# Lab 03: ROS Tutorial 1

By Sean Lu, last modified on 09/25 2018.

The objective of this tutorial is to introduce to the ROS concept, nodes, publishers and subscribers, and also demonstrate some commonly used command line tools. Spatial transformation between each component is an important thing in your robot, luckily, tf helps us to do the job. We have assumed you have the basic coding ability to both C++ and Python. You should also understand homogeneous transformation matrix, if not, Prob. Yang’s [Robotics](http://ocw.nctu.edu.tw/course_detail-v.php?bgid=8&gid=0&nid=554&v5=UiKi5-Arce4) would be a good material for you. You should have installed ROS in your laptop, for those who use native Ubuntu please refer to the Hardware and Software Setup part.

## Hardware and Software Setup

We have already installed ROS in the VirtualBox image we provided, for those who use native Ubuntu, please follow the step [here](http://wiki.ros.org/kinetic/Installation/Ubuntu).

Clone the repo and make the workspace

**laptop $ cd ~/**

**laptop $ git clone** [**https://github.com/Sensing-Intelligent-System/sis\_lab\_all\_2020**](https://github.com/Sensing-Intelligent-System/sis_lab_all_2020)

**# You will be asked to type your Github username and password**

**laptop $ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws**

**laptop $ catkin\_make # This may take awhile**

**laptop $ source devel/setup.bash # Make ROS knows where your packages are, do this every time you open a new terminal**

Download the bags we prepared here

[temperature\_pressure\_bag](https://drive.google.com/file/d/1L1_FFHAlKbnt2ZbPPRoSRLqFl98A0-Vu/view?usp=sharing)

[Download in your VirtualBox]

Put the bag in **sis\_lab\_all\_2020/03-ROS\_tutorial\_1/bag**

**laptop $ cd ~/Downloads**

**laptop $ mkdir ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/bag && mv \*.bag ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/bag/**

[**Lab 03: ROS Tutorial 1**](#_74gram21yqax) **1**

[Hardware and Software Setup](#_chc57ow4x0s1) 1

[Overview](#_x5puhc1p2jjo) 2

[Topics and Activities](#_g0fw0bgwxhdt) 2

[Topic/Activity 1 Command Line Tools](#_aadm982kyolo) 2

[Topic/Activity 2 Publisher/ Subscriber](#_sf5mhl8ztpdm) 6

[Topic/Activity 3 TF](#_v4uchiwk8car) 9

[Assignment Tasks](#_kack8n2wqwv3) 12

[Task 1 Publisher/ Subscriber](#_f467azdvd3oa) 12

[Task 2 TF](#_hn4u1zkqq0p0) 13

[Reference](#_uwwglharlfb) 15

## Overview

Estimated Time to Finish: 2 hours

After completing this tutorial you should

* understand the basic ROS concept: nodes, publishers and subscribers
* understand how to use tf APIs
* be able to write a simple node to publish/ subscribe the specific topic

## Topics and Activities

### Topic/Activity 1 Command Line Tools

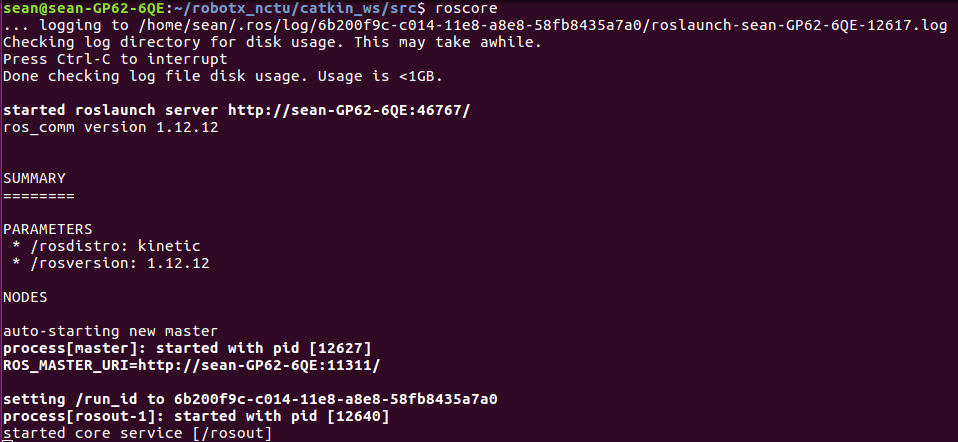
ROS provides many convenient command line tools, let’s get familiar with them.

**Topic 1.1 Initialize your ROS master**

Open a new terminal, type

**laptop $ roscore**

You will see something like the following



Open another terminal, type

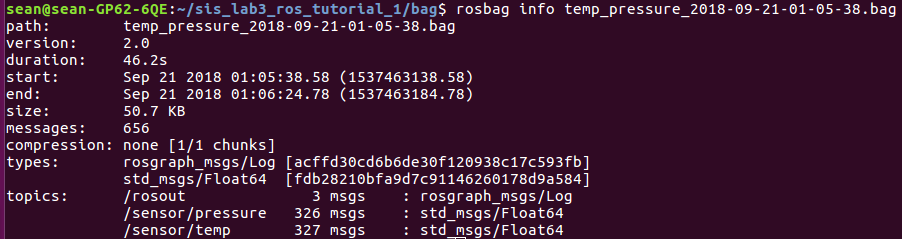
**laptop $ rostopic list**

You will see there are two topics, /rosout and /rosout\_agg

**Topic 1.2 Play the bag**

Let’s see what surprise inside the bag,

**laptop $ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/bag**

**laptop $ rosbag info temp\_pressure\_2018-09-21-01-05-38.bag**

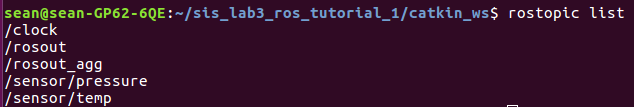
As you can see, the information include the duration, start and end time, bag size, total number of messages, the message type and topic name.

**laptop $ rosbag play temp\_pressure\_2018-09-21-01-05-38.bag -l # -l means play in loops**

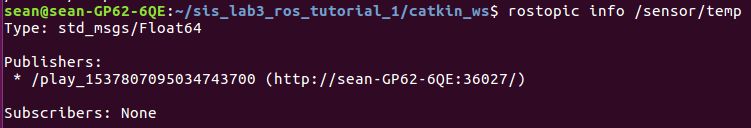
Open another terminal, type

**laptop $ rostopic list**

You will see two new topics as expected



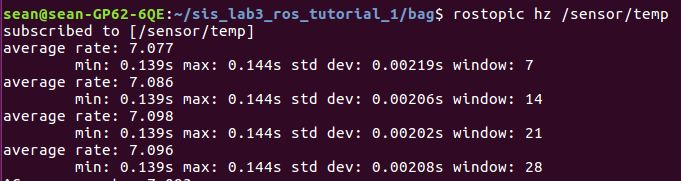
Info helps us see who publishes and subscribes the topic

**laptop $ rostopic info /sensor/temp**

/sensor/temp is published by the node play, and no subscriber yet.

If we are interested in the rate of the topic:

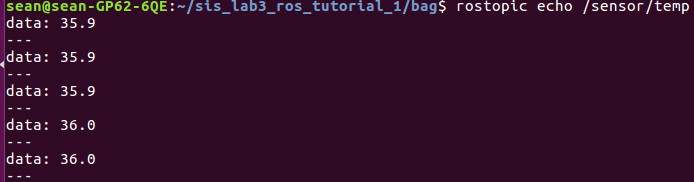
**laptop $ rostopic hz /sensor/temp**



The terminal keeps printing data until you press Ctrl+C.

Echo prints the data of the topic:

**laptop $ rostopic echo /sensor/temp**

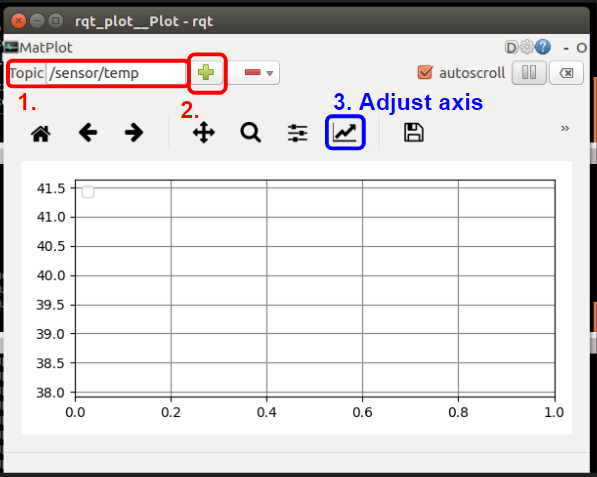


Press Ctrl+C to stop.

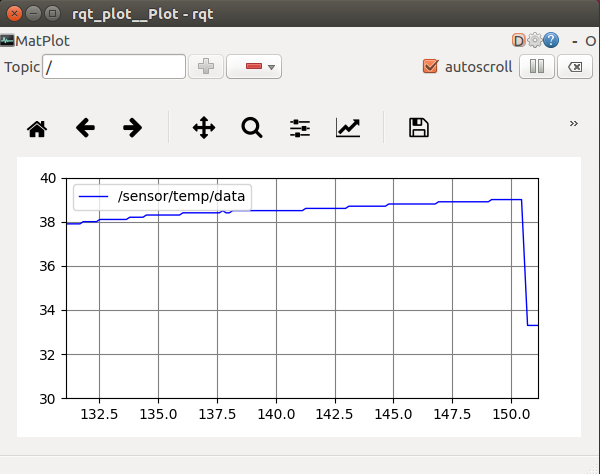
We can also visualize the temperature-time curve,

**laptop $ rqt\_plot**

A window with name rqt\_plot will show with empty canvas, in Topic field, add /sensor/temp



Then you will see the curve as following,



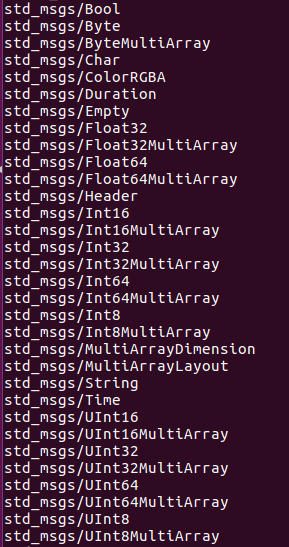
you can adjust the axes range in the blue box in above picture. Press ‘X’ to turn off the interface.

**Topic 1.3 What messages and services I have?**

There are many default messages and services in ROS, let’s take a look.

**laptop $ rosmsg list**

There will be several lines, for prefix ‘s’, you will see a family std\_msgs:



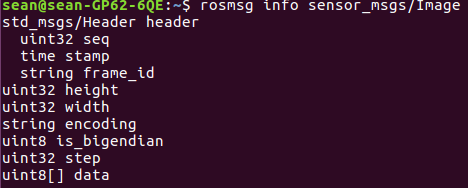
These are the most basically message types in ROS, you can find them in /opt/ros/kinetic/share/std\_msgs/msg.

Other usually used message family include geometry\_msgs, sensor\_msgs and visualization\_msgs.

If you want to see what information inside the message, for example, sensor\_msgs/Image, you can type

**laptop $ rosmsg info sensor\_msgs/Image**

The result is as following



Just like rosmsg,

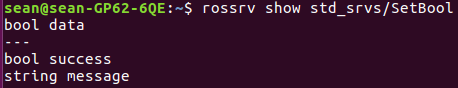
**laptop $ rossrv list**

You will see many available services in your system, a screenshot of the result is as below



So what SetBool do?

**laptop** **$ rossrv show std\_srvs/SetBool**



You call the service with ‘data’, the service will return a response to you whether if it is ‘success’ and a string ‘message’.

**Checkpoint 1: Show TA your rqt\_plot**

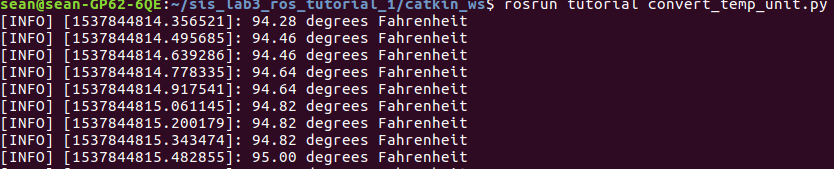
### Topic/Activity 2 Publisher/ Subscriber

Now we are going to run a ROS node, open a terminal

**laptop $ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws && source devel/setup.bash**

**laptop $ rosrun tutorial convert\_temp\_unit.py**

The result should be as following



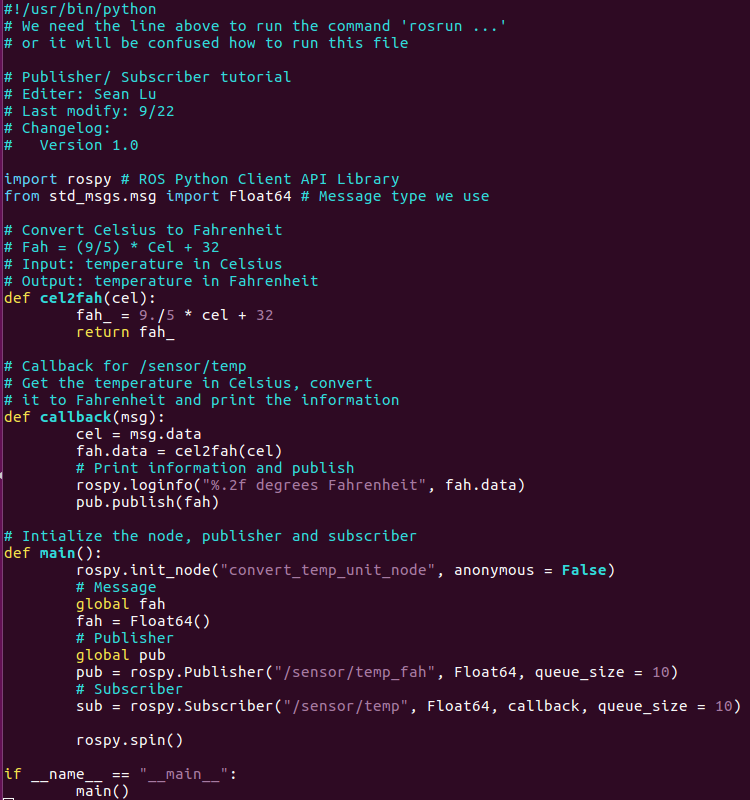
The number after INFO is the time now, you can get similar string in Python if you print time.time(). The string after colon is our information, what degrees Fahrenheit were at that time.

Let’s see what contents in the file, open a new terminal

**laptop $ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws && source devel/setup.bash**

**laptop $ roscd tutorial/src**

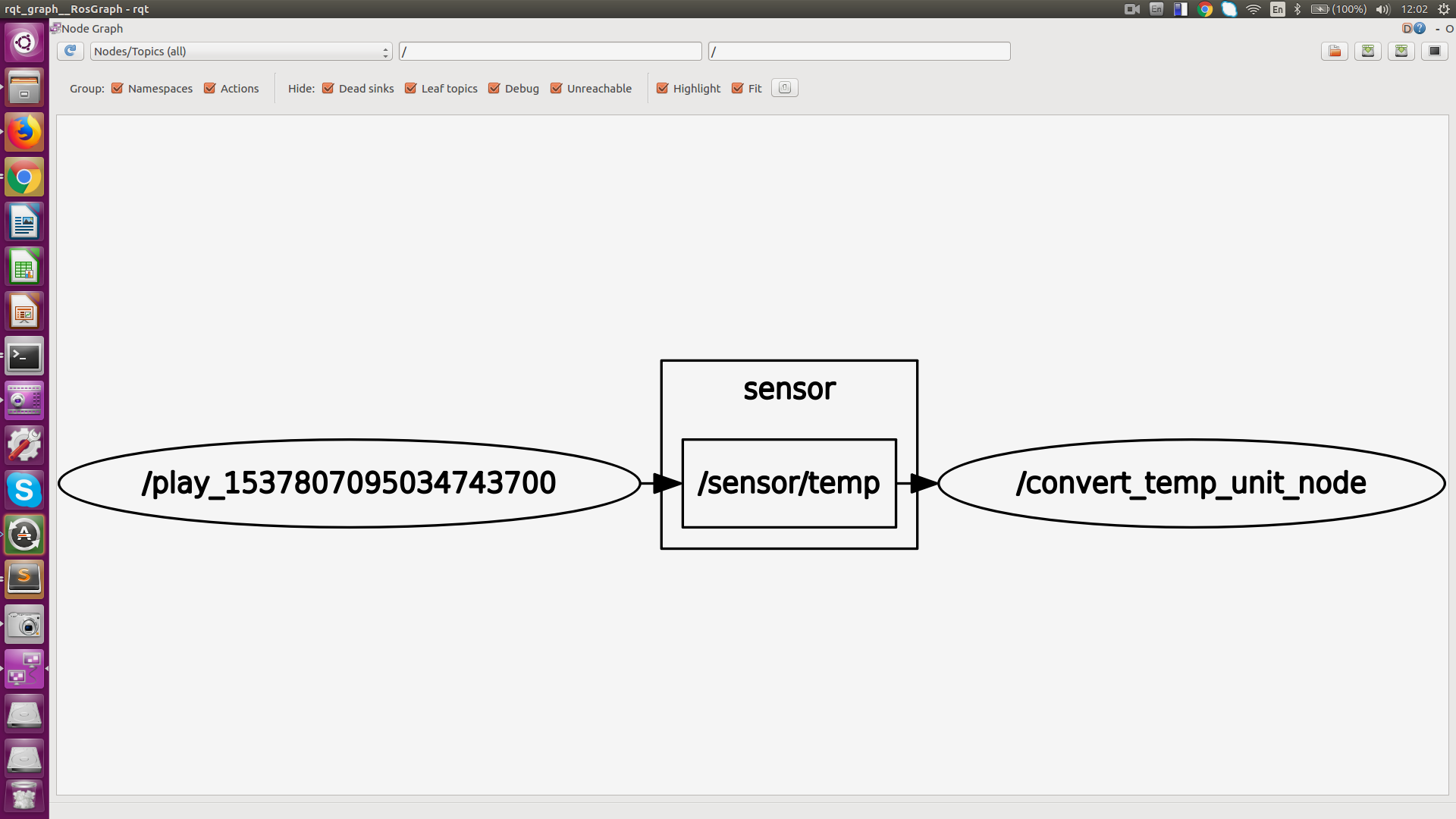
**laptop $ vim convert\_temp\_unit.py (or $ code convert\_temp\_unit.py)**



In this node, we subscribe to /sensor/temp, convert its unit to Fahrenheit and then publish as topic /sensor/temp\_fah, we also print the information as above.

What is our nodes/ topics flow now? Exit vim and type

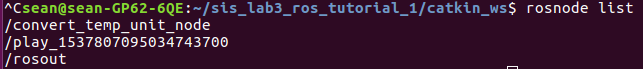
**laptop $ rqt\_graph**



/play publishes /sensor/temp and /convert\_temp\_unit\_node subscribes to it, the topics with no subscribers will not shown in rqt\_graph.

With command

**laptop $ rosnode list**



we can list all active nodes just like “rostopic list”

How to create an user-defined message if we want to publish temperature and pressure simultaneously? Follow the step [here](http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv) to create the new message type, my\_message, with two float64: temp and pressure.

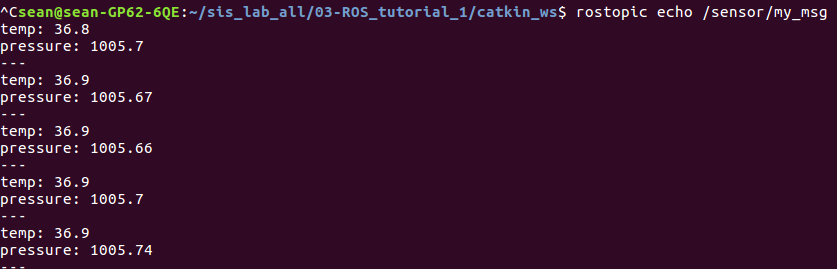
Next,

**laptop $ rosrun tutorial test\_my\_message.py**

In another terminal,

**laptop $ rostopic echo /sensor/my\_msg**

You will see



If you want to know what this node has done, see **sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws/src/tutorial/src/test\_my\_message.py**

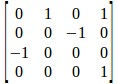
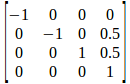
We subscribe to both /sensor/temp and /sensor/pressure, get the data and publish our new type message.

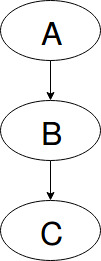
**Checkpoint 2: Show TA the picture above**

### Topic/Activity 3 TF

Notation:

**Topic 3.1 Publish a transform**

Given  and , how to build the transform tree like the figure below with the help of TF?



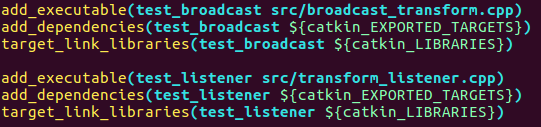
In this tutorial, we use C++ to construct our node, while don’t forget that ‘rosrun’ must be feed with an executable file, so you have to make your C++ source code be executable. In ROS Kinetic, we use **CMake** and **catkin** to lighten our burden. Take a look at **sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws/src/tutorial/CMakeLists.txt**

**laptop** **$ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws && source devel/setup.sh**

**laptop** **$ roscd tutorial**

**laptop** **$ vim CMakeLists.txt**

At Line 155, you will see



For more detail what the function of each tag, please refer to [here](http://wiki.ros.org/catkin/CMakeLists.txt). Don’t edit the other part of CMakeLists.txt manually unless you know what you are doing. Every time you create a new C++ file, add these three tags to your CMakeLists.txt; or if you have modified your C++ file, back to **~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws**, and then

**laptop $ catkin\_make # You can skip this step until Assignment 2**

Run the node, stop all your terminal unless roscore and in one terminal

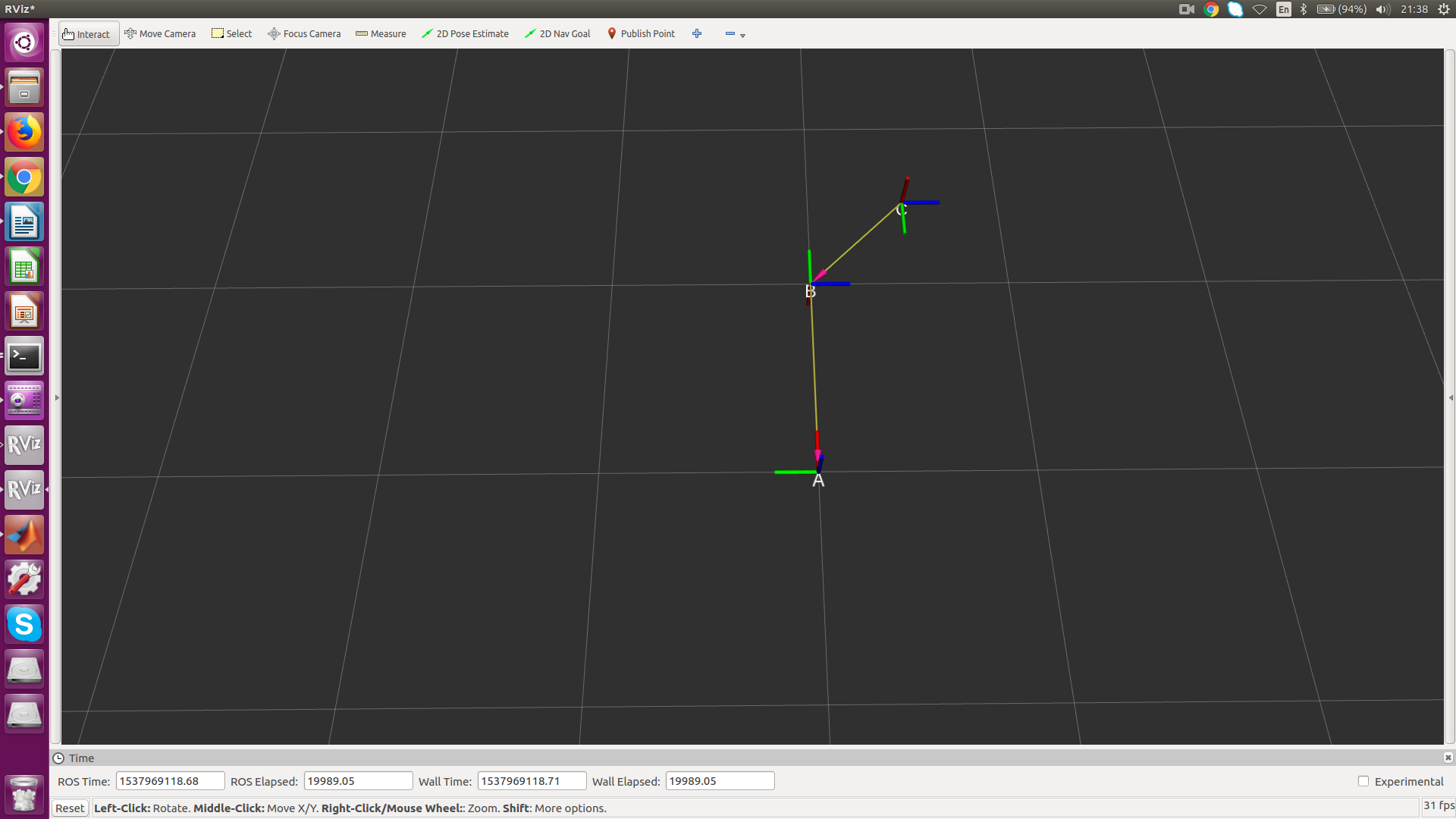
**laptop** **$ rosrun tutorial test\_broadcast**

And in another terminal,

**laptop** **$ cd ~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/rviz**

**laptop** **$ rviz -d tf.rviz**

You will see something like below

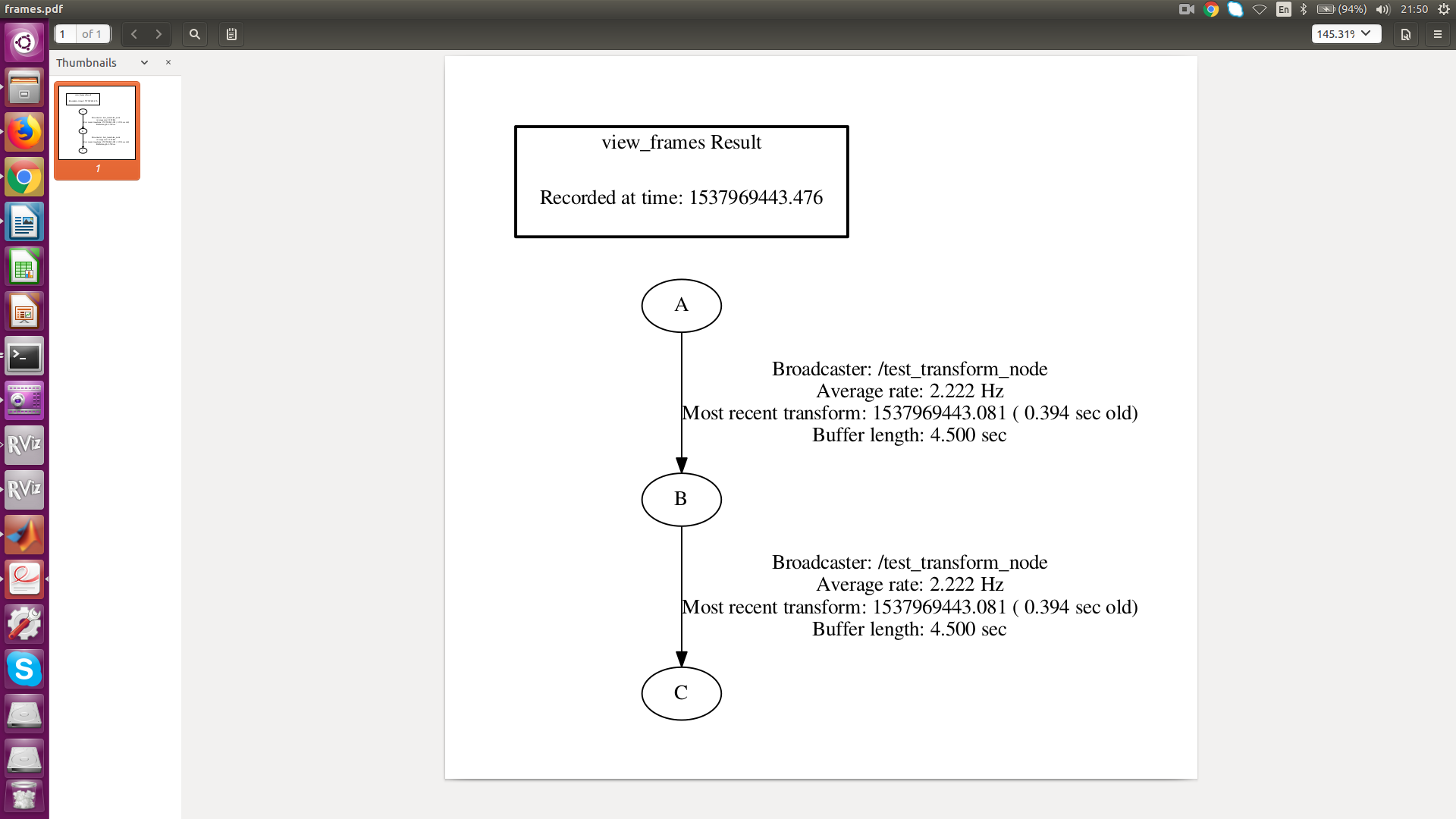


Let’s see the transform tree, in another terminal

**laptop** **$ cd ~/ && rosrun tf view\_frames**

It will produce an PDF file named frames.pdf, use evince to take a look

**laptop** **$ evince frames.pdf**

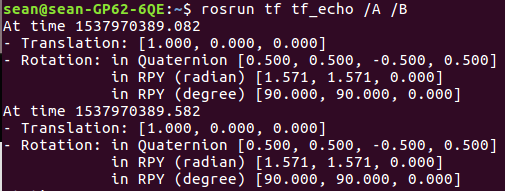


Just like the structure we expected.

Print the transform between frame A and frame B by

**laptop** **$ rosrun tf tf\_echo /A /B**

Expected result

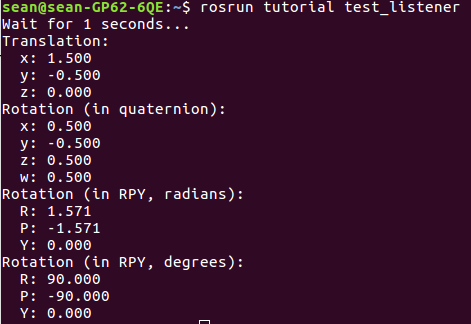


**Topic 3.2 Listen to the transform**

Sometimes we have to know the transformation between two frames, for pick and place scenario, we already know the transformation between the camera and the object, but what we want to know is the one from the base of the robot arm to the object, so that we could pick it with the tool on the arm. TF provides listener API which complete this job, let’s study it. In one terminal,

**laptop** **$ rosrun tutorial test\_listener**

This node will print the transformation between frame A and frame C in one shot rather than tf\_echo. The expected result will be looked like the following



You may take a look at **sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws/src/tutorial/src/transform\_listener.cpp**

**Checkpoint 3: Show TA your rviz, and echo the transform from frame A to frame C**

## Assignment Tasks

Students should do this by themselves

### Task 1 Publisher/ Subscriber

In this assignment, we ask you to do the similar task with Topic 2, convert temperature from Celsius to Fahrenheit, while this time we want to know the pressure in mmHg rather than in hPa. Edit **~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws/src/tutorial/src/assignment1.py**

such that you have the result as following

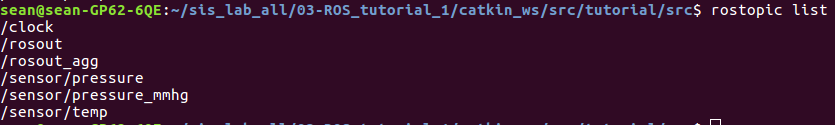
**Expected Result:**

**Fig. 1.1 $ rosrun tutorial assignment1.py**

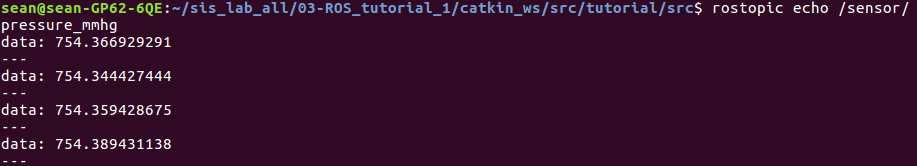
**[Hint: Don’t forget to make your .py file executable by**

**‘$ chmod +x assignment1.py’]**

**Fig. 1.2 $ rostopic list**

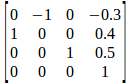
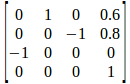
****

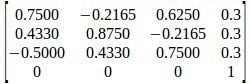
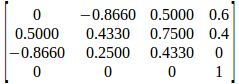
**Fig. 1.3 $ rostopic echo /sensor/pressure\_mmhg**

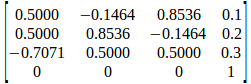
****

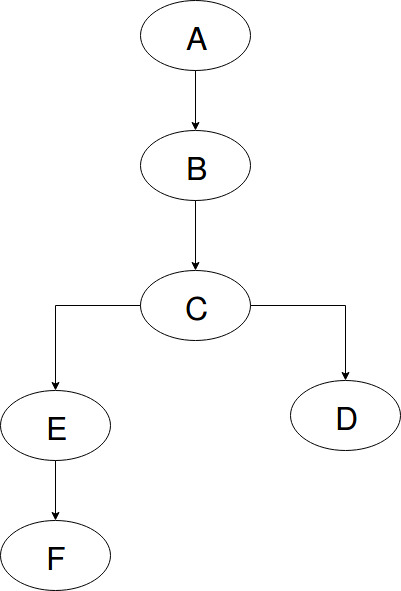
### Task 2 TF

In this assignment, we ask you to do the similar task with Topic 3, while this time with more complicated transform tree as following, where

, ,

, 

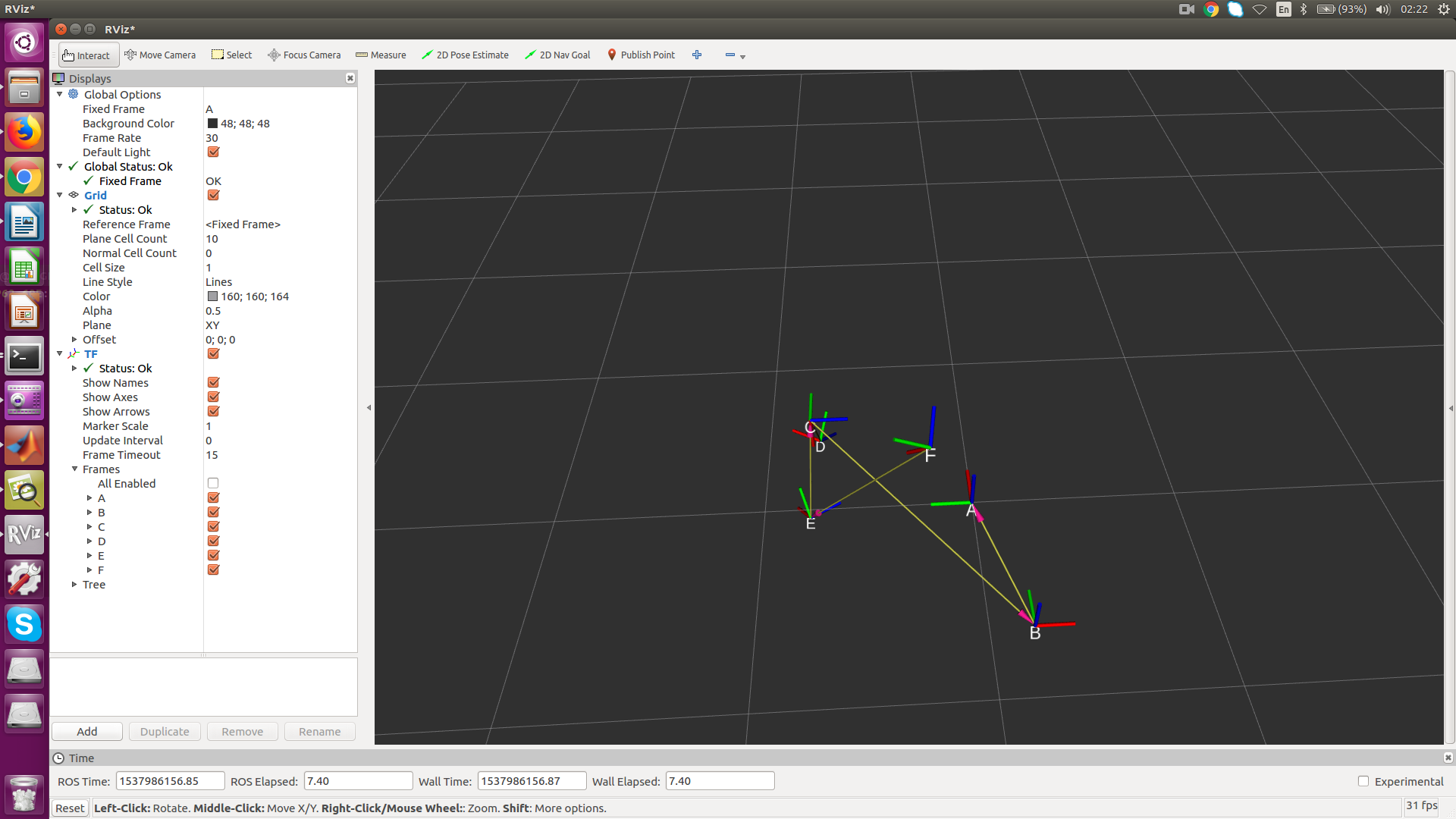




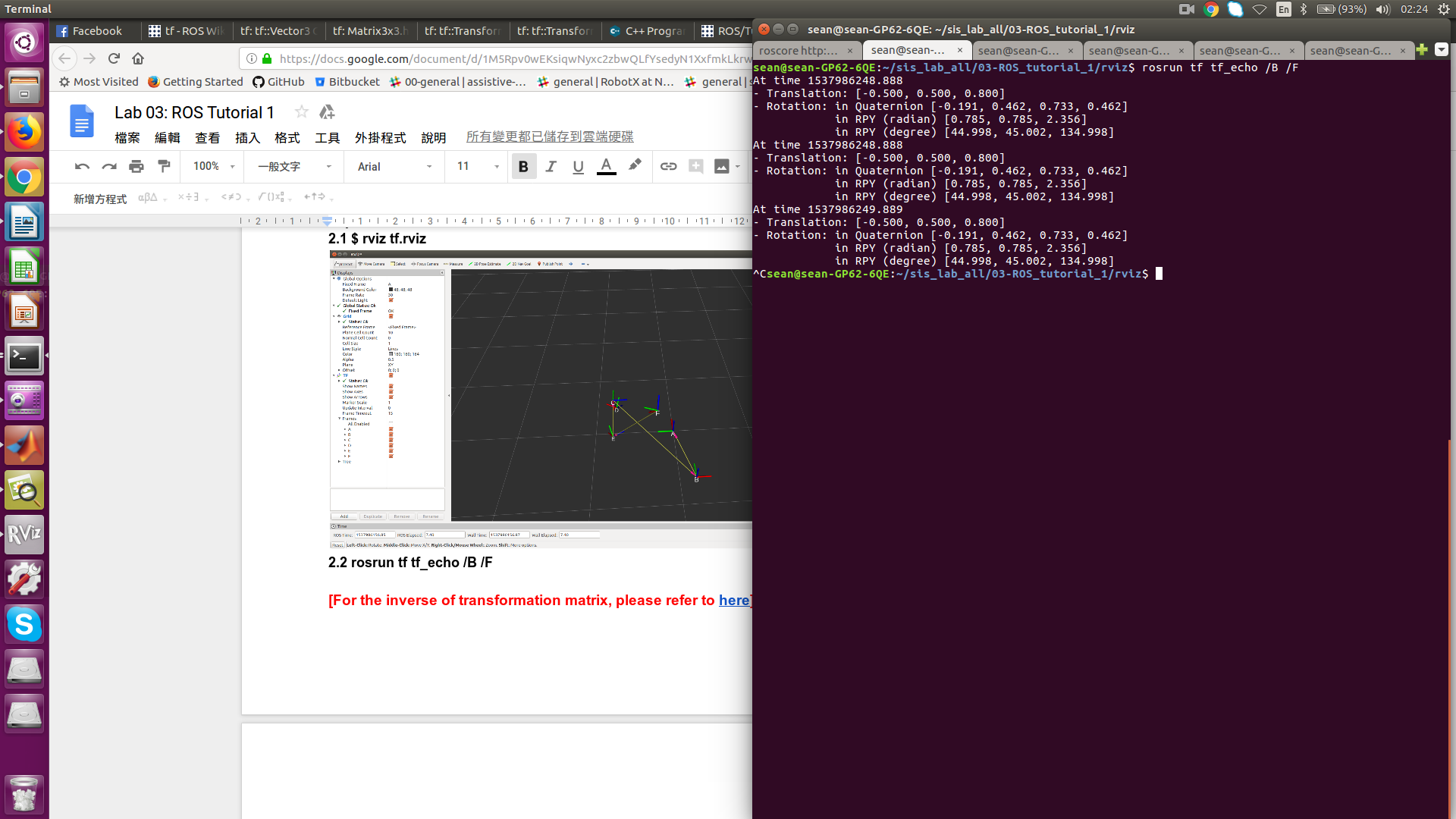
Edit **~/sis\_lab\_all\_2020/03-ROS\_tutorial\_1/catkin\_ws/src/tutorial/src/assignment2.cpp** such that you have the expected result illustrated below

**Expected Result:**

**Fig. 2.1 $ rviz -d tf.rviz**

****

**Fig. 2.2 $ rosrun tf tf\_echo /B /F**

****

**[Hint: Many tools available** [**here**](http://docs.ros.org/melodic/api/tf/html/c++/namespacetf.html)**]**

## Reference

[rospy API document](http://docs.ros.org/api/rospy/html/)