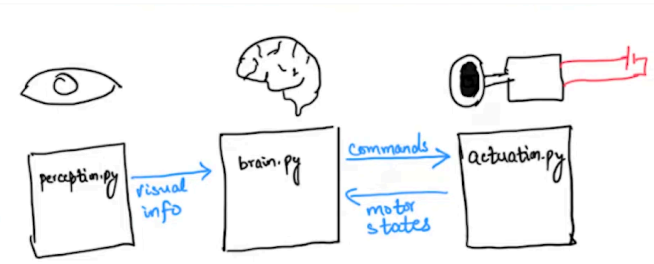
Communication in ROS

Effective communication between different modules is crucial in a robotic system, enabling seamless data exchange and coordination. In ROS (Robot Operating System), three primary communication methods facilitate interactions between different components: **messages, services, and actions**. Each method serves a unique purpose, depending on whether the communication needs to be continuous, request-based, or interactive with feedback.

Example of a Trash-Picking Robot

To understand how these communication methods work, consider a trash-picking robot. This robot continuously detects trash, moves toward it, picks it up, and deposits it into a bin. Its software consists of three key modules:

1. **Perception** – Detects and identifies trash.
2. **Brain** – Processes the perception data and decides on actions.
3. **Actuation** – Controls the robot's movement and physical interactions.



For the robot to function correctly, these modules must exchange data efficiently. The perception module must communicate visual information to the brain, the brain must send movement commands to the actuation module, and the actuation module must report motor states, such as battery levels, back to the brain.

Communication Method 1: Messages (Topics)

One of the fundamental communication methods in ROS is **message passing** via **topics**. A **topic** acts as a channel where one module publishes data, and other modules subscribe to receive updates. This model is useful for continuous, asynchronous data exchange.

- In the trash-picking robot, the **perception module publishes trash detection data** on a specific topic.
- The **brain subscribes to this topic**, staying updated on detected trash.
- This setup ensures that whenever the perception module identifies trash, the brain module immediately receives the information.

The key characteristic of topic-based messaging is **asynchronous communication**—the publisher can send updates at any time, and subscribers receive them as they occur without needing to request data explicitly.

Communication Method 2: Services

Unlike topics, **services** allow a module to request specific information or trigger an action from another module in a **client-server model**. This method is useful for request-response interactions rather than continuous data flow.

- In the trash-picking robot, the **brain may need to check the battery status** before executing a movement command.
- The **brain sends a request to the actuation module (server)** to retrieve battery information.
- The **actuation module responds with the requested battery level**.

This approach ensures that the brain only retrieves the data **when necessary**, reducing unnecessary data transmission. Services also support requests with parameters, allowing for specific queries (e.g., checking the battery level of a particular motor).

Communication Method 3: Actions

Actions are a more advanced communication method used for tasks that require **feedback and the ability to cancel execution**. Unlike services, which are single request-response interactions, actions provide ongoing status updates during task execution.

- Suppose the **brain instructs the actuation module to move the robot to a specific position**.
- The **actuation module continuously sends feedback** about the robot's current location.
- If needed, the **brain can cancel the movement command mid-execution**—a feature not available with services.

Actions are particularly useful for **long-running operations** where real-time monitoring is necessary, such as motion control, navigation, and robotic arm manipulation.

Differences in ROS Versions

One notable difference between **ROS 1 and ROS 2** is how they handle service calls. In **ROS 1**, service calls are **blocking**, meaning the system must wait for a response before proceeding. In **ROS 2**, non-blocking calls allow the program to continue execution while waiting for a response, improving efficiency in distributed systems.

Conclusion

ROS provides **three primary communication methods** to handle different types of interactions in a robotic system:

1. **Messages (Topics)** – Used for continuous, asynchronous data sharing.
2. **Services** – Best suited for request-response interactions.
3. **Actions** – Ideal for tasks requiring feedback and the ability to cancel execution.

Choosing the appropriate method depends on the specific needs of the system. Understanding these communication mechanisms is essential for designing efficient, scalable robotic applications.