

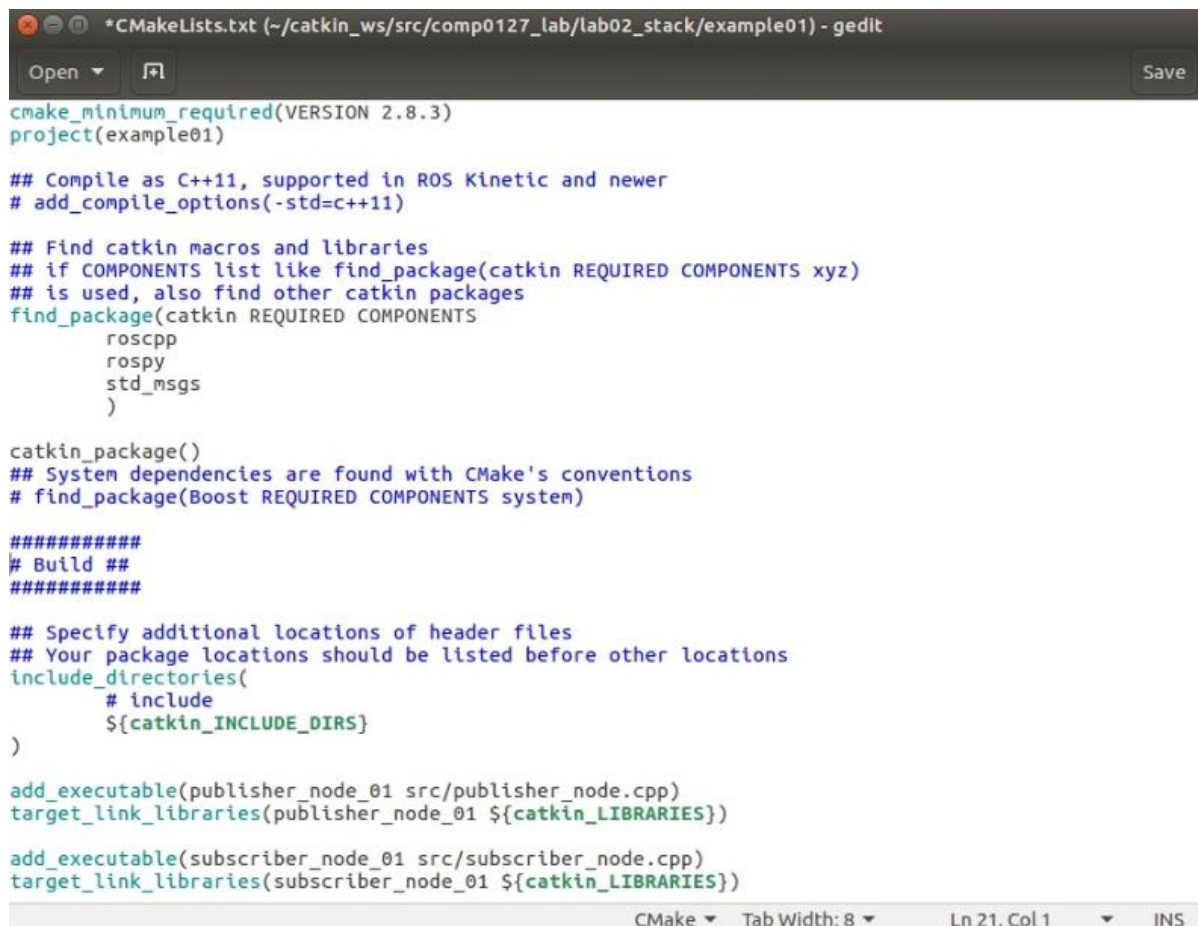
Robotic Systems Engineering Lab 02

This document contains examples and exercises on writing code for simple ROS communication. After we go through three examples, you will need to complete four tasks on your own.

Navigate into the 'lab02' folder inside your workspace

1 CMakeLists.txt

Before jumping into the examples let's go through a standard CMakeLists.txt file to better understand how ROS works.

A screenshot of a text editor window titled '*CMakeLists.txt (~/.catkin_ws/src/comp0127_lab/lab02_stack/example01) - gedit'. The editor shows a CMakeLists.txt file with the following content:

```
cmake_minimum_required(VERSION 2.8.3)
project(example01)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
    roscpp
    rospy
    std_msgs
)

catkin_package()
## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)

#####
# Build ##
#####

## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(
    # include
    ${catkin_INCLUDE_DIRS}
)

add_executable(publisher_node_01 src/publisher_node.cpp)
target_link_libraries(publisher_node_01 ${catkin_LIBRARIES})

add_executable(subscriber_node_01 src/subscriber_node.cpp)
target_link_libraries(subscriber_node_01 ${catkin_LIBRARIES})
```

The editor's status bar at the bottom shows 'CMake', 'Tab Width: 8', 'Ln 21, Col 1', and 'INS'.

- The first line sets the minimum version of CMake for this project. This version is somewhat arbitrary in this example but providing a version number allows for future support for your build environment. The second line sets the project and package name.
- The 'find_package' command finds and loads settings from an external project. If the package is found, package-specific information is provided through variables and Imported Targets documented by the package itself. We need 'roscpp' and 'rospy', which are pure C++ and Python client libraries for ROS respectively. These libraries enable programmers to quickly interface with ROS Topics, Services and Parameters. We also need 'std_msgs' which contains standard ROS messages including common message types and basic message constructs.
- As we specified, the 'find_package' command is a common CMake command and is needed to load the catkin macros and specify dependencies to other ROS packages. The 'catkin_package' command is one of these catkin macros. It is responsible for the ROS-specific configuration of the package.
- The 'include_directories' command is self-explanatory. It adds the given directories to those the compiler uses to search for include files.
- The 'add_executable' command defines our binary with all linked source files. It adds an executable target to be built from the source files listed in the command invocation.
- And finally, the 'target_link_libraries' command specifies libraries or flags to use when linking a given target. In a few words, we tell CMake to link those libraries to our executable.

2 Examples

On GitHub you can find packages for examples 01 and 04. This is because Example 02 is just a continuation of Example 01 and the package for Example 03 you need to create yourselves. Example 04 has no instructions. You will use it later in Task 3.3.

2.1 Example 01

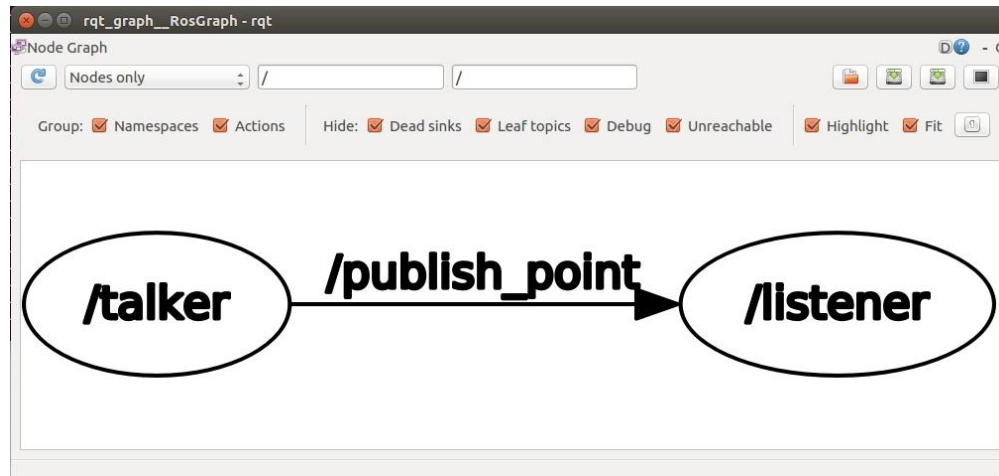
Open the two source files (choose either the .py or .cpp ones) and try to find out what information the two nodes will exchange. Then, run the example using three separate terminals. The first one will simply run the ROS master. To run the nodes, source the environment in both terminals and then type:

```
roslaunch package_name executable_name
```

Those names are found in the CMakeLists.txt file. Make sure that your guess of what was going to happen was correct, and if not, go back to the code and find out what you got wrong.

2.2 Example 02

Continuing Example 01, on a fourth terminal type the command 'rqt_graph' to see how the communication between the two nodes is taking place using the GUI plugin for visualizing the ROS computation graph.



2.3 Example 03

In this example we will create a package with two nodes inside a catkin workspace. We will use C++ code from the ROS tutorials for our source files instead of writing our own in C++ or python, since the purpose of this example is to demonstrate how to operate within ROS, and not to test your programming skills.

Copy and paste the code below into 'Text Editor' and save the file as 'CMakeLists.txt'

https://raw.githubusercontent.com/ros/catkin_tutorials/master/create_package_pubsub/catkin_ws/src/beginner_tutorials/CMakeLists.txt

It is important that you remove this part below from the file

```
## Declare ROS messages and services  
add_message_files(FILES Num.msg)  
add_service_files(FILES AddTwoInts.srv)
```

Copy and paste the code below into 'Text Editor' and save the file as 'talker.cpp'

https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/talker/talker.cpp

Copy and paste the code below into 'Text Editor' and save the file as 'listener.cpp'

https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp

Now we have the source .cpp files and the CMakeLists file of the package we want to create. First step is to create a catkin workspace called 'catkin_ws'. Skip this step if you already created a workspace during Lab 01.

```
mkdir -p ~/catkin_ws/src  
cd ~/catkin_ws/src  
catkin_init_workspace  
cd ~/catkin_ws/  
catkin_make
```

Next, we must create our package

```
cd ~/catkin_ws/src  
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

Navigate to `/catkin_ws/src/beginner_tutorials` and replace `CMakeLists.txt` with the one you created at the start of this example, and then put the two `.cpp` files into `/catkin_ws/src/beginner_tutorials/src` folder.

Now we must update the workspace

```
cd ~/catkin_ws  
catkin_make
```

Now that our workspace and package are ready, it's time to run our nodes.

In a terminal initiate the master

```
roscore
```

In another terminal navigate to the workspace, source it, and run the first node

```
cd ~/catkin_ws/  
source devel/setup.bash  
roslaunch beginner_tutorials talker
```

In a third terminal navigate to the workspace, source it, and run the second node

```
cd ~/catkin_ws/  
source devel/setup.bash  
roslaunch beginner_tutorials listener
```

The result should be a successful communication between the 'talker' and the 'listener'.

3 Tasks

Using what you have learnt, complete tasks 01-04 (these tasks are **NOT** assessed):

3.1 Task 01: ROS publisher/subscriber

1. Create a publisher node that publishes a message which contains a random number between '0' and the variable 'count'.
2. Create a subscriber that receives this incoming message and prints it out.

3.2 Task 02: ROS publisher/subscriber with array

1. Create a publisher node that publishes an array. For each publishing loop, keep appending the array with the variable 'count'. For example, if count = 4, then message = [1 2 3 4].
2. create a subscriber that receives the message, computes the sum of the series $S = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$, and prints it out.

3.3 Task 03: ROS publisher/subscriber with arguments in a callback function

1. Create a publisher node that publishes a point on a unit circle on the XZ-plane.
2. Create a subscriber that receives the message and adds an offset similar to Example 04 from GitHub.
3. Create a launch file to run both nodes at the same time.

3.4 Task 04: ROS launch and youbot control

1. Go through the structure of the package 'lab01/youbot_traj_example'.
2. Create a new package 'task04' in 'lab02' with the same structure as the youbot example.
3. Create a publisher node that publishes commands for each joint of the youbot arm. Each command is a function of time as following. You can take a look at ROS::Time documentations (cpp: <http://wiki.ros.org/roscpp/Overview/Time> , py: <http://wiki.ros.org/rospy/Overview/Time>)

$$\begin{aligned}\theta_1(t) &= \frac{200\pi}{180} \sin\left(\frac{2\pi t}{10}\right) \\ \theta_2(t) &= \frac{50\pi}{180} \sin\left(\frac{2\pi t}{12}\right) \\ \theta_3(t) &= -\frac{80\pi}{180} \sin\left(\frac{2\pi t}{15}\right) \\ \theta_4(t) &= \frac{60\pi}{180} \sin\left(\frac{2\pi t}{11}\right)\end{aligned}$$

4. Create a launch file that runs the simulation and the robot.