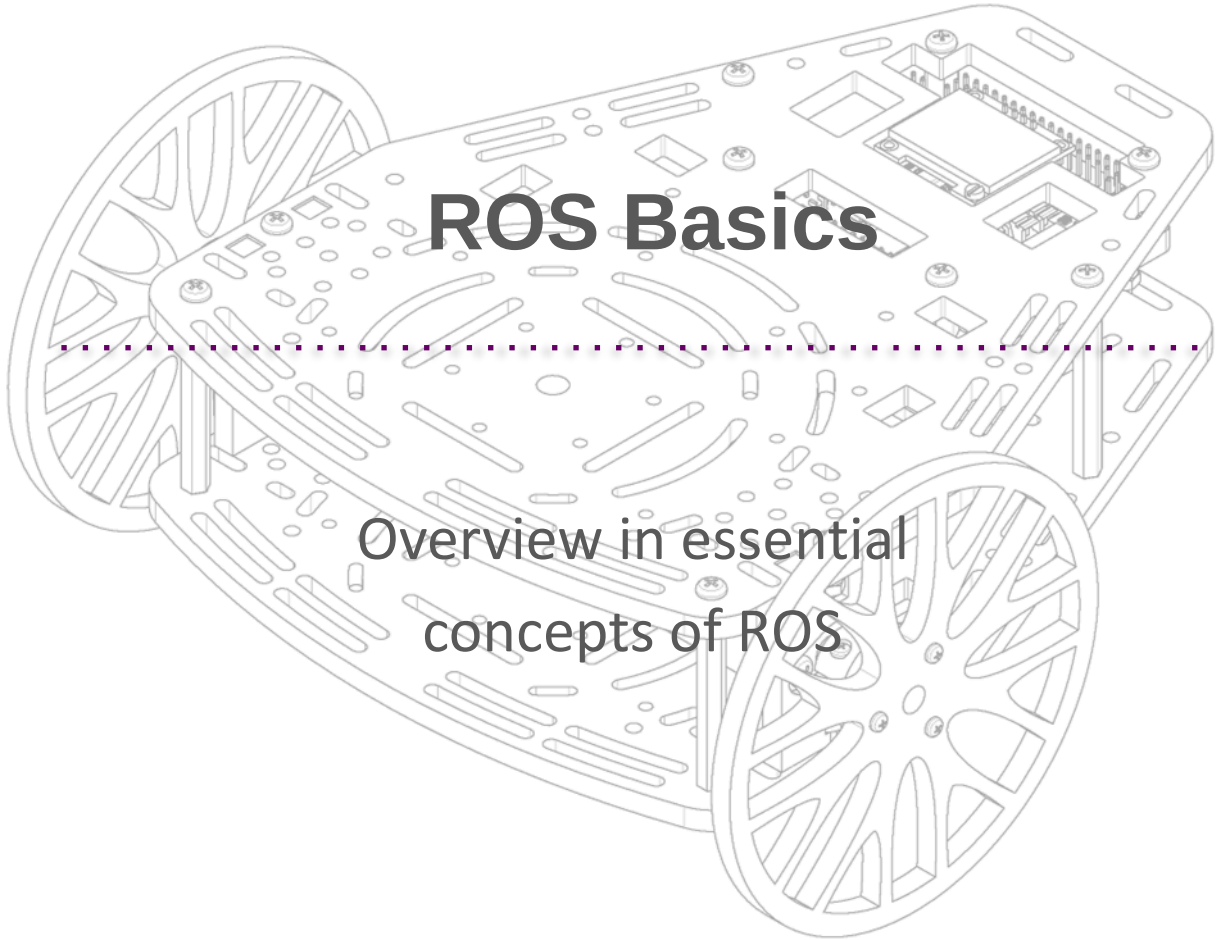


ROS Basics

Overview in essential concepts of ROS

Session 2: ROS Basics.

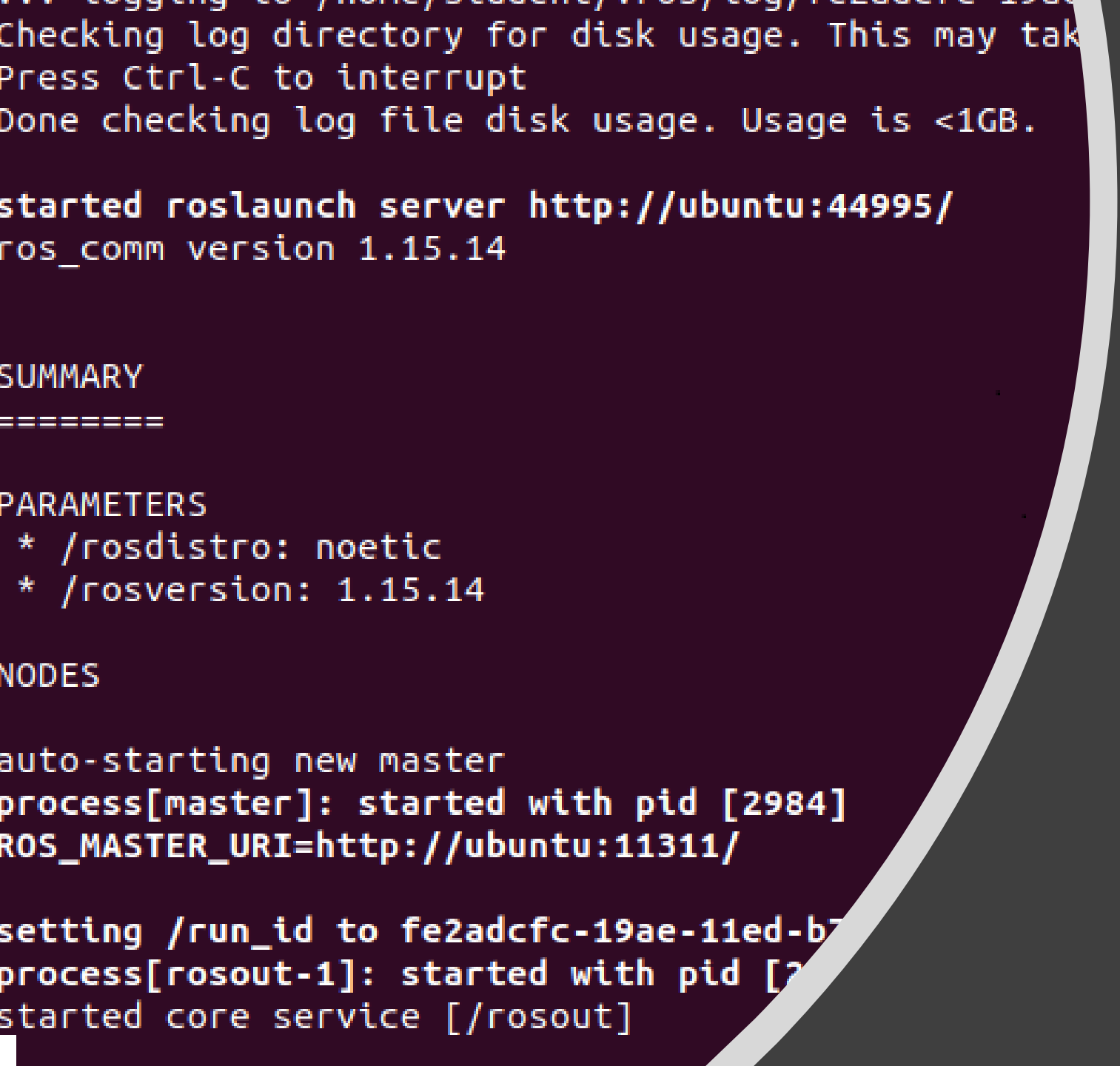
I. Basic concepts of ROS (35 min approx.) *Video*
   - What is ROS?
   - ROS features: nodes, topics, services, actions etc.
   - ROS Messages
   - ROS Publishers and Subscribers
   - Create a Catkin Space

III. Activity 1: Create a basic communication scheme (30 min)

III. ROS services and visualisation tools (25 hours approx.) *Video*
   - ROS services: comparison with topics, custom messages.
   - ROS tools Commands
   - ROS rqt

IV. Activity 2: Create a node that publishes a sine wave and examine the signal using rqt tools (35 min)

Requirements: Laptop, ROS preinstalled, Ubuntu preinstalled, Basic Knowledge of Python.

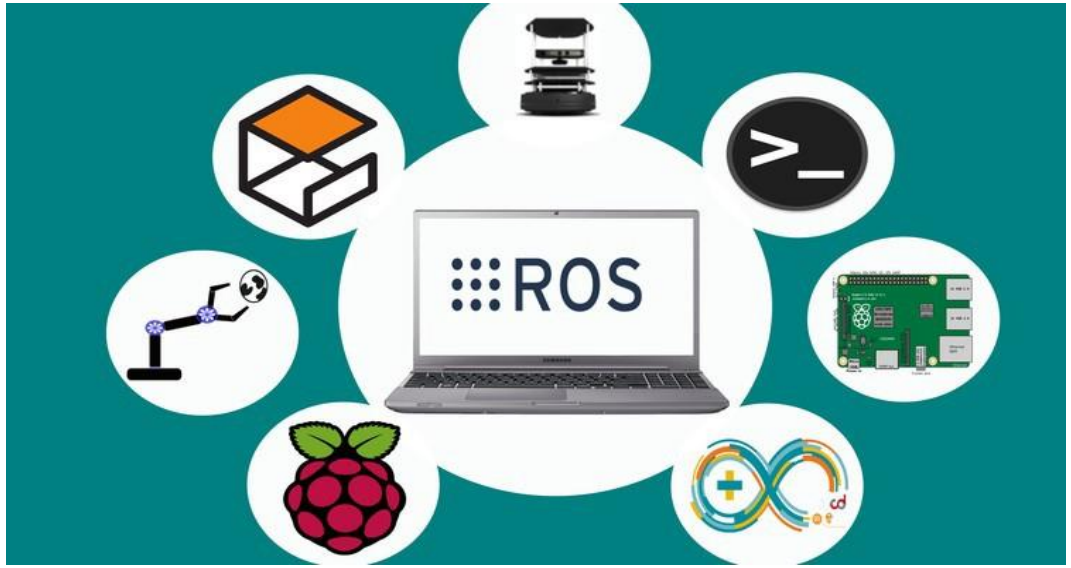


Tecnológico de Monterrey

NVIDIA

Basic concept of ROS

ROS basics
What is ROS ?

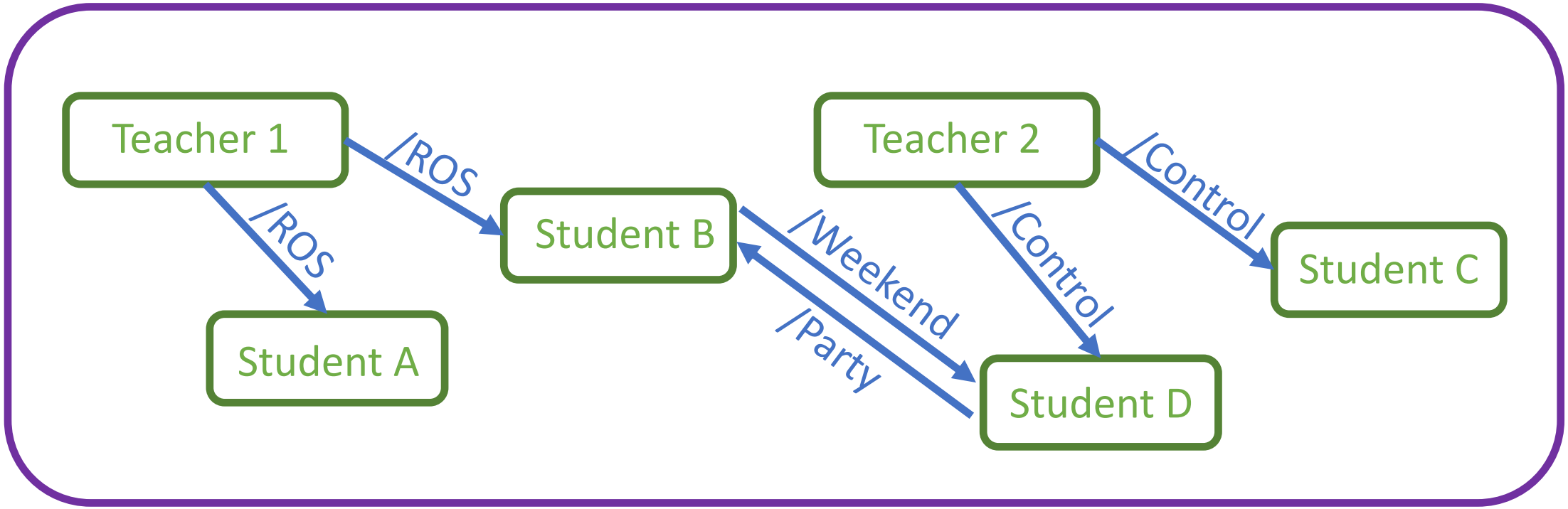


“The ROS is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.”

ROS basics

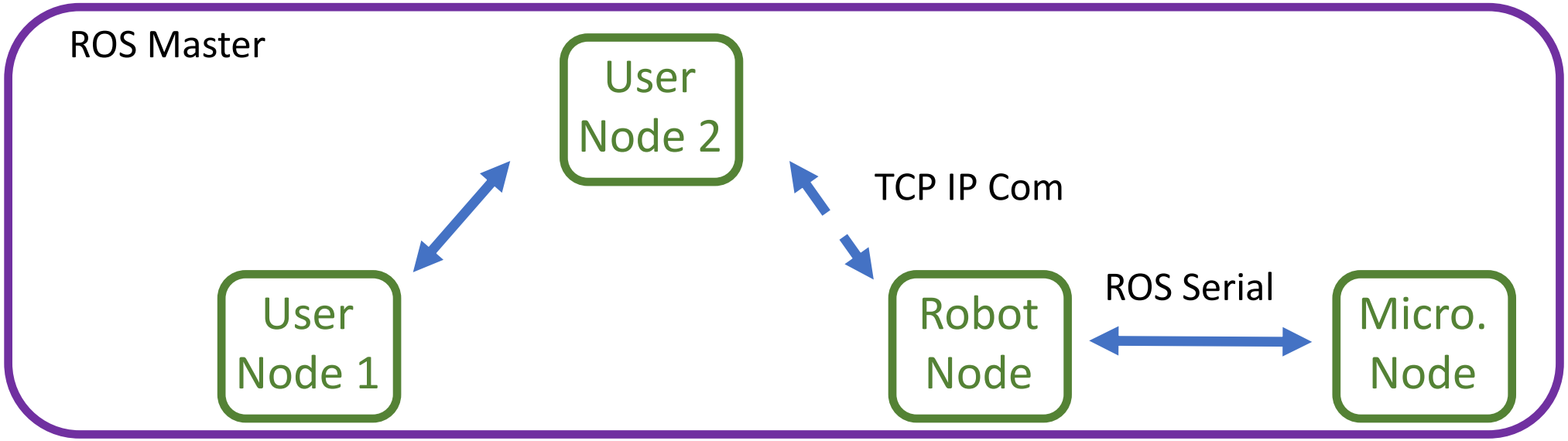
The conceptual idea behind ROS

University



ROS Architecture
ROS Master

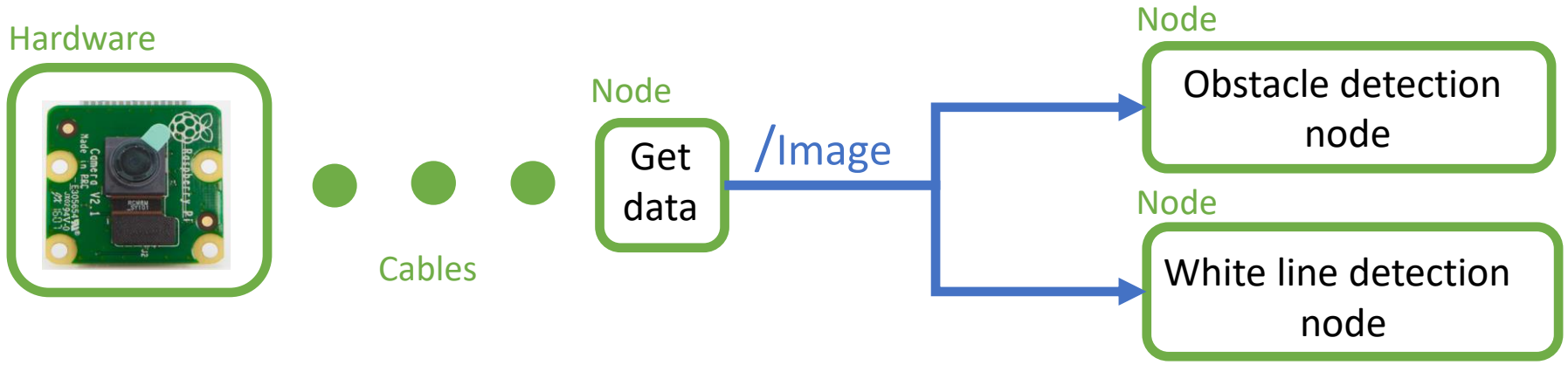
- Brain of ROS -> Process manager that enables the communication between different agents (devices) in the network.
- Allows communication between different computers or robots using a Server-Client architecture



Run using the "roscore" command. This requires its own terminal window that must be running all the time.

ROS Architecture
Nodes and Topics

- Piece of software that acts as an element in the network.
- It is in charge of executing a part of the code and can be programmed in Cpp, Python or Lisp.
- Each topic has a unique msg type.



Use “rostopic” command to send or read information in the topic

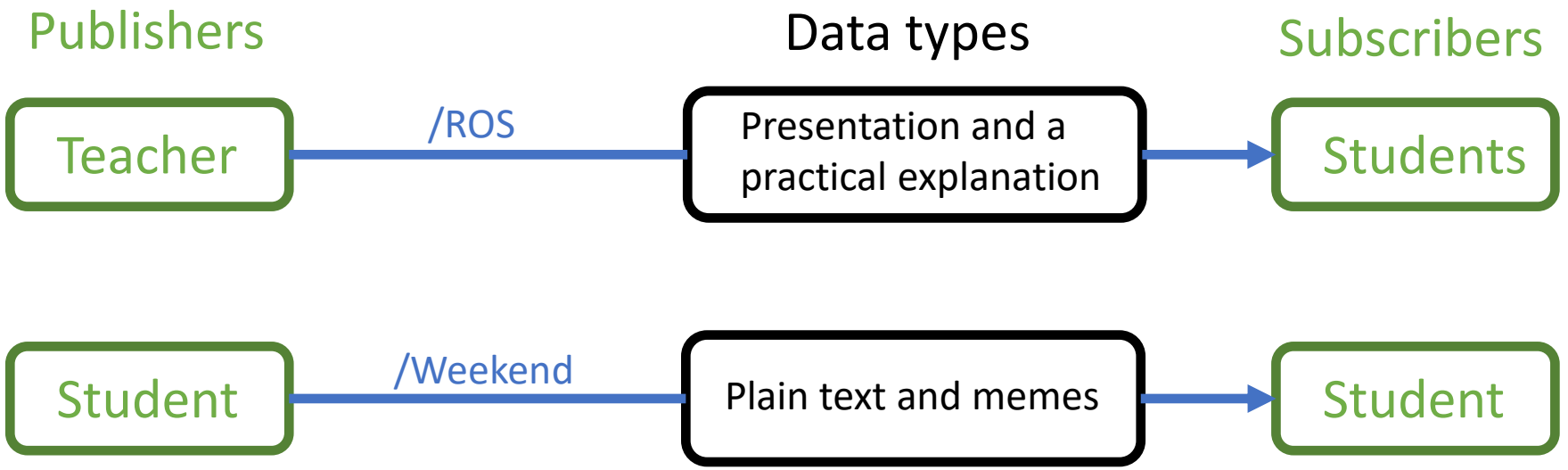
Use “rosrun” to execute a node. Also, use the " rosnode " command to know the list of available nodes or your node's information.

ROS basics

ROS Messages

What is the information delivered?

- Any class has a specific format, which both the teacher and the student know and is expected to be followed.
- Between two friends, you are not expecting a PowerPoint presentation but some plain text.



ROS Architecture
messages

std_msgs/Float32

float 32 data

sensor_msgs/Image

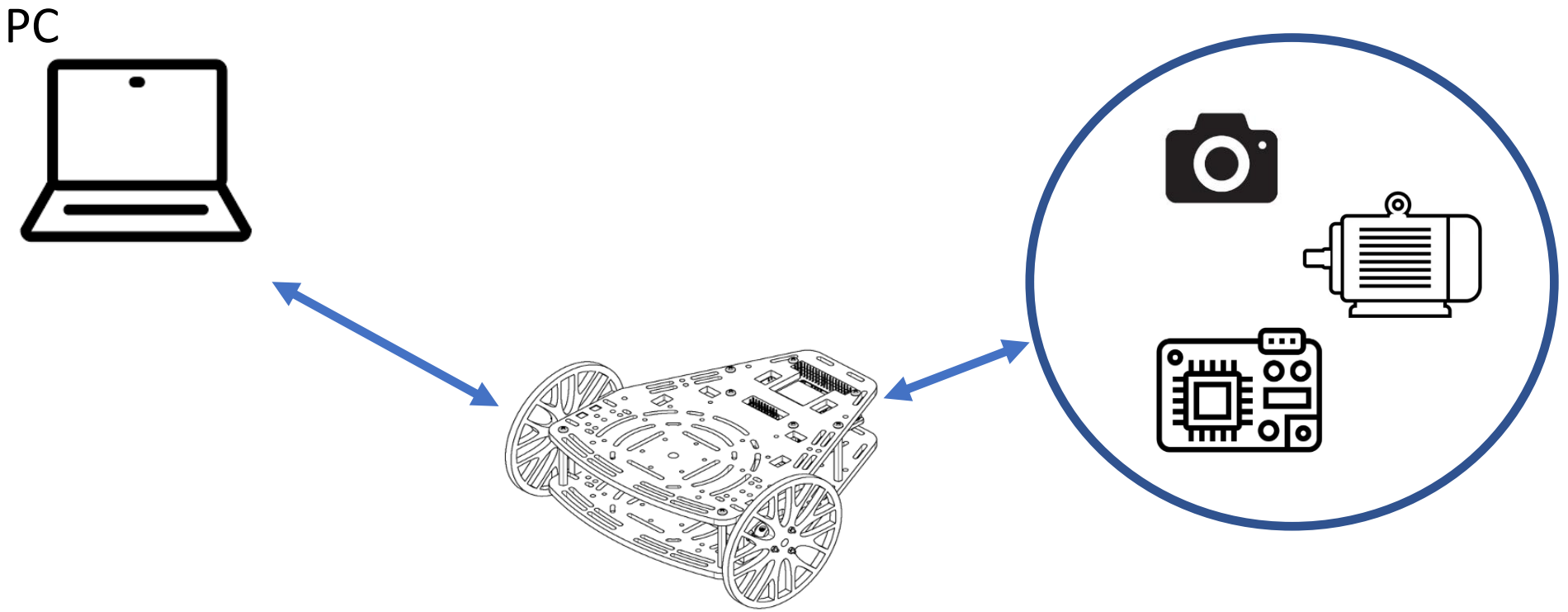
std_msgs/Header header
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data

geometry_msgs/PoseStamped

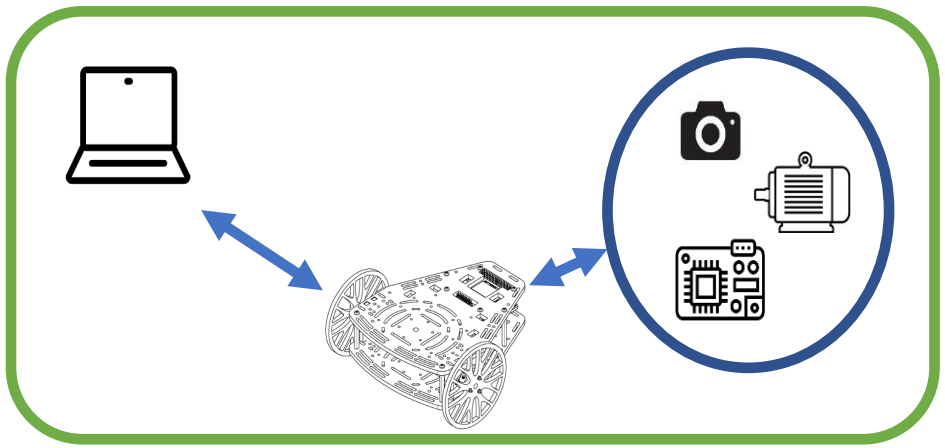
std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
geometry_msgs/Pose pose
    geometry_msgs/Point position
        float64 x
        float64 y
        float64 z
    geometry_msgs/Quaternion orientation
        float64 x
        float64 y
        float64 z
        float64 w

ROS Architecture

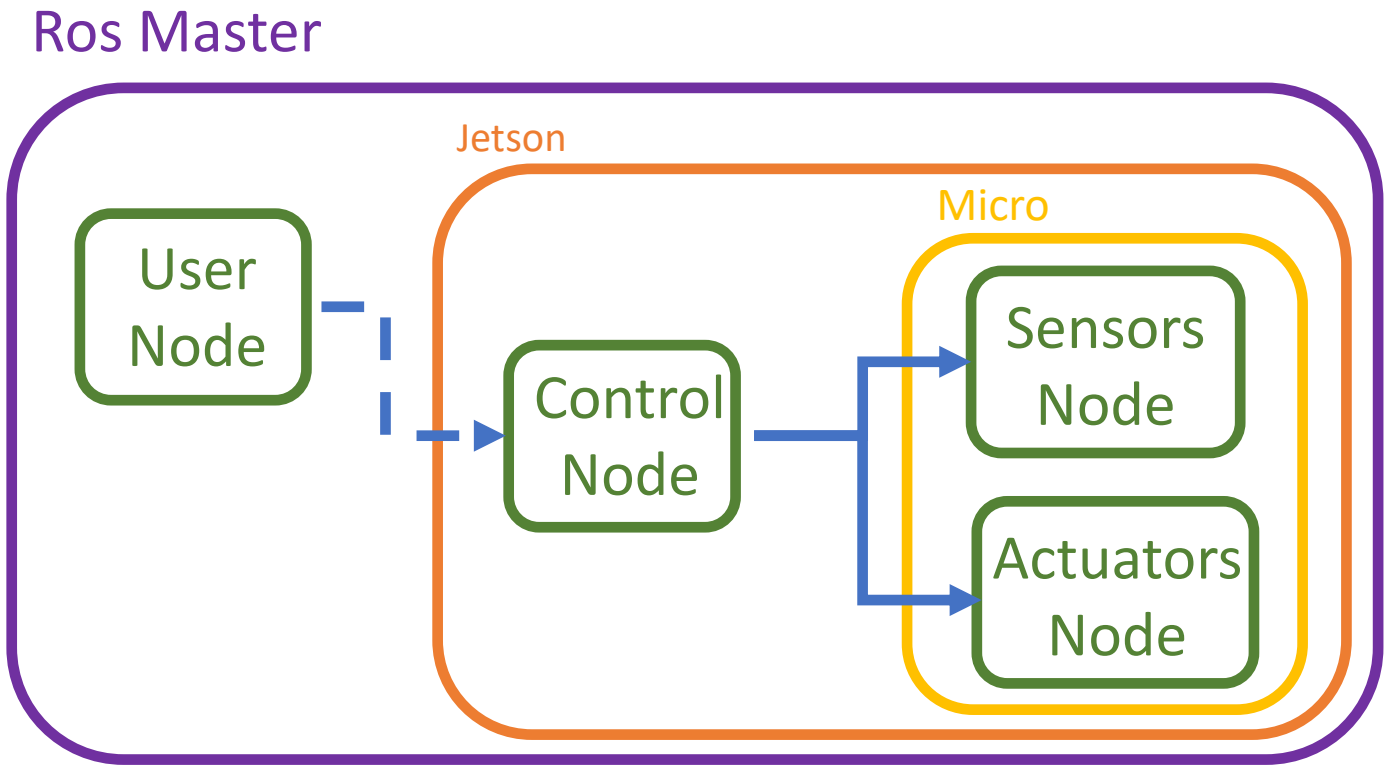
A practical example



ROS Architecture
Overview



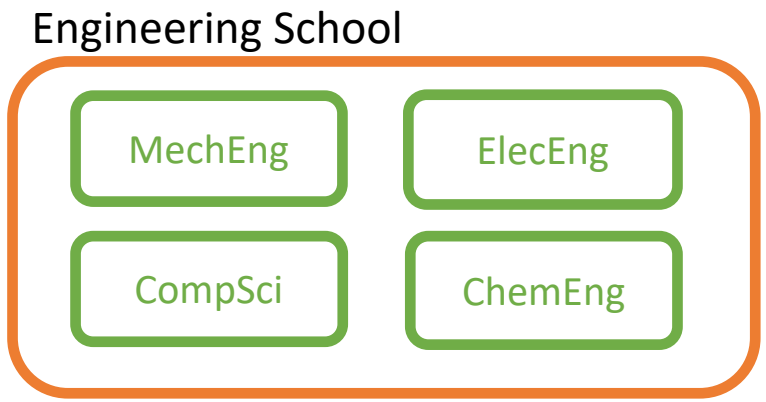
Physical System

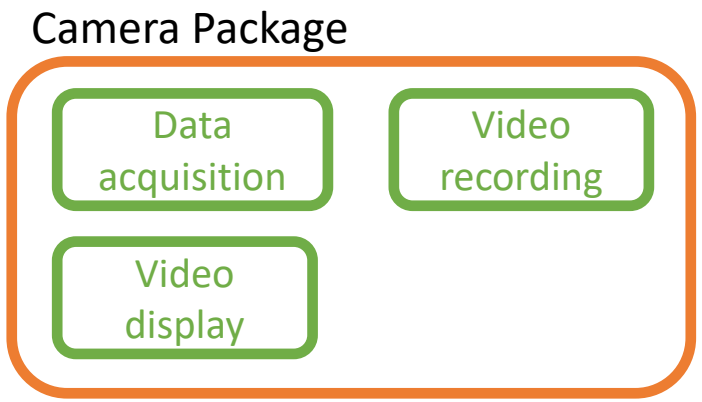


ROS Implementation

ROS Architecture
ROS Packages

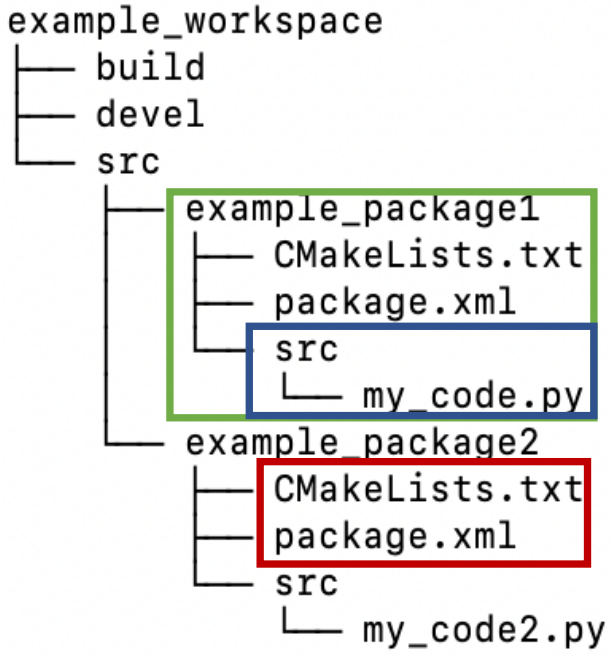
- ROS Packages are a way of organising code pieces related to each other.
- The same way teachers are gathered within the Engineering school nodes are gathered in ROS packages





ROS Tools
ROS File structure

- ROS projects are organised using workspaces, which contain a collection of grouped folders called packages.
- Instructions for the compiler need to be allocated in .cmake and package files



- Package files are exportable between projects.
- Configuration files used to establish code dependencies
- Code that we will execute
- Catkin_make will generate ‘src’ and ‘devel’ for you when compiling the workspace

ROS Tools

ROS Compilation tools and other useful commands

- ROS requires compiling each package, generating dependencies related to other packages, external libraries or custom messages, services and actions.
- The preferred compilation tool is known as `catkin` and uses two separated files, `package.xml` and `CMakeLists.txt`. The command `catkin_make` is used to start the compilation.
- The command to create packages is:

```
catkin_create_pkg [name] [list of dependencies]
```

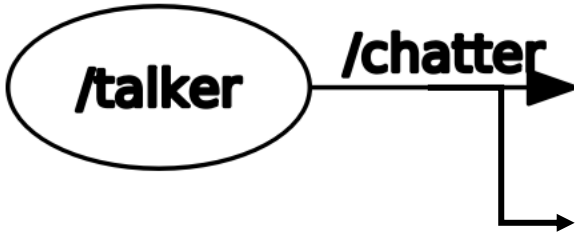
  Which generates an empty package and templates of both CMakeLists and package files.
- More information about the syntaxis of these files can be found at http://wiki.ros.org/ROS/Tutorials/CreatingPackage
- Also, information on how to create a catkin workspace can be found at http://wiki.ros.org/catkin/Tutorials/create_a_workspace

ROS Code

Talker and listener framework

Simplest communication scheme.

ROS Code
Talker

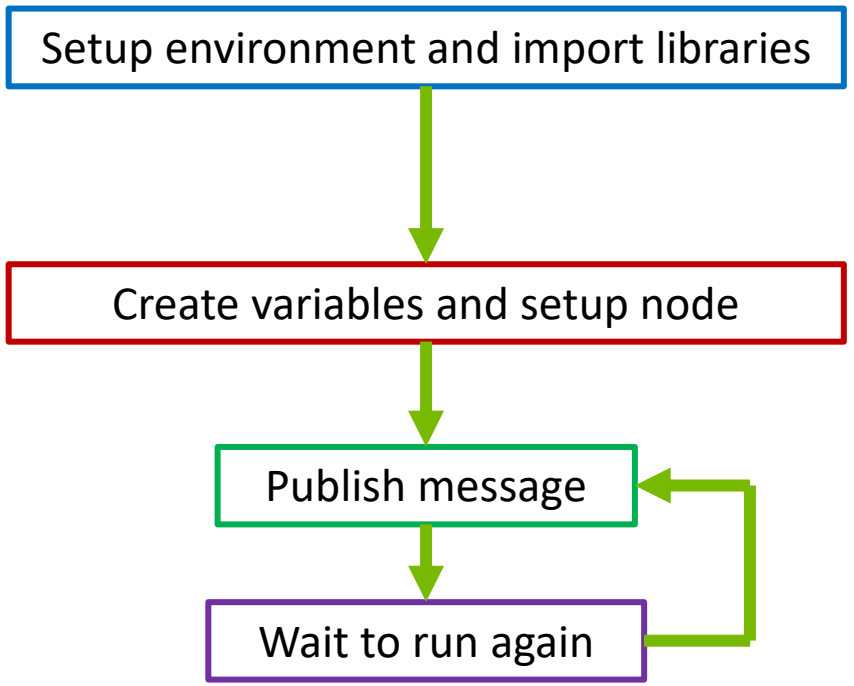



```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
```

```python
if __name__ == '__main__':
    pub = rospy.Publisher("chatter", String, queue_size=10)
    rospy.init_node("talker")
    rate = rospy.Rate(10)
```

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

```python
        rate.sleep()
```



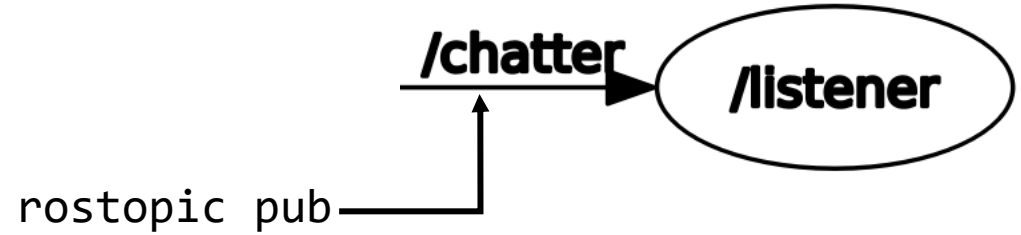
Terminal Commands

```
roscore
rosrun basic_comms talker.py
rostopic echo /chatter
```

Remember to make the node executable

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

ROS Code
Listener




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

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

if __name__ == '__main__':
    rospy.init_node('listener')
    rospy.Subscriber("chatter", String, callback)

    rospy.spin()
```

Setup environment and import libraries

↓

Define Callback

↓

Create variables and initialise node

↓

Continuously check for new messages

Remember to make the node executable

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

Terminal Commands

```
roscore
rosrun basic_comms listener.py
rostopic pub /chatter <tab complete>
```

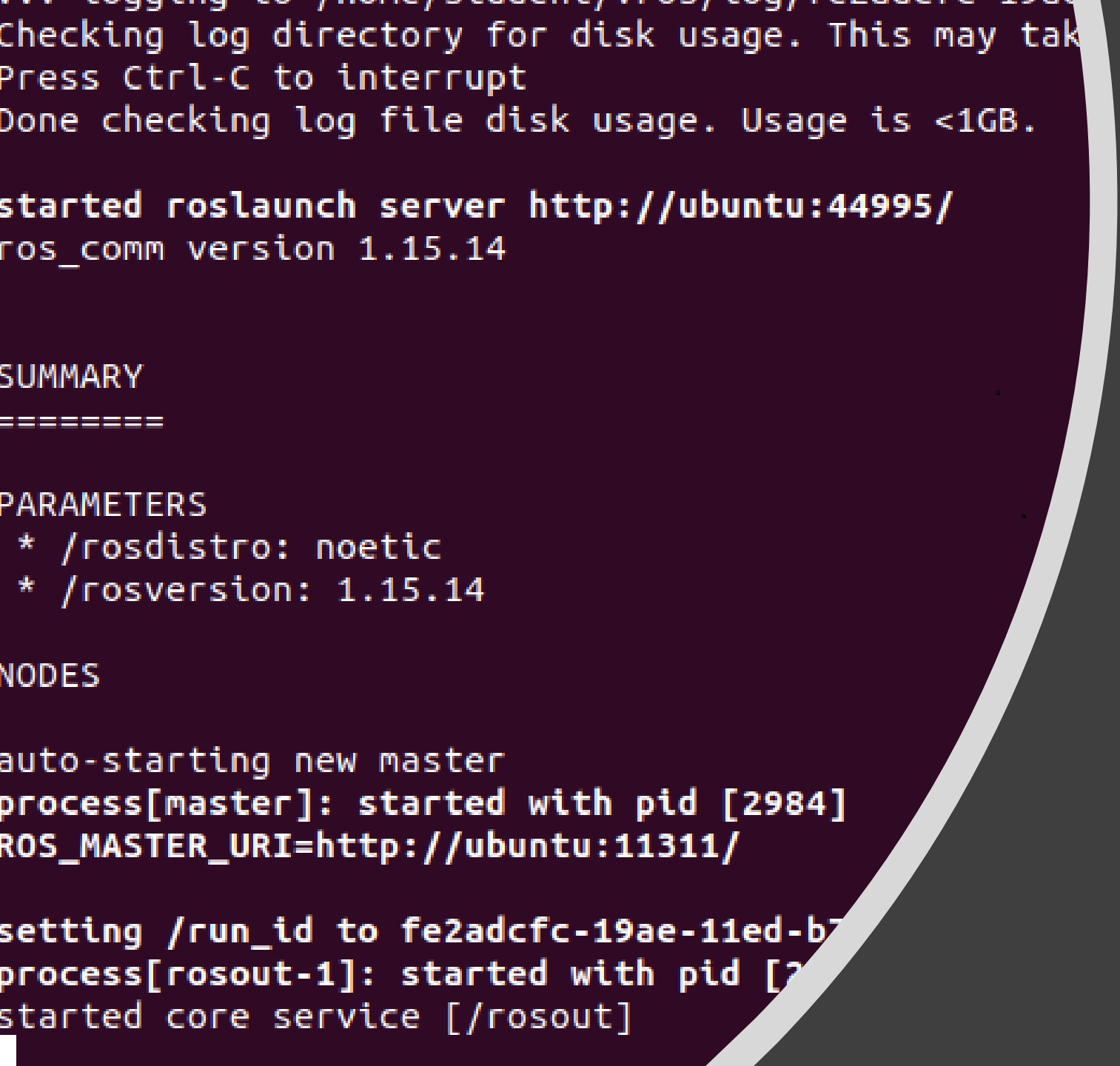
ROS Tools

ROS Terminal tools summary

- roscore
  - Must be executed before working with ROS
  - Handles the correct functionality of the network
- rostopic
  - Contains useful tools related to topics
  - *rostopic echo [topic name]*
  - *rostopic list*
  - *rostopic pub [topic name] [tab + input]*
- rosrun
  - Executes a given node
  - *rosrun [package] [node name]*
- rosnode
  - Contains useful tools related to nodes
  - *rosnode list*
  - *rostopic info [node]*

Activity 1: Talker and listener.

- Create your package called “basic_comms” **(the dependencies used are rospy and std_msg)**
- Implement the code in Python. (This must be placed in the “src” folder of your package)
- Modify your Cmakelist.txt and the package.xml file to include your code. You can find help here.
- Use some of the command tools to verify the correct functioning of the system. **Hint: Go back to slides about the talker and the listener.**



Tecnológico de Monterrey

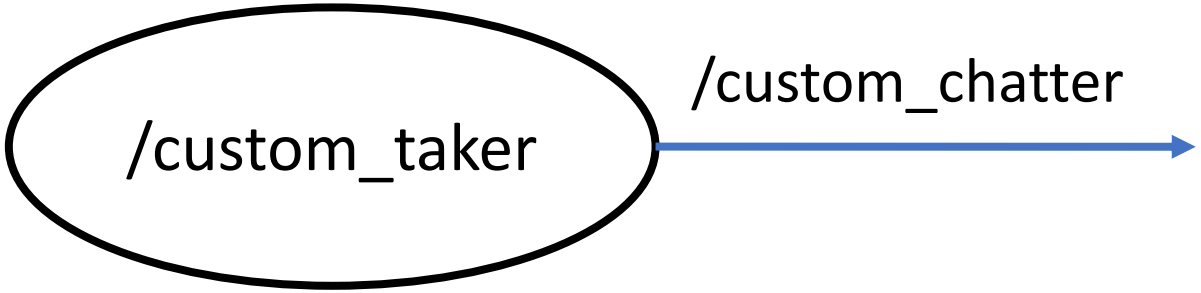
NVIDIA

Custom messages, ROS Services and additional tools

ROS Tools
ROS Custom message

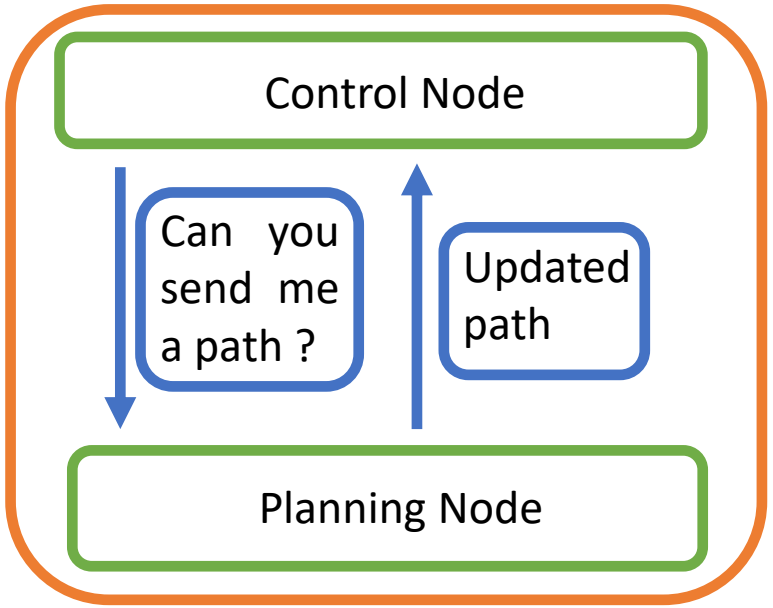
- Custom messages to personalise your message
- Custom messages are created and linked to the package
- Add folder called “msg”, to your package
- Inside the folder “msg” create a file.msg, this file should include parameters inside the custom message.
- Edit CmakeList.txt and Package.xml, to add your node and to create your custom message.

Custom msg /Randomised
int32 counter
float32 rand_num

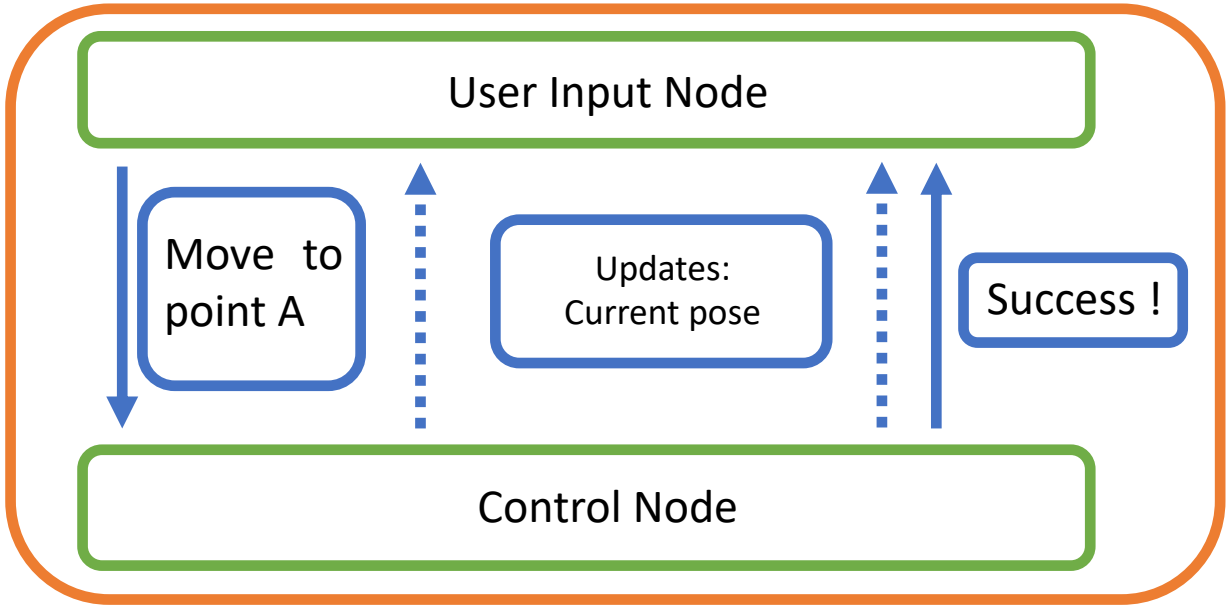


ROS Architecture
Services and actions

- While topics are one-way communication, services and actions allow feedback and prevent unnecessary load in the network.
- Services follow a question-answer structure, while actions are designed to execute a task and give feedback during the process.



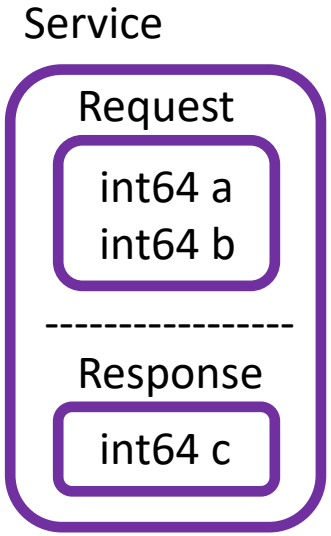
Service

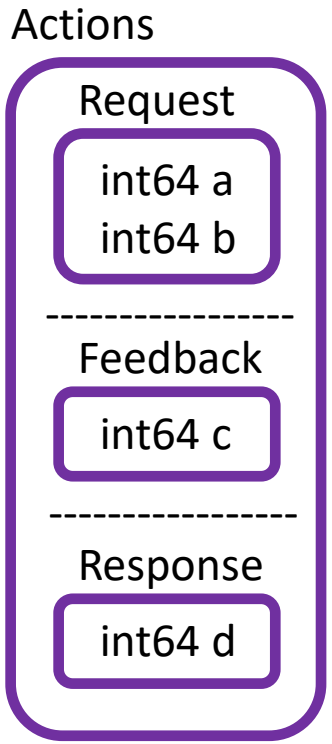


Action

ROS Architecture
Services and actions

- When defining either an action or a service, we need to establish each member involved in the communication. Usually, both are project dependent, so users define their own structures.

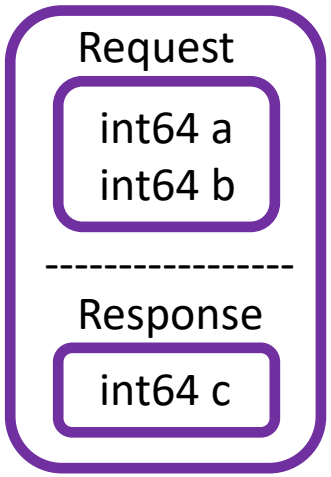




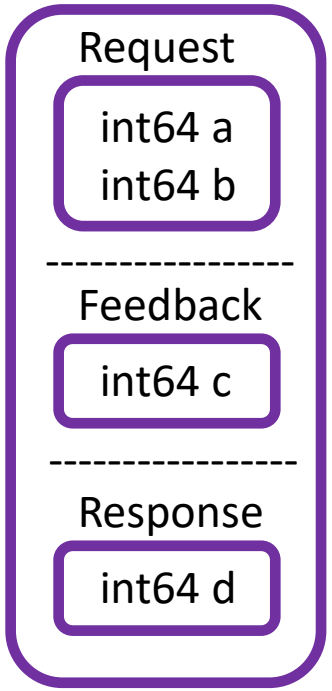
ROS Architecture
Services and actions

- When defining either an action or a service, we need to establish each member involved in the communication. Usually, both are project dependent, so users define their own structures.

Service



Actions



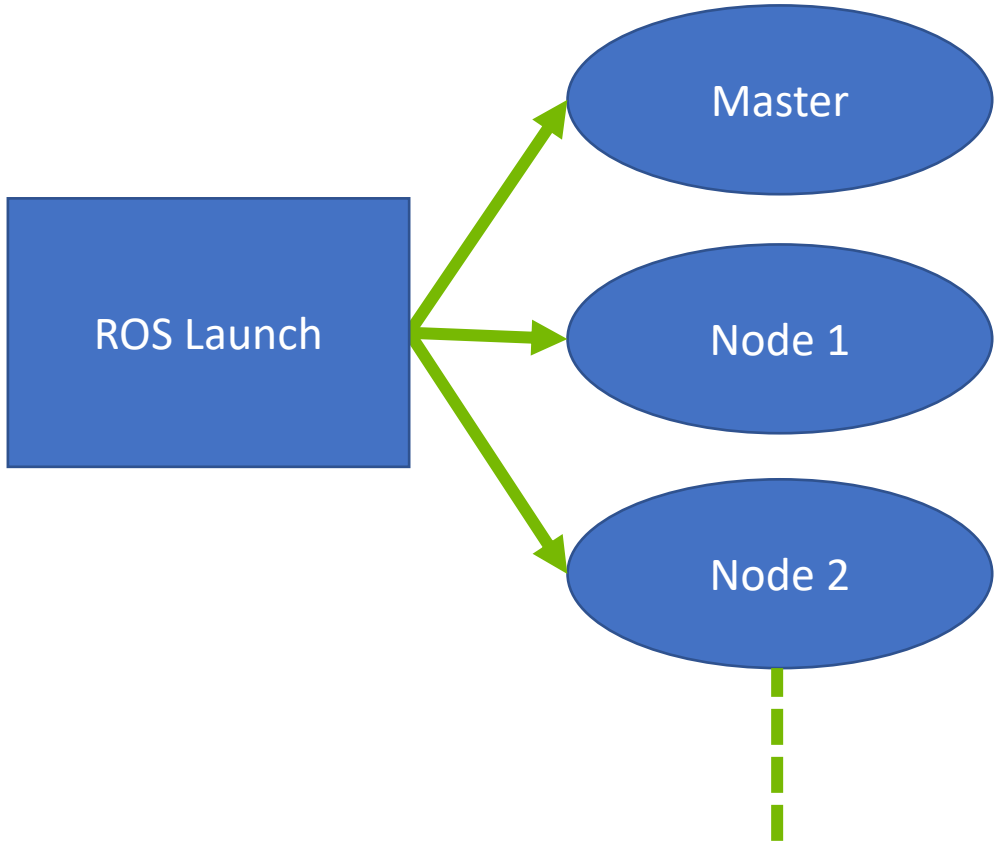
Find additional information here

ROS Tools
ROS Launch

- Launch files are sets of commands written in XML that allow executing various scripts simultaneously.
- The general syntaxis is the following

```
<?xml version="1.0"?>
<launch>
   [Body of the launchfile]
</launch>
```

- It allows you to run any object used within the ROS architecture. It has a wide variety of tools to parametrise the launch file to adapt to your project's requirements.
- An extensive documentation can be found at http://wiki.ros.org/roslaunch



ROS Tools
ROS Launch code examples

- Running a node

```
<node name=“listener” pkg=“basic_comms” type=“listener.py” output=“screen”/>
```

- Running a ROS tool

```
<node pkg="rqt_plot" type="rqt_plot" output="screen" name="rqt_plot"/>
```

- Running another file or launch file

```
<include file="$(dirname)/other.launch" />
```

- Set parameters

```
<param name="publish_frequency" type="double" value="10.0" />
```

- Pass args to the launch file

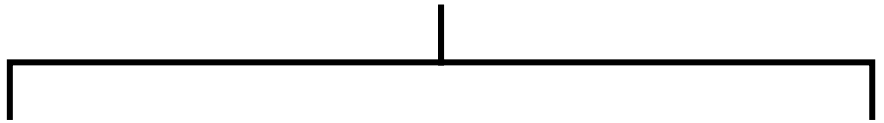
```
<arg name=”camera_id" value=”cam_3" />
```

ROS Tools
ROS Terminal tools summary

- roslaunch
  - Allows launching multiple nodes with a single command by calling .launch files
  - *roslaunch [package] [.launch]*

- rqt

ROS visualization tool that provides graphical information of the system status



  - rqt
  - rqt_graph
  - rqt_plot
  - rqt_image_view

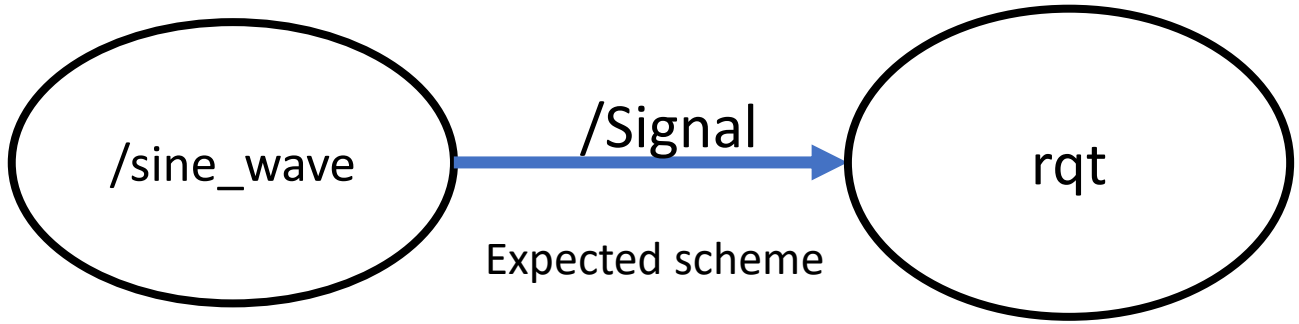
Activity 2
Implement a sine wave generator

- Download from GitHub the activity folder, add the package “sine_wave_gen” in your workspace.
- Use the node template “sine_wave.py” to code the node in Python. **You must use a custom message for the signal.**
- Edit the message file in the msg folder of your template**.** Launch both the sine wave node and rqt. Use the following command:

roslaunch sine_wave_gen sine_wave_gen.launch

- If everything goes well, you will be able to see your signal in rqt.

Custom message /Signal
Float32 time_x
Float32 signal_y



. **NOTE: Use rostopic to debug your code.**
. **NOTE 2: The files Cmakelist.xml, and package.xml are ready to used, you do not need to modify them**
NOTE 3: Find help with custom messages here, here and here.