Team Desciption paper: BabyTigers - R 2025

Ryuki Takemura, Sugimoto Harusuke, Sotaro Takeuchi, Yuka Komori, Kosuke Nakajima, Shohei Yasuda, and Wataru Uemura[0000−0001−6012−0029]

Ryukoku University, 1-5, Seta Ohe, Otsu City, Shiga, Japan
robocup@vega.elec.ryukoku.ac.jp, wataru@rins.ryukoku.ac.jp
https://vega.elec.ryukoku.ac.jp/trac/wiki/BabyTigers-R

Abstract. This paper presents Baby-Tigers-R, a team comprising members from Electronics, information and communication engineering course, Faculty of advanced science and technology at Ryukoku University. For 2025, we change the robot from Robotino to Kachaka as mobile robots. So, this team description paper describes the new hardware and the software of our team.

Keywords: Logistics League, RoboCup, BabyTigers - R, Kachaka, autonomous mobile robot.

1 Introcution

This paper introduces Baby-Tigers-R[1], which is a RoboCup team consisted of members belonging to Electronics, Information and communication engineering course, Faculty of advanced science and technology, Ryukoku University. At 2011, the history of BabyTigers-R started Logistics League (Sponsored by Festo: LLSF) at RoboCup 2011 Istanbul. We got the 3rd place of Logistics League at RoboCup 2015 Hefei. At RoboCup JapanOpen 2019, we had the demonstration for Logistics League. And from 2020, we organized Logistics League at RoboCup JapanOpen. We got 1st place from 2020 to 2023, and 2nd place at 2024 of RoboCup JapanOpen.

At recent years, the robot specification of the rule at Logistics League is changed. The rule allows not only Robotino but also another robot which specification is similar as Robotino. So, this year, we use Kachaka instead of Robotino. Kachaka is probided by Preferred Robotics[2] which is a Japanese company. And it is lower cost and lighter weight than Robotino. However, Kachaka cannot run on over the small obstacles, for example, 100-200V power cables on the venue, because the wheels are smaller than Robotino.

Section 2 describes a hardware construction of our robot. In particullary, we show the specification of Kachaka. Section 3 describes a software of our robot, for example, ROS2, communication, routing, planning, and so on. Finally, we conclude this paper at Section 4.

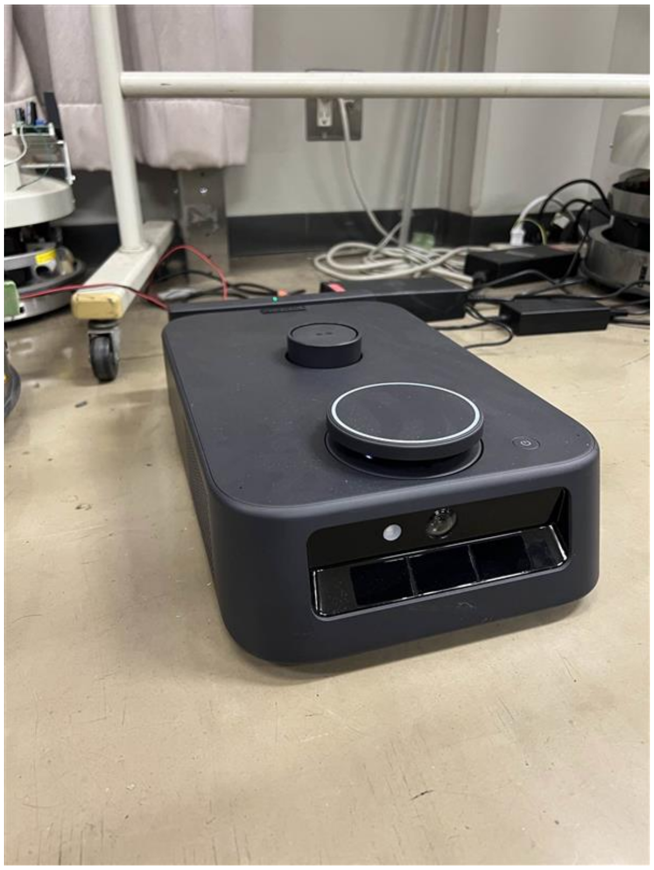

Fig. 1. Front side of Kachaka

2 Hardware

We use Kachaka as a mobile robot. Fig. 1 shows the front side of Kachaka.

Kachaka can dock and undock the shelf. We mount some devices on the shelf including the computer to control Kachaka (shown as Fig. 4). So, the communication between Kachaka and the computer is wireless. At the competition venue of RoboCup, Wireless communication is not allowed, in particularly 2.4GHz and 5GHz. Preferred Robotics can modify Kachaka as the RoboCup Option that means Kachaka has the wired network interface connector. However, this year, our Kachaka does not has the wired connection because there is not enough time to modify.

And for the computer, we need the power source. This year, we use a mobile battery, Anker A1637 (which capacity is 10000mAh), which can provide type-C connection with about 12 Voltage (22.5W: 10V-2.25A or 30W: 15V-2A). The

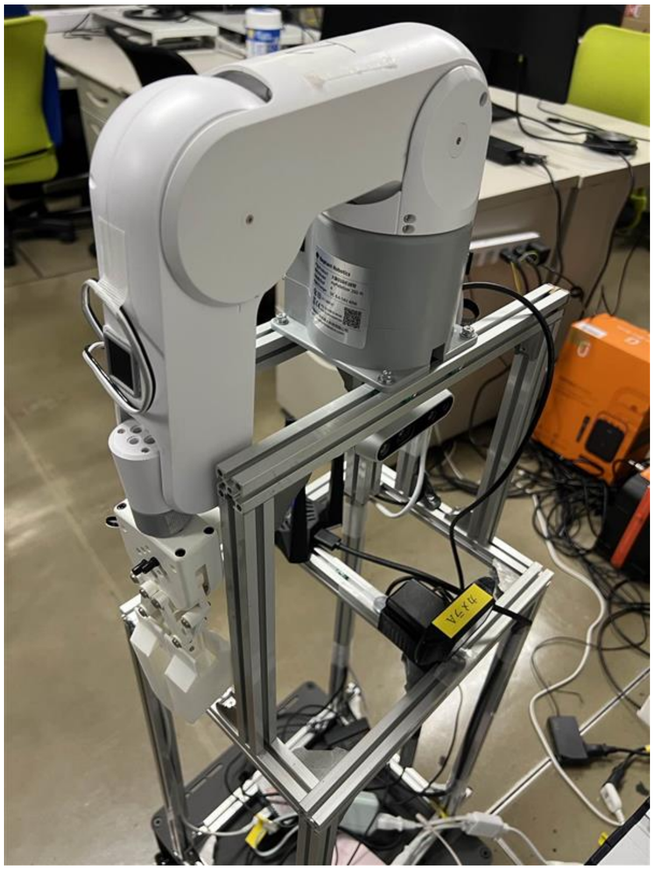

Fig. 2. myPalletizer for getting and putting the work as the arm-typed robot

computer on the shelf is N97-G5 mini PC, which CPU is Intel Alder Lake-N97. We install Ubuntu 22.04 to the computer and use ROS2 humble.

To get and put the work, we use myPalletizer made by Elephant Robotics in Chinese company (shown as Fig. 2). This arm-typed robot has four free degrees and controlled by Raspberry pi 4. Last year, we set this arm by rotating against the vertical. However, this year, we set this arm normal direction that means this arm can rotate for not vertical but horizontal. In order to avoid the dependence of ROS version, our program to control myPalletizer uses SSH connection. So, the service of ROS waits for the control command, and after receiving it, our program run the script on myPalletizer using ssh connection. In this system, using myPalletizer does not depend on the ROS version.

To detect the work, we use the stereo camera made by Intel, Realsense D435 (shown as Fig. 3). And to detect the conveyor belt, we use the pattern matching method at the face of conveyor belt.

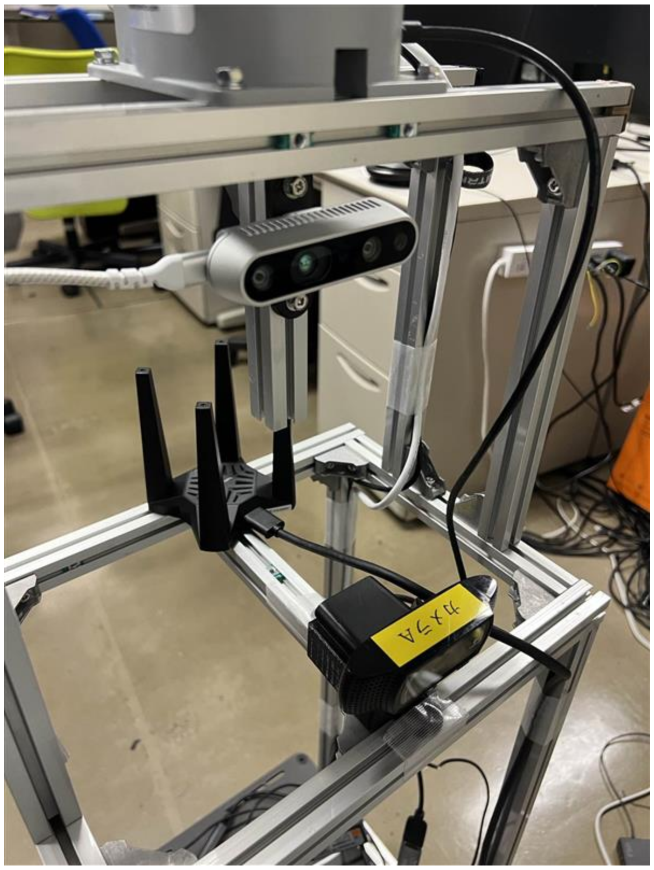

Fig. 3. USB-camera

And to detect the MPS, we use the USB-camera (shown as Fig. 3) and LiDar (shown as Fig. 4). The USB-camera is made by Logitech, C920 which recognizes the marker on MPS. And the LiDar is made by Slamtech, A2M8 or A2M12, which can perform 2D 360 degree scan within a 12-meter range.

2.1 the specification of Kachaka

This subsection describes the specification of Kachaka. Kachaka has two driving wheels and two assistance wheels (shown as Fig. 5). When the left wheel turns the negative motion of the right wheel, Kachaka can turn around at the same place. And the center of turning of Kachaka is the mounting point for the shelf. Then, the shelf can turn around at the same place too.

The maximum speed of Kachaka is about up to 0.8 m/s. And the size of Kachaka is 24.0cm $\times$ 38.7cm. The maximum payload is 30kg or 20kg including

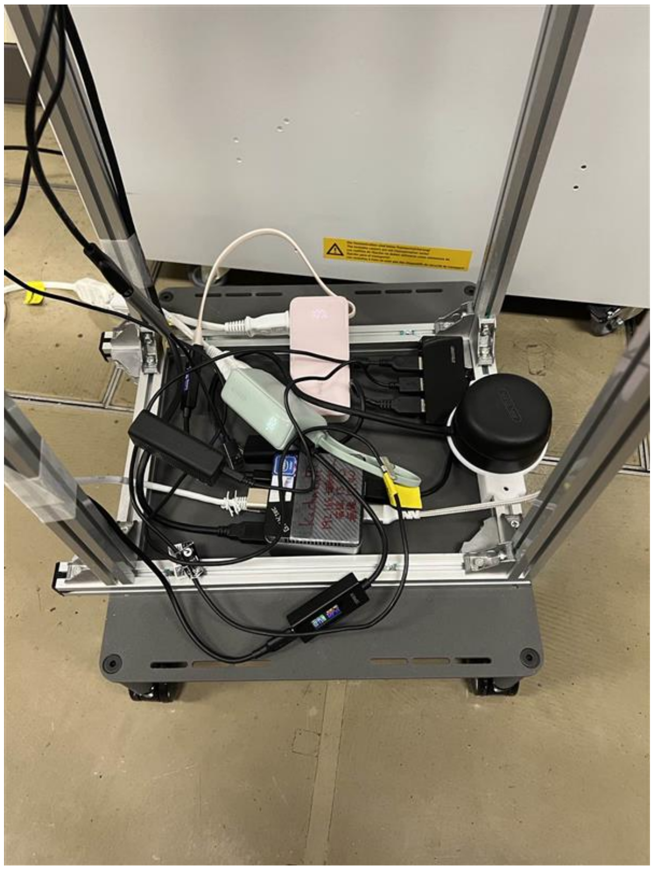

Fig. 4. LiDAR

the shelf by Kachaka Pro and normal Kachaka, respectively. Table 1 shows these specifications.

And the API is provided and we can connect to Kachaka via ROS2 using API. Table 2 shows the API list.

For GermanOpen 2025, we use ROS2 and control programs were made by us. However, for RoboCup 2025 Salvador in Brazil, we may use just API for Kachaka. That is depended on the progress.

3 Software

This section describes the software on our robot. By the previous section, we do not decide about using kachaka's API. One is to use API and python, and the other is to use ROS2 and python. Whatever we select, we need to communicate with RefBox via ROS2. So, we develop the communication program between ROS2 and RefBox.

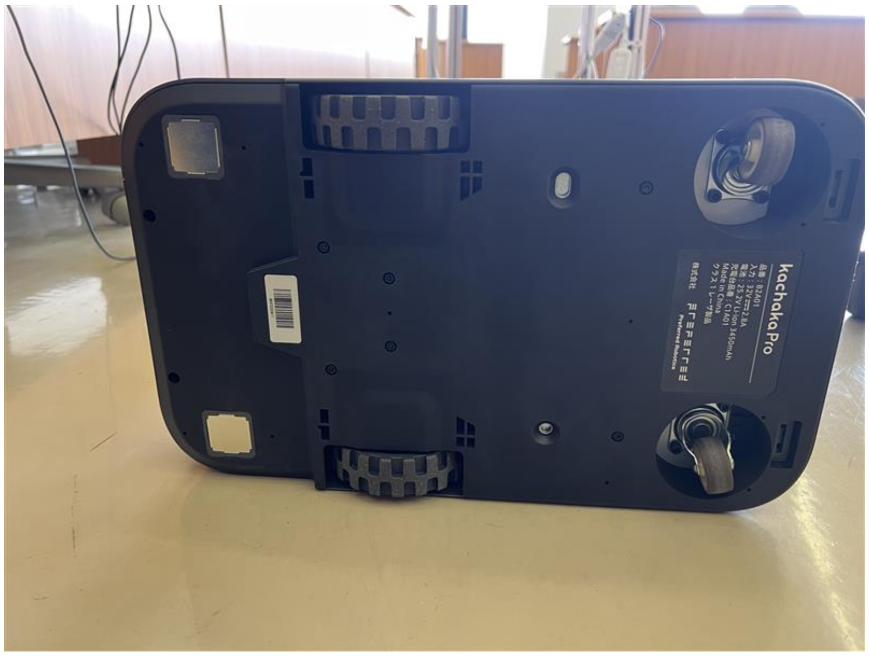

Fig. 5. Bottom surface of Kachaka

3.1 ROS2: RCLL RefBox Peer

Before 2017, Dr. Tim Niemueller provided the communication program between ROS and RefBox at git[3]. At 2021, Dr. Wataru Uemrua updated and maintained this program to the current RefBox Protocol. This year, we adapt it for ROS2 and publish it at the git repository[4]. And for the type of message, we use refbox_msgs including the same repository.

3.2 Aruco Marker

We can detect the Aruco marker using the library of OpenCV[5]. Using this library, we can get not only ID but also the direction and distance between the marker and the camera. And the marker of DICT_ARUCO_ORIGINAL is consists of only four patterns in order to keep the hamming distance between codes. So we use the error correcting ability not only for whole patterns but

Table 1. The specification of Kachaka

Specification	
The width, depth and height	240mm × 387mm × 124mm
Weight	10.0 kg
Battery	Rechargeable lithium ion battery
Rated voltage	DC 25.2 V
Maximum speed	800 mm/s (500 mm/s when shelf mounted)
Communication	Wi-Fi
Camera	RGB camera ×2
Sensor	LiDAR ×1, 3D sensor of ToF ×1, step sensor ×2 and furniture cognitive sensor ×1
Sound	microphone ×4 and speaker ×1

Table 2. The API of Kachaka

API	
Command execution	Furniture transportation, destination movement
	Arbitrary position posture movement
	Speed command
	Speech
	Furniture list acquisition, destination list acquisition
	Acquisition of command execution history list
Sensor output	Odometry (self-position and current speed) acquisition
	IMU data acquisition
	LiDAR point cloud data acquisition
	Front camera image acquisition
	Object recognition result (person, furniture, charger, door) acquisition
	Kachaka's map acquisition

also for each row pattern in order to get the robustness for the error[6, 7]. We got the award provided by the Japanese Society of Artificial Intelligence for this research.

4 Conclusion

This year, we use Kachaka instead of Robotino. Kachaka is lower cost and lighter weight than Robotino. So it is good for us to buy and bring to the convention venue. And Kachaka opens a lot of useful API.

To control Kachaka requres ROS2. So we must change a lot of our programs. In order to communicate with RefBox, we provide the ros2-rcll_refbox_peer program. And we publish it at the git repository.

In this time (as the deadline of TDP), we do not decide our policy of this year. We use only API or API and ROS2. And the competition venue is too far from Japan because the Brazil is the opposite place on the earth from Japan. So, it is difficult for our most students to attend RoboCup 2025. We must participate with few human resources. However, BabyTigers-R can attend the Logistics League continuously.

References

1. BabyTigers - R, https://friede.elec.ryukoku.ac.jp/trac/lab/wiki/BabyTigers-R
2. Preferred Robotics, https://www.pfrobotics.jp/
3. Tim Niemueller, RoboCup Logistics League Referee Box ROS Integration, https://github.com/timn/ros-rcll_refbox_peer
4. Wataru Uemura, RoboCup Logistics League Referee Box ROS2 Integration, https://github.com/wadaru/ros2-rcll_refbox_peer
5. Open Computer Vision Library, https://opencv.org/, and https://github.com/opencv/opencv
6. Kosuke Nakajima, Ryota Tanabe, Shohei Yasuda, Nagashima Takeru, and Wataru Uemura: About an Error Correcting Method for a Mobile Robot to Detect ArUco Markers, Proceedings of the 65th Meeting of Special Interest Group on AI Challenges, SIG-Challenge-065-02, 2024.
7. Kosuke Nakajima, Takeru Nagashima, Yuka Komori, Shohei Yasuda, Ryota Tanabe, and Wataru Uemura: About an Error Correcting Method of ArUco Markers Considering Row Data for a Mobile Robot. GCCE 2024: pp. 273 – 275.