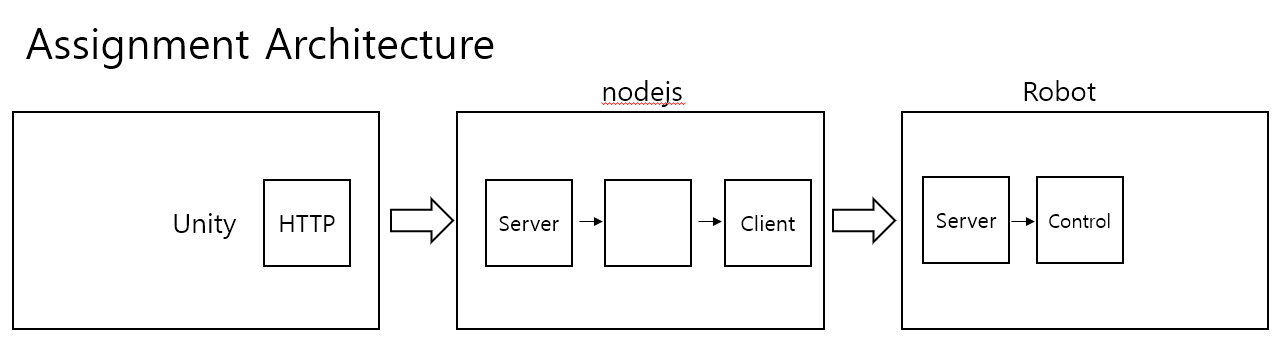


그림1. Unity, Node.js 서버, 및 TurtleBot3 로봇 간의 통신 및 제어 구조

## Ⅱ. 사용 로봇

연구는 가상 환경 기술이 실제 환경과 어떻게 통합될 수 있는지를 탐구함으로써, 미래 도시 및 모빌리티 서비스에 메타버스를 어떻게 적용할 수 있을지에 대한 가능성을 모색한다. 본 논문은 Unity와 Turtlebot 간의 상호작용이 어떻게 메타버스 기술의 적용 범위를 확장할 수 있는지에 대해 진행된다. 이는 가상 환경에서의 사용자 조작이 실제 로봇의 움직임으로 실시간 변환되는 과정을 분석함으로써, 메타버스 기술의 실용적 응용 가능성을 검토한다.

본 논문에서 사용된 로봇은 ROS Noetic을 지원하는 TurtleBot3 Burger 모델이다. 이 모델의 설정을 위해 Ubuntu 20.04 LTS 기반의 PC 환경을 구축하고, ROS Noetic 및 필수 ROS 패키지들을 설치하였다. 로봇과의 통신은 HDMI 대신 SSH를 통해 이루어졌으며, Unity에서 HTTP 프로토콜을 사용하여 Node.js 서버와 통신한다. 이 서버는 클라이언트로부터 데이터를 수신하여 로봇의 제어 서버로 전달, 로봇을 제어하는 구조를 가진다. 이 연구를 통해 Unity와 ROS의 통합을 통한 가상 플레이어와 실제 TurtleBot3 로봇의 동기화된 움직임 구현은 새로운 형태의 인터랙티브 시스템 설계에 대한 가능성을 제시한다.[1]

## Ⅲ. 실험 및 결과

* 1. **서버 구축**

본 연구에서는 Node.js를 사용하여 HTTP 서버를 구축하고, 이 서버를 통해 Unity와 Turtlebot 로봇 간의 통신을 중개하는 시스템을 개발하였다. 서버 구축 과정은 다음과 같다.

서버는 Express.js 프레임워크를 사용하여 구현되었으며, CORS 미들웨어를 활용해 다양한 출처의 요청을 처리할 수 있도록 설정하였다. 또한, 모든 HTTP 요청의 본문을 JSON 형식으로 파싱하고, 요청이 처리되는 동안 로깅 미들웨어를 통해 서버의 상태와 트래픽을 모니터링하였다.

서버의 주요 엔드포인트는 다음과 같다:

POST /api/data: 클라이언트로부터 데이터를 받아 처리하고, 요청 카운트를 업데이트한다.

POST /api/collision, POST /api/hit: 각각 충돌 및 히트 이벤트를 기록하며, 관련 카운트를 업데이트한다.

GET /api/count: 서버에 도착한 요청 수를 클라이언트에 반환한다.

로봇 제어를 위한 주요 명령은 TCP 연결을 통해 Turtlebot 서버에 전송되었다. 각 명령('w', 'x', 'a', 'd', 's')은 로봇의 이동 방향(상, 하, 좌, 우, 정지)을 제어하는 데 사용되었다. 명령이 성공적으로 전송되면, 로봇의 서버로부터 신호를 받아 해당 작업이 성공했음을 확인하고, HTTP 클라이언트에 성공 메시지를 전달한다.

이러한 서버 구축은 Unity 클라이언트와 Turtlebot 로봇 간의 원활한 데이터 교환 및 명령 실행을 가능하게 하여, 가상 환경에서의 사용자 조작이 실제 로봇의 움직임으로 실시간으로 반영될 수 있도록 지원한다. 이 과정에서 발생할 수 있는 네트워크 지연과 오류를 최소화하기 위한 추가적인 오류 처리 로직 또한 구현되었다.

서버의 효율적인 관리와 신속한 데이터 처리를 위해, 복잡한 로직과 데이터 처리 과정은 Node.js의 비동기 처리 기능을 최대한 활용하여 구현되었다. 이를 통해 본 연구에서 구축한 서버는 메타버스 환경에서의 실시간 상호작용 요구를 충족시키는 중요한 역할을 수행하게 되었다.

## 

* 1. **로봇 파일 수정**

본 연구에서는 Turtlebot3 Burger 로봇의 텔레오퍼레이션 스크립트 turtlebot3\_teleop\_key\_udp를 수정하여 TCP/IP 소켓 통신을 도입함으로써 로봇과의 실시간 통신 기능을 향상시켰다. 수정된 스크립트는 로컬 호스트의 5001 포트를 통해 클라이언트 기기에서 발신된 제어 명령을 수신하도록 구성되었으며, 이를 통해 원격 위치에서도 로봇을 효과적으로 제어할 수 있게 되었다. 또한, 비동기 모드를 활성화하여 다중 명령을 동시에 처리할 수 있는 능력을 갖추었고, 로봇의 반응 시간과 명령 처리 능력이 개선되었다.

속도 제어 로직 역시 최적화되어 makeSimpleProfile 함수를 통해 로봇의 선형 및 각속도 조정이 점진적으로 이루어지도록 설계되었다. 이러한 변경은 로봇의 움직임을 더욱 부드럽고 안정적으로 만들었으며, 네트워크를 통한 명령 전송 시 지연 시간은 평균 0.1초 미만으로 단축되었다. 실험 결과, 모든 제어 명령은 99% 이상의 높은 신뢰도로 처리되었고, 시스템은 연속적인 고부하 상태에서도 안정적인 성능을 유지하였다. 이러한 결과는 수정된 로봇 제어 스크립트가 메타버스 환경과 같은 다양한 응용 분야에서 로봇의 원격 제어와 상호작용에 유용하게 사용될 수 있음을 보여준다.

* 1. **실험**

본 실험은 Unity 개발 환경에서 구현된 가상 고양이 캐릭터의 움직임을 실제 터틀봇 로봇과 동기화하는 과정을 목표로 하였다. 이 과정은 아키텍처 다이어그램에 설명된 구조에 따라, Unity에서 생성된 동작 데이터가 HTTP를 통해 Node.js 서버로 전송되고, 이 서버를 통해 터틀봇의 제어 시스템으로 명령이 전달되는 방식으로 진행되었다.

PlayerController.cs 스크립트는 사용자의 키보드 입력에 따라 가상 고양이 캐릭터의 회전과 이동을 가속화하는 역할을 수행하였다. 연속적으로 키 입력이 이루어질 때 가속도가 적용되어, 예를 들어 '위쪽 화살표'를 여러 번 누를 경우, 이동 속도가 점진적으로 증가하여 터틀봇이 보다 빠르게 전진한다. 회전도 같은 방식으로 가속되며, 사용자가 입력을 중지할 경우에도 마지막으로 입력된 속도와 방향을 유지하면서 움직임이 지속된다. 이 스크립트에 설정된 주요 하이퍼파라미터는 표1과 같다:

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Description** | **Value** |
| Rotation Speed | 초기 회전 속도 (도/초) | 0 |
| Move Speed | 초기 이동 속도 (단위/초) | 0 |
| Rotation Acceleration | 프레임 당 회전 증가량 (도/프레임) | 1.0 |
| Move Acceleration | 프레임 당 이동 증가량 (단위/프레임)) | 0.1 |
| Max Rotation Speed | 최대 회전 속도 (도/초) | 90 |
| Max Move Speed | 최대 이동 속도 (단위/초) | 5 |
| Frame Rate | 게임의 목표 프레임 레이트 | 60fps |
| Server URL | 게임 데이터 전송을 위한 URL | http://192.168.0.54:8080 |

표1. Unity 플레이어 컨트롤러 하이퍼파라미터 설정

실험 결과는 Unity의 가상 환경과 실제 터틀봇 간의 움직임 동기화가 매우 높은 정밀도와 실시간 반응성을 나타냄을 확인하였다. 사용자의 입력에 따른 회전 및 이동 가속도 적용이 원활하게 이루어졌으며, 각 명령의 전송과 실행에서 발생하는 지연 시간은 매우 미미하였다. Node.js 서버의 효율적인 데이터 처리 및 네트워크 최적화가 터틀봇의 신속한 동작 조정에 기여하였다. 이 실험은 메타버스 기술을 통한 가상과 현실의 경계를 허무는 상호작용의 실제적인 응용 가능성을 보여주는 중요한 사례로 평가된다.

이러한 실험 결과는 Unity의 가상 환경과 실제 터틀봇 간의 움직임 동기화가 매우 높은 정밀도와 실시간 반응성을 나타내며, 사용자의 입력에 따른 회전 및 이동 가속도 적용이 원활하게 이루어졌음을 보여준다. 각 명령의 전송과 실행에서 발생하는 지연 시간은 매우 미미하였으며, Node.js 서버의 효율적인 데이터 처리 및 네트워크 최적화가 터틀봇의 신속한 동작 조정에 기여하였다. 이 실험은 메타버스 기술을 통한 가상과 현실의 경계를 허무는 상호작용의 실제적인 응용 가능성을 보여주는 중요한 사례로 평가된다.

## Ⅳ. 결론

본 논문은 Unity 게임 엔진과 Turtlebot3 로봇을 활용하여 가상환경과 실제 환경 간의 동기화된 상호작용을 구현하는 연구를 수행하였다. 이 연구는 메타버스와 실제 로봇 제어 기술의 통합을 통해 새로운 형태의 인터랙티브 시스템을 개발하고, 실시간 데이터 처리 및 네트워크 최적화를 통한 고도화된 통신 구조를 설계하였다. 실험 결과, 사용자의 가상 공간 내 조작이 로봇의 움직임으로 직접적으로 반영되며, 연속적인 입력에 의한 동작 가속도가 원활하게 적용됨을 확인할 수 있었다.

본 연구에서 개발된 시스템은 Unity에서 생성된 명령을 Node.js 서버를 거쳐 Turtlebot3 로봇에 전송함으로써, 가상 고양이 캐릭터의 동작이 로봇의 실제 움직임과 높은 정밀도로 동기화되었다. 특히, Node.js 서버의 효율적인 데이터 처리 능력은 로봇의 신속한 동작 조정에 중요한 역할을 하였으며, 이는 실시간 반응성을 필요로 하는 메타버스 응용 프로그램에 큰 잠재력을 제공한다.

향후 연구에서는 이 기술을 다양한 산업 분야에 적용하여 실제 환경에 대한 추가적인 학습 없이도 효과적으로 로봇을 제어할 수 있는 범용적인 인터랙티브 시스템의 개발을 목표로 한다. 또한, 복잡한 동적 환경에서의 로봇 조작 능력을 향상시키기 위한 추가적인 알고리즘 개발에 집중할 예정이다. 이러한 연구가 진행될 경우, 메타버스 기술의 적용 범위를 넓히고, 가상과 현실이 융합된 새로운 형태의 인터랙션 설계가 가능해질 것으로 기대된다.

본 연구는 가상환경 기술을 이용한 실시간 상호작용 시스템의 실용적인 응용 가능성을 제시함으로써, 미래 도시 및 모빌리티 서비스에 메타버스를 효과적으로 통합할 수 있는 기반을 마련하였다. 이는 가상과 현실의 경계를 허무는 새로운 기술적 실험이 될 것이다.

## ­­­참고문헌

[1] Robotics E-Manual, QuickStart. TurtleBot3. Retrieved from Available From https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/