IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 April 2023, accepted 22 April 2023, date of publication 1 May 2023, date of current version 10 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3272045

RESEARCH ARTICLE

Dual-Channel EtherCAT Control System for 33-DOF Humanoid Robot TOCABI

JUNEWHEE AHN[1], SUHAN PARK[1], JAEHOON SIM[1], AND JAEHEUNG PARK[2,3], (Member, IEEE)

[1]Department of Intelligence and Information, Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, Republic of Korea
[2]Department of Intelligence and Information, Graduate School of Convergence Science and Technology, ASRI, RICS, Seoul National University, Seoul 08826, Republic of Korea
[3]Advanced Institutes of Convergence Technology (AICT), Suwon 16229, Republic of Korea

Corresponding author: Jaeheung Park (park73@snu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government, Ministry of Science and ICT (MIST), under Grant 2021R1A2C3005914.

ABSTRACT Multi-axis actuation robotics systems comprising many joints with a floating base require more stability and safety than fixed robots and thus require more considerations. In this paper, we describe the implementation of a real-time EtherCAT control system for the TOCABI humanoid robot with 33 degrees of freedom (DOFs) is described. The focus is on the development of a high-performing EtherCAT MainDevice, which enables control of the robot's high DOF at fast communication cycles. We also explore the use of a dual-channel EtherCAT MainDevice as a redundancy mechanism to handle communication disruptions and show that this configuration reduces the burden on the communication network and increases the communication cycle, leading to good real-time performance. To demonstrate the advantages of the system, we examine the performance of the EtherCAT communication and evaluate the impact of RTnet on real-time performance, demonstrating that a high-performing EtherCAT MainDevice having hard real-time capabilities can be established using open-source software. The results of this work demonstrate the potential of using dual channels in EtherCAT MainDevice configurations and utilizing open-source software to implement low-cost EtherCAT MainDevice systems. The paper's contribution to the field is its indication of developing stable robot systems with high DOF, which require hard real-time capability, even with open-source software.

INDEX TERMS Framework, humanoid robot, real-time, robot control system.

I. INTRODUCTION

Astable system is one of the most significant aspects of robots. From safety algorithms such as collision avoidance and emergency stops to complex and precise algorithms for locomotion and manipulation, none can be implemented without a stable system. The importance of a stable system for floating-base robots is higher than for other robots because, with no fixed base frame, they can cause harm if they lose their stability. Floating-base robots, such as humanoids and quadrupeds, must keep actuating their joints to maintain their balance. Thus, a stable system with the robot components, such as motor controllers and sensors, is necessary.

Most control algorithms require precise timing to calculate time derivatives of state variables. The precision of the defined delta-T, which is the time interval between two consecutive control system updates, has a significant impact on the dynamic performance of the robot to control the motion of each joint precisely satisfying the desired motion. When the defined delta-T jitters occur in a discrete time system, it significantly affects the noise level of the time derivative variables. Consequently, the noise can cause system instability, that is, the falling of the floating-base robots. In addition, the communication frequency affects time delay on the control system. The time delay in the motor controllers deteriorates

The associate editor coordinating the review of this manuscript and approving it for publication was Min Wang.

FIGURE 1. Humanoid robot TOCABI.

the control stability [1], [2]. Therefore, the communication system for the components of the humanoid robot must be robust, stable, fast, and strictly periodic.

A stable system with a high control frequency can be achieved through the EtherCAT protocol. EtherCAT is a real-time industrial Ethernet technology originally developed by Beckhoff Automation. The EtherCAT protocol, disclosed in the IEC standard IEC61158, is suitable for hard and soft real-time requirements in automation technology. During the development of EtherCAT, the primary focus was on short cycle times ($\leq$100 μs), low jitter for accurate synchronization ($\leq$1 μs), and low hardware expense.

The precise and stable synchronization requires a real-time system to decrease communication latency, which can cause the control cycle to fail. Two major projects for real-time implementation, Preempt-RT and Xenomai, are available on the Linux platform. Preempt-RT is a simple and powerful real-time priority management patch for the Linux kernel. It provides fast support for the latest Linux kernel and easy patching for general usage. Xenomai [3] is an open-source software project in which engineers from a wide background collaborate to build a robust and resource-efficient real-time core for Linux, following the dual kernel approach, for applications with stringent latency requirements. Prior studies [4], [5] have evaluated the real-time performance of Xenomai and Preempt-RT by measuring average and maximum latencies. As a result, Xenomai exhibited better performance than did Preempt-RT. Reference [6] compared real-time performance with multi-core processors between Xenomai and Preempt-RT. Another study [7] focused on IgH EtherCAT MainDevice with real-time Linux kernel [8]. Real-time Linux kernel is important for real-time performance, but the network interface driver must also be real-time to secure real-time performance at EtherCAT communication. RTnet [9], which is a hard real-time network protocol stack, was applied for EtherCAT communication at [10]. Recently, RTnet was applied to SOEM [11] on a Xenomai Linux system [12], but the application was not robotics. An EtherCAT MainDevice system with robotics application was implemented by [13], [14], [15], and [16]. Nevertheless, no previous study has applied EtherCAT with RTnet that enables real-time network interface card (NIC) to humanoid robots to ensure hard real-time performance and to stabilize the system. Most previous studies also used Linux kernels or software that were incompatible with recent hardware.

This paper describes the components of the communication system to control the humanoid robot and to set up the system based on open-source software. The arrangement of the entire EtherCAT system is described. Further, a description on setting up the EtherCAT MainDevice through RTnet and SOEM for the hard real-time performance of EtherCAT communication is provided, as opposed to a simple real-time EtherCAT system without a real-time NIC driver applied in previous papers. Furthermore, the structure of the control system software of TOCABI is presented. The maximum and average latencies of the established system, as revealed by the experiment, are described and compared under different system settings.

The major contributions of this study are as follows:

- the establishment of an EtherCAT MainDevice using open-source software, achieving high frequency and hard real-time performance of EtherCAT communication cycle with a novel configuration on the real-time network interface and dual-channel EtherCAT MainDevice;
- the demonstration of the importance of using real-time network interfaces by comparing the real-time performance of EtherCAT communication between a generic network interface driver and a real-time network interface driver (RTnet);
- the application of a stable control system to high-degree-of-freedom (DOF) robots with floating bases and the multi-threaded control software architecture; and
- demonstration of the easy implementation of a real-time EtherCAT MainDevice system with multi-axis robots and elucidation of the requirements for a stable system on the basis of successful implementation results as well as detailed system and software configurations.

The paper is organized as follows: Section II presents a comprehensive overview of the hardware and software system of the humanoid robot TOCABI. Section III provides an in-depth exploration of the configurations of the TOCABI's EtherCAT system. Section IV introduces the control system software of TOCABI, and Section V reports the results of real-time and EtherCAT communication

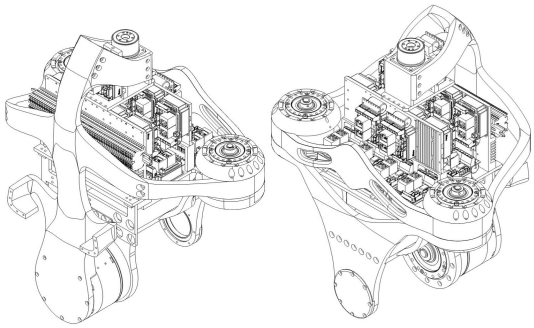

FIGURE 2. Chest Link of TOCABI: figure on the left is back side of chest link and figure on the right is front side of chest link.

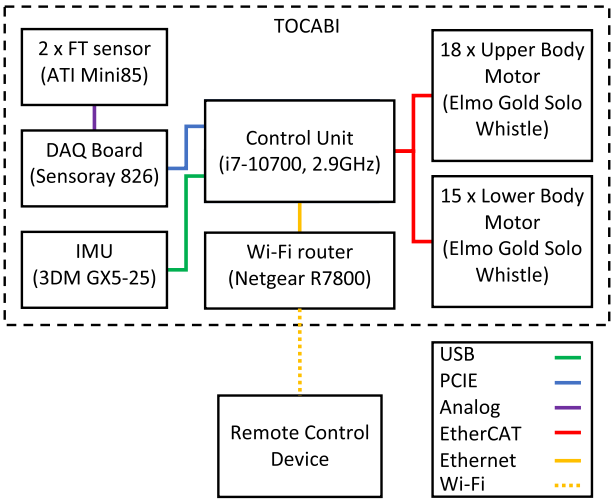

FIGURE 3. The system of TOCABI.

performance evaluations. Finally, Section VI concludes the paper. Additionally, the appendix provides detailed configurations of the real-time Linux kernel and BIOS to aid other researchers in setting up similar systems for future robot development.

II. SYSTEM OVERVIEW

A. HARDWARE OVERVIEW

The TOCABI [17] was designed to reflect the height of an adult male, whose weight and height are 100 Kg and 1.8 m, respectively. TOCABI is shown in Fig. 1. TOCABI has 33 joints, with two DOFs in the neck, eight in each arm, three in the waist, and six in each leg. The joints are powered by Parker and Kormollgen motors with a harmonic gear that has a 100:1 reduction ratio, and are controlled by Elmo Gold Solo Whistle motor controllers using the EtherCAT communication protocol. Due to the limited space in a humanoid robot, the motor drivers are densely packed in the chest of TOCABI, as shown in Fig. 2.

In addition to the motors, the TOCABI also features an inertial measurement unit (IMU), Lord Microstrain sensing 3DM-GX5-25. The IMU is attached to the pelvis of the robot and connected via USB, and it sends data at 500 Hz. Each foot is equipped with an ATI 6-Axis force-torque (FT) sensor, and the sensor data is obtained using a DAQ, Sensory 826. The control PC for the robot includes a mini-ITX Gigabyte H470i AORUS PRO AX motherboard, Intel Core i7-10700 processor, 32 GB (16 GBx2) of RAM, 250 GB of NVMe SAMSUNG SSD, and a quad-port Intel I340 PCI LAN card for the EtherCAT communication protocol. The entire system is illustrated in Fig. 3.

B. SOFTWARE OVERVIEW

Ubuntu 20.04 was selected as the operating system for the real-time EtherCAT MainDevice. For the Linux kernel, the real-time patched kernel Linux 5.4.134 with Xenomai 3.2.1 was used. After installing the Xenomai-patched kernel, the Xenomai library was required to be installed. As only the POSIX layer of Xenomai was used, POSIX-dependent options were selected, which is the Cobalt Core of Xenomai. The according options are described in Appendix A. RTnet was designed to lend real-time networking functionality to the Xenomai environment. As it worked with a limited set of Ethernet cards, an Intel I340 4 port PCI LAN card was used for EtherCAT communication. As Ubuntu 18.04 has a compatibility issue with RTnet that prevents the EtherCAT communication through RTnet, it is not recommended for use with RTnet. The EtherCAT MainDevice on the control computer was implemented with the RTnet SOEM library, which was an RTnet-patched SOEM. Detailed information about RTnet SOEM is described in Appendix B. The robot operating system (ROS) was used to implement data viewing and task commands via an external remote computer. The graphical user interface (GUI) was built on RQT for the robot. The GUI supports commanding, checking, and visualization of the robot. Other task commands and external command data could be received with ROS topic. There is also a web application to control TOCABI. This web application supports many devices that can access the web, such as a tablet, smartphone, PC, and laptop.

III. ETHERCAT CONFIGURATION

This section describes the configurations for the stable EtherCAT system of the humanoid robot, TOCABI.

A. EtherCAT DRIVER INSTALLATIONS AND WIRING

Motor drivers were distributed on several links for the lower body. However, most motor drives for the upper body were placed on the chest link to reduce the weight of each link. The chest link of TOCABI and deployment of the EtherCAT SubDevice can be observed in Fig. 2 These motor driver configurations helped to alleviate the weight of each link. However, the cramped motor drivers on the upper body limited the installation of LAN cables for EtherCAT communication. Although the EtherCAT technology group provided several installation guidelines to prevent electromagnetic disturbance from affecting communication performance, this system could not retain a few of them owing to the dense configuration. Most of the communication issues, including intermittent packet loss problems experienced during

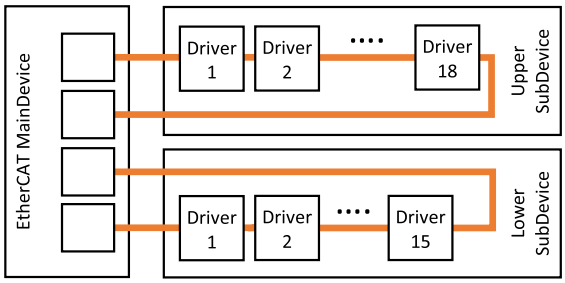

FIGURE 4. The EtherCAT network setup of TOCABI. The Upper SubDevice represent Upper Body motor drivers of TOCABI, and The Lower SubDevice represent lower body motor drivers of TOCABI.

development over several years, were solved by using an adequately shielded CAT7 LAN cable. However, these intermittent communication issues could not be solved just by using a good LAN cable because quality challenges also exist in good LAN cables. Therefore, it is important to identify where the problem occurs, when there is a communication problem between drivers. Hardware issues, such as a lost link or corrupted packet, could be detected by checking the hardware error counter of each driver.

B. DUAL-CHANNEL EtherCAT WITH REDUNDANCY

The EtherCAT technology enables the realization of cable redundancy. For this purpose, an additional cable connection was established from the last EtherCAT SubDevice to the MainDevice. In the event of cable breakage, telegrams were sent seamlessly via the alternative connection.

EtherCAT redundancy mode appeared to have good fail-safe functions. However, if two communication failures occurred simultaneously, and a particular group of SubDevice was isolated from the outside, the group would experience communication failures, despite the redundancy mode. Therefore, to acquire stability and fail-safe ability, the total topology of the EtherCAT SubDevice was divided into two parts: the upper body (head and both arms, 18 DOFs) and the lower body (waist and both legs, 15 DOFs). Upper and lower EtherCAT systems were connected to the control computer, which was the EtherCAT MainDevice, with a redundancy mode for fail-safe. The diagram of the EtherCAT connection is described in Fig. 4.

To prevent malfunctions in the software under development, each EtherCAT channel was operated as an independent process. The communication cycle of both processes must be synchronized for better control performance. To synchronize the EtherCAT communication command of both processes, communication commands of channel 2 are triggered by signals in channel 1.

Each process communicates with the EtherCAT SubDevice in the corresponding channel through this synchronization method, reading torque commands from shared memory in every loop and writing joint states data to shared memory.

IV. CONTROL SYSTEM SOFTWARE

The entire control software is divided into three major parts, each executed as a different process for independent execution, so that any errors that may occur do not cause problems in the overall system. The following subsection describes each part of the overall software system of TOCABI.

TABLE 1. TPDO and RPDO configuration of EtherCAT communication.

PDO	Index	Object	Bit Len
TPDO	0x1605	Target Position	32 bit
		Target Velocity	32 bit
		Max Torque	16 bit
		Control Word	16 bit
		Modes of Operation	16 bit
RPDO	0x1a00	Position Actual Value	32bit
		Digital Inputs	32 bit
		Status Word	16 bit
	0x1a11	Velocity Actual Value	32 bit
	0x1a13	Torque Actual Value	16 bit
	0x1a1e	Auxiliary position value	32 bit

A. EtherCAT MainDevice

To configure the EtherCAT MainDevice with real-time NIC driver, the RTnet SOEM library was used, which is the RTnet-patched SOEM for the real-time network. The EtherCAT MainDevice is divided into two processes to cover the two channels, each responsible for 18 joints in the upper body and 15 joints in the lower body. The EtherCAT MainDevice communicates with each SubDevice at a 4 kHz cycle. Process Data Objects (PDO) is a communication protocol in EtherCAT. PDO is used to transfer process data between nodes in real-time. In this system, PDO is used to communicate between SubDevice and the MainDevice. The data objects and size that are communicated through PDO communication between SubDevice and the MainDevice in TOCABI are shown in Table 1. TPDO and RPDO mean transmit PDO and receive PDO, respectively. The total size of TPDO is 112 bits, and the total size of RPDO is 160 bits. Each process consists of two threads: a communication thread and a status check thread.

Each communication thread is a real-time thread, and each thread was fixed to a different CPU core to avoid real-time performance interference with each other. In the communication thread, a code for sending and receiving the Ethernet frame that contains the EtherCAT data is used for EtherCAT communication. The EtherCAT MainDevice waits for the target start time for each communication cycle and sends an EtherCAT frame to SubDevice. SubDevice receive a frame sent from the MainDevice and then send a frame containing information from the SubDevice to the MainDevice, which waits for the frame to return and be received. Additionally, a few simple safety algorithms are included in communication thread, such as joint and speed limits for each joint.

The status check thread continuously checks the status of each driver to find which part of the communication has an issue. This status checking is done by continuously receiving the hardware error counter of each driver, such as CRC error, RX error, and Link Lost Event, through the FPRD command of EtherCAT, which is used to read data from a specific address of EtherCAT SubDevice devices.

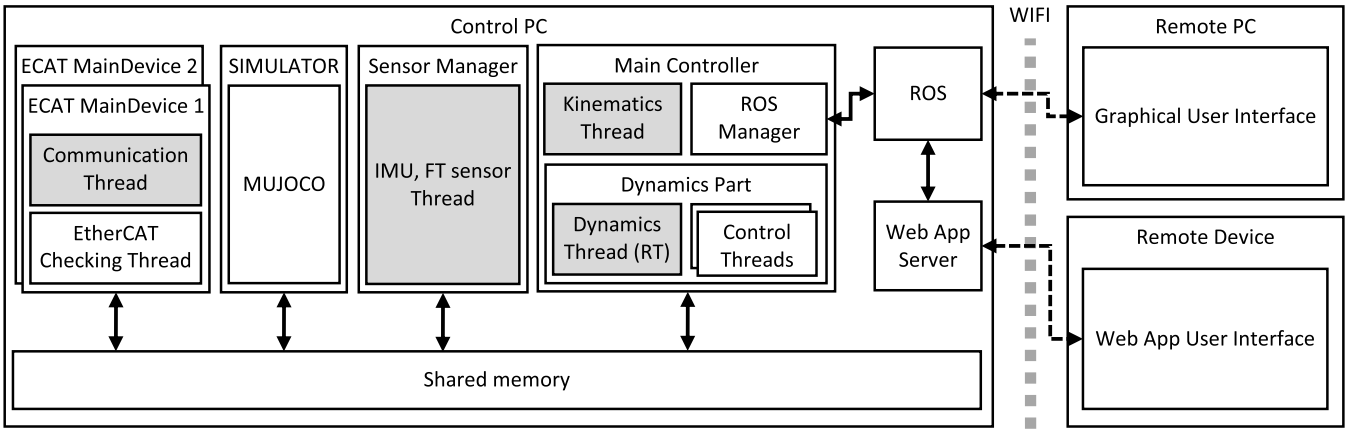

FIGURE 5. The Software architecture of TOCABI.

B. SENSOR MANAGER

The sensor manager manages the IMU and FT sensors. It receives the IMU sensor data from the USB and FT sensor data from the sensoray826 DAQ board with a 500 Hz frequency. The sensor manager also manages the status of sensors and sensor data filtering. The obtained sensor data are stored in the shared memory.

C. MAIN CONTROLLER

The main controller is divided into two major parts of the kinematics part and dynamics part. It also contains additional parts for status checking and logging and other command communication. The kinematics part is a single thread, which is the real-time thread that runs with a 2 kHz cycle. It updates kinematics and mass matrix information with the rigid body dynamics library (RBDL) [18], after reading updated joints and sensor data from shared memory. The kinematics update thread includes the state estimation calculations, robot state publishers, and task command subscribers. Additionally, this kinematic thread is responsible for communicating with the sensor manager and the EtherCAT MainDevice through the shared memory.

The dynamics part comprises several threads. Among them, the main dynamics thread is a real-time thread that is in charge of dynamics parts that can be calculated quickly, such as whole-body control and joint control. It stores torque commands to match the 4 kHz cycle. Additionally, the other dynamics threads are responsible for time-consuming operations such as planning, optimizing, and dynamics calculations. Furthermore, a checking and logging part checks all system states for safety and records the status. This part compresses and saves all the states of the system. These relationship plots can be observed in Fig. 5. Gray boxes represent real-time threads.

D. SIMULATION

Performing simulations to test and verify the TOCABI system before actual experiments is essential. The use of MuJoCo [19] as a simulator is crucial in this regard. MuJoCo is selected owing to its advanced contact engine capabilities which are essential for tasks such as grasping, manipulation and locomotion. MuJoCo's solvers are optimized for simulating articulated systems with many DOFs, making it well-suited for simulating humanoid robots. Its real-time performance enables the simulation of complex dynamics such as walking. Studies have also shown that MuJoCo is one of the best physics engines for simulating humanoid robots. Comparing it with other physics engines, MuJoCo has been found to have the best accuracy while achieving the fastest simulation speeds. This makes it an ideal choice for simulating the TOCABI system. The use of MuJoCo enables TOCABI to thoroughly test and verify the system before deploying it on the physical robot.

It is essential to conduct simulations to test and verify the system before the actual experiment. MuJoCo was used for

FIGURE 6. Web application-based control interface of TOCABI on a tablet (left) and a smartphone (right). The interface allows users to command and control the robot's movements and status from any device.

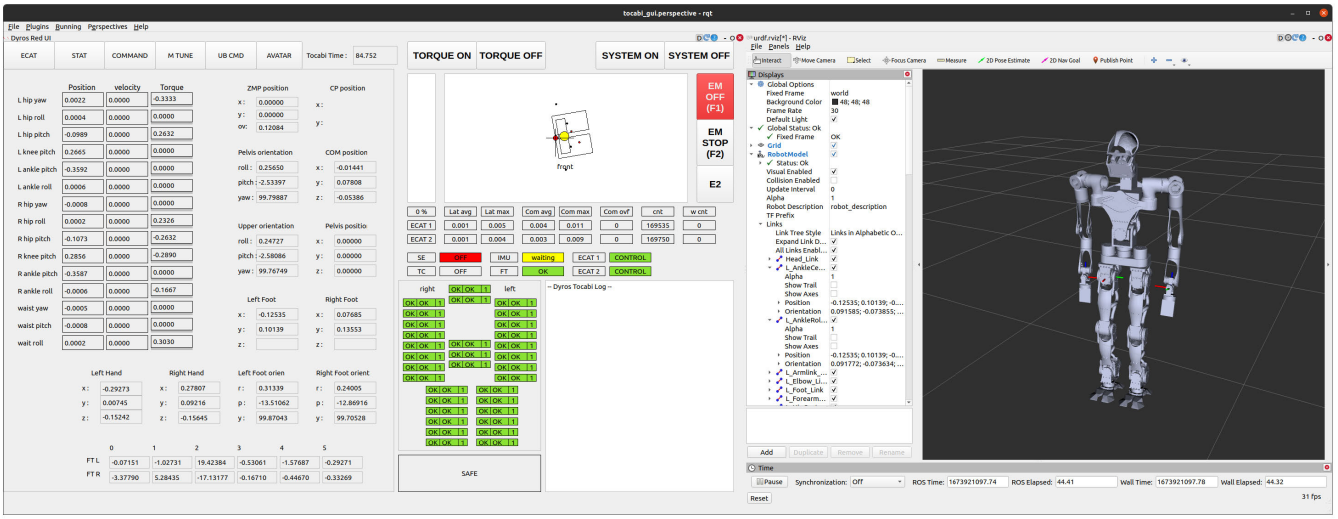

FIGURE 7. TOCABI commanding GUI program.

TABLE 2. Upper body Execution time comparison at 4 kHz communication(μs).

		AVG	MIN	MAX	STD	OVF
Latency	RTNET	0.76	-0.32	10.64	0.34	
	GENERIC	0.76	-0.50	136.12	0.43	
Send	RTNET	2.94	1.88	9.45	0.47	
	GENERIC	16.46	9.63	70.10	3.13	
Recv	RTNET	87.57	74.71	105.57	2.69	0
	GENERIC	77.74	37.07	305.33	2.94	717

TABLE 3. Lower body Execution time comparison at 4 kHz communication(μs).

		AVG	MIN	MAX	STD	OVF
Latency	RTNET	2.40	-0.23	13.30	0.70	
	GENERIC	1.13	-0.17	33.27	0.61	
Send	RTNET	5.40	2.16	16.57	0.98	
	GENERIC	15.26	10.07	155.93	1.71	
Recv	RTNET	74.76	63.66	91.10	2.43	0
	GENERIC	69.02	27.51	251.26	2.91	1

the simulation of TOCABI since MuJoCo supports a good contact engine. A prior study [20] conducted comparisons of the simulation engine. Among several physics engines, the MuJoCo engine showed the best result for the humanoid robot simulation. It showed the best accuracy while achieving the fastest speed.

E. CONTROL USER INTERFACE

There are two types of interface programs available for commanding and controlling the humanoid robot TOCABI. One is a web application-based user interface that can be accessed from any device, including laptops, desktop computers, smartphones, and tablets. This interface is hosted on the main computer of TOCABI and is built using the Tornado [21] and ROSbridge [22] frameworks. The other interface program is a GUI program that can be launched from Ubuntu and displays the current status of TOCABI in 3D. Users can also check the status of each joint and the EtherCAT MainDevice from this interface. Both interfaces provide users with the ability to command and control the robot's movements and status. Fig. 6 shows the web application interface, and Fig. 7 shows the GUI program from Ubuntu.

V. REAL-TIME PERFORMANCE MEASUREMENTS AND VALIDATION

The real-time performance of an EtherCAT MainDevice system is critical as it must complete its tasks within a specified time frame. To evaluate the performance, the execution time of the three key components of the system: latency, receive, and send were measured. Latency represents the real-time performance by comparing the desired execution time to the actual execution time for each cycle. Therefore, latency is calculated as $latency = t_{execution} - t_{desired}$, where $t_{execution}$ is the execution time point of communication and $t_{desired}$ is the desired execution time point of communication. The receive component involves the EtherCAT MainDevice receiving the EtherCAT frame from the SubDevice, while the send component involves the EtherCAT MainDevice sending the frame to the SubDevice. Unlike latency, receive and send represent the execution time of the function in the program.

TABLE 4. Upper body Execution time comparison at 8 kHz communication(μs).

		AVG	MIN	MAX	STD	OVF
Latency	RTNET	1.46	-0.30	15.48	0.66	
	GENERIC	11.96	-0.49	495.57	42.15	
Send	RTNET	4.68	2.13	18.05	1.15	
	GENERIC	13.26	8.67	113.19	2.15	
Recv	RTNET	88.24	74.52	107.30	2.88	0
	GENERIC	88.16	6.14	516.77	34.72	728E+4

The experiments were conducted with two communication cycles, 4 kHz and 8 kHz, using RTNET and a generic driver. Each experiment took 3 hours. The experiment was

TABLE 5. Lower body Execution time comparison at 8 kHz communication(μs).

		AVG	MIN	MAX	STD	OVF
Latency	RTNET	1.44	-0.30	13.54	0.66	
	GENERIC	0.74	-0.50	172.83	0.62	
Send	RTNET	4.81	2.13	22.47	0.66	
	GENERIC	12.47	9.08	197.65	1.59	
Recv	RTNET	75.69	61.19	125.28	2.75	0
	GENERIC	69.70	24.93	278.65	1.77	9102

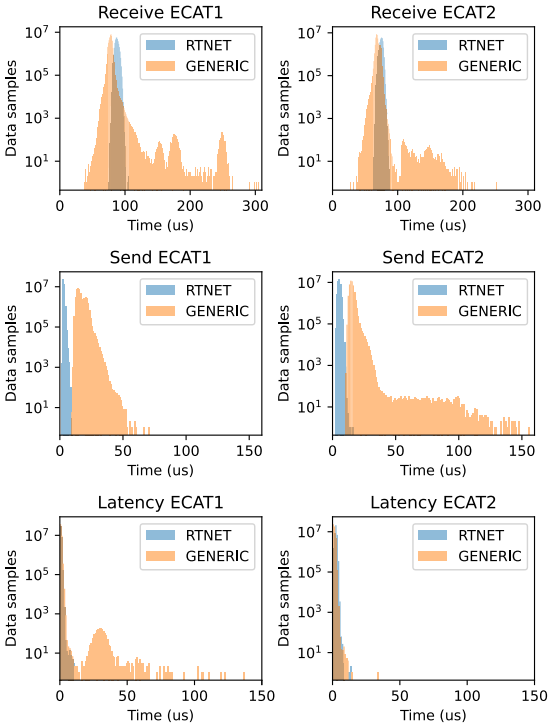

FIGURE 8. Execution time comparison histogram of 4 kHz experiment.

conducted to evaluate the impact of RTnet on the real-time stability of the TOCABI's whole-body control algorithm [23], which was performed simultaneously with kinematics and dynamics calculations. The results of the 4 kHz experiment, including the average, minimum, maximum, and standard deviation of latency, and receive and send times are shown in Table 2 and Table 3 and represented in histograms in Fig. 8. The results of the 8 kHz experiment are shown in Table 4 and Table 5 and represented in histograms in Fig. 9. The experiments were carried out on Xenomai real-time Linux, with RTNET and GENERIC indicating whether RTNET was applied to the NIC. ECAT1 and ECAT2 refer to the parts responsible for the upper and lower body of TOCABI in a dual-channel EtherCAT system.

The results showed that when using the generic driver, overflows occurred in both 4 kHz and 8 kHz cases, exceeding the specified timeout. However, when using RTNET, both 4 kHz and 8 kHz cases showed excellent results without any timeout. The results also indicated that when RTNET was used, the data were concentrated on the mean, and the

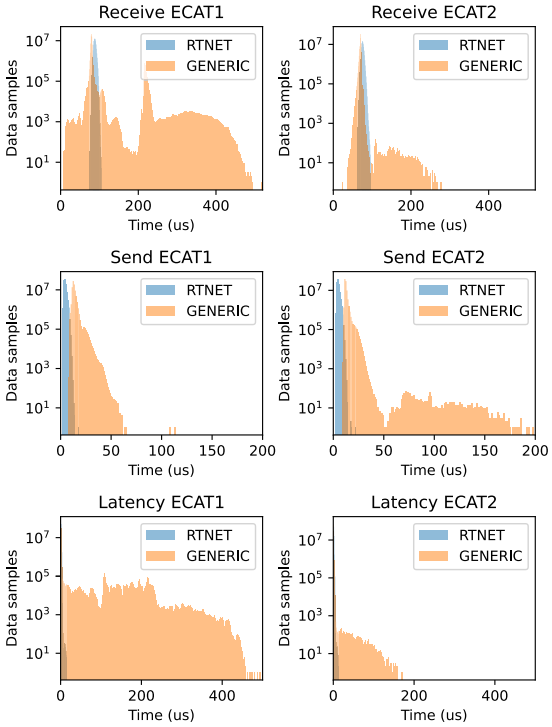

FIGURE 9. Execution time comparison histogram of 8 kHz experiment.

standard deviation was smaller, with the maximum and minimum values not deviating significantly from the average. This indicated that the real-time performance was improved by the use of RTNET. This improvement was also evident in the histograms, which confirm the uniformity of communication when using RTNET.

VI. CONCLUSION

In this study, we constructed a real-time dual-channel EtherCAT control system framework that was applied to the humanoid robot TOCABI. We also developed a user interface for control and simulation environments for TOCABI. Further, we demonstrated the configuration of the dual-channel EtherCAT MainDevice that enables the control of robots with high degrees of freedom at faster communication cycles. The real-time performance of EtherCAT communication was analyzed and confirmed to be successful.

The experiment aimed to evaluate the impact of RTnet on real-time performance in the context of high-speed communication. It was observed that by configuring the system through RTnet, a high-performing EtherCAT MainDevice having hard real-time capabilities can be established. Additionally, the topology of the EtherCAT system was divided into two dual-channel EtherCAT MainDevice to examine its performance. The results showed that the dual-channel configuration was effective in reducing the burden on the communication network, increasing the communication cycle, and demonstrating good real-time performance on both channels. Furthermore, the dual-channel EtherCAT MainDevice was configured as a redundancy mechanism to handle communication

disruptions. This paper highlights the potential of using multiple channels in EtherCAT MainDevice configurations, rather than relying on a single channel, by utilizing open-source software, which enables researchers to implement low-cost EtherCAT MainDevice systems.

The stability of the proposed system is also experimentally confirmed through the ANA AVATAR XPRIZE competition, a teleoperation system competition, showing stable operation of the robot TOCABI. Selecting the right compatible elements among rapidly changing hardware and software for the stable and proper movement of multi-axis robots is time-consuming. We anticipate that this study can serve as an indicator for developing stable robot systems with high DOF, which require hard real-time capability even with open-source software.

APPENDIX A
DETAILED CONFIGURATION FOR REAL-TIME COMPUTING

A. LINUX KERNEL CONFIGURATION

Table 6 shows the important kernel options for a Xenomai-patched Linux 5.4.134 kernel.

TABLE 6. Selected kernel configuration.

General setup	
Timer subsystem > High-Resolution Timer Support	Enable
Processor type and features	
Linux guest support	Disable
Multi-core scheduler support > CPU core priorities scheduler support	Disable
Xenomai/cobalt	
Xenomai/cobalt > Sizes and static limits	
Number of registry limits	4096
Size of system heap (Kb)	4096
Size of private heap (Kb)	256
Size of shared heap (Kb)	256
Maximum number of POSIX timers per process	512
Xenomai/cobalt > Drivers > RTnet	
RTnet, TCP/IP socket interface	Enable
Add-Ons > Real-Time Capturing Support	Enable
Power management and ACPI options	
CPU Frequency scaling	Disable
ACPI Support > Processor	Disable
CPU Idle : CPU idle PM support	Disable
Memory Management Options	
Transparent Hugepage Support	Disable
Allow for memory compaction	Disable
Contiguous Memory Allocator	Disable
Allow for memory compaction > Page Migration	Disable
Device Drivers	
Staging drivers > Unisy3s SPAR driver support	Disable
Kernel hacking	
Debug preemptible kernel	Disable
KGBD: kernel debugger	Disable

B. LINUX GRUB BOOTING OPTIONS

The following commands show grub options, which enable kernel options.

```
GRUB_CMDLINE_LINUX_DEFAULT="\ldots
    xonemai.smi=enabled noapic
    isolcpus=4-7"
```

System Management Interrupts (SMIs) are special interrupts at the highest priority, causing the x86 CPU to enter the System Management Mode, a variant of the flat real mode for executing some handler, which was implemented by the BIOS. SMIs do not go through the interrupt controller. They are detected by the CPU logic in between instructions and unconditionally dispatched from thereon. This introduces critical issues for real-time systems. xenomai.smi options were enabled to avoid interruption by SMI. Isolcpus prevents the scheduler from assigning other processes to specific CPU cores, so that real-time processes are not interrupted by logical real-time processes.

C. BIOS OPTION FOR REAL-TIME PERFORMANCE

There are certain features in the x86-based modern computer that can reduce power consumption and increase performance. However, these features can introduce latency to real-time performance. Therefore, disabling these features from BIOS helps improve the real-time performance. The followings are the features that we disabled for real-time performance:

- Enhanced Intel Speed Step Technology
- Intel Speed Shift Technology
- Intel Hyper-Threading Technology
- C-states

APPENDIX B
RTnet_SOEM

This paper introduces RTnet-patched SOEM with open-source codes while the original version of SOEM does not support RTnet. The source codes can be found on GitHub https :// github .com/saga0619/rtnet_soem.

REFERENCES

[1] G. Stépán and T. Insperger, "Stability of time-periodic and delayed systems—A route to act-and-wait control," *Annu. Rev. Control*, vol. 30, no. 2, pp. 159–168, Jan. 2006.

[2] J. Jung, S. Hwang, Y. Lee, J. Sim, and J. Park, "Analysis of position tracking in torque control of humanoid robots considering joint elasticity and time delay," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot. (Humanoids)*, Nov. 2017, pp. 515–521.

[3] *Xenomai*. Accessed: Mar. 13, 2023. [Online]. Available: http://www.xenomai.org

[4] C. Huang, C.-H. Lin, and C.-K. Wu, "Performance evaluation of Xenomai 3," in *Proc. 17th Real-Time Linux Workshop (RTLWS)*, 2015, pp. 21–22.

[5] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio, "Performance comparison of VxWorks, Linux, RTAI, and Xenomai in a hard real-time application," in *Proc. 15th IEEE-NPSS Real-Time Conf.*, Apr. 2007, pp. 1–5.

[6] R. Delgado and B. W. Choi, "New insights into the real-time performance of a multicore processor," *IEEE Access*, vol. 8, pp. 186199–186211, 2020.

[7] M. Cereia, I. Bertolotti, and S. Scanzio, "Performance of a real-time EtherCAT master under Linux," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 679–687, Nov. 2011.

[8] *IgH EtherCAT Master for Linux*. Accessed: Mar. 13, 2023. [Online]. Available: https://etherlab.org/en/ethercat

[9] J. Kiszka and B. Wagner, "RTnet—A flexible hard real-time networking framework," in *Proc. IEEE Conf. Emerg. Technol. Factory Automat.*, vol. 1, Sep. 2005, p. 456.

[10] Y.-S. Moon, N.-Y. Ko, K.-S. Lee, Y.-C. Bae, and J.-K. Park, "Real-time EtherCAT master implementation on Xenomai for a robot system," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 9, no. 3, pp. 244–248, Sep. 2009.

[11] *Simple Open EtherCAT Master (SOEM)*. Accessed: Mar. 13, 2023. [Online]. Available: https://openethercatsociety.github.io/
[12] S. J. Kang, "A study on implementation of real-time EtherCAT master," *J. Semicond. Display Technol.*, vol. 20, no. 2, pp. 131–136, 2021.
[13] G. Zhang, Z. Li, F. Ni, and H. Liu, "A real-time robot control framework using ROS control for 7-DoF light-weight robot," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2019, pp. 1574–1579.
[14] T. Asfour, L. Kaul, M. Wachter, S. Ottenhaus, P. Weiner, S. Rader, R. Grimm, Y. Zhou, M. Grotz, F. Paus, D. Shingarey, and H. Haubert, "ARMAR-6: A collaborative humanoid robot for industrial environments," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2018, pp. 447–454.
[15] F. Sygulla, R. Wittmann, P. Seiwald, T. Berninger, A.-C. Hildebrandt, D. Wahrmann, and D. Rixen, "An EtherCAT-based real-time control system architecture for humanoid robots," in *Proc. IEEE 14th Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2018, pp. 483–490.
[16] M. Ferrati, A. Settimi, L. Muratore, A. Cardellino, A. Rocchi, E. Mingo Hoffman, C. Pavan, D. Kanoulas, N. G. Tsagarakis, L. Natale, and L. Pallottino, "The walk-man robot software architecture," *Frontiers Robot. AI*, vol. 3, p. 25, May 2016.
[17] M. Schwartz, J. Sim, J. Ahn, S. Hwang, Y. Lee, and J. Park, "Design of the humanoid robot TOCABI," in *Proc. IEEE-RAS 21st Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2022, pp. 322–329.
[18] M. L. Felis, "RBDL: An efficient rigid-body dynamics library using recursive algorithms," *Auton. Robots*, vol. 41, no. 2, pp. 495–511, Feb. 2017.
[19] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[20] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, MuJoCo, ODE and PhysX," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 4397–4404.
[21] M. Dory, A. Parrish, and B. Berg, *Introduction to Tornado: Modern Web Applications With Python*. Sebastopol, CA, USA: O'Reilly Media, 2012.
[22] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: ROS for non-ROS users," in *Robotics Research*. Cham, Switzerland: Springer, 2017, pp. 493–504.
[23] Y. Lee, J. Ahn, J. Lee, and J. Park, "Computationally efficient HQP-based whole-body control exploiting the operational-space formulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 5197–5202.

JUNEWHEE AHN received the B.S. degree in mechanical engineering from Sungkyunkwan University, Seoul, South Korea, in 2016. He is currently pursuing the Ph.D. degree with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include the whole body control of a humanoid robot, optimal control, and control system framework.

SUHAN PARK received the B.S. degree in robotics engineering from Kwangwoon University, Seoul, South Korea, in 2017. He is currently pursuing the Ph.D. degree with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include grasp and motion planning, trajectory optimization, and planning-control system framework.

JAEHOON SIM received the B.S. degree in mechanical engineering from Korea University, Seoul, South Korea, in 2015. He is currently pursuing the Ph.D. degree with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include robot design, actuators, and mechatronics.

JAEHEUNG PARK (Member, IEEE) received the B.S. and M.S. degrees in aerospace engineering from Seoul National University, South Korea, in 1995 and 1999, respectively, and the Ph.D. degree in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2006. From 2006 to 2009, he was a Postdoctoral Researcher and later a Research Associate with the Stanford Artificial Intelligence Laboratory. From 2007 to 2008, he worked part-time with Hansen Medical Inc., a medical robotics company, USA. Since 2009, he has been a Professor with the Department of Intelligent Convergence Systems, Seoul National University. His research interests include robot-environment interaction, contact force control, robust haptic teleoperation, multicontact control, whole-body dynamic control, biomechanics, and medical robotics.