

Note: during lectures have students install mobaXterm

1 ROS In a Nutshell

- **Peer-to-Peer:** Individual programs communicate over defined API
- **Distributed:** Programs can run on any number of computers
- **Multilingual:** Programs can be written in any language with a client library (Official: C++/Python, Unofficial: Java, Javascript, Rust, Julia)
- **Lightweight:** Libraries are very thin/have low overhead
- **Free and Open Source:** ROS in its entirety is open source

2 ROS Master

- Manages comms between nodes (processes)
- Every node registers with the master at startup
- Started with roscore.

3 ROS Nodes

- Single purpose executable program
- Individually compiled, executed, managed
- Organized in *packages*
- Run a node with `roslaunch <package_name> <node_name>`
- See active nodes with `roslaunch list`
- See information about a node with `roslaunch info <node_name>`

4 ROS Topics

- Nodes communicate over *topics*
 - Nodes can *publish* or *subscribe* to a topic
 - Typically, 1 publisher and n subscribers
- Topic is a name for a stream of messages
- List active topics with `rostopic list`
- Subscribe and print the contents of a topic with `rostopic echo <topic_path>`
- Show information about a topic with `rostopic info <topic_path>`

5 ROS Messages

- Data structure defining the *type* of a topic
- Comprised of a nested structure of integers, floats, booleans, strings, etc. and arrays of objects
- Defined in **.msg* files
- See the type of a topic `rostopic type <topic_path>`
- Publish a message to a topic `rostopic pub <topic_path> <topic_type> <data_string>`

6 ROS Workspace Environment

- Setup students with workspaces on odroid
- `setup <name> <github_name> <github_email>`
- `mkdir ws; cd ws; mkdir src; catkin_make; source devel/setup.bash`
- check it worked with `echo $ROS_PACKAGE_PATH`

7 Catkin Build System

- The `src` directory contains source code. This is where you clone, create, edit code.
- The `build` directory contains the code built whenever you run `catkin_make`.
- The `devel` directory contains scripts to make `ros` scripts work with this workspace.
- Don't touch the `build` or `devel` directories. Just use the `src` directory.

8 ROS Packages

- ROS software is organized into packages, which contains source code, launch files, config files, message definitions, data and docs
- Packages can depend on other packages
- To create a package, go to the `src` directory and run `catkin_create_pkg <package_name> <dependencies>`.
- Packages have a `CMakeLists.txt` and a `package.xml`. We shouldn't need to touch the `package.xml`, unless we change what ROS dependencies we have.
- Go over `CMakeLists.txt`

9 rospy example

- run `catkin_create_pkg example1 rospy std_msgs`
- edit `scripts/node`

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

def run():
    pub = rospy.Publisher('test_wilson', String,
                          queue_size=10)
    rospy.init_node('test_wilson')
    rate = rospy.Rate(10)
    while not rospy.is_shutdown():
        test_str = 'hello world {}'.format(
            rospy.get_time())
        pub.publish(test_str)
        rate.sleep()

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

- edit `CMakeLists.txt`

```
catkin_install_python(PROGRAMS scripts/node
DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
```