Jornadas de Automática

Web-based Interface to control autonomous robotic systems in hospital scenarios

Tirado-Bou, Amparo.[a,*], Marín-Prades, Raúl.[a], Sanz, Pedro J.[a,b]

[a] *Universitat Jaume I de Castellón*
[b] *ValgrAI—Valencian Graduate School and Research Network for Artificial Intelligence (Valencia, Spain)*

To cite this article: Tirado-Bou, Amparo, Marín-Prades, Raúl., Sanz, Pedro J. 2024. Web-based Interface to control autonomous robotic systems in hospital scenarios.
Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10878

Resumen

Desarrollamos una interfaz gráfica para un robot de apoyo autónomo que asiste al personal de salud en el cuidado de pacientes infecciosos, como los de COVID-19. Usando el Framework React, la interfaz está en fase experimental y ha sido probada en un robot real. Su objetivo principal es permitir el control remoto de motores y la visualización de datos Lidar, compatible con cualquier configuración del Sistema Operativo de Robots (ROS) a través de ROSbridge, que utiliza websockets para exponer los canales de comunicación de ROS. El sistema utiliza roslibjs, React y React Three Fiber, aprovechando WebGL y Three.js para una integración fluida en la web. La interfaz de usuario incluye elementos interactivos para las transmisiones de cámara, control de motores y visualización de datos Lidar. Esto mejora las capacidades de ROS más allá de las redes locales, abriendo nuevas aplicaciones para la robótica remota. El módulo de control forma parte de un sistema integral para gestionar tareas relacionadas con el cuidado de pacientes.

Palabras clave:
Robots Móviles Autónomos, Telemedicina, Fusión de información y sensores, Telerobótica, Robots móviles, Sistemas de Control de Movimiento, Trabajo en entornos reales y virtuales, Interacción multimodal, Adquisición de datos de sensores remotos, Programación y Visión

Web-based Interface to control autonomous robotic systems in hospital scenarios

Abstract

A graphical interface has been developed for an autonomous support robot to assist health personnel with the care of infectious patients, such as those with COVID-19. Using the React Framework, the interface design is in the experimental phase, tested on a real robot. Its main goal is to enable remote motor control and Lidar data visualization, compatible with any Robot Operating System (ROS) setup via ROSbridge, which uses websockets to expose ROS communication channels. The system leverages roslibjs, React, and React Three Fiber, utilizing WebGL and Three.js for smooth web integration. The user interface includes interactive elements for camera feeds, motor control, and Lidar data visualization. This enhances ROS capabilities beyond local networks, fostering new remote robotics applications. The control module is part of a comprehensive system for managing tasks related to patient care.

Keywords: Autonomous Mobile Robots, Tele-medicine, Information and sensor fusion, Telerobotics, Mobile robots, Motion Control Systems, Work in real and virtual environments, Multi-modal interaction, Remote sensor data acquisition, Programming and Vision

---

[*] Autor para correspondencia: atirado@uji.es

1. Introduction

In recent years and especially after the appearance of COVID-19, we have seen how various initiatives have proliferated to provide support in hospital environments so that the exposure of healthcare personnel to various sources of contagion could be minimized. We need to work together with health professionals to define their needs and to what extent we can help them in the best possible way.

The implementation of autonomous robotic technologies in isolated hospital environments has emerged as an innovative solution to address various challenges related to healthcare.

This study provides an advanced interface designed to empower users to interact with them, whether it's for mission planning, remote operation, data analysis, or maintenance purposes. They leverage cutting-edge technologies in human-computer interaction, data visualization, and artificial intelligence to provide a seamless and intuitive user experience. Recent advances in the design, development, and application of autonomous robots in hospital contexts, particularly in restricted access areas or isolated settings such as isolation rooms and intensive care units.

The current works are focused on the direct interaction with the robotic system, with a user profile for which the system is prepared. For example, in (Arce et al., 2022), the type of interaction allowed by the GUI is seen in its tests. In other works, such as (Wang et al., 2021), a specific aspect like visual and audio data is also focused on.

In the work presented by (Szafr and Szafr, 2021), it is already taken into account that as robot capabilities advance, human-robot interfaces will increasingly need to support such data-centric activities.

Our objective is for the final interface to allow direct interaction with the robot, the management of its tasks by healthcare personnel, and for it to be intuitive for the patient.

To avoid focusing our interface on direct interaction with the robot, it was decided to create a modular framework with which to work on the different aspects of the interface.

In this paper, a modular framework designed to revolutionize patient care and monitoring in isolated hospital environments is presented. This framework enables the seamless control of autonomous robots that are capable of manipulating their surroundings to assist in critical healthcare tasks, that is specifically tailored to enhance patient monitoring, manage vital signs, and facilitate communication between patients and healthcare professionals, ensuring efficient and effective medical care even in the most isolated settings.

The modular design allows for easy integration and scalability, accommodating various robot types and functionalities based on specific hospital needs, supports multiple modules, each dedicated to specific tasks such as patient monitoring, environmental interaction, and communication. Presented in Section 3

2. React Framework

At the beginning of the project, it was decided to use Python to develop the first version of the interface, and a functional version was developed using the Kivy library.

The robot development was started at the project's inception, and a quick and agile interface was needed to start validating the hardware components.

2.1. Languages and frameworks

Developing a graphical user interface (GUI) for robotics can be accomplished using various programming languages and frameworks. Some of the most popular options and examples are provided below:

1. Python-Based Frameworks, In (Jensen et al., 2014) presents FroboMind[1] a robot control system software platform designed for robotics research and innovation. Then (Velamala et al., 2017) presents UI for control of the WAM-V (Wave Adaptive Modular Vehicle) Autonomous Surface Vehicle a GUI for control of an autonomous surface vehicle, Qt, when combined with ROS, allows for the creation of a powerful GUI for controlling robots and autonomous vehicles.
2. JavaScript and Web Technologies
3. C++ Frameworks, (Sousa et al., 2024) presents a framework for the Robot@Factory 4.0
4. Java Frameworks
5. C# and .NET Frameworks
6. Cross-Platforms Developers Tools as Unity, for example in (Varela-Aldás et al., 2024) describes an open-access simulator for aerial robotic manipulators

A development framework needs to be employed that enables the creation of a multichannel interface (see Figure 1), integrating task management, patient care, and healthcare personnel coordination with robotic devices. Programmable assistance for healthcare staff and the management of medical sensors should be facilitated by this framework, allowing for comprehensive monitoring of both patients and autonomous robots.

Various features and capabilities are offered by these languages and frameworks, which can be leveraged to create effective and efficient robotic GUIs, depending on the specific requirements and constraints of the project.

After a stable version that allowed continued hardware testing was achieved, the requirements that the interface should meet were reviewed.

Once the system control module is functional and allows the progress of tests with the developed prototype, it has been integrated into a complete system that allows the management of the robot's tasks, the monitoring of patients, and facilitates communication with healthcare personnel in complete safety.

A GUI must refer to a system or platform that allows interaction across multiple channels or mediums (smartphones (iOS and Android), tablets, PCs).

[1]https://frobomind.org/

Table 1: Software platform for robotics systems.

	GUI	ROS	Components	Multichannel
FroboMind	Python	✓	C++	NO
Robot@Factory	Rviz	✓	C++	NO
Aerial Robotic	Unity	✓	Python	NO
WAM-V	Qt Creator	✓	C++	NO

2.2. *Advantages and disadvantages of React*

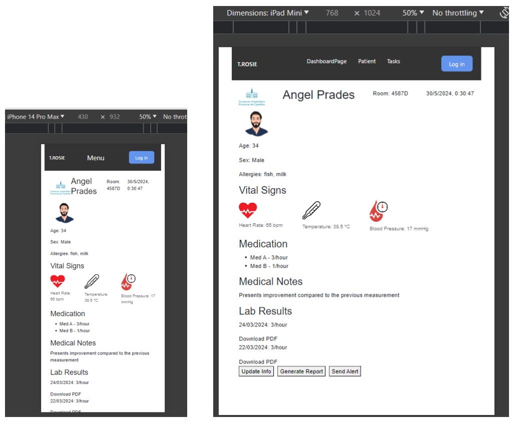
- React is a powerful JavaScript library for building user interfaces, allowing developers to create encapsulated components that manage their own state and logic. This modularity is ideal for complex systems where different parts of the application (such as patient monitoring, robot control, and communication interfaces) can be developed, tested, and maintained independently.

- React promotes the creation of reusable UI components, ensuring consistency and reducing development time. Components like vital sign monitors, alert systems, and communication interfaces can be reused across the application.

- React's Virtual DOM improves performance by minimizing direct manipulation of the actual DOM, leading to faster updates and rendering. This is crucial for real-time updates, such as displaying live patient vitals or robot states, ensuring a smooth and responsive user experience.

- With a vast ecosystem and strong community support, React offers numerous libraries and tools that accelerate development. For example, libraries like Roslib, React Router, and Material-UI can be seamlessly integrated to enhance functionality and user experience.

- React allows the same codebase to be used for building mobile applications through React Native, ensuring a consistent interface across desktops, tablets, and mobile phones, which is beneficial for healthcare providers accessing the system from various devices.



Angel Prades Room: 4587D 30/5/2024, 0:30:47

Age: 34

Sex: Male

Allergies: fish, milk

Vital Signs

Heart Rate: 65 bpm Temperature: 38.5 °C Blood Pressure: 17 mmHg

Medication

- Med A - 3/hour
- Med B - 1/hour

Medical Notes

Presents improvement compared to the previous measurement

Lab Results

24/03/2024: 3/hour

Download PDF
22/03/2024: 3/hour

Download PDF

(a) Iphone view (b) Ipad view

Figure 1: Detail patient page

- React's ease of integration with backend services through APIs is essential for managing data. It facilitates communication with robot control systems, patient monitoring devices, and hospital databases, ensuring smooth integration with backend services.

- Finally, React supports modern JavaScript and TypeScript, enabling developers to use the latest language features and practices, ensuring robust, maintainable, and future-proof code.

Here, a series of works implementing monitoring and control systems using React and Rosbridge are presented.

In (Ivanov et al., 2021), a tool for monitoring and visualization of Robot Operating System (ROS) data in a standard browser is presented. It is compatible with ROS and utilizes ReactJS to visualize topic data.

An example of a web interface developed by S. Nukala, M. Sugaya, and S. Nagarajan (Nukala et al., 2023) is showcased, developed in React and using the rosbridge library to connect with the topics running on the server.

3. Interface Modules

1. Dashboard Module: Centralizes all the system's functionalities and adapts to the profile of the connected user.
   (a) Patient: gives access to the list of patients and the file of the selected patient
   (b) Communications: allows the start of a video call with health personnel
   (c) Robots: allows us access to the list of available robots, their metrics and planning
   (d) Tasks: allows us to manage planned tasks and see their status
2. Task Module: Allows users to define complex missions and tasks for the robotic system
   (a) Task scheduling and mission planning
   (b) Diagnostic tools for troubleshooting and debugging.
   (c) Graphical interface for defining waypoints, areas to explore, and specific tasks to perform.
   (d) Support for multi-agent coordination and collaboration.
   (e) Integration with GIS data for environmental mapping and route optimization.
   (f) Simulation mode for testing and validating missions before deployment.
3. Teleoperation Module: Enables remote control of the robotic system for manual intervention or supervision, (see Figure 2)

(a) Live video streaming from onboard cameras with low-latency feedback.

(b) Virtual joystick or gamepad controls for intuitive navigation.

(c) Fine-grained control over robotic actuators and manipulators.

(d) Emergency stop and safety features for overriding autonomous behaviors.

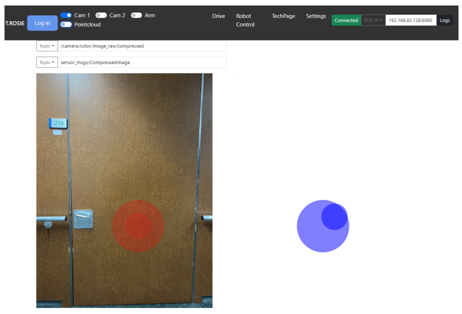


Figure 2: Teleoperation Module.

4. Data Visualization Module: Provides tools for analyzing and visualizing data collected by the robotic system.

(a) 3D reconstruction.

(b) Point cloud visualization for mapping and localization. (see Figure 3).

(c) Integration with machine learning models for real-time object detection and classification.

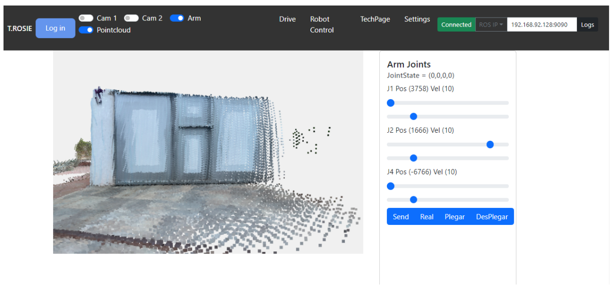


Figure 3: Point cloud visualization.

5. Patient Module: for managing patient data and connecting to sensors for health monitoring.

(a) Patient Listing: Display a list of patients.

(b) Patient Details: View details of a selected patient.

(c) Delete Patient: Remove a patient from the list.

(d) Sensor Data Connection: Connect to sensors and display health monitoring data.

[2]https://www.mongodb.com/

4. Technical Overview

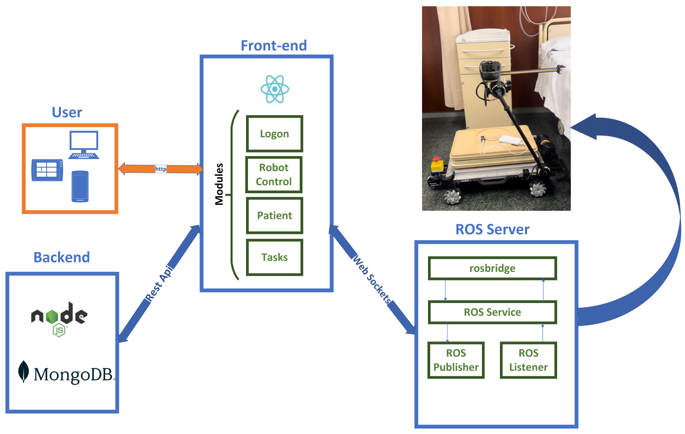


Figure 4: Overall system description.

In the rapidly advancing field of web and robotic technologies, a dynamic and evolving architecture is represented by our current system designed to meet diverse needs. At its core, the system (see Figure 4) utilizes a web server built using the React framework, enabling efficient data access through APIs. Various functionalities, including user authentication, patient management, monitoring, task assignments, and communication, are supported by this architecture. Furthermore, seamless integration with autonomous robots via Rosbridge is facilitated, enabling real-time communication and control through WebSockets. A detailed technical overview of the components and interactions within this system is provided in this article.

The foundation of our system is a web server developed using the React framework. React's component-based architecture and efficient state management make it an ideal choice for building interactive and scalable web applications. User interactions and data flow across different modules are managed by our web server, which serves as the central hub.

4.1. API Integration

To facilitate data access and manipulation, a suite of RESTful APIs is exposed by the web server. These APIs handle various critical functions, including:

- **Login**: Secure authentication mechanisms to ensure that only authorized users can access the system.
- **Patients**: Management of patient records, including creation, retrieval, updates, and deletions.
- **Monitorizations**: Real-time monitoring of patient data, allowing healthcare providers to track vital signs and other health metrics.
- **Tasks**: Assignment and tracking of tasks to ensure that all system operations are carried out efficiently.
- **Communications**: Enabling secure and efficient communication between different users and system components.

4.2. *MongoDB Atlas*

Managing vast amounts of data efficiently is crucial. MongoDB Atlas[2] , a cloud-based NoSQL database service, provides robust and scalable solutions for storing, querying, and analyzing data. Combined with a RESTful API, can seamlessly interact with their data from various applications and platforms, simplifies the deployment and management of MongoDB databases.

In Figure 5, an initial version of the database used can be seen, containing the basic management data for the robot's missions. This includes a table with the tasks to be performed, linking the patient, their room, and the robot assigned to carry out the task. Additionally, the monitoring data collected from the patient will be stored.

Furthermore, the management of the different types of users who will utilize the application must be addressed, as varying profiles (types) will exist: doctors, nurses, patients, and technicians. Depending on the user type, access to different modules of the system will be granted.

4.3. *Rosbridge*

The integration of autonomous robots into our system is facilitated through Rosbridge[3], a middleware layer that allows for seamless communication between web applications and Robot Operating System (ROS) nodes. WebSockets are used by Rosbridge to establish real-time, bidirectional communication channels with ROS, enabling precise control and monitoring of robotic functionalities. A persistent connection between the web server and the ROS Master is provided by WebSockets, allowing for low-latency data exchange and control commands. This setup is crucial for real-time applications where immediate feedback and control are required.

4.4. *Autonomous Robot Features*

Various features of autonomous robots are supported by our system, including:

- **Position Tracking**: Real-time updates on the robot's location and movement within its operational environment.
- **Sensor Data**: Access to data from the robot's sensors, such as cameras, lidar, and ultrasonic sensors, which are essential for navigation and obstacle detection.
- **Task Execution**: The ability to send commands to the robot to perform specific tasks, such as moving to a designated location or interacting with objects in its environment.

An evolving system, built on a robust React framework and integrated with autonomous robots via Rosbridge, is exemplified by our approach to web and robotic technologies. APIs for data access and WebSockets for real-time communication are leveraged to ensure a responsive and efficient platform capable of meeting diverse application requirements. This integration not only enhances user interaction but also extends the capabilities of autonomous robots, paving the way for innovative applications in various fields.

[3]http://wiki.ros.org/rosbridge_suite

[4]https://www.youtube.com/watch?v=K6JbrTWOzTIt=8s

4.5. *Context in React*

In the development of complex web applications, especially those with multiple interconnected modules, the efficient management and sharing of state are crucial. A powerful and efficient solution for passing data through the component tree without the need to manually pass props down at every level is provided by React's Context API.

The advantages of using React Context for sharing information between different modules during a session can be seen in our system.

The problem of prop drilling, where data must be passed through many layers of components even if only a few of them need it, is solved by the Context API in React. A global state that can be accessed by any component within the application is created by Context, providing a cleaner and more maintainable codebase.

Code maintainability, readability, and performance are enhanced by providing a centralized and efficient way to handle global state with Context. In our system, seamless integration and synchronization of data across different functionalities are enabled, ensuring a cohesive and robust user experience. React Context is leveraged to ensure that our application remains scalable and easy to maintain as it continues to evolve. Main benefits:

A way to centralize the state that is needed across multiple components is provided by Context, this results in a more readable and understandable codebase, furthermore updates or changes to the global state are made simpler.

In our system, which includes user authentication, patient management, real-time monitoring, task assignments, and communication, a vital role in managing and sharing session-based information is played by React Context (see Figure 6).

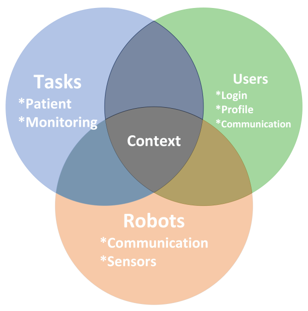


Figure 6: Application of React Context in Our System.

5. Experimental results

The GUI created using React over the ROS framework provides us with a simple and easy to use control interface.

Due to the flexibility of React and ROS, adding new modules to the code is highly simplified, as each module is implemented as a separate component with the only main requirements being the topic to which data is being published and the data type.

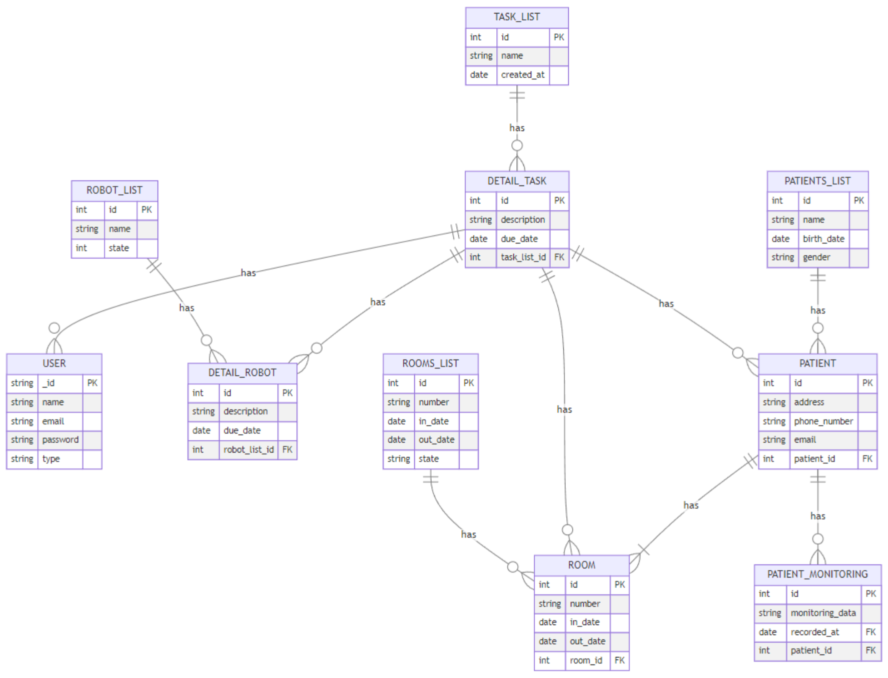


Figure 5: Entity-Relationship (ER) diagram

During the preliminary tests, the acceptance of the device by end users and its ability to interact with its environment have been observed (see evidence [4].

6. Conclusion and future work

This work methodology has allowed progress in other aspects such as teaching, enabling students to be involved in the project, utilizing the generated prototypes, and contributing their vision and ideas to the project.

Looking ahead, enhancements to our system are planned by incorporating advanced features such as machine learning for predictive analytics, enhanced security protocols, and more sophisticated robotic capabilities. This continuous evolution will enable us to stay at the forefront of technology and provide more robust and advanced solutions to our users.

Acknowledgements

This work has been carried out by the CIRTESU team, "Center for Research in Robotics and Underwater Technologies" at the University Jaume I, IRS-Lab group (Interactive and Robotic Systems Lab), UJI-B2021-30 (AUDAZ), and the project H2020-Peacetolero-NFRP-2019-2020-04. The authors wish to thank the "Facultad de Ciencias de la Salud" and "Hospital Provincial de Castellón" for the support they provided, enabling us to conduct preliminary tests in a real-world setting.

References

Arce, D., Balbuena, J., Menacho, D., Caballero, L., Cisneros, E., Huanca, D., Alvites, M., Beltran, C., Cuellar, F., 2022. Design and implementation of telemarketing robot with emotion identification for human-robot interaction. In: 2022 Sixth IEEE International Conference on Robotic Computing (IRC). pp. 177–180.
DOI: `10.1109/IRC55401.2022.00037`

Ivanov, A., Zakiev, A., Tsoy, T., Hsia, K.-H., 2021. Online monitoring and visualization with ros and reactjs. In: 2021 International Siberian Conference on Control and Communications (SIBCON). pp. 1–4.
DOI: `10.1109/SIBCON50419.2021.9438890`

Jensen, K., Larsen, M., Nielsen, S. H., Larsen, L. B., Olsen, K. S., Jørgensen, R. N., 2014. Towards an open software platform for field robots in precision agriculture. Robotics 3 (2), 207–234.
URL: `https://www.mdpi.com/2218-6581/3/2/207`
DOI: `10.3390/robotics3020207`

Nukala, S., Sugaya, M., Nagarajan, S., 2023. Web based lidar point cloud visualization and teleoperation tool for robots. In: 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT). pp. 1–6.
DOI: `10.1109/ICCCNT56998.2023.10307953`

Sousa, R. B., Rocha, C. D., Martins, J. G., Pedro Costa, J., Padrão, J. T., Sarmento, J. M., Carvalho, J. P., Lopes, M. S., Costa, P. G., Moreira, A. P., 2024. A robotic framework for the robot@factory 4.0 competition. In: 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). pp. 66–73.
DOI: `10.1109/ICARSC61747.2024.10535935`

Szafr, D., Szafr, D. A., 2021. Connecting human-robot interaction and data visualization. In: 2021 16th ACM/IEEE International Conference on Human-Robot Interaction (HRI). pp. 281–292.

Varela-Aldás, J., Recalde, L. F., Guevara, B. S., Andaluz, V. H., Gandolfo, D. C., 2024. Open-access platform for the simulation of aerial robotic manipulators. IEEE Access 12, 49735–49751.
DOI: `10.1109/ACCESS.2024.3384986`

Velamala, S. S., Patil, D., Ming, X., 2017. Development of ros-based gui for control of an autonomous surface vehicle. In: 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO). pp. 628–633.
DOI: `10.1109/ROBIO.2017.8324487`

Wang, H., Li, X., Zhang, X., 2021. Multimodal human-robot interaction on service robot. In: 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). Vol. 5. pp. 2290–2295.
DOI: `10.1109/IAEAC50856.2021.9391068`