Introduction to ROS summary

A. Angelou, I. Karakostas, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 1.2

Artificial Intelligence & Information Analysis Lab

What is ROS?

- ROS stands for “Robotic Operating System”
- It’s not an operating system, but a development tool
- Runs through Linux
- Is Open Source
- Supports C++ and Python programming languages

ROS Applications

- ROS is used:
  - For research purposes
  - In Research and Development (R&D) Departments in Industry
  - By individuals for personal projects

- ROS can be used in a wide range of applications such as:
  - Autonomous Driving
  - Controlling Robotic Arms
  - Drones
  - Object Detection/Tracking
  - Gesture Recognition

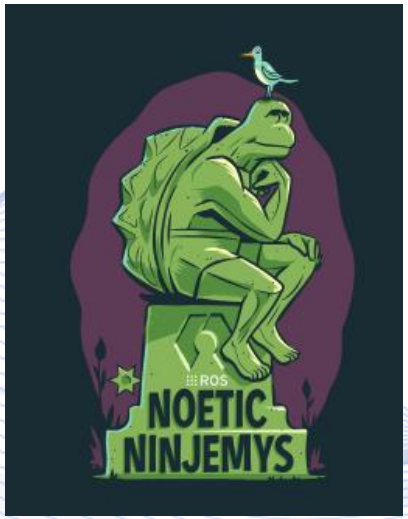
IRB 120 Robot [ABB]

BlueROV2 [BLRV]

ROS Distributions

The most stable and recent ROS Distributions are:

- ROS Melodic Morenia (Ubuntu 18.04 - Bionic Beaver)
- ROS Noetic Ninjemys (Ubuntu 20.04 - Focal)



http://wiki.ros.org/Distributions

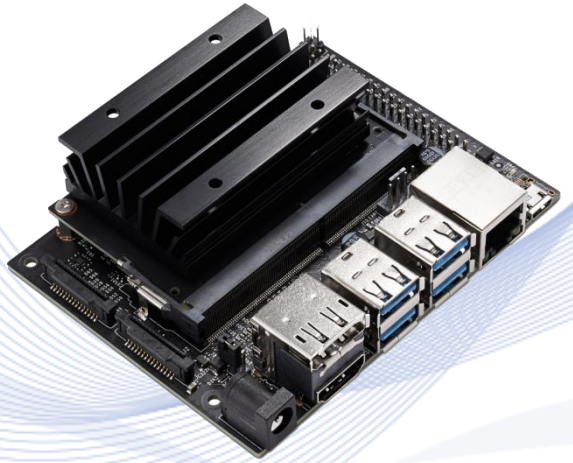
ROS Hardware

For ROS application can be used a variety of computer boards:

- Raspbery Pi (Raspberry Pi 4 B)
- PC motherboards (Ashrock X570 Extreme4)
- Embedded motherboards (Nvidia Jetson Nano)

Raspberry Pi 4
[RASP]

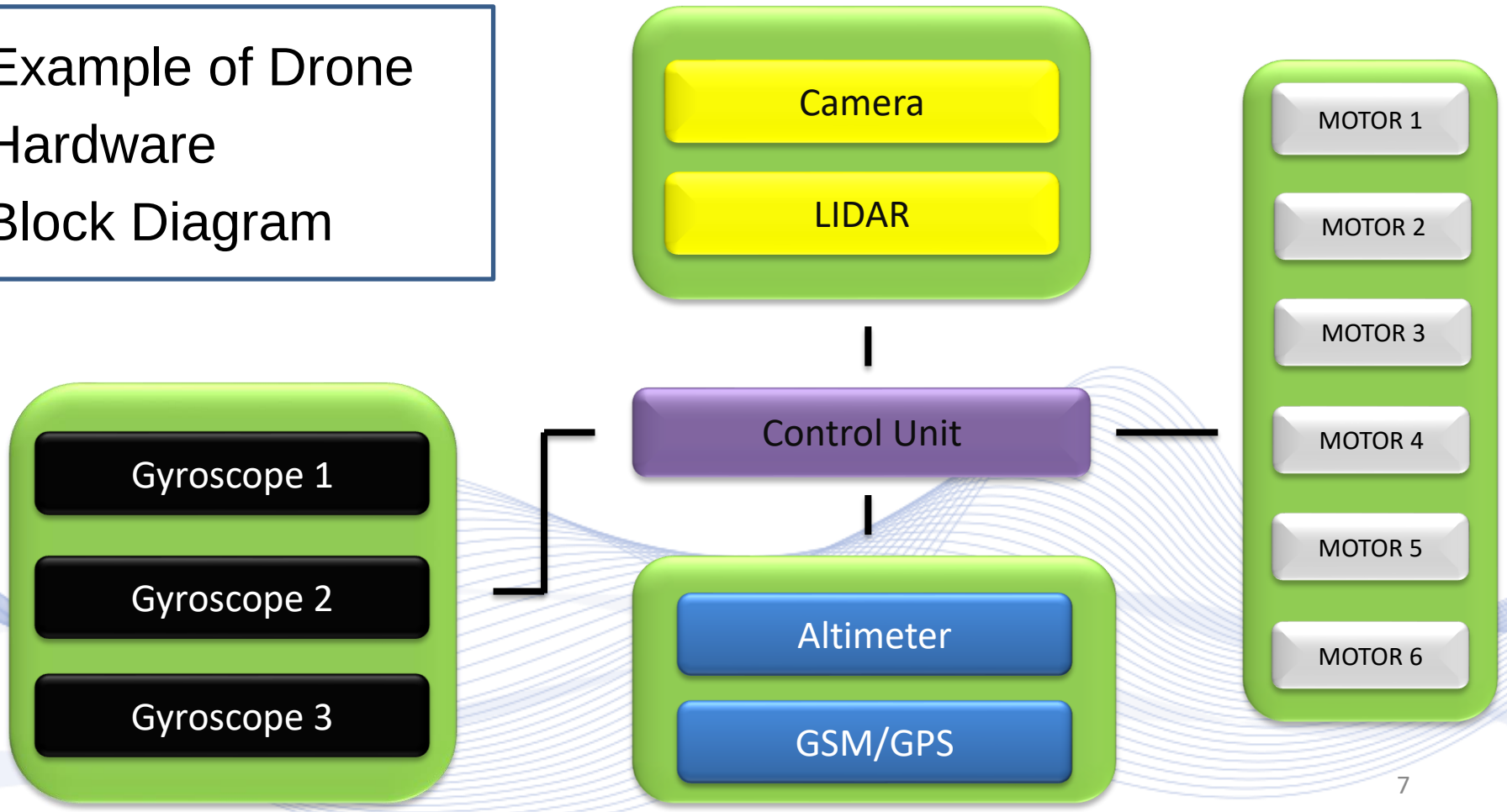
ASHROCK PC Motherboard
[ASHR]



NVIDIA Jetson Nano
[NVID]

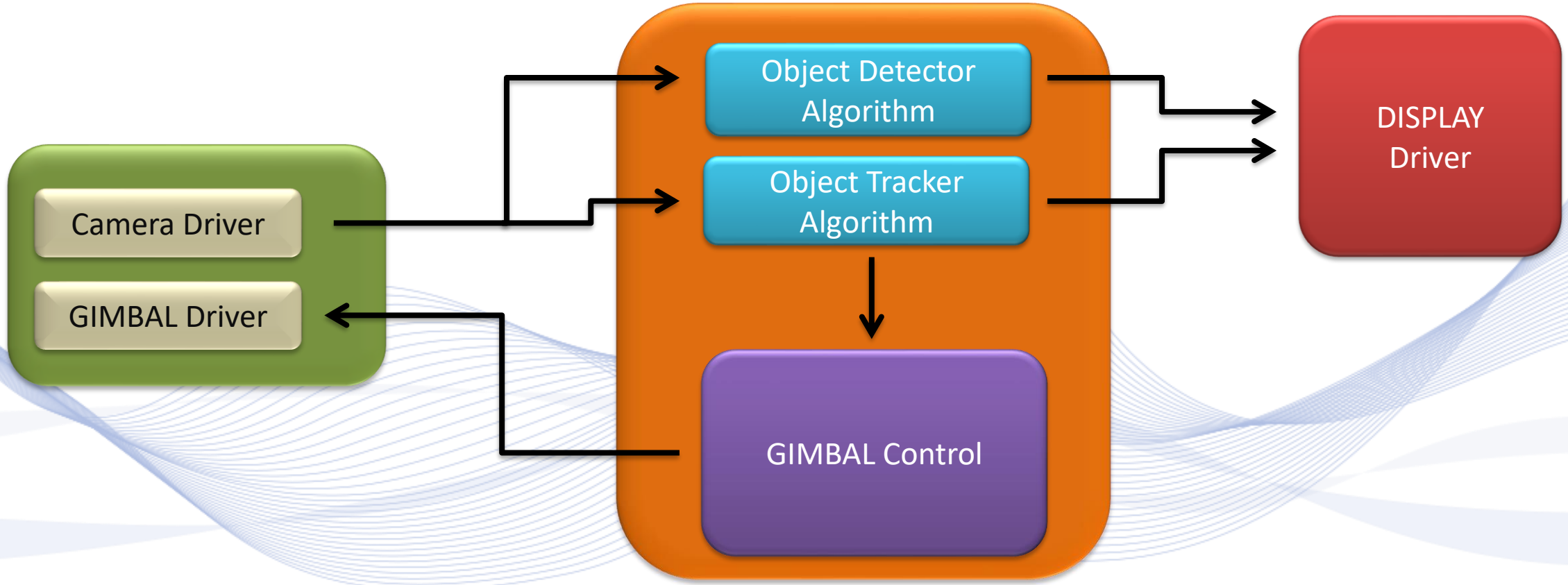
Robot Hardware Architecture

Example of Drone Hardware Block Diagram



Robot Software Architecture

Example of Drone Software Block Diagram



ROS communication

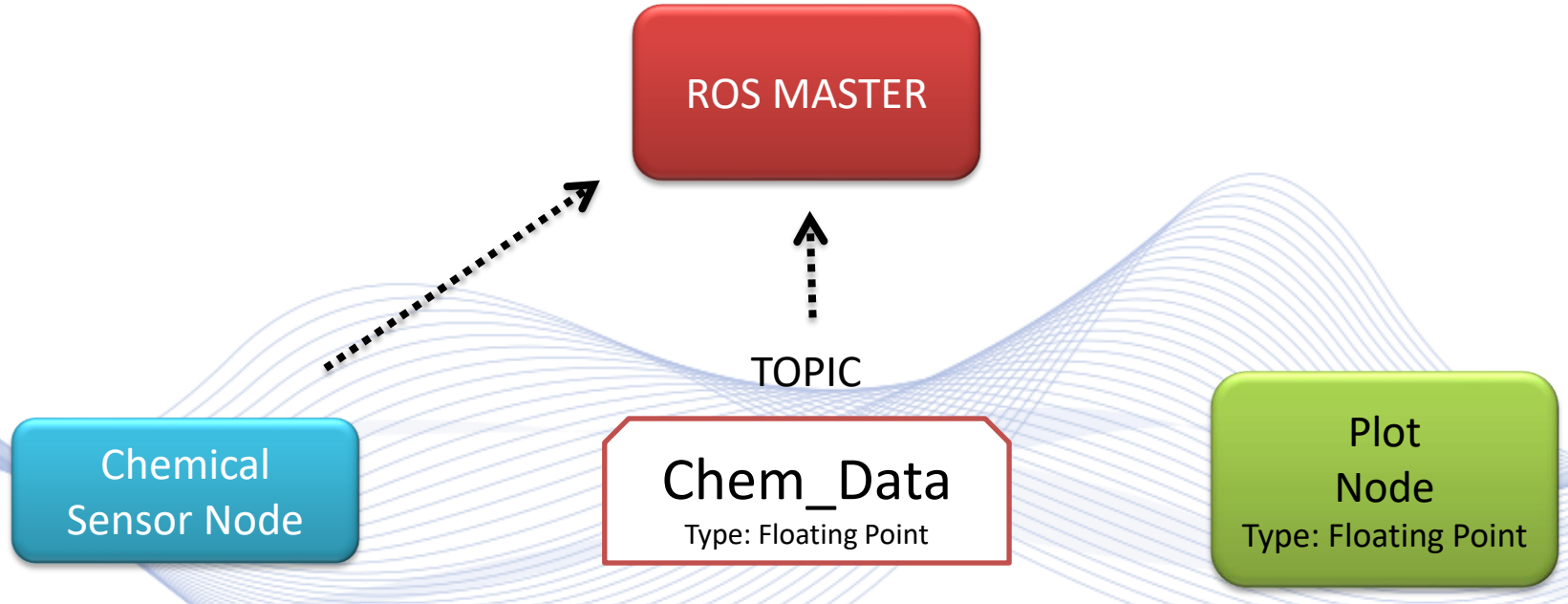
- **Topics**: Used for sending or reading messages of specific types.
- **Services**: Used for synchronous client/server communication. (e.g. change a setting, trigger a task – start detection/tracking).
- **Actions**: They are based on topics and provide an asynchronous client/server architecture. The client can send a request that takes a long time and can asynchronously monitor the state of the server.

ROS Master

- The ROS Master is the coordinator of the communication between nodes.
- All Nodes, Topics Services are registered to ROS Master.
- When a Node wants to sent a message to a Topic or exchange messages with another Node, ROS Master provides a way to the Nodes to locate each other.
- After the Nodes identify each other, they are communicating

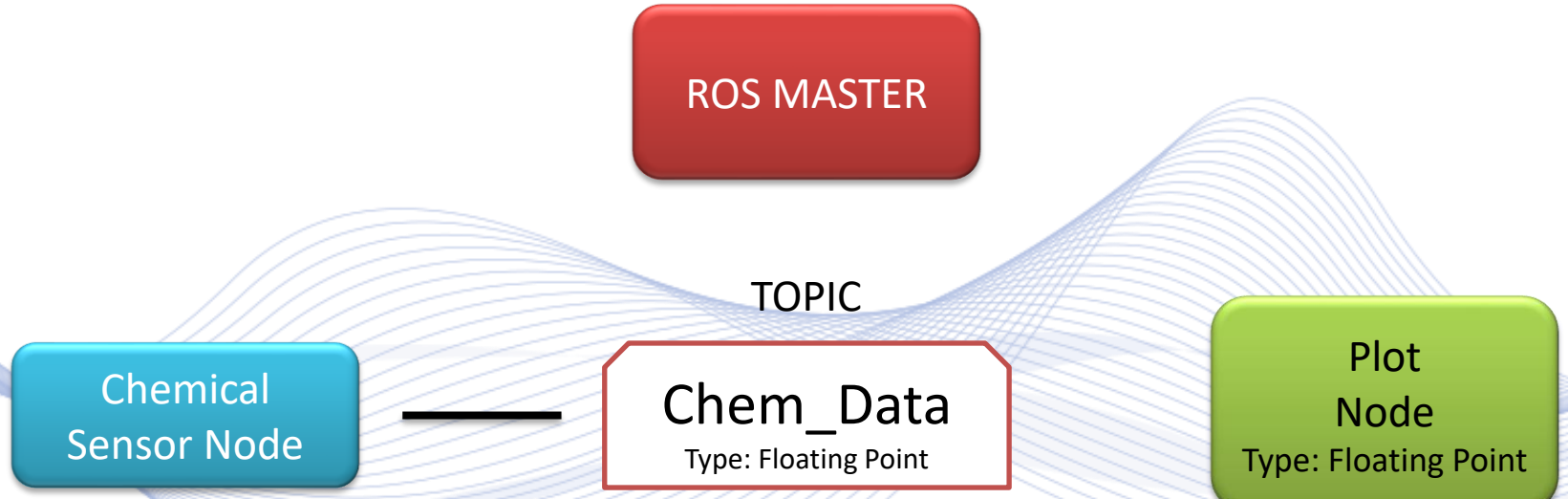
ROS Master

- When a Node wants to publish a message to a Topic, the Publisher Node notify ROS Master to send data to the Topic.



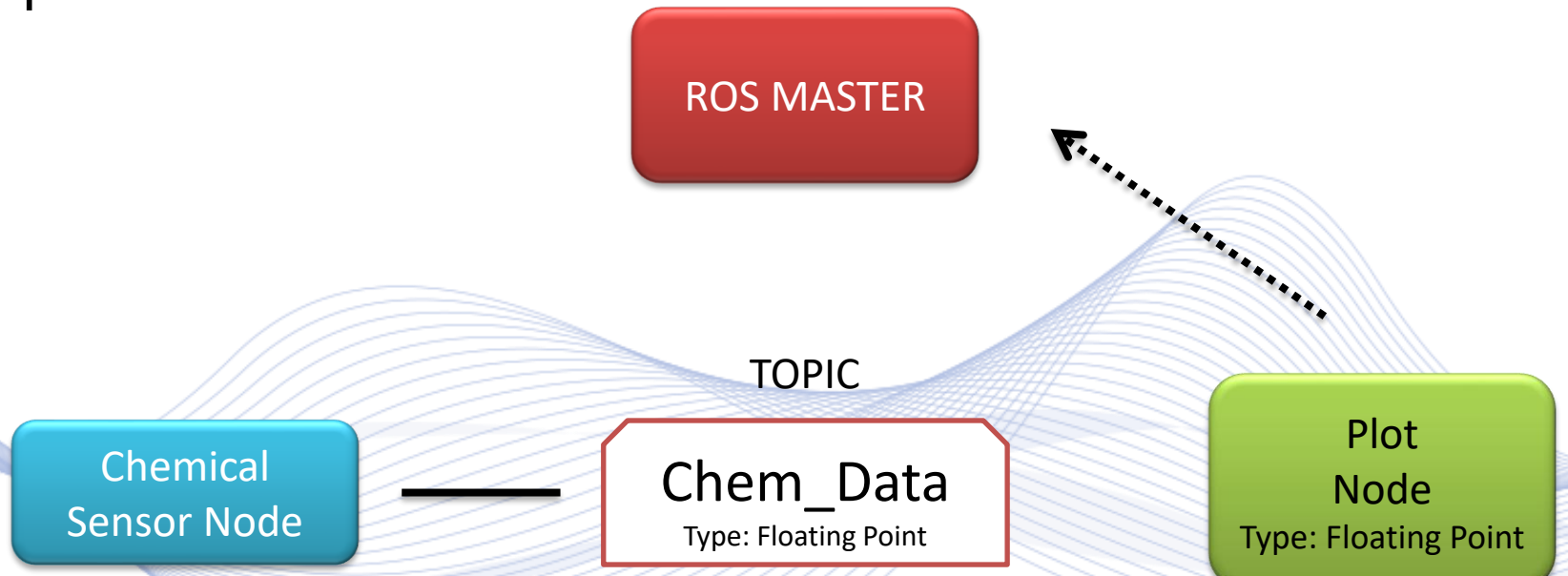
ROS Master

- After the notification, the Publisher Node establishes connection the Topic. At this point, the publisher doesn't send any message to the Topic unless a Subscriber Node notify ROS Master.



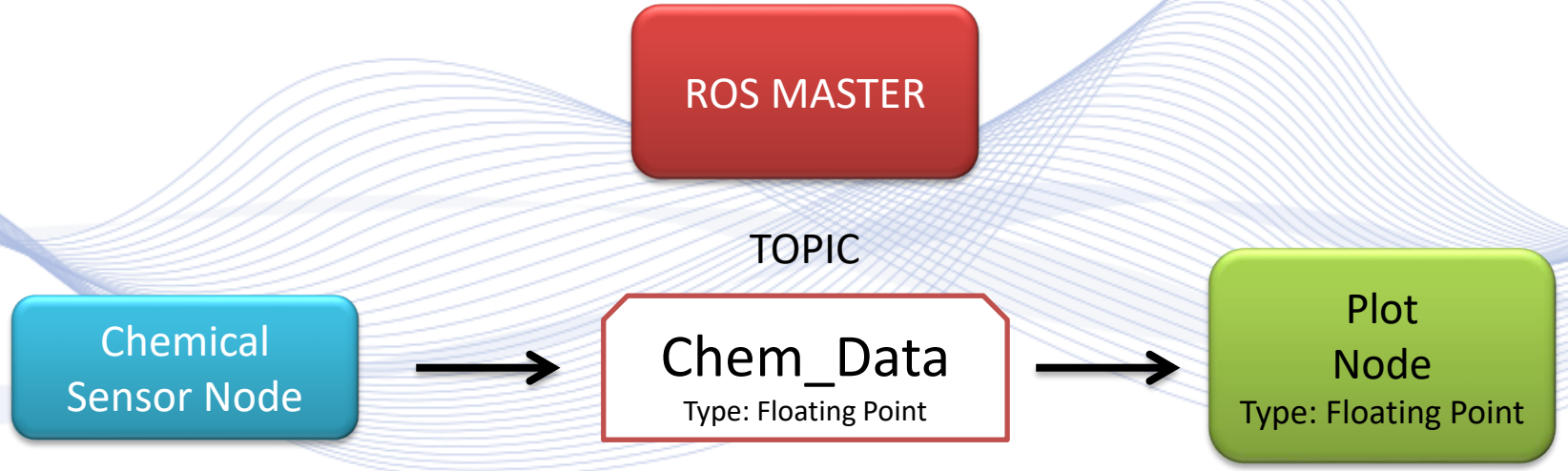
ROS Master

- When an **Subsciber Node** wants to subscibe to a message from a Topic, the **Subsciber Node** notifies ROS Master to connect to the Topic.



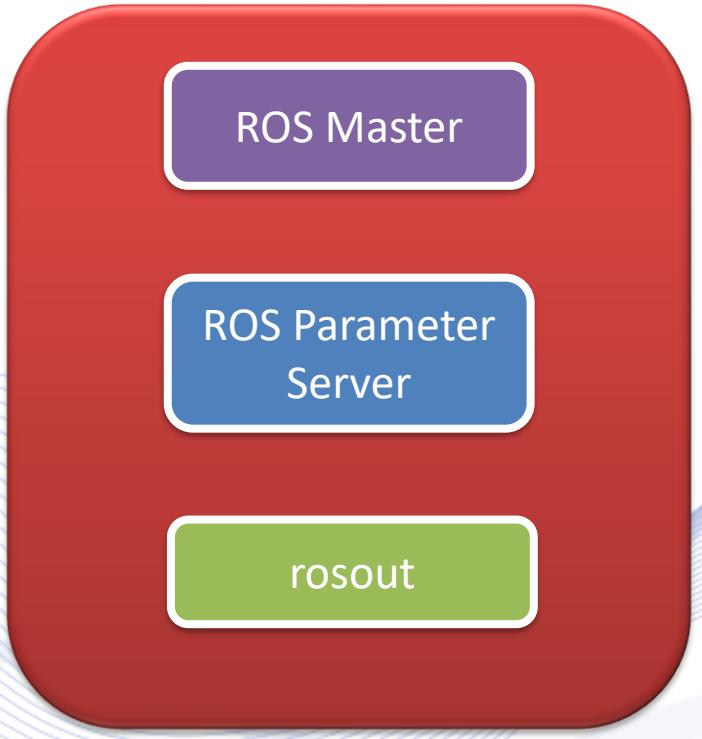
ROS Master

- After the notification, the Subsciber Node connects to the Topic.
- At this point the Publisher Node publishes the data to the Topic and the Subsciber, subscibes to the Topic.
- The data is transmitted from the Publisher Node to the Subsciber Node through the Topic.



ROS Core

- ROS core is a collection of routines, nodes, libraries that are essential for ROS system
- It runs at the background.
- ROS Core starts the ROS Master to enable the registration of all Nodes, Topics and Services.



ROS Tools

ROS provides a variety of tools to build, debug and simulate . The Most common tools are:

- Catkin
- rqt_graph
- Opencv Library
- Gazebo

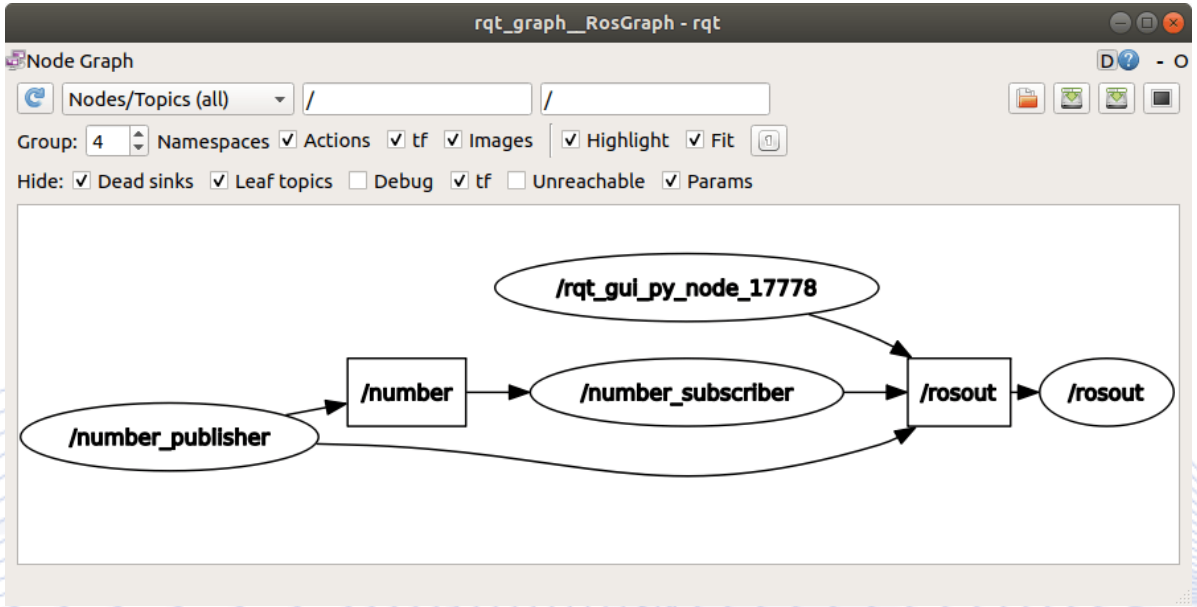
Catkin

- Catkin is a tool that is included with ROS and it is used to build packages.
- The name Catkin was given by the Willow Garage Company that created ROS.
- It was created for easy package installation and distribution.
- It consist of macro instructions and scripts to build packages

Image of male Catkin [CTKN]

Rqt graph

- Rqt_graph is GUI tool that shows the function of all nodes and topics of a ROS project.



A typical rqt_graph showing the nodes and topics at a graph level [RQTG]

OpenCV Library

- OpenCV is an open-source library for computer vision, machine learning and real-time applications. The library includes functions for:
  - Object Detection
  - Deep Neural Networks
  - Machine Learning
  - Image Processing
  - Video Analysis
  - 3D Reconstruction with Camera
  - Image or Video Input and Output

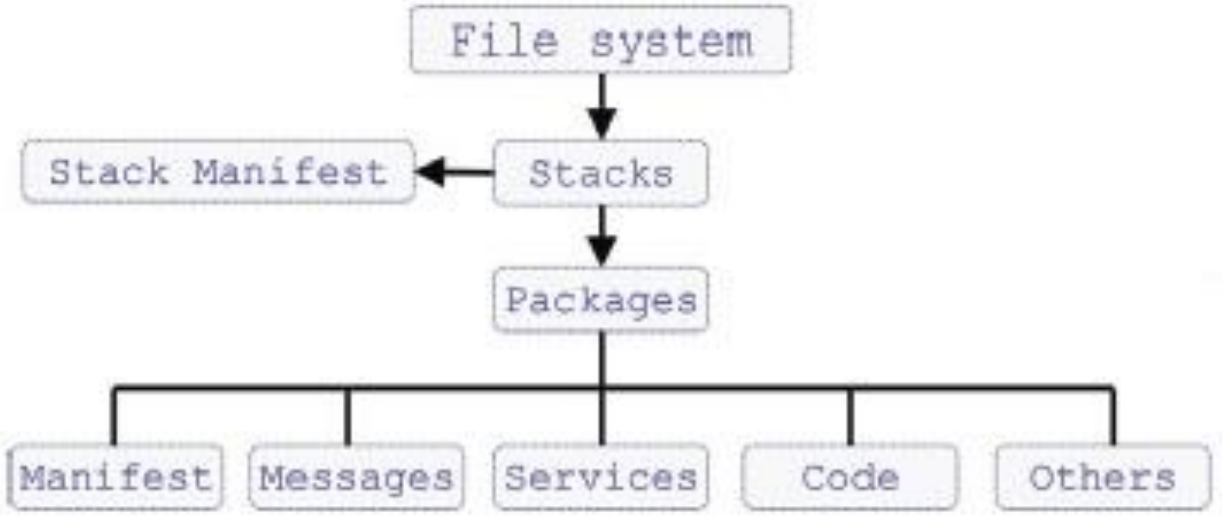
Gazebo

- Gazebo is a simulator for testing and training robots using realistic scenarios in virtual environments

A simulation of a scenario with various robots in Gazebo [GZBO]

ROS File System

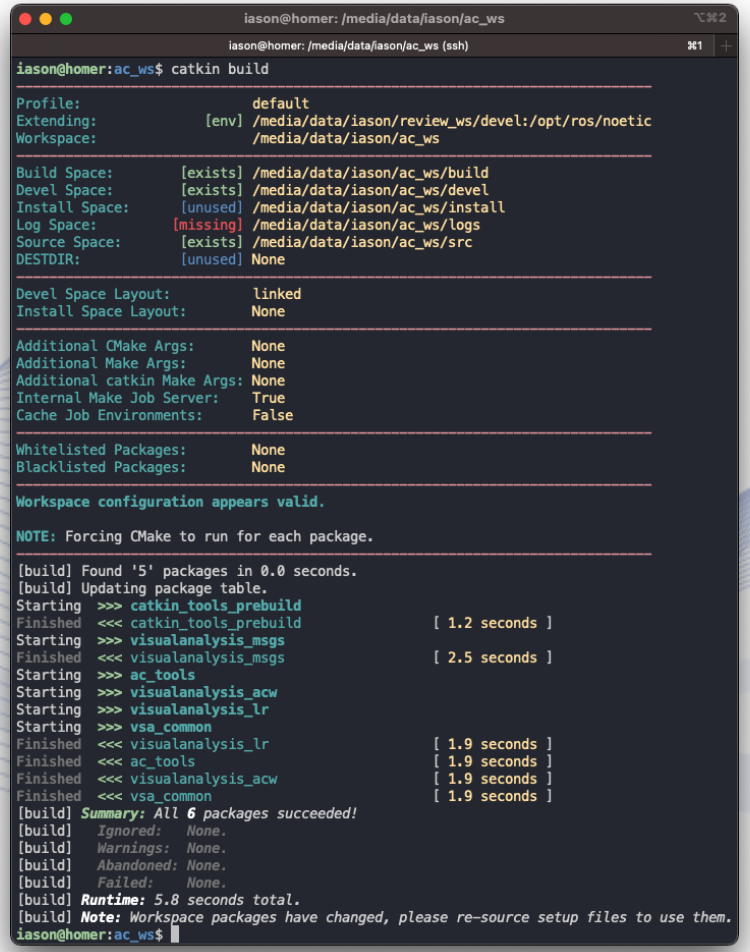
- ROS nodes and packages are organized in a specific way.
- It is common to define a workspace for each ROS application (e.g., icarus_ws).
- The location of this workspace can be anywhere in our system.



The image is from the paper [RRSham]

ROS Workspace

- A typical ROS workspace structure can be:
  - ros_ws/
    - src/
    - build/
    - devel/
    - logs/
  - The directories build, logs and devel are auto-generated when building the workspace (e.g., with catkin build).



ROS Node in Python

```
import rospy, rospkg

def main(args):
   rospy.init_node('TestNode', anonymous=True,  log_level=rospy.DEBUG)
   try:
      rospy.spin()
   except KeyboardInterrupt:
      print("Shutting down")
```

ROS Publisher in Python

```
import rospy, rospkg, cv2
from cvbridge import CvBridge, CvBridgeError
from sensor_msgs.msg import Image


image_pub = rospy.Publisher(“image_topic/camera_raw", Image,
            queue_size=1)


cv_image = cv2.imread('file/path.jpg’)
while True:
            pub_frame = self.bridge.cv2_to_imgmsg(cv_image, "bgr8")
            self.image_pub.publish(pub_frame)
```

ROS Subscriber in Python

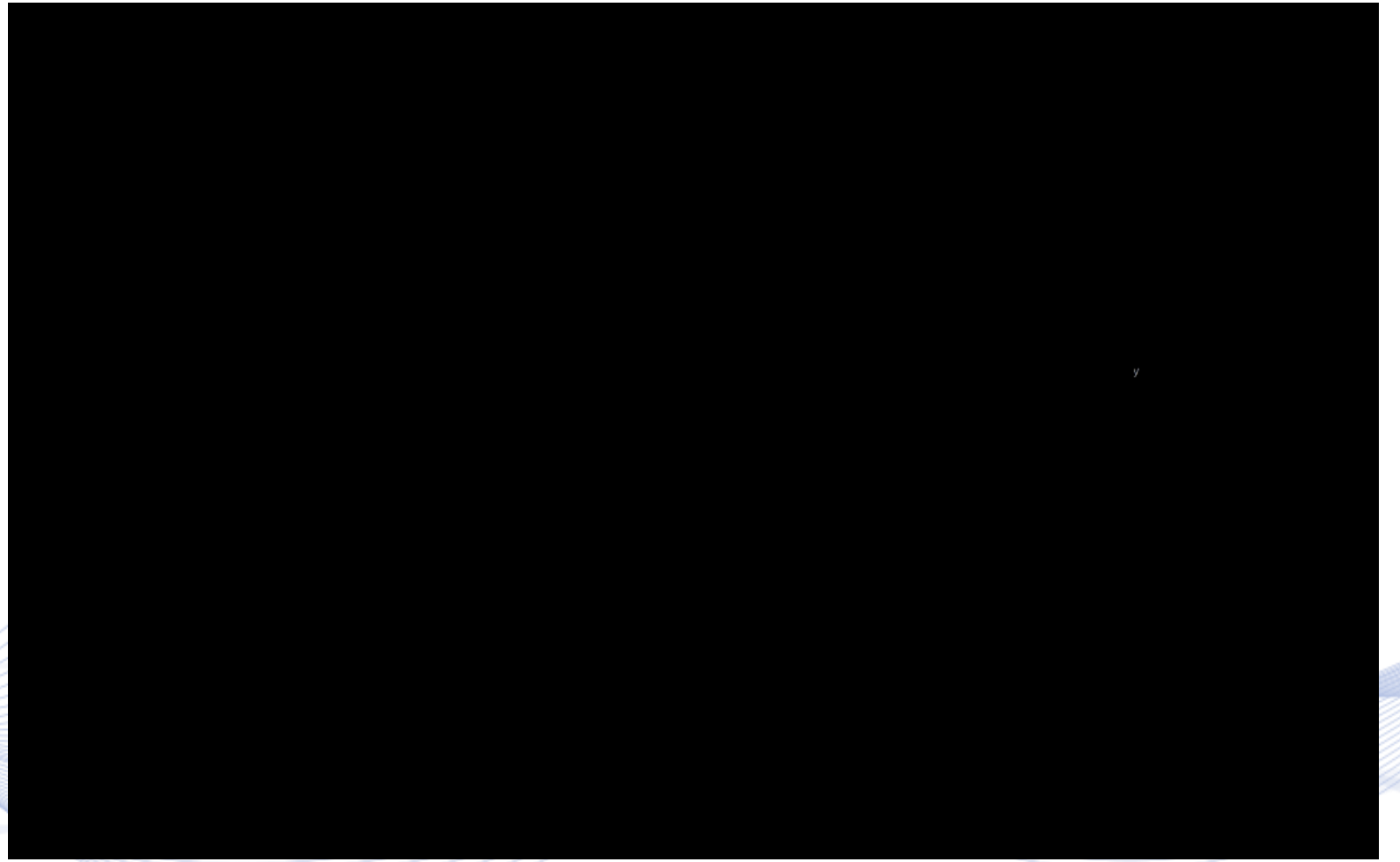
```
import rospy, rospkg
from cvbridge import CvBridge, CvBridgeError
from sensor_msgs.msg import Image

image_sub = rospy.Subscriber("image_topic/camera_raw",
                             Image, self.image_callback, queue_size=1,
                             buff_size=2 ** 24)
```

ROS Subscriber in Python

```
def image_callback(self, data):
        try:
                cv_image = self.bridge.imgmsg_to_cv2(data, "rgb8")
                self.img = cv_image
                rospy.loginfo("img")
        except CvBridgeError as e:
                print e
                self.img = None
                return
        # maybe do something else here with the Image?
        return
```

Object Detector and Tracker



Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr