Standard Operating Procedure (SOP) for Magnet Control System Using ROS2 Services

Overview

This SOP details the steps to operate a magnet control system utilizing ROS2 services. The system comprises a client (MagnetClient) that sends requests to toggle the state of a magnet and a server (MagnetServer) that executes these requests. The server runs on a Raspberry Pi connected to a magnet through a relay, and the client can run on any computer on the same network as the Raspberry Pi. This guide assumes the IP of the Raspberry Pi is 192.168.202.150.

Prerequisites

- Basic understanding of Raspberry Pi4, ROS2, SSH (Secure Shell), and Python 3.
- ROS2 installed on both the client computer and the Raspberry Pi.
- Network connectivity between the client computer and the Raspberry Pi.
- Physical setup involving a relay connected to the Raspberry Pi and the magnet.

Equipment and Software Setup

Raspberry Pi Connection:

- Ensure the Raspberry Pi is connected to the magnet via a relay on GPIO pin 26.
- Connect the Raspberry Pi to the network.

Client Computer Preparation:

- Verify network connectivity to the Raspberry Pi.

ROS2 Environment:

- Confirm ROS2 is installed and configured on both the client computer and Raspberry Pi.

Script Preparation:

- Place the server script (MagnetServer) on the Raspberry Pi.
- Place the client script (MagnetClient) on the client computer.

Operational Procedure

Step 1: Start the Server on the Raspberry Pi

Open Terminal on Client Computer:

- Access the terminal or command prompt.

SSH into Raspberry Pi:

- Use the command ssh [username]@192.168.202.150, replacing [username] with your Raspberry Pi's username.

Navigate to Server Script Location:

- Use cd to navigate to the directory containing the server script.

Run the Server Script:

- Execute python3 <server_script_name.py>, replacing <server_script_name.py> with the actual server script filename.
- The server will start and wait for service requests.

Step 2: Operate the Client to Toggle the Magnet

Open a New Terminal on the Client Computer:

- You do not need to close the SSH session; simply open a new terminal window/tab.

Navigate to Client Script Location:

- Use cd to navigate to the directory containing the client script.

Run the Client Script:

- Execute python3 <client_script_name.py>, replacing <client_script_name.py> with the actual client script filename.
- Follow the prompts in the client script to toggle the magnet on or off, or to exit the script.

Important Notes

- Concurrent Operation: The server must be running before starting the client script to ensure service availability.
- Network Configuration: Ensure both the Raspberry Pi and the client computer are on the same network to facilitate communication.
- Security: Use secure passwords and follow best practices when accessing devices over the network.

Troubleshooting

- Service Unavailability: If the client reports waiting for the server, verify that the server script is running on the Raspberry Pi and the network connection is intact.
- GPIO Issues: Ensure the GPIO pin number and setup match the physical connections on the Raspberry Pi.
- ROS2 Configuration: Verify ROS2 environments are sourced correctly on both the client and server sides.

This SOP provides a structured approach to controlling a magnet via a ROS2 service, ensuring clear and efficient operation from setup to execution.

Understanding the Magnet Control System Code

MagnetServer Code Explained

The MagnetServer script is designed to run on a Raspberry Pi and acts as a ROS2 service server to control a magnet through GPIO pins. Here's how it works:

Initialization: The server node, named 'magnet_server', is initialized and a ROS2 service named 'toggle_magnets_service' is created. This service uses the SetBool service type, which takes a boolean request and returns a boolean response.
GPIO Setup: The GPIO pin connected to the relay (pin 26 by default) is set up as an output. This setup is crucial for controlling the magnet (turning it on/off).
Request Handling: The handle_toggle_request method is called whenever a request is received. It logs the request, toggles the GPIO pin based on the request data (True to turn the magnet on, False to turn it off), and returns a success response.
Cleanup: Upon destruction of the node, GPIO resources are cleaned up to ensure no resources are left hanging, preventing potential conflicts or issues in subsequent runs.

MagnetClient Code Explained

The MagnetClient script is intended to run on any computer within the same network as the Raspberry Pi. It acts as a client sending requests to the MagnetServer to control the magnet. Here's its workflow:

Initialization: The client node, named 'magnet_client', is initialized, and it creates a client for the 'toggle_magnets_service' service.
Service Wait: The client waits for the service to become available. This is crucial for ensuring the server is ready to process requests.
User Input: The script prompts the user to turn the magnet on, off, or exit. Based on the input, it sends a service request to toggle the magnet's state.
Request Sending: A SetBool request is sent to the server. The call_async method is used for asynchronous communication, allowing the client to continue running without blocking while waiting for a response.
Response Handling: Once the request is processed, the server's response is logged. This indicates whether the action (turning the magnet on/off) was successful.

How They Work Together

- Service Communication: The client sends a toggle request (on/off) to the server via the toggle_magnets_service ROS2 service. The server receives this request, performs the action on the GPIO pin to control the magnet, and responds back with the success status of the operation.

- Network Requirement: Both the client and server must be on the same network. This allows for seamless communication between the two scripts, utilizing ROS2's underlying network communication mechanisms.
- ROS2 Framework: Utilizing ROS2 services provides a reliable and efficient way to communicate between different parts of a robotics system or any system requiring remote control functionality. The SetBool service type simplifies the request-response mechanism, making it ideal for binary (on/off) operations like magnet control.

This setup demonstrates a practical application of ROS2 services for controlling hardware devices remotely, ensuring modularity, scalability, and ease of integration into larger systems. The separation of client and server scripts allows for distributed operation, making it suitable for various applications where remote control of devices is needed.