Thor: A 6-Axis Robotic Arm with ROS 2 and AI

Welcome to the Thor project! This repository contains the complete software stack to build, simulate, and control a 6-axis robotic arm using ROS 2, MoveIt 2, and advanced AI capabilities with LeRobot. This project is designed to be a comprehensive platform for robotics research, learning, and development, from basic teleoperation to complex, AI-driven manipulation tasks.

✨ Features

- **Complete 6-Axis Arm:** A fully articulated 6-DoF robotic arm designed for manipulation tasks.
- **ROS 2 Humble Integration:** Built on the modern, stable ROS 2 Humble Hawksbill distribution.
- **Realistic Simulation:** High-fidelity Gazebo simulation for safe testing and development.
- **Advanced Motion Planning:** Full integration with **MoveIt 2** for collision-aware trajectory planning.
- **Dual Teleoperation:** Control the robot in real-time using either:
  - **Joint-based control** for precise individual joint movements.
  - **Pose-based (Cartesian) control** to intuitively "fly" the end-effector.
- **Physical Hardware Interface:** A complete `ros2_control` hardware stack to bridge the software with physical motors and encoders using a Raspberry Pi and Arduino.
- **AI-Ready Data Collection:** A dedicated package for collecting high-quality, multi-camera demonstration data using a joystick, perfectly formatted for training AI policies with **LeRobot**.

🏗️ System Architecture

The Thor project uses a modular architecture that separates simulation from hardware, allowing for seamless development and deployment.

- **High-Level Control (ROS 2 on Raspberry Pi):** Handles motion planning (MoveIt), teleoperation, and AI policy inference.
- **Low-Level Control (Firmware on Arduino):** Manages real-time tasks like reading encoders and running PID loops to drive the motors.
- **Communication:** The Raspberry Pi and Arduino communicate over a standard USB serial connection.

📦 Package Overview

This workspace is organized into several key ROS 2 packages:

- `thor_urdf`**:** Contains the robot's 3D model (URDF/XACRO) and physical properties.
- `thor_moveit_config`**:** The configuration package for MoveIt 2, including the SRDF and planning pipelines.
- `thor_teleop`**:** Provides keyboard-based teleoperation nodes for both joint and pose control.
- `thor_hardware`**:** The crucial hardware interface package. Contains the C++ `ros2_control` driver, Arduino firmware, and launch files for the physical robot.
- `thor_learning`**:** The AI data collection package, featuring a powerful script for joystick-based teleoperation and synchronized multi-camera data recording for LeRobot.

🚀 Getting Started

Prerequisites

1. **Ubuntu 22.04** with **ROS 2 Humble Hawksbill** installed.
2. **MoveIt 2:** Install the MoveIt binaries: `sudo apt install ros-humble-moveit`
3. **ros2_control & Gazebo:** Install the necessary packages:
```
sudo apt install ros-humble-ros2-control ros-humble-ros2-controllers
ros-humble-gazebo-ros2-control
```
4. **Colcon:** The standard ROS 2 build tool.
5. **Python Dependencies (for AI):**
```
pip install h5py
```

Installation

1. Clone the Repository:
   Clone this repository into your ROS 2 workspace's src folder.
```
cd ~/your_ros2_ws/src
git clone https://github.com/anishk85/common-dp.git
```
2. Install Dependencies:
   Navigate to your workspace root and install any missing dependencies using rosdep.
```
cd ~/your_ros2_ws
rosdep install --from-paths src --ignore-src -r -y
```
3. Build the Workspace:
   Build all the packages using colcon.
```
colcon build --symlink-install
```

🕹️ Usage

Before running any command, always source your workspace in a new terminal: `source install/setup.bash`.

1. Running the Simulation (Gazebo)

This is the best way to test the robot's functionality without any hardware.

```
ros2 launch thor_urdf demo_gazebo.launch.py
```

This command will start Gazebo, spawn the Thor arm, and launch MoveIt, RViz, and all necessary controllers. You can now use the MoveIt RViz plugin to plan and execute motions.

2. Controlling the Physical Robot

A. Hardware Setup:

- Ensure you have assembled the hardware as per the recommended stack (Raspberry Pi, Arduino, goBILDA motors).
- Upload the firmware from `thor_hardware/firmware` to your Arduino Mega.
- Connect the Arduino to the Raspberry Pi via USB.
- Power on the motor power supply.

B. Launching the Hardware:
To start the drivers, ros2_control, and MoveIt for the physical robot, run the main hardware launch file:

```
ros2 launch thor_hardware hardware.launch.py
```

3. Using the Teleop Nodes

Once the simulation or the physical robot is running, you can control it using the keyboard. Open a new terminal and run one of the following commands:

- **For Joint-by-Joint Control:**
  ```
  ros2 run thor_teleop teleop_joint_control.py
  ```
- **For End-Effector Pose (Cartesian) Control:**
  ```
  ros2 run thor_teleop teleop_pose_control
  ```
  (Note: This runs the C++ version, which is typically more performant).

4. Collecting AI Data with LeRobot

Follow the detailed instructions in the `thor_learning/README.md` file to set up your cameras and joystick. The general launch sequence is:

1. **Launch Hardware:** `ros2 launch thor_hardware hardware.launch.py`
2. **Launch Joystick:** `ros2 run joy joy_node`
3. **Launch Cameras:** Start your camera drivers (e.g., `v4l2_camera_node`).
4. **Launch Recorder:**
   ```
   ros2 run thor_learning data_recorder.py
   ```

   You can now use the joystick to control the arm and record demonstrations for training an AI policy.

🛠️ Hardware Guide

For a detailed list of recommended components and a complete wiring diagram, please refer to the documents generated during our development session. The two main hardware stacks are:

1. **goBILDA DC Motor Stack:** Simpler assembly with integrated encoders (Recommended).
2. **NEMA Stepper Motor Stack:** More powerful but requires complex assembly with external encoders.

🗺️ Roadmap

- [x] URDF and Simulation Setup
- [x] MoveIt 2 Integration
- [x] Keyboard Teleoperation (Joint & Pose)
- [x] `ros2_control` Hardware Interface for Physical Robot
- [x] AI Data Collection Package for LeRobot
- [ ] **Next:** Train an ACT / Diffusion Policy with the collected data.
- [ ] **Next:** Create an "inference" node to run the trained policy on the robot.