MKEROS2_NODE v1.0

C++ ROS2 Node For MkE Point Cloud Publishing

Magik Eye Inc.

COLLABORATORS

	TITLE : MKEROS2_NODE v1.0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Magik Eye Inc.	Jan 2021	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
2101-00-EN-00	Jan 2021	Initial version	MEI

Contents

1 Introduction

This document describes v1.0 of `mkeros2` C++ v1.0 ROS2 package. This package contains ROS2 [1] node called `mkeros2_node` used for publishing 3D point cloud data provided by Magik Eye sensors. Currently, `mkeros2_node` connects to Magik Eye devices that provide 3D data using the TCP/IP protocol. The `mkeros2_node` codebase depends on the MkE API[mkeapi] C++ client implementation `libmkeclient`. The following table lists the officialy supported platforms for `mkeros2_node`:

Ubuntu Version	**ROS Distribution**
Ubuntu 18.04 64bit	*ROS Crystal*
Ubuntu 20.04 64bit	*ROS Foxy*

Note
This document assumes that the reader has a working knowledge of ROS2 and the ROS2 package compilation procedure. Documentation or explanation of any of these topics is out of the scope of this document.

Upon launch, the `mkeros2_node` binary registers a new node `mkeros2_node_`*NAME*, where *NAME* is the node's unique identifier that depends on the command line parameters passed to the node executable. It also publishes two services: `mkeros2_startpublish_`*NAME* and `mkeros2_stoppublish_`*NAME*. Once the `mkeros2_startpublish_`*NAME* service is invoked, the node connects to a Magik Eye sensor via TCP/IP network and starts publishing the sensor's 3D data stream under the `mkeros2_node_pcd_`*NAME* topic. The topic is unpublished and the connection to the sensor closed upon invocation of the `mkeros2_stoppublish_`*NAME* service.

2 Compilation

The `mkeros2_node` ROS2 node compilation is based on the CMake build system and Colcon [2]. Let's suppose that the ROS2 distribution has been installed into the `${ROS2_ROOT}` directory and the `mkeros2_node` codebase resides in the `${MKEROS2_ROOT}` ROS2 package in the `${ROS2_WS}` ROS2 workspace. The following BASH commands will compile the `mkeros2_node` into the `${ROS2_WS}/build/` directory:

```
$ mkdir "${ROS2_WS}/build"
$ cd "${ROS2_WS}/build"
$ source ${ROS2_ROOT}/setup.bash
$ colcon build --packages-select mkeros2 --base-paths ..
```

This will create the installation directory `install` in the `${ROS2_WS}/build` directory. To test the `mkeros2_node` compilation, execute the following commands:

```
$ source ${ROS2_WS}/build/install/setup.bash
$ ros2 run mkeros2 mkeros2_node --help
```

[1] ros2.org
[2] https://colcon.readthedocs.io/en/released/user/quick-start.html

2.1 Dependencies

The `mkeros2_node` codebase depends on the MkE API[mkeapi] C++ client implementation `libmkeclient`. In the case the `libmkeclient` library is not automatically found by the CMake system, a root path of the `libmkeclient` installation can be provided via the `MKECLI_ROOT` variable:

```
$ colcon build --packages-select mkeros2 \
    --cmake-args -DMKECLI_ROOT=/path/to/mkecli/installation \
    --base-paths ..
```

Alternatively, path to the source directory of `libmkeclient` can be provided:

```
$ colcon build --packages-select mkeros2 \
    --cmake-args -DMKECLI_URL=/path/to/mkecli/sources/ \
    --base-paths ..
```

3 Execution

Once installed, the `mkeros2_node` can be invoked through the `mkeros2` package. The `help` parameter lists and describes the available command line parameters:

```
$ source ${ROS2_WS}/build/install/setup.bash
$ ros2 run mkeros2 mkeros2_node --help
```

3.1 MkE Sensor Discovery

In order to connect to a Magik Eye sensor, the `mkeros2_node` executable needs to be provided with the IP address of unit ID of the sensor in question. Since all Magik Eye TCP/IP-enabled sensors implement network discovery using the SSDP protocol, `mkeros2_node` executable provides the `discover` command line option that will list all MagikEye sensors connected to the local TCP/IP network. In the following example, the `mkeros2_node` executable was able to discover two MagikEye sensors:

```
$ ros2 run mkeros2 mkeros2_node --discover
MagikEyeOne-0242be55:192.168.0.100
MagikEyeOne-0242ac2a:192.168.4.101
```

The list specifies the unit ID's and respective IP addresses of the discovered sensors.

The `discover` parameter can be also used in combination with the `device` parameter to check the availability of a particular sensor. The value of the `device` parameter can be an IP adress or a unit ID:

```
$ ros2 run mkeros2 mkeros2_node --discover --device MagikEyeOne-0242 ↩
   be55
$ echo $?
0
$ ros2 run mkeros2 mkeros2_node --discover --device 192.168.0.100
$ echo $?
0
```

```
$ ros2 run mkeros2 mkeros2_node --discover --device 192.168.0.102
$ echo $?
1
```

3.2 Launching

The `mkeros2_node` node can be launched either by providing the connection information through the command line parameters or through a launch file.

3.2.1 Launching Through the Command Line.

The `mkeros2_node` node is launched if the `launch` and `device` parameters are provided:

```
$ ros2 run mkeros2 mkeros2_node --launch \
                                --device MagikEyeOne-0242be55
[ INFO] [...]: Launching node: mkeros2_node_MagikEyeOne_0242be55
[ INFO] [...]: Starting service:  ↩
   mkeros2_startpublish_MagikEyeOne_0242be55
[ INFO] [...]: Starting service:  ↩
   mkeros2_stoppublish_MagikEyeOne_0242be55
```

The above will launch a node called `mkeros2_node_MagikEyeOne-0242be55` and start two services called `mkeros2_startpublish_MagikEyeOne_0242be55` and `mkeros2_stoppublish_MagikEyeOne_0242be55` respectively. Again, the `device` parameter can also contain the sensor's IP address. The sensor specific part of the node and services names can be overriden using the `alias` parameter:

```
$ ros2 run mkeros2 mkeros2_node --launch \
                                --device MagikEyeOne-0242be55
                                --alias s1
[ INFO] [...]: Launching node: mkeros2_node_s1
[ INFO] [...]: Starting service: mkeros2_startpublish_s1
[ INFO] [...]: Starting service: mkeros2_stoppublish_s1
```

Note

The node will *not* connect to the sensor upon launch, nor will it check the availability of the sensor. The connection will only be attempted upon invocation of the `mkeros2_startpublish_`*NAME* service. For an immediate check of the sensor's availability, use the `discover` parameter.

3.2.2 Launching Through a Launch File.

The `launch` file present in `${MKEROS2_ROOT}/launch/mkeros2_cpp.launch.py` can be used to launch the `mkeros2_node` with default parameters described `${MKEROS2_ROOT}/config/mkeros2_config.yaml`. The `device` parameter is a mandatory of the `launch` file. The `alias` parameter is optional. Note that if the `device` parameter or the `alias` parameter is updated, the `mkeros2` package needs to be reinstalled.

For example, the launch file `mkeros2_config.yaml` can look as follows:

```
# Default configurations
MKEROS_NODE:
ros__parameters:
    device : "MagikEyeOne-34cff660"
    alias : "sensor1"
```

Launching the node using a launch file with the above parameters can be done using the `ros2 launch` command:

```
$ ros2 launch mkeros2 mkeros2.launch.py
[INFO] [launch]: All log files can be found below /home/magikeye/.ros/ ↩
   log/2021-02-10-15-51-54-259398-magikeye-Yoga-Slim-7-14IIL05-35797
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [mkeros2_node-1]: process started with pid [35799]
[mkeros2_node-1] [INFO] [1612952514.348838967] [rclcpp]: Launching  ↩
   node: mkeros2_node_sensor1
[mkeros2_node-1] [INFO] [1612952514.350210571] [rclcpp]: Starting  ↩
   service: mkeros2_startpublish_sensor1
[mkeros2_node-1] [INFO] [1612952514.350940490] [rclcpp]: Starting  ↩
   service: mkeros2_stoppublish_sensor1
```

4 Services

Upon execution, the `mkeros2_node` binary publishes two services: `mkeros2_startpublish_`*NAME* and `mkeros2_stoppublish_`*NAME*.

4.1 Start Publishing

Once the `mkeros2_startpublish_`*NAME* service is invoked, the node connects to a Magik Eye sensor via TCP/IP network and starts publishing the sensor's 3D data stream under the `mkeros2_node_pcd_`*NAME* topic. If the sensor has been specified via its IP address, the node will try to connect to the sensor directly. In the case the sensor has been specified using its unit ID, the discovery procedure to recover its IP address will be performed. Once the connection is established, the `mkeros2_node_pcd_`*NAME* topic is published.

The `mkeros2_node` binary provides a convenience parameter `start` to call the start service. The device can be specified via the `device` or `alias` options:

```
$ ros2 run mkeros2 mkeros2_node --start --alias s1
[INFO] [1612953661.365947423] [rclcpp]: Calling service:  ↩
   mkeros2_startpublish_sensor1
[INFO] [1612953662.439238763] [rclcpp]: Service called successfully:  ↩
   mkeros2_startpublish_sensor1
```

4.2 Stop Publishing

The `mkeros2_node_pcd_`*NAME* topic is unpublished and the connection to the sensor closed upon invocation of the `mkeros2_stoppublish_`*NAME* service.

The `mkeros2_node` binary provides a convenience parameter `stop` to call the stop service. The device can be specified via the `device` or `alias` options:

```
$ ros2 run mkeros2 mkeros2_node --stop --alias s1
[INFO] [1612953665.295884780] [rclcpp]: Calling service:  ↩
   mkeros2_stoppublish_sensor1
[INFO] [1612953665.409713437] [rclcpp]: Service called successfully:  ↩
   mkeros2_stoppublish_sensor1
```

5 Accessing The Point Cloud Data

While publishing, the sensor data will be available on the topic called `mkeros2_node_pcd_`*NAME*. The message format of the data published on this topic is `sensormsgs::PointCloud2`.

6 Bibliography

- [] *MagikEye API v1.0*, 2020, Magik Eye Inc.