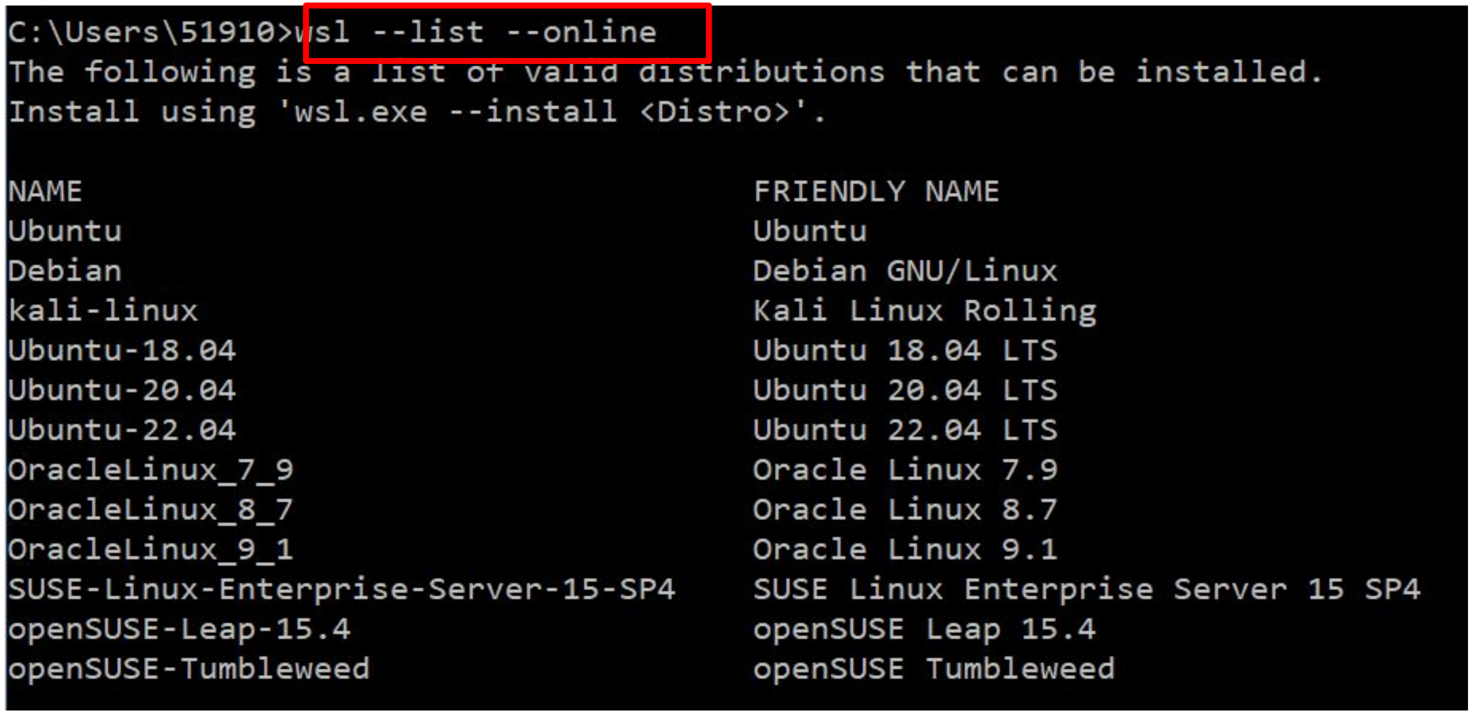
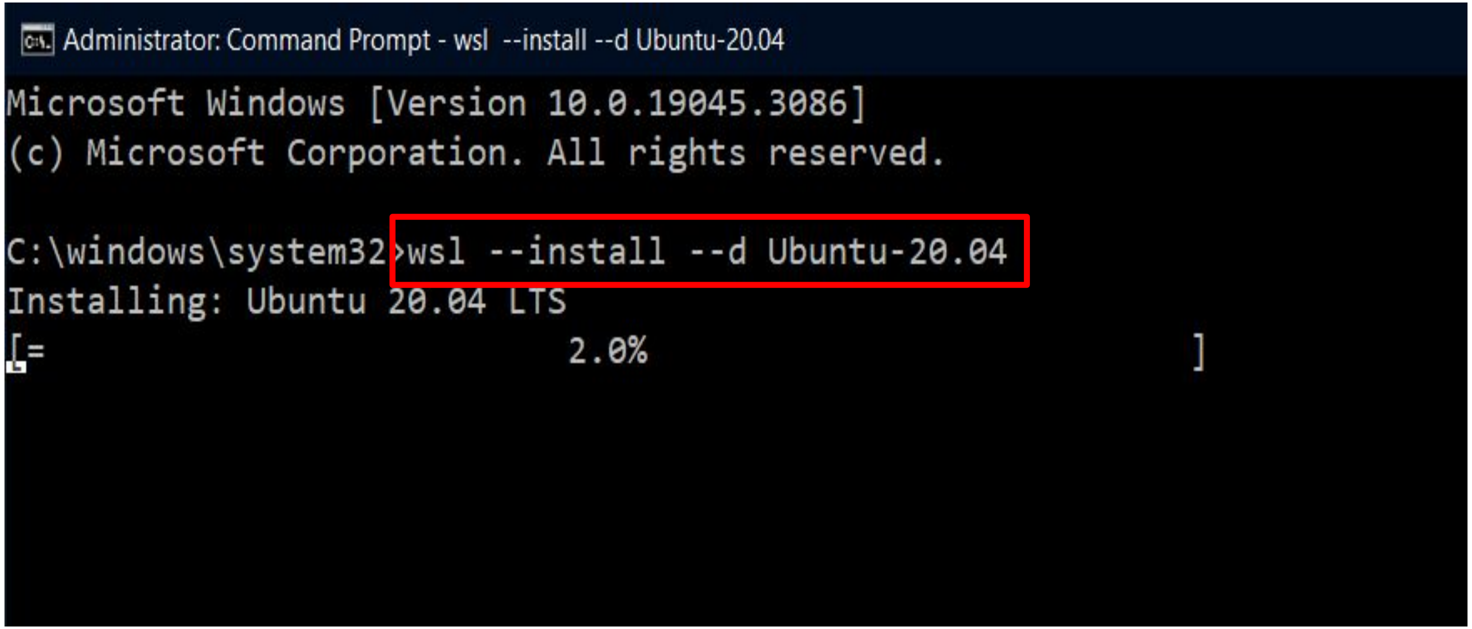
Installing Ubuntu 20.04 on WSL2



```
C:\Users\51910>wsl --list --online
The following is a list of valid distributions that can be installed.
Install using 'wsl.exe --install <Distro>'.

NAME                                   FRIENDLY NAME
Ubuntu                                 Ubuntu
Debian                                 Debian GNU/Linux
kali-linux                             Kali Linux Rolling
Ubuntu-18.04                           Ubuntu 18.04 LTS
Ubuntu-20.04                           Ubuntu 20.04 LTS
Ubuntu-22.04                           Ubuntu 22.04 LTS
OracleLinux_7_9                        Oracle Linux 7.9
OracleLinux_8_7                        Oracle Linux 8.7
OracleLinux_9_1                        Oracle Linux 9.1
SUSE-Linux-Enterprise-Server-15-SP4    SUSE Linux Enterprise Server 15 SP4
openSUSE-Leap-15.4                     openSUSE Leap 15.4
openSUSE-Tumbleweed                    openSUSE Tumbleweed
```

Note: Always run cmd or PowerShell as administrator.

Installing Ubuntu 20.04 on WSL2


Administrator: Command Prompt - wsl --install --d Ubuntu-20.04

```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\windows\system32>wsl --install --d Ubuntu-20.04
Installing: Ubuntu 20.04 LTS
[=                        2.0%                         ]
```

Note: If it is the first time to install WSL, you have to reboot the system for user and password setup.

Ubuntu terminal

Most powerful tool to interact with your computer. (Ctrl + Alt + T).

Commands:

```
cd
pwd
mkdir directory_name
rm -r directory_name
ls (ls > output.txt)
```

Ubuntu terminal

Commands:

```
mv file destination_directory/
cp file destination_directory/file_name
rm file
rm -r
sudo + another command
```

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

Ubuntu terminal

1. **sudo apt-get update:**

It will get the updated information of each installed package (metadata).

1. **sudo apt-get upgrade:**

Upgrade packages to the new version.

Robot Operating System (ROS)

ROS is a framework for building robot applications. Its main objective is to facilitate the reuse of code, packages and libraries in robotics research and development.

ROS distributes processes into nodes, which allows executables to be individually designed.

- Replace components with similar interfaces quickly.
- Connect components made in different programming languages making software development easier.
- There are a lot of reusable packages that are easy to integrate due to the architecture of the system.

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

ROS: Distributions

Distro	Release date	EOL date
ROS Noetic Ninjemys (**Recommended**)	May 23rd, 2020	May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018	June 27, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017	May, 2019
ROS Kinetic Kame	May 23rd, 2016	April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015	May, 2017
ROS Indigo Igloo	July 22nd, 2014	April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013	May, 2015
ROS Groovy Galapagos	December 31, 2012	July, 2014

http://wiki.ros.org/Distributions

ROS: Installation

1. Setup your computer to accept software from packages.ros.org:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Set up keys

```
$ sudo apt install curl
$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo
apt-key add -
```

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

ROS: Installation

3. Update Ubuntu:

```
$ sudo apt update
```

4. Install ROS Noetic:

```
$ sudo apt install ros-noetic-desktop
```

5. Source the bash file:

```
$ source /opt/ros/noetic/setup.bash
```

o

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

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

ROS: Installation

6. Install dependencies for building ROS packages:

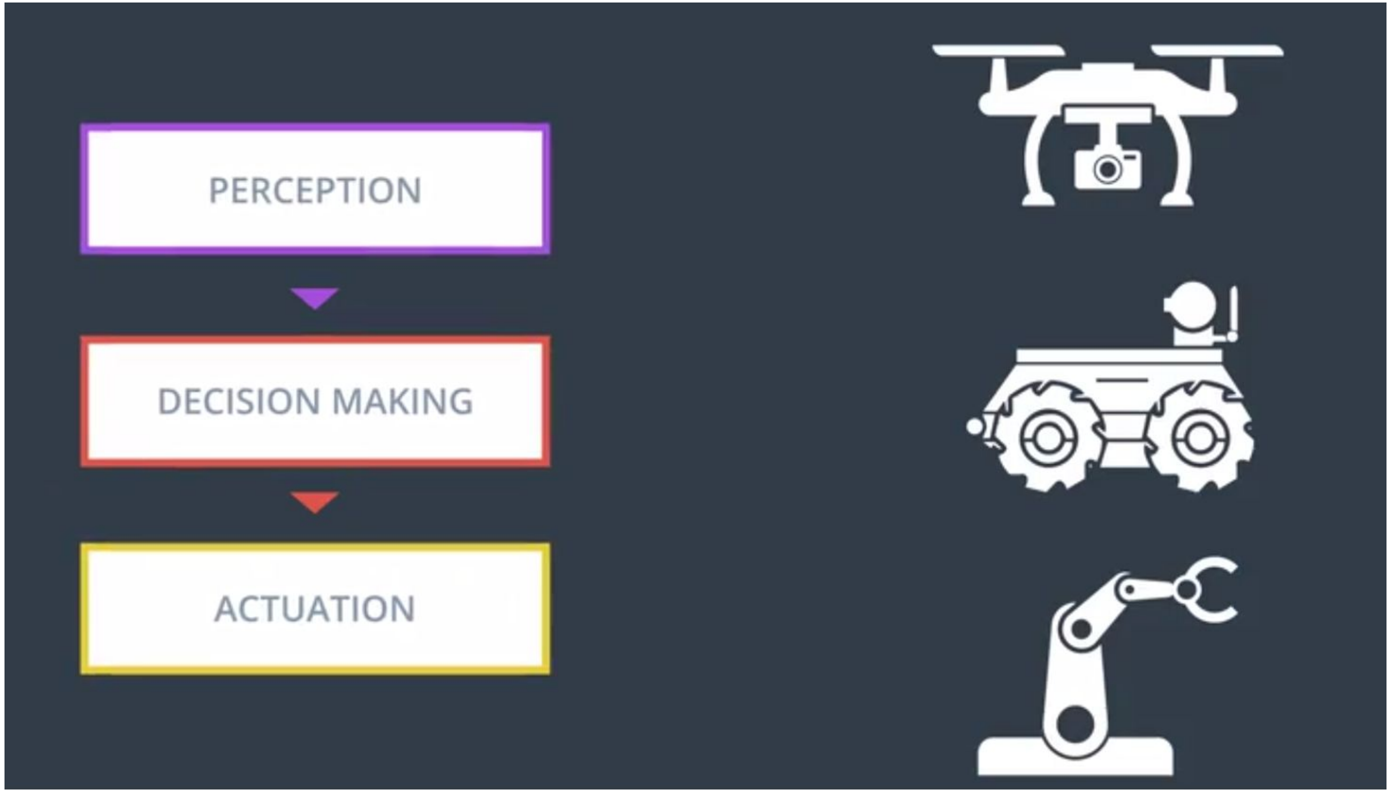
```
$ sudo apt install python3-rosdep python3-rosinstall
python3-rosinstall-generator python3-wstool build-essential
```

7. Initialize rosdep:

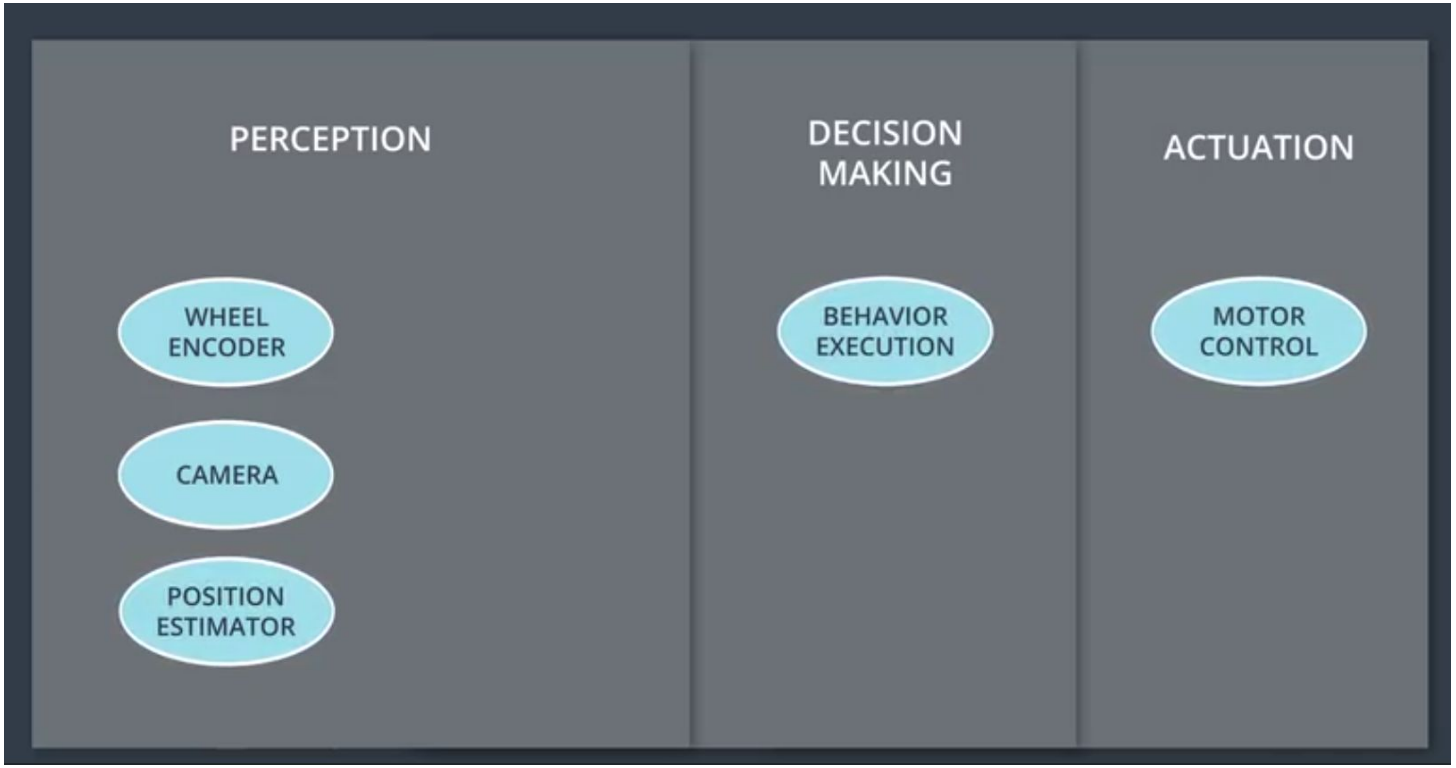
```
$ sudo apt install python3-rosdep
$ sudo rosdep init
$ rosdep update
```

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

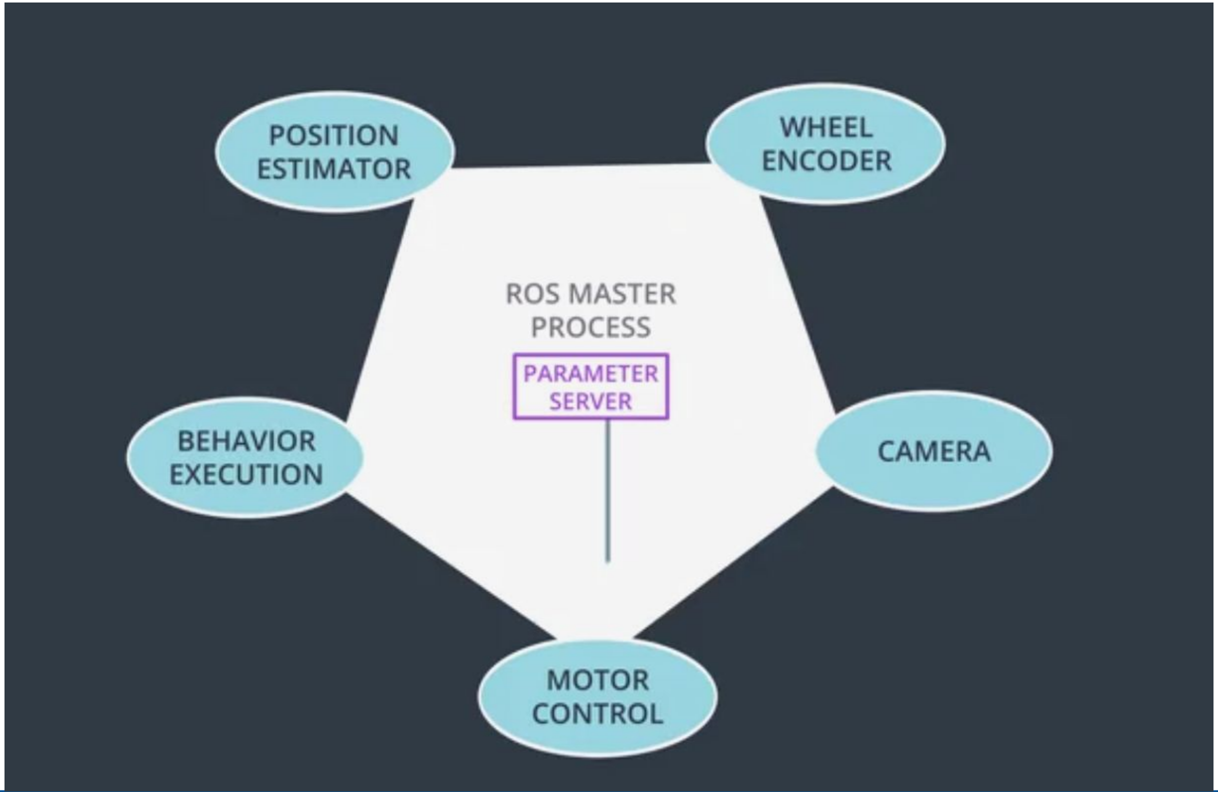
ROS: Nodes



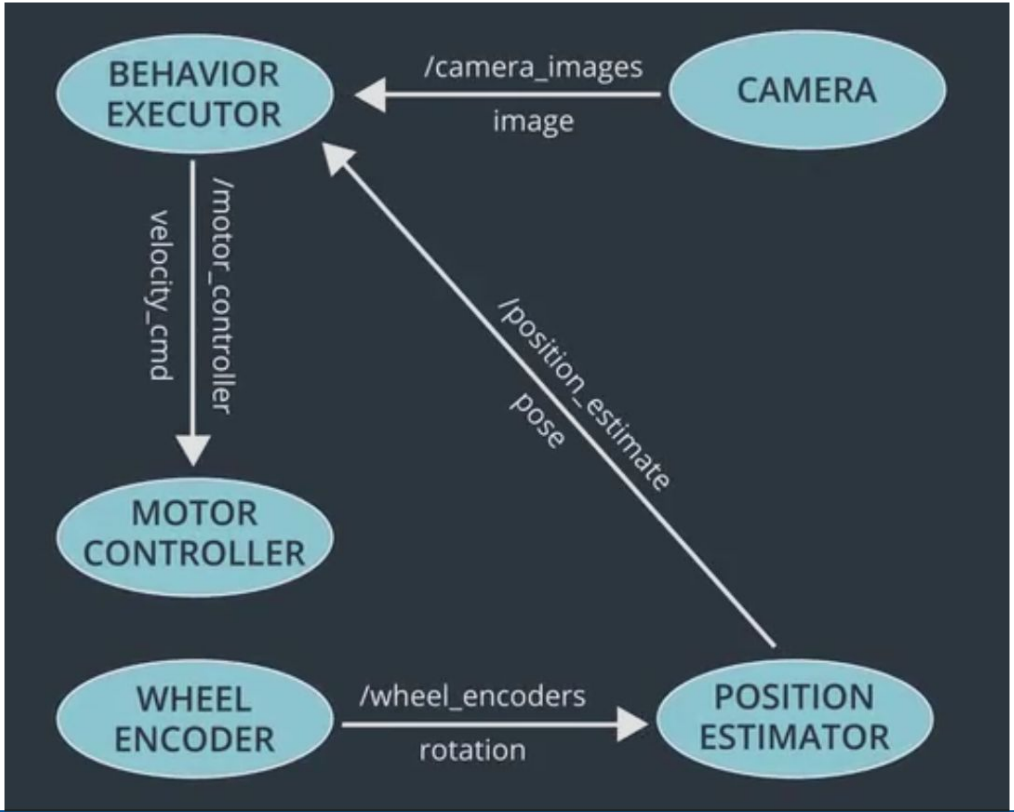
ROS: Nodes

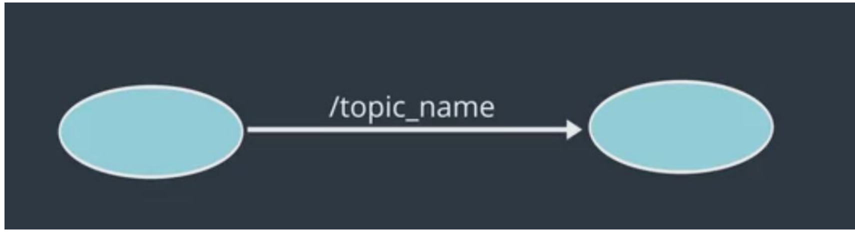

PERCEPTION
DECISION MAKING
ACTUATION
WHEEL ENCODER
CAMERA
POSITION ESTIMATOR
BEHAVIOR EXECUTION
MOTOR CONTROL

ROS: Nodes

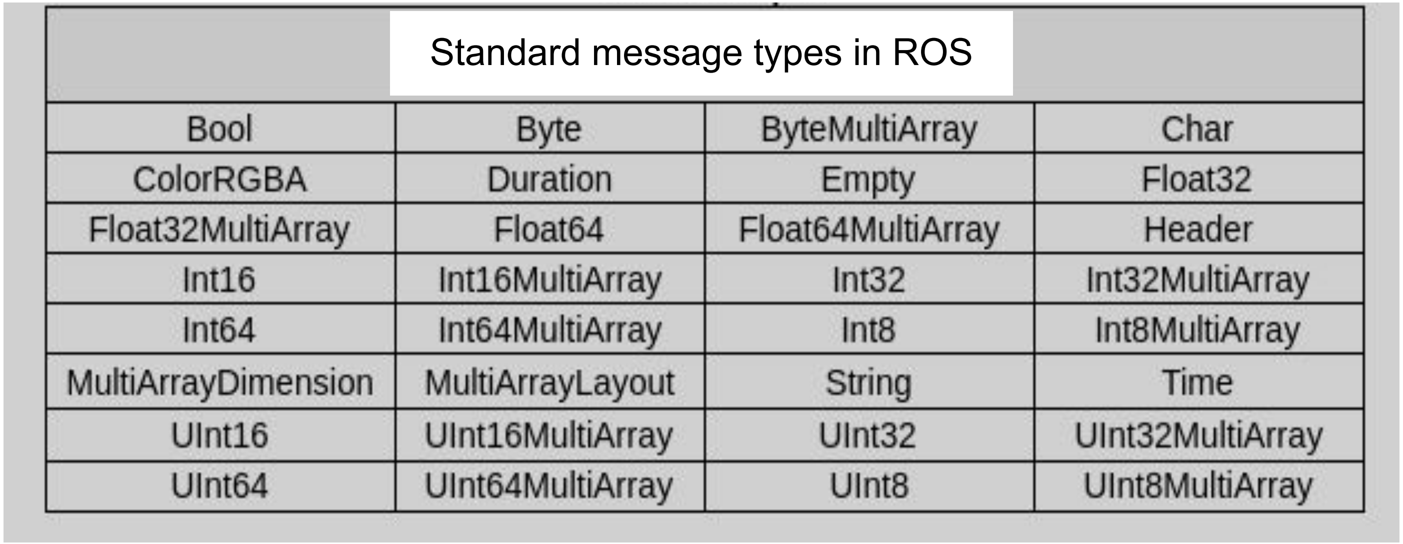

POSITION ESTIMATOR
WHEEL ENCODER
ROS MASTER PROCESS
PARAMETER SERVER
BEHAVIOR EXECUTION
CAMERA
MOTOR CONTROL

ROS: Nodes and topics





ROS: Messages



Standard message types in ROS			
Bool	Byte	ByteMultiArray	Char
ColorRGBA	Duration	Empty	Float32
Float32MultiArray	Float64	Float64MultiArray	Header
Int16	Int16MultiArray	Int32	Int32MultiArray
Int64	Int64MultiArray	Int8	Int8MultiArray
MultiArrayDimension	MultiArrayLayout	String	Time
UInt16	UInt16MultiArray	UInt32	UInt32MultiArray
UInt64	UInt64MultiArray	UInt8	UInt8MultiArray

http://wiki.ros.org/std_msgs

http://wiki.ros.org/sensor_msgs

ROS: Workspace

Definition: It is a folder where packages are stored. Here you can modify, build and install caktin packages.

We create a Catkin (compiler used in ROS) Workspace in the root of the system. To create we only create the following directories:

```
$ mkdir -p ros_ws/src
```

Then we enter the folder where we declare our workspace with the caktin_make command:

```
$ cd ros_ws
$ catkin_make
```

ROS: Workspace

Once our workspace is initialized, it must be indicated to the system that this is a ros workspace and it will contain packages. This indication is made with the following command:

```
$ source /home/user/ros_ws/devel/setup.bash
```

ROS: Package

Definition: Packages are the most atomic building unit and the launch unit. The codes or nodes are stored in this directory.

Create a new package in our workspace with the following code:

```
$ cd ros_ws/src
$ catkin_create_pkg new_package dependencies_name
```

ROS: Package

Every package must include two files: CMakeLists.txt y package.xml.

```
$ cd ~/ros_ws/src
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
$ cd ~/ros_ws
$ catkin_make
$ source devel/setup.bash
```

ROS: Turtlesim node

Run in a terminal:

```
$ roscore
```

In another terminal run:

```
$ rosrun turtlesim turtlesim_node
```

In a third terminal run:

```
$ rosnode list
$ rostopic list
```

ROS: Turtlesim node

```
$ rosrun turtlesim turtle_teleop_key
```

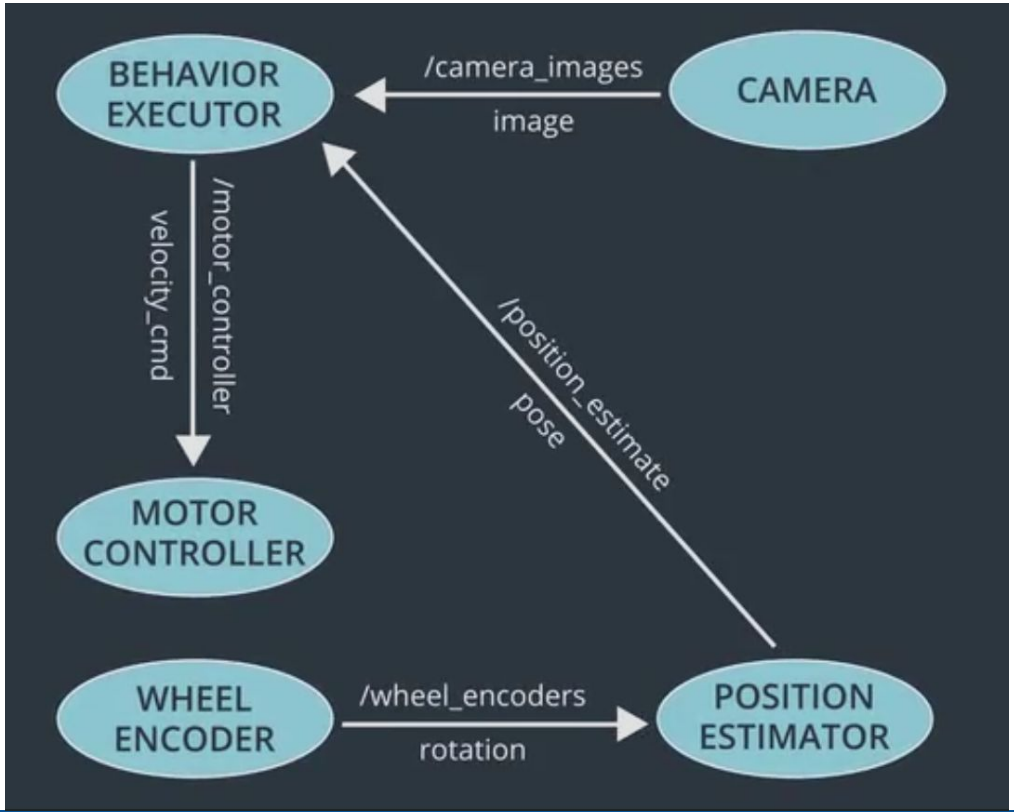
In a fourth terminal run:

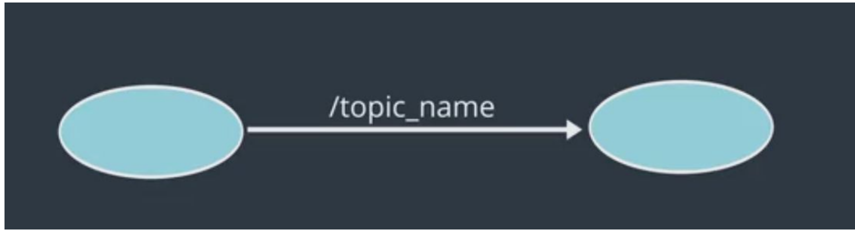
```
$ sudo apt-get install ros-noetic-rqt
$ sudo apt-get install ros-noetic-rqt-common-plugins
$ rosrun rqt_graph rqt_graph
```

ROS: Turtlesim node

```
$ rostopic echo /turtle1/cmd_vel

$ rostopic type /turtle1/cmd_vel

$ rosmsg show geometry_msgs/Twist

$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0,
0.0, 1.8]'
```

ROS: Nodes and topics





ROS Nodes: Publisher

Change directory into the beginner_tutorials package:

```
$ cd beginner_tutorials
```

Create a script directory inside the package:

```
$ mkdir scripts
$ cd scripts
```

Create a python script called talker.py (using nano or gedit).

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

ROS Nodes: Publisher

```
#!/usr/bin/env python3

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

ROS Nodes: Publisher

```
import rospy
from std_msgs.msg import String
```

You need to import rospy if you are writing a ROS Node

```
pub = rospy.Publisher('chatter', String, queue_size=10)
rospy.init_node('talker', anonymous=True)
```

With rospy.Publisher you declares that your node is publishing information.

rospy.Publisher(‘topic name’, ‘message type’, ‘buffer size’)

With rospy init node you set your node name.

anonymous = True ensures that your node has a unique name

```
rate = rospy.Rate(10) # 10hz
```

Allows you to create loops at a desired speed

ROS Nodes: Publisher

```
while not rospy.is_shutdown():
    hello_str = "hello world %s" % rospy.get_time()
    rospy.loginfo(hello_str)
    pub.publish(hello_str)
    rate.sleep()
```

Your script have to check *is_shutdown()* to check if your program should exit. There are 3 ways of exit:

1. Ctrl-C is pressed.
2. A node with the same name is initialized
3. Roscore is stopped.

rospy.loginfo(‘message’) allows to display the message on the screen.
pub.publish(‘message’) publishes a string to our topic.
rate.sleep() maintain the desired rate through the loop.

ROS Nodes: Publisher

Convert the script into an executable file

```
 $ chmod +x talker.py
```

Add the following to your **CMakeLists.txt.** This makes sure the python script gets installed properly, and uses the right python interpreter.

```
catkin_install_python(PROGRAMS scripts/talker.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

ROS Nodes: Subscriber

Inside your package create a python script called listener.py (using nano or gedit).

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    rospy.spin()

if __name__ == '__main__':
    listener()
```

ROS Nodes: Subscriber

```
def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
```

Every time the node receives a message, it will execute the "callback" function.

```
def listener():

    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    rospy.spin()
```

With rospy.Subscriber you declares that your node is receiving information.
rospy.Subscriber('topic name', 'message type', 'callback function name').

rospy.spin () keeps your node working until the node has been shutdown

ROS Nodes: Subscriber

Convert the script into an executable file

```
$ chmod +x listener.py
```

Add the following to your **CMakeLists.txt.** This makes sure the python script gets installed properly, and uses the right python interpreter.

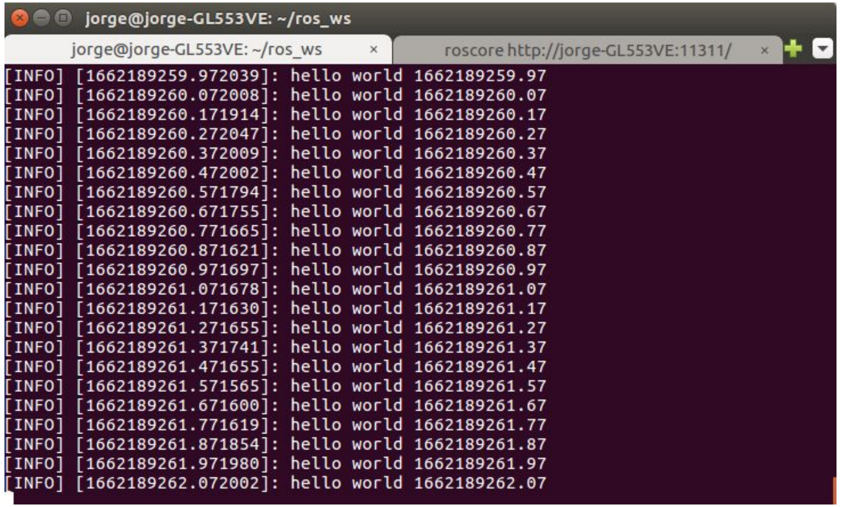
```
catkin_install_python(PROGRAMS scripts/talker.py  scripts/listener.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

In order to build the nodes

```
$ cd ~/catkin_ws
$ catkin_make
```

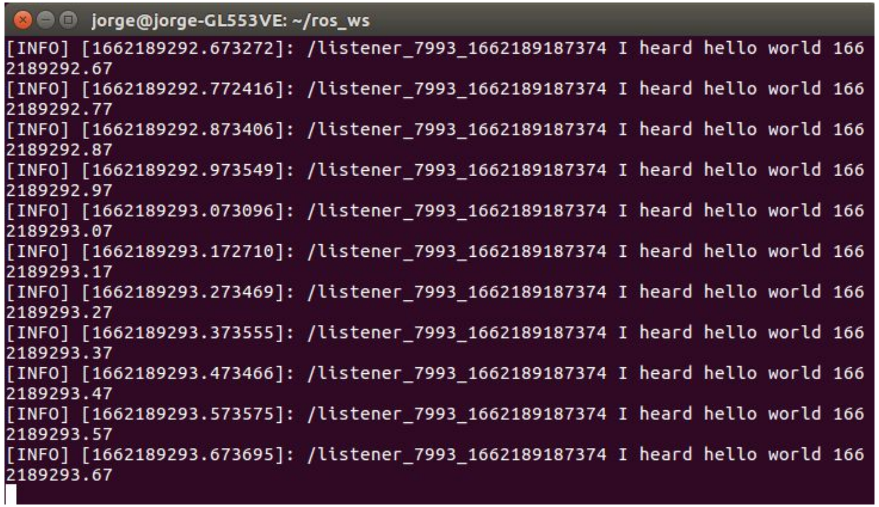
ROS Nodes

talker.py


jorge@jorge-GL553VE: ~/ros_ws
jorge@jorge-GL553VE: ~/ros_ws
roscore http://jorge-GL553VE:11311/

```
[INFO] [1662189259.972039]: hello world 1662189259.97
[INFO] [1662189260.072008]: hello world 1662189260.07
[INFO] [1662189260.171914]: hello world 1662189260.17
[INFO] [1662189260.272047]: hello world 1662189260.27
[INFO] [1662189260.372009]: hello world 1662189260.37
[INFO] [1662189260.472002]: hello world 1662189260.47
[INFO] [1662189260.571794]: hello world 1662189260.57
[INFO] [1662189260.671755]: hello world 1662189260.67
[INFO] [1662189260.771665]: hello world 1662189260.77
[INFO] [1662189260.871621]: hello world 1662189260.87
[INFO] [1662189260.971697]: hello world 1662189260.97
[INFO] [1662189261.071678]: hello world 1662189261.07
[INFO] [1662189261.171630]: hello world 1662189261.17
[INFO] [1662189261.271655]: hello world 1662189261.27
[INFO] [1662189261.371741]: hello world 1662189261.37
[INFO] [1662189261.471655]: hello world 1662189261.47
[INFO] [1662189261.571565]: hello world 1662189261.57
[INFO] [1662189261.671600]: hello world 1662189261.67
[INFO] [1662189261.771619]: hello world 1662189261.77
[INFO] [1662189261.871854]: hello world 1662189261.87
[INFO] [1662189261.971980]: hello world 1662189261.97
[INFO] [1662189262.072002]: hello world 1662189262.07
```

listener.py



```
[INFO] [1662189292.673272]: /listener_7993_1662189187374 I heard hello world 166
2189292.67
[INFO] [1662189292.772416]: /listener_7993_1662189187374 I heard hello world 166
2189292.77
[INFO] [1662189292.873406]: /listener_7993_1662189187374 I heard hello world 166
2189292.87
[INFO] [1662189292.973549]: /listener_7993_1662189187374 I heard hello world 166
2189292.97
[INFO] [1662189293.073096]: /listener_7993_1662189187374 I heard hello world 166
2189293.07
[INFO] [1662189293.172710]: /listener_7993_1662189187374 I heard hello world 166
2189293.17
[INFO] [1662189293.273469]: /listener_7993_1662189187374 I heard hello world 166
2189293.27
[INFO] [1662189293.373555]: /listener_7993_1662189187374 I heard hello world 166
2189293.37
[INFO] [1662189293.473466]: /listener_7993_1662189187374 I heard hello world 166
2189293.47
[INFO] [1662189293.573575]: /listener_7993_1662189187374 I heard hello world 166
2189293.57
[INFO] [1662189293.673695]: /listener_7993_1662189187374 I heard hello world 166
2189293.67
```