ISSN 1229-800X [Print] / ISSN 2586-7792 [Online]
The Transactions of the Korean Institute of Electrical Engineers, vol. 71P, no. 3, pp. 169~174, 2022
https://doi.org/10.5370/KIEEP.2022.71.3.169

Networked Control System for Self-Balancing Robot based on Public Cloud

퍼블릭 클라우드 기반 셀프 밸런싱 로봇 네트워크 제어 시스템

Sewan Gu · Dongweon Yoon

구세완* · 윤동원†

Abstract

In a networked control system, a controlled model and a controller are remotely operated via network communication. In this paper, we propose a public cloud-based real-time networked control system and demonstrate the feasibility of its real-time control performance by building a self-balancing robot and conducting experiments with it. We first build the self-balancing robot as the controlled model and implement the controller into a docker container for the networked control system. In addition, we utilize the google cloud platform as the computing and networking resources to establish Kubernetes platform, which operates the control system. Through the experiments, we show the self-balancing robot is controlled and maintains its balance in the cloud-based networked control system.

Key Words

Networked Control System, Self-Balancing Robot, Public Cloud, ROS2, Container, Docker, Kubernetes, Google Cloud Platform

1. 서 론

네트워크 제어 시스템(Networked Control System)은 통신 네트워크를 통해 제어 대상 모델과 원격 제어기를 연결하여 원격 제어기가 모델의 센서값을 수신하고 제어 연산을 수행한 후 제어 명령을 모델에 전송하는 일종의 폐루프 제어 시스템이다 [1]. 클라우드 기술의 발전으로 네트워크 제어기를 클라우드 상에서 구현하여 실시간 제어를 실현하고 클라우드를 활용하여 로봇과 자동화 장비를 연결하는 연구가 보고되고 있다 [2]-[4].

이 때, 클라우드 적용을 위한 핵심 기술 중 하나는 가상화 기술로, 컴퓨터 자원을 공유하지만 기능별로 독립적인 운영체제를 유지할 수 있는 도커 컨테이너(Docker Container) 방식이 있다. 이러한 도커 컨테이너 기술은 로컬에서만 운영했던 많은 기술과 응용들을 클라우드에서 구현하여 분산처리, 원격 관리, 자동 업데이트, 장애 허용(Fault Tolerance) 등의 자동화 서비스를 가능하게 한다 [5].

본 논문에서는 클라우드 기술을 활용한 실시간 네트워크 제어 시스템을 제안하고, 퍼블릭 클라우드(Public Cloud) 기반의 실제 환경과 로봇 시스템에 적용하여 구현한 후 성능을 검증한다. 성능을 검증하기 위한 제어 대상 모델로 2개의 바퀴(Wheel)을 갖는 셀프 밸런싱 로봇(Two-Wheeled Self-Balancing Robot)을 제작한 후 제안한 네트워크 제어 시스템의 구동 성능을 검증한다.

셀프 밸런싱 로봇의 자세 제어를 위해서는 비례미분 제어기를 조정하여 적용하며, 클라우드 네트워크 제어 시스템 을 실행시키고 운영하는 플랫폼으로는 쿠버네티스를 채택한다 [6]. 이를 통해 향후 서비스로서의 제어기 (Control as a Service)를 고려할 수 있으며, 재사용성 및 제어기 조정도 매우 용이하게 할 수 있을 것이다 [7].

한편, 퍼블릭 클라우드와 네트워크를 통해 제어되는 셀프 밸런싱 로봇에서 안정성 및 성능에 영향을 줄 수 있는 주요 인자 중 하나는 지연(Latency)이다. 특히, 네트워크를 통해 송수신되는 센서 측정값 및 제어 명령은 64 바이트(byte) 미만으로 대역폭에 미치는 영향은 무시할 수 있다. 따라서, 본 논문에서는 지연을 성능 지표의 주요 인자로 설정하고 실험을 진행한다. 이를 통해 퍼블릭 클라우드 및 네트워크의 지연에 민감한 제어 시스템을 실증하여 클라우드 기반 실시간 네트워크 제어 시스템 구축이 가능함을 보인다. 본 논문에서 얻어지는 실증 결과를 바탕으로 향후 지연에 대하여 다양한 제어기 설계 기법을 클라우드 기반 제어 시스템에 도입할 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 제어 대상이 되는

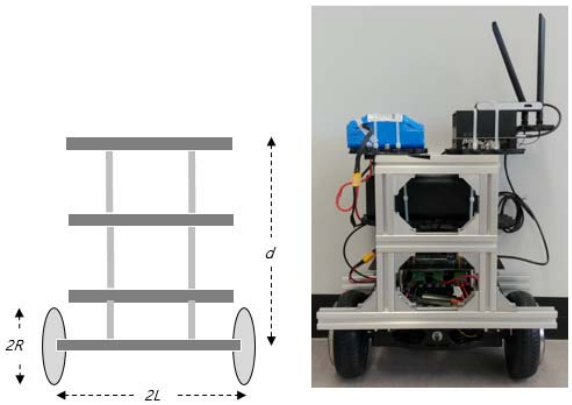
*Author : Dept. of Electronics and Computer Engineering, Hanyang University, Korea
E-mail:swgu@hanyang.ac.kr
https://orcid.org/0000-0001-9547-7626
† Corresponding Author : Dept. of Electronic Engineering, Hanyang University, Korea
E-mail:dwyoon@hanyang.ac.kr
https://orcid.org/0000-0001-9631-3500
Received : Aug. 19, 2022 **Revised :** Aug. 25, 2022 **Accepted :** Aug. 28, 2022

셀프 밸런싱 로봇의 모델링 및 제작한 시스템 구성에 대하여 설명하고, 3장에서 클라우드 기반 네트워크 제어 시스템을 제안한다. 4장에서는 실험을 통해 클라우드 네트워크 제어 시스템 실증 결과를 보이며, 5장에서 결론을 맺는다.

2. 셀프 밸런싱 로봇 모델링 및 시스템 구성

2.1 모델 제작 및 제원

본 논문에서는 제안하는 시스템에 대하여 실현 가능성에 대한 실증을 보이기 위해서 먼저, 실제 로봇을 제작한다. 그림 1에는 제작한 셀프 밸런싱 로봇 모델과 실제 구현한 실물 사진을 나타내었다. 그림 1과 같이 제작된 로봇은 바퀴, 모터 구동부(Driver), 센서부, 배터리, 응용 프로세서 (Application Processor) 등으로 구성되며, 알루미늄 샷시(Chassis)로 전체 조립한 로봇의 유동이 없도록 고정하였다.



(a) Robot Model (b) Picture of the Implementation

그림 1 셀프 밸런싱 로봇
Fig. 1 Self-Balancing Robot

셀프 밸런싱 로봇 모델링 및 제어는 많은 논문에서 다루어져 왔는데 본 논문에서는 [8]에서 제안한 방법을 기반으로 수학적 모델링을 수행한다. 본 논문에서 실험에 사용하기 위하여 제작한 셀프 밸런싱 로봇의 하드웨어 기구 사양을 표 1에 정리하여 나타내었다. 실제 관성값 (Inertia value)은 측정이 불가하여 [8]의 내용을 참조하였으며, 제어기의 비례이득과 미분이득을 조정하여 보정하는 방법을 사용하였다.

표 1 제작된 주요 하드웨어 기구 사양
Table 1 Main Specifications of Hardware and Fabrication

기호	값	설명	단위
ms	14.3	몸체 무게	kg
d	0.4	축과 무게중심 거리	m
g	9.8	중력가속도	m/sec2
mc	2.35	휠 무게	kg
I2	3.679·10-3	N3 방향 관성	kg·m2
I3	28.07·10-3	N2 방향 관성	kg·m2
R	0.0825	휠 반지름	m
L	0.18	바퀴 사이 거리/2	m

2.2 클라우드 기반 네트워크 제어 시스템 구성

클라우드 기반 네트워크 제어 대상으로 제작한 셀프 밸런싱 로봇 시스템의 구성은 그림 2 와 같다. 먼저, 로봇의 위치 측정을 위해서는 [9]와 같이 관성 측정 장치 (Inertial Measurement Unit: IMU)를 장착하여 직각을 기준으로 기울어진 Pitch (각도)와 Pitch rate (각속도) 값을 측정하고, 이를 범용으로 많이 사용하는 아두이노 메가 (Arduino Mega)를 통하여 측정값을 수집한다. 이렇게 측정하고 획득된 값을 통신 네트워크를 통해 퍼블릭 클라우드 상에 있는 제어기에 송신하여 셀프 밸런싱 로봇에 피드백 되어 입력될 제어 명령을 계산한다.

이 때, 모터 2개를 직접 구동하는 제어 보드에는 56V까지 지원하는 ODriver v3.6 을 사용하였다 [10]. 제작된 셀프 밸런싱 로봇은 클라우드 상의 원격 제어 시스템으로 부터 제어 입력을 수신하고, 이를 젯슨 자비어 (Jetson Xavier)를 통해 직접 모터 구동부에 입력하여 모터를 제어한다 [11].

한편, 하드웨어 운용을 위한 소프트웨어 구성은 다음과 같다. 먼저, 셀프 밸런싱 로봇의 두뇌라 할 수 있는 응용 프로세서의 운용체제는 범용성을 위하여 리눅스 우분투 (Ubuntu) 20.04 를 사용하였다. 셀프 밸런싱 로봇과 서버 사이의 센서 메시지 송수신, 제어 명령어 송수신은 ROS2 플랫폼을 사용하였으며 [12], 셀프 밸런싱 로봇과 클라우드 서버는 일반 와이파이 (WiFi)를 통해 공중망으로 연결되도록 설계하였다. 셀프 밸런싱 로봇이 연결된 서브 네트워크와 클라우드 서버가 연결된 서브 네트워크는 서로 다른 네트워크로서 일반적으로 ROS2 통신을 할 수 없다. 이를 위해 본 논문에서는 오픈소스인 클라우드 브리지(Cloud Bridge)를 이용하여 셀프 밸런싱 로봇이 연결된 서브 네트워크와 클라우드 서버가 연결된 서브 네트워크 간에 ROS2 메시지를 송수신 할 수 있도록 구현하였다 [13].

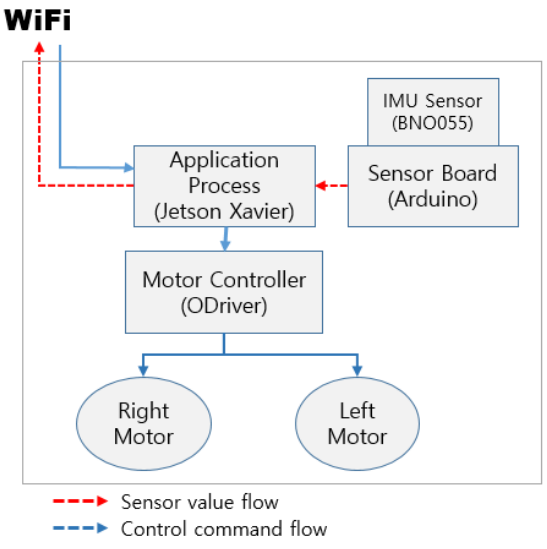


그림 2 셀프 밸런싱 로봇 시스템 구성도
Fig. 2 Block Diagram of Self-Balancing Robot System

3. 클라우드 기반 네트워크 제어 시스템 구현

3.1 제어기 구현

클라우드 상의 비례미분 제어기를 고려한 셀프 밸런싱 로봇의 폐루프 피드백 제어 시스템은 그림 3과 같이 나타낼 수 있다. 그림 3에서 기준 입력 신호 *r(t)*는 0 으로 가정하여 로봇으로 측정되어 받은 센서값을 그대로 오차(error)로 가정하고 비례이득(*Kp*) 값과 미분이득(*Kd*) 값을 조정한다. 또한, 퍼블릭 클라우드에서 제어기를 운영하기 위해 다음과 같은 두가지를 추가로 구현한다. 첫 번째는 쿠버네티스 시스템을 직접 퍼블릭 클라우드에 구축하고, 두 번째는 여기에 제어기를 실행하기 위해 우분투 20.04를 운영체제로 하고 Foxy를 ROS2 버전으로 하여 제어기 모듈을 도커 컨테이너 이미지로 구성한다. 제어기를 포함한 제어 시스템에 대해서는 다음 절에서 보다 상세히 다루도록 한다.

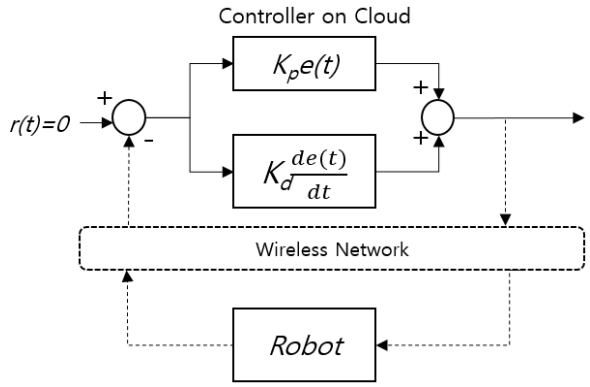


그림 3 클라우드 기반 폐루프 피드백 제어 시스템 구성도
Fig. 3 Block Diagram of Cloud-based Closed-loop Feedback Control System

3.2 클라우드 기반 네트워크 제어 시스템

그림 4에는 클라우드 기반 네트워크 제어 시스템의 대략적인 구성도를 나타낸다. 그림 4에서 퍼블릭 클라우드로 구글 클라우드 플랫폼 (Google Cloud Platform: GCP)을 활용하였으며, 해당 클라우드 서버에 컴퓨트 인스턴스 (Compute Instance)를 생성하여 직접 쿠버네티스 플랫폼을 구축한다.

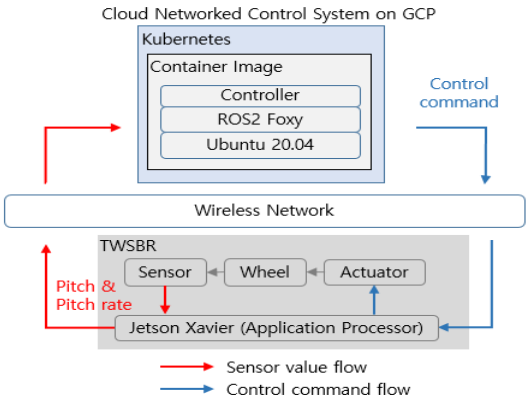


그림 4 클라우드 기반 네트워크 제어 시스템 구성도
Fig. 4 Block Diagram of Cloud-based Networked Control System

그림 5에는 구글 클라우드의 컴퓨트 인스턴스를 구성하고, 이를 쿠버네티스로 구축한 결과를 캡처하여 나타내었다. 그림 5에서 controller-1 은 마스터 노드(Master Node)로 쿠버네티스 내부 컨테이너 운영 (Orchestration)과 외부 통신을 관리하는 마스터 역할을 하며, worker-1은 워커 노드(Worker Node)로 쿠버네티스에서 실제 컨테이너를 실행한다. 마스터 노드와 워커 노드는 클라우드 플랫폼 내에서 독립적인 컴퓨팅 자원으로 각각의 인터넷 주소 (Internal IP)를 클라우드 플랫폼으로부터 할당 받는다.

성능 검증 실험을 위한 컴퓨팅 자원 2개의 하드웨어 사양은 중앙 연산 처리 장치의 코어 2개, 속도는 2.2GHz, RAM은 8GB에 해당하는 머신 타입(Machine type)으로 e2-standard-2 를 설정하였다. 또한, 컴퓨팅 자원의 지역적인 위치는 구글 클라우드 서비스 지역(Zone) 내에서 실험하는 곳과 가장 가까운 서울 지역을 선정하여 해당 지역인 asia-northeast3-a 로 설정한다.

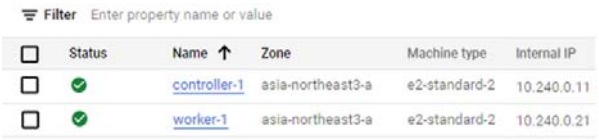


그림 5 구글 클라우드 플랫폼의 쿠버네티스를 구성하는 컴퓨트 인스턴스
Fig. 5 Compute Instance on Google Cloud Platform for Kubernetes

그림 6에는 구글 클라우드 컴퓨팅 자원 상에서 운영되는 네트워크 제어 시스템의 소프트웨어 구조 블록도를 나타내었는데 그림 6에서 클라우드 기반 제어 시스템 운영을 위해 필요한 구성은 다음과 같다. 컨테이너 레지스트리는 제어 시스템의 컨테이너 이미지를 버전별로 저장하고 관리하며, 쿠버네티스 플랫폼에서 지정된 버전의 다운로드 요청에 응답하여 버전을 선택하여 다운로드 할 수 있도록 한다. 쿠버네티스 플랫폼은 마스터 노드와 워커 노드로 구성되며, 마스터 노드는 제어기 컨테이너와 클라우드 브리지 컨테이너의 시작, 종료, 상태 관리 등을 수행하고 제어기 이미지와 클라우드 브리지 이미지를 지정된 워커 노드에 실행하도록 요청한다. 워커 노드는 마스터 노드의 요청에 따라서 제어기와 클라우드 브리지 이미지를 컨테이너 레지스트리로 부터 다운 받아 직접 실행하는 역할을 한다.

이후, 이들 컨테이너가 실행되면서, 각각 제어기 파드 (Pod)와 클라우드 브리지 파드가 생성되고, 각 파드는 내부 인터넷 주소 (Pod IP)를 부여받으며, 통신을 위해 인터넷 주소를 사용한다. 로드 밸런서는 쿠버네티스 내부와 외부에 각각 인터넷 주소와 포트를 부여하여, 내부 데이터 송신과 외부 데이터 수신을 통제하며, 셀프 밸런싱 로봇과 직접 통신한다. 특히, 직접 데이터를 처리하여 제어기에 ROS2 메시지로 송수신하는 기능은 클라우드 브리지에서 담당하며, 이때, 센서와 제어 명령의 데이터 흐름은 제어기 – 클라우드 브리지 – 로드 밸런서 – 셀프 밸런싱 로봇과 같은 흐름을 갖는다.

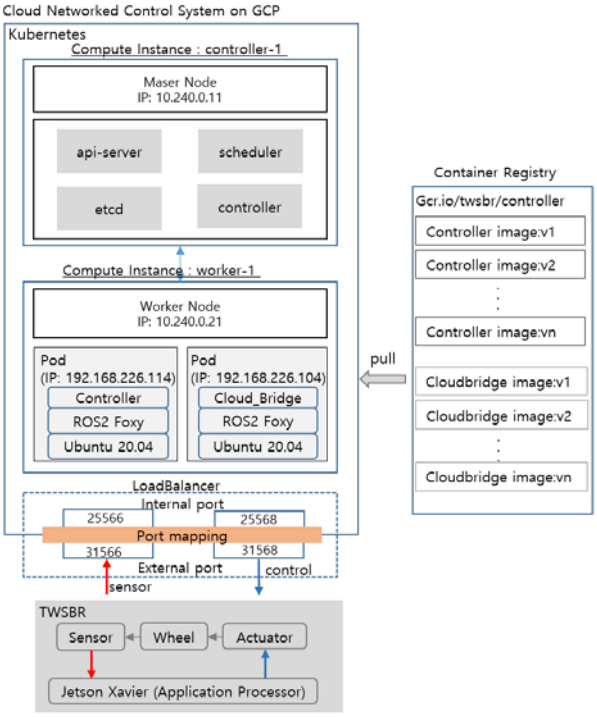


그림 6 퍼블릭 클라우드 기반 네트워크 제어 시스템 상세 구조도
Fig. 6 Detailed Block Diagram of Public Cloud-based Networked Control System

4. 성능 실험 및 결과

4.1 실험 조건 및 방법

본 절에서는 제안한 방식에 대하여 제작한 셀프 밸런싱 로봇을 대상으로 상세 실험 내용과 결과에 대해 기술한다. 본 논문에서 제작한 셀프 밸런싱 로봇은 와이파이를 활용하여 공중망에 연결하여, 쿠버네티스 로드 밸런서에서 제공한 인터넷 주소와 포트를 통해 네트워크 제어 시스템에 접속한다.

그림 7에는 셀프 밸런싱 로봇과 제어 시스템 사이의 센서 및 제어 명령어 송수신 과정에 대한 시간 절차를 나타내었다. 여기서, 제어기에서 센서값을 받아 제어 명령을 계산하는 시간 $c_{sn} - c_{an}$ 은 CPU, RAM 등의 클라우드 연장 자원을 충분히 사용하기 때문에 네트워크의 시간 지연에 비하여 매우 작아서 실험에 큰 영향을 미치지 않는다고 가정한다. 그림 7에서 송신측 센서값을 측정해서 제어기로 송신하는 주기 $r_{s2} - r_{s1}$ 은 10ms (100Hz) 이며, 이것은 실제 셀프 밸런싱 로봇의 센서에서 피치값을 측정하는 주기와 같도록 설정한다. 마찬가지로 클라우드 상의 제어기도 $r_{c2} - r_{c1}$ 의 주기를 갖고 있으며, 실험에서는 센서 주기와 같이 10ms 로 설정한다. 그림 7에서 $r_{a1} > r_{s2}$ 인 경우에는, 로봇에서 센서값을 측정하여 원격 제어기에 송신하고, 그에 해당하는 제어 명령을 수신하는 왕복시간 (Round Trip Time)이 10ms 를 초과한다는 의미이다. 이러한 네트워크 지연이 지속되어 누적된다면, 로봇의 동작이 불안정해지고, 회복 불가능한 상태가 지속되어 최종적으로는 균형을 잃고 넘어지게 된다. 따라서, 이를 방지하기 위해서는 지연에 따른 센서값을 예측하여 제어 명령을 계산하여 셀프 밸런싱 로봇에 송신하는 절차가 있어야 하며 [14], 본 논문에서는 비례이득과 미분이득을 상황에 맞추어 휴리스틱(Heuristic)하게 조정한다.

보다 현실적인 실험을 위해서 로봇과 제어기가 위치한 서버 사이 시간 동기화를 위해 네트워크 타임 프로토콜 (Network Time Protocol: NTP)을 서버 및 셀프 밸런싱 로봇에 설치하고, NTP 서버는 asia.pool.ntp.org 로 세팅하여 같은 위치의 시간으로 동기화 한다. 셀프 밸런싱 로봇이 위치한 장소에서 퍼블릭 클라우드 기반 제어 시스템까지의 통신 속도를 측정하기 위해서 핑 (ping)을 통하여 64 바이트 패킷의 왕복 시간을 측정하여 왕복 시간을 기반으로 비례이득과 미분이득을 휴리스틱 하게 조정한다. 특히, ROS2 플랫폼에서 제공하는 API (Application Programming Interface)를 활용하여 피치, 피치율, 토크의 주기를 측정한다. 또한 셀프 밸런싱 로봇의 균형 유지 여부를 실증을 통해 확인하고, 클라우드 네트워크 제어 시스템의 실현 가능성을 확인한다.

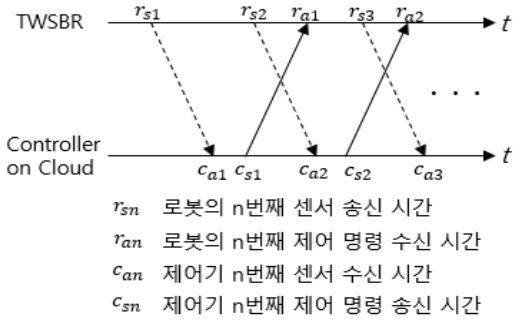


그림 7 센서와 제어 명령 송수신 시간 절차
Fig. 7 Time Sequence for Communication of Sensor value and Control command

4.2 실험 결과

셀프 밸런싱 로봇에서 클라우드 네트워크 제어 시스템까지의 왕복 시간을 측정하기 한 결과, 530개의 패킷(packet)을 송신했을 때, 100% 수신율을 보여 패킷 손실은 0% 였다. 표 2에는 왕복 시간에 대하여 최대, 최소, 평균 시간 측정값을 나타내었다. 표 2에서와 같이, 왕복 시간의 평균은 5.667ms 로서 셀프 밸런싱 로봇 센서 송신 주기와 제어 명령어의 송신 주기 10ms 보다 작아서, 셀프 밸런싱 로봇이 균형을 유지하는 조건을 만족함을 알 수 있다.

이를 보다 더 자세히 살펴보기 위하여 그림 8에는 셀프 밸런싱 로봇으로부터 클라우드 제어 시스템까지의 각 패킷 별 왕복 시간을 나타내었다. 그림 8에서와 같이 총 530개 패킷 중에서 10ms를 초과하는 패킷은 14개로 약 2.4%였으나, 10ms 이하의 패킷이 516개로 약 97.6%에서 셀프 밸런싱 로봇이 균형을 유지하는 조건을 만족함을 확인할 수 있다. 본 논문에서는 이러한 실험 환경에 적합하도록 최대한 휴리스틱하게 제어기 이득을 조정하였다.

표 2 왕복 시간 측정 결과 (ms)
Table 2 Round Trip Time of Measurements (ms)

	min	avg.	max	deviation
왕복시간	3.032	5.667	24.585	2.111

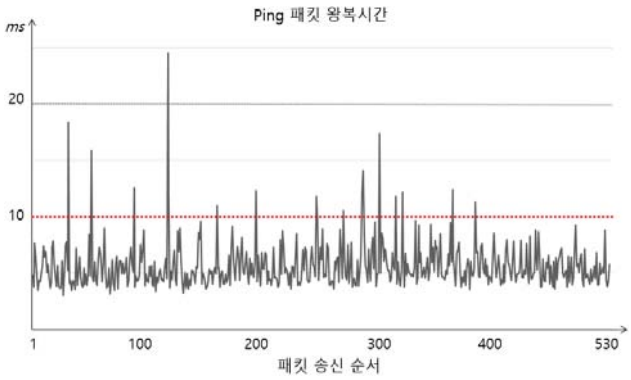


그림 8 패킷별 왕복 시간
Fig. 8 Round Trip Time for Packet

셀프 밸런싱 로봇과 클라우드 네트워크 제어 시스템간의 연결을 통해 셀프 밸런싱 로봇 균형 유지를 제어하는 실제 실험 시간은 총 530초 동안 진행 되었다. 셀프 밸런싱 로봇이 작동 시작부터 약 175초까지 균형을 잘 유지함을 확인한 후 로봇에 인위적으로 충격을 가해서 고의적인 외란을 발생시켰는데, 이 때 약 2초 동안 로봇에 앞뒤 진동이 있었으나, 이후 바로 균형을 유지하며 530초 동안의 실험을 성공적으로 마쳤다.

그림 9에는 시간별 피치값과 피치율에 대한 결과를 나타내었다. 그림 8에서 피치와 피치율이 로봇에 인위적으로 충격을 가해서 고의적인 외란을 발생시킨 175초 부근에서 외부 충격으로 크게 변화하는 것을 볼 수 있다. 한편 그림 9에는 모터에 가해지는 토크를 시간별로 측정한 결과를 나타내었다. 그림 10에서도 같은 이유로 175초 부근에서 토크값이 크게 변화하는 것을 볼 수 있는데 이는 셀프 밸런싱 로봇이 앞뒤로 크게 진동하는 것에 대해서 균형을 유지하기 위해 모터에 가해지는 힘이 증가했기 때문이다.

한편, 셀프 밸런싱 로봇이 피치값와 피치율값을 송신하는 주기는 평균 10.15ms (98.5Hz) 이었고, 셀프 밸런싱 로봇에서 수신하는 토크값의 주기는 평균 10.10ms (99Hz)로 측정되었다. 따라서 센서값의 전송 주기 보다 모터의 전송주기가 더 작기 때문에 특이한 통신상의 외란이 없다면 퍼블릭 클라우드 기반 네트워크 제어 시스템은 충분히 실현가능 할 것이다.

위에서 살펴본 바와 같이, 본 논문에서 제작한 셀프 밸런싱 로봇과 구글 클라우드 기반 네트워크 제어 시스템의 실험 결과를 통해 셀프 밸런싱 로봇이 클라우드 네트워크 제어 시스템을 통해서 균형을 유지하며 제어 가능한 것을 확인 할 수 있다.

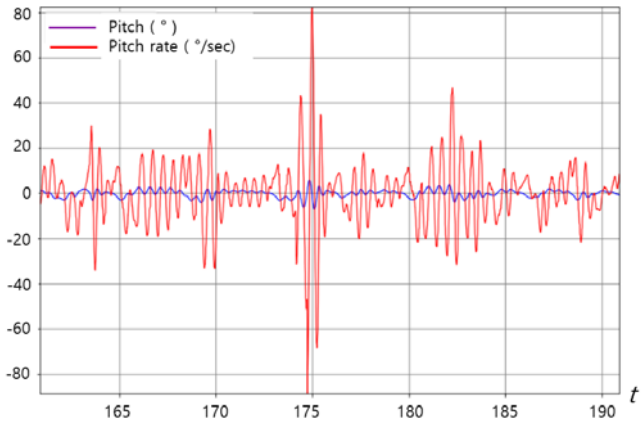


그림 9 피치(pitch), 피치율(pitch rate)의 시간별 측정
Fig. 9 Pitch(°) and Pitch Rate(°/sec) according to Time (ms)

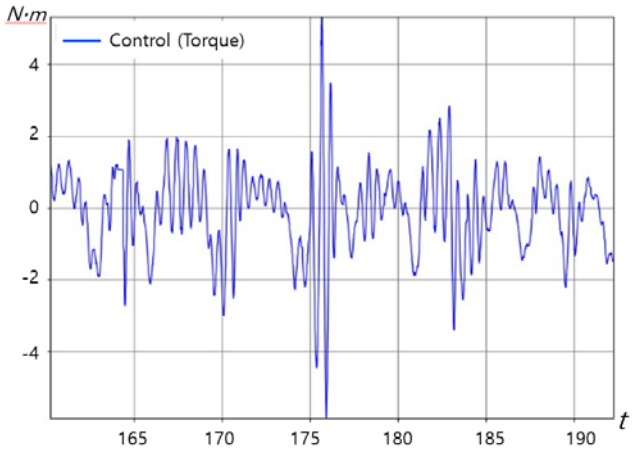


그림 10 모터에 가해지는 토크의 시간별 측정
Fig. 10 Torque (N·m) according to Time (ms)

5. 결 론

본 논문에서는 클라우드 기반 네트워크 제어 시스템을 제안하고 셀프 밸런싱 로봇 제어 실험을 통해 타당성을 살펴보았다. 먼저, 2개의 바퀴, 모터 구동부, 센서부, 응용 프로세서로 구성되는 셀프 밸런싱 로봇을 제작하였다. 이후, 클라우드 기반 제어 시스템 구성을 위해 구글 클라우드 플랫폼을 활용하여 2개의 컴퓨트 인스턴스를 생성하고, 이를 통해 쿠버네티스 플래폼을 구축하였다. 제어기와 클라우드 브리지를 도커 컨테이너 이미지로 만들어 쿠버네티스 플랫폼에서 실행시키고, 로드 밸런서를 만들어 셀프 밸런싱 로봇과 통신을 할 수 있도록 설계하였다.

성능 검증용 실험을 위해 네트워크 타임 프로토콜을 클라우드 서버와 셀프 밸런싱 로봇에 설치하여 동기화를 수행했으며, 클라우드 네트워크 제어 시스템과 연결을 통해 셀프 밸런싱 로봇의 균형을 제어하는 실험을 진행하였다. 실험 결과 측정된 클라우드 네트워크 제어 시스템에서 제어를 위해 송신하는 토크값의 주기가 셀프 밸런싱 로봇의 피치값과 피치율값의 주기

가 보다 작아 안정적으로 제어 가능함을 확인할 수 있었으며, 실험 시간 전체에 걸쳐 클라우드 네트워크 제어 시스템에 연동된 셀프 밸런싱 로봇이 균형을 유지함을 확인하였다. 본 논문에서의 클라우드 기반 네트워크 제어 시스템을 통해 셀프 밸런싱 로봇 제어가 가능함을 보임으로써 실시간의 폐루프 네트워크 제어 시스템의 향후 여러 응용 시스템에 적용 가능성을 확인 할 수 있었다.

본 논문에서는 클라우드 기반 네트워크 제어 시스템을 셀프 밸런싱 로봇에 적용하여 그 가능성을 살펴보았다. 본 논문에서 제안한 방식이 실제 적용되기 위해서는 앞으로 다양하게 변화하는 지연을 고려한 강인한 제어기를 설계하여 안정적인 폐루프 제어 시스템 구성이 추가적으로 요구된다. 또한, 장애 허용 기능의 클라우드 네트워크 제어 시스템 설계가 추가적으로 이루어진다면 쿠버네티스 플랫폼을 구성하는 마스터 노드와 워커 노드에 장애가 발생하거나 컨테이너 이미지를 구동하는 파드에 장애가 발생하는 경우에도 안정적인 제어 시스템 운영이 가능할 것이다.

References

[1] N.J. Ploplys, P.A. Kawka, A.G. Alleyne, “Closed-loop control over wireless networks,” IEEE Control Systems Magazine, vol. 24, Issue 3, pp. 58-71, Jun. 2004, DOI: 10.1109/MCS.2004.1299533.

[2] Liang Ma; Yuanqing Xia; Yasir Ali; Yufeng Zhan, “Engineering problems in initial phase of cloud control system,” 2017 36th Chinese Control Conference (CCC), Jul, 2017, DOI: 10.23919/ChiCC.2017.8028603.

[3] Ben Kehoe; Sachin Patil; Pieter Abbeel; Ken Goldberg, “A Survey of Research on Cloud Robotics and Automation,” IEEE Trans. on Automation Science and Engineering, vol. 12, Issue 2, pp. 398-409, April 2015, DOI: 10.1109/TASE.2014.2376492.

[4] S. Gu, S. Kang, W. Jeong, H. Moon, H. Yang, Y. Kim, “Validation of Cloud Robotics System in 5G MEC for Remote Execution of Robot Engines,” Journal of Korea Robotics Society, vol. 17, pp.118-409, Jun. 2022, DOI:/10.7746/jkros.2022.17.2.118.

[5] Containers & Docker: Emerging roles & future of Cloud technology, 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 804-807, Jul. 2016, DOI: 10.1109/ICATCCT.2016.7912109.

[6] Kubernetes, [Online] http://kubernetes.io, Accessed: Aug. 18, 2022.

[7] J. A. Bigheti, M. M. Fernandes, and E. P. Godoy, “Control as a Service: A Microservice Approach to Industry 4.0,” 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), Naples, Italy, 2019, DOI: 10.1109/METROI4.2019.8792918.

[8] Y. Kim, S. Kim and Y. KWAK, “Dynamic Analysis of a Nonholonomic Two-Wheeled Inverted Pendulum Robot,” Journal of Intelligent and Robotic Systems, vol 44, pp. 25-46, 2005, DOI: 10.1007/s10846-005-9022-4.

[9] BNO055 [Online] https://www.bosch-sensortec.com/ products/smart-sensors/bno055, Accessed: Aug. 11, 2022.

[10] ODriver [Online] https://odriverobotics.com/shop/ odrive-v36, Accessed: Aug. 11, 2022.

[11] Jetson AGX Xavier [Online] https://developer. nvidia.com/embedded/jetson-agx-xavier-developer-kit, Accessed: Aug. 12, 2022.

[12] ROS2 Platform [Online] https://docs.ros.org/en/foxy/ Installation.html, Accessed: Aug. 12, 2022.

[13] S, Kang, cloud_bridge, [Online] https://github.com/lge-ros2/cloud_bridge, Accessed: Aug. 18, 2022.

[14] Z. Music, F. Molinari, S.Gallenmüller, O. Ayan, S. Zoppi, W. Kellerer, G. Carle, T. Seel and Jörg Raisch, “Design of a Networked Controller for a Two-Wheeled Inverted Pendulum Robot,” IFAC, vol. 52, Issue 20, pp. 169-174, 2019, DOI: 10.1016/j.ifacol.2019.12.153.

저자소개

구세완 (Sewan Gu)

1994년 2월 한양대학교 전자통신공학(공학사)
1996년 2월 한양대학교 전자통신공학(공학석사)
1997~2000년 2월 한국철도기술연구원
2022년 현재 : 한양대학교 전자컴퓨터통신공학과 박사과정
2000년~현재 LG전자 책임연구원

윤동원 (Dongweon Yoon)

1989년 2월 한양대학교 전자통신공학과(공학사)
1992년 2월 한양대학교 전자통신공학과(공학석사)
1995년 8월 한양대학교 전자통신공학과(공학박사)
2022년 현재 : 한양대학교 융합전자공학부 교수