# SE498 Introduction to Autonomous Vehicle System

# Laboratory Assignment 1: Introduction to Robot Operating System (ROS)

**Goals for this Lab Assignment:**

1. What is ROS?
2. ROS commands (roslaunch, rosrun, rostopic, rosmsg, rosservice, etc)
3. ROS programming using C++

**Exercise 1 – What is ROS?**

ROS package: http://wiki.ros.org/Packages

ROS node: http://wiki.ros.org/Nodes

ROS master: http://wiki.ros.org/Master

ROS message types: http://wiki.ros.org/msg

ROS service types: http://wiki.ros.org/srv

ROS topics: http://wiki.ros.org/Topics

ROS services: http://wiki.ros.org/Services

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Create ROS workspace

```
$ cd ~
$ mkdir -p ~/catkin_NETID/src
$ cd ~/catkin_NETID/
$ catkin_make
$ source devel/setup.bash
```

**Exercise 2 – ROS commands**

$ **roscore**

$ **rosrun** turtlesim turtlesim_node

$ **rosrun** turtlesim turtle_teleop_key

(1) **roslaunch**: a tool for easily launching multiple ROS nodes locally and remotely via SSH, as well as setting parameters on the Parameter Server.

```
$ roslaunch <ros_package_name> <ros_launch_file.launch>
```

(2) **rosrun**: run a ROS node of a ROS package

```
$ rosrun <ros_package_name> <executable_file>
```

(3) **rostopic**: display debug info about ROS topics, including publishers, subscribers, publishing rate, and ROS messages.

```
$ rostopic list
$ rostopic info <ros_topic_name>
$ rostopic echo <ros_topic_name>
$ rostopic hz <ros_topic_name>
$ rostopic type <ros_topic_name>
$ rosmsg show <ros_topic_type>
$ rostopic pub <ros_topic_name> <topic(message)_type> [args]
$ rostopic pub -r 10 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

(3) **rosservice**: tool for listing and querying ROS services.

```
$ rosservice list
$ rosservice info <ros_service_name>
$ rosservice type <ros_service_name>
$ rossrv show <ros_service_type>
$ rosservice call <ros_service_name> [args]
$ rosservice call /spawn 2 2 0.2 ""
```

**Exercise 3 – ROS programming using C++**

```
$ cd ~/catkin_NETID/src
$ catkin_create_pkg se498_lab1 roscpp
$ cd .. && catkin_make
$ cd ~/catkin_NETID/src/se498_lab1/src
$ touch lab1.cpp
```

**lab1.cpp**

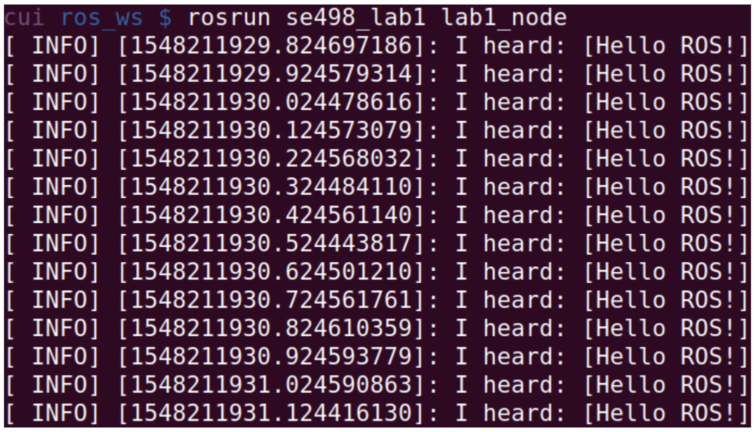
```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>

void fooCallback(const std_msgs::String::ConstPtr& msg) {
    ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv) {

    ros::init(argc, argv, "tutorial");
    ros::NodeHandle n;
    ros::Publisher bar_pub = n.advertise<std_msgs::String>("foo_topic", 1000);
    ros::Subscriber sub = n.subscribe("foo_topic", 1000, fooCallback);
    ros::Rate loop_rate(10);

    while (ros::ok()) {
        std_msgs::String msg;
        std::stringstream ss;
        ss << "Hello ROS!";
        msg.data = ss.str();
        bar_pub.publish(msg);
        ros::spinOnce();
        loop_rate.sleep();
    }

    return 0;

}
```

**Modify CMakeLists.txt**

```
$ cd ~/catkin_NETID/src/se498_lab1/
$ gedit CmakeLists.txt
```

```
add_executable(lab1_node src/lab1.cpp)
target_link_libraries(lab1_node ${catkin_LIBRARIES})
add_dependencies(lab1_node se498_lab1_generate_messages_cpp)
```

```
$ cd ~/catkin_NETID/
$ source devel/setup.bash
$ roscore
$ rosrun se498_lab1 lab1_node
```



```
cui ros_ws $ rosrun se498_lab1 lab1_node
[ INFO] [1548211929.824697186]: I heard: [Hello ROS!]
[ INFO] [1548211929.924579314]: I heard: [Hello ROS!]
[ INFO] [1548211930.024478616]: I heard: [Hello ROS!]
[ INFO] [1548211930.124573079]: I heard: [Hello ROS!]
[ INFO] [1548211930.224568032]: I heard: [Hello ROS!]
[ INFO] [1548211930.324484110]: I heard: [Hello ROS!]
[ INFO] [1548211930.424561140]: I heard: [Hello ROS!]
[ INFO] [1548211930.524443817]: I heard: [Hello ROS!]
[ INFO] [1548211930.624501210]: I heard: [Hello ROS!]
[ INFO] [1548211930.724561761]: I heard: [Hello ROS!]
[ INFO] [1548211930.824610359]: I heard: [Hello ROS!]
[ INFO] [1548211930.924593779]: I heard: [Hello ROS!]
[ INFO] [1548211931.024590863]: I heard: [Hello ROS!]
[ INFO] [1548211931.124416130]: I heard: [Hello ROS!]
```

**Checkoff**

(1) Successfully demo of Exercise 3 and detailed comments of lab1.cpp.