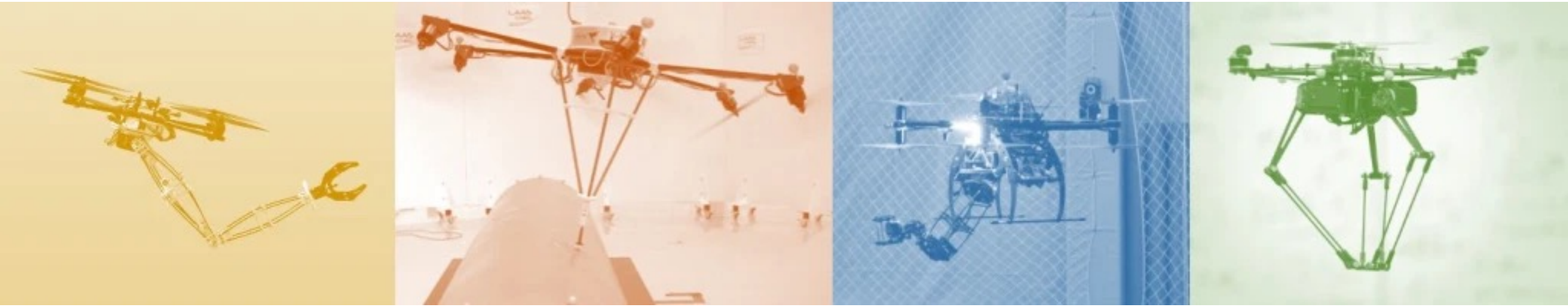
Autonomous Systems



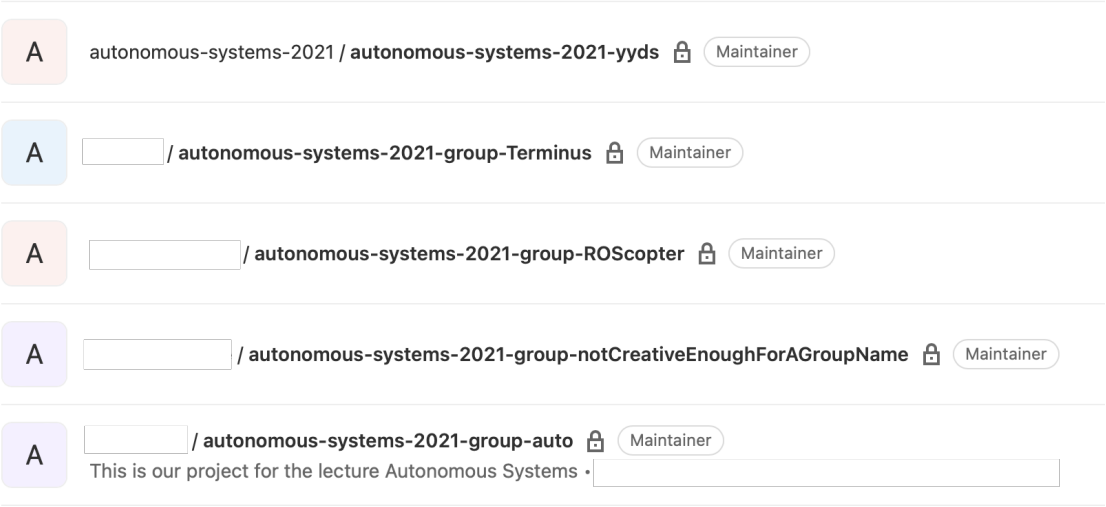
Christoph Killing

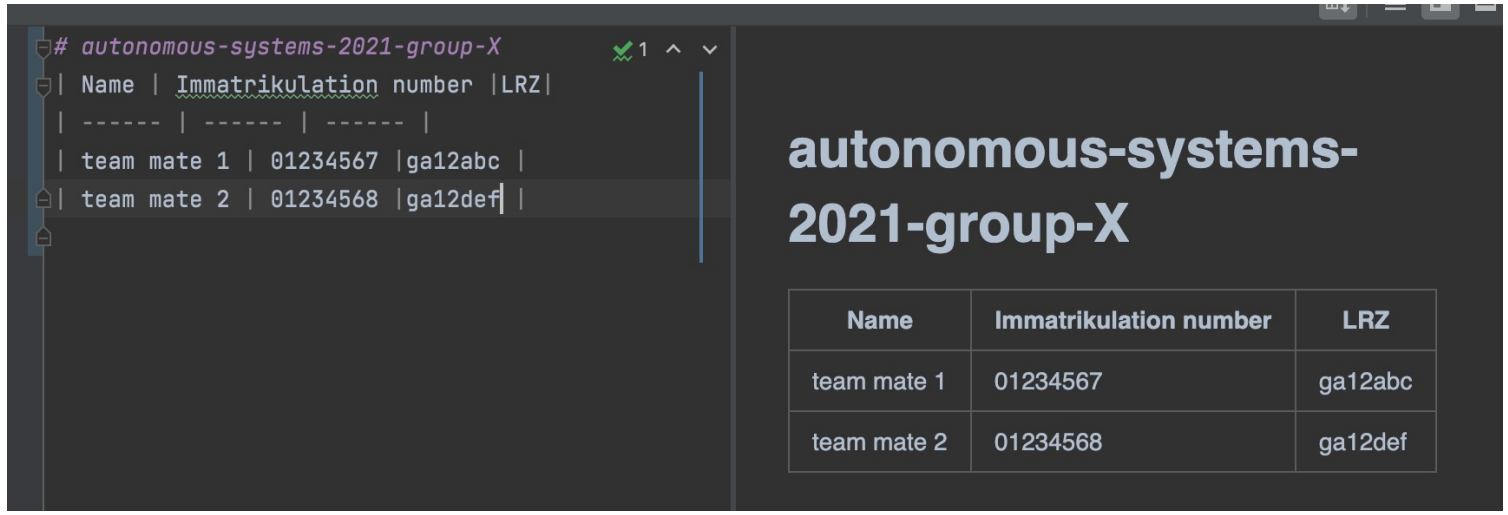
Lab 2 – Intro to ROS

Autonomous Aerial Systems

Homework & Groups

- Please share a REPO!
- README info:





Autonomous Aerial Systems

What is ROS

ROS = **R**obot **O**perating **S**ystem

What is ROS

ROS = **R**obot **O**perating **S**ystem

=

Plumbing

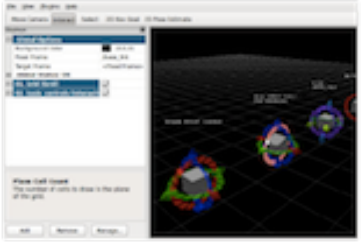
- **Modular software architecture**
- **Message passing**
- **Plugin library**
- **...**

What is ROS

ROS = Robot Operating System

=

+



Plumbing

- Modular software architecture
- Message passing
- Plugin library
- ...

Tools

- Introspection
- Visualization
- Data logging
- Build system
- ...

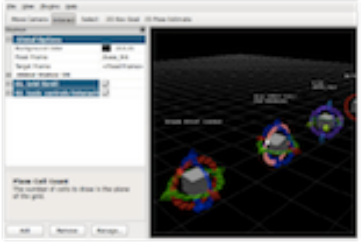
Autonomous Aerial Systems

What is ROS

ROS = Robot Operating System

=

+



+

Plumbing

- Modular software architecture
- Message passing
- Plugin library
- ...

Tools

- Introspection
- Visualization
- Data logging
- Build system
- ...

Capabilities

- Localization
- Mapping
- Motion planning
- Manipulation
- ...

What is ROS

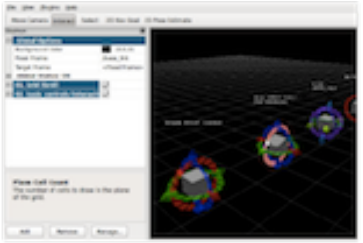
ROS = Robot Operating System

=

Plumbing

- Modular software architecture
- Message passing
- Plugin library
- ...

+



Tools

- Introspection
- Visualization
- Data logging
- Build system
- ...

+

Capabilities

- Localization
- Mapping
- Motion planning
- Manipulation
- ...

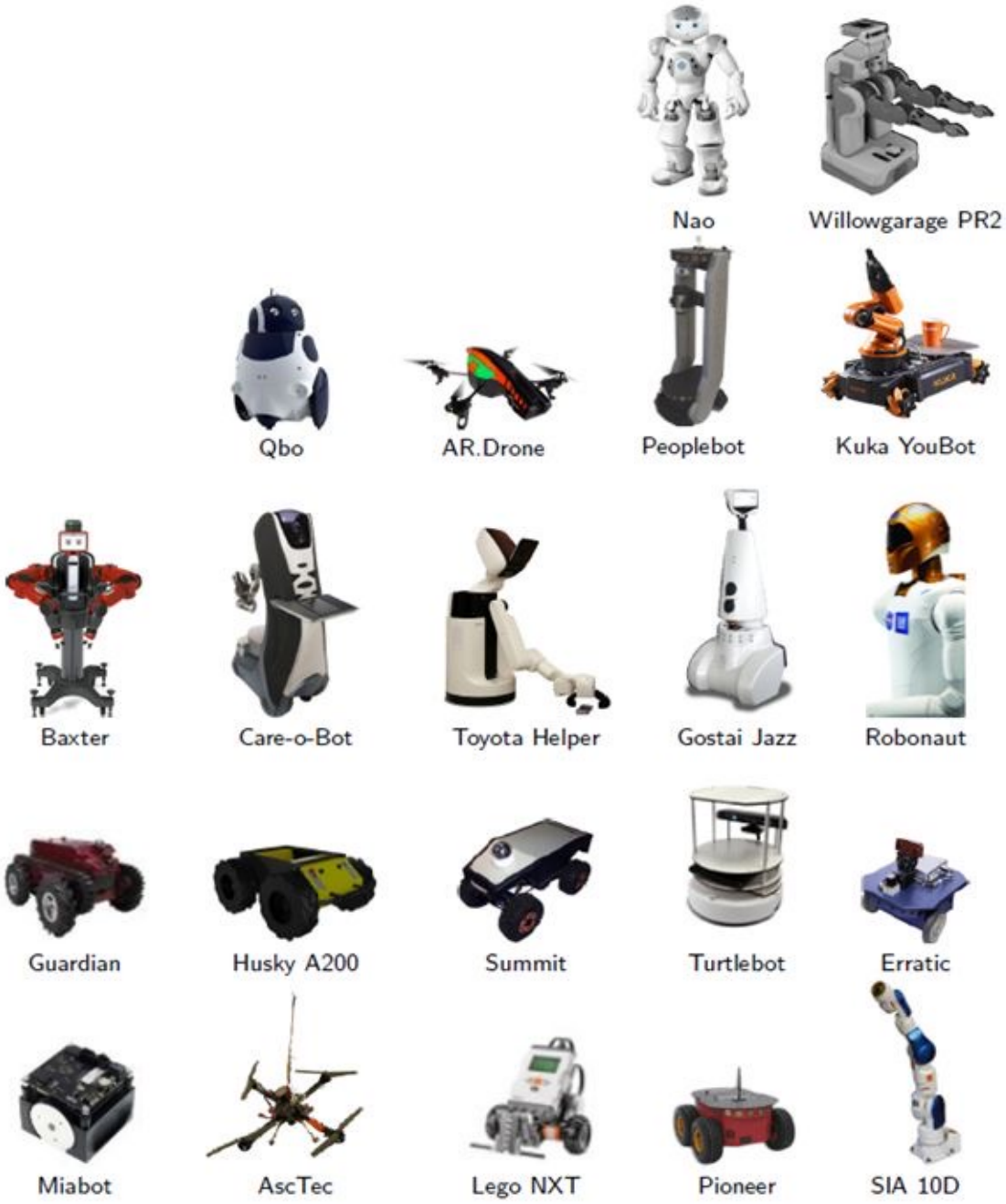
+

Ecosystem

- Software distribution
- Community
- Documentation
- Best practices
- ...

History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- De facto standard for robot programming



Autonomous Aerial Systems

ROS Philosophy

- Peer to peer
  Individual programs communicate over defined API (ROS messages, services, etc.)

ROS Philosophy

- **Peer to peer**
  Individual programs communicate over defined API (ROS messages, services, etc.)
- **Distributed**
  Programs can be run on multiple computers and communicate over networks

ROS Philosophy

- **Peer to peer**
  Individual programs communicate over defined API (ROS messages, services, etc.)
- **Distributed**
  Programs can be run on multiple computers and communicate over networks
- **Multi-lingual**
  ROS modules can be written in any language for which a client library exists (C++, Python, Matlab, Java, Giulia, etc.)

ROS Philosophy

- **Peer to peer**
  Individual programs communicate over defined API (ROS messages, services, etc.)
- **Distributed**
  Programs can be run on multiple computers and communicate over networks
- **Multi-lingual**
  ROS modules can be written in any language for which a client library exists (C++, Python, Matlab, Java, Giulia, etc.)
- **Light-weight**
  Stand alone libraries are wrapped around with a thin ROS layer

ROS Philosophy

- **Peer to peer**
  Individual programs communicate over defined API (ROS messages, services, etc.)
- **Distributed**
  Programs can be run on multiple computers and communicate over networks
- **Multi-lingual**
  ROS modules can be written in any language for which a client library exists (C++, Python, Matlab, Java, Giulia, etc.)
- **Light-weight**
  Stand alone libraries are wrapped around with a thin ROS layer
- **Free and open-source**
  Most ROS software is open-source and free to use

ROS Workspace Environment

- Defines context for the current workspace
- Default workspace loaded with

```
> source /opt/ros/[ROS distro melodic/noetic]/setup.bash
```

Overlay your catkin workspace with

```
> cd ~/catkin_ws
> source devel/setup.bash
```

Check your workspace with

```
> echo $ROS_PACKAGE_PATH
```

Autonomous Aerial Systems

ROS Nodes

- Single-purpose, executable program
- Individually compiled, executed and managed
- Organized in *packages*

Run a node with

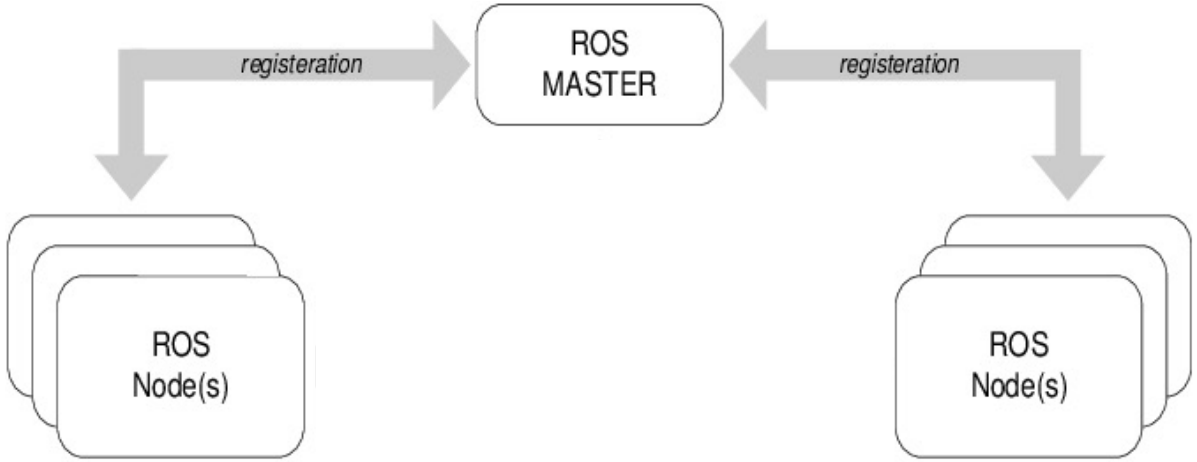
```
> rosrun package_name node_name
```

See active nodes with

```
> rosnode list
```

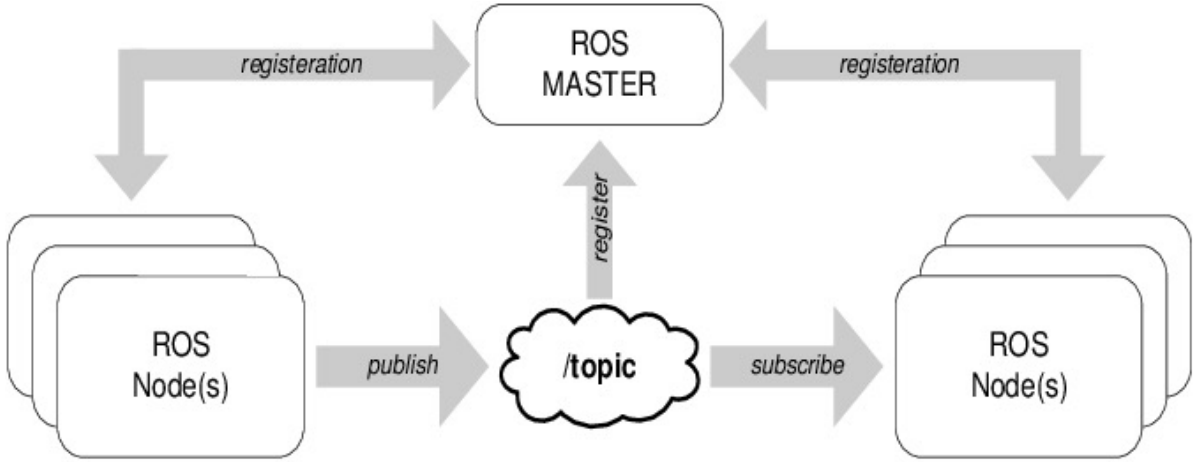
Retrieve information about a node with

```
> rosnode info node_name
```



ROS Topics

- Nodes communicate over *topics*
  - Nodes can *publish* or *subscribe* to a topic
  - Typically, n subscribers and 1 publisher
- Topic is a name for a stream of *messages*



List active topics with

```
> rostopic list
```

Subscribe and print the content of a topic with

```
> rostopic echo /topic_name
```

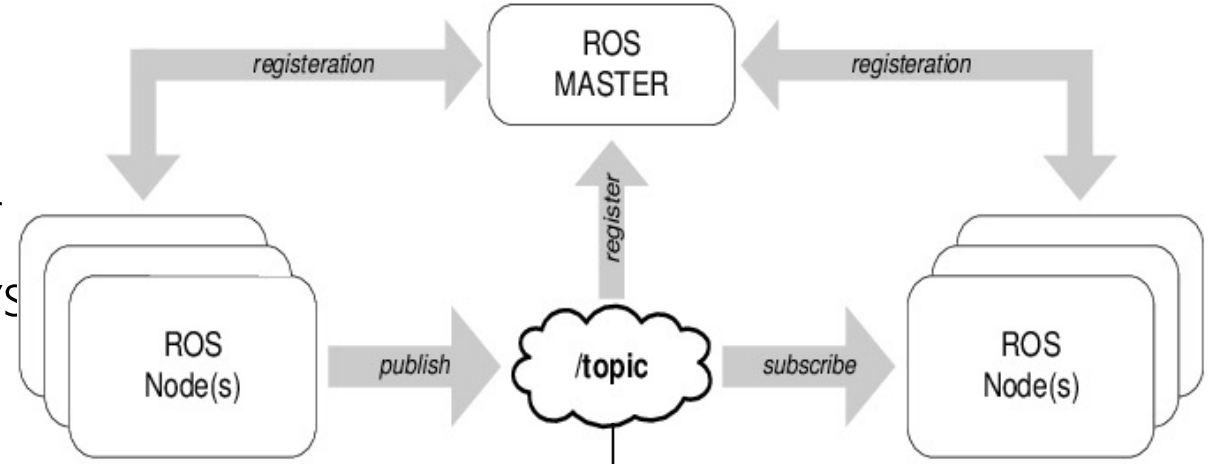
Show information about a topic with

```
> rostopic info /topic_name
```

Autonomous
Aerial Systems

ROS Messages

- Data structure defining the type of a topic
- Compromised of a nested structure of integers, floats, booleans, etc. and arrays of objects
- Defined in *.msg files



*.msg

Message definition

int number
double width
string description
etc.

See the type of a topic

```
> rostopic type /topic_name
```

Publish a message to a topic

```
> eostopic pub /topic type args
```

Autonomous
Aerial Systems

ROS Messages

geometry_msgs/Point.msg

```
float64 x
float64 y
float64 z
```

sensor_msgs/Image.msg

```
std_msgs/Header header
        uint32 seq
        time stamp
        string frame_id
uint 32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
        uint32 seq
        time stamp
        string frame_id
Geometry_msgs/Pose pose
        geometry_msgs/Point position
                float64 x
                float64 y
                float64 z
        geometry_msgs/Quaternion orientation
                float 64 x
                float 64 y
                float 64 z
                float 64 w
```

Autonomous Aerial Systems

Installing ROS

- http://wiki.ros.org/ROS/Installation

A simple example

Useful tools:
- rviz (ROS visualizer)
- rosbag & rqt_bag
- rqt_graph

- Setup workspace

```
mkdir ~/catkin_ws
cd ~/catkin_ws
catkin init
```

- Launch roscore

```
roscore
```

- Launch first node

```
rosrun turtlesim turtlesim_node
rosrun turtlesim turtle_teleop_key
```

➢ http://wiki.ros.org/tf/Tutorials

Autonomous
Aerial Systems