## **Simulation:**

**Tutorials:**

1. Richard Wang: <https://www.youtube.com/watch?v=8ckSl4MbZLg&list=PLTEmcYHVE7dPWixFnzkd68jPjwGzxHjUG&index=1>
2. Constructism:

<https://www.youtube.com/watch?v=q01dDxeHTMs>

<https://www.youtube.com/watch?v=AZGNIVFqXI8>

<https://www.youtube.com/watch?time_continue=510&v=f7b5tSZW1Ig>

1. Acutronic robotics

<https://www.youtube.com/watch?v=Zmuqx5e-1qQ>

1. Gazebo Website

<http://gazebosim.org/tutorials?tut=build_world&cat=build_world>

1. Erle robotics

<http://docs.erlerobotics.com/erle_robots/erle_copter/examples/visualizing_imu>

## **Creating a new workspace**

It's always better to create a new workspace when starting a new project so that you don’t disturb your existing projects.

> mkdir RMI\_PROJECT/src -p

> cd RMI\_PROJECT/src/

> catkin\_init\_workspace

> cd ..

> catkin\_make

You will have to add 2 lines to bash rc

If you have visual studio code then use code otherwise gedit will also work

> code ~/.bashrc

The lines are:

1. source /opt/ros/kinetic/setup.bash
2. source ~/RMI\_PROJECT/devel/setup.bash

Previously line 2 would have been source ~/catkin\_ws/devel/setup.bash if you had a catkin workspace  
The first line will mostly be there if you have used ROS previously. Second line is where you will have to make changes. If you are working with multiple work spaces only the top one will be sourced when you source the bashrc script so it is advised to comment other workspaces. After performing all the steps source the bashrc script using command

> source ~/.bashrc

For checking the whole process go to the terminal and do

> roscd

If you have followed all the steps then the current working directory will become ~/RMI\_PROJECT/devel

RMI\_PROJECT is the name of my workspace it can be different for you.

## **Using Gazebo**

Gazebo tutorials:

<http://gazebosim.org/tutorials>

ROS full desktop installs Gazebo so there is no need to install it separately.

The ROS package that integrates Gazebo with ROS is called gazebo\_ros\_pkgs.

After running roscore gazebo can be run using the command

> rosrun gazebo\_ros gazebo

Or

> roslaunch gazebo\_ros empty\_world.launch

When the previous command is run the gazebo runs two executables-the Gazebo server and the Gazebo client. The Gazebo server will execute the simulation process in the backend and the Gazebo client can be the Gazebo GUI which acts as a front end. Using the previous command, the Gazebo client and server will run in parallel.

By default, the modeling of robots and environments in Gazebo is done using the

SDF file. If we are using a ROS interface for Gazebo, we have to create a URDF file that contains all the parameters of the robot and has Gazebo-specific tags to mention the simulation properties of the robot. When we start the simulation using URDF, it will convert to an SDF file using some tools and display the robot in Gazebo.

To know more about gazebo user interface go to <http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b2>

In the edit menu use building editor to create buildings for drones. To learn more go to <http://gazebosim.org/tutorials?cat=build_world&tut=building_editor>

Components of gazebo

<http://gazebosim.org/tutorials?tut=components&cat=get_started>

Gazebo uses a distributed architecture with separate libraries for physics simulation, rendering, user interface, communication, and sensor generation. Additionally, gazebo provides two executable programs for running simulations:

* a server gzserver for simulating the physics, rendering, and sensors
* a client gzclient that provides a graphical interface to visualize and interact with the simulation

The client and server communicate using the gazebo communication library.

The physics, sensor, and rendering libraries support plugins. These plugins provide users with access to the respective libraries without using the communication system.

## **URDF**

Unified Robot Description Format(URDF) is an XML format that describes a robot, its parts, its joints, dimensions, and so on.

## **Using Hector Quadrotor simulation in kinetic**

Main Link: <https://darienmt.com/autonomous-flight/2018/10/20/flying-ros-and-hector.html>

Information on hector quadrotor package: <http://wiki.ros.org/hector_quadrotor>

Commands to be executed

> sudo apt-get install ros-kinetic-geographic-info

> sudo apt-get install ros-kinetic-ros-control

> sudo apt-get install ros-kinetic-gazebo-ros-control

> sudo apt-get install ros-kinetic-joy

> sudo apt-get install ros-kinetic-teleop-twist-keyboard

> cd ~

> cd RMI\_PROJECT/src

> mkdir hector

> cd hector

> wstool init src <https://raw.github.com/tu-darmstadt-ros-pkg/hector_quadrotor/kinetic-devel/tutorials.rosinstall>

> cd ..

> cd ..

> catkin\_make

Source the bash file

Now to see the simulation and control it

> roslaunch hector\_quadrotor\_demo outdoor\_flight\_gazebo.launch

> rosservice call /enable\_motors "enable: true"

> rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py

## **Errors**

1. E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)

E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?

Soln: > sudo rm /var/lib/apt/lists/lock

The next two lines may not be needed

> sudo rm /var/cache/apt/archives/lock

> sudo rm /var/lib/dpkg/lock

1. E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)

E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?

Soln: > sudo rm /var/lib/dpkg/lock-frontend

1. If sudo apt-get update doesn’t download

Soln: > echo 'Acquire::ForceIPv4 "true";' | sudo tee /etc/apt/apt.conf.d/99force-ipv4

1. Key error:

Soln: > sudo -E apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654