논문 2025-2-10 http://dx.doi.org/10.29056/jsav.2025.06.10

휴머노이드 로봇의 기본 제어에 관한 연구

황동하*, 김재웅*† , 김동현**

A Study on the Basic Control of Humanoid Robot

Dong-Ha Hwang*, Jae-Woong Kim*† , Dong-Hyun Kim**

요 약

이 논문은 파이썬 기반 SDK 및 API 프레임워크를 활용한 휴머노이드 로봇의 기본 제어 시스템 구현 및 실험 분석을 제시한다. 휴머노이드 로봇이 의인화된 설계 및 동작 능력으로 인해 다양한 응용 분야에서 점점 더 많이 채택됨에 따라, 정밀하고 모듈화된 제어 기술의 개발이 필수적이다. 본 연구는 제어 정확도와 시스템 응답성을 종합적으로 평가하여 이동, 팔 조작, 균형 유지를 포함한 핵심 기능을 구현하고 검증하였다. 실시간 센서 데이터 수집 및 처리 시스템은 ROS(로봇 운영 체제) 통합을 통해 개발되었으며, 특히 LiDAR와 IMU 센서 스트림을 대상으로 하였다. 센서 통합 실험은 지연 시간, 데이터 안정성, 처리 효율성이라는 세 가지 주요 성능 지표를 기반으로 체계적으로 평가되고 센서 입력과 제어 명령 간의 상호작용을 모듈화하여 강화 학습 기반 자율 제어 알고리즘과 호환되는 프레임워크를 구축했다. 실험 결과는 이동 명령의 평균 응답 시간 0.2초와 위치 정확도 5% 오차 범위 내에서 신뢰할 수 있는 제어 성능을 달성하는 데 있어 시스템의 효과를 입증하였고 개발된 프레임워크는 휴머노이드 로봇 공학에서 고급 자율 제어 시스템과 다중 모달 센서 융합 응용을 위한 견고한 기반을 제공한다.

Abstract

This paper presents the implementation and experimental analysis of a fundamental control system for humanoid robots utilizing Python-based SDK and API frameworks. As humanoid robots gain increasing adoption across diverse applications due to their anthropomorphic design and motion capabilities, the development of precise and modularized control technologies becomes essential. This study implements and validates core functionalities including locomotion, arm manipulation, and balance maintenance, with comprehensive evaluation of control accuracy and system responsiveness. A real-time sensor data acquisition and processing system was developed through ROS integration, specifically targeting LiDAR and IMU sensor streams. The sensor integration experiments were systematically evaluated based on three key performance metrics: latency, data stability, and processing efficiency. Furthermore, the interaction between sensor inputs and control commands was modularized to establish a framework compatible with reinforcement learning-based autonomous control algorithms. Experimental results demonstrate the system's effectiveness in achieving reliable control performance with average response times of 0.2 seconds for locomotion commands and positioning accuracy within 5% error margins. The developed framework provides a robust foundation for advanced autonomous control systems and multi-modal sensor fusion applications in humanoid robotics.

한글키워드 : 휴머노이드 로봇, 센서데이터, 기본 제어, 보행, 로봇운영체제

keywords : Humanoid Robot, Sensor Data, Basic Control, Walking, ROS(Robot Operation System)

* 공주대학교 컴퓨터공학과
** 나사렛대학교 IT인공지능학부
† 교신저자: 김재웅(email: jykim@kongju.ac.kr)

접수일자: 2025.06.01. 심사완료: 2025.06.13.
게재확정: 2025.06.20.

1. 서 론

휴머노이드 로봇은 인간과 유사한 동작과 형태를 가지고 있는 로봇으로[1], 사람과 자연스러운 상호작용 및 다양한 환경에서의 적응 능력을 기반으로 산업, 돌봄, 의료 등 여러 분야에서 활용 가능성이 확대되고 있다. 최근 인공지능, 센서 융합 기술, 경량 소재 기술 등의 발전은 휴머노이드 로봇의 기능 향상에 크게 기여하고 있으며, 특히 고령화 사회와 더불어 인구 감소로 인한 생산 노동인구 부족 문제에 대한 해결방안으로 그 중요성이 부각되고 있다.

대한민국은 65세 이상의 인구 비율이 크게 증가하고 있는 추세이며[2,3], 출산율이 세계 최저 수준에 머무르면서 노동력 감소와 부양 부담이 심화되고 있는 상황이다[4]. 이에 휴머노이드 로봇의 정밀 제어, 환경 인식, 자율 의사결정 역량 강화를 위하여 하드웨어와 소프트웨어를 통합한 제어 구조에 관한 연구가 필요하다. 본 논문은 고성능 로봇 플랫폼에 파이썬 SDK(Software Development Kit)로 제공되는 API(Application Programm ing Interface)를 ROS(Robot Operating System)에 연동하는 기법을 적용하여 센서 데이터 처리 및 제어 프레임워크를 구현하고 분석함으로써 자율 제어 시스템 개발 기반을 제안하고자 한다. 이를 위하여 휴머노이드 로봇의 기본 제어 시스템을 코드로 구현하고, 실험 로봇을 이용하여 검증하는 것을 목표로 다음과 같이 세 가지의 연구 범위를 설정하였다. 첫째, 기본 동작 제어 구현으로서, 파이썬 기반의 SDK를 활용하여 로봇의 보행, 팔 동작, 균형 유지 등의 핵심 기능을 구현하고, 고수준 및 저수준 API의 제어 특성과 활용 방안을 분석한다. 둘째, ROS 기반 센서 연동으로서, IMU(Inertial Measurement Unit), LiDAR, 발바닥 압력 센서 등 다양한 센서 데이터를 ROS를 통하여 수집하고, 이 데이터를 실시간으로 처리하는 프레임워크를 구성하여 센서 기반 제어 구조를 검토한다. 셋째, 제어 성능 실험 및 분석으로서 제어 명령의 반응 속도, 센서 데이터 수신 주기, 동작 정밀도 등을 중심으로 실험을 설계하고, 이를 통하여 SDK 및 API의 실질적 성능을 정량적으로 평가한다. 이를 통하여 휴머노이드 로봇에 SDK, API 활용법을 체계화함으로서, 초기 개발 단계에 실질적인 가이드를 제공할 수 있으며, ROS 연동으로 실시간 센서 데이터 처리 및 제어 구조를 구현하여 자율 제어 알고리즘 적용 기반을 마련하고, 제어 응답성, 동작 정확성, 센서 활용 성능을 실험 평가해 지능형 로봇 시스템의 실질적 성능 데이터의 제시가 가능할 것으로 기대된다.

2. 관련 연구

2.1 휴머노이드 로봇의 개념과 발전

휴머노이드 로봇은 인간의 신체 구조와 유사한 형태로 보행, 조작, 상호작용 기능을 수행하며, 센서 기반 환경 인식과 자율 제어 알고리즘이 통합된 시스템이다[5]. 현대의 휴머노이드 로봇은 신체를 구성하는 기계적 구조와 주변 환경을 인식하기 위한 센서 시스템, 이를 융합하고 종합적으로 제어 하기 위한 제어 소프트웨어로 구성되어 있다. 신체에 해당하는 기계적 구조는 인간과 유사한 형태의 관절 배치를 갖고 있고 로봇의 성능에 따라 더 많은 자유도를 갖게 된다. 센서시스템은 로봇의 내부 상태를 모니터링 하는 기능을 포함하여 외부의 환경 인식에 주력하고 있다. 이를 위해 다양한 센서들이 활용된다. 마지막으로 제어 소프트웨어는 펌웨어와 SDK로 나위어 지고 있으며 Secondary Development를 위해 다양한 SDK를 통해 접근 할 수 있다. 다음의 표 1은 본 논문에서 사용된 Unitree G1 로봇의 구성 요소를 나타낸 것이다.

표 1. Unitree G1 로봇 구성요소
Table 1. Main components of Unitree G1 robot

Item	Specifications
Physical	Height 127 cm Weight 35 kg 최대보폭 0.65 m 최대 보행속도 2.0 m/Sec
Joint	DOF : 23(Arms 6*2, Legs 6*2, Waist 1) 구동기 : 고토크 서보모터 위치 정확도 : ±0.02 rad (약 1.15°) 최대 토크 : 200Nmr
Control System	통신 : WiFi 802.11ac, Ethernet 제어주기 : 1kHz(저수준), 100Hz(고수준) SDK지원 : Python, C++, ROS
Sensor System	IMU :6축 관성측정장치 LiDAR :2D, 10m 범위, 360° 회전 RGB카메라:1080p, 30fps 발바닥압력센서 : 4점/각발 Depth 카메라 : Intel RealSense

휴머노이드 로봇은 경량화된 프레임, 다자유도 관절 시스템, 고출력 모터, 다양한 센서 모듈, 실시간 제어 및 통신 시스템으로 구성되어 있으며, 일반적으로 20개 이상의 관절 DOF(Degrees of Freedom)를 가지고 있고, 정밀한 움직임과 안정적인 균형 유지를 위한 구조로 설계된다. 2010년대에는 ASIMOP, NAO, Atlas 등이 균형 유지, 인간 상호작용 기능을 보유한 로봇들이 출시되었고, 2020년 이후에는 강화 학습 기반 보행 최적화와 멀티모달 센서 융합 기술이 적용된 Sophia, Tesla Optimus, Unitree G1과 같은 최신의 휴머노이드가 개발되어[6], 지능형 로봇의 실제 환경 응용 가능성이 크게 확대되고 있는 실정이다.

2.2 제어 소프트웨어

제어 소프트웨어는 2차 개발의 편의성을 위해서 SDK로 배포되고 있으며 파이썬 및 C++ 기반으로 되어 있다[7]. API는 저수준 제어와 고수준 제어로 제공 되며 고수준 제어는 보행, 팔 동작, 균형 유지와 같은 명령을 간단하게 구현한다. 저수준 API는 각각의 관절단위의 속도, 토크, 위치를 세밀하게 제어하여 다양한 모션을 수행할 수 있는 기능을 제공한다[8]. 고수준 API는 이미 학습되거나 프로그램되어 저장되어 있는 로봇의 모션을 번호를 전달하는 함수 호출로 수행 하게 된다[9]. 저수준 API는 로봇의 각 모터의 세부적인 설정값을 입력할 수 있어 정밀한 움직임을 구현하거나 특정 동작을 재현 하는데 활용된다[10]. 다음의 그림 1은 특정한 관절(joint_id)을 지정한 위치 0.5로 이동시키는 파이썬 코드로 API를 통한 저수준의 제어 명령 전송 방식을 나타낸 것이다.

```python
from unitree_sdk import Robot

robot = Robot()
robot.set_joint_position(joint_id=1, position=0.5)
# 1번 관절을 0.5 위치로 이동
```

그림 1. SDK를 이용한 제어 코드
Fig. 1. Control code using SDK

2.3 ROS 및 센서 데이터 활용

ROS는 로봇 소프트웨어 개발을 위한 오픈소스 프레임워크로, 메시지 기반 구조를 통하여 센서 데이터 수집, 제어 명령 송신, 시스템 통합 등을 효율적으로 지원한다[11]. 다음의 그림 2는 rospy를 이용하여 /g1/imu 토픽에서 IMU 메시지를 구독하고, 콜백 함수를 통해 오리엔테이션 데이터를 출력하는 코드이다. 이는 센서 데이터를 실시간으로 처리하는 기본적인 ROS 노드 구조를 나타낸 것이다. 각각의 노드는 분산 시스템으로 구성되어 독립적으로 동작하게 되고 노드들간의 통신은 토픽, 서비스, 액션의 세가지 메커니즘을 통해 이루어 진다. 토픽은 Publish과 & Subscribe의 패턴을 사용된다.

```
import rospy
from sensor_msgs.msg import Imu
def imu_callback(data):
    print("IMU 데이터:", data.orientation)
rospy.init_node('g1_imu_listener')
rospy.Subscriber("/g1/imu", Imu, imu_callback)
rospy.spin()
```

그림 2. ROS 기반 IMU 센서 데이터 수신 코드
Fig. 2. ROS-based IMU sensor data reception code

3. 휴머노이드 로봇 제어 시스템

휴머노이드 로봇의 기본 제어 시스템 구현을 위하여 제어 구조 설계[12], SDK 및 API 설정, 로봇 동작 구현[13], 센서 데이터 처리[14], 실험 설계 및 평가 방법[15]으로 구성하였다.

3.1 시스템 구성

휴머노이드 로봇 제어를 위한 시스템은 PC, 로봇 본체, 네트워크 인터페이스, 소프트웨어 개발 환경으로 구성된다. ROS는 생성된 센서 Node를 통해 데이터 실시간 수집을 하고, 이를 이용, 분석하여 강제적인 저수준 API를 이용한 제어나 고수준의 후속 동작에 대한 제어 알고리즘에 반영된다. 다음의 그림 3은 휴머노이드 로봇 제어를 위한 PC와 로봇간의 구성과 ROS의 역할을 나타낸 시스템의 구조도이다.

시스템의 구조는 제어 PC와 G1 휴머노이드 로봇 간의 WiFi 연결, 실시간 데이터 수집을 위한 ROS로 구성되어 있다.

다음의 표 2는 휴머노이드 로봇 제어 시스템의 구성요소를 구체화 하였다. 제어 PC는 파이썬 SDK, ROS를 활용하여 시나리오 기반의 휴머노이드 제어 알고리즘을 구현하였다.

ROS 노드를 통해 주기적으로 수집되는 센서 데이터는 DRGB(Depth Red Green Blue)카메라, LiDAR, IMU, 발바닥 압력 센서 등이다. 동작 명령이 전송된후 수집되는 데이터의 분석 결과는 다시 제어 명령으로 이어져 피드백 루프를 형성하게 된다.

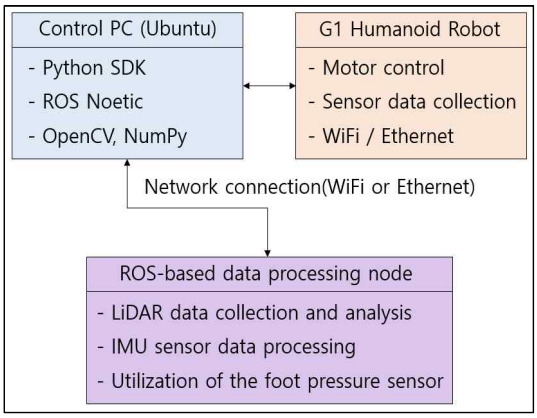

그림 3. 시스템 구조도
Fig. 3. System Architecture Diagram

표 2. 제어 시스템의 구성 요소
Table 2. Components of a control system

Component	Contents
Control PC	Ubuntu 20.04 LTS, Python 3.8
Robot	Unitree G1 Humanoid
Network	WiFi
ROS	ROS Noetic + Unitree SDK
Sensor	IMU, LiDAR, DRGB camera, foot pressure sensor
SDK/API	Motion Control, Motor control using Python API
실험 공간	공간 : 5m x5m 평면(실내) 바닦 : 고무매트 조명 :균일한 조명 온습도 : 22±2°C, 45±5% RH 안전장치 : 비상정지, 안전하네스

3.2 시퀀스 제어 프로세스

다음의 그림 4는 휴머노이드 로봇 제어 시퀀스 프로세스이다.

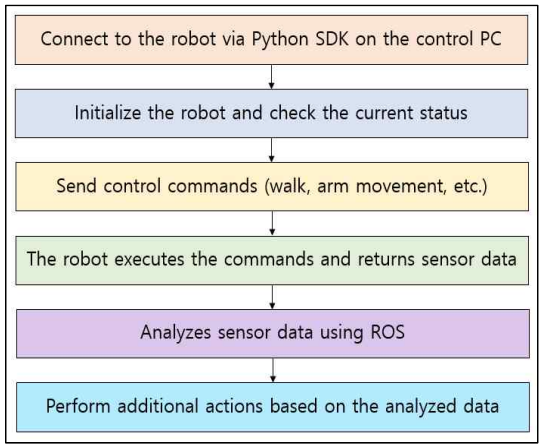

그림 4. 시스템 단계별 시퀀스 프로세스
Fig. 4. System step-by-step sequence process

파이썬 SDK가 설치된 제어 PC에서 로봇과 연결된 후, 로봇의 초기화 및 상태 확인, 최초 제어 명령 전송후 로봇이 실시간으로 수집 전송하는 센서데이터를 수신한다. 수집된 데이터를 분석하고 결과를 기반으로 하여 추가 동작 수행까지의 전체적인 제어 흐름이다.

다음의 그림 5는 휴머노이드 로봇 제어 시스템의 소프트웨어 동작 프로세스이다. 파이썬 기반의 제어 PC에서 G1 휴머노이드에 명령을 전달한다. 이때 전달되는 명령은 walking, arm movement 등과 같이 고수준, 또는 저수준 API를 이용할 수 있다. 전달된 명령어는 WiFi를 통해 전달되어 로봇이 해당 명령을 실행하게 되고 실행하는 동안 발생하는 센서의 데이터 값들은 ROS를 통해 주기적, 또는 실시간으로 수집된다. 이 데이터는 일련의 처리 과정을 거쳐 결과에 따라 연속적인 제어 알고리즘에 반영되게 된다. 이 제어 흐름은 폐루프 시스템을 형성 로봇의 자율적 적응 능력을 제공하는데 활용 할 수 있다. 로봇은 슬레이브 노드로서 명령을 수행하면서 로봇 자신의 상태 데이터를 피드백한다. 이때 네트워크 지연 시간은 최소화 함으로서 실시간 제어가 가능하도록 하여야 한다.

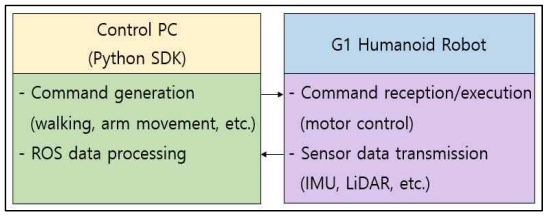

그림 5. 소프트웨어 동작 프로세스
Fig. 5. Software operation process

3.3 SDK 및 API 설정

개발 환경 구축은 anaconda를 이용해서 가상환경하에서 작업을 하였으며 파이썬 3.8 이상과 다음과 같이 3가지가 필요하다. 첫째, 필수 패키지 설치, 둘째, Unitree_sdk2_python 설치, 셋째, 네트워크 설정이다. 필수 패키지와 Unitree_sdk2_python은 requirements.txt를 통해 손쉽게 설치가 가능하지만, 네트워크 설정에는 반드시 제어 PC와 로봇 간에 준수해야 할 규칙이 있다. 예를 들어 이더넷 인터페이스 enp2s0를 사용하는 경우, 해당 인터페이스의 IP 주소를 로봇과 동일한 서브넷으로 설정해야 한다. 로봇의 기본 IP 주소가 192.168.123.161인 경우, 제어 PC의 IP 주소를 192.168.123.162와 같이 C Class에서 동일한 설정을 해야 한다.

Ubuntu 20.04 LTS에서 ROS Noetic과 파이썬 SDK를 설치 및 설정하여 휴머노이드 로봇 제어 기반을 준비했다. ROS 설치와 설정은 다음과 같이 3단계로 이루어진다. 첫째, 다음의 그림 6과 같이 ROS Noetic 설치/환경 설정을 해야 한다.

```
sudo apt update
sudo apt install ros-noetic-desktop-full
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

그림 6. ROS Noetic 설치 및 환경 설정 코드
Fig. 6. ROS Noetic Installation and Configuration Code

둘째, 의존성 패키지 설치 및 워크스페이스 생성으로 ROS 환경 구축을 위해 필수적인 의존성

패키지를 설치하고, 작업 공간(workspace)을 구성하였다. rosdep, rosinstall, wstool, build-essential 등을 설치하고 catkin_ws를 생성해 패키지 의존성을 관리하였다. 셋째, 파이썬 SDK 연동 및 ROS 패키지 구성으로catkin_ws/src 디렉터리에 파이썬 SDK와 관련된 ROS 패키지를 링크하거나 복사하여, catkin_make 명령을 통해 워크스페이스에 통합하였다. 이를 통해 ROS 노드에서 센서 데이터를 구독하고 처리할 수 있는 구조를 구축하였다.

3.4 ROS 통신 검증

로봇과 제어 PC간의 주기적 센서 데이터 수집을 위해서 제어 PC의 ROS가 정상 동작하는 것을 확인 하기 위해 예제 노드를 실행하였다. 검증 과정은 로봇의 움직임 없이 고정(Hanging) 상태에서 전원만 인가하였으며 센서 토픽의 수신 상태 및 데이터 흐름을 확인하였다. 단순하게 로봇의 상태를 읽어 들이는 것은 SDK의 API를 이용해서 다음의 그림 7과 같이 구현할 수 있다. 코드는 로봇 객체를 생성하고, get_status() 함수를 통해 로봇의 기본 연결 및 초기 상태를 확인할 수 있다.

```
from unitree_sdk import Robot
robot = Robot()
print(robot.get_status())  # 로봇 상태 확인
```

그림 7. API를 이용한 로봇 상태 확인 코드
Fig. 7. Robot status check code using API

3.5 G1 휴머노이드 로봇 기본 동작 구현

G1 휴머노이드 로봇에 내장된 모션을 호출하여 동작을 재현하기 위해서 로봇 객체를 생성하고 초기화 하는 과정을 거쳐야 하며 이후 로봇 객체의 walk_forward 메소드를 호출함으로서 로봇의 기본적인 제어 동작을 구현할 수 있다. 다음의 그림 8은 SDK 파이선 API를 활용한 로봇 초기화 코드이다. unitree_sdk의 로봇 객체를 통해 로봇을 생성한 뒤 .initialize() 메서드를 호출하여 초기화를 수행하며, 초기화 완료 메시지를 출력하여 정상적으로 제어 환경이 준비되었음을 확인할 수 있다.

```
from unitree_sdk import Robot
robot = Robot()
robot.initialize()
print("로봇이 초기화되었습니다.")
```

그림 8. 로봇 초기화
Fig. 8. Robot initialization

보행은 walk_forward(speed, duration)로 전달되면 로봇은 좌, 우 각각 1STEP을 실행한다. 이는 각 로봇에 미리 저장된 walk_forward() 메소드에 해당하는 모션에 따라 다르다. 이후 stop()으로 정지된다. 이는 고수준 함수를 이용해서 안정적인 보행은 간단하게 구현 할 수 있음을 볼수 있다. 그러나 이는 지속적인 상태 모니터링을 통해 다음 동작을 제어하기 위한 알고리즘에 반영해야 하는 과제가 남게 된다. 그림 9는 휴머노이드 로봇의 보행 제어 코드를 나타낸 것이다. 이 코드는 양발 1 step의 이동 동작을 구현한 것이다.

```
robot.walk_forward(speed=0.5, duration=3)
robot.stop()
```

그림 9. 로봇 보행 제어 코드
Fig. 9. Robot walking control code

팔의 10번과 11번 조인트를 개별 제어하여 실시간 센서 등의 피드백 데이터를 통해 프로그램되어 있지 않은 모션, 즉 즉흥적인 동작을 수행하도록 하는 동작 제어는 저수준 API에 해당하는 set_joint_position() 메소드를 활용하여 개별 관절의 위치를 지정함으로써 수행된다. 그림 10은 G1 휴머노이드 로봇의 팔 동작 제어 코드이

다. 양쪽 팔 관절에 해당하는 10번 및 11번 조인트에 대해 0.5 라디안 위치로 이동시키는 예제이며, 이는 팔을 앞으로 올리는 동작을 구현하는 데 사용되었다. set_joint_position() 함수는 특정 관절 ID에 대해 목표 위치를 지정함으로써 관절의 회전을 유도한다.

```
robot.set_joint_position(joint_id=10, position=0.5)
robot.set_joint_position(joint_id=11, position=0.5)
```

그림 10. 로봇 팔 미세 제어
Fig. 10. Robot arm value control

센서 데이터에 대한 결과치 예측의 오류나 알고리즘의 오류로 인해 휴머노이드 로봇이 폭주할 수 있는 위험이 발생하지 않도록 알고리즘을 구성 하여야 한다.

센서 데이터의 IMU는 로봇의 자세, 가속도, 회전 속도 정보를 실시간으로 제공한다. 본 논문에서는 get_imu_data() 메소드로 데이터를 수신해 터미널에 출력하며 통신을 검증하였다. 수집된 데이터는 자세 추정, 균형 제어, 강화학습 피드백에 활용 가능하며, 실험에서는 정상 데이터 흐름과 실시간 반응성 여부를 확인하였다.

```
imu_data = robot.get_imu_data()
print("IMU 데이터:", imu_data)
```

그림 11. IMU 센서 데이터 수신 코드
Fig. 11. IMU sensor data reception code

get_imu_data() 함수는 로봇의 IMU 센서로부터 실시간 데이터를 수신하며, 해당 정보를 출력하는 과정을 통해 센서 통신의 정상 작동 여부를 확인할 수 있다.

4. 실험 계획 및 평가 방법

다음의 표 3은 제안 시스템의 기본 제어 기능 평가를 위하여 실험 항목 및 평가 기준을 정리한 것이다.

표 3. 실험 항목 및 평가 기준
Table 3. Experimental items and Evaluation criteria

Items	Evaluation Criteria
Walking Test	Speed, Direction Change Response Speed
Arm Movement Test	Accuracy and Reaction Time of Joint Control
Utilization of Sensor Data	Speed of Sensor Data Processing

첫째, 보행 테스트는 속도, 방향 명령에 대한 반응 속도와 이동 정확도를 평가한다. 둘째, 팔 동작 테스트는 관절 제어 정확성과 응답 지연 시간을 측정한다. 셋째, 센서 활용 실험은 IMU, 압력 센서, LiDAR 데이터를 ROS로 수집해 실시간 처리 능력과 속도를 검증한다. 이를 통해 기본 동작의 안정성을 확인하고 자율 제어 시스템 개발 기반을 마련한다.

5. 실험 결과 및 분석

5.1 실험 환경 구성

실험에서 사용된 하드웨어는 12th Gen Intel Core i7-12700 2.10 Mhz 프로세서와 GPU를 사용하지 않았으며 32GB의 RAM을 사용하였고 OS는 Ubuntu 20.04를 이용하였다. G1 휴머노이드 로봇은 WiFi를 통해 제어 PC와 동일 C Class 네트워크 밴드로 연결되었다. 파이썬은 Anaconda를 이용하여 설치하였고, 가상 환경하에서 Unitree의 SDK 및 필요한 패키지를 설치하여 알고리즘을 구현하였으며, ROS와 통신을 통해 센서 데이터를 수신하여 처리 결과 값에 따라 후속 동작을 처리할 수 있도록 구성되었다.

5.2 실험 수행 절차

다음의 표 4는 로봇 기본 제어 기능에 대한 실험 항목 및 측정 항목을 정리한 것이다.

표 4. 실험 항목 및 측정 항목
Table 4. Experimental items and measurement items

Items	Measurement items
Walking Test	Speed, Response Speed, Balance
Arm movement test	Target positioning accuracy, Response speed
Balance maintenance experiment	Balance recovery time, Compensation action
Utilization of sensor data	Data collection speed, Real-time processing performance

보행 실험은 G1 휴머노이드 로봇의 전방 이동 명령에 대한 로봇의 응답성과 속도 제어의 정확성을 측정 하였다. 실험에서는 walk_forward() 메서드 함수를 이용하여 지시한 속도로 이동 하도록 하였으며 이는 안전을 위하여 1번의 명령에 대하여 3초간 진행 되도록 하였다. 수행이 완료되고 나서 stop() 명령을 통해 G1휴머노이드 로봇은 동작을 종료하였다. 동작은 평균 0.2초 이내의 지연 시간 내에 시작되었으며, 명령 지속 시간과 실제 이동 시간 간의 오차율은 약 5% 이하로 나타났다. 다음의 그림 12는 전방으로 3초간 2회 이동하고 이어서 왼쪽 방향으로 0.3m/sec 의 속도로 2초간 이동하는 코드를 보여 주고 있다.

```
robot.walk_forward(speed=0.5, duration=3)  # 0.5m/s 속도로 3초간 전진
robot.walk_forward(speed=-0.5, duration=3)  # 0.5m/s 속도로 3초간 후진
robot.walk(direction="left", speed=0.3, duration=2)  # 왼쪽으로 이동
robot.stop()  # 정지
```

그림 12. 보행 실험 코드
Fig. 12. Walking experiment code

G1 휴머노이드 로봇의 이동 실험 결과에 대한 측정 항목은 정해진 보행 속도, 지정된 이동 거리, 도착 후 안정된 균형 유지 여부로 하였다. 측정은 시간 동기화를 위해 제어 PC와 연결된 웹 카메라를 사용하였으며, 휴머노이드 내부의 IMU 센서를 이용하여 로봇의 가속도 방향 축 값을 이용하여 분석하였다.

팔 동작 실험은 휴머노이드 로봇의 팔 관절에 대해 개별적인 위치 제어 명령을 전달하고, 해당 명령이 얼마나 정확하고 신속하게 수행되는지를 평가하기 위해 수행되었다. set_joint_position() 함수를 사용하여 양쪽 팔 관절(10번, 11번 조인트)을 목표 위치 0.5 라디안으로 설정하였다. 측정 항목은 목표 위치 도달 정확성, 응답 시간을 측정하였고, 측정 방법은 로봇 각 관절의 목표 위치 값과 실제 도달 위치를 비교하고, 명령 송신 시점부터 목표 위치 도달까지의 시간(응답 지연 시간)을 측정하였다. 관절 상태는 SDK API를 통해 주기적으로 폴링 하였다. 실험 결과, 각 관절은 평균 0.18초 내에 목표 위치에 도달하였다.

다음 그림 13은 팔 관절 실험 제어의 결과를 보여 주고, 그림 14는 팔 관절의 제어 실험 코드로 set_joint_position() 명령을 사용하여 양쪽 팔 관절을 각각 0.5 라디안 위치로 제어, 이동시키는 코드이다. 이 명령은 각각의 개별 관절의 위치를 직접 지정하여 정밀한 상지 제어 동작의 구현이 가능하게 한다.

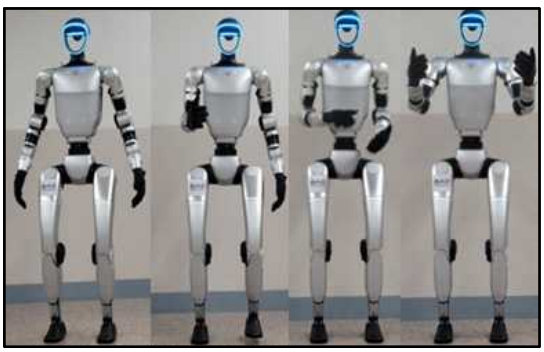

그림 13. 팔 관절 제어 실험 결과
Fig. 13. Result of arm joint control experiment

```
robot.set_joint_position(joint_id=10, position=0.5)  # 오른팔 올리기
robot.set_joint_position(joint_id=11, position=-0.5)  # 왼팔 내리기
```

그림 14. Arm joint 제어 실험 코드
Fig. 14. Arm joint control experiment code

균형 유지를 위한 실험은 외부의 물리적 힘에 대한 자세 조정 및 불규칙한 지면 조건에서 로봇이 자세를 안정적으로 유지하는가를 평가하기 위해 수행하였다. 실험은 첫째, 측면에서 손으로 밀어 외부 충격을 가한 후 자세 복원 여부 확인, 둘째, 5°, 10°, 15°로 기울어진 경사면에서 정적 자세 유지 여부를 평가하였다. 측정 항목은 균형 유지, 자세 복원 시간, 흔들림 진폭이며, 이는 는 IMU 센서를 통한 pitch/roll 변화량의 측정과 외부 카메라 촬영 기반 프레임 분석을 함께 활용하였다. 실험 결과, 충격 이후 평균 1.2초 이내에 자세가 안정화되었으며, pitch 및 roll 진폭은 각각 ±5도 이내로 유지되었다. 또한, 10° 경사면까지는 안정적인 자세 유지를 보였다. 15° 경사면에서는 자세가 흔들리며 전도 현상이 2회 중 1회 빈도로 발생하였다. 그림 15는 충격 이후 IMU pitch 데이터를 입력 받아 분석한 균형 복원 곡선율을 나타낸 것으로, 충격 직후 최대 5.5°까지 기울어졌으며, 약 3초 이내에 0°로 안정화되었다. 해당 결과를 통해 휴머노이드 로봇의 균형 유지 능력 및 자세 안정화 성능을 정량적그래프로 나타낼 수 있다.

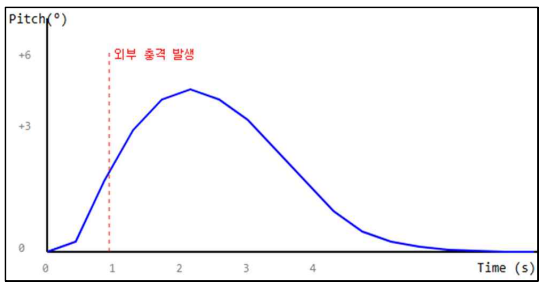

그림 15. 균형 안정화 그래프
Fig. 15. Balance restoration graph

센서 데이터 활용 실험은 로봇의 상태 및 주변 환경을 주변 환경을 실시간으로 모니터링 하고 데이터를 수집하기 위해 실행되었다. 실험에서는 로봇의 자세 제어의 안정성을 파악하기 위해서 IMU와 LiDAR 센서로부터 실시간으로 데이터를 수집하였다. 측정 항목은 센서 데이터 수집 속도, 센서 출력 정확도이며, 측정 방법은 SDK API를 통해 IMU 및 LiDAR 데이터를 주기적으로 수신하였다. ROS 로그 파일을 통해 수집 간격 및 원시 데이터를 비교할 수 있었다. 다음의 그림 16은 IMU와 Lidar의 데이터 수신을 위한 코드이다. get_imu_data()와 get_lidar_data() 함수를 활용하여 G1 휴머노이드 로봇으로부터 IMU 및 LiDAR 센서 데이터를 실시간으로 수신하는 예시이다.

```
imu_data = robot.get_imu_data()
lidar_data = robot.get_lidar_data()
print("IMU:", imu_data)
print("LiDAR:", lidar_data)
```

그림 16. IMU / LiDAR 수신 코드
Fig. 16. IMU / LiDAR reception code

5.3 실험 결과 분석

고수준 API를 이용한 보행 성능 평가 결과 정해진 시간 만큼 이동 명령을 주었을 때 로봇은 오차 범위 1% 미만의 성능으로 명령을 수행하였으며, IMU 센서와 Lidar 센서를 통해 이동 상황을 평가하였을 때 G1 휴머노이드 로봇은 장애물을 명확하게 회피하여 이동을 진행하였고, 이는 별도의 명령을 전달하거나, 하지 않았음에도 안정적인 반응을 보였다. 또한, IMU를 통해서 로봇의 보행 중 불규칙한 노면은 물론 경사진 노면에서의 보행에서도 안정적인 균형을 유지할 수 있었다. 향후 DRGB 카메라를 이용한 실시간 자율 보행과 사물 인식 후 사물에 접근하기 위한 팔동

작 제어를 위한 저수준 API까지 성공하였는데, 이는 향후 강화학습에 의한 자율적인 동작이나 자연어 대화 및 멀티모달에 따른 로봇의 자율행동 연구에 기초가 될 수 있다. 다음의 표 5는 속도 설정에 따른 보행 성능 평가 결과를 나타낸 것이다.

표 5. 속도 설정에 따른 보행 성능 평가 결과
Table 5. Results of walking performance evaluation according to speed setting

Speed setting (m/s)	Average distance traveled (m)	Average response time (sec)	Balance
0.5	1.45	0.2	Good
1.0	2.85	0.18	A little bit of shaking
1.5	4.20	0.15	Loss of balance

실험 결과에서 낮은 속도(0.5m/s)에서는 균형을 안정적으로 유지하며 동작하였으나, 속도가 증가할수록 균형 유지에 어려움을 보였다. 이는 현존하는 거의 모든 로봇들의 공통된 한계 또는 현상이라 할 수 있다. 다음의 표 6은 팔 동작에 대한 실험 결과를 나타낸 것이다.

표 6. 팔 동작 평가 결과
Table 6. Arm movement evaluation results

Targer location (rad)	Actual arrival location (rad)	Error (rad)	Response time (s)
0.5	0.48	0.02	0.12
-0.5	-0.46	0.04	0.14

균형 유지 성능 평가는 다양한 경사면 환경에서 휴머노이드 로봇이 안정적으로 자세를 유지할 수 있는지를 확인하기 위해 수행되었다. 다음의 표 7은 균형 유지 성능 평가 결과이다.

표 7. 균형 유지 평가 결과
Table 7. Balance maintenance assessment results

Angle of incline (angle)	Keep the balance	Balance restoration time (s)
5	Good	0.5
10	Good	1.2
15	Bad	-

센서 데이터 처리 성능 평가는 IMU와 LiDar만을 대상으로 수행하였고, 다음의 표 8은 센서 데이터의 평가 결과를 나타낸 것이다.

표 8. 센서 데이터 평가 결과
Table 8. Sensor data evaluation results

Sensor type	Data Processing speed (Hz)	Average Latency (ms)
IMU	100	10
LiDAR	30	133

다음 표 9는 G1 로봇과 NAO[16][17] 의 이동성능에 대한 비교표 이다.

표 9. Unitree G1 vs Aldebaran NAO 휴머노이드 로봇 비교
Table 9. Comparison between Unitree G1 and Aldebaran NAO Humanoid Robots

Comparison Item	G1	NAO
최대속도 (m/Sec)	2.0	0.095
보행응답 시간(Sec)	0.20	0.40-0.60*
연속동작 시간(Hours)	2-4	1.5

5.4 연구 성과 및 향후 응용 가능성

본 연구의 주요 성과는 첫째, 검증된 제어 성능으로 보행 명령 응답시간 0.2초, 위치 정확도 오차 5% 이내, 균형 복원시간 1.2초의 실용적 성능을 확보하고 둘째, 확장 가능한 아키텍처로 23 DOF 복합 제어, 4종 센서 실시간 융합을 통한 모듈화를 달성하였다. 셋째, 재현 가능한 연구 방법론으로 상세한 개발환경, 실험 프로토콜, 성능 지표를 제시하여 후속 연구 기반을 제공하였다. 연구에서 구축한 기본 제어 프레임워크를 기반으로 다음과 같은 단계별 확장 연구를 계획하였다. 2단계인 자율 제어 시스템 개발에서는 강화학습 기반 적응적 보행 제어 알고리즘 개발, 환경 인식 및 장애물 회피 시스템 구현, 동적 환경에서의 자율 내비게이션 성능 평가를 통해 복잡한 환경에서의 자율 이동과 상황 적응적 행동 생성을 목표로 한다. 3단계인 지능형 반려 로봇 시스템 개발에서는 자연어 처리 기반 음성 명령 인터페이스를 구현하고, 개별 사용자 맞춤형 상호작용 패턴 학습 시스템, 감정 인식 및 표현을 통한 사회적 로봇 기능 개발을 통해 일상 대화형 반려 로봇과 개인화된 서비스 제공을 달성하고자 한다. 기술적 파급효과는 다음 분야로 확산될 것으로 예상된다. 서비스 로봇 분야에서는 병원, 호텔, 상점 등에서의 고객 응대 로봇, 노인 돌봄 및 재활 보조 로봇, 교육용 상호작용 로봇으로 활용될 수 있다. 산업 로봇 분야에서는 인간-로봇 협업 제조 시스템, 위험 환경 작업용 휴머노이드 로봇, 건설 및 물류 자동화 로봇으로 응용 가능하다. 연구 플랫폼 분야에서는 휴머노이드 로봇학 교육 플랫폼, 인공지능 알고리즘 검증 테스트베드, 인간-기계 상호작용(HRI) 연구 도구로 기여할 수 있다. 특히 3단계 반려 로봇 연구는 다음과 같은 사회적 가치를 창출할 것으로 기대된다. 고령화 사회의 독거노인 돌봄 및 정서적 지원, 발달장애 아동의 사회성 발달 및 치료 보조, 반려동물 알레르기 환자를 위한 대안적 반려 솔루션, 코로나19와 같은 팬데믹 상황에서의 비대면 돌봄 서비스 등이 가능하다. 이러한 연구 확장을 통해 본 논문에서 제시한 기본 제어 프레임워크가 단순한 로봇 동작 구현을 넘어서 미래 지능형 로봇 사회의 핵심 기술로 발전할 수 있을 것으로 판단된다.

6. 결론 및 향후 연구

논문에서는 G1 휴머노이드 로봇의 기본 제어 시스템을 구축하고, Unitree SDK 중 Python API를 활용한 제어의 성공 여부와 와 명령 반응에 따른 성능을 실험적으로 분석하였다. 이는 고수준 API를 이용한 학습된 모션 또는 행동 제어와 저수준 API를 이용한 즉각적인 팔 동작의 반응과 이 동작의 수행중에 변화하는 센서 데이터의 수집을 구현하였으며, 이를 활용한 후속 동작의 지시, 또는 후속 관절제어 명령을 통한 움직임을 구현할 수 있음을 확인하였다. 이를 통해 센서 기반 상황 판단 및 동작 반영이 가능한 프레임워크가 구축되었음을 확인하였다. 본 논문에서 구축한 제어 시스템을 기반으로 강화학습 기반 자율 제어 시스템 구현, 멀티 모달 센서 융합, 음성 명령 및 자연어처리 기술을 접목한 지능형 로봇 시스템에 대한 연구가 이루어질 것으로 기대된다.

참 고 문 헌

[1] Kim, T. K., Pan, Y. H., “A Study on Humanoid Robot Trends and Development Standards by Type: Based on Leisure and Entertainment Area”, Korea Service Design Council, Vol.2024(1), pp.91-96,

2024.
[2] Giri, B., Singh, D. B., Chattu, V. K., “Aging population in South Korea: burden or opportunity?”, International Journal of Surgery: Global Health, Vol.7, No.6, e00517, 2024.
DOI: 10.1097/GH9.0000000000000517
[3] Statistics Korea, 2023 Elderly Statistics, https://kostat.go.kr/board.es?mid=a10301010000&bid=10820&list_no=427252&act=view&mainXml=Y, 2023.
[4] Kim, C. Y., Chung, S. H., “Demographic transition in South Korea: implications of falling birth rates”, Clinical and Experimental Pediatrics, Vol.67, No.10, pp.498-509, 2024.
DOI: 10.3345/cep.2023.01599
[5] Tong, Y., Liu, H., Zhang, Z., “Advancements in humanoid robots: A comprehensive review and future prospects”, IEEE/CAA Journal of Automatica Sinica, Vol.11, No.2, pp.301-328, 2024.
DOI: 10.1109/JAS.2023.124140
[6] Gu, Z., et al., “Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning”, arXiv preprint arXiv:2501.02116. pp.1-39, 2025.
DOI: 10.48550/arXiv.2501.02116
[7] Assad-Uz-Zaman, M., Islam, M. R., Rahman, M. H., Wang, Y. C., McGonigle, E., “Kinect controlled NAO robot for telerehabilitation”. Journal of Intelligent Systems, Vol.30, No.1, pp.224-239, 2020. DOI: 10.1515/jisys-2019-0126
[8] Ramkumar, A., Sasidaran, K., Yogesh, M., Sureshkumar, R., “Autonomous rover controlled by NAO”, International Journal of Engineering and Advanced Technology (IJEAT), Vol.8, No.6S3, pp.226-230, 2019. DOI: 10.35940/ijeat.F1295.0986S319
[9] Sher, A. J., Huzaifa, U., Li, J., Jain, V., Zurawski, A., LaViers, A., “An embodied, platform-invariant architecture for connecting high-level spatial commands to platform articulation”, Robotics and Autonomous Systems, Vol.119, pp.263-277, 2019. DOI: 10.1016/j.robot.2019.07.006
[10] Bolotnikova, A., Gergondet, P., Tanguy, A., Courtois, S., Kheddar, A., “Task-space control interface for softbank humanoid robots and its human-robot interaction applications”, In 2021 IEEE/SICE International Symposium on System Integration (SII), IEEE, pp.560-565, 2021. DOI: 10.1109/IEEECONF49454.2021.9382685
[11] Novotny, G., Morales-Alvarez, W., Smirnov, N., Olaverri-Monreal, C.. “Development of a ros-based architecture for intelligent autonomous on demand last mile delivery”, In International Conference on Computer Aided Systems Theory, Cham: Springer Nature Switzerland, pp.337-344, 2023.
DOI: 10.1007/978-3-031-25312-6_39
[12] Angelini, F., Della Santina, C., Garabini, M., Bianchi, M., Bicchi, A., “Control architecture for human-like motion with applications to articulated soft robots”, Frontiers in Robotics and AI, Vol.7, pp.1-17, 2020. DOI: 10.3389/frobt.2020.00117
[13] Radosavovic, I., Xiao, T., Zhang, B., Darrell, T., Malik, J., Sreenath, K., “Real-world humanoid locomotion with reinforcement learning”, Science Robotics, Vol.9, No.89, eadi9579, 2019. DOI: 10.1126/scirobotics.adi9579
[14] Novotny, G., Morales-Alvarez, W., Smirnov, N., Olaverri-Monreal, C., “Development of a ros-based architecture for intelligent autonomous on demand last mile delivery”, In International Conference on Computer Aided Systems Theory, Cham: Springer Nature Switzerland, pp.337-344, 2022.
https://arxiv.org/pdf/2305.18276
[15] Stasse, O., Giraud--Esclasse, K., Brousse,

E., Naveau, M., Régnier, R., Avrin, G., Souères, P., “Benchmarking the hrp-2 humanoid robot during locomotion”, Frontiers in Robotics and AI, Vol.5, pp.1-17, 2018. DOI: 10.3389/frobt.2018.00122

[16] Syamimi Shamsuddin, Luthffi Idzhar Ismail, Hanafiah Yussof, Mohd Saiful Bahari Shaari, “Humanoid robot NAO: Review of control and motion exploration”, In 2011 978-1-4577-1642-3/11/$26.00 ©2011 IEEE, pp. 511-516, DOI: 10.1109/ICCSCE.2011.6190579

[17] Gouaillier, David, et al. “The nao humanoid: a combination of performance and affordability“, arXiv preprint arXiv:0807.3223 (2008).

저 자 소 개

황동하(Dong-Ha Hwang)

2011.2 한국방송통신대학교 법학과 학사
2022.8 우송대학교 IT융합 석사
2024.3 공주대학교 컴퓨터공학과 박사과정
2023.3-현재 우송대학교 IT융합 겸임교수
<주관심분야> 휴머노이드 로봇, 인공지능, 인공지능 에이전트

김재웅(Jae-Woong Kim)

1983.2 중앙대학교 전자계산학과 학사
1988.2 중앙대학교 컴퓨터공학과 석사
2002.2 대전대학교 컴퓨터공학과 박사
1992.8-현재 국립공주대학교 천안공과대학 컴퓨터공학부 교수
<주관심분야> 소프트웨어공학, 인공지능 시스템, 멀티미디어공학, 빅데이터

김동현(Dong-Hyun Kim)

1986.2 중앙대학교 전기공학과 학사
2005.2 공주대학교 컴퓨터멀티미디어공학과 석사
2010.2 공주대학교 컴퓨터공학과 박사
2021.9-현재 나사렛대학교 IT인공지능학부 교수
<주관심분야> 인공지능, 로봇 제어, 멀티미디어 시스템