==========
Bibliography on ROS 2019-2020
==========
by A. Paulo Coimbra @ 2019-11-30
==========

https://www.toptal.com/robotics/introduction-to-robot-operating-system

https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/

Youtube tutorial:
https://www.youtube.com/watch?v=bJB9tv4ThV4

http://wiki.ros.org/kinetic/Installation/Ubuntu

http://wiki.ros.org/ROS/Tutorials

ROS Core Components:
https://www.ros.org/core-components/

UNIX Tutorial for Beginners:
http://www.ee.surrey.ac.uk/Teaching/Unix/
unixtut.zip
unixtut.tar.gz

==========

Notes/concepts:

apt -> Advanced Package Tool, or APT, is a free-software user interface that works with core libraries to handle the installation and removal of software on Debian, Ubuntu, and related Linux distributions.

catkin -> catkin is the official build system of ROS and the successor to the original ROS build system, rosbuild. catkin combines CMake macros and Python scripts to provide some functionality on top of CMake's normal workflow.

client libraries -> ROS client libraries allow nodes written in different programming languages to communicate: rospy (python client library), roscpp (c++ client library).

nodes -> A node is an executable that uses ROS to communicate with other nodes. ROS nodes use a ROS client library to communicate with other nodes. Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.

master -> Name service for ROS (i.e. helps nodes find each other)

messages -> ROS data type used when subscribing or publishing to a topic.
msg files are simple text files that describe the fields of a ROS message. They are used to generate source code for messages in different languages. msg files are stored in the msg directory of a package. msgs are just simple text files with a field type and a field name per line.

packages -> Packages are the software organization unit of ROS code. Each package can contain libraries, executables, scripts, or other artifacts. Each package must have its own folder. This means no nested packages nor multiple packages sharing the same directory.

package manifest -> The package manifest is an XML file called package.xml that must be included with any catkin-compliant package's root folder. This file defines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages.

recording and playing back data -> record data from a running ROS system into a *.bag file, and then play back the data to produce similar behavior in a running system.

roscore -> roscore is the first thing you should run when using ROS. roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate. It is launched using the roscore command (in a terminal window do: $ roscore).

To run anything in ROS, a core process needs to be launched. It’s as easy as opening a new terminal window and typing: roscore.
In your whole connected network of devices, roscore needs to be launched only once, on the device that will host the central hub for communication dispatching.
The main role of roscore is to tell nodes which other nodes they should connect to, and in which way (whether via a network port or shared memory). The goal is to allow nodes to only care about what data they want to know, rather than what node they want to connect to, while minimizing the time and bandwidth needed to perform all communication.

rosout: ROS equivalent of stdout/stderr

rospack -> rospack is a command-line tool for retrieving information about ROS packages available on the filesystem. Usage: $ rospack find [package_name]

RQt -> RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. One can run all the existing GUI tools as dockable windows within RQt! The tools can still run in a traditional standalone method, but RQt makes it easier to manage all the various windows in a single screen layout.

services -> ROS services are another way that nodes can communicate with each other. Services allow nodes to send a request and receive a response. An srv file describes a service. It is composed of two parts: a request and a response. srv files are stored in the srv directory. srv files are just like msg files, except they contain two parts: a request and a response. The two parts are separated by a '---' line.

topics -> Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.