# Intelligent Robots Lab

PROF. QI HAO

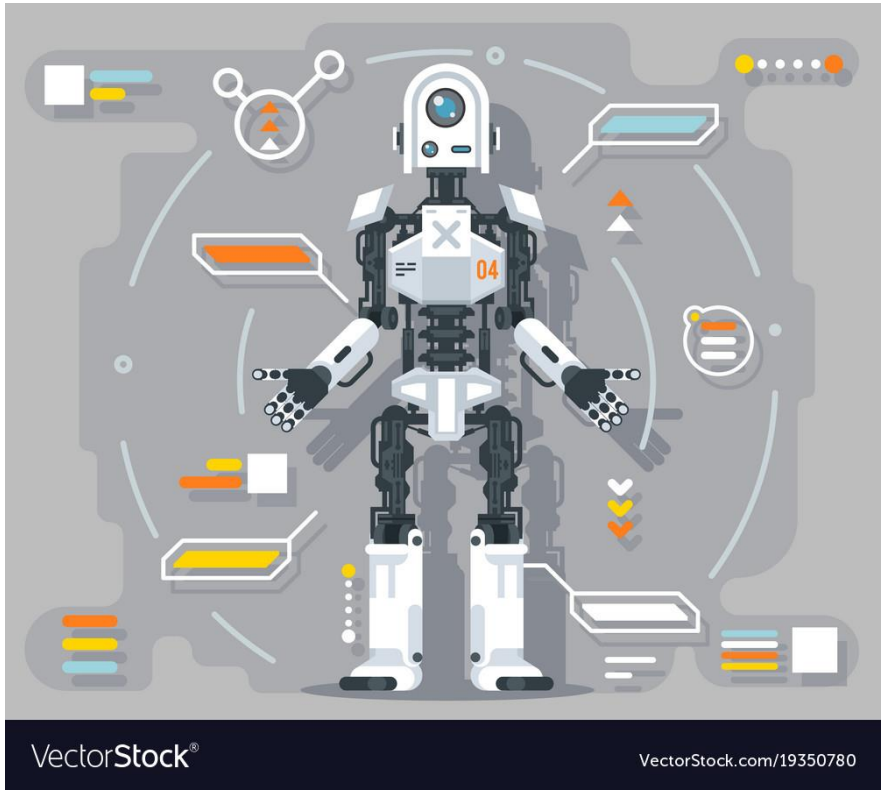
02/19/2019

- ROS
- MATLAB Robotics Toolbox
- TurtleBot
- Gazebo

# A brief introduction to ROS

- What is ROS?

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.


http://www.ros.org/

## History and version

Started from Stanford University and Willow Garage

| Versions | Date |
|---|---|
| Lunar Loggerhead | 2017.5 |
| Kinetic Kame | 2016.5 |
| Jade Turtle | 2015.5 |
| Indigo Igloo | 2014.7 |
| Hydro Medusa | 2013.9 |
| Groovy Galapagos | 2012.12 |
| Fuerte Turtle | 2012.4 |
| Electric Emys | 2011.8 |

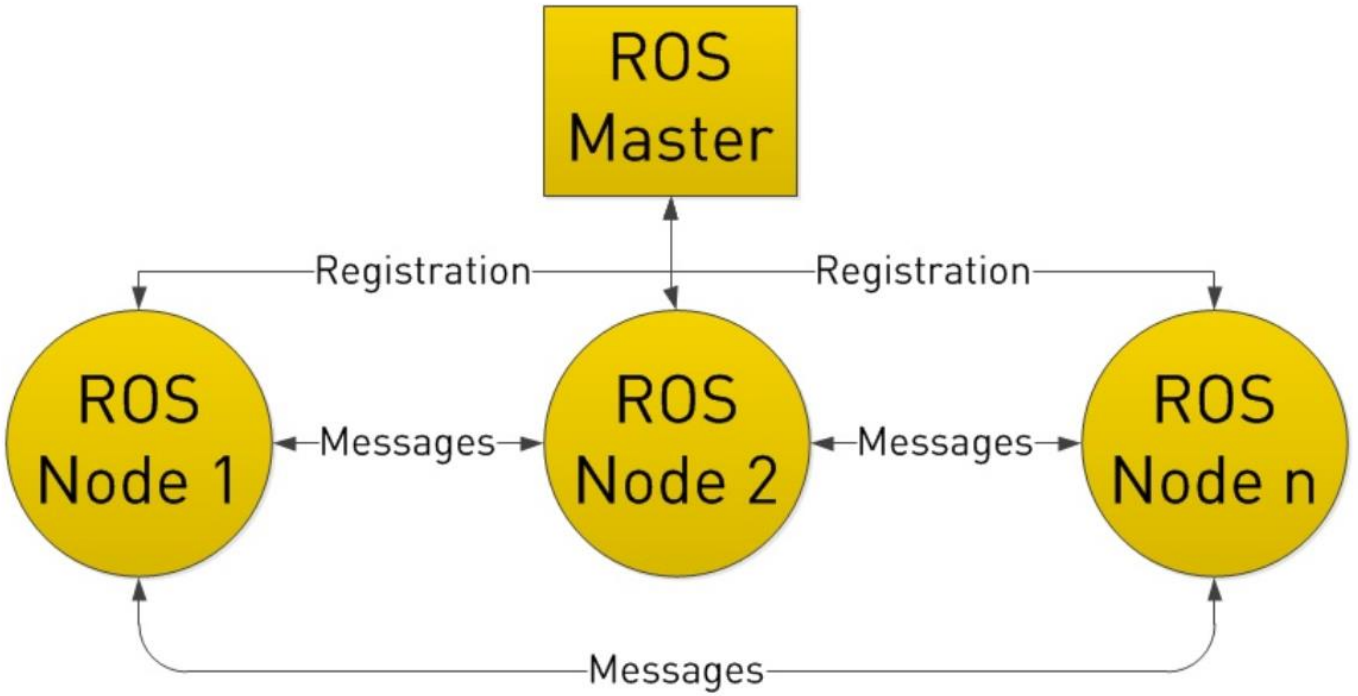

**Request Ubuntu 16.04**

## How ROS works?

ROS is composed by a pile of files. When it runs, a sequence of nodes are loaded. Among them, a special node called Master is important and should be loaded first, using the following command.

```
roscore
```

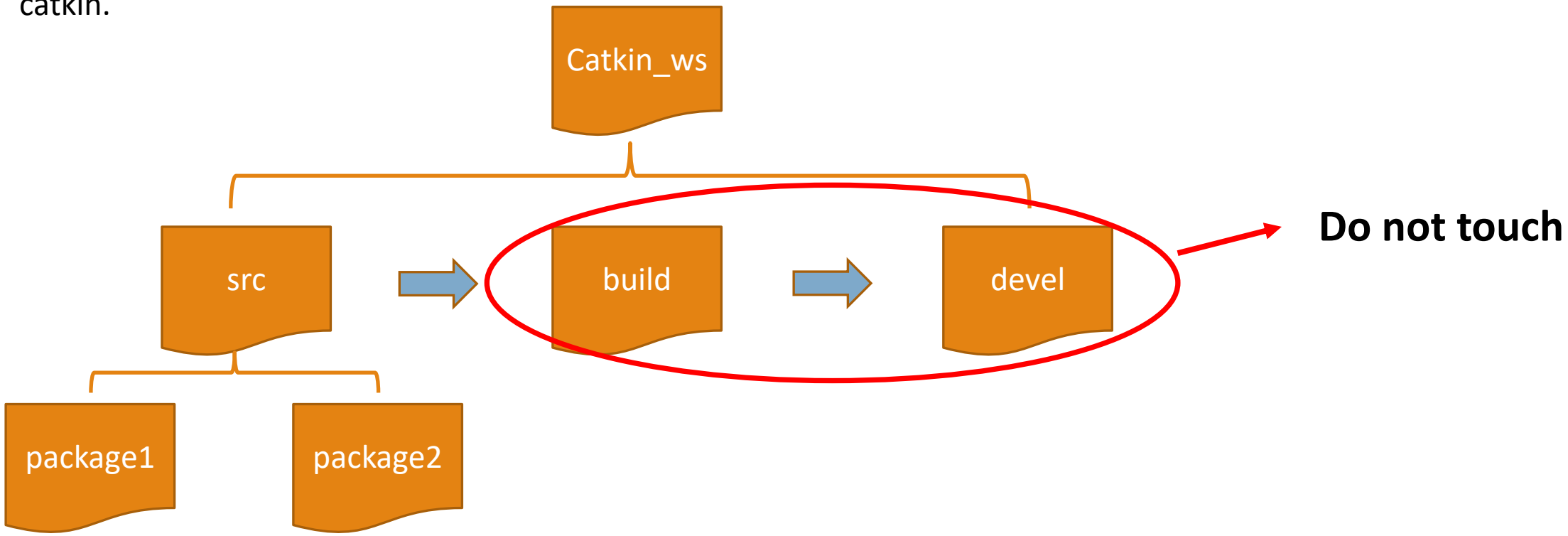
Master organizes other nodes. Every other node does a specific job like collect data from sensor, process data and control the wheel… They cannot communicate with each other before registering with Master.



# ROS: file system

- Catkin work space

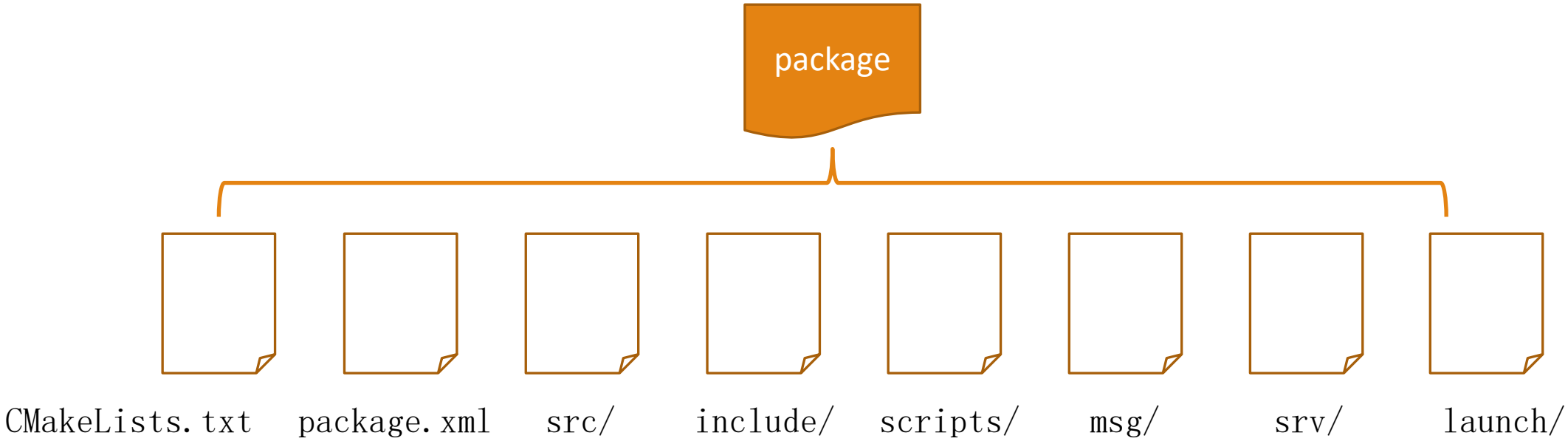
  Our own code (source code) will be built in a directory called Catkin Work Space, because it is built by catkin.



# ROS: file system

- Package

  Our own code (source code) will be built in a directory called Catkin Work Space, because it is built by catkin.



# ROS: catkin

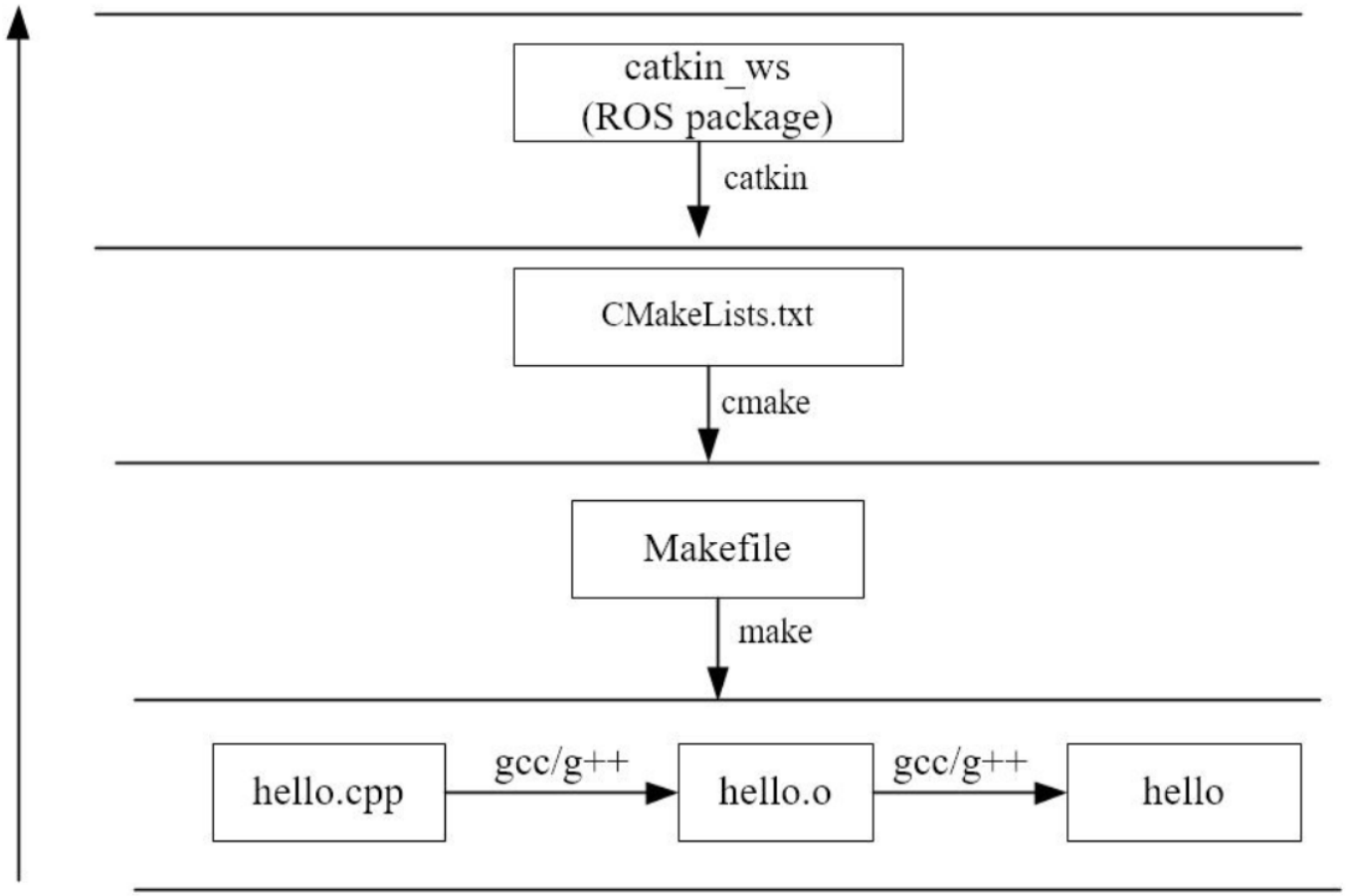
- Build with catkin

  Catkin is a powerful and useful tool to build our source code. It is a higher level encapsulation of Cmake.

  e.g. in a catkin work space (catkin_ws), to build our source code, we just need one command:

  $ cd ~/catkin_ws

```
$ catkin_make
```



# Task 1

Complete the tutorial of Beginner Level on

*http://wiki.ros.org/ROS/Tutorials*

from 1 to 8

Tips:

When you use *apt-get install* to install ROS, choose the version:

```
sudo apt-get install ros-kinetic-desktop-full
```

It integrates useful ROS packages.

## 1.1 Beginner Level

1. Installing and Configuring Your ROS Environment

   This tutorial walks you through installing ROS and setting up the ROS e
2. Navigating the ROS Filesystem

   This tutorial introduces ROS filesystem concepts, and covers using the
3. Creating a ROS Package

   This tutorial covers using roscreate-pkg or catkin to create a new packa
4. Building a ROS Package

   This tutorial covers the toolchain to build a package.
5. Understanding ROS Nodes

   This tutorial introduces ROS graph concepts and discusses the use of r
   tools.
6. Understanding ROS Topics

   This tutorial introduces ROS topics as well as using the rostopic and rqt
7. Understanding ROS Services and Parameters

   This tutorial introduces ROS services, and parameters as well as using
   tools.
8. Using rqt_console and roslaunch

   This tutorial introduces ROS using rqt_console and rqt_logger_level for
   nodes at once. If you use ROS fuerte or ealier distros where rqt isn't fu
   page that uses old rx based tools.
9. Using rosed to edit files in ROS

# MATLAB Robotics System Toolbox

- What can Robotics System Toolbox do?

  **Design and test algorithms for robotics applications**

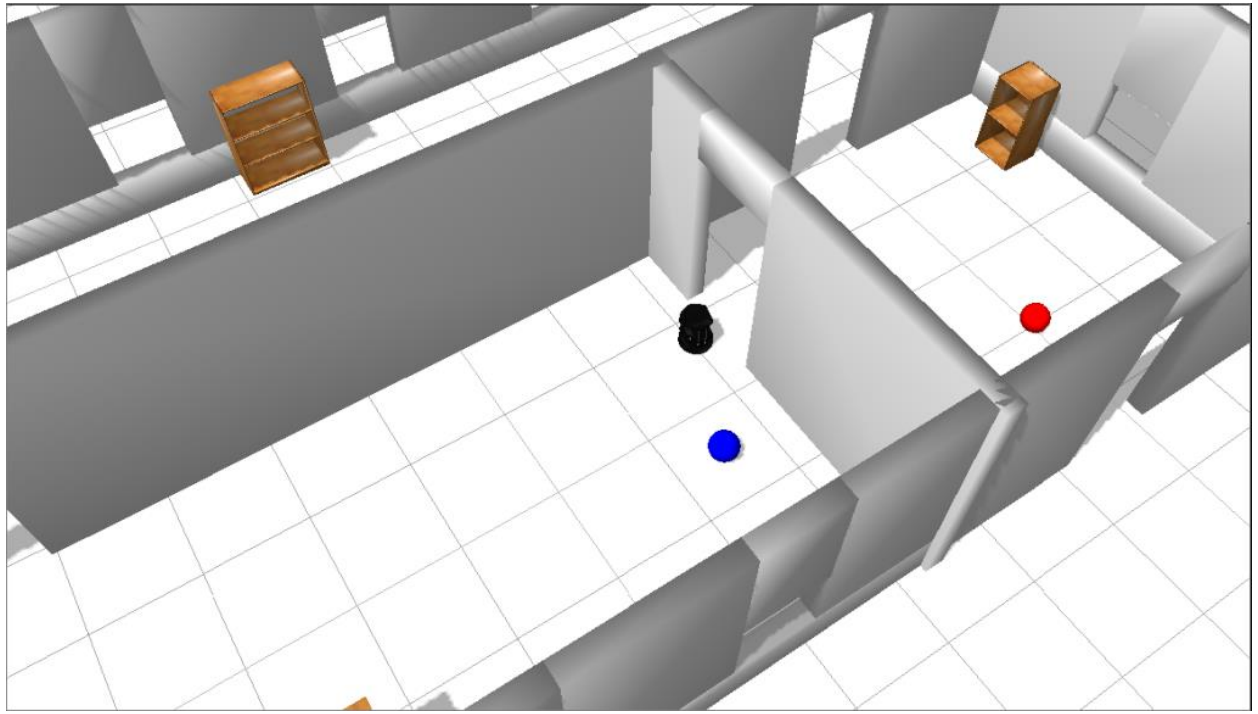
  provides algorithms and hardware connectivity for developing autonomous robotics applications for aerial and ground vehicles, manipulators, and humanoid robots.

- Key features

  - Map utilization, path planning, path following, and state estimation algorithms
  - Robot localization and environment mapping using lidar sensors
  - Rigid body tree kinematics and dynamics algorithms
  - Bidirectional communication with live ROS-enabled robots
  - rosbag data import, message extraction, and coordinate transformation
  - ROS node generation from Simulink models (with Simulink Coder™)

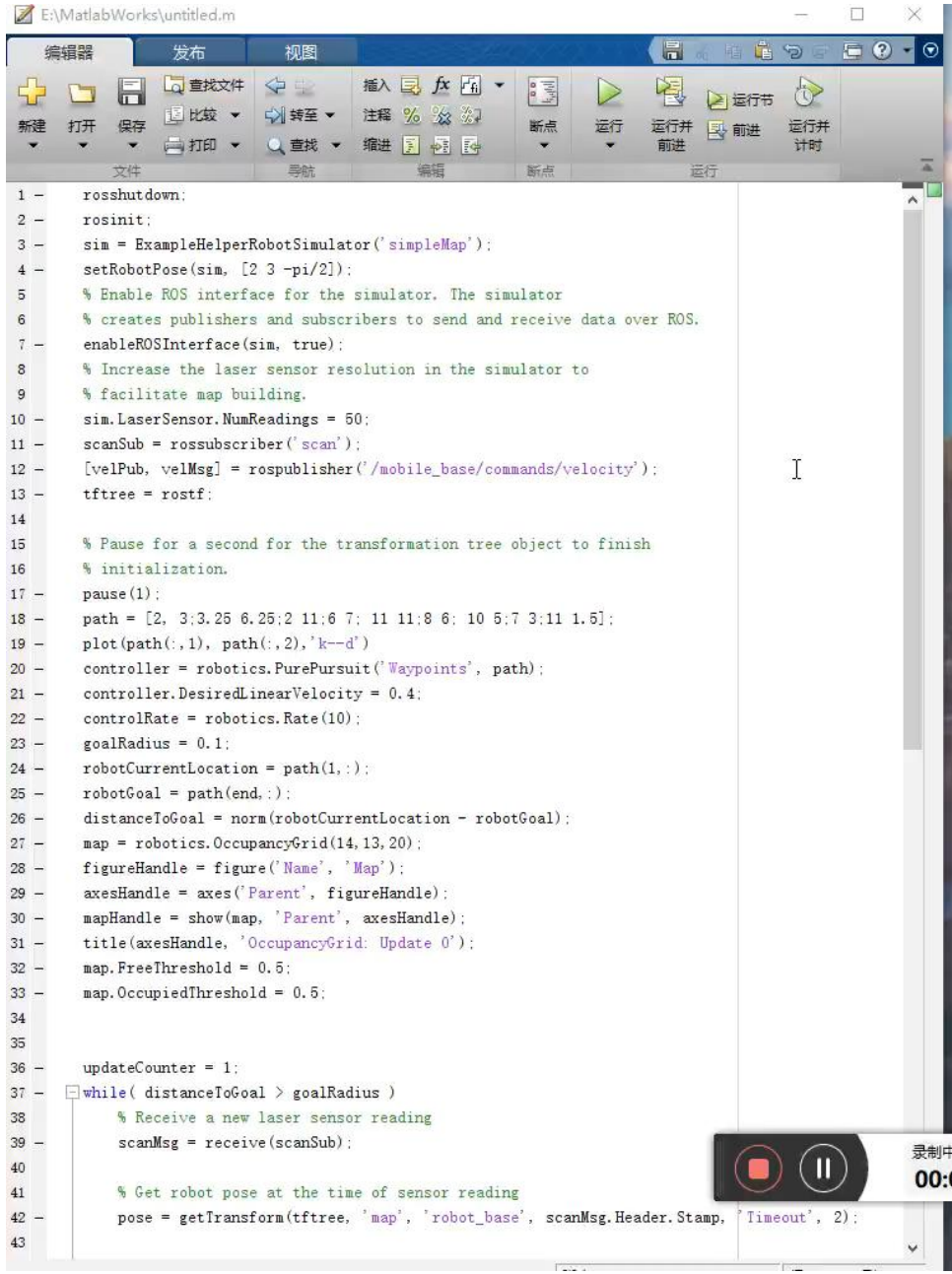
# MATLAB Robotics System Toolbox

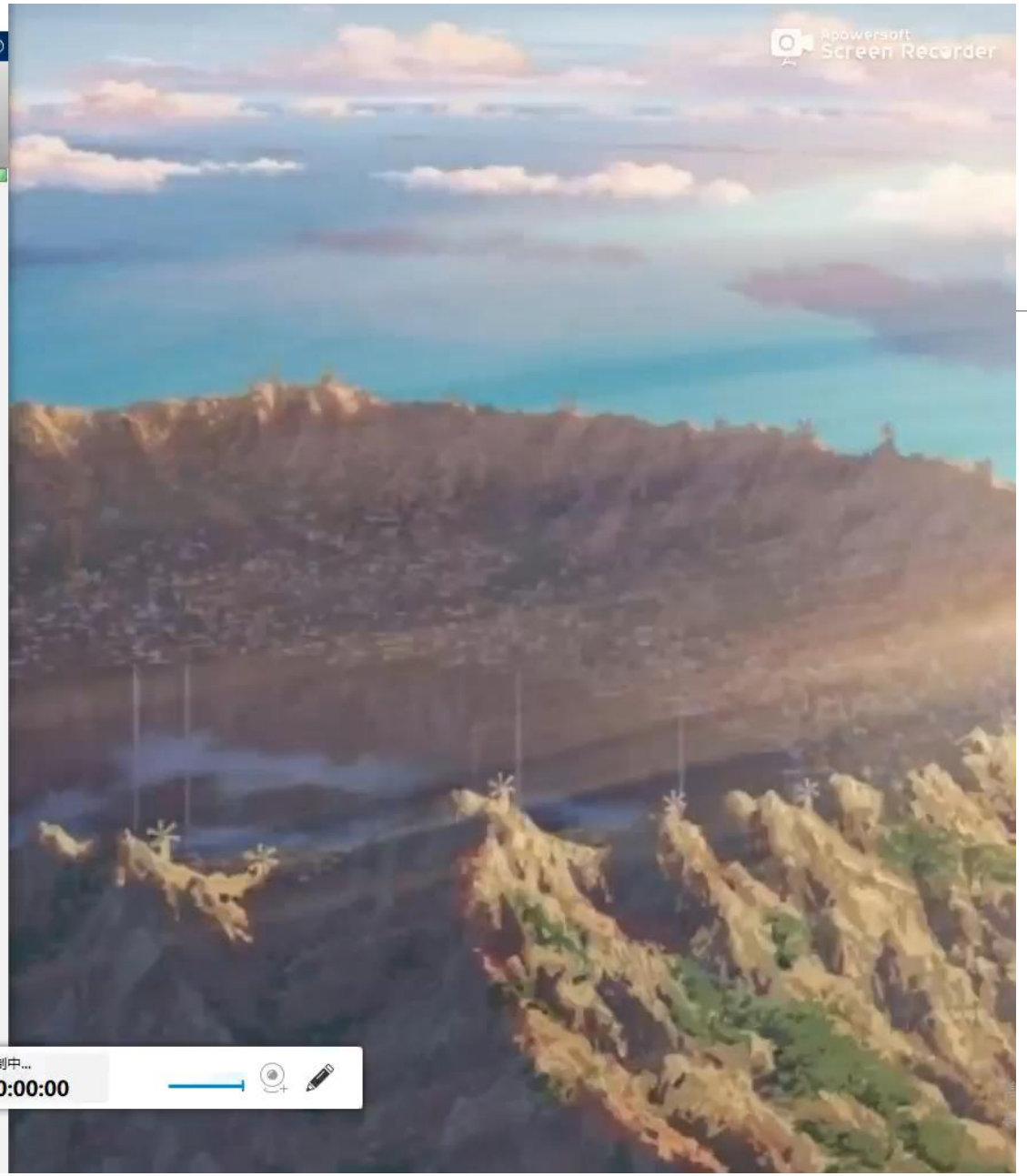
- Implement algorithms both in virtual environment and real robot



## **Robotics System Toolbox** Demo: Mapping With Known Poses

This example shows how to create a map of the environment using range sensor readings if the position of the robot is known at the time of sensor reading.


E:\MatlabWorks\untitled.m
编辑器
发布
视图
新建
打开
保存
查找文件
比较
打印
转至
查找
插入
注释
缩进
断点
运行
运行并前进
运行节
前进
运行并计时
文件
导航
编辑
断点
运行
rosshutdown;
rosinit;
sim = ExampleHelperRobotSimulator('simpleMap');
setRobotPose(sim, [2 3 -pi/2]);
% Enable ROS interface for the simulator. The simulator
% creates publishers and subscribers to send and receive data over ROS.
enableROSInterface(sim, true);
% Increase the laser sensor resolution in the simulator to
% facilitate map building.
sim.LaserSensor.NumReadings = 50;
scanSub = rossubscriber('scan');
[velPub, velMsg] = rospublisher('/mobile_base/commands/velocity');
tftree = rostf;
% Pause for a second for the transformation tree object to finish
% initialization.
pause(1);
path = [2, 3;3.25 6.25;2 11;6 7; 11 11;8 6; 10 5;7 3;11 1.5];
plot(path(:,1), path(:,2),'k--d')
controller = robotics.PurePursuit('Waypoints', path);
controller.DesiredLinearVelocity = 0.4;
controlRate = robotics.Rate(10);
goalRadius = 0.1;
robotCurrentLocation = path(1,:);
robotGoal = path(end,:);
distanceToGoal = norm(robotCurrentLocation - robotGoal);
map = robotics.OccupancyGrid(14,13,20);
figureHandle = figure('Name', 'Map');
axesHandle = axes('Parent', figureHandle);
mapHandle = show(map, 'Parent', axesHandle);
title(axesHandle, 'OccupancyGrid: Update 0');
map.FreeThreshold = 0.5;
map.OccupiedThreshold = 0.5;
updateCounter = 1;
while( distanceToGoal > goalRadius )
% Receive a new laser sensor reading
scanMsg = receive(scanSub);
% Get robot pose at the time of sensor reading
pose = getTransform(tftree, 'map', 'robot_base', scanMsg.Header.Stamp, 'Timeout', 2);


Apowersoft Screen Recorder
录制中...
00:00:00

# Task 2

Complete the **first 2** sections of tutorial in *Getting Start* at

*https://ww2.mathworks.cn/help/robotics/index.html?searchHighlight=robotics&s_tid=doc_srchtitle*

1

## Robotics System Toolbox

Design and test algorithms for robotics applications

2

Robotics System Toolbox™ provides algorithms and hardware connec
applications for aerial and ground vehicles, manipulators, and humanc
planning and path following for differential drive robots, scan matchir
manipulator robots, the system toolbox includes algorithms for invers
using a rigid body tree representation.

The system toolbox provides an interface between MATLAB® and Sim
that enables you to test and verify applications on ROS-enabled robo
examples showing how to work with virtual robots in Gazebo and witl

Robotics System Toolbox supports C++ code generation, enabling yo
and automatically deploy it to a ROS network. Support for Simulink ex
parameters while your deployed model is running.

**Getting Started**
Learn the basics of Robotics System Toolbox

**Coordinate System Transformations**
Units, coordinate conversion functions

**Robot Operating System (ROS)**
Access ROS networks, robots, and simulators

**Sensor Data**
Collect and analyze sensor data utilizing ROS messages

## Getting Started with Robotics System Toolbox

Robotics System Toolbox Product Description
System Requirements

### Tutorials

3

**Get Started with ROS**
Robot Operating System (ROS) is a communication interface that e

**Connect to a ROS Network**
A ROS network consists of a single *ROS master* and multiple *ROS n*

**Explore Basic Behavior of the TurtleBot**
This example helps you to explore basic autonomy with the TurtleB

**Track and Follow an Object**
In this example, you explore autonomous behavior that incorporate

# TurtleBot

- What is TurtleBot

TurtleBot is a low-cost, personal robot kit with open-source software.

With TurtleBot, you'll be able to build a robot that can drive around your house, see in 3D, and have enough horsepower to create exciting applications.

In addition to the TurtleBot hardware kit, users can download the TurtleBot SDK from the ROS wiki.

Support ROS wiki website and tutorial:
*http://wiki.ros.org/Robots/TurtleBot*


- We will see a demo using **TurtleBot 2** to implement a SLAM algorithm called gmapping, which is comprised in

```
ros-kinetic-desktop-full
```

# Gazebo


- What is Gazebo?

  Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.

- A few key features of Gazebo include:

  - multiple physics engines,
  - a rich library of robot models and environments,
  - a wide variety of sensors, IMU, LIDAR, and camera...
  - convenient programmatic and graphical interfaces

- We can use Gazebo to:

  - testing robotics algorithms,
  - designing robots,
  - performing regression testing with realistic scenarios

# Gazebo & ROS


- Integration of ROS and Gazebo

  Although Gazebo is a stand-alone application which can be used independently ROS, the integration makes them stronger!

```
sudo apt-get install ros-kinetic-desktop-full
```

***That includes relative versions of Gazebo***

- Metapackage *gazebo_ros_pkgs*

  A set of packages provides wrappers around the stand-alone Gazebo.

  They provide the necessary interfaces to simulate a robot in Gazebo using ROS messages, services and dynamic reconfigure.

```
gazebo
```

***Two ways to run Gazebo, one by it alone, the other from ROS***

```
rosrun gazebo_ros gazebo
```

# Gazebo example: TurtleBot3

ROBOTIS e-Manual

- Learn more about Gazebo...

  http://gazebosim.org/tutorials

- A Gazebo application example

  http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#

  In this application, we show the convenience and power of Gazebo in simulation of SLAM and navigation.

ROBOTIS e-Manual

# Gazebo example: TurtleBot3

# Tasks 3 (optional)

Following the instructions on website: *http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#*

and complete **until 11.2.1.6 (include).**

Tips: the **remote PC** is your own PC, it is in terms of the real TurtleBot .