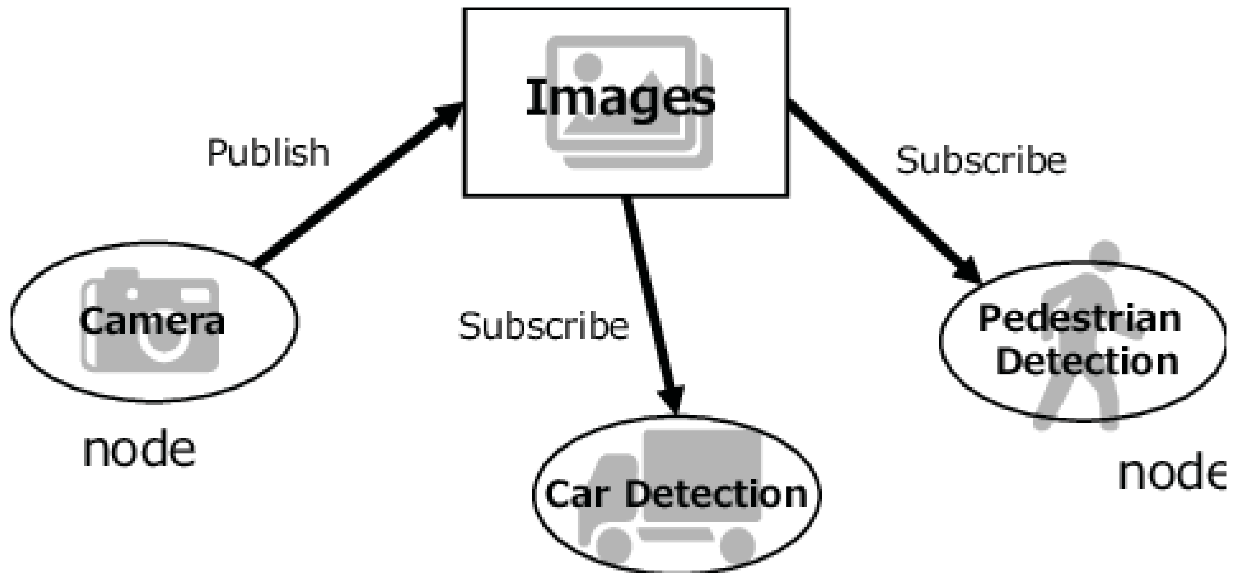
# PUBLISHER & SUBSCRIBER IMPLEMENTATION

Prepared by: Eng. Miriam Hani

# REMEMBER…

- Publisher node: A ROS node that generates information is called a publisher. A publisher sends information to nodes through topics.

- Subscriber node: A ROS node that receives information is called a subscriber. It's subscribed to information in a topic and uses topic callback functions to process the received information.



## PUBLISHER & SUBSCRIBER USING PYTHON:

- Objective:
  - Write a python file that initializes a talker node and publishes on it the string msg "Hello World"
  - Write a python file that initializes a listener node and subscribes to the talker node and prints out the msg received

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

import rospy
from std_msgs.msg import String

def talker():

    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz

    while not rospy.is_shutdown():
        hello_str = "hello world"
        pub.publish(hello_str)
        rate.sleep()

if __name__== '_main_':

    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

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

def callback(data):
      print ("I heard " + data.data)

def listener():

     rospy.init_node('listener', anonymous=True)

     rospy.Subscriber("chatter", String, callback)

     # spin() simply keeps python from exiting until this node is stopped
     rospy.spin()

if __name__== '_main_':
     listener()
```

## LET'S PRACTICE

- Implement publisher-subscriber communication between two nodes to send numbers.

## PUBLISHER CODE:

#include "ros/ros.h“

• Include that includes all the headers necessary to use the most common public pieces of the ROS system.

#include "std_msgs/String.h“

• This includes the std_msgs/String message, which resides in the std_msgs package. This is a header generated automatically from the String.msg file in that package.

ros::init(argc, argv, "talker");

• Initialize ROS.

ros::NodeHandle nh;

• Create a handle to this process' node.

ros::Publisher chatter_pub = nh.advertise<std_msgs::String>("chatter", 1000);

• Tell the master that we are going to be publishing a message of type std_msgs/String on the topic chatter. This lets the master tell any nodes listening on chatter that we are going to publish data on that topic. The second argument is the size of our publishing queue. In this case if we are publishing too quickly it will buffer up a maximum of 1000 messages before beginning to throw away old ones.

• NodeHandle::advertise() returns a ros::Publisher object, which serves two purposes:

1) it contains a publish() method that lets you publish messages onto the topic it was created with, and
2) when it goes out of scope, it will automatically unadvertise.

ros::Rate loop_rate(10);

• allows you to specify a frequency that you would like to loop at

std_msgs::Int32 count;

Create a variable count of type Int32

Count.data=0

Initialize count variable

while (ros::ok())

{

• Loop untill Ctrl+C handling.

```
chatter_pub.publish(count);
```

• Now we actually broadcast the message to anyone who is connected.

```
ros::spinOnce();
```

• For trigering callbacks, not needed in this program.

```
loop_rate.sleep();
```

• Now we use the ros::Rate object to sleep for the time remaining to let us hit our 10hz publish rate.

```
++count.data;
}
Return 0;
}
```

## SUBSCRIBER CODE:

```
#include <ros/ros.h>
#include <std_msgs/Int32.h>
 void counterCallback(const std_msgs::Int32::ConstPtr& msg)
```

Define a function called 'callback' that receives a parameter named 'msg'

```
{
ROS_INFO("%d" , msg->data);
```

Print the value 'data' inside the 'msg' parameter

```
 }
int main(int argc, char **argv)
{
ros::init(argc, argv, "topic_subscriber");
```

Initiate a Node called 'topic_subscriber’

```
ros::NodeHandle nh;
ros::Subscriber sub = nh.subscribe("counter" , 1000, counterCallback);
```

Create a Subscriber object that will listen to the /counter topic and will call the 'callback' function each time it reads something from the topic

```
 ros::spin();
```

Create a loop that will keep the program in execution

```
return 0;
}
```

## TASK:

- Implement a publisher-subscriber code using cpp to send “hello”
- Modify the code to let the publisher send “hello <your name> “