Using a Device in a Template

Universal Robots A/S

Version 1.14.0

Abstract

In PolyScope versions 3.11.0/5.5.1, gripper drivers were introduced. This was the first concept of a device driver/contribution in PolyScope. With the introduction of PolyScope version 3.12.0/5.6.0, it is now possible to use such devices in templates. This document will describe how to do so using a gripper-based template as a running example.

Contents

1 Introduction

Creating template URCaps is a powerful way of helping the end user to easily configure the robot to perform a specific complex task. Prior to the introduction of the concept of devices in PolyScope, a template URCap developer would have to rely on inserting a Set program node, Comment program node or similar to have the user do the final configuration of a device operation.

By using a device-based program node this is no longer necessary. The template program node can automatically insert the desired device node and configure it to perform its operation.

The following sections will describe the pick functionality of a URCap template program node for executing a Pick and Place operation. The template node will use a Gripper program node for performing the grip and release actions. Even though a gripper is used in the example, the steps taken can be generalized to working with other Polyscope devices as well.

2 Getting a List of Devices

When working with devices, it is first necessary to get the appropriate device manager. This is done using the `getDeviceManager(Class)` method in the `ProgramAPI` interface (which is accessible through the `ProgramAPIProvider` interface). The method should be called with the class of the specific device manager. Using `GripperManager.class` as parameter will return the gripper manager interface. This interface can, for instance, be used to retrieve a list of the installed PolyScope gripper devices. An example of this can be seen in the code snippet in Listing 1.

Listing 1: Getting the list of installed gripper devices

```
1  GripperManager gripperManager = apiProvider.getProgramAPI().getDeviceManager(
       GripperManager.class);
2  List<GripperDevice> grippers = gripperManager.getGrippers();
```

The retrieved list can be presented to the end user in a combo box, so he can select which gripper to use in the template. If only one gripper device is installed, this could be used directly. If no gripper devices are installed, the end user should be informed of this and appropriate guiding actions of how to resolve this should also be presented.

3 Adding a Device Program Node

When the device to use in the template has been selected, this will now form the basis for the program node(s) to be inserted in the template. To create a device-based program node, the device node factory must be used. This factory is found in the device manager inteface described in the previous section. Call the `getGripperProgramNodeFactory()` method in this interface to get the node factory. On the node factory interface, call the `createGripperNode(GripperDevice)` method using the selected device as argument. This will return a new program node which can be inserted as a child of the template node.

A root tree node is required to insert a child program node. To obtain the root tree node of the template, use the `getRootTreeNode()` method in the `ProgramModel` interface available in the `ProgramAPI` interface used in the previous section. An example of this is shown in the code snippet in Listing 2.

Listing 2: Inserting a Gripper program node

```
1  GripperDevice selectedGripper = getSelectedGripper();
```

```
GripperNodeFactory factory = gripperManager.getGripperProgramNodeFactory();
GripperNode gripperNode = factory.createGripperNode(selectedGripper);
TreeNode root = programModel.getRootTreeNode(this);
try {
  root.addChild(gripperNode);
} catch (TreeStructureException e) {
  e.printStackTrace();
}
```

4 Configuring the Device Program Node

The Gripper program node created by the factory can be configured to help the end user of the template as much as possible. This is done by creating the desired configuration using one of the configuration builders provided by the `GripperNode` interface. A call to the `build()` method will return a new instance of (a subtype of) `GripperNodeConfig` interface with the desired configuration. Before calling the `build()` method, the builder can potentially have other setter methods that can be used for setting different parameters of the configuration (see next section for more details).

The newly built configuration instance can now be set on the Gripper program node itself. An example of this can be seen in Listing 3.

Listing 3: Configuring a Gripper program node

```
GripActionConfigBuilder builder = node.createGripActionConfigBuilder();
GripActionConfig config = builder.build();
node.setConfig(config);
```

4.1 Capability Support

The various devices (of a specific type, e.g. grippers) available in PolyScope can support different sets of capabilities ranging from a very basic device to a more advanced device supporting several capabilities. A device can optionally provide additional capabilities besides the basic mandatory functionality that all devices of that type must support. In the case of a gripper device, the device must support basic grip and release functionality and it can additionally offer more advanced capabilities, such as force gripping and support for multiple individual grippers (i.e. the multi-gripper capability).

All the specific capabilities, that a gripper device can optionally support, are represented by interfaces extending the `CapabilitySupport` base interface. Each interface provides the information required for configuring the supported capability for a specific gripper device (the information is not applicable for other gripper devices). This can used to configure a Gripper program node which uses that specific gripper device.

An interface representing the support for a specific capability can be obtained using the method `getCapabilitySupport(Class<T>)` in the `GripperDevice` interface. The method should be called with the class of the specific capability support. Listing 4 shows an example of how to access the support for the multi-gripper capability using `MultiGripperSupport.class` as parameter to the `getCapabilitySupport(Class<T>)` method.

Listing 4: Getting the support for the multi-gripper capability for a gripper device

```
Optional<MultiGripperSupport> multiGripperSupport = gripperDevice.
    getCapabilitySupport(MultiGripperSupport.class);
```

The `getCapabilitySupport(Class<T>)` method can also be used just to check if the device supports a specific capability (or set of capabilities). The result can be used to determine how to configure the gripper device or to help decide which gripper device to select based on its capabilities.

The capability support interface returned by the `getCapabilitySupport(Class<T>)` method is wrapped in an `Optional` instance (as seen in Listing 4). The `isPresent()` method of `Optional` instance can be used to determine, if the specific gripper device supports the specified capability. If the capability is supported, the capability support interface can be accessed using the `Optional` interface's `get()` method.

Since the various grippers available in PolyScope can support different sets of capabilities, the `isPresent()` method should always be used to check, if a specific capability is available before calling the `get()` method. Failing to do so will result in a `NoSuchElementException` being thrown, if the capability is not supported.

The example in Listing 5 shows how to select a specific individual gripper in a Gripper node configuration for a gripper device supporting multiple grippers (i.e. the multi-gripper capability). This is achieved using the list af individual grippers available in the multi-gripper provided by the `MultiGripperSupport` interface.

Listing 5: Configuring the multi-gripper capability

```
1  if (multiGripperSupport.isPresent()) {
2    List<SelectableGripper> selectableGrippers = multiGripperSupport.get().
         getSelectableGrippers();
3    if (!selectableGrippers.isEmpty()) {
4      gripConfigBuilder.setGripperSelection(selectableGrippers.get(0));
5    }
6  }
```

For simplicity, the first available gripper is always selected in the example in Listing 5.

The `setConfig(GripperNodeConfig)` method in the `GripperNode` interface will throw an `UnsupportedConfig` exception, if an attempt is made to apply a newly created configuration that uses a capability that is not supported by the gripper device (used by the Gripper program node). The exception will also be thrown, if an existing configuration is taken from one Gripper program node and reapplied to another Gripper program node, which uses a gripper device different from the gripper device used by the original node (i.e. when reapplying an existing configuration the two gripper devices must be identical).