README

This repo contains the ROS 2 (Foxy) packages with the robot description, Moveit configuration files and controller files.

What is this repository for?

The repository is used for motion planning and control of the cartesian positioner using Moveit and ros2_control. This includes controlling the cartesian positioner in both static and dynamic scenarios. Note: For controlling the actual hardware the odrive_ros2_control repository is required.

How do I get set up?

- Checkout this repository
- Checkout the odrive_ros2_control repository in the src folder
- run `colcon build`

Usage guide

Currently, there are two demonstrations available: without (1) and with (2) belt movement. After startup the motors/encoders must be calibrated and homed first. Procedures to do this and to start one of the demonstrations are described below.

General note

Please note that in the current implementation, joint limits are not set, as this would not work with the dynamic scenario. Currently, hard limits are set in the transmission that combines the planning and compensating joints, but this **does not guarantee** that the joint cannot move beyond its limits.

Calibration and Homing

Before using the positioner, the motors must be calibrated and homed. For safety, the calibration does not happen automatically when powering on the oDrives, as this would let the motors move directly on power on. Instead the calibration sequence has been combined with the homing sequence. Homing will ensure that the oDrives are at a known position before using the positioner. It is not possible to set the motors into any type of control mode before the calibration and homing is performed.

In order to run the calibration and homing sequence, run the `homing_python.py` script located at:

```
~/ros2_ws/src/cartesian_positioner/src/cartesian_positioner_controller/scripts/homing_python.py
```

Note: Please note that the homing sequence is currently sensorless, meaning that no homing switches are used. Homing is done based on velocity, so manually stopping the positioner

before it has reached its endstop will cause it to home at the wrong position, which will cause issues during usage. Furthermore, note that the positioner will move in positive X, Y and Z directions at the start of the calibration and homing sequence in order to make enough room to home into the negative directions. Because of this, please ensure that there is enough room in the positive directions before starting the calibration and homing sequence.

Starting the demonstration with the conveyor belt at a fixed stationary position

For the static demonstration the belt has to be aligned such that pre-programmed weed and crop postiions correspont to the actual positions of these weeds and crops on the belt. To do so, make sure that the retroreflective tape on the conveyor belt is aligned with the **green/yellow** tape next to the belt. The belt can be operated by pressing the green start button on the FRENIC inverter to the right of the control cabinet. Speed can be adjusted with the knob.

Launching the background nodes

In order to run any of the programs, firstly the background nodes like the `move_group` and `ros2_controllers` have to started. This can be done by running the `all_control` launch file:

```
ros2 launch cartesian_positioner_controller all_control.launch.py
```

Launching rviz (optional)

When desired, RVIZ be launched seperately from the background nodes. This can be done by running the `rviz` launch file:

```
ros2 launch cartesian_positioner_controller rviz.launch.py
```

Launching the static demo scenario

To run the static demo scenario, ensure that the system is calibrated and homed and make sure the background nodes are running. Next run the 'demo_mode_static` launch file:

```
ros2 launch cartesian_positioner_controller demo_mode_static.launch.py
```

Starting the demonstration with a moving conveyor belt

To start the demonstration with moving belt properly, the belt has to be stationary first. In order to stop the conveyor belt, the red button on the FRENIC inverter can be pressed. Steps are similar compared to the static demonstration and are described below.

Launching the background nodes

In order to run any of the programs, firstly the background nodes like the `move_group` and `ros2_controllers` have to started. This can be done by running the `all_control` launch file:

```
ros2 launch cartesian_positioner_controller all_control.launch.py
```

Launching rviz (optional)

When desired, RVIZ be launched seperately from the background nodes. This can be done by running the `rviz` launch file:

```
ros2 launch cartesian_positioner_controller rviz.launch.py
```

Launching the dynamic scenario

To run the dynamic scenario, ensure that the background nodes are running. Next run the 'demo_mode_dynamic` launch file:

```
ros2 launch cartesian_positioner_controller demo_mode_dynamic.launch.py
```

Make sure you get proper output in the console from the `read_conveyor` node. If not, you have to change the USB port defined in the `read_conveyor.py` script, such that the node is able to communicate to the arduino which receives belt positions.

After the nodes are properly started, the conveyor belt can be started using the green button on the FRENIC inverter. The system needs to know the start point of the belt: the pre-programmed positions are sent when the retroreflective marker passes the IR sensor below the belt. The first crops and weeds which are sent are those after the retroreflective tape. So the robot will only start to move after the retroreflective tape has passed the yellow tape next to the belt.

Starting the demonstration in simulation only

It is also possible to run the software without the physical demonstrator setup. In order to do so, the launch argument `sim:=true` can be used for **both** the `all_control` and the `demo_mode_dynamic` launch files:

```
ros2 launch cartesian_positioner_controller all_control.launch.py sim:=true
```

```
ros2 launch cartesian_positioner_controller rviz.launch.py
```

```
ros2 launch cartesian_positioner_controller demo_mode_dynamic.launch.py
sim:=true
```