This build process needs a better way to check for errors in the build.

Linear Slider Drivers

This is a package that includes the hardware description, hardware interface, and controller interface for the linear slider.

Running the linear slider

Simulation

Gazebo

Real-world

Actual hardware

Package Contents

`linear_slider_bringup`: Launches all of the interfaces.

`linear_slider_controllers`: Manages the control node for interfacing with ROS2.

`linear_slider_description`: Defines the geometry and links in a URDF file. Also specifies controller interfaces through `<ros2_control>` tag.

`linear_slider_hardware_interface`: Defines the hardware and controller interfaces for the linear slider

Testing the linear slider

Each of the packages has a set of launch files that help guide the creation of your drivers. A brief description of each of the test files is below:

1. `linear_slider_description/launch/view_robot.launch.py` -- Spawns a `joint_state_publisher_gui` window, allowing the user to monitor the URDF's build process.
2. 'linear_slider_bringup/launch/linear_slider.launch.py`-- Spawns the`/controller_manager` node, allowing the activation of various controller libraries. This also links the controller interface to the hardware interface.

Linear Slider Description:

Default state interface values can be found in `linear_slider_description/config/initial_state.yaml`.

Under the `<ros2_control>` tag, there are two ros2_control hardware interface types, **Joints** and **Sensors**. The `<joint>` tag defines the state and command interfaces for the controllers defined in `linear_slider_bringup/config/linear_slider_controllers.yaml`.

Hardware Interface

This package defines the UDP communication with the external microcontroller, and presents a custom `LinearSliderHardware` class to store the state of the system. The class also translates the rpm of the motor the the horizontal distance of the `moving_base` and vice versa. The class is an attribute within the high-level `LinearSliderSystemInterface` class.

This high-level class inherits the boiler-plate ros2_control `hardware_interface::SystemInterface` class, where several class methods must be overwritten:

- `on_init()`: runs at the startup, reads parameters, allocates memory, etc. Hardware enters unconfigured state.
- `on_configure()`: Establish comms. Takes us to the inactive state
- `on_activate()`: Engage actuators. Takes us to the active state. Here, we can send read/write commands.
- `on_deactivate()`: Disengage actuators. Takes us back to the inactive state.
- `on_cleanup()`: Takes us from the inactive state to the unconfigured state, disabling comms.
- `on_shutdown()`: Shuts down the whole thing (gracefully).
- `read()`: Read the value from the external device.
- `write()`: Write a value to the external device.

These methods provide us with a simple way to engage, communicate, and disengate with our external hardware. Much of the functionality takes place within the `read` and `write` methods. The values are stored in `LinearSliderSystemInterface::linear_slider_` and eventually published to the `/joint_states` topic.

Linear Slider Description: Defines the physical robot. This includes STL meshes, URDF and SRDF files, MoveIt configurations, etc.

Things to look at:

```
https://control.ros.org/master/doc/gazebo_ros2_control/doc/index.html
```