

AAE3004

Dynamical Systems and Control

Lab 1a: Autonomous ROS car control project

Instructors: Dr Weisong Wen, Mr Jian LIU (TA), and Mr Baoshan Song (TA)

Assistant Professor

Department of Aeronautical and Aviation Engineering

The Hong Kong Polytechnic University

Office: PQ408

Tel: 3400 8234

Email: welson.wen@polyu.edu.hk

- 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



Department of
Aeronautical and Aviation Engineering
航空及民航工程學系



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

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?

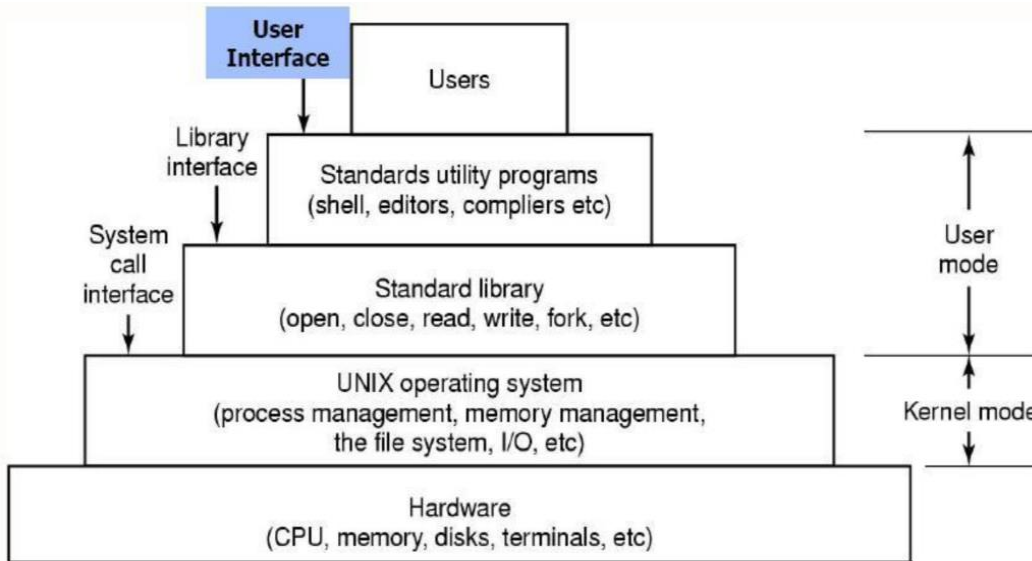
Operating System is a *program* that mediates between the *user* and the computer *hardware*.

- 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.

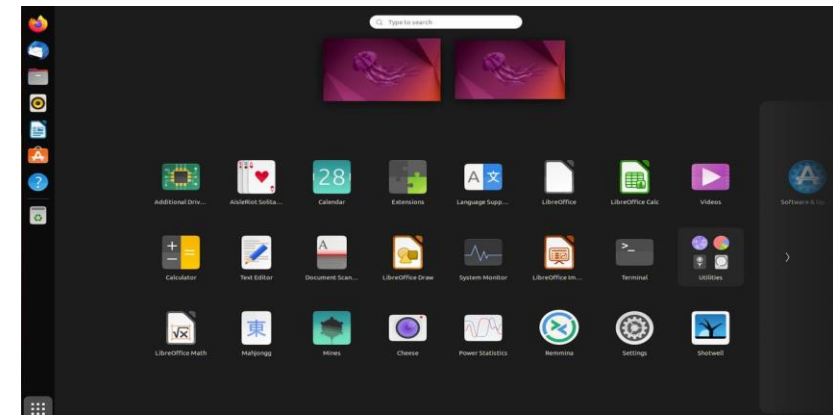
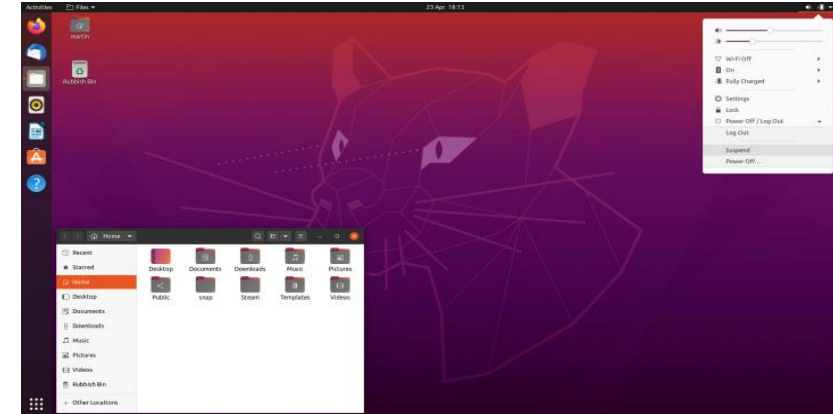
2. Linux Operating System



Linux Structure



Graphic User Interface (GUI)



2. Linux Operating System



Linux History

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



*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=PLS1QuIW01RIb9WVQGJ_vh-RQusbZgO_As

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

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

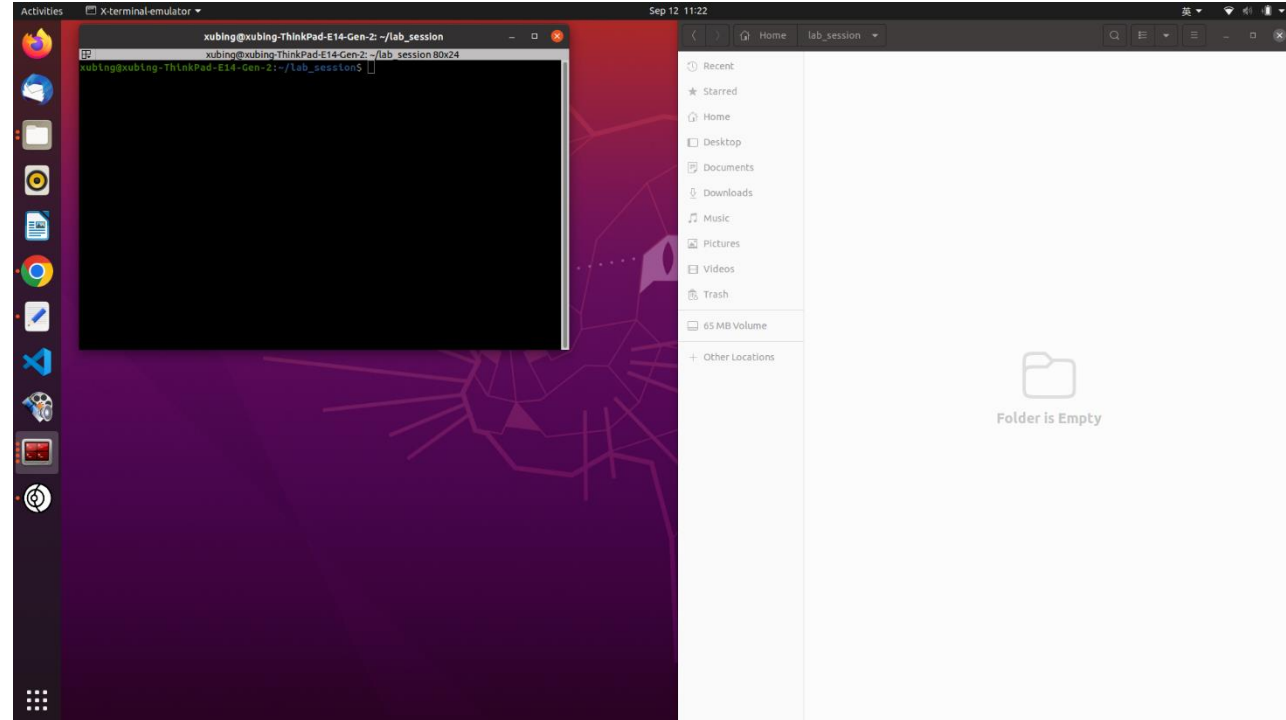
2. Linux Operating System



Helloworld demo with Linux

Step 1: Open the terminal

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



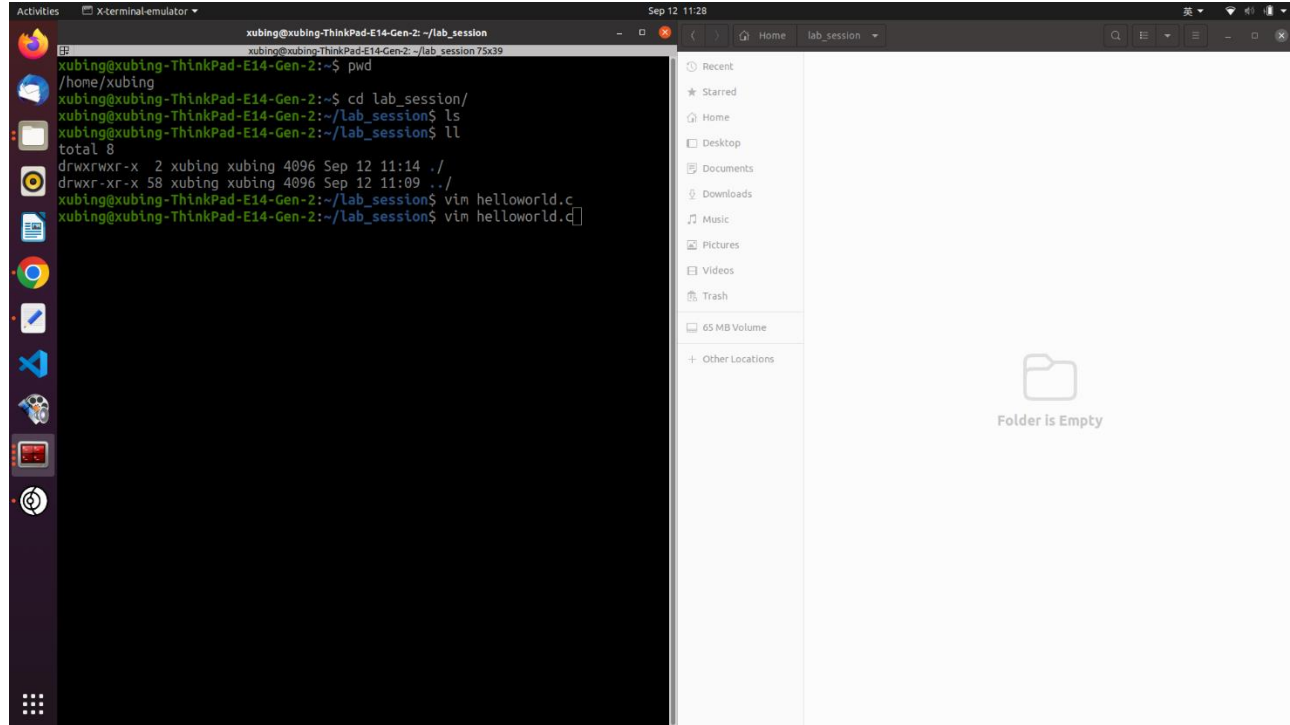
2. Linux Operating System



Helloworld demo with Linux

Step 2: Create the C file

- `$ vim helloworld.c`



The screenshot displays a Linux desktop environment. On the left, a vertical dock contains icons for various applications including a web browser, file manager, and terminal. The main workspace features two windows. The first is a terminal window titled 'xubing@xubing-ThinkPad-E14-Gen-2: ~/lab_session' with the following command history:

```
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
```

The second window is a file manager titled 'lab_session' showing an empty folder with the message 'Folder is Empty'. The sidebar of the file manager lists standard Linux directories like Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, and Trash.

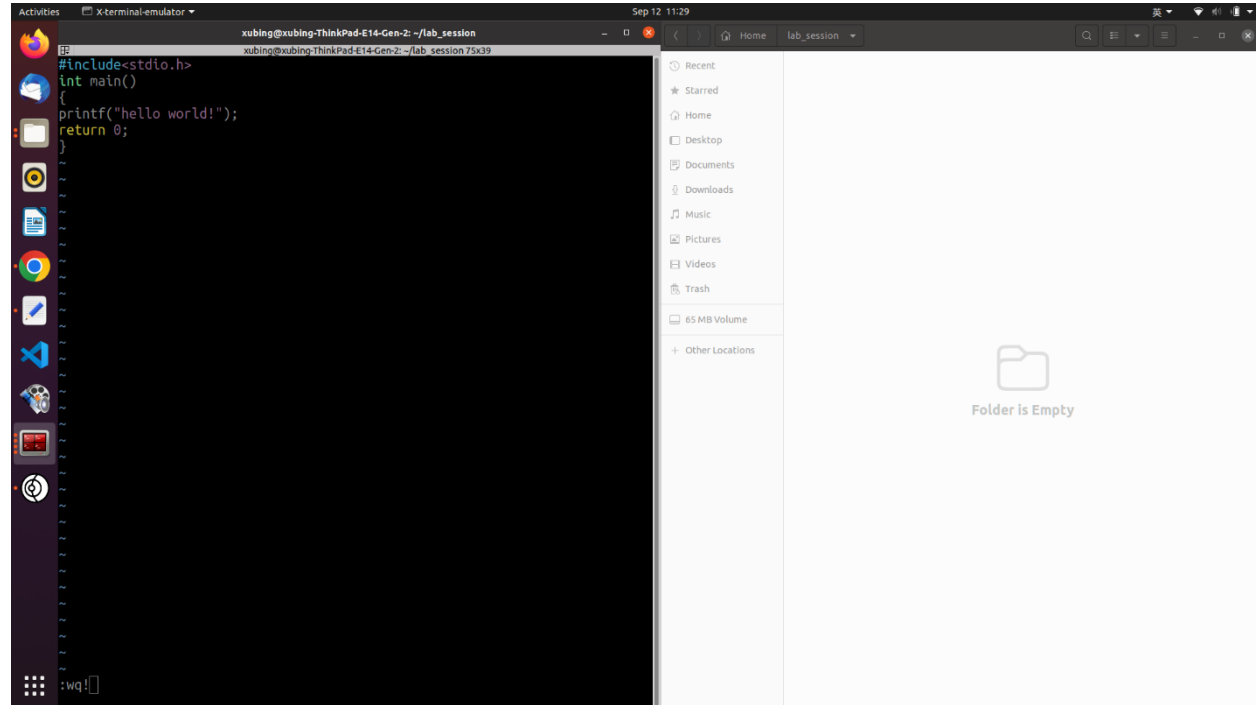
2. Linux Operating System



Helloworld demo with Linux

Step 3: Input the code and
save it

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



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;
}
```

The screenshot shows a terminal window titled 'xubing@xubing-ThinkPad-E14-Gen-2: ~/lab_session'. The user has navigated to the 'lab_session' directory and listed its contents, showing a file named 'helloworld.c'. The user then uses the 'vim' editor to open 'helloworld.c'. The terminal output shows the file's permissions and creation date. The user then runs the program, and the output 'hello world.' is displayed in the terminal window.

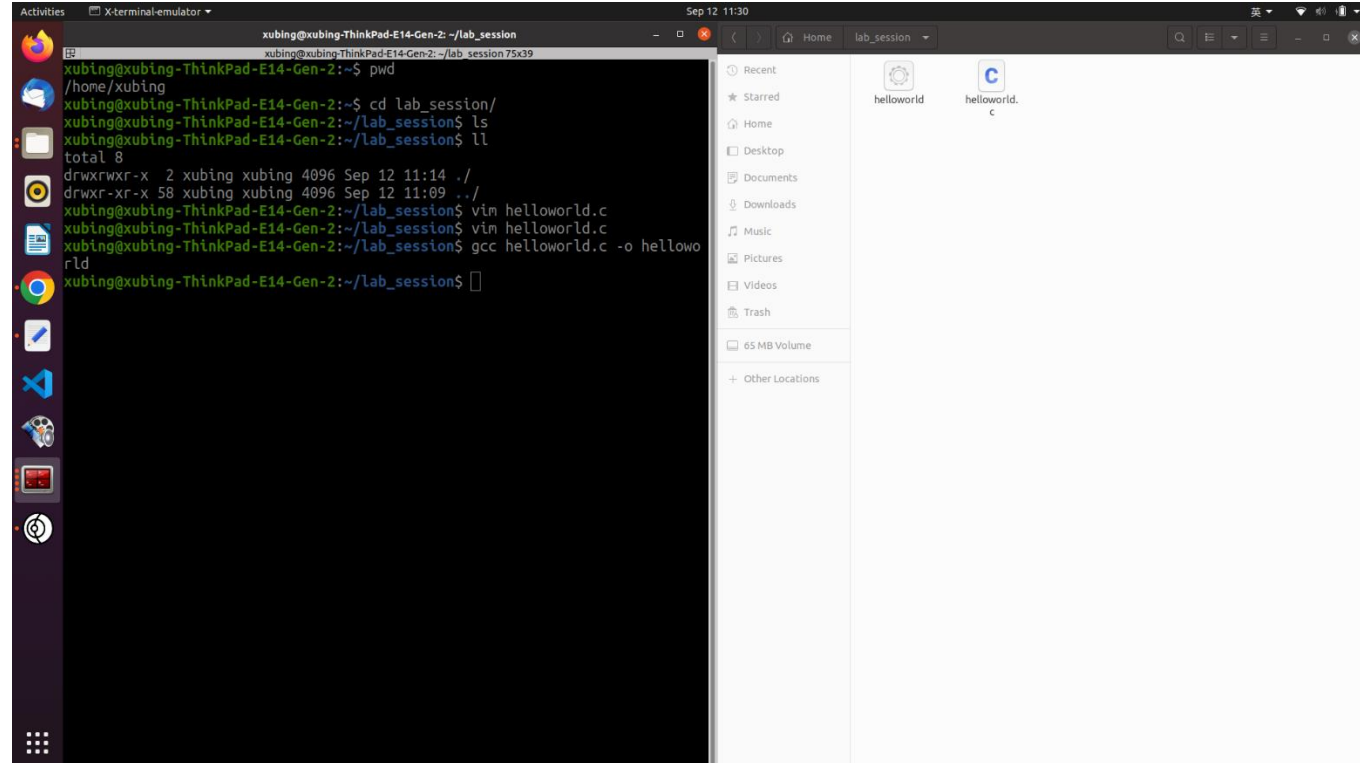
2. Linux Operating System



Helloworld demo with Linux

Step 4: Compile

- `$ gcc helloworld.c -o helloworld`



```
xubing@xubing-ThinkPad-E14-Gen-2: ~/lab_session
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 helloworld
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: ~/lab_session
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ pwd
/home/xubing
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ 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 helloworld
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$ ./helloworld
hello world!
xubing@xubing-ThinkPad-E14-Gen-2:~/lab_session$
```

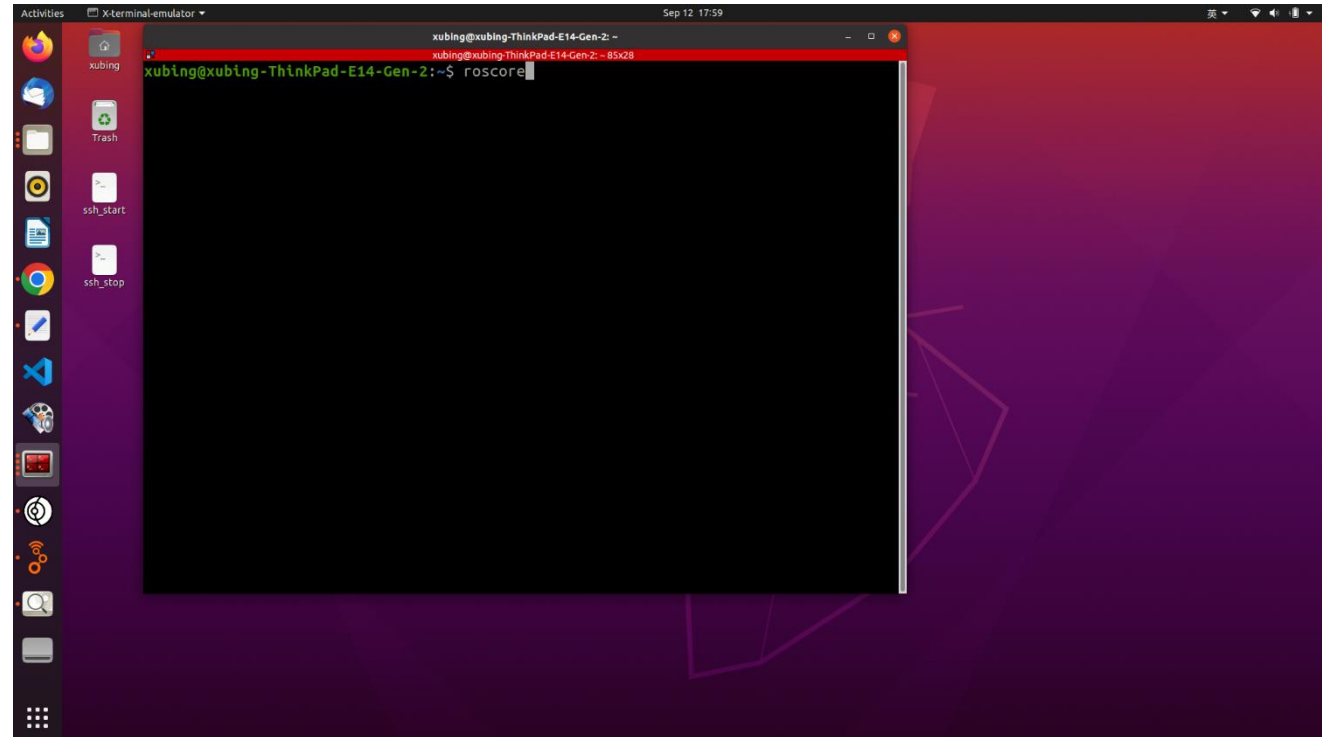
2. Linux Operating System



ROS turtles demo

Step 1: create the master node

- `$ roscore`



2. Linux Operating System



ROS turtles demo

Step 2: create a node to show
turtle

- `$ rosrn turtlesim
turtlesim_node`

```
Activities  turtlesim_node
roscore http://xubing-ThinkPad-E14-Gen-2:11311/
roscore http://xubing-ThinkPad-E14-Gen-2:11311/ 58x19

SUMMARY
=====
PARAMETERS
* /roscore: noetic
* /roscore: 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: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~$ rosrn turtlesim turtlesim_node
[INFO] [1662977219.476786337]: Starting turtlesim with node name /turtlesim
[INFO] [1662977219.478446137]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

TurtleSim
```

2. Linux Operating System



ROS turtles demo

Step 3: create the keyboard node

- `$ rosrn turtlesim turtlesim_node`

The screenshot displays a Linux desktop environment with three terminal windows and a visualization window. The leftmost terminal window shows the output of the `roscore` command, displaying the ROS master's status and the start of the `turtlesim` node. The middle terminal window shows the output of the `roslaunch` command, which starts the `turtlesim` node. The rightmost terminal window shows the output of the `rostopic echo` command, displaying the turtle's position and orientation. The visualization window, titled "Turtlesim", shows a blue square representing the turtle's environment, with a small green turtle icon in the center.

```
roscore http://xubing-ThinkPad-E14-Gen-2:11311/
roscore http://xubing-ThinkPad-E14-Gen-2:11311/ 58x19

SUMMARY
=====
PARAMETERS
* /roscore: noetic
* /roscore: 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[roscout-1]: started with pid [112152]
started core service [/roscout]

xubing@xubing-ThinkPad-E14-Gen-2: ~
xubing@xubing-ThinkPad-E14-Gen-2: ~$ roslaunch turtlesim turtlesim_node
[INFO] [1662977219.476780638]: Starting turtlesim with node name /turtlesim
[INFO] [1662977219.478448137]: Spawning turtle [turtle1] at x=[5.544445], y=[5.444445], theta=[0.000000]

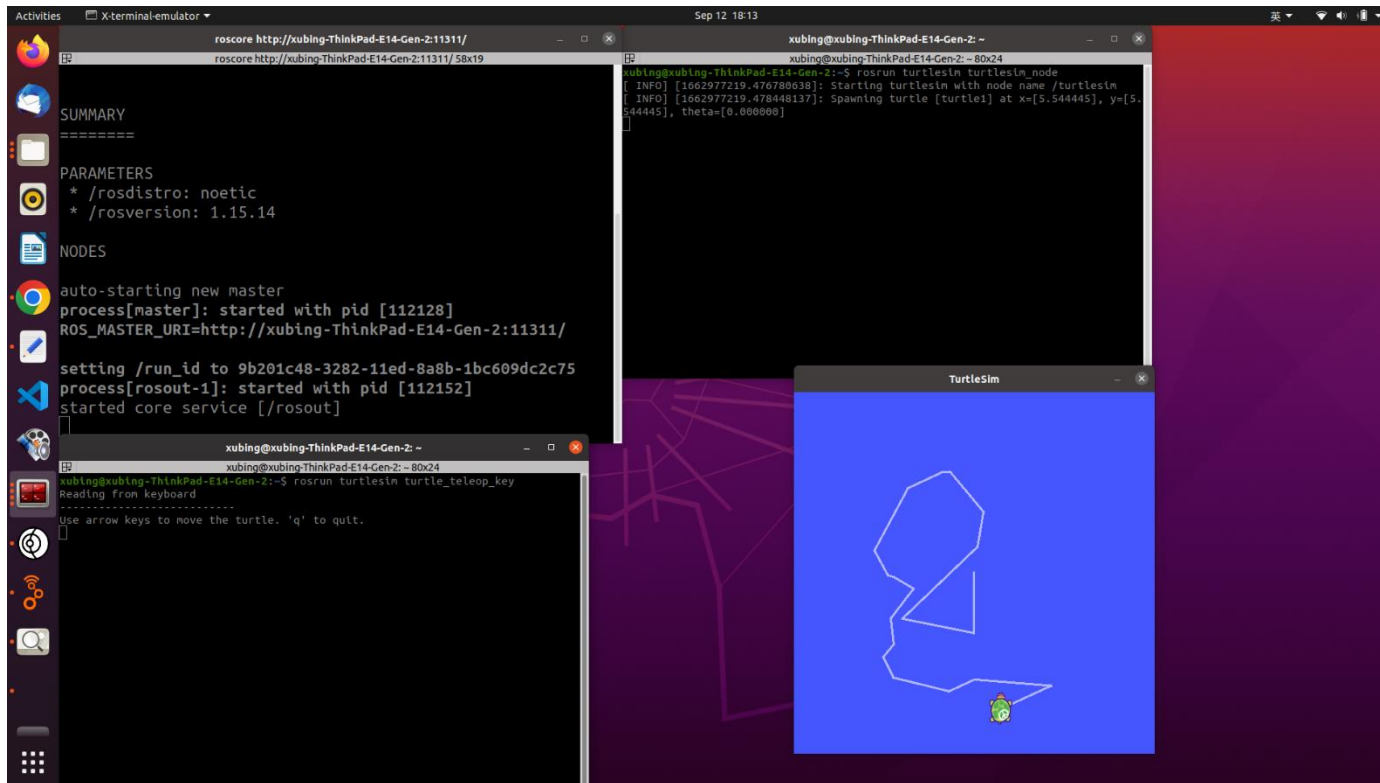
xubing@xubing-ThinkPad-E14-Gen-2: ~$ rostopic echo /turtle1/pose
xubing@xubing-ThinkPad-E14-Gen-2: ~$
```

2. Linux Operating System



ROS turtles demo

Step 4: control the turtle





AAE3004



Department of
Aeronautical and Aviation Engineering
航空及民航工程學系



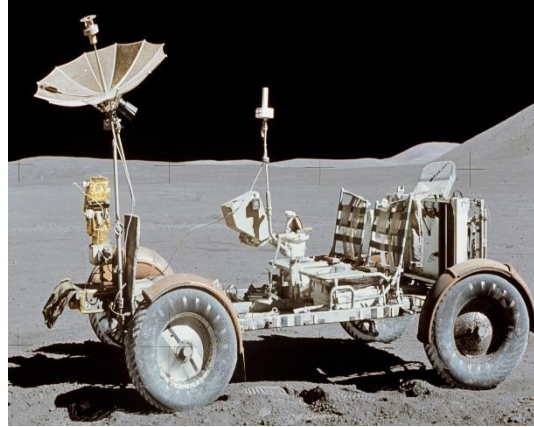
THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Part III – ROS

3. ROS— Robot Operating System



Mechanical Arm



Lunar Rover



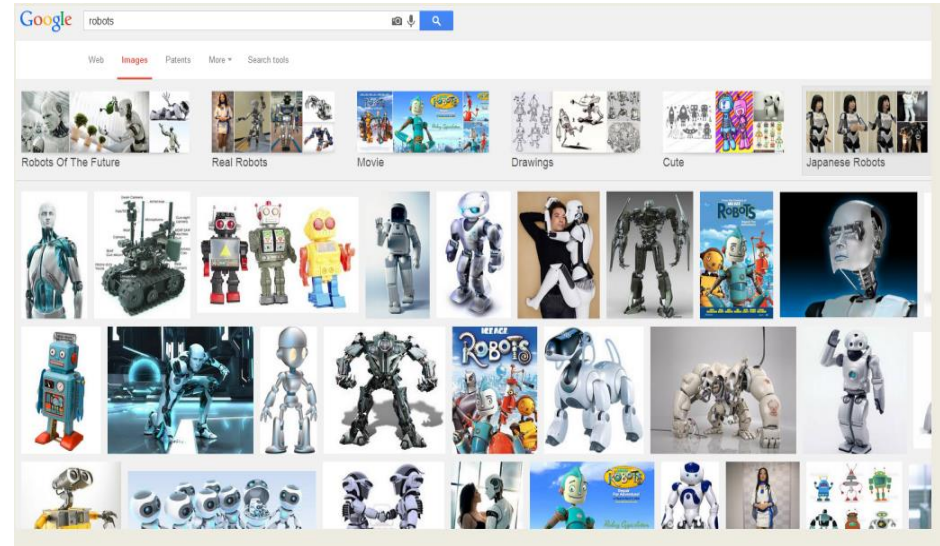
Humanoid Robot



The Problem:
Lack of standards for
robotics

3. ROS— Robot Operating System

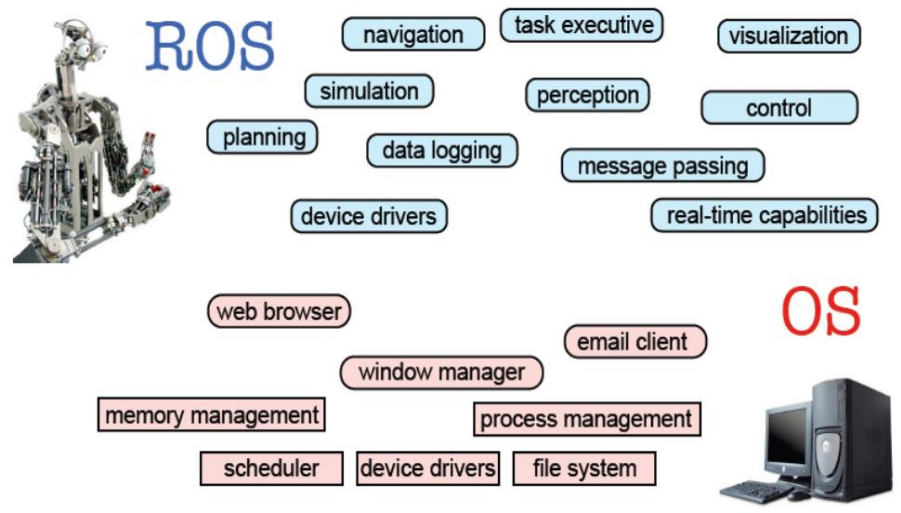
- ROS is an **open-source** robot operating system
- A set of software libraries and tools that help you build robot applications that work across a wide variety of robotic platforms
- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow 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.

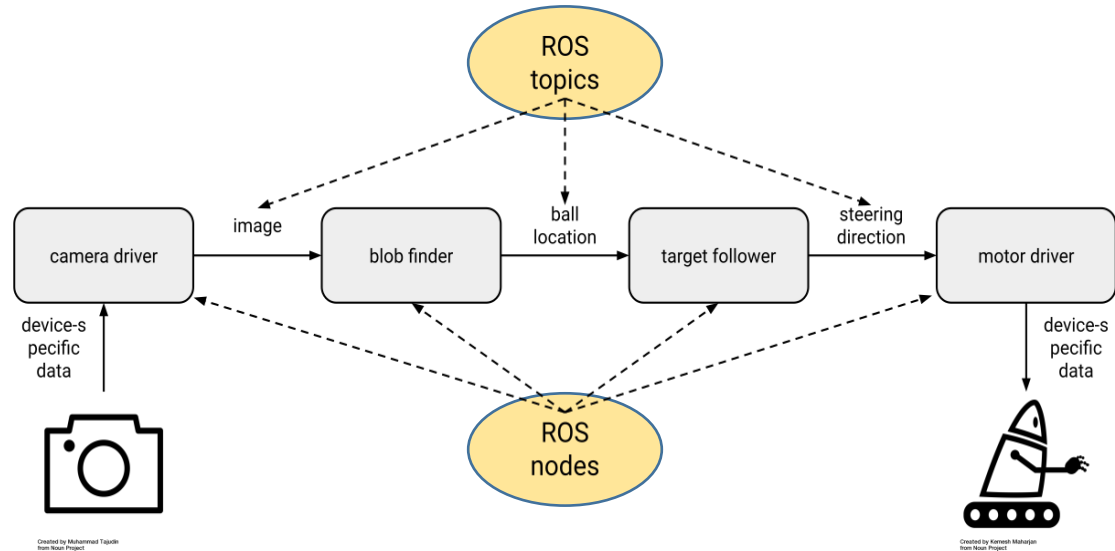
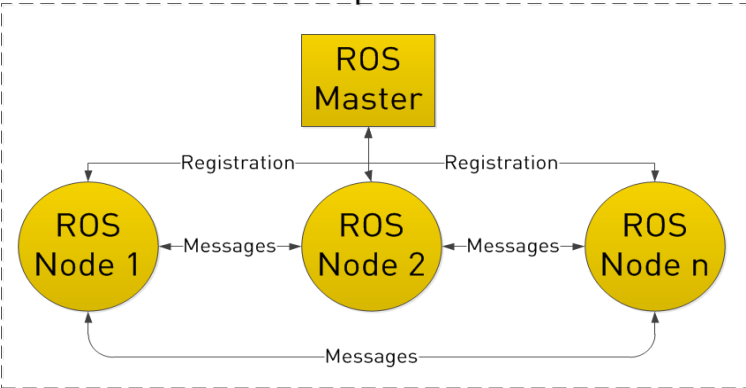
3. ROS



ROS Computation graph model

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

Computer 1



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

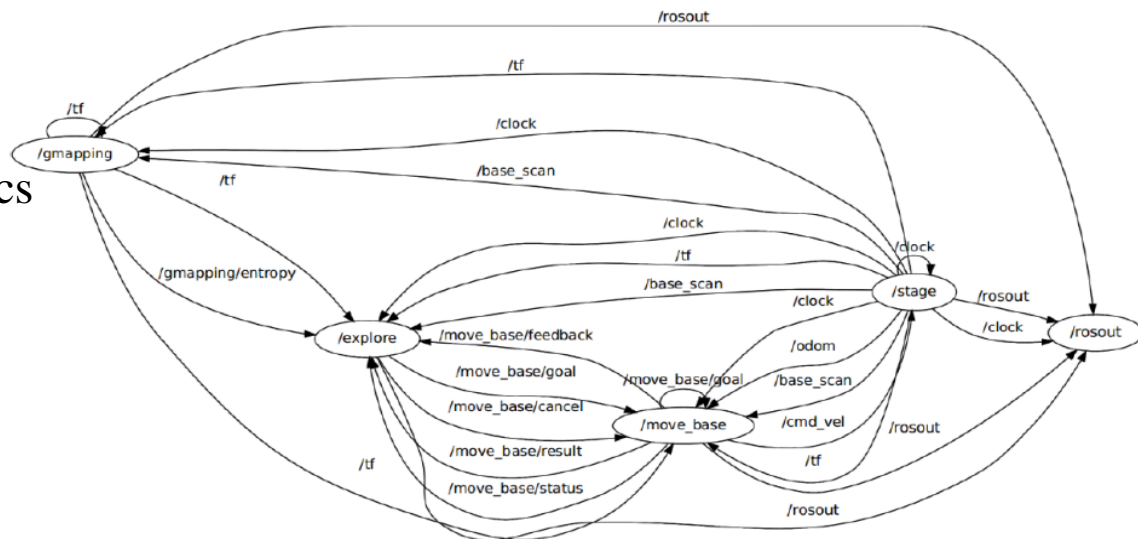
ROS Core Concepts

- 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

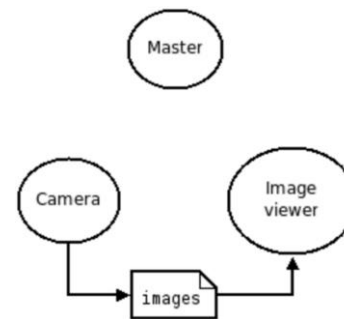
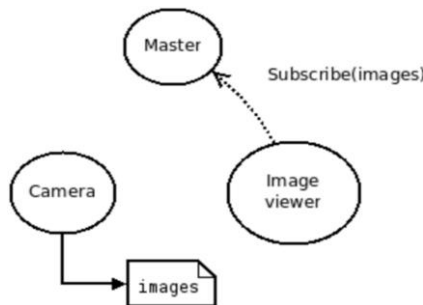
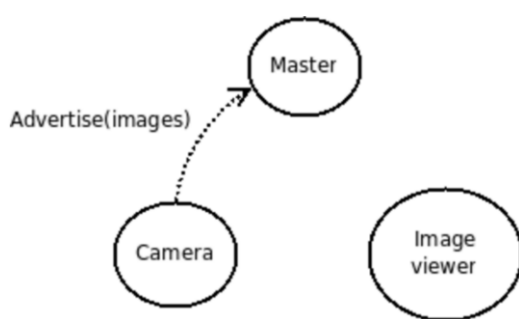
ROS Core Concepts

- **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 they 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:

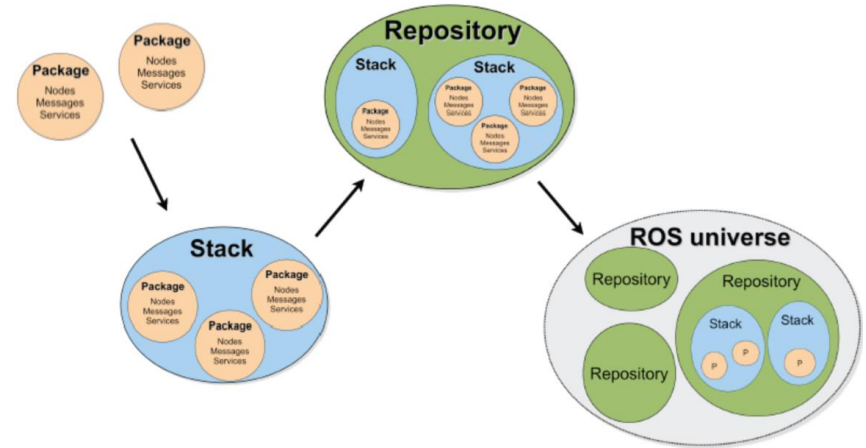
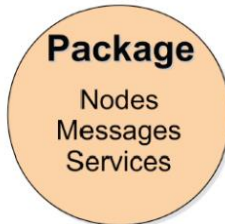
- map_server/static_map – retrieves the current grid map used by the robot for navigation

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](http://wiki.ros.org/ROS/Installation)

Tutorials:

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

- [ROS Tutorial Videos](#)

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

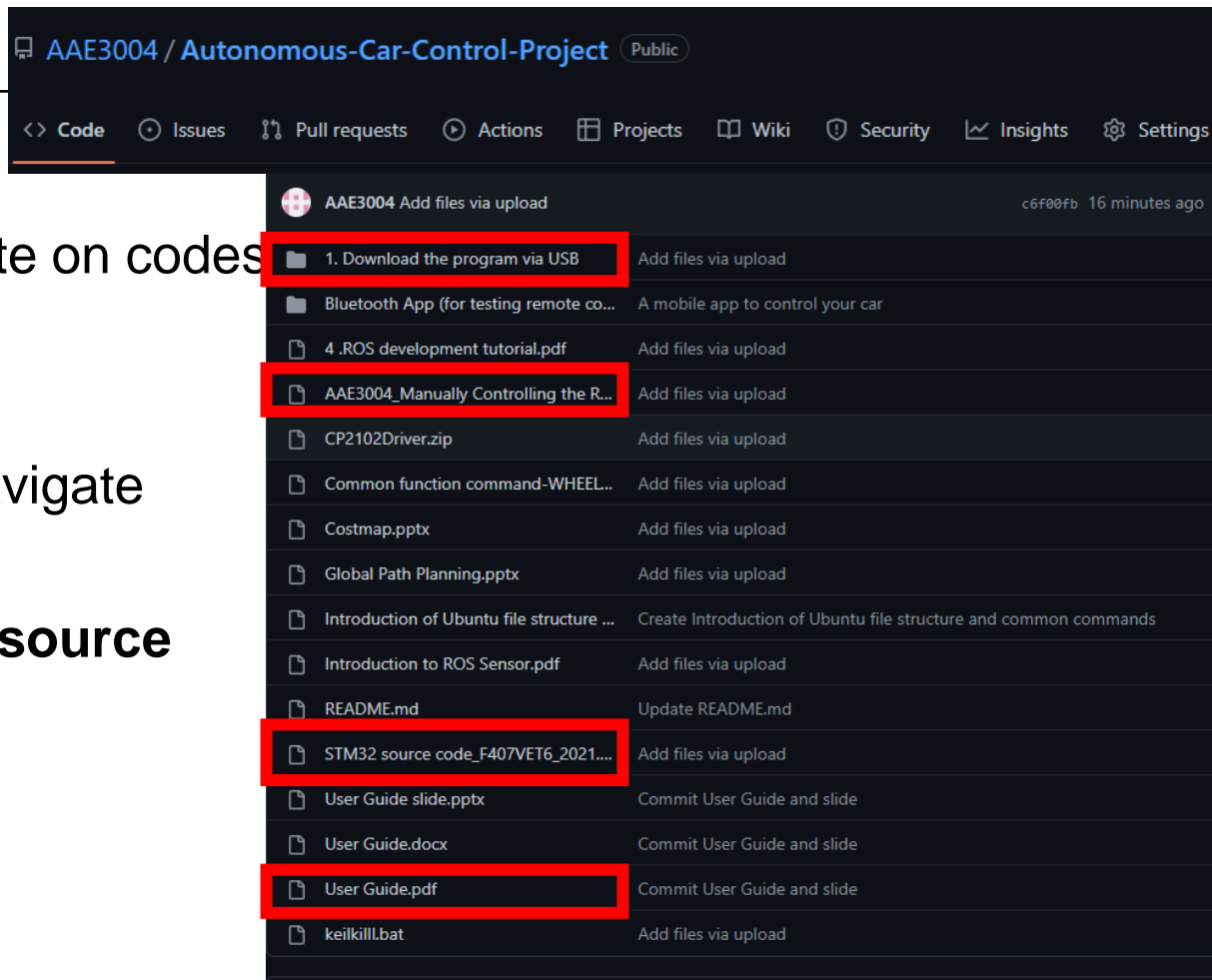
– Issues and Pull requests

Controlling the car

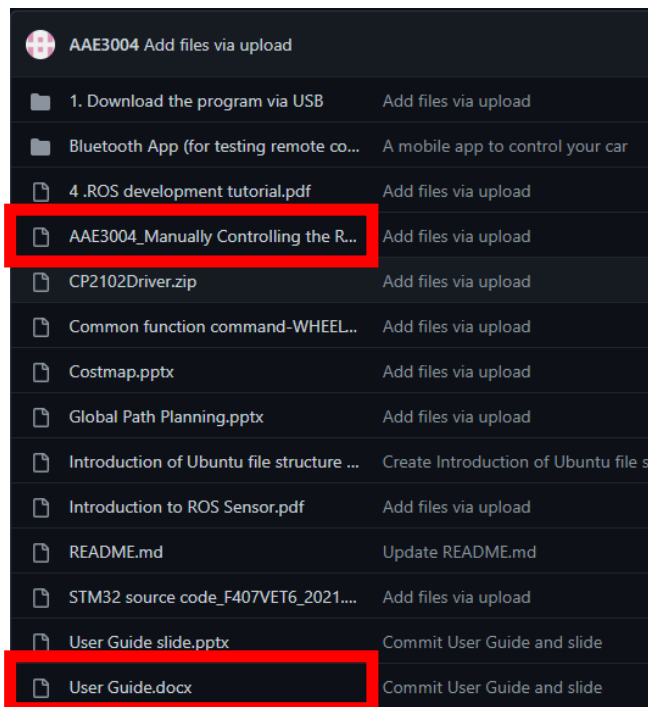
Using LiDAR to map and navigate

STM32 source code

Tutorials to install STM32 source code



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 s
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

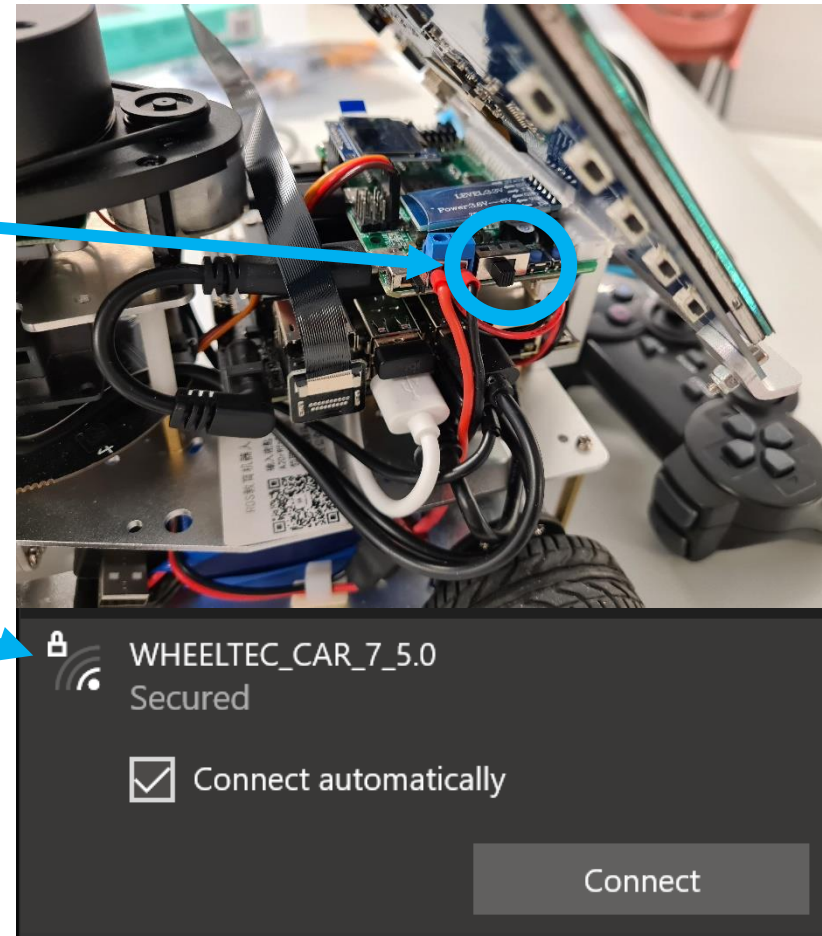
1. Using PS2 controller
2. Manually on the car -
*AAE3004_manually
Controlling the ROS Car*
3. Remotely using VM – *User
Guide*

<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



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
```

```
$ roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch
```

```
passoni@passoni: ~  
passoni@passoni: ~ 80x24  
passoni@passoni:~$ ssh -Y wheeltec@192.168.0.100  
wheeltec@192.168.0.100's password: █
```

```
/home/wheeltec/wheeltec_robot/src/turn_on_wheeltec_robot/launch/turn_on_wheeltec_r...  
/home/wheeltec/wheeltec_robot/src/turn_on_wheeltec_robot/launch/turn_on_wheeltec_robot.launch  
process[robot_state_publisher-9]: started with pid [4815]  
[ INFO] [1607510796.120191805]: Data ready  
[ INFO] [1607510796.132677879]: wheeltec_robot serial port opened  
[ INFO] [1607510796.145027416]: output frame: odom_combined  
[ INFO] [1607510796.151218157]: base frame: base_footprint  
[ INFO] [1607510796.491093842]: Initializing Odom sensor  
[ INFO] [1607510796.491461861]: Initializing Imu sensor  
[ INFO] [1607510796.540958527]: Odom sensor activated  
[ INFO] [1607510796.541145842]: Imu sensor activated  
[ INFO] [1607510796.551794990]: Kalman filter initialized with odom measurement
```

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
```

```
/home/wheeltec/wheeltec_robot/src
ROS_MASTER_URI=http://192.168.0.100:11311

process[turtlebot_teleop_keyboard-1]: started with pid [4971]

Control Your Turtlebot!
-----
Moving around:
  u      i      o
  j      k      l
  m      ,      .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly
b : switch to OmniMode/CommonMode
CTRL-C to quit

currently: 1 speed 0.2      turn 1
```



Changing the car parameters - Demonstration

PID controller
STM32
source code
Kp
Ki

```
system.c* robot_select_init.c
16 Updated: 2021-01-29
17
18 All rights reserved
19 *****/
20
21 #include "system.h"
22
23 //Robot software fails to flag bits
24 //u8 Flag_Stop=1;
25
26 //The ADC value is variable in segments, depending on the number of car models. Currently there are 6 car models
27 //ADC0: 0~1023; ADC1: 0~1023; ADC2: 0~1023; ADC3: 0~1023; ADC4: 0~1023; ADC5: 0~1023
28 int Divisor_Mode;
29
30 // Robot type variable
31 //u8 Robot_Type=0;
32 //0=Meo_Car1=1=Omni_Car2=2=Akm_Car3=3=Diff_Car4=4=FourWheel_Car5=5=Tank_Car
33 u8 Car_Mode=0;
34
35 //Servo control PWM value, Ackerman car special
36 //u16 Servo_Pulse=0;
37 int Servo;
38
39 //Default speed of remote control car, unit: mm/s
40 //u16 RC_Speed=1000;
41 float RC_Velocity=1000;
42
43 //Vehicle three-axis target moving speed, unit: m/s
44 //u16 Move_X, Move_Y, Move_Z;
45 float Move_X, Move_Y, Move_Z;
46
47 //PID parameters of Speed control
48 //u16 Velocity_KP=300, Velocity_KI=300;
49 float Velocity_KP=300, Velocity_KI=300;
50
51 //Smooth control of intermediate variables, dedicated to omni-directional moving cars
52 //u16 Smooth_Control;
53 float Smooth_Control;
54
55 //The parameter structure of the motor
56 //u16 Motor_Parameter;
57 Motor_Parameter MOTOR_A, MOTOR_B, MOTOR_C, MOTOR_D;
58
59 //***** D: u16 D; *****
60 //***** D: u16 D; *****
```