MagikEye

INVERTIBLE LIGHT™
TECHNOLOGY

MKEROS1_NODE v1.0

C++ ROS1 Node For MkE Point Cloud Publishing

Magik Eye Inc.

COLLABORATORS

	TITLE : MKEROS1_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 MKEROS1_NODE (also spelled `mkeros1_node`) v1.0, a ROS1 [1] node present in `mkeros` package for publishing 3D point cloud data provided by Magik Eye sensors. Currently, `mkeros1_node` connects to Magik Eye devices that provide 3D data using the TCP/IP protocol. The `mkeros1_node` codebase depends on the MkE API[mkeapi] C++ client implementation `libmkeclient`. The following table lists the officialy supported platforms for `mkeros1_node`:

Ubuntu Version	**ROS Distribution**
Ubuntu 18.04 64bit	*ROS Melodic*
Ubuntu 20.04 64bit	*ROS Noetic*

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

Upon launch, the `mkeros1_node` binary registers a new node `mkeros1_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: `mkeros1_startpublish_`*NAME* and `mkeros1_stoppublish_`*NAME*. Once the `mkeros1_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 `mkeros1_node_pcd_`*NAME* topic. The topic is unpublished and the connection to the sensor closed upon invocation of the `mkeros1_stoppublish_`*NAME* service.

2 Compilation

The `mkeros1_node` ROS node compilation is based on the CMake build system and Catkin [2]. Let's suppose that the ROS distribution has been installed into the `${ROS_ROOT}` directory and the `mkeros1_node` codebase resides in the `${MKEROS_ROOT}` ROS package of `${ROS1_WS}` ROS workspace. The following BASH commands will compile the `mkeros1_node` into the `${ROS1_WS}/build/` directory:

```
$ mkdir "${ROS1_WS}/build"
$ cd "${ROS1_WS}/build"
$ source ${ROS_ROOT}/setup.bash
$ catkin_make --pkg mkeros --source ..
```

2.1 Dependencies

The `mkeros1_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:

[1] ros.org
[2] http://docs.ros.org/en/api/catkin/html

```
$ catkin_make --pkg mkeros --source .. --cmake-args \
              -DMKECLI_ROOT=/path/to/mkecli/installation
```

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

```
$ catkin_make --pkg mkeros --source .. --cmake-args \
              -DMKECLI_URL=/path/to/mkecli/sources/
```

2.2 Installation

The compiled `mkeros1_node` resides in the `mkeros` ROS package. In order to install the package, run the following command:

```
$ catkin_make --source .. install --cmake-args
```

This will create the installation directory `install` in the `${ROS1_WS}/build` directory. Alternatively, the installation path can be changed using the `CMAKE_INSTALL_PREFIX` CMake option. Let's assume that the `${ROS1_WS_INST}` variable points to the desired installation directory:

```
$ catkin_make --source .. install --cmake-args \
              -DCMAKE_INSTALL_PREFIX=${ROS1_WS_INST}
```

To test the `mkeros1_node` compilation, execute the following commads:

```
$ source ${ROS1_WS_INST}/setup.bash
$ rosrun mkeros mkeros1_node --help
```

3 Execution

Once installed, the `mkeros1_node` can be invoked through the `mkeros` package. The `help` parameter lists and describes the available command line parameters:

```
$ source ${ROS1_WS_INST}/setup.bash
$ rosrun mkeros mkeros1_node --help
```

3.1 MkE Sensor Discovery

In order to connect to a Magik Eye sensor, the `mkeros1_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, `mkeros1_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 `mkeros1_node` executable was able to discover two MagikEye sensors:

```
$ rosrun mkeros mkeros1_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:

```
$ rosrun mkeros mkeros1_node --discover --device MagikEyeOne-0242be55
$ echo $?
0
$ rosrun mkeros mkeros1_node --discover --device 192.168.0.100
$ echo $?
0
$ rosrun mkeros mkeros1_node --discover --device 192.168.0.102
$ echo $?
1
```

3.2 Launching

The `mkeros1_node` node can be launched either by providing the connection information through the command line parameters or through a launch file.

Note
The `roscore` (rosmaster) process must already be running in order for the `roslaunch` or `rosrun` commands to work.

3.2.1 Launching Through the Command Line

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

```
$ rosrun mkeros mkeros1_node --launch \
                             --device MagikEyeOne-0242be55
[ INFO] [...]: Launching node: mkeros1_node_MagikEyeOne_0242be55
[ INFO] [...]: Starting service:  ↩
   mkeros1_startpublish_MagikEyeOne_0242be55
[ INFO] [...]: Starting service:  ↩
   mkeros1_stoppublish_MagikEyeOne_0242be55
```

The above will launch a node called `mkeros1_node_MagikEyeOne-0242be55` and start two services called `mkeros1_startpublish_MagikEyeOne_0242be55` and `mkeros1_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:

```
$ rosrun mkeros mkeros1_node --launch \
                             --device MagikEyeOne-0242be55
                             --alias s1
[ INFO] [...]: Launching node: mkeros1_node_s1
[ INFO] [...]: Starting service: mkeros1_startpublish_s1
[ INFO] [...]: Starting service: mkeros1_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 `mkeros1_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 `${MKEROS_ROOT}/launch/mkeros1_cpp.launch` can be used to launch the `mkeros_node` with default parameters described `${MKEROS_ROOT}/config/mkeros1_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 `mkeros` package needs to be reinstalled.

For example, the launch file `mkeros1_config.yaml` can look as follows:

```
# Default configurations
device : "192.168.0.117"
# alias : "s1"
```

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

```
$ roslaunch mkeros mkeros1_cpp.launch
```

Note
If roslaunch is used to launch the `mkeros1_node` using the above method, then `rosrun` should not be invoked to launch start and stop services or for other CLI parameters.

4 Services

Upon execution, the `mkeros1_node` binary publishes two services: `mkeros1_startpublish_`*NAME* and `mkeros1_stoppublish_`*NAME*.

Note
If `roslaunch` is used to launch the `mkeros1_node`, then the services `mkeros1_startpublish_`**NAME** and `mkeros1_stoppublish_`**NAME** should be called using `rosservice call` command of ROS API.

4.1 Start Publishing

Once the `mkeros1_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 `mkeros1_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 `mkeros1_node_pcd_`*NAME* topic is published.

The `mkeros1_node` binary provides a convenience parameter `start` to call the start service. The device can be specified via the `device` or `alias` options:

```
$ rosrun mkeros mkeros1_node --start --alias s1
[ INFO] [...]: Calling service: mkeros1_startpublish_s1
[ INFO] [...]: Service called successfully: mkeros1_startpublish_s1
```

4.2 Stop Publishing

The `mkeros1_node_pcd_`*NAME* topic is unpublished and the connection to the sensor closed upon invocation of the `mkeros1_stoppublish_`*NAME* service.

The `mkeros1_node` binary provides a convenience parameter `stop` to call the stop service. The device can be specified via the `device` or `alias` options:

```
$ rosrun mkeros mkeros1_node --stop --alias s1
[ INFO] [...]: Calling service: mkeros1_stoppublish_s1
[ INFO] [...]: Service called successfully: mkeros1_stoppublish_s1
```

5 Accessing The Point Cloud Data

While publishing, the sensor data will be available on the topic called `mkeros1_node_pcd_`*NAME*. The message format of the data published on this topic is `sensormsgs::PointCloud2`.

6 Bibliography

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