Introduction to ROS

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

Version 1.0
Date: 8/12/2020

Artificial Intelligence & Information Analysis Lab

Presentation Outline

- What is ROS
- Robot Architecture
- ROS Architecture
    - Computational Level
    - File System Level
- Examples
    - Publisher and Subscriber Nodes
    - Object Detector and Tracker Node
    - Object Detector Node

Presentation Outline

- **What is ROS**
- Robot Architecture
- ROS Architecture
  - Computational Level
  - File System Level
- Examples
  - Publisher and Subscriber Nodes
  - Object Detector and Tracker Node
  - Object Detector Node

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***
- ***Object Tracking***
- ***Object Recognition***
- ***Gesture Recognition***
- ***Control of Multi-Drones***

BlueROV2 [BLRV]

IRB 120 Robot [ABB]

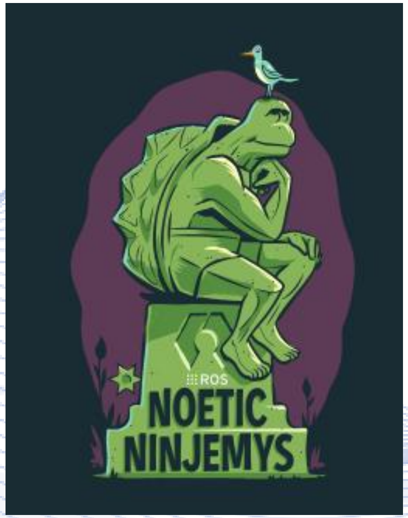
ROS Software

- ROS is an open source software and can be installed in any computer with Linux operating system.
- ROS 2 is supported by Windows 10, MacOS and Linux.
- Each ROS distribution is supported by a specific Linux Distribution.
- ROS has tools, libraries and drivers for both C++ and Python programming languages that helps you out to develop your application.
- Has a large community
- It is well supported. Provides tutorials, documentation, libraries etc
- Events (RoboCon, ROS-Industrial Conference, ROS Summer School)

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

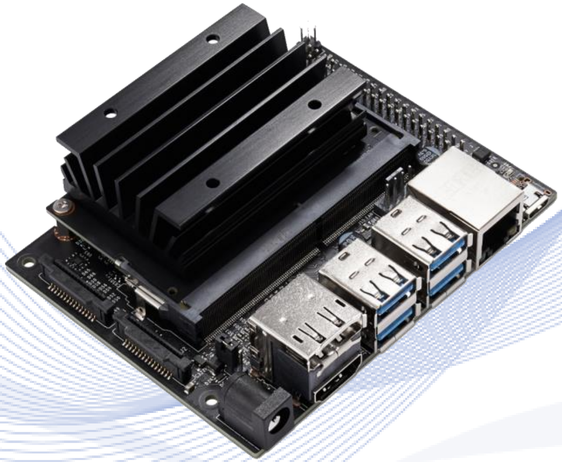
ROS Hardware

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]

Presentation Outline

- What is ROS
- **Robot Architecture**
- ROS Architecture
  - Computational Level
  - File System Level
- Examples
  - Publisher and Subscriber Nodes
  - Object Detector and Tracker Node
  - Object Detector Node

Robot Architecture

- A Robot may have a variety of hardware devices that a developer has to program and control such as
  - Sensors (Altimeter, LIDAR, Gyroscopes, Humidity Sensors etc.)
  - Motors (Brushless Motors, AC Motors, Stepper Motors etc.)
  - Displays (LCD Display, TFT etc.)
  - Communication Devices (GPS/GSM, Bluetooth, Wifi, IR)

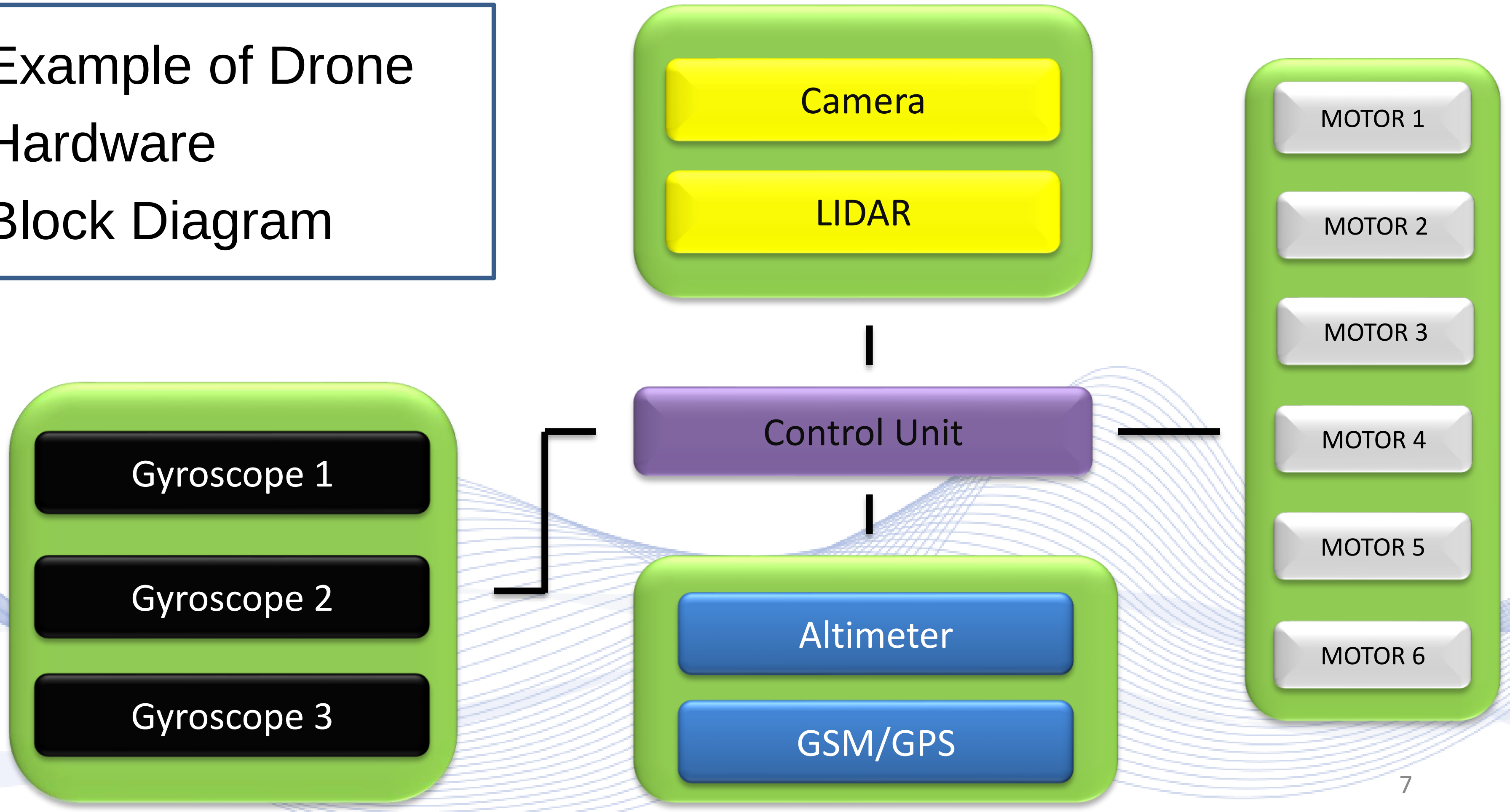
- But also may develop algorithms necessary for
  - ***Processing Data***

    (Calculations, Filters, Object Detector, Image Segmentation etc)
  - ***Optimization***

    (PID Controller, Bee-Hive Algorithm, Wolf-Pack Algorithm, Differential Evolution Algorithms etc.)

Robot Hardware Architecture

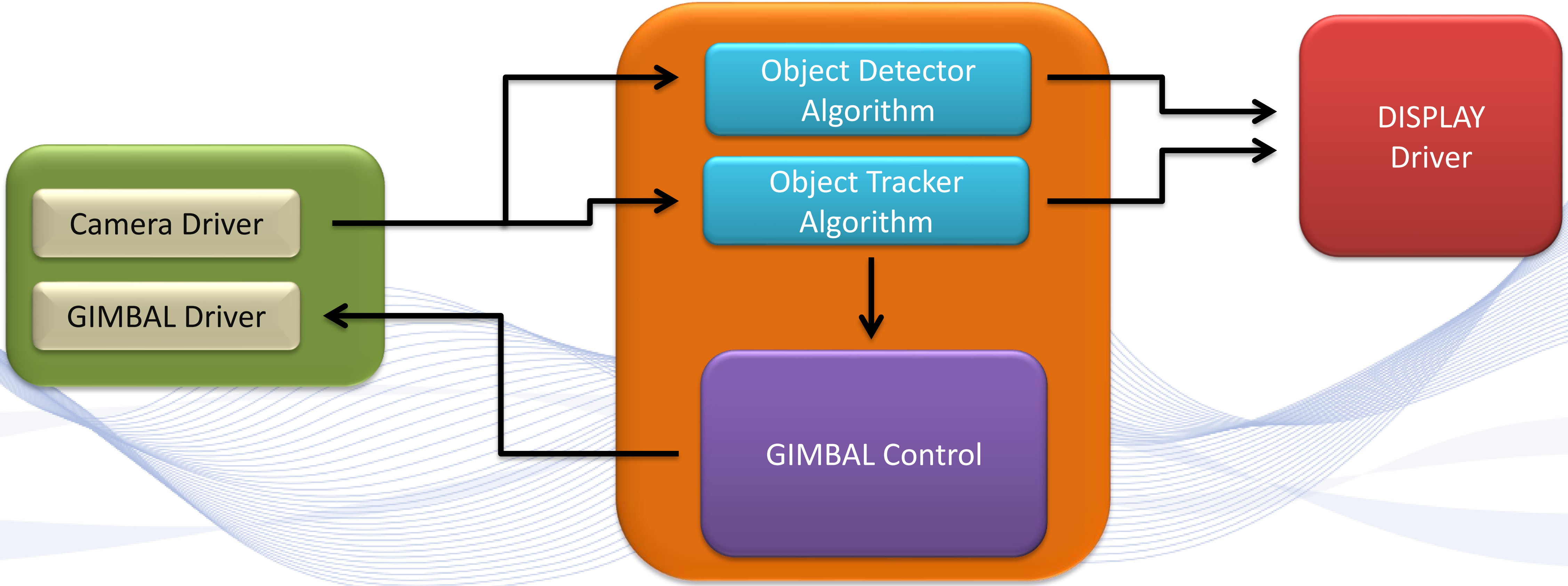
Example of Drone Hardware Block Diagram



Artificial Intelligence & Information Analysis Lab

Robot Software Architecture

Example of Drone Software Block Diagram



Presentation Outline

- What is ROS
- Robot Architecture
- **ROS Architecture**
  - Computational Level
  - File System Level
- Examples
  - Publisher and Subscriber Nodes
  - Object Detector and Tracker Node
  - Object Detector Node

ROS Architecture

- We can describe ROS in two levels:
  - File System level
  - Computational level

A] File System Level

How ROS Directories, Folders and files are organized in the system.

B] Computational Level

How ROS programs communicate with each other.

A) ROS Computational Level

The basic ROS concepts are:

- Nodes
- Topics
- Messages and Bags
- Services
- Packages
- ROS Master
- ROS Core
- ROS Tools

Nodes

- ❑ A Node is a piece of code that performs a specific function to the robot.
- ❑ They are simple programming files with the extension
  - .py (Python)
  - .cpp (C++)

A Robot usually consist of many nodes, such as

- Camera control
- LIDAR
- GIMBAL Control
- Display
- Object Detector
- Object Tracker
- Image Filter
- Etc.

Artificial Intelligence & Information Analysis Lab

Messages and Bags

MESSAGES

ROS uses messages so as nodes can sent or receive data. Messages contains Fields and Constants.

- ***Fields***

  Fields contain the data. The supported data types are integer, float, boolean, arrays, structures etc.

- ***Constants***

  Constants are values that can used to interpret the fields.

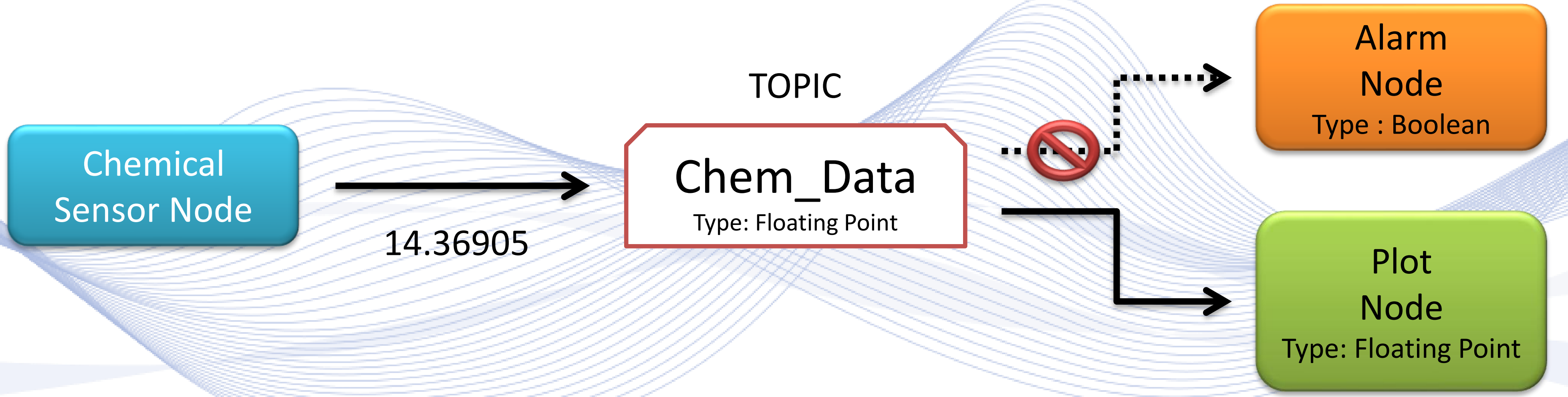
BAGS

A bag contains many messages.

http://wiki.ros.org/Messages

Topics

- Topics are names that are used from the Nodes to communicate with each other and exchange messages.
- A topic is associated with the message type and thus only nodes with the same type can send or receive messages. For example, a Node that sends floating point data to a topic, the data cannot be received by a boolean type Node.



Services

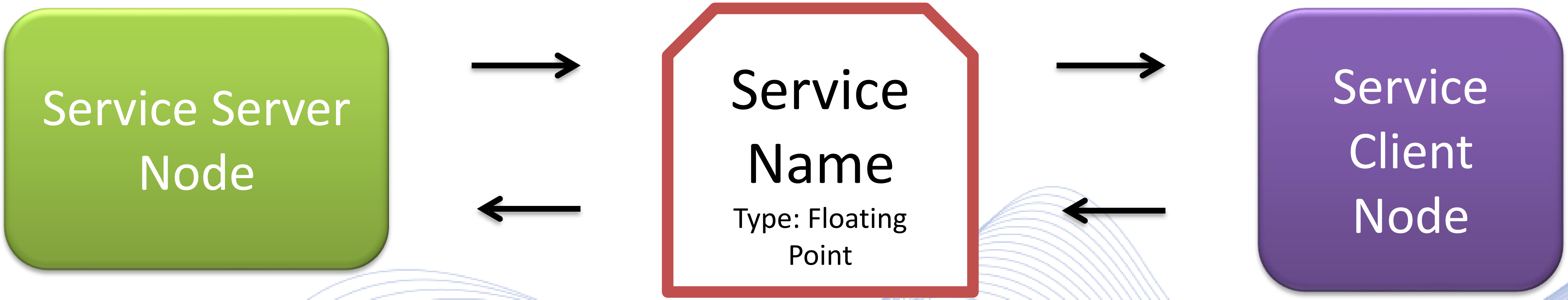
A Publish/Subscriber model is a very convenient one-way communication between nodes. In distribution systems, there's a need for data exchange between nodes in a two-way direction. For this reason the Service model is used.

A Service is a request/reply model for node communication using two types of messages:

- A Request message
- A Reply message

Like messages, a service type depends on the data type.

Services


Service Server Node
Service Name
Type: Floating Point
Service Client Node

Packages

Packages

Packages are a set of nodes, manifests, libraries, files (pictures, videos, data), code pieces that are gathered together in a folder so it can be reused easily.

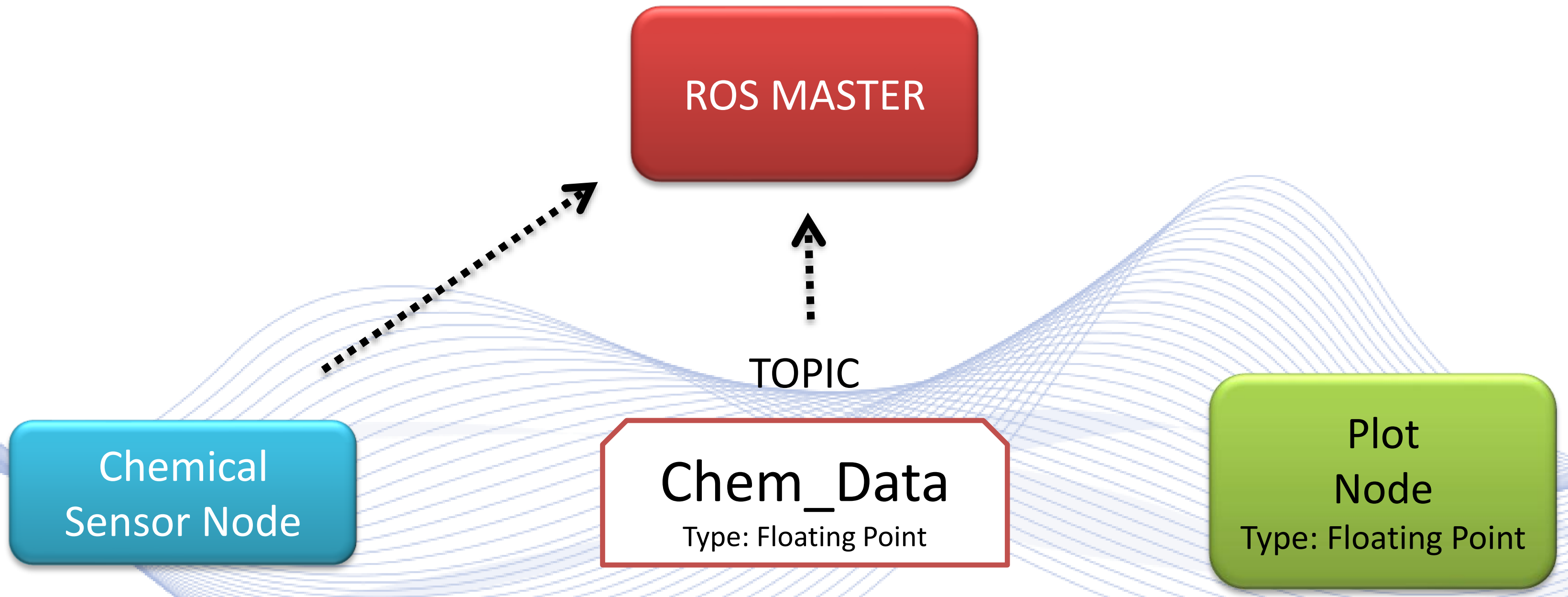
Packages have a specific file structure when they are created. That doesn’t mean that the developer cannot create his own folders and files inside the package folder.

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 the another Node, ROS Master provides a way to the Nodes to locate each other.
- After the Nodes identify each other, they are communicating

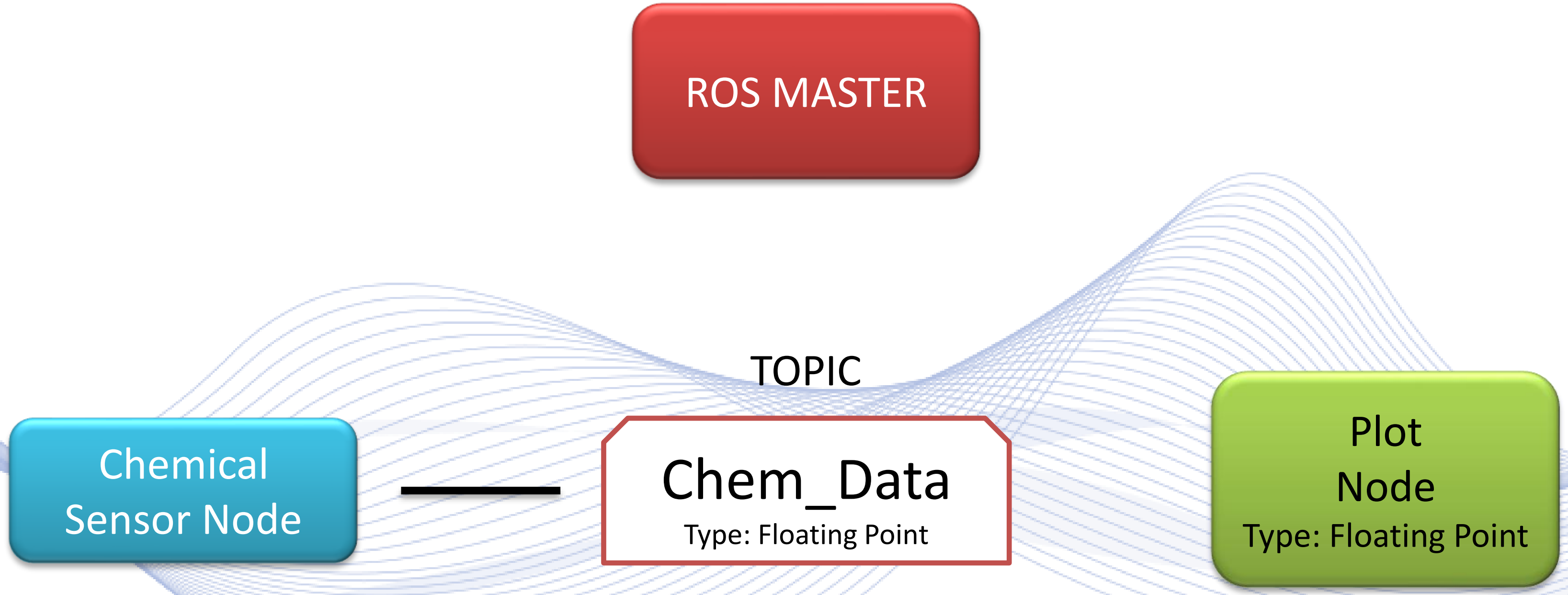
ROS Master

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



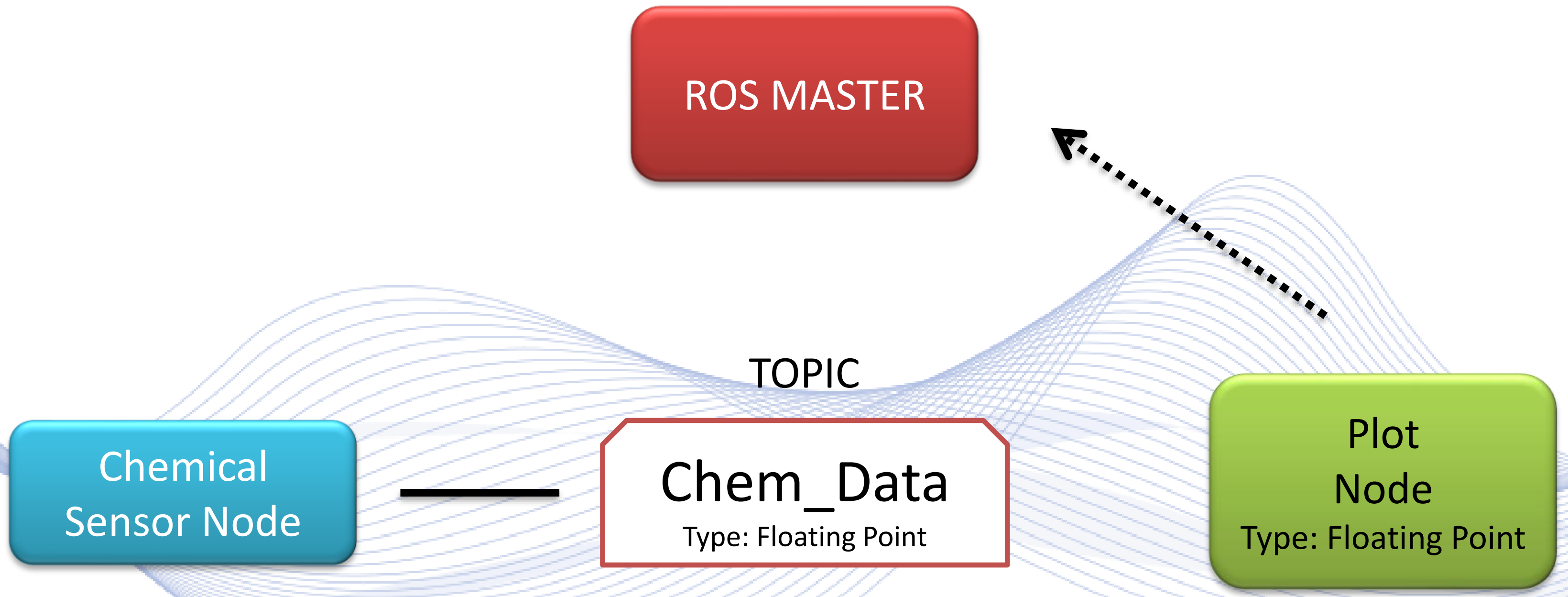
ROS Master

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



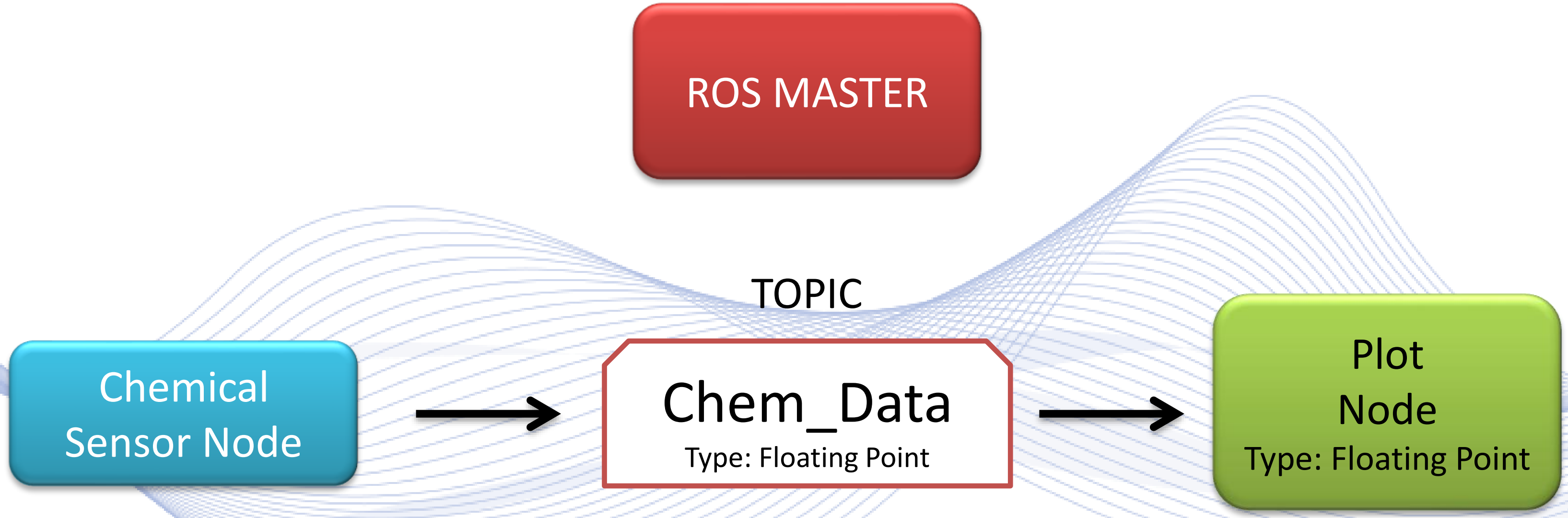
ROS Master

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



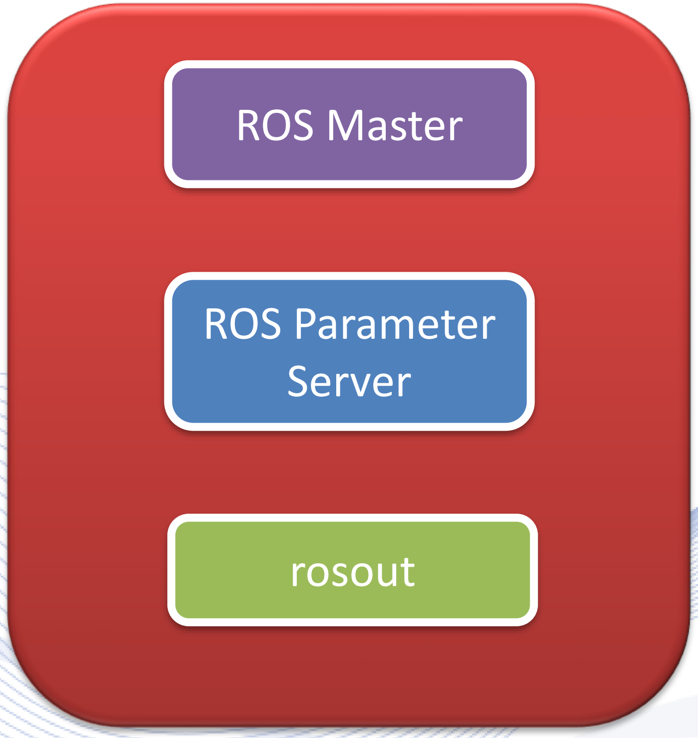
ROS Master

- 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 Core

```
File  Edit  View  Search  Terminal  Help
atlantis@TESLA:~$ roscore
... logging to /home/atlantis/.ros/log/5b8f2e22-1df6-11eb
-9a7f-0ceee69e28f4/roslaunch-TESLA-3619.log
Checking log directory for disk usage. This may take a wh
ile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://TESLA:43519/
ros_comm version 1.14.5


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES

auto-starting new master
process[master]: started with pid [3629]
ROS_MASTER_URI=http://TESLA:11311/

setting /run_id to 5b8f2e22-1df6-11eb-9a7f-0ceee69e28f4
process[rosout-1]: started with pid [3640]
started core service [/rosout]
```

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

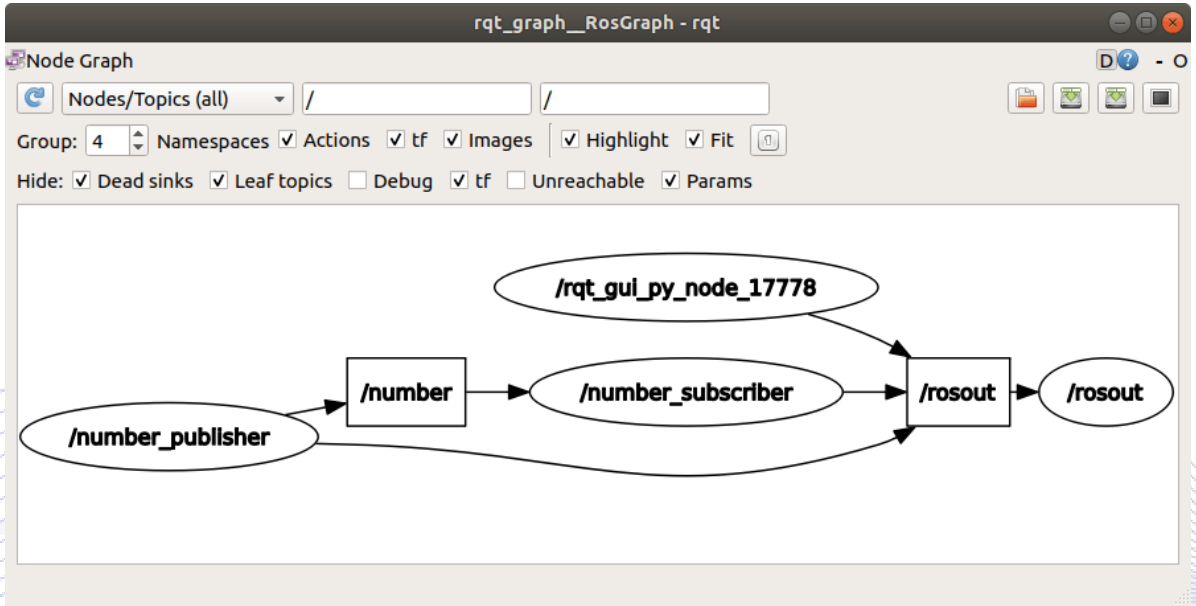
What is Catkin?

- 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

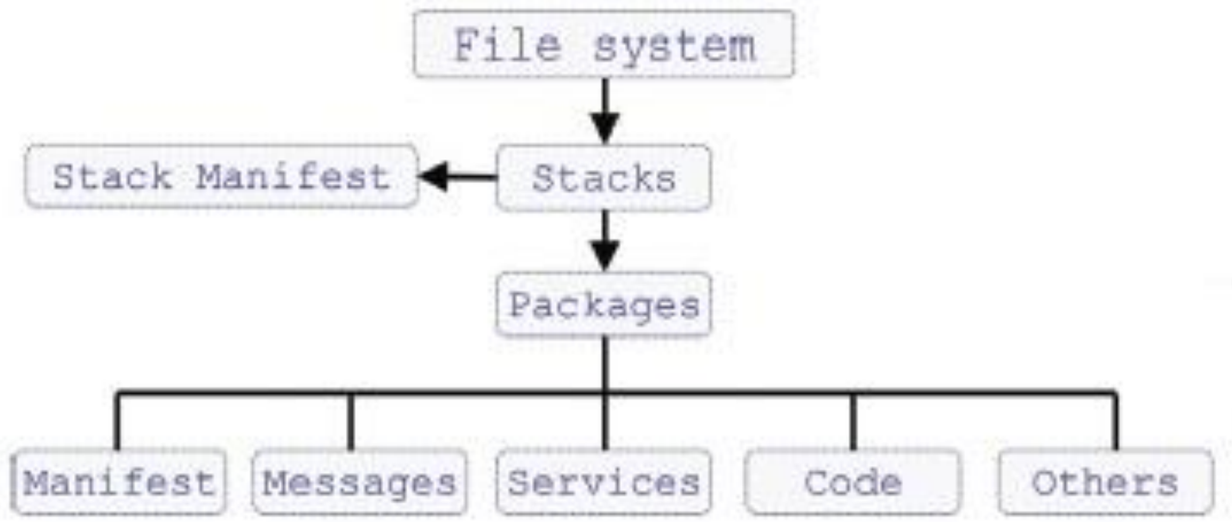
Gazebo

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

GAZEBO

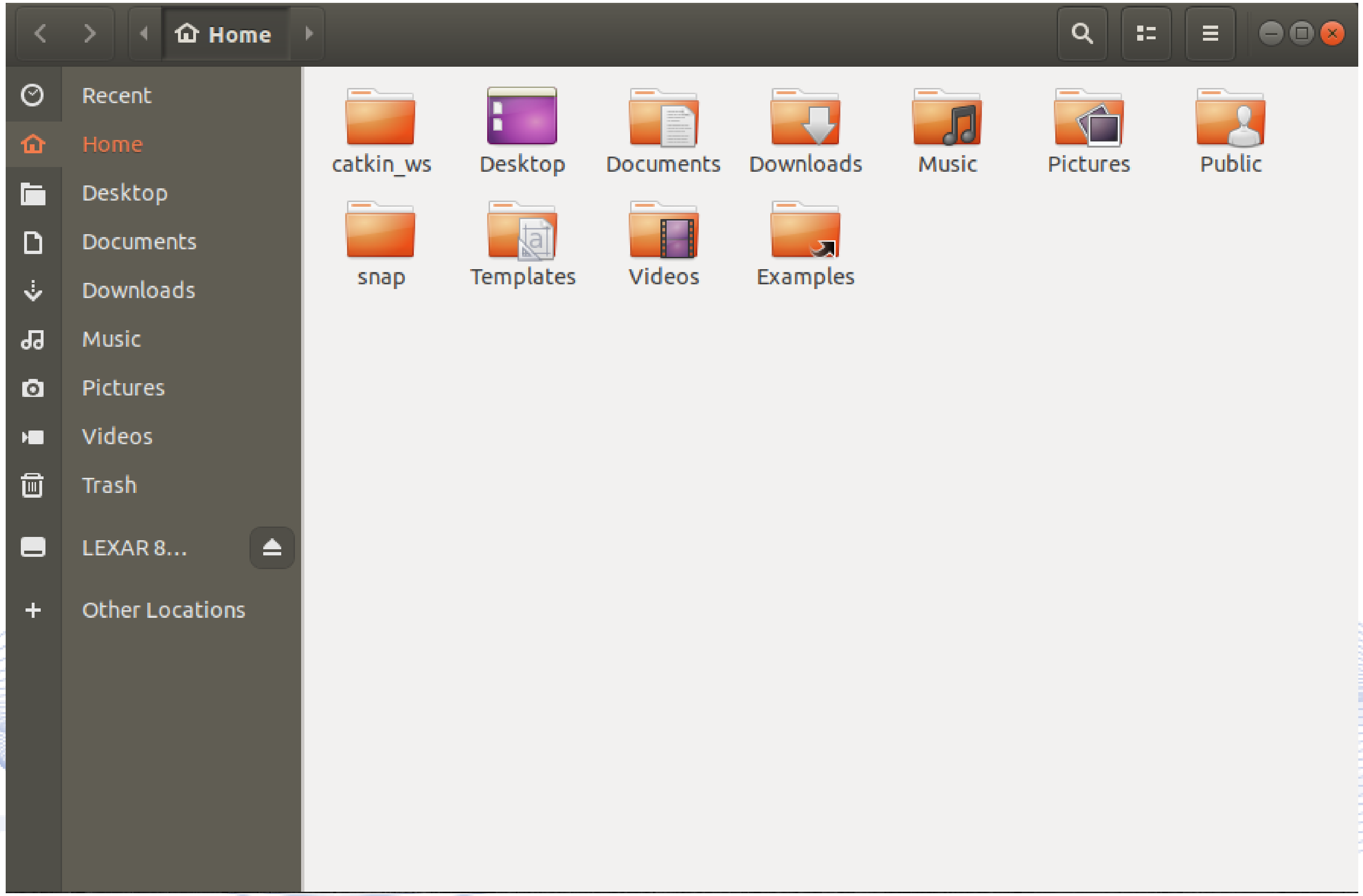
A simulation of a scenario with various robots in Gazebo [GZBO]

B) File System Level

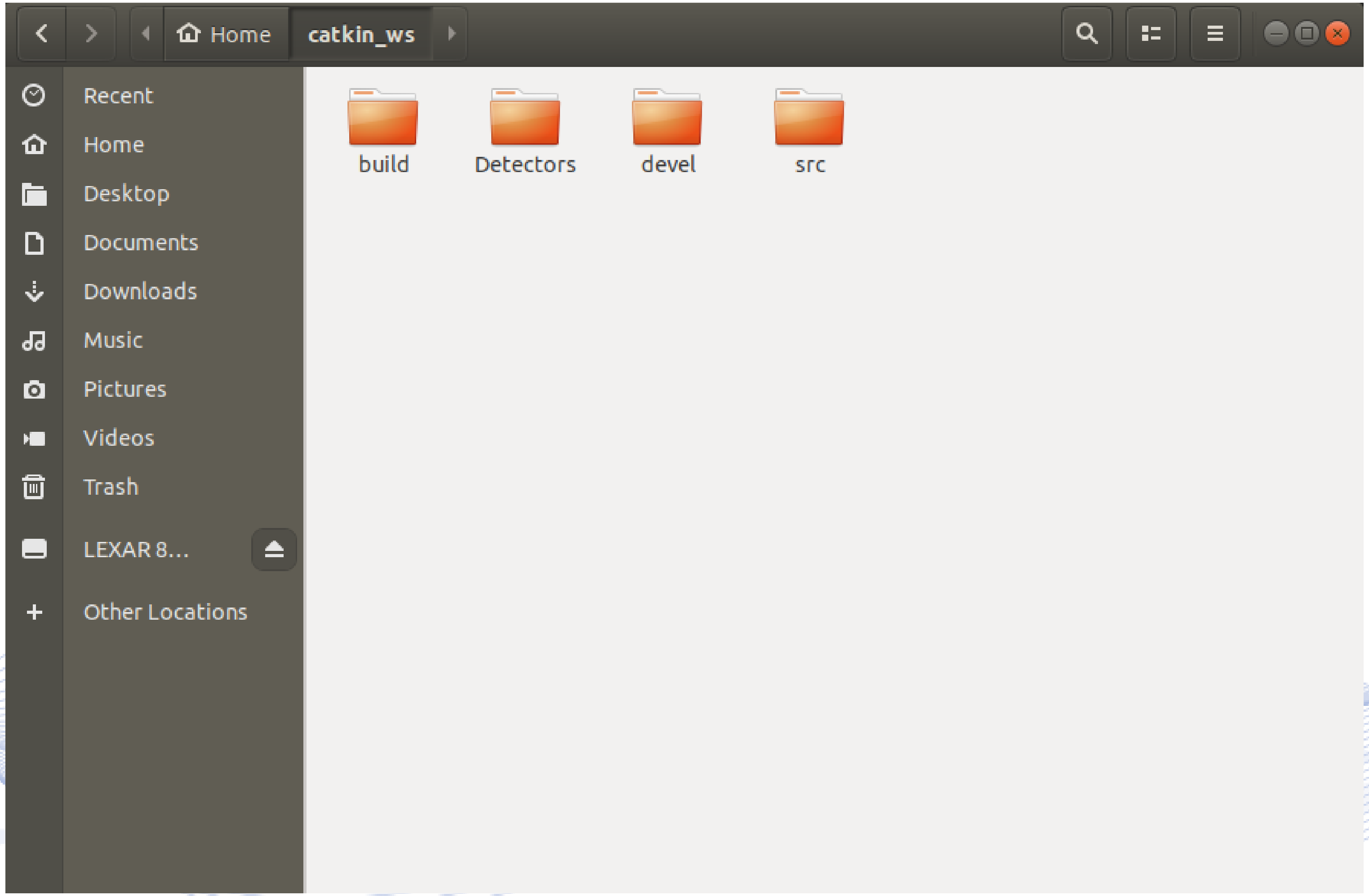


The image is from the paper [RRSham]

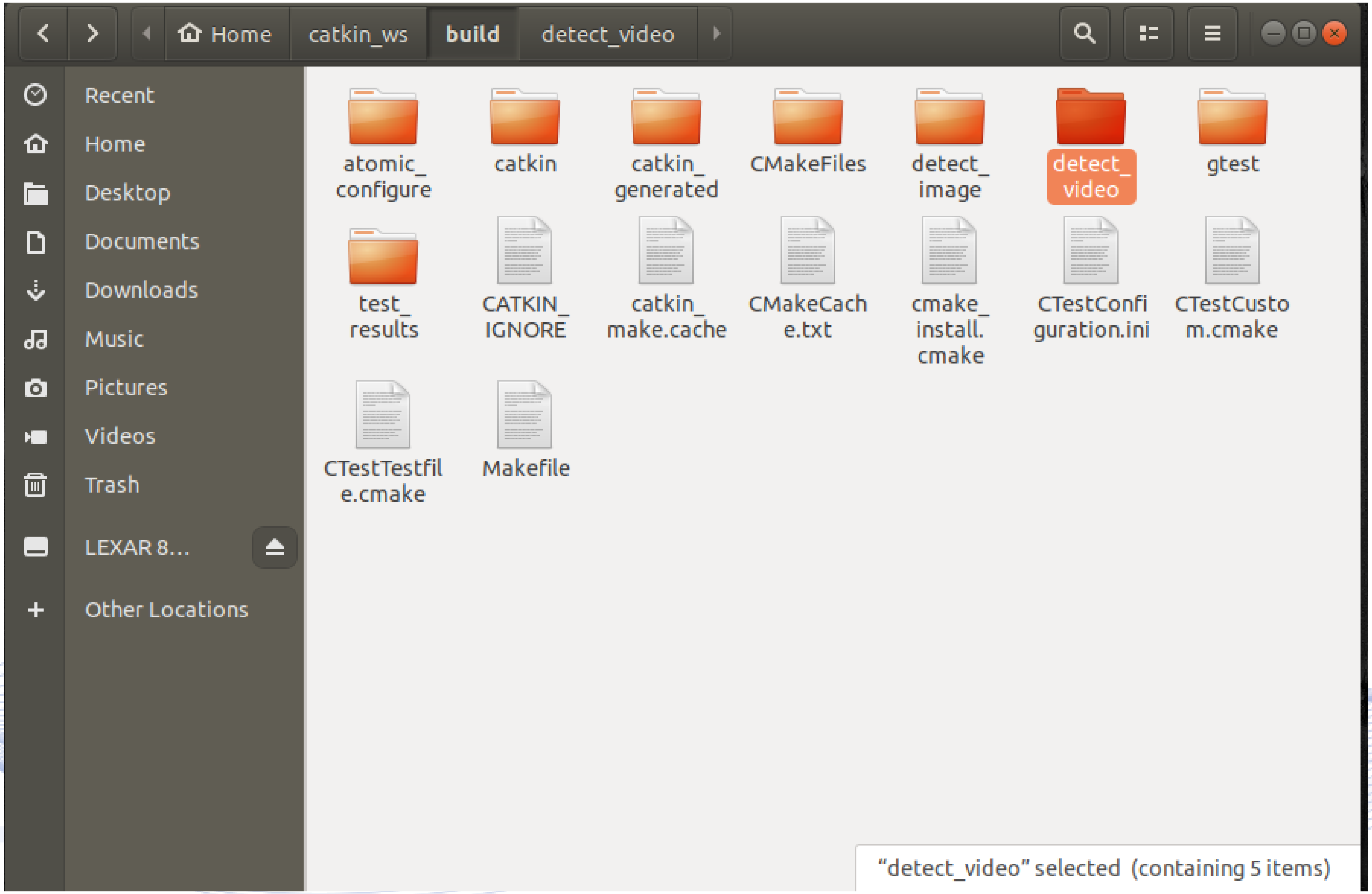
Catkin Workspace Folder Location



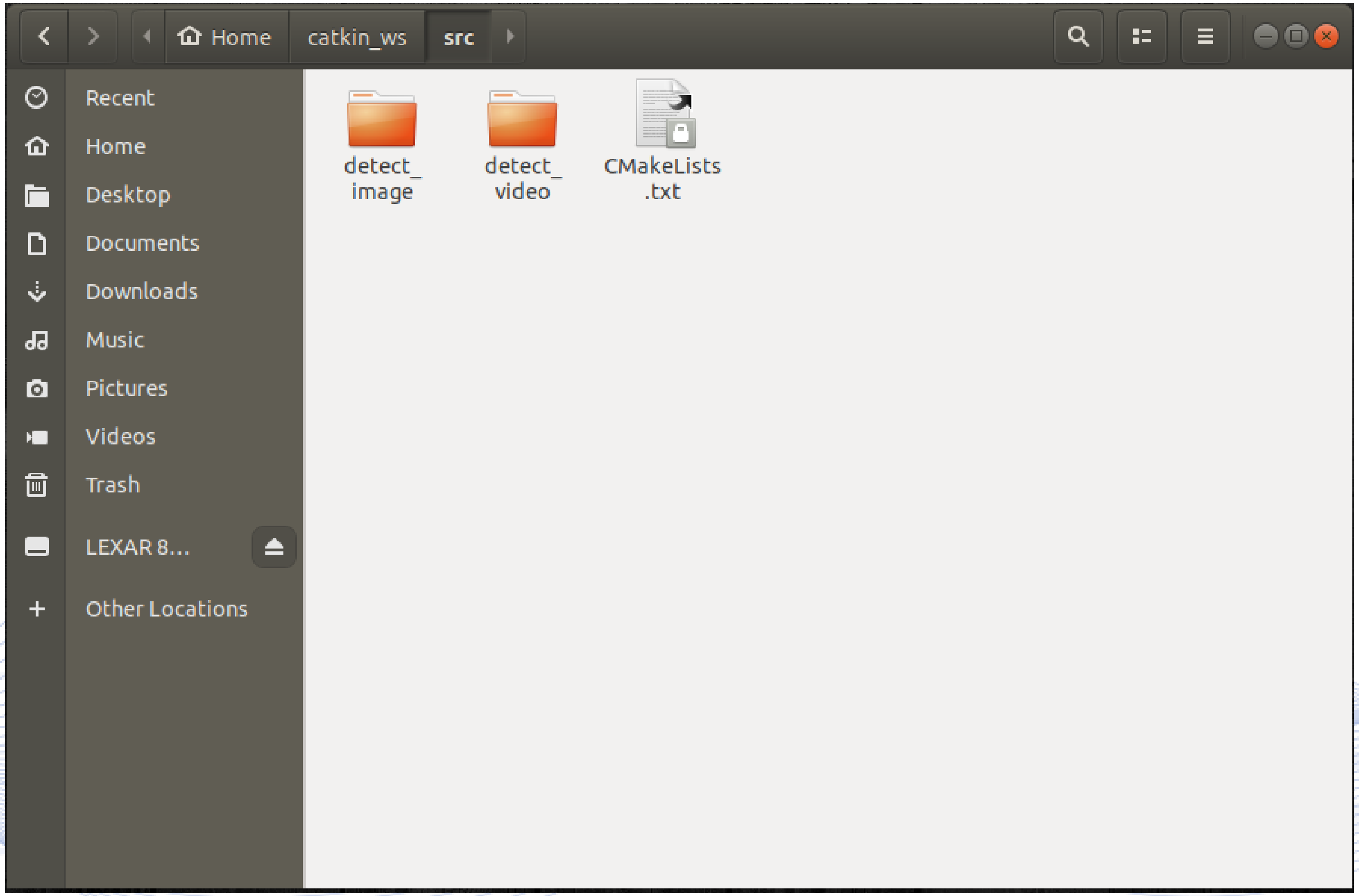
Catkin Workspace



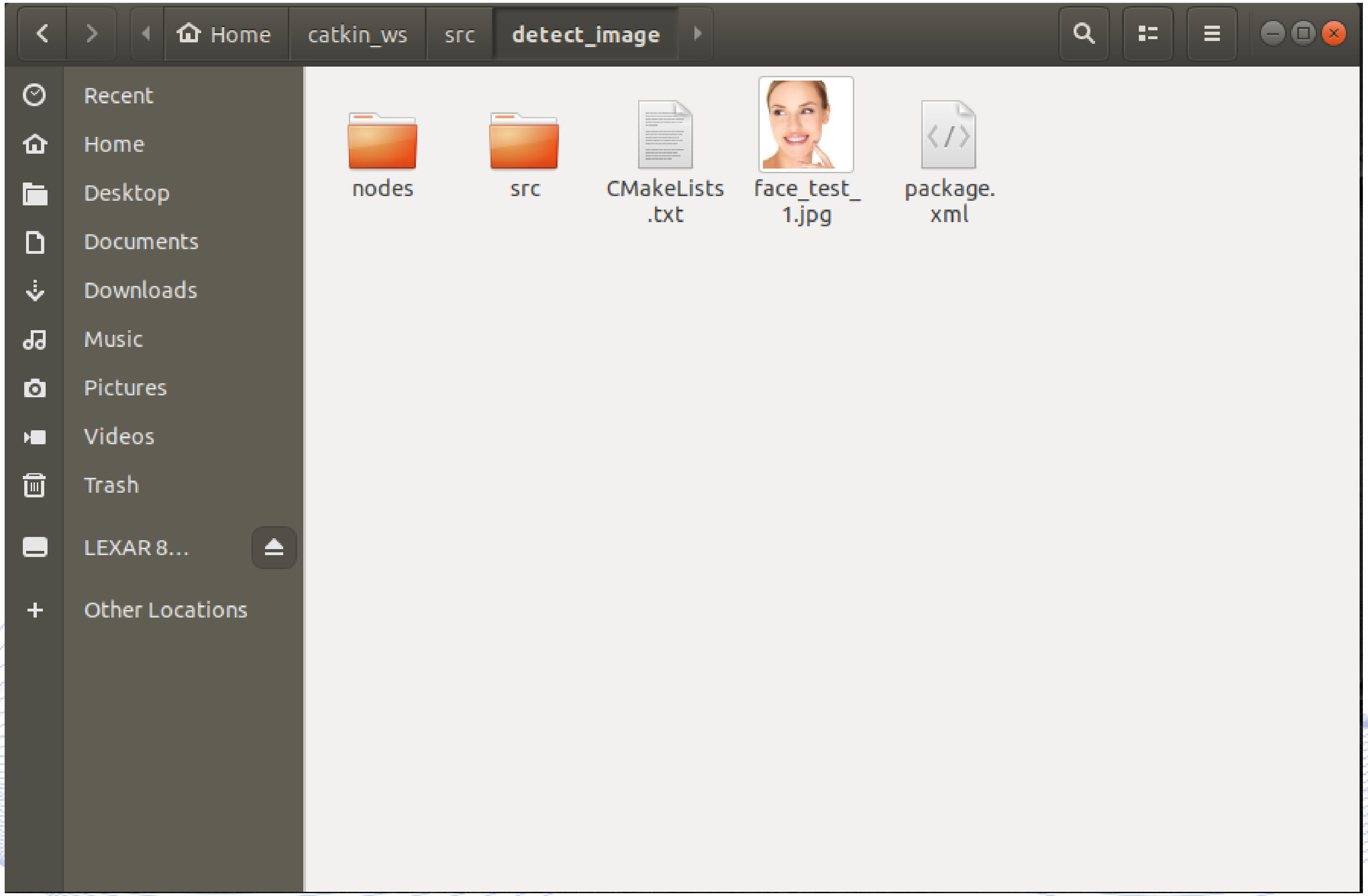
Catkin Build folder



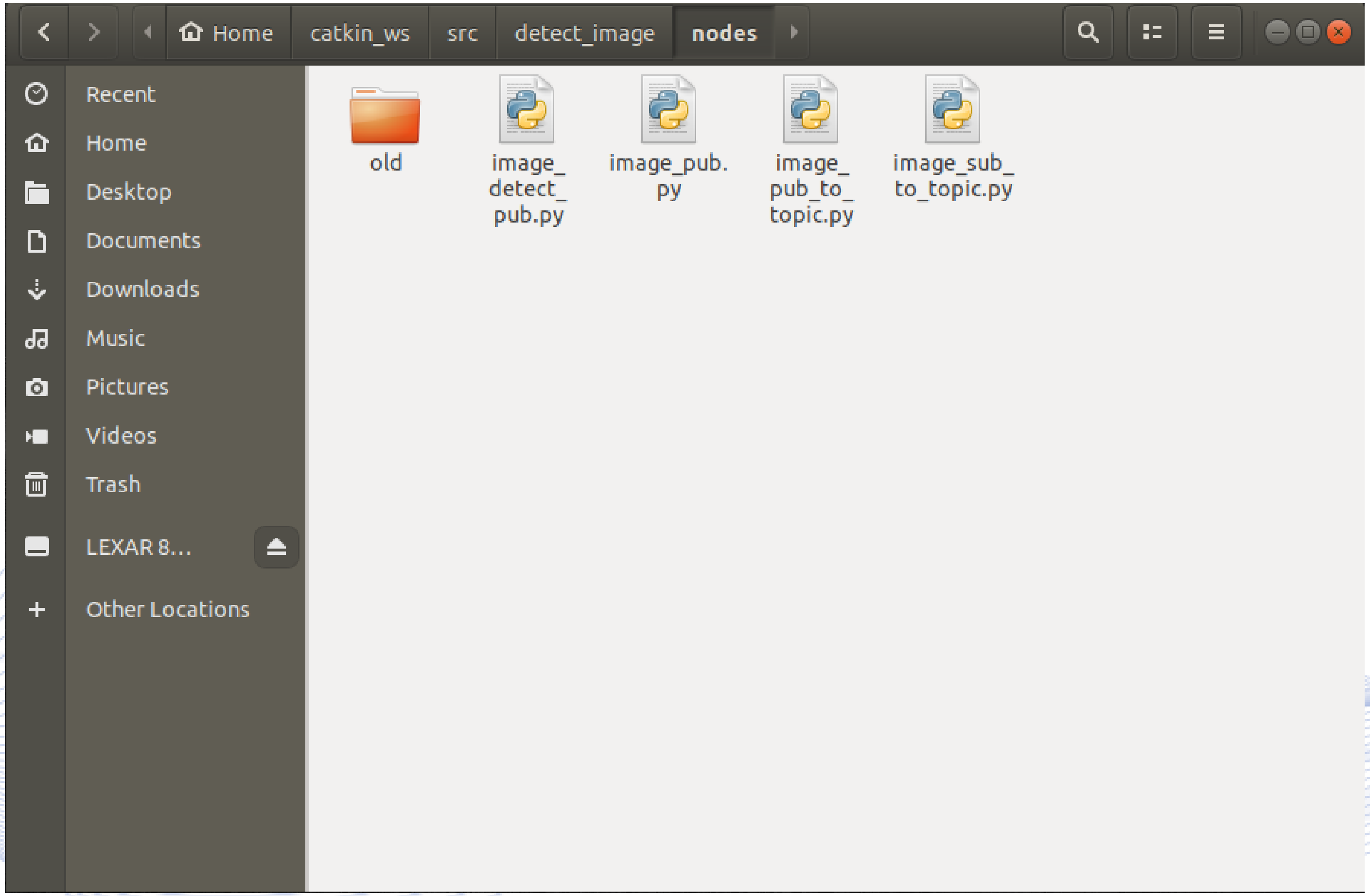
Src Folder with Packages



Package Folder



Node Folder



Artificial Intelligence & Information Analysis Lab

Presentation Outline

- What is ROS
- Robot Architecture
- ROS Architecture
  - Computational Level
  - File System Level
- **Examples**
  - Publisher and Subscriber Nodes
  - Object Detector and Tracker Node
  - Object Detector Node

Presentation Outline

- What is ROS
- Robot Architecture
- ROS Architecture
  - Computational Level
  - File System Level
- Examples
  - **Publisher and Subscriber Nodes**
  - Object Detector and Tracker Node
  - Object Detector Node

Publisher and Subscriber Nodes

Assume that we receive a n image from a camera and we want to show this image. We must create a Publisher Node, a Topic and a subscriber Node.

Publisher Node

```python
#!/usr/bin/env python
import roslib
import time

import rospy
import sys
import cv2
from cv_bridge import CvBridge
from sensor_msgs.msg import Image

def sender():
    rospy.init_node('sender')
    rospy.loginfo('sender node started')
    pub = rospy.Publisher('imageview', Image)
    rate = rospy.Rate(1.0) #1Hz

    path = '/home/aegis/catkin_ws/src/detect_image/face_test_1.jpg'
    img = cv2.imread(path) #save image to img variable
    bridge = CvBridge()
    cv_image = bridge.cv2_to_imgmsg(img)
    while not rospy.is_shutdown():
        rospy.loginfo('PUBLISH IMAGE')
        pub.publish(cv_image)
        rate.sleep()

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

Subscriber Node

```python
#!/usr/bin/env python
import roslib
import time

import rospy
import sys
import cv2
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


def callback(data):
    bridge = CvBridge()
    rospy.loginfo('receiving image')
    cv_image = bridge.imgmsg_to_cv2(data)
    cv2.imshow("image", cv_image)
    cv2.waitKey(1)

def receiver():
    rospy.init_node('receiver')
    rospy.loginfo('receiver node started')
    pub = rospy.Subscriber('imageview', Image, callback)
    rospy.spin()
    cv2.destroyAllWindows()

#if _name_=='_main_':
#    image_sub()
```

Artificial Intelligence & Information Analysis Lab

Presentation Outline

- What is ROS
- Robot Architecture
- ROS Architecture
  - Computational Level
  - File System Level
- Examples
  - Publisher and Subscriber Nodes
  - **Object Detector and Tracker Node**
  - Object Detector Node

Object Detector and Tracker Node

```python
#!/usr/bin/env python2.7
import rospy
import sys
import cv2


def start_node():
    rospy.init_node('image_pub')
    rospy.loginfo('image_pub node started')

tracker = cv2.TrackerKCF_create()
#face_cascade = cv2.CascadeClassifier('../opencv_detection/models/haarcascade_frontalface_default.xml')

face_cascade = cv2.CascadeClassifier('/home/atlantis/catkin_ws/Detectors/haarcascades/haarcascade_frontalface_default.xml')

path2 = '/home/atlantis/catkin_ws/Detectors/haarcascades/haarcascade_frontalface_default.xml'
path = '/home/atlantis/catkin_ws/src/detect_video/test_video/test_video_1.3g2'

# naming and linking the cv2.VideoCapture to video_open and the test video
# "test_video_1.3g2"

video_open = cv2.VideoCapture(path)
ok=False
initBB=None
```

Example Structure

```python
while True:

    ret,frame = video_open.read()

    if not ret:
        print("FAILED")
        break

    (H, W, D) = frame.shape
    r = 500.0 / W
    dim = (500, int(H * r))
    frame = cv2.resize(frame, dim)

    if not ok:

        #Detector
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        faces= face_cascade.detectMultiScale(gray_frame, 1.3, 5)

        for(x, y, w, h) in faces:
            cv2.rectangle(frame, (x,y), (x+w, y+h), ( 255, 0, 0),2)

        if len(faces) > 0 :
            initBB = tuple(faces[0])
            tracker.init(frame, initBB)
            ok=True
```

Example Structure

```
    else:
        (ok,box) = tracker.update(frame)
        if not ok:
            tracker = cv2.TrackerKCF_create()
        else:
            (x, y, w, h)=[int(v) for v in box]
            cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 255,0), 2)

# Show the image frame
    cv2.imshow("Test Video 1", frame)


# Press "q" to close the video. waitKey(45) for 45ms frame rate
    key = cv2.waitKey(25) & 0xFF

        if key == ord("q"):
                break

video_open.release()
cv2.destroyAllWindows()
```

Object Detector and Tracker

Artificial Intelligence &
Information Analysis Lab

Presentation Outline

- What is ROS
- Robot Architecture
- ROS Architecture
  - Computational Level
  - File System Level
- Examples
  - Publisher and Subscriber Nodes
  - Object Detector and Tracker Node
  - **Object Detector Node**

Object Detector Node

```python
#! /usr/bin/env python

import rospy
from std_msgs.msg import Header
from sensor_msgs.msg import CompressedImage
from detector.msg import BoundingBox
from std_srvs.srv import Empty, EmptyResponse
from keras.applications.imagenet_utils import preprocess_input

from keras.preprocessing import image
import cv2
import tensorflow as tf
from deep_learning.ssd_detector.ssd import SSD300 as SSD
```

Object Detector Node

```
class DetectionPoseNode :
    def __init__(self) :
        models_path = join(os.path.dirname(os.path.realpath( __file__) ) , ’ models ’ )
        self.pub_bbox = rospy.Publisher(’face_detector/bbox ’ , BoundingBox ,
                                        queue_size = 10)
        self.class_names = [ ” background ” , ’face’]
        self.num_classes =len(self.class_names)
        self.input_shape = (300 , 300 , 3)
        self.conf_thresh = 0.6

        self.model = SSD (self.input_shape , num_classes = self. num_classes )

        self.model.load_weights (join(models_path , ’ fddb_model. hdf5 ’ ) )

        self.pose_model = Pose_Estimator(join(models_path ,
                                        ‘pose_estimation_model.h5’))
        self.bbox_util = BBoxUtility(self.num_classes )
        self.graph = tf.get_default_graph( )
```

Object Detector Node

```
def listener(self):
    rospy.init_node('detector', anonymous=True)
    rospy.loginfo('Detector node started!')
    rospy.Subscriber("/usb_cam/image_raw/compressed",
        CompressedImage, self.detection_callback, queue_size=1)

    rospy.spin()
```

Object Detector Node

```python
def detection_callback(self, data) : ,
    # Compressed image
    np_arr = np.fromstring(data.data , np.uint8)
    cv_image = cv2.imdecode (np_arr , cv2.IMREAD_COLOR)
    # For uncompressed images we need to use the cv2 bridge
    # cv_image = self.bridge.imgmsg_to_cv2( data , ” bgr8 ” )
    self.process_image(cv_image)
```

Object Detector Node

```
if __name__ == ’__main__’:
    node = DetectionPoseNode( )
    node.listener( )
```

Object Detector Node

```
def process_image(self, orig_image):
    resized = cv2.resize(orig_image, (self.input_shape[0], self.input_shape[1]))

    rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)


    # Use model to predict
    with self.graph.as_default():
        x = preprocess_input(np.array([image.img_to_array(rgb)]))

        y = self.model.predict(x)
        results = self.bbox_util.detection_out(y)

    timestamp = rospy.get_rostime()
```

Object Detector Node

```
if len(results) > 0 and len(results[0]) > 0 :
    det_label , det_conf, det_xmin = results[0][ : , 0 ] ,
        results[0][: , 1 ] , results[0][: , 2 ]
    det_ymin , det_xmax , det_ymax = results[0][: , 3 ] ,
        results[0][: , 4 ] , results[0][: , 5 ]

    top_indices = [ i for i , conf in enumerate(det_conf) if
        conf >= self.conf_thresh]

    top_conf = det_conf[top_indices]
    top_label_indices = det_label[top_indices]. tolist()
    top_xmin , top_ymin , top_xmax , top_ymax = det_xmin [
        top_indices] , det_ymin [top_indices], det_xmax [
        top_indices], det_ymax [top_indices]
```

Object Detector Node

```
for i in range ( top_conf.shape[0] ) :
    xmin , ymin = top_xmin [ i ] , top_ymin [ i ]
    xmax , ymax = top_xmax [ i ] , top_ymax [ i ]
    bbox = BoundingBox ( header=Header ( stamp=timestamp ) ,
            x_min=xmin , x_max=xmax , y_min=ymin , y_max=ymax)
    self.pub_bbox.publish( bbox )
```

References

[BLVR] https://bluerobotics.com/store/rov/bluerov2

[ABBR] https://new.abb.com/products/robotics/industrial-robots/irb-120

[RASP] https://www.raspberrypi.org/products

[ASHR] https://www.asrock.com/mb/AMD/X570%20Extreme4/index.asp

[NVID] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[CTKN] https://upload.wikimedia.org/wikipedia/commons/8/83/Ostrya_carpinifolia_in_Italy_male_catkins.jpg

[RRSham] Redmond R. Shamshiri, Ibrahim A. Hameed, et.al, *“Robotic Harvesting of Fruiting Vegetables: A Simulation Approach in V-Rep, ROS and MATLAB*”, Automation in Agriculture - Securing Food Supplies for Future Generations, InTechOpen, Feb, 2018

[RQTG] https://roboticsbackend.com/rqt-graph-visualize-and-debug-your-ros-graph

[GZBO] https://en.wikipedia.org/wiki/File:Gazebo_screenshot_v5.0.png

[BLVR] https://bluerobotics.com/store/rov/bluerov2

[ABBR] https://new.abb.com/products/robotics/industrial-robots/irb-120

References

References for Further Reading

Book 1 : *“Robot Operating System for Absolute Beginners, Robotics Programmings Made Easy”,* Lentin Joseph, Apress, 2018

Book 2 : “*Robot Operating System – The Complete Reference*”, Anis Koubaa, Springer, Vol 1, 2016

Book 3 : “*Robot Operating System – The Complete Reference*”, Anis Koubaa, Springer, Vol 2, 2017

Book 4 : “*ROS By Example, A Do-it Yourself Guide to the Robot Operating System*”, R. Patrick Goebel, Vol 1, 2012

ROS Wiki Link : http://wiki.ros.org

OpenCV Link : https://opencv.org

Gazebo Link : http://gazebosim.org

Robots with ROS: https://robots.ros.org

Q & A

Thank you very much for your attention!

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