AAE3004 Dynamical Systems and Control

Lab 1a: Autonomous ROS car control project

Instructors: Dr Bing XU, Mr Jian LIU (TA), and Mr Sergio VICENZO (TA)
Research Assistant Professor
Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University
Office: QR832
Tel: 3400 8010
Email: pbing.xu@polyu.edu.hk

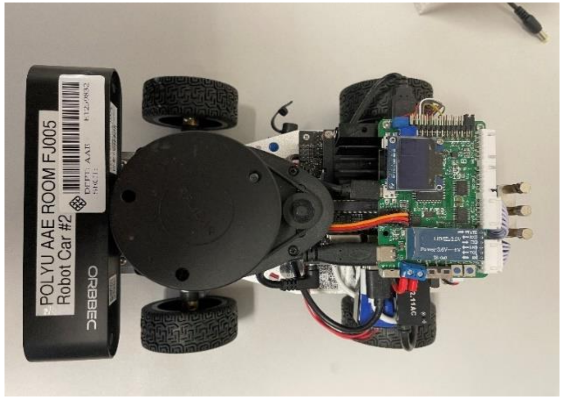
AAE3004

- 1. Robot Car Platform Overview
- 2. Linux Operating System
- 3. ROS

Part I – Overview of ROS Car Platform

1. Robot Car Platform Overview

- raspberry pie and STM32 motherboards
- Supporting ros2 / opencv
- Multiline lidar
- Visual processing
- deep learning framework (Yolo/Tensorflow)
- navigation and map building



AAE3004

Part II – Linux Operating System

2. Linux Operating System (OS)

What is Operating system?

- windows system for PC,
- IOS system for Apple products,
- Android system for mobile devices

2. Linux Operating System (OS)

What is Operating system?

Operating System is a ***program*** that mediates between the ***user*** and the computer ***hardware***.

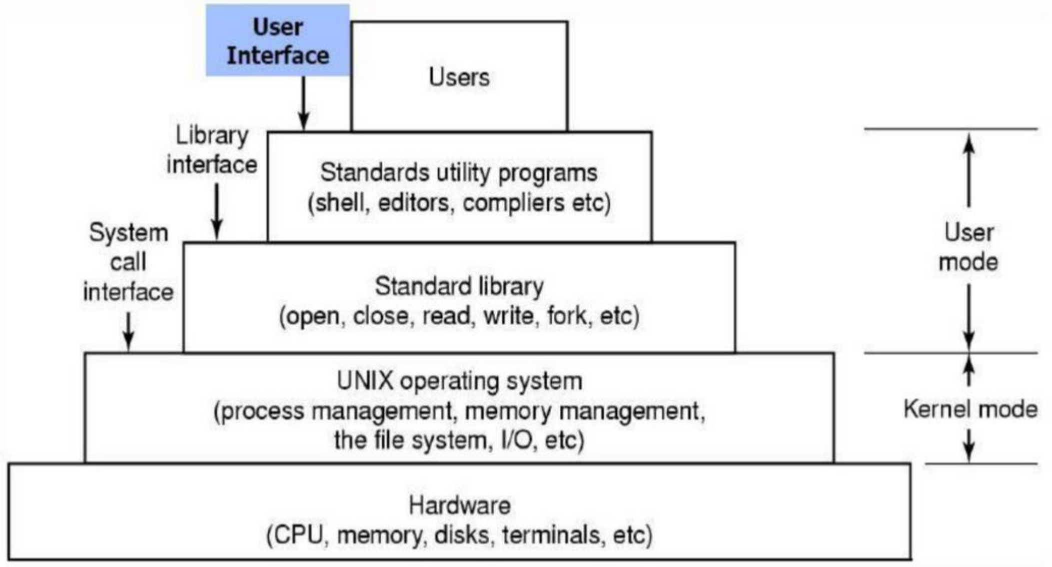
• Hides hardware details.

• Manages software resources
– resources means objects necessary to execute the program, e.g. memory, processor (CPU), input/output, communication ports
– strategies for allocation and deallocation of resources

• Other activities: security, job accounting, error detecting tools, etc.

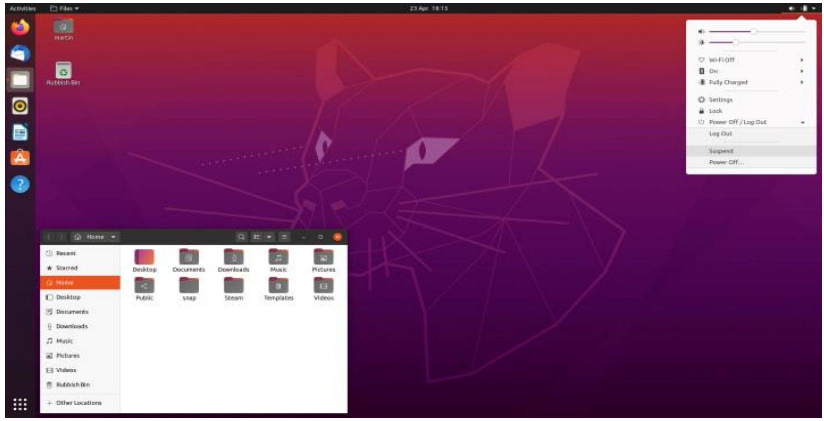
2. Linux Operating System

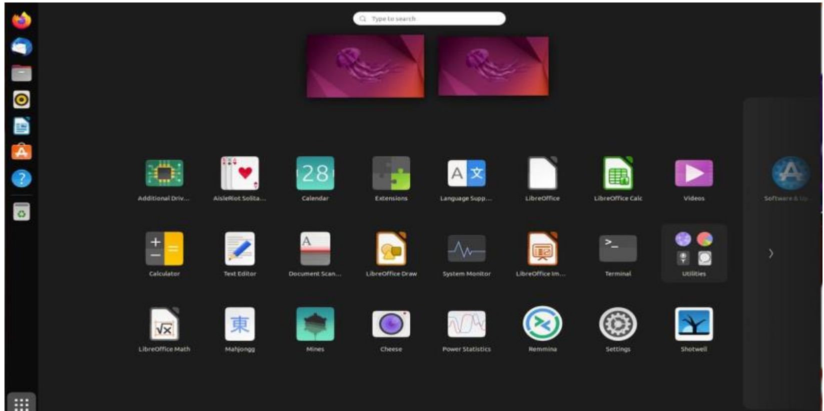
Linux Structure



User Interface
Users
Library interface
Standards utility programs (shell, editors, compliers etc)
System call interface
Standard library (open, close, read, write, fork, etc)
UNIX operating system (process management, memory management, the file system, I/O, etc)
Hardware (CPU, memory, disks, terminals, etc)
User mode
Kernel mode

Graphic User Interface (GUI)





2. Linux Operating System

Linux History

Linux Torvalds, Finland, born in the same year as UNIX, i.e. 1969, creator of the Linux kernel and the Git version control sysem.

Linus Torvalds announcing Linux 1.0, 30.03.1994

Linus Torvalds in 2019

Richard Stallman, founder of the GNU project and the Free Software Foundation, co-creator of the GNU GPL license, creator of the Emacs editor, GCC compiler, GDB debugger.

Richard Stallman in 2019

May 1991, version 0.01: no support for the network, limited number of device drivers, one file system (Minix), processes with protected address spaces
The Linux Kernel Archives – https://www.kernel.org/
– **2022-02-23**, latest stable version **5.16.11**
– **2022-02-20**, latest mainline **5.17-rc5**

2. Linux Operating System

Linux Basic Feature

- **Multi-access** system (with time sharing) and multi-tasking
- **Multi-process** system, simple mechanisms to create hierarchy of processes, kernel preemption
- **Available for many architectures**
- Simple standard user interface that can be easily replaced (shell, command interpreter)
- File systems with a multi-level tree structure
- Shared libraries, loaded into memory dynamically (one code used simultaneously by many processes)
- Compliance with the POSIX 1003.1 standard
- Different formats of executable files

Free Distribution!

2. Linux Operating System

Tutorial

1. The Linux Programming Interface: A Linux and UNIX System Programming

The **Linux Programming Interface** offers **in-depth information** about the system and library functionalities.

2. How Linux Works: What Every Superuser Should Know

How Linux Works is a conceptual book that explains brief information about the **Linux internals**. This book also explains how Linux firewalls, interfaces, and networks work.

3. Linux Tutorial for Beginners: Introduction to Linux Operating System

https://www.youtube.com/watch?v=V1y-mbWM3B8

4. Linux Command Line Tutorial For Beginners

https://www.youtube.com/watch?v=YHFzr-akOas&list=PLS1QulWo1RIb9WVQGJ_vh-RQusbZgO_As

5. Ubuntu Complete Beginners Guide (Full Course in one video!)

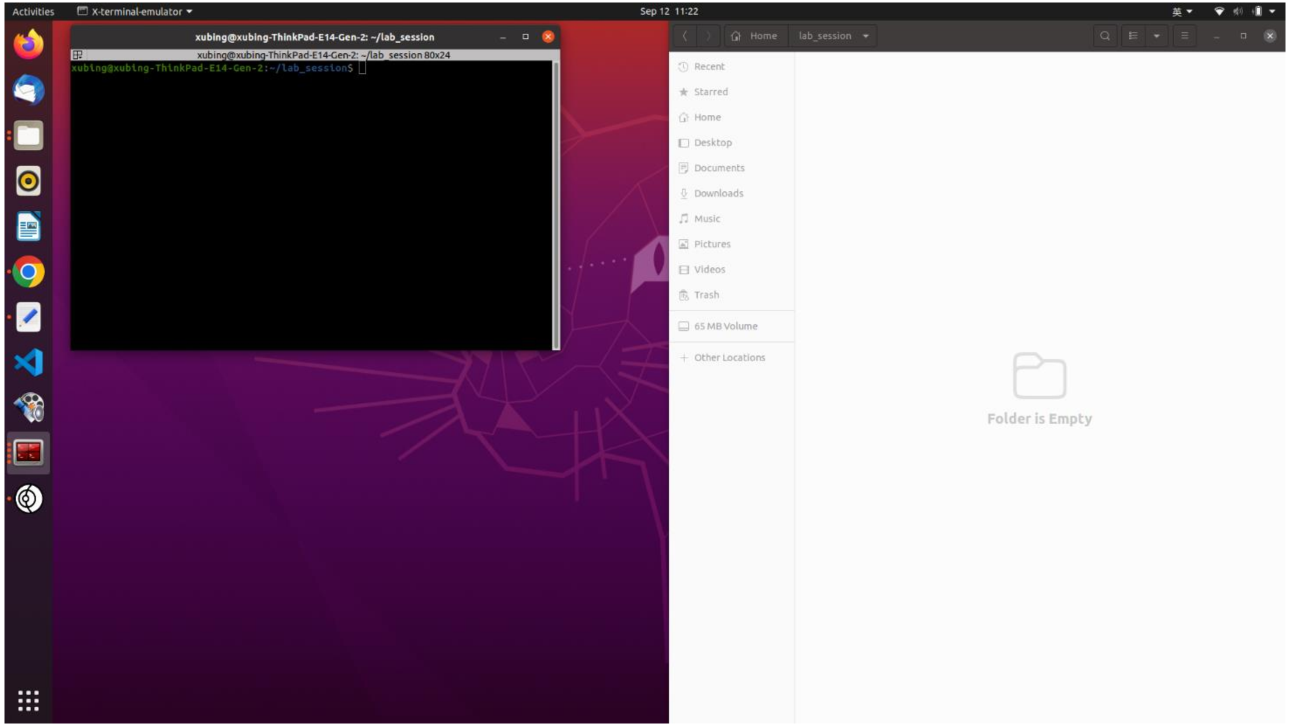
https://www.youtube.com/watch?v=D4WyNjt_hbQ&t=36s

2. Linux Operating System

Helloworld demo with Linux

Step 1: Open the terminal

- **right click on the margin**
- **Ctrl+Alt+T**



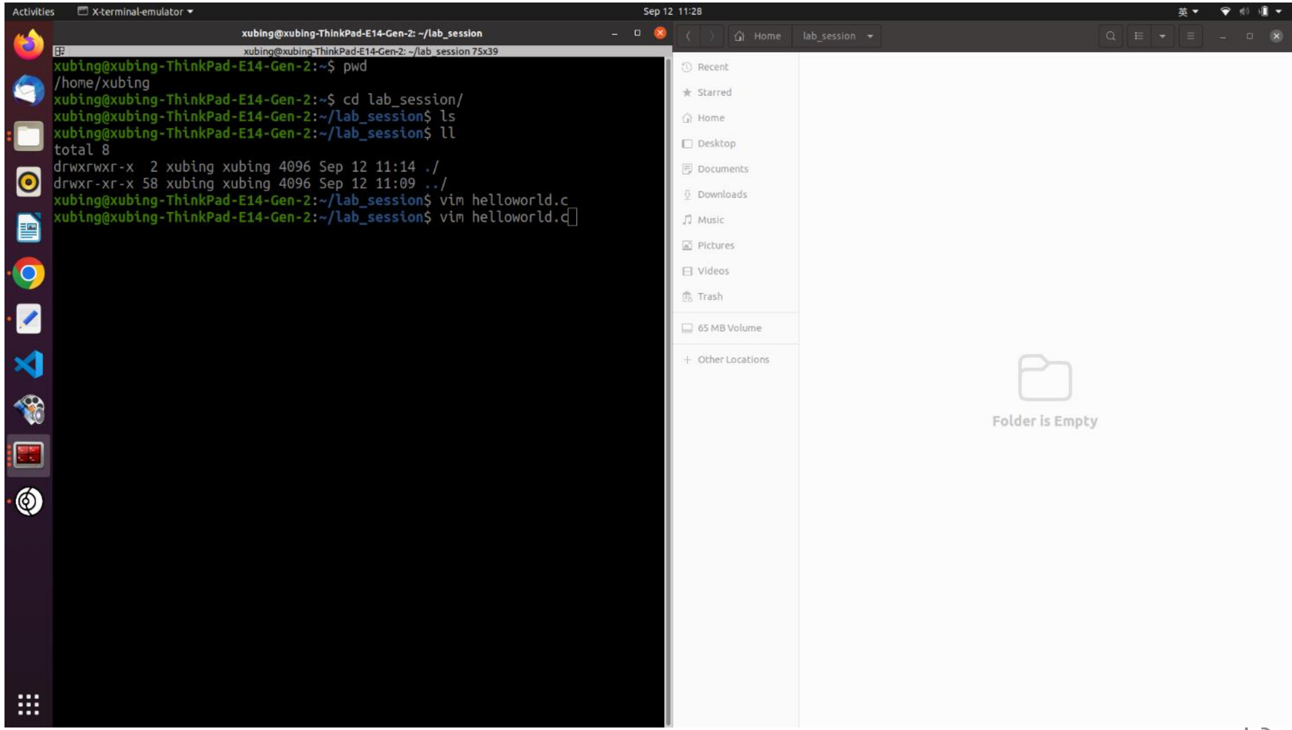
```
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$
```

2. Linux Operating System

Helloworld demo with Linux

Step 2: Create the C file

- $ vim helloworld.c



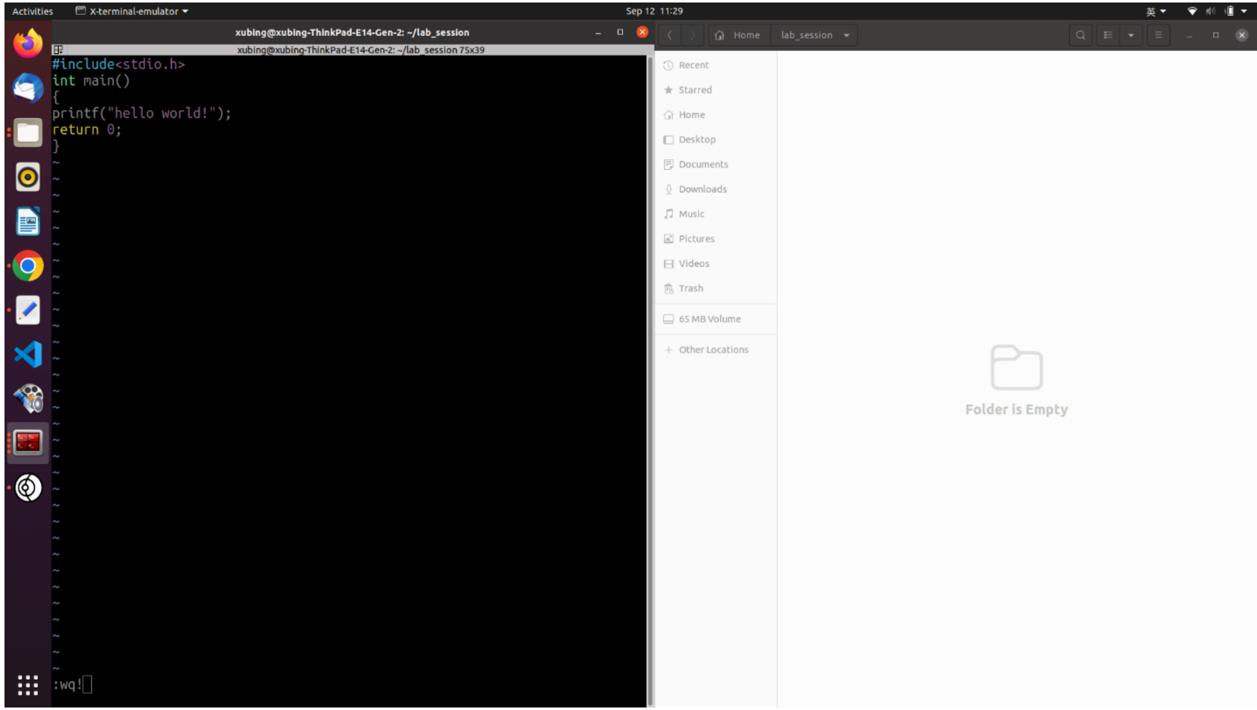
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ pwd
/home/xubing
xubing@xubing-ThinkPad-E14-Gen-2:~$ cd lab_session/
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ls
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ll
total 8
drwxrwxr-x  2 xubing xubing 4096 Sep 12 11:14 ./
drwxr-xr-x 58 xubing xubing 4096 Sep 12 11:09 ../
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
```

2. Linux Operating System

Helloworld demo with Linux

Step 3: Input the code and save it

#include<stdio.h>
Int main()
{
printf("hello world!");
return 0;
}



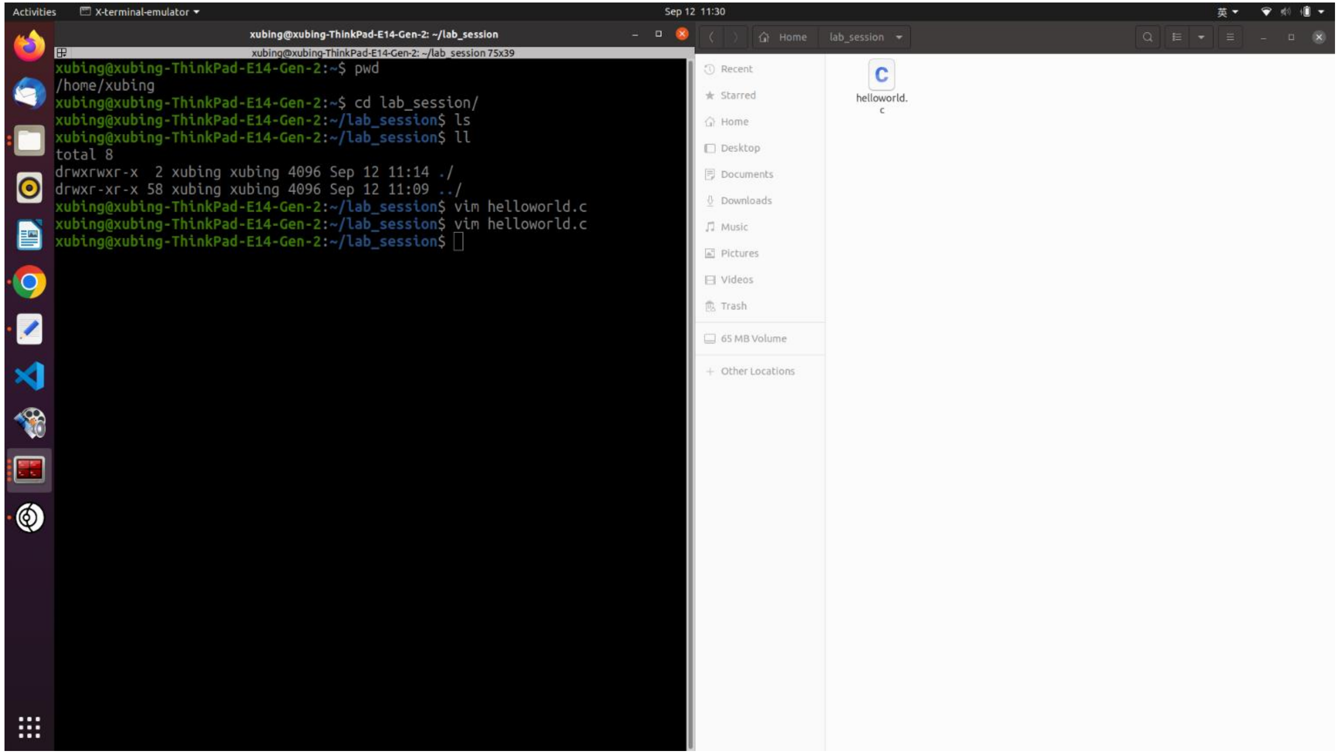
```
xubing@xubing-ThinkPad-E14-Gen-2: ~/lab_session
#include<stdio.h>
int main()
{
printf("hello world!");
return 0;
}
~
:wq!
```

2. Linux Operating System

Helloworld demo with Linux

Step 3: Input the code and save it

#include<stdio.h>
Int main()
{
printf(“hello world!”);
return 0;
}



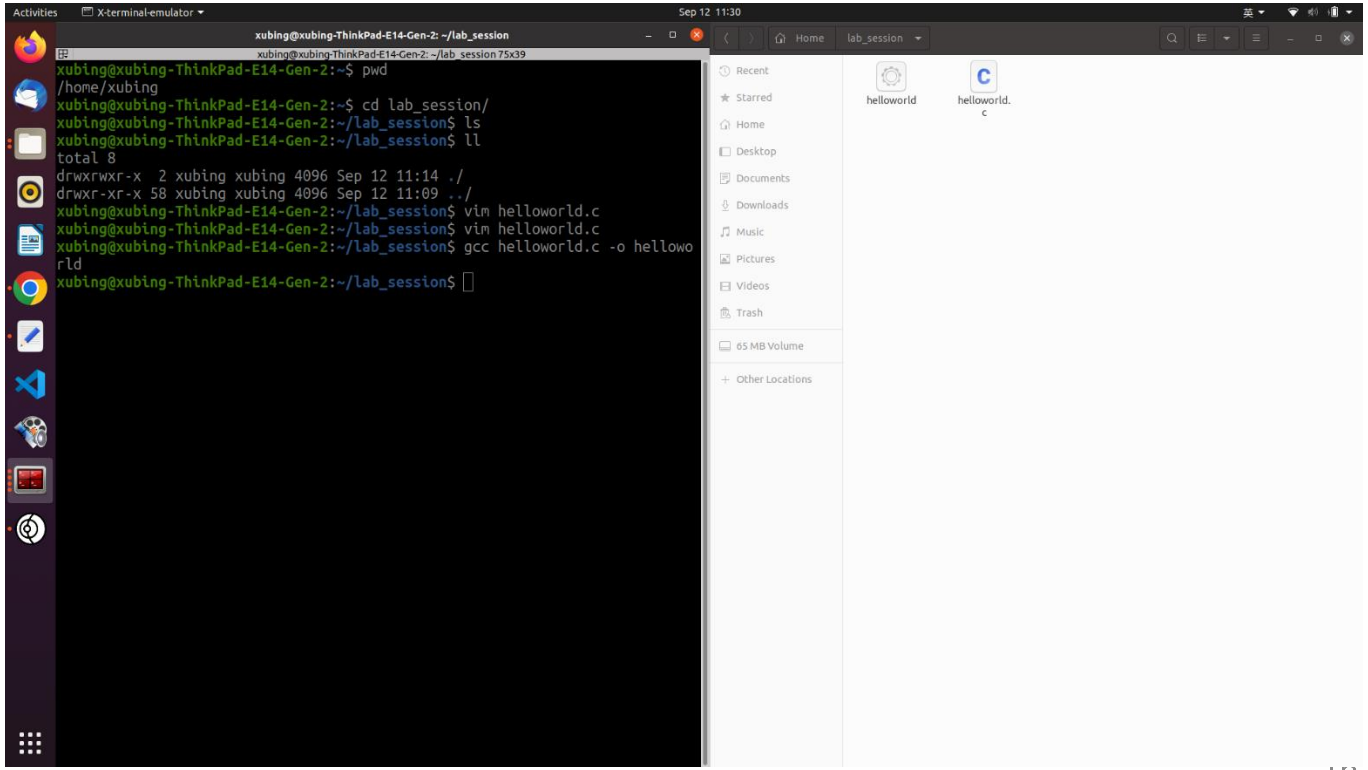
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ pwd
/home/xubing
xubing@xubing-ThinkPad-E14-Gen-2:~$ cd lab_session/
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ls
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ll
total 8
drwxrwxr-x  2 xubing xubing 4096 Sep 12 11:14 ./
drwxr-xr-x 58 xubing xubing 4096 Sep 12 11:09 ../
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ 
```

2. Linux Operating System

Helloworld demo with Linux

Step 4: Compile

- $ gcc helloworld.c –o helloworld



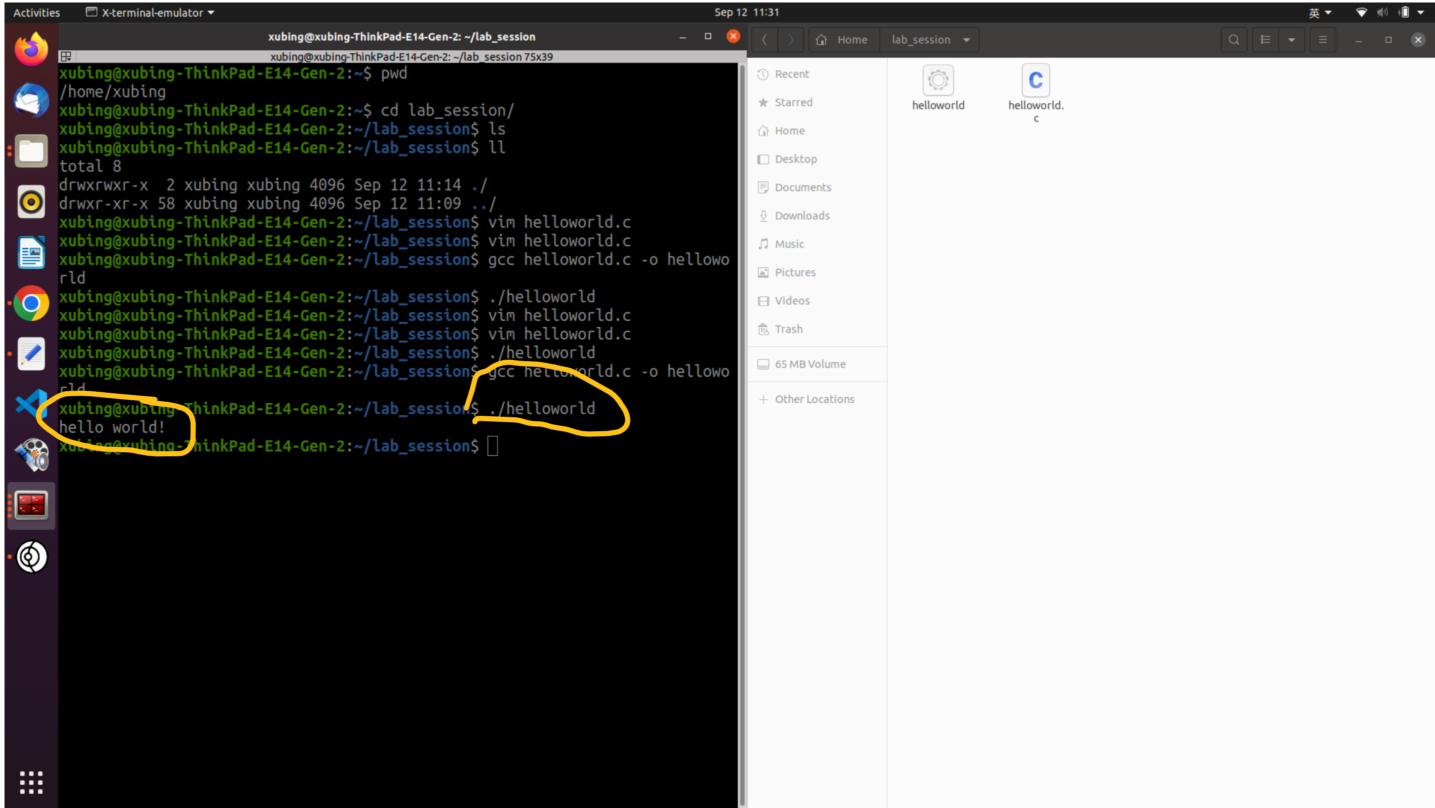
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ pwd
/home/xubing
xubing@xubing-ThinkPad-E14-Gen-2:~$ cd lab_session/
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ls
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ll
total 8
drwxrwxr-x  2 xubing xubing 4096 Sep 12 11:14 ./
drwxr-xr-x 58 xubing xubing 4096 Sep 12 11:09 ../
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ gcc helloworld.c -o hellowo
rld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$
```

2. Linux Operating System

Helloworld demo with Linux

Step 5: Execute

- ./helloworld



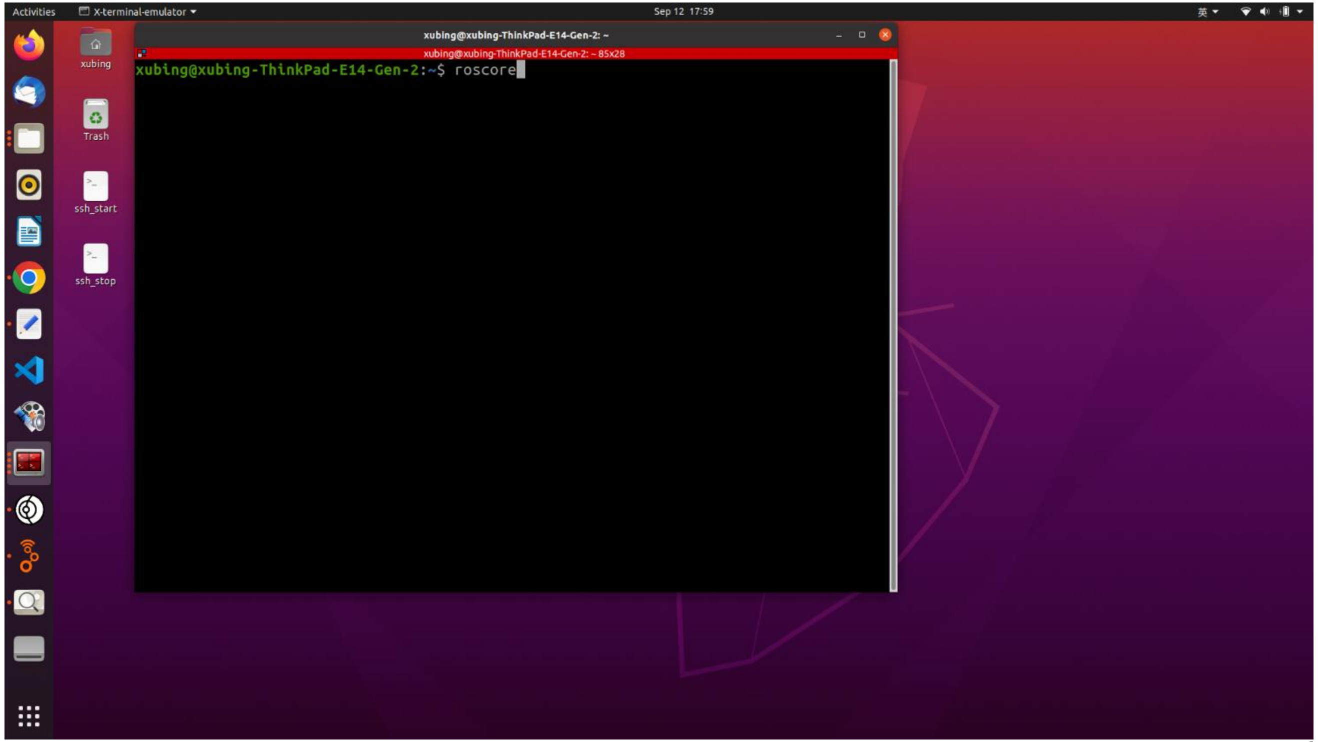
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ pwd
/home/xubing
xubing@xubing-ThinkPad-E14-Gen-2:~$ cd lab_session/
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ls
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ll
total 8
drwxrwxr-x  2 xubing xubing 4096 Sep 12 11:14 ./
drwxr-xr-x 58 xubing xubing 4096 Sep 12 11:09 ../
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ gcc helloworld.c -o hellowo
rld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ./helloworld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ vim helloworld.c
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ./helloworld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ gcc helloworld.c -o hellowo
rld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ./helloworld
hello world!
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$
```

2. Linux Operating System

ROS turtles demo

Step 1: create the master node

- **$ roscore**


Activities
X-terminal-emulator
Sep 12 17:59
xubing
Trash
ssh_start
ssh_stop
xubing@xubing-ThinkPad-E14-Gen-2: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~ 85x28

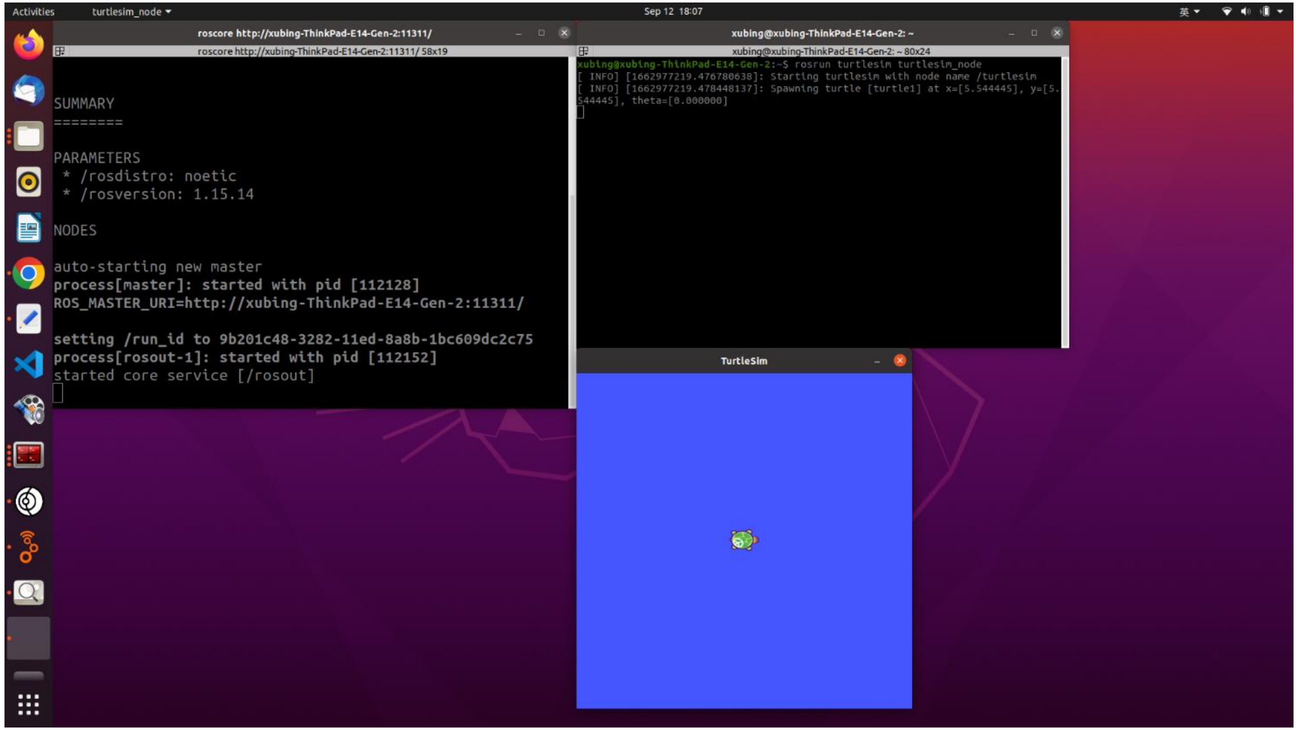
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ roscore
```

2. Linux Operating System

ROS turtles demo

Step 2: create a node to show turtle

- **$ rosrun turtlesim turtlesim_node**


Activities
turtlesim_node
Sep 12 18:07
roscore http://xubing-ThinkPad-E14-Gen-2:11311/
roscore http://xubing-ThinkPad-E14-Gen-2:11311/ 58x19
xubing@xubing-ThinkPad-E14-Gen-2: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~ 80x24
TurtleSim

```
SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES

auto-starting new master
process[master]: started with pid [112128]
ROS_MASTER_URI=http://xubing-ThinkPad-E14-Gen-2:11311/

setting /run_id to 9b201c48-3282-11ed-8a8b-1bc609dc2c75
process[rosout-1]: started with pid [112152]
started core service [/rosout]
```

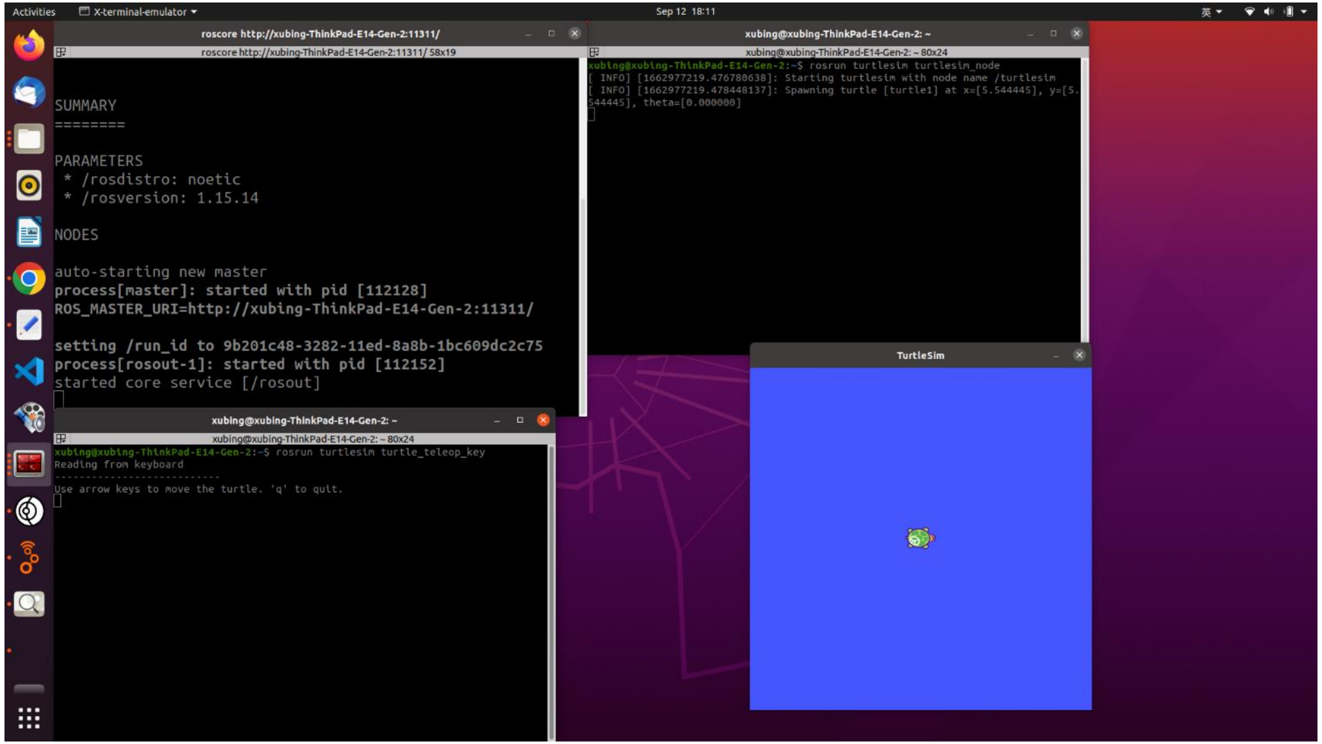
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ rosrun turtlesim turtlesim_node
[ INFO] [1662977219.476780638]: Starting turtlesim with node name /turtlesim
[ INFO] [1662977219.478448137]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

2. Linux Operating System

ROS turtles demo

Step 3: create the keyboard node

- **$ rosrun turtlesim turtlesim_node**


Activities
X-terminal-emulator
Sep 12 18:11
roscore http://xubing-ThinkPad-E14-Gen-2:11311/
roscore http://xubing-ThinkPad-E14-Gen-2:11311/ 58x19
xubing@xubing-ThinkPad-E14-Gen-2: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~ 80x24
TurtleSim

```
SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES

auto-starting new master
process[master]: started with pid [112128]
ROS_MASTER_URI=http://xubing-ThinkPad-E14-Gen-2:11311/

setting /run_id to 9b201c48-3282-11ed-8a8b-1bc609dc2c75
process[rosout-1]: started with pid [112152]
started core service [/rosout]
```

```
xubing@xubing-ThinkPad-E14-Gen-2:~$ rosrun turtlesim turtlesim_node
[ INFO] [1662977219.476780638]: Starting turtlesim with node name /turtlesim
[ INFO] [1662977219.478448137]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

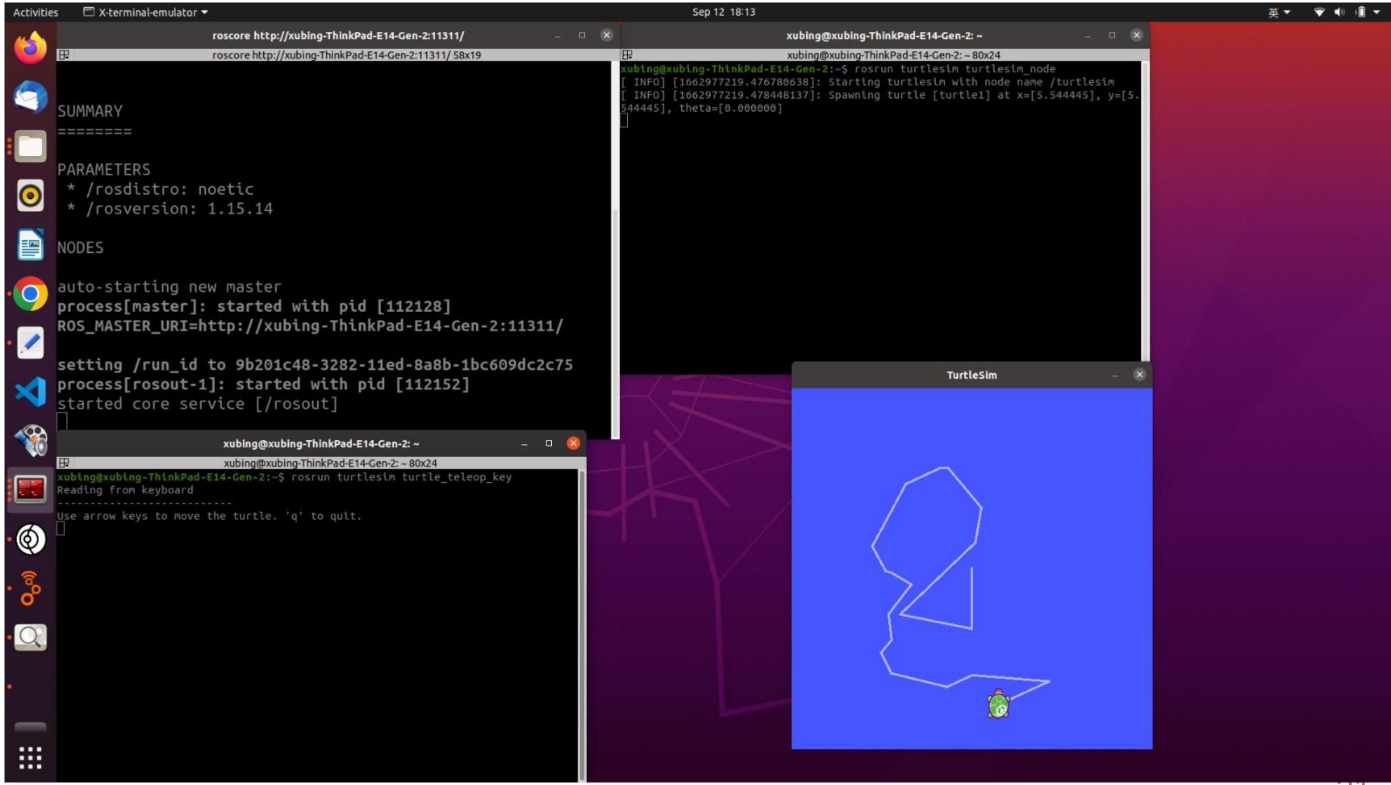
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```

2. Linux Operating System

ROS turtles demo

Step 4: control the turtle

↑ ↓ ← →


Activities
X-terminal-emulator
Sep 12 18:13
roscore http://xubing-ThinkPad-E14-Gen-2:11311/
roscore http://xubing-ThinkPad-E14-Gen-2:11311/ 58x19
xubing@xubing-ThinkPad-E14-Gen-2: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~ 80x24
TurtleSim

```
SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES

auto-starting new master
process[master]: started with pid [112128]
ROS_MASTER_URI=http://xubing-ThinkPad-E14-Gen-2:11311/

setting /run_id to 9b201c48-3282-11ed-8a8b-1bc609dc2c75
process[rosout-1]: started with pid [112152]
started core service [/rosout]
```

```
xubing@xubing-ThinkPad-E14-Gen-2:~$ rosrun turtlesim turtlesim_node
[ INFO] [1662977219.476780638]: Starting turtlesim with node name /turtlesim
[ INFO] [1662977219.478448137]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

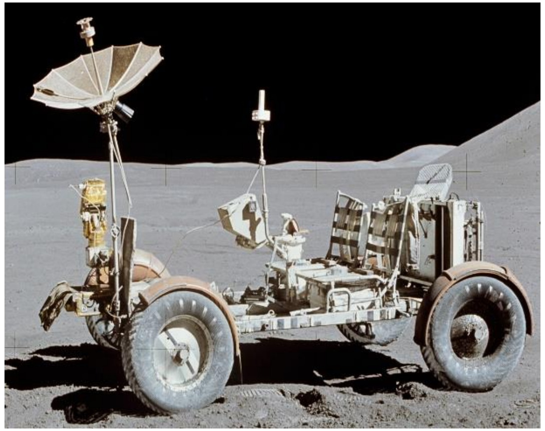
```
xubing@xubing-ThinkPad-E14-Gen-2:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```

AAE3004

Part III – ROS

3. ROS— Robot Operating System

Mechanical Arm

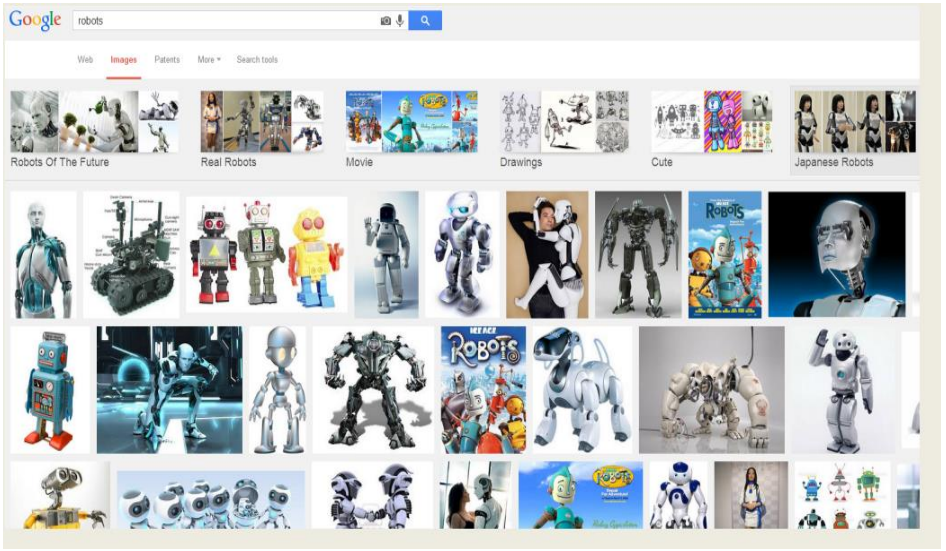

Lunar Rover

Humanoid Robot

The Problem:
Lack of standards for robotics

3. ROS— Robot Operating System

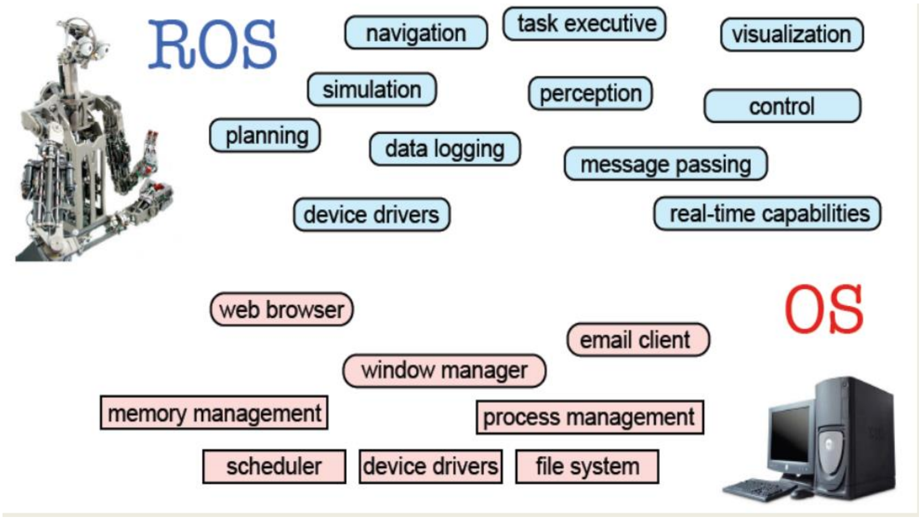
- ROS is an **open-source** robot operating system
- A set of software libraries and tools that help you build robot applications that work across a wide variety of robotic platforms
- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow Garage
- Since 2013 managed by OSRF (Open Source Robotics Foundation)



3. ROS

ROS Main Features

- The operating system side, which provides standard operating system services such as: – hardware abstraction – low-level device control – implementation of commonly used functionality – message-passing between processes – package management

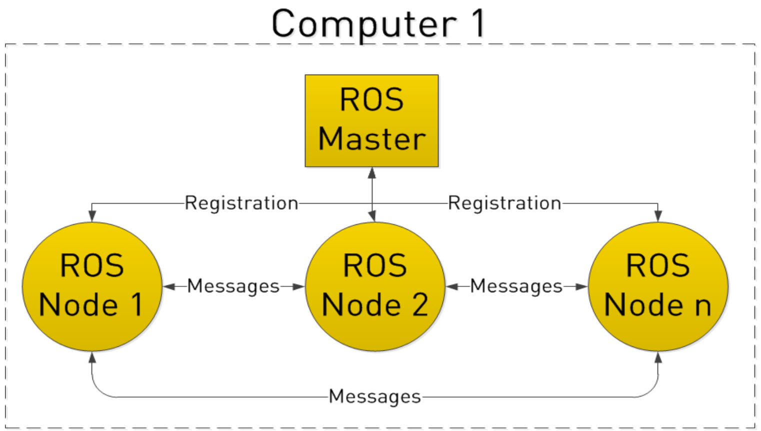


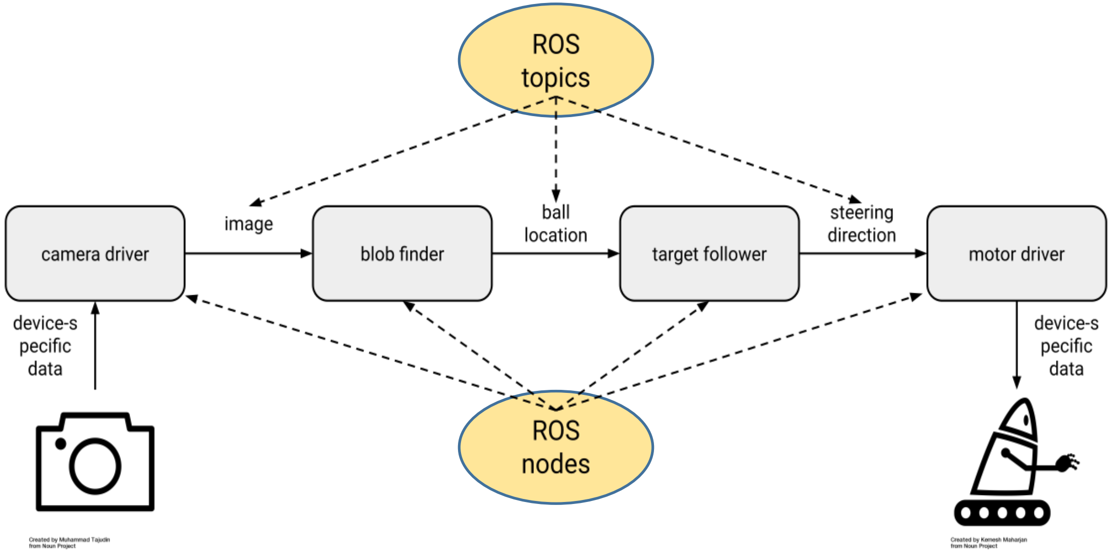
- A suite of user contributed packages that implement common robot functionality such as SLAM, planning, perception, vision, manipulation, etc.

3. ROS

ROS Computation graph model

The Computation Graph is the peer-to-peer **network of ROS processes** that are processing data together.





Decentralized !!

3. ROS

ROS Core Concepts

- ***Nodes***

(1)Single-purposed executable ***programs*** – e.g. sensor driver(s), actuator driver(s), mapper, planner, UI, etc.

(2) Individually compiled, executed, and managed

(3) Nodes can **publish** or **subscribe** to a Topic

(4) Nodes can also provide or use a service

(5) Nodes are written using a ROS client library

– roscpp – C++ client library

– rospy – python client library.

3. ROS

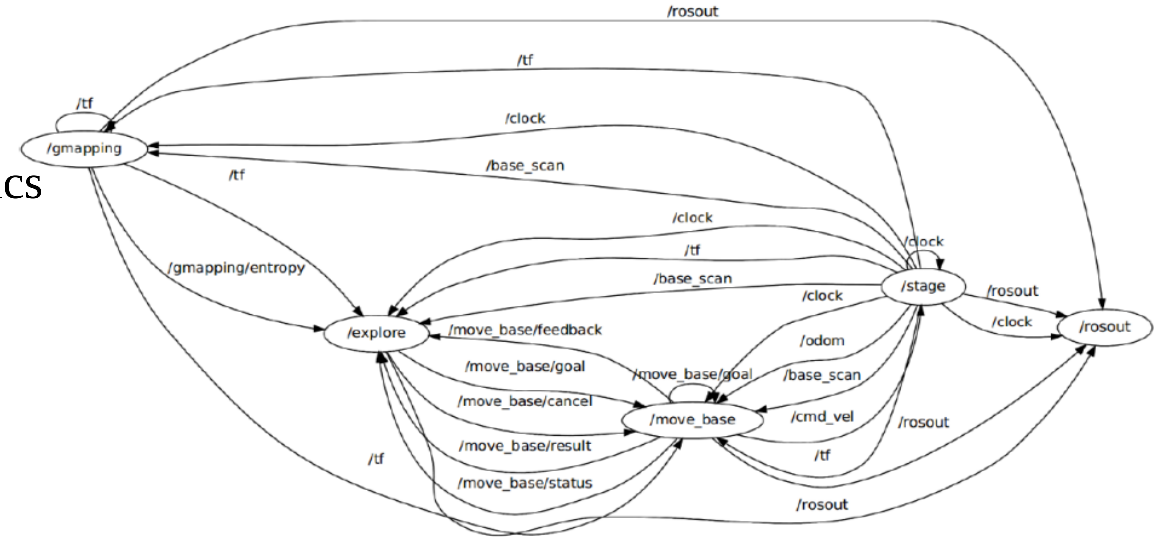
ROS Core Concepts

- ***Topics*:** Topics are named buses over which nodes exchange messages.

– e.g., data from a laser range-finder might be sent on a topic called scan, with a message type of LaserScan

(2) Nodes communicate with each other by publishing messages to topics

(3) Publish/Subscribe model: 1-to-N broadcasting



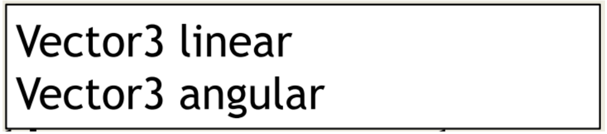
ROS Topics Graph

3. ROS

ROS Core Concepts

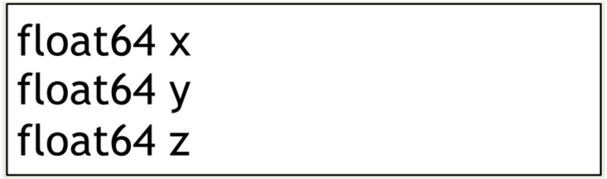
- ***Messages*:** Strictly-typed data structures for inter-node communication

For example, geometry_msgs/Twist is used to express velocity commands:



```
Vector3 linear
Vector3 angular
```

Vector3 is another message type composed of:



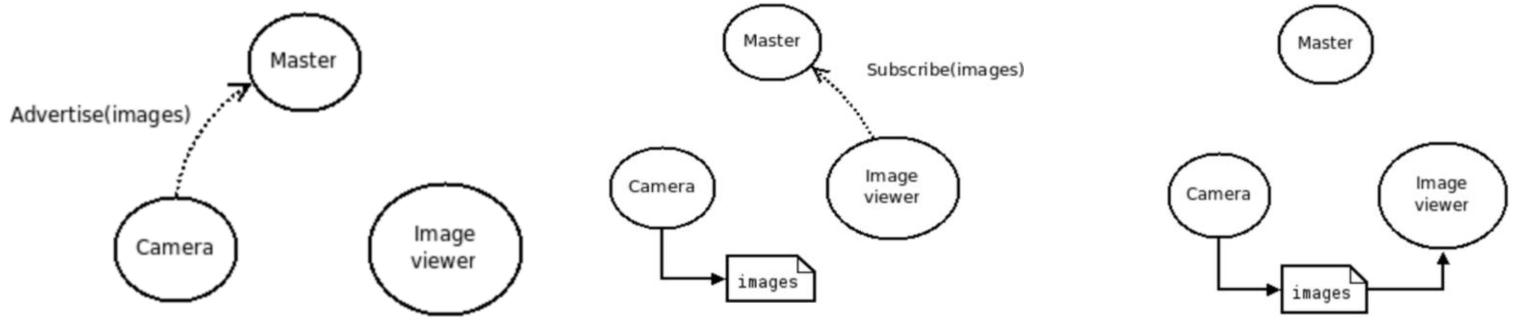
```
float64 x
float64 y
float64 z
```

3. ROS

ROS Core Concepts

- ***ROS Master***

Provides connection information to nodes so that they can transmit messages to each other
– Every node connects to a master at startup to register details of the message streams they publish, and the streams to which that they to subscribe.
– When a new node appears, the master provides it with the information that it needs to form a direct peer-to-peer connection with other nodes publishing and subscribing to the same message topics.



3. ROS

ROS Core Concepts

- ***ROS Services*****:** Synchronous inter-node transactions

(1)Service/Client model: 1-to-1 request-response

(2)Service roles:

– carry out remote computation

– trigger functionality / behavior

(3) Example:

– map_server/static_map – retrieves the current grid map used by the robot for navigation

3. ROS

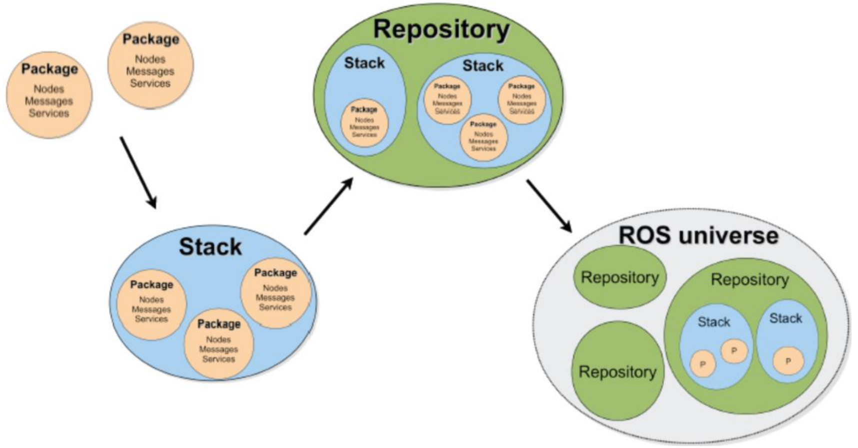
ROS Core Concepts

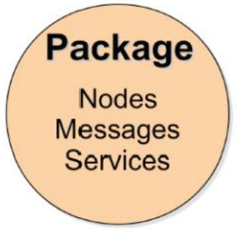
- ***ROS Package***

(1)Software in ROS is organized in packages.

(2) A package contains one or more nodes and provides a ROS interface.

(3) Most of ROS packages are hosted in GitHub.





3. ROS

Tutorial

- http://wiki.ros.org/

Installation:

- http://wiki.ros.org/ROS/ Installation

Tutorials:

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

- ROS Tutorial Videos

http://www.youtube.com/playlist? list=PLDC89965A56E6A8D6

- ROS Cheat Sheet

http://www.tedusar.eu/files/summerschool2013/ ROScheatsheet.pdf

GitHub

Online platform to collaborate on codes

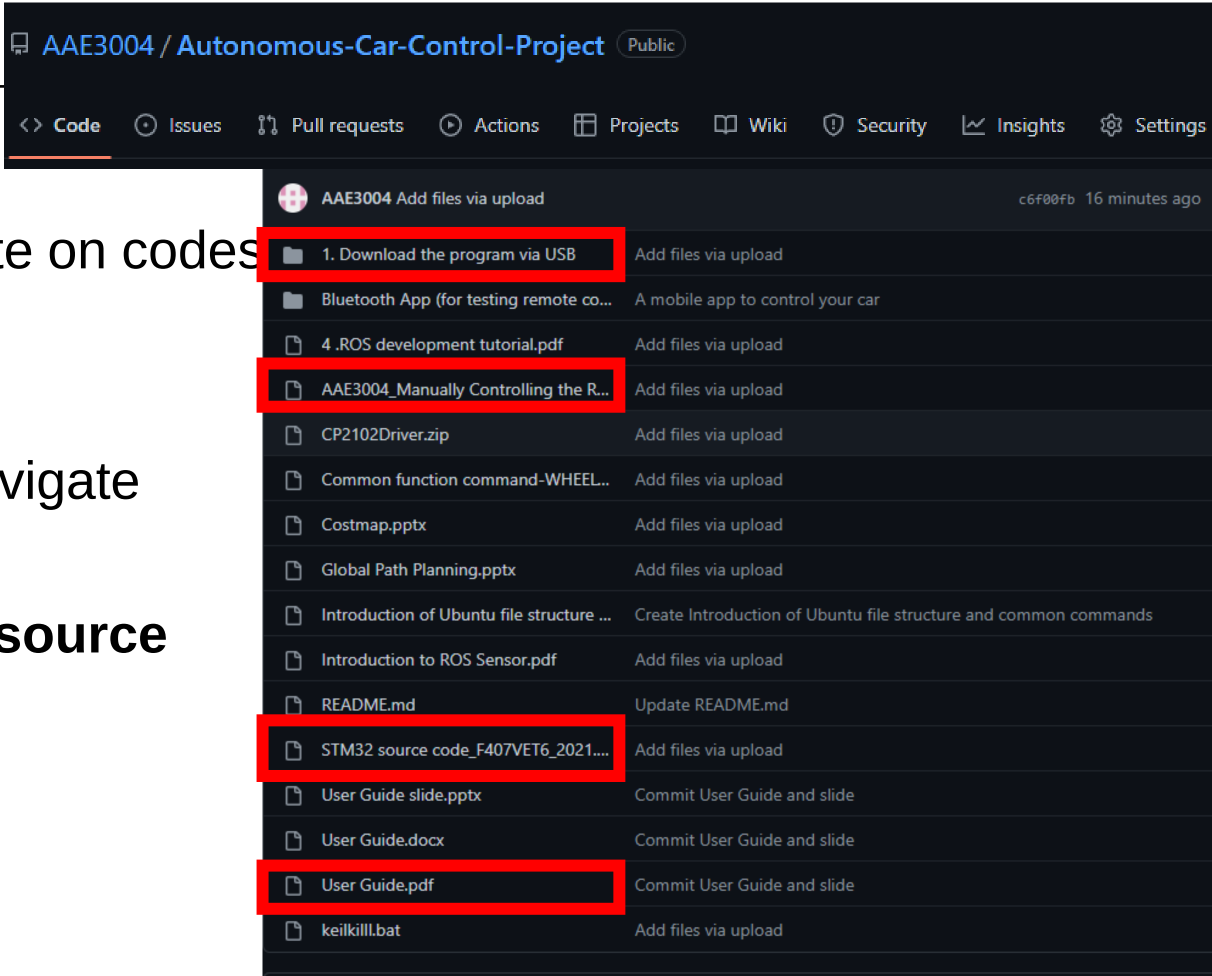
– Issues and Pull requests

Controlling the car

Using LiDAR to map and navigate

STM32 source code

Tutorials to install STM32 source code

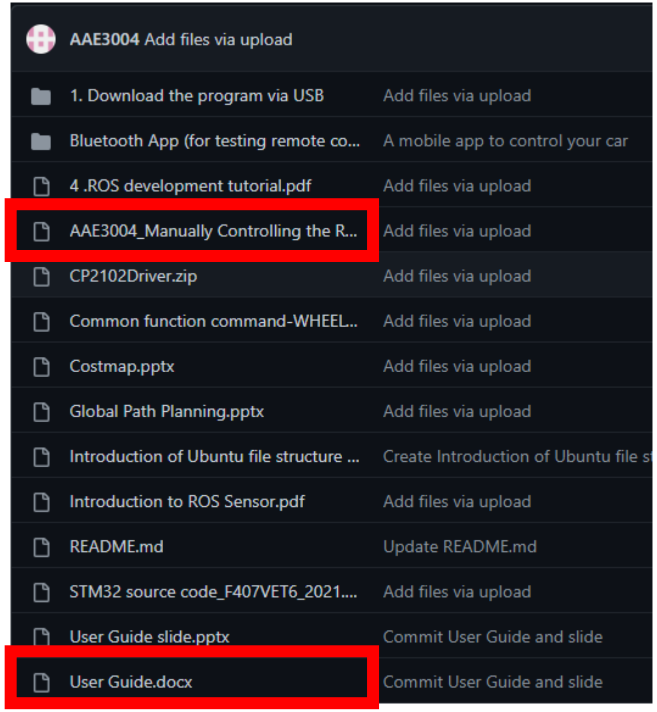


AAE3004 / Autonomous-Car-Control-Project Public

AAE3004 Add files via upload c6f00fb 16 minutes ago

1. Download the program via USB	Add files via upload
Bluetooth App (for testing remote co...	A mobile app to control your car
4 .ROS development tutorial.pdf	Add files via upload
AAE3004_Manually Controlling the R...	Add files via upload
CP2102Driver.zip	Add files via upload
Common function command-WHEEL...	Add files via upload
Costmap.pptx	Add files via upload
Global Path Planning.pptx	Add files via upload
Introduction of Ubuntu file structure ...	Create Introduction of Ubuntu file structure and common commands
Introduction to ROS Sensor.pdf	Add files via upload
README.md	Update README.md
STM32 source code_F407VET6_2021....	Add files via upload
User Guide slide.pptx	Commit User Guide and slide
User Guide.docx	Commit User Guide and slide
User Guide.pdf	Commit User Guide and slide
keilkilll.bat	Add files via upload

Controlling the car



AAE3004 Add files via upload

1. Download the program via USB	Add files via upload
Bluetooth App (for testing remote co...	A mobile app to control your car
4 .ROS development tutorial.pdf	Add files via upload
AAE3004_Manually Controlling the R...	Add files via upload
CP2102Driver.zip	Add files via upload
Common function command-WHEEL...	Add files via upload
Costmap.pptx	Add files via upload
Global Path Planning.pptx	Add files via upload
Introduction of Ubuntu file structure ...	Create Introduction of Ubuntu file st
Introduction to ROS Sensor.pdf	Add files via upload
README.md	Update README.md
STM32 source code_F407VET6_2021....	Add files via upload
User Guide slide.pptx	Commit User Guide and slide
User Guide.docx	Commit User Guide and slide

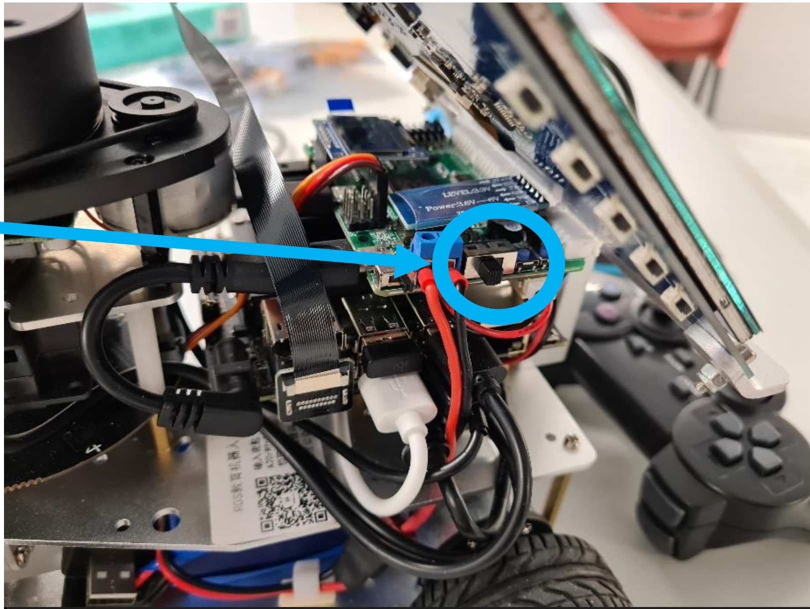
1. Using PS2 controller
2. Manually on the car - *AAE3004_manually Controlling the ROS Car*
3. Remotely using VM – *User Guide*

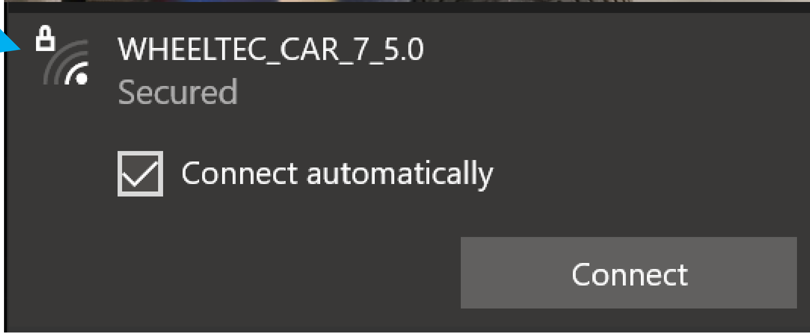
https://github.com/AAE3004/Autonomous-Car-Control-Project/tree/main

Keyboard control - Steps

1. Turn on the ROS car
2. Install Ubuntu on a Virtual Machine
3. Connect the Ubuntu to the ROS car Wi-Fi

Password: dongguan





WHEELTEC_CAR_7_5.0
Secured

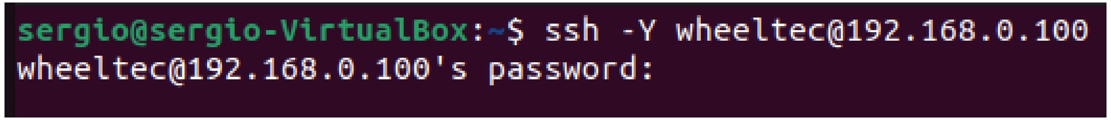
[✓] Connect automatically

Steps

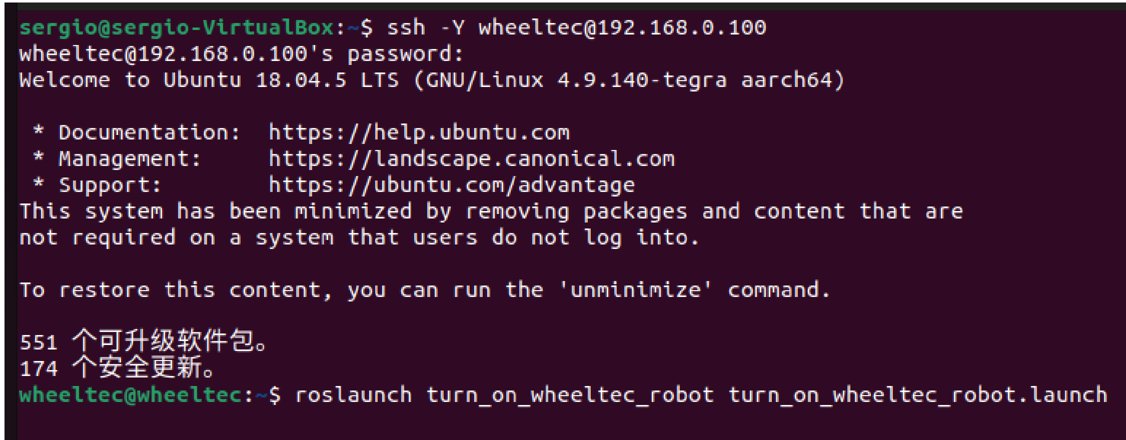
4. In Ubuntu VM, open a terminal (ctrl+alt+t) and type the following

$ sudo apt-get install openssh-server
$ ssh –Y wheeltec@192.168.0.100
Password: dongguan

$ roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch



```
sergio@sergio-VirtualBox:~$ ssh -Y wheeltec@192.168.0.100
wheeltec@192.168.0.100's password:
```

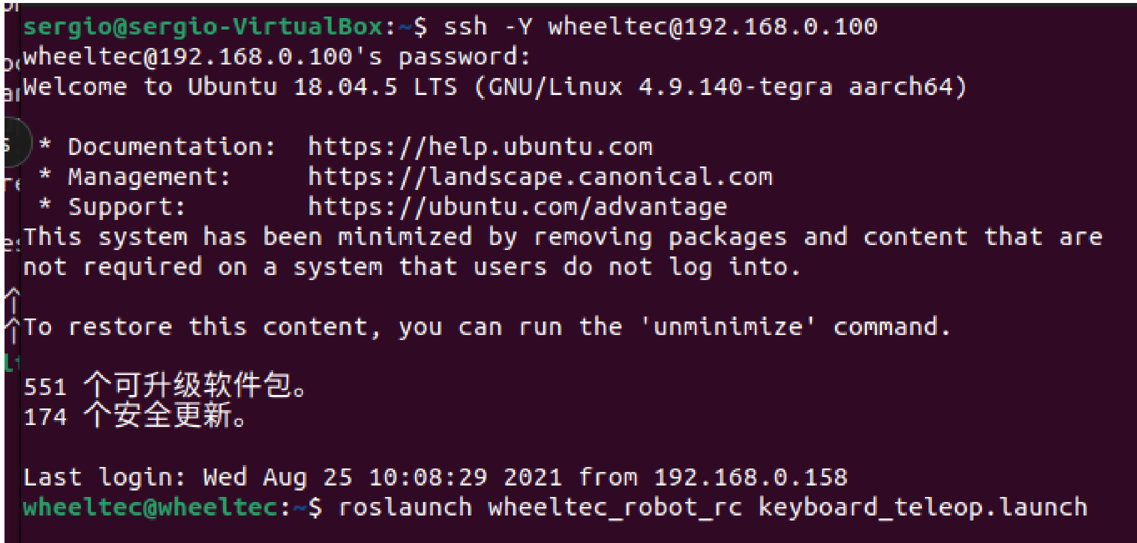


```
sergio@sergio-VirtualBox:~$ ssh -Y wheeltec@192.168.0.100
wheeltec@192.168.0.100's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.9.140-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

551 个可升级软件包。
174 个安全更新。
wheeltec@wheeltec:~$ roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch
```

Steps

5. Open another terminal (ctrl+alt+t) and type the following – DO NOT CLOSE THE PREVIOUS TERMINAL

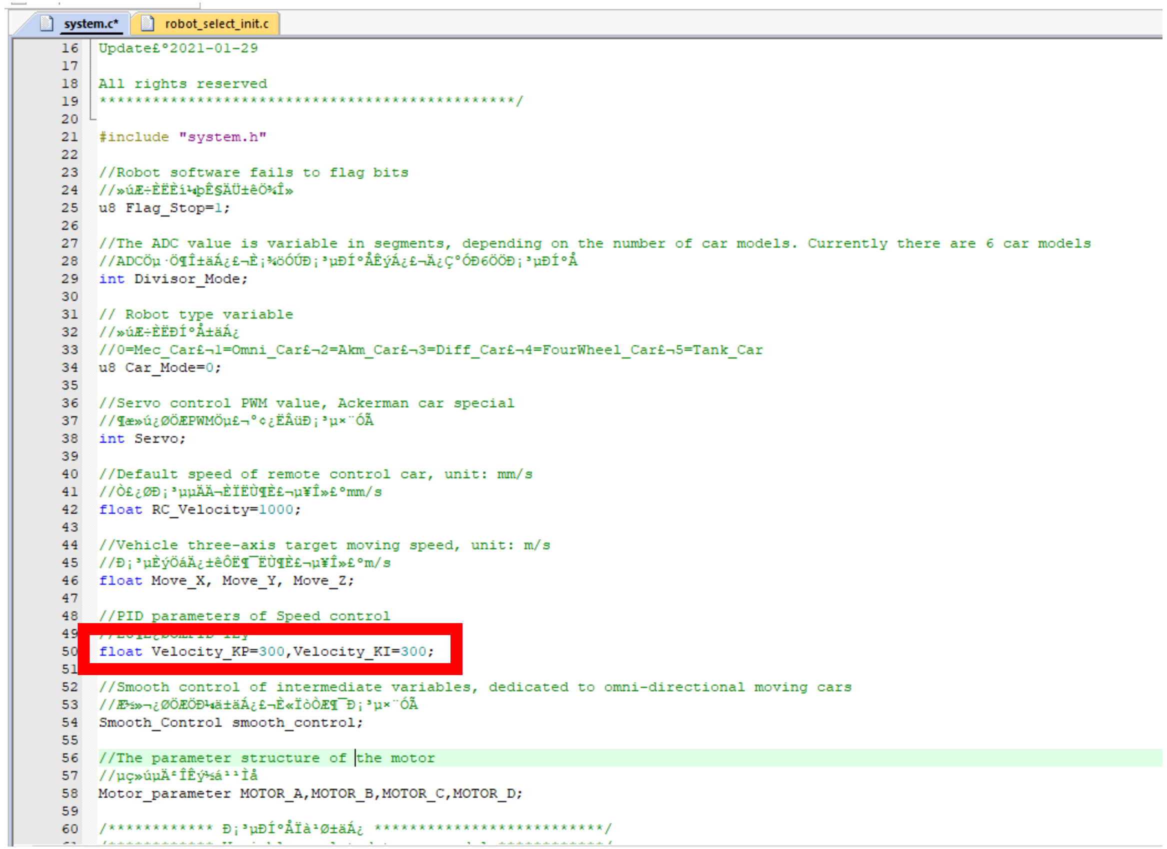
```
$ ssh –Y wheeltec@192.168.0.100
$ roslaunch wheeltec_robot_rc keyboard_teleop.launch
```



```
sergio@sergio-VirtualBox:~$ ssh -Y wheeltec@192.168.0.100
wheeltec@192.168.0.100's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.9.140-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

551 个可升级软件包。
174 个安全更新。

Last login: Wed Aug 25 10:08:29 2021 from 192.168.0.158
wheeltec@wheeltec:~$ roslaunch wheeltec_robot_rc keyboard_teleop.launch
```

Changing the car parameters - Demonstration

- PID controller
- STM32 source code
- Kp
- Ki



system.c* | robot_select_init.c

```c
Update£º2021-01-29

All rights reserved
***********************************************/

#include "system.h"

//Robot software fails to flag bits
//»úÆ÷ÈËÈí¼þÊ§ÄÜ±êÖ¾Î»
u8 Flag_Stop=1;

//The ADC value is variable in segments, depending on the number of car models. Currently there are 6 car models
//ADCÖµ·Ö¶Î±äÁ¿£¬È¡¾öÓÚÐ¡³µÐÍºÅÊýÁ¿£¬Ä¿Ç°ÓÐ6ÖÖÐ¡³µÐÍºÅ
int Divisor_Mode;

// Robot type variable
//»úÆ÷ÈËÐÍºÅ±äÁ¿
//0=Mec_Car£¬1=Omni_Car£¬2=Akm_Car£¬3=Diff_Car£¬4=FourWheel_Car£¬5=Tank_Car
u8 Car_Mode=0;

//Servo control PWM value, Ackerman car special
//¶æ»ú¿ØÖÆPWMÖµ£¬°¢¿ËÂüÐ¡³µ×¨ÓÃ
int Servo;

//Default speed of remote control car, unit: mm/s
//Ò£¿ØÐ¡³µµÄÄ¬ÈÏËÙ¶È£¬µ¥Î»£ºmm/s
float RC_Velocity=1000;

//Vehicle three-axis target moving speed, unit: m/s
//Ð¡³µÈýÖáÄ¿±êÔË¶¯ËÙ¶È£¬µ¥Î»£ºm/s
float Move_X, Move_Y, Move_Z;

//PID parameters of Speed control
//[illegible]
float Velocity_KP=300,Velocity_KI=300;

//Smooth control of intermediate variables, dedicated to omni-directional moving cars
//Æ½»¬¿ØÖÆÖÐ¼ä±äÁ¿£¬È«ÏòÒÆ¶¯Ð¡³µ×¨ÓÃ
Smooth_Control smooth_control;

//The parameter structure of the motor
//µç»úµÄ²ÎÊý½á¹¹Ìå
Motor_parameter MOTOR_A,MOTOR_B,MOTOR_C,MOTOR_D;

/************ Ð¡³µÐÍºÅÏà¹Ø±äÁ¿ **************************/
```