인공지능 모바일 로봇 개발과정

백 진철

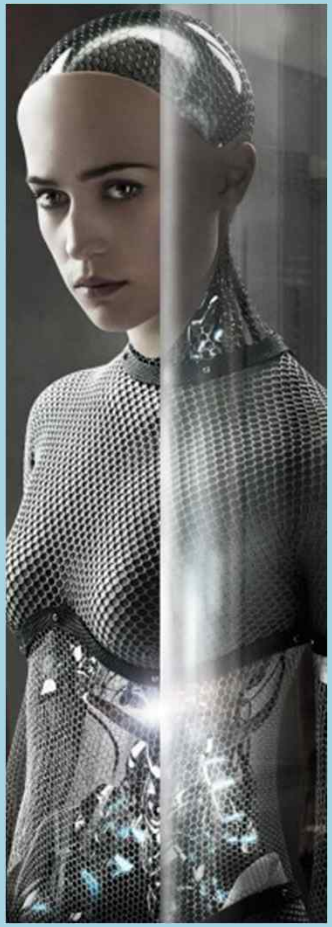
젊은세상

과정진행

- ROS는 무엇인가?
- Jetson TX를 사용해야 하는 이유
- ROS 기본 명령 사용
- ROS 프로그램 개발1 (topic 이용)
- ROS 프로그램 개발2 (Service 이용)
- 인공지능 기본
- MNIST를 이용한 숫자 인식
- 카메라를 이용한 숫자 인식

젊은세상

어떤 로봇을 만들어야 하나?



- 엑스마키나 채피 아이로봇

젊은세상

현실적인 로봇

젊은세상

ROS 지원 로봇 및 센서

- 100개이상의 로봇지원
- 90개이상의 센서지원

젊은세상

ROS(Robot Operating System)

- 운영체제가 아니라 메타운영체제라 부름
- 로봇 소프트웨어를 작성하기위한 유연한 프레임 워크
- 복잡하고 강력한 로봇 동작을 생성하는 작업을 단순화하는 것을 목표로하는 도구, 라이브러리 및 규칙 모음

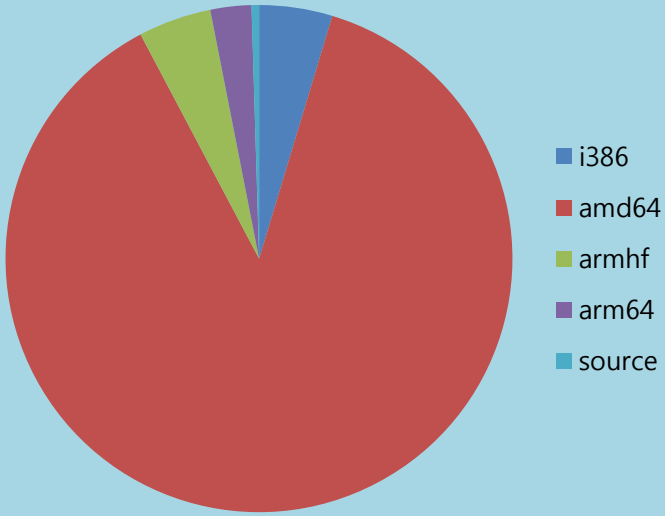
젊은세상

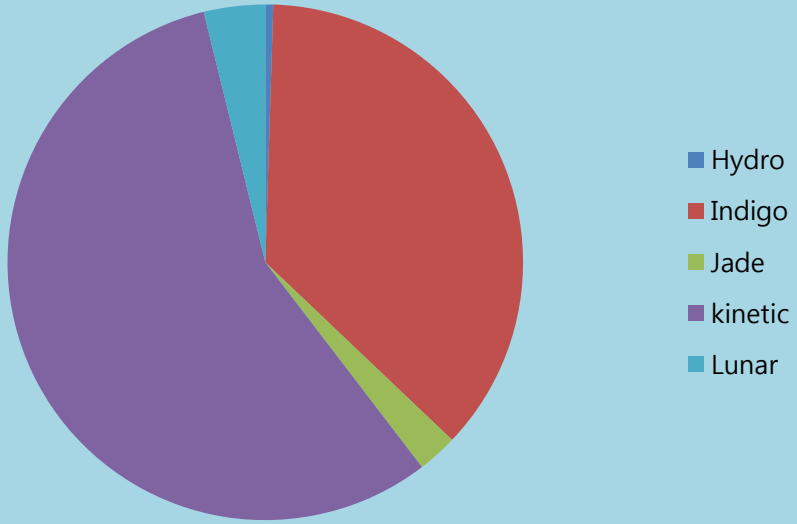
ROS 버전

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame (**Recommended**)	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015

젊은세상

ROS 이용 상황(2017.7)



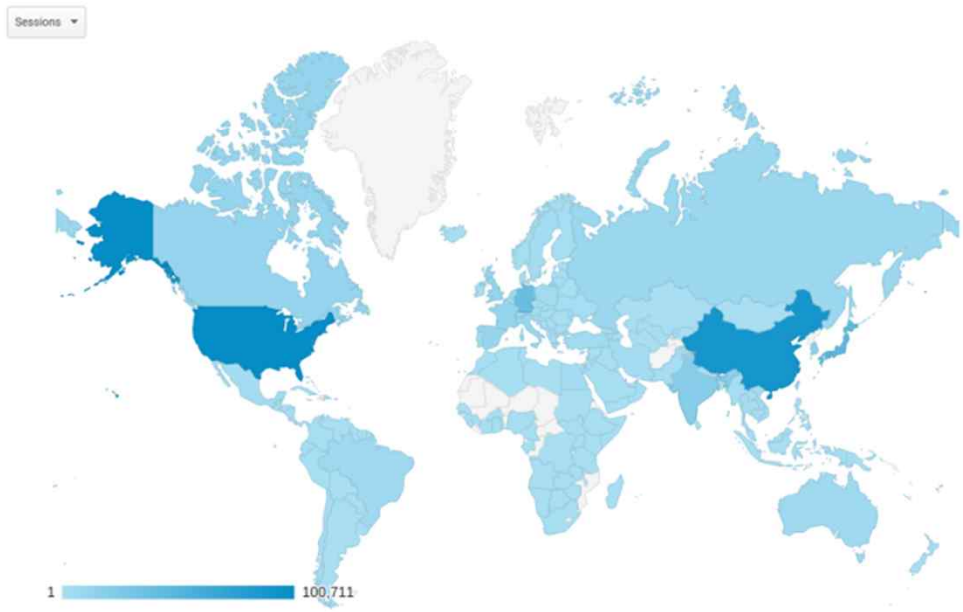


젊은세상

나라별 ROS 방문상황

1.	United States	100,711 (20.08%)
2.	China	90,120 (17.97%)
3.	Japan	45,834 (9.14%)
4.	Germany	39,590 (7.89%)
5.	India	20,632 (4.11%)
6.	South Korea	16,683 (3.33%)
7.	United Kingdom	12,784 (2.55%)
8.	Taiwan	11,809 (2.35%)
9.	Canada	11,685 (2.33%)
10.	France	11,651 (2.32%)
11.	Spain	10,445 (2.08%)
12.	Singapore	9,751 (1.94%)
13.	Italy	9,366 (1.87%)
14.	Hong Kong	9,289 (1.85%)
15.	Russia	8,380 (1.67%)
16.	Australia	6,346 (1.27%)
17.	Brazil	5,959 (1.19%)
18.	Switzerland	4,474 (0.89%)
19.	Turkey	4,399 (0.88%)
20.	Netherlands	4,343 (0.87%)
21.	Poland	4,176 (0.83%)
22.	Sweden	3,159 (0.63%)
23.	Portugal	3,150 (0.63%)
24.	Mexico	3,124 (0.62%)
25.	Greece	2,683 (0.54%)

wiki.ros.org visitor locations:



Source: Google Analytics
Site: wiki.ros.org in July 2017

넓은세상

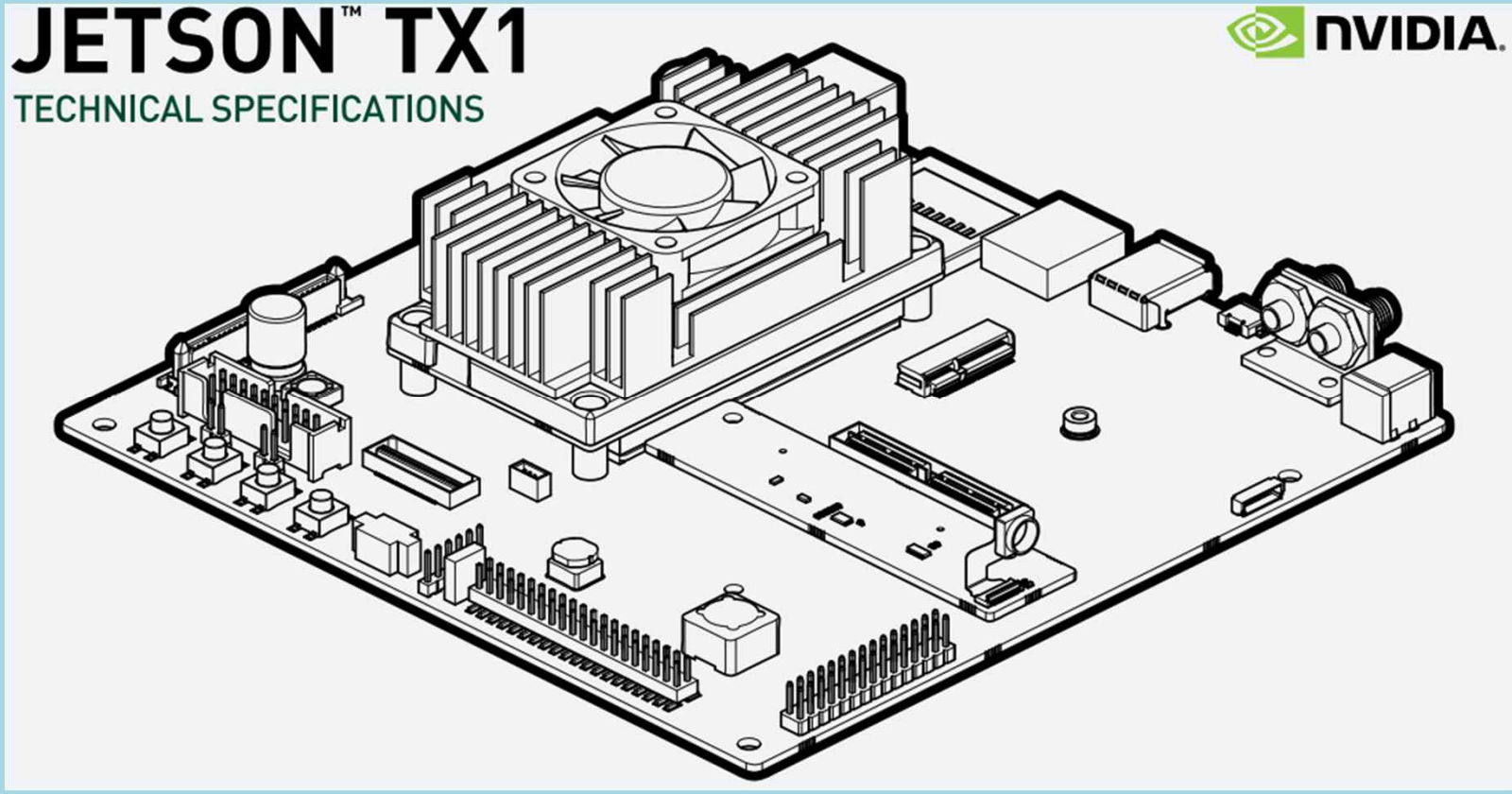
ROS install

- sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
- sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
- sudo apt-get update
- rosinstall.sh

젊은세상

젊은세상

Jetson TX



젊은세상

Jetson TX

DEVELOPER KIT	
GPU	1 TFLOP/s 256-core Maxwell
CPU	64-bit ARM A57 CPUs
Memory	4 GB LPDDR4 | 25.6 GB/s
Storage	16 GB eMMC
Connectivity	Connects to 802.11ac Wi-Fi and Bluetooth-enabled devices
Networking	10/100/1000Mbit Ethernet
Camera	5MP Fixed Focus
USB	USB 3.0 + USB 2.0
PCIE	Gen 2 1x1 (M.2) + 1x4 (full x4 slot)
Size	170mm x 170mm
Deployment	Module (Jetson TX1)

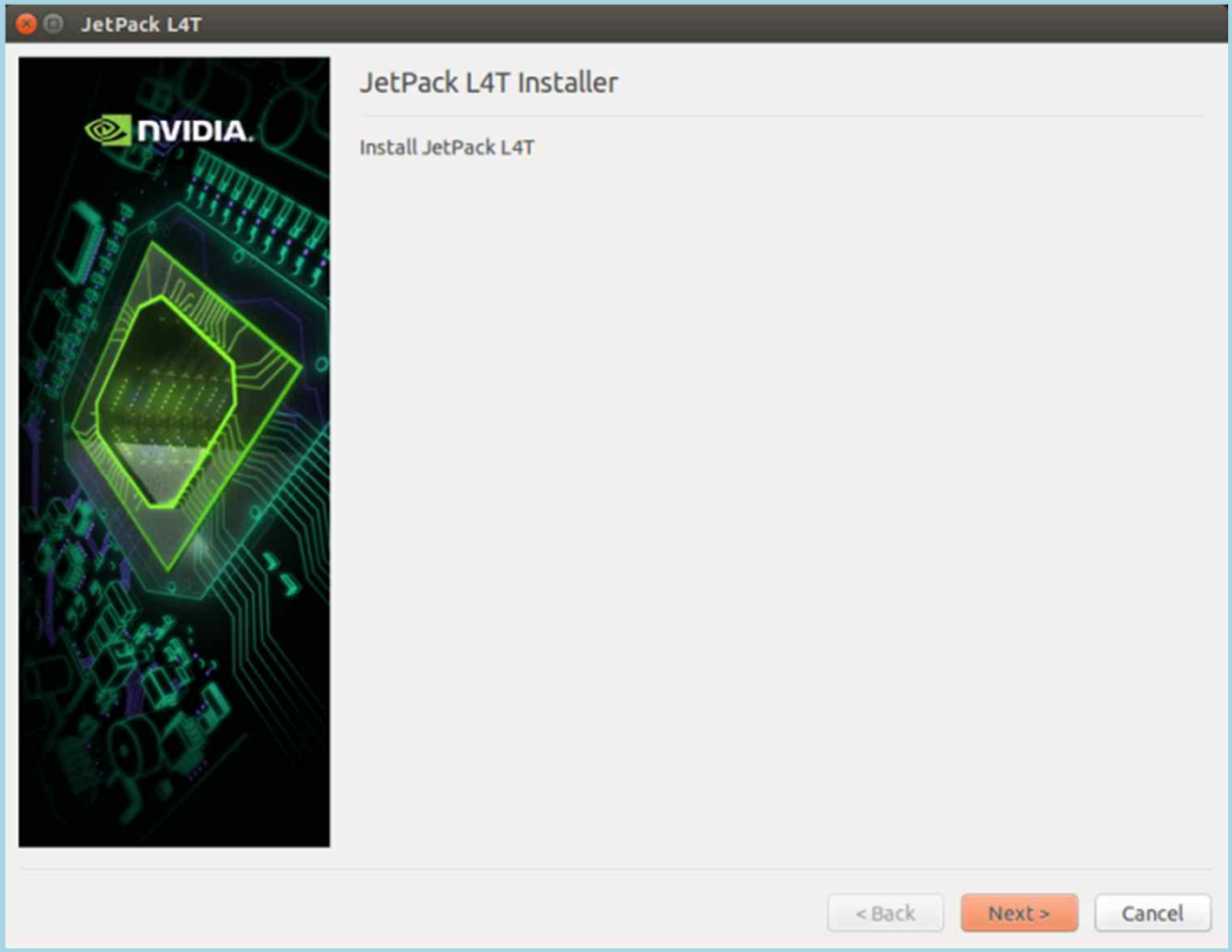
젊은세상

Jetson TX

- **Key Features in JetPack 3.1**
  - **TensorRT 2.1**
  - **cuDNN 6.0**
  - **VisionWorks 1.6**
  - **CUDA 8.0**
  - **Multimedia API**
  - **L4T : 64-bit Ubuntu 16.04, Kernel 4.4**
  - **Development Tools**
    - Tegra System Profiler 3.8
    - Tegra Graphics Debugger 2.4
    - OpenGL ES 2.0, 3.0, 3.1, and 3.2
    - OpenGL 4.3, 4.4, and 4.5

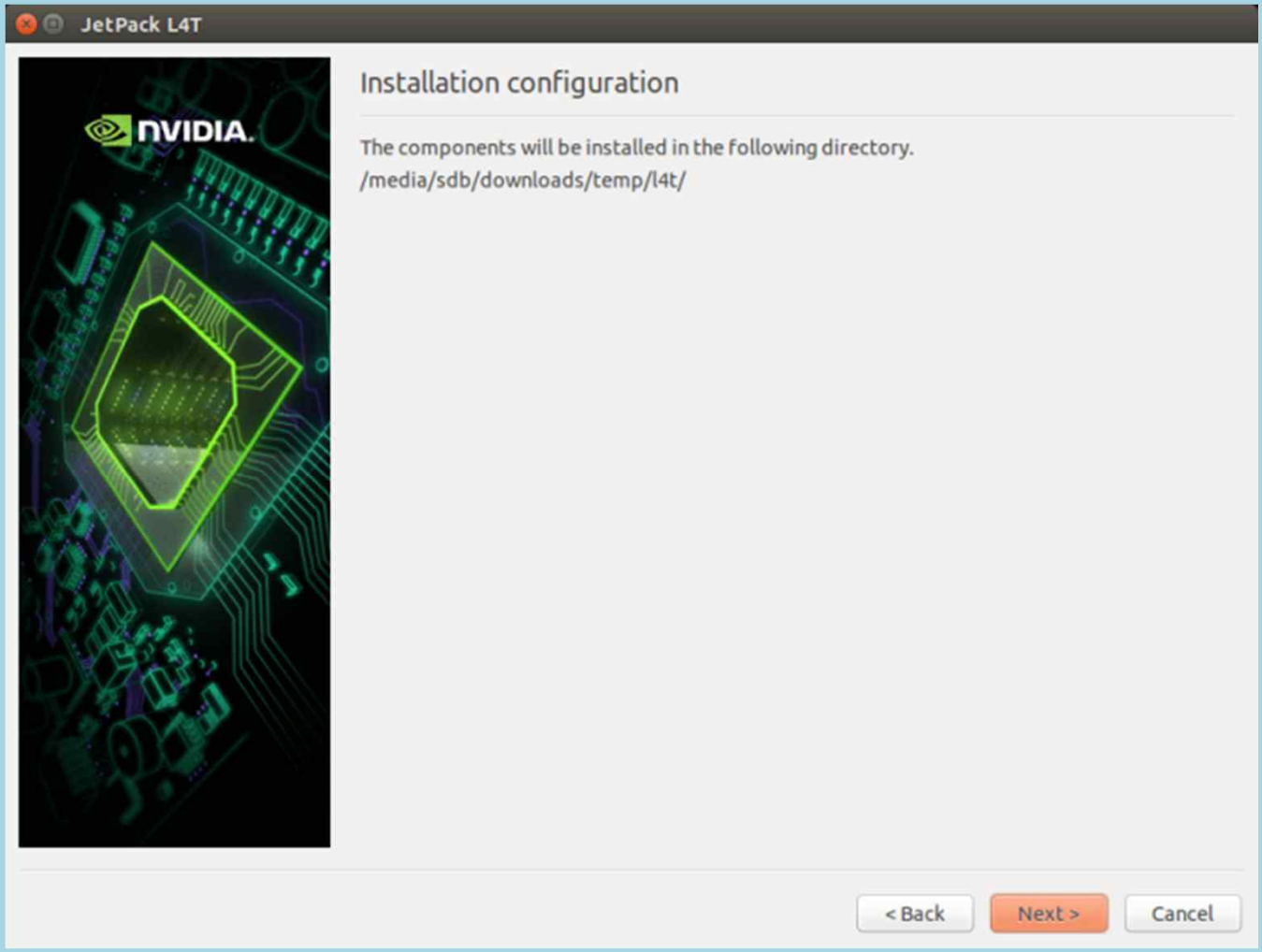
젊은세상

Jetpack 3.1 install



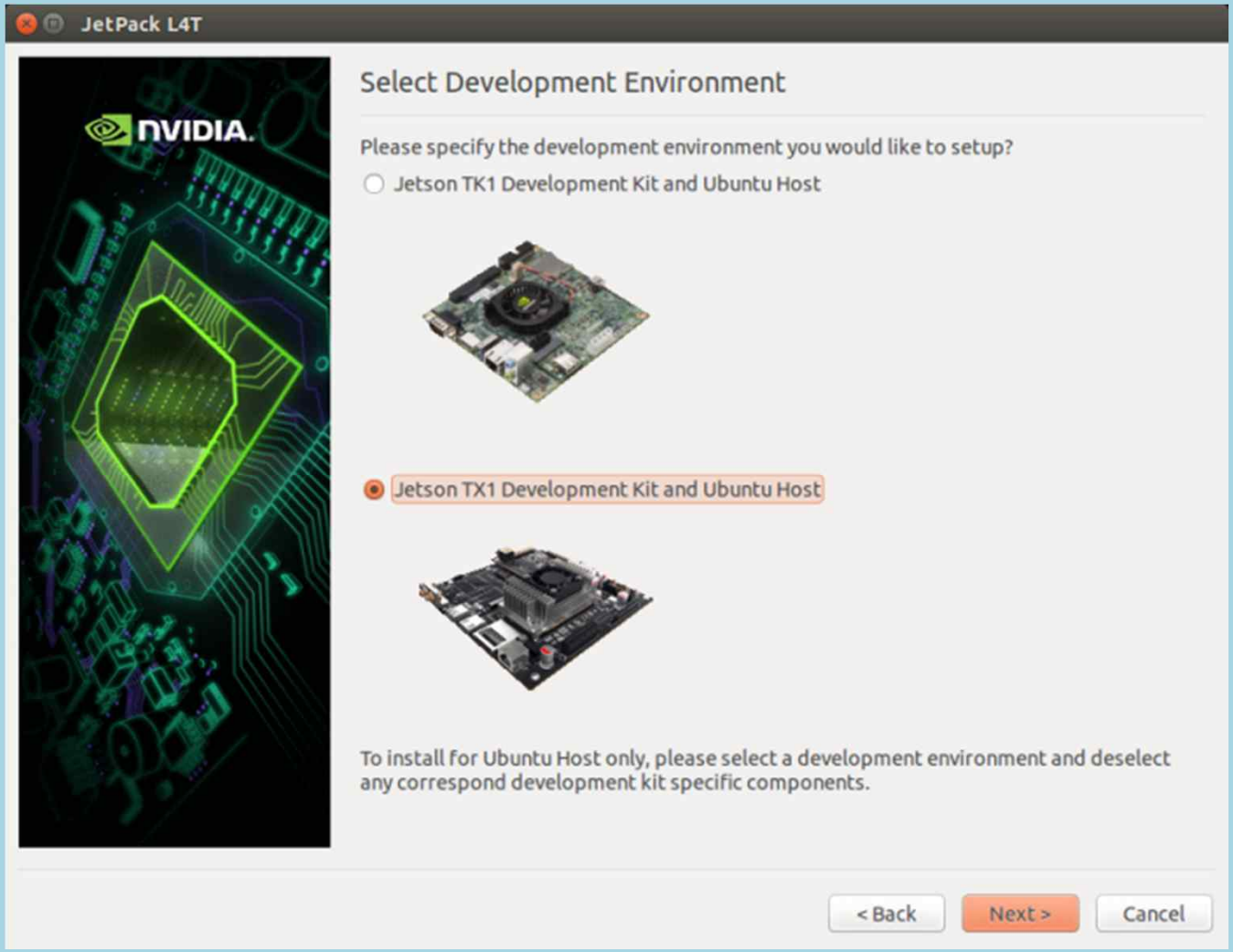
젊은세상

Jetpack 3.1 install



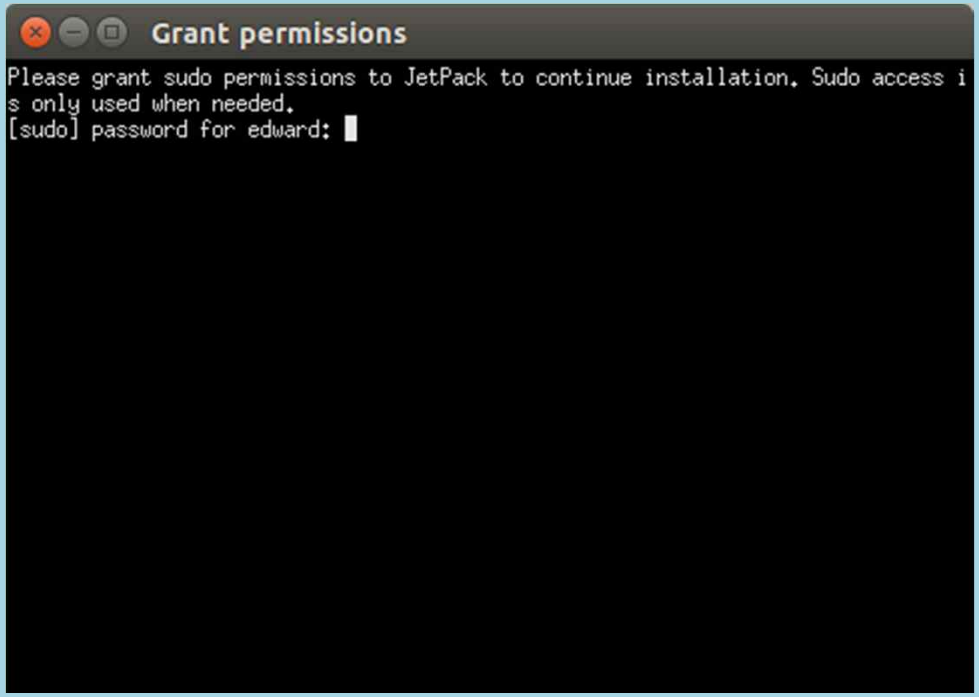
젊은세상

Jetpack 3.1 install



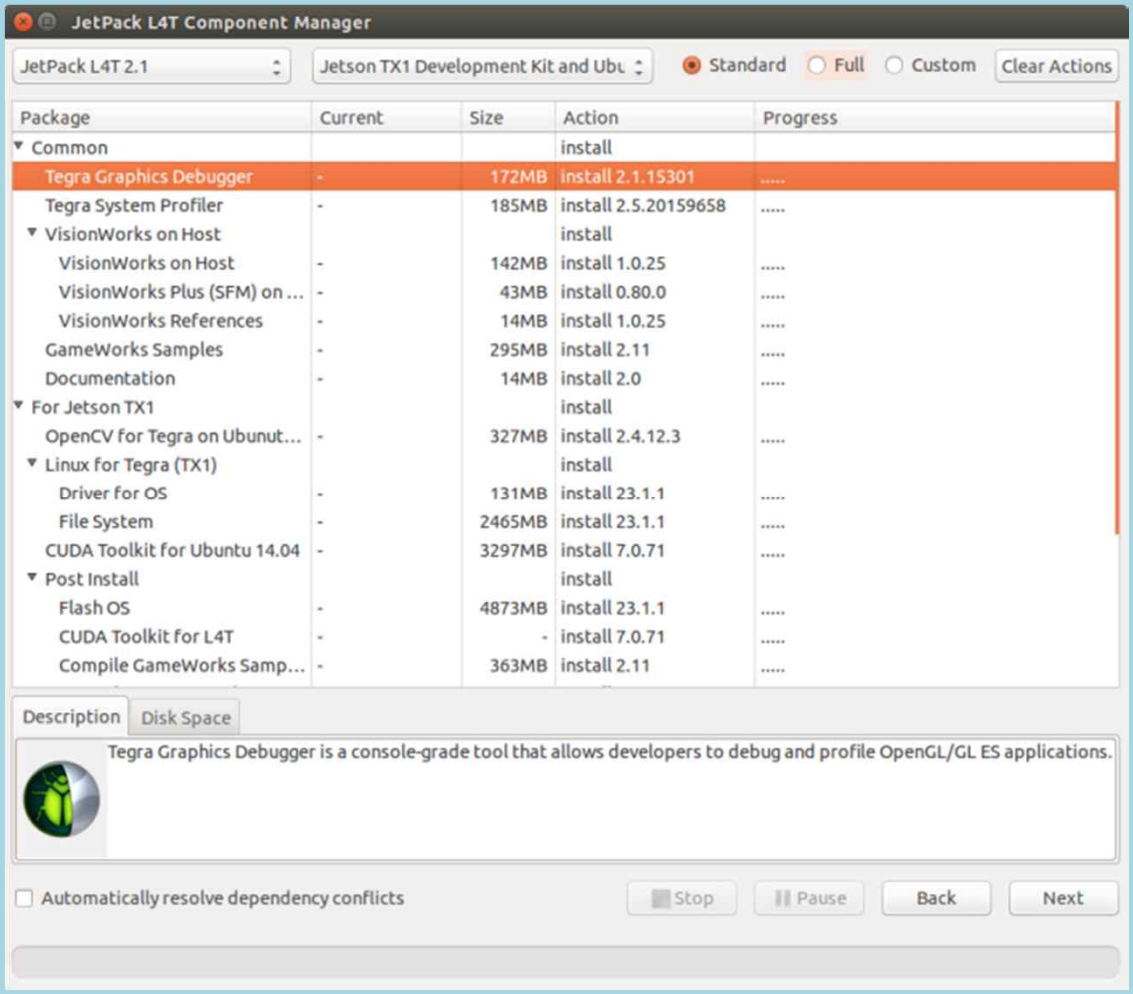
젊은세상

Jetpack 3.1 install



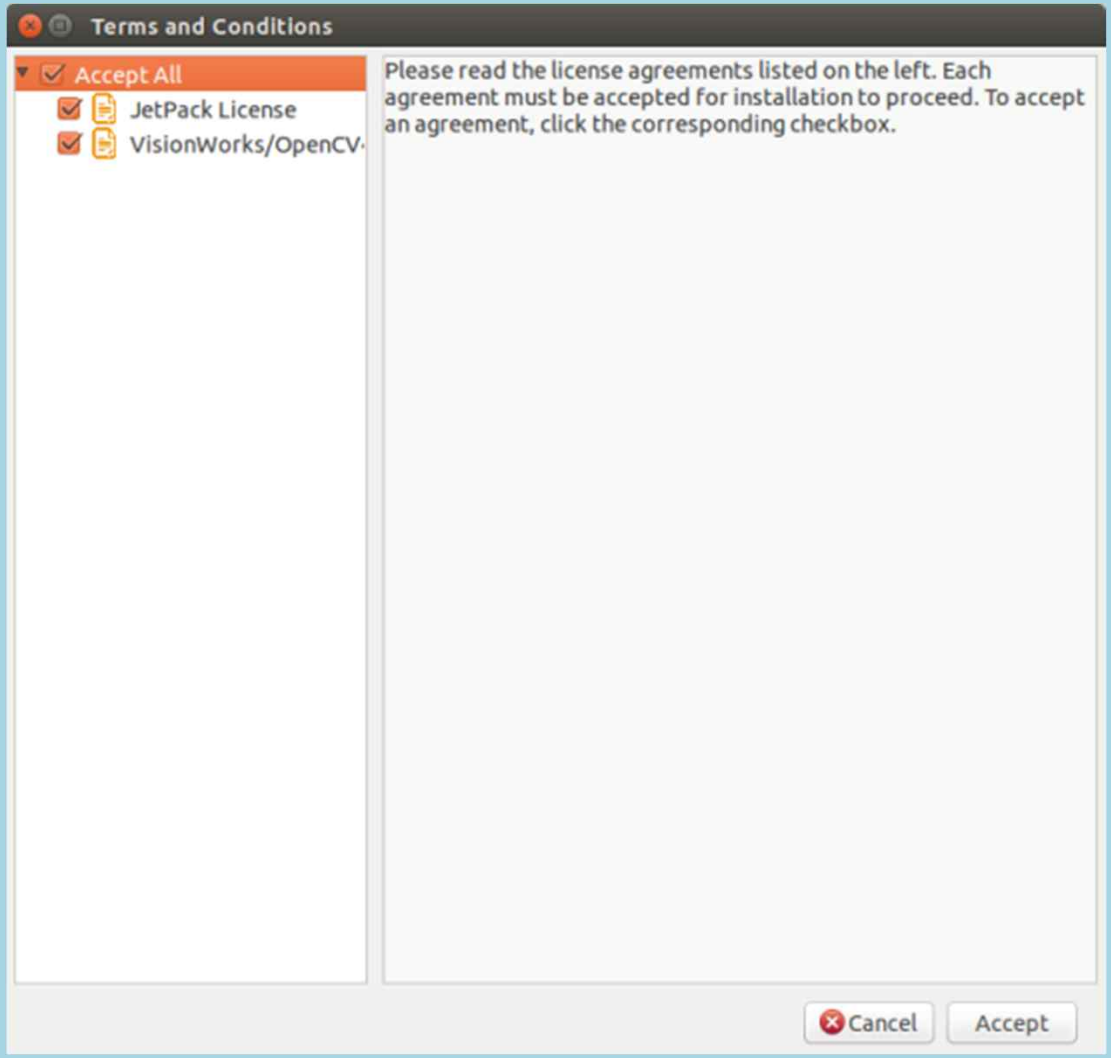
젊은세상

Jetpack 3.1 install



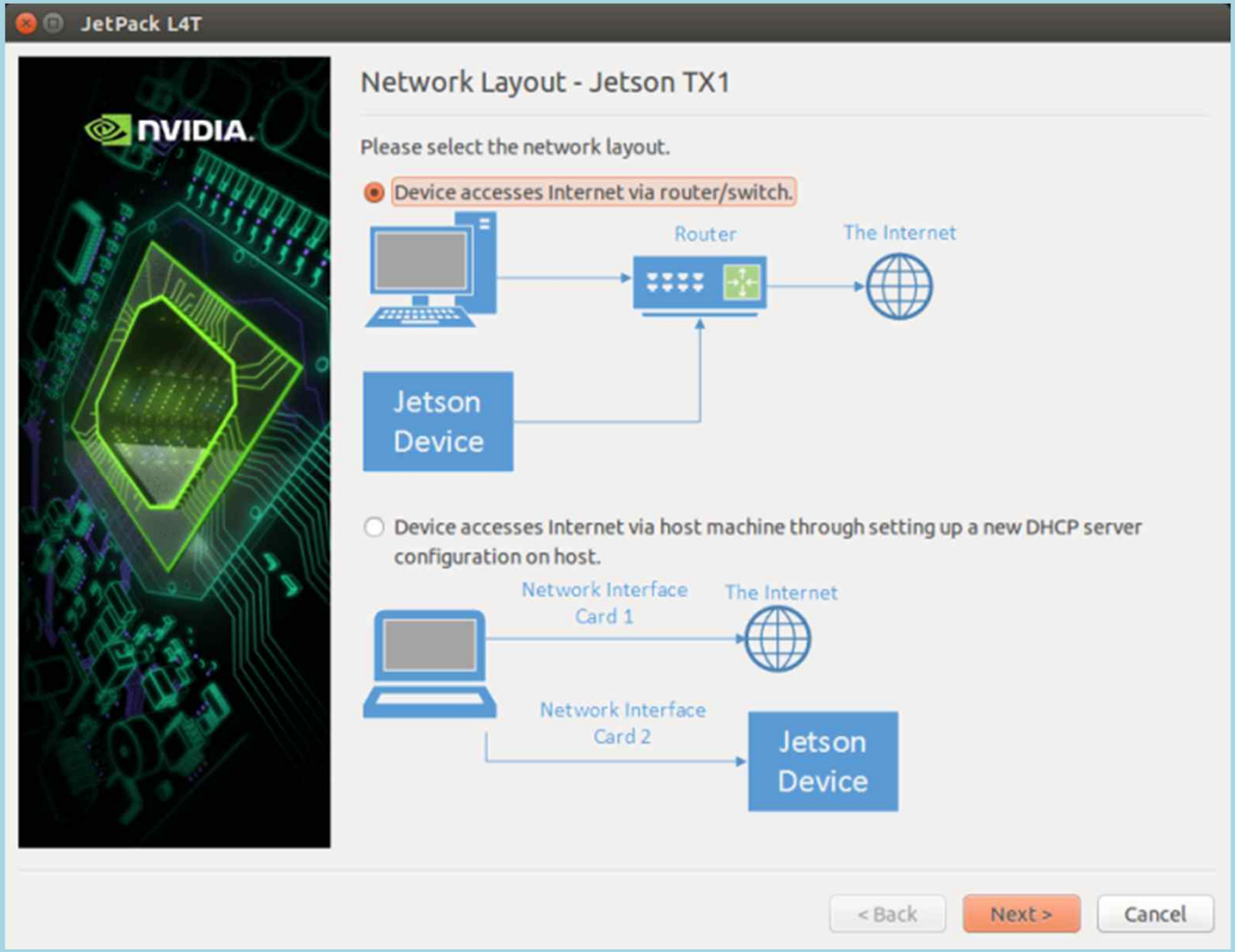
젊은세상

Jetpack 3.1 install



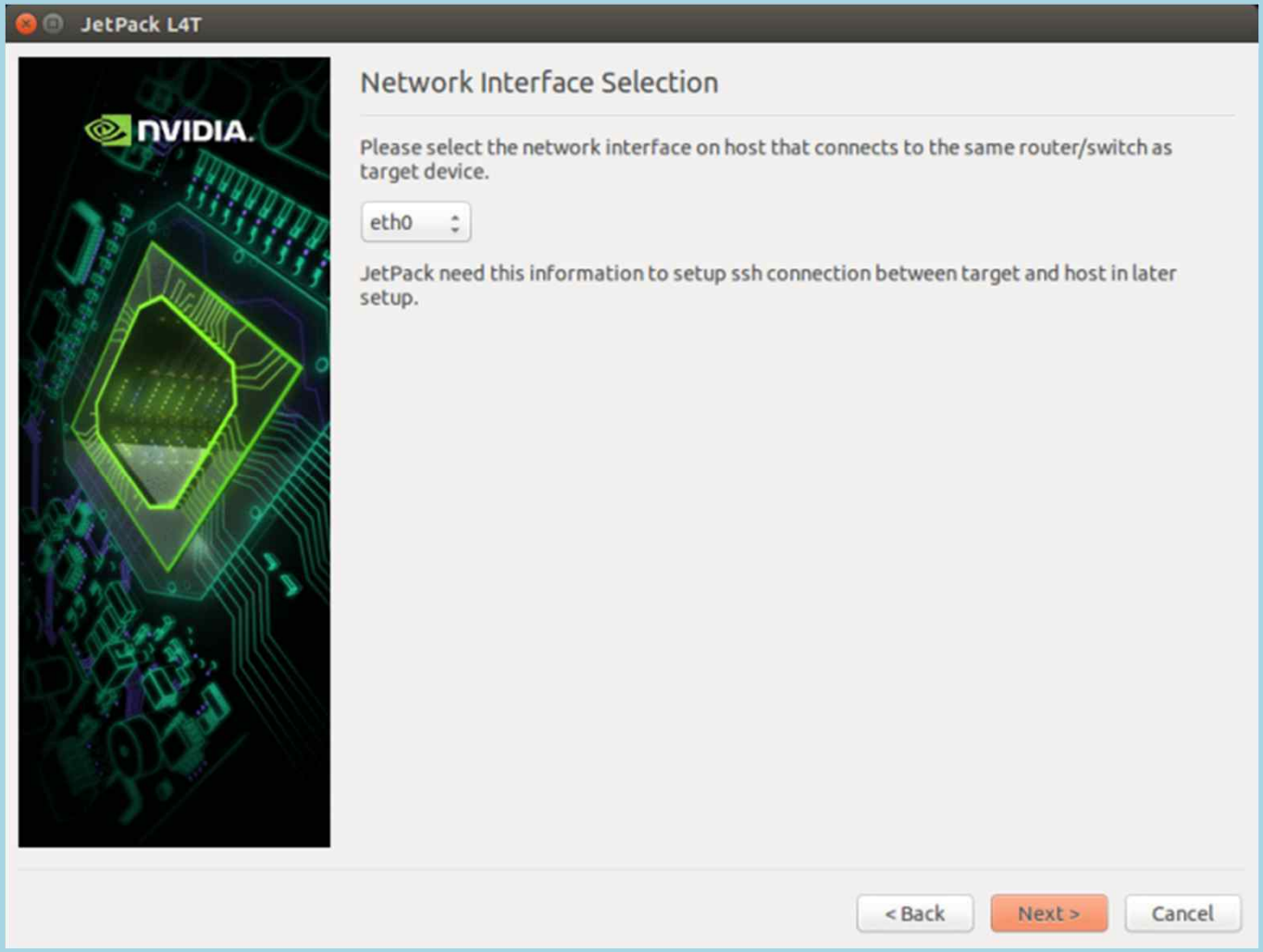
젊은세상

Jetpack 3.1 install



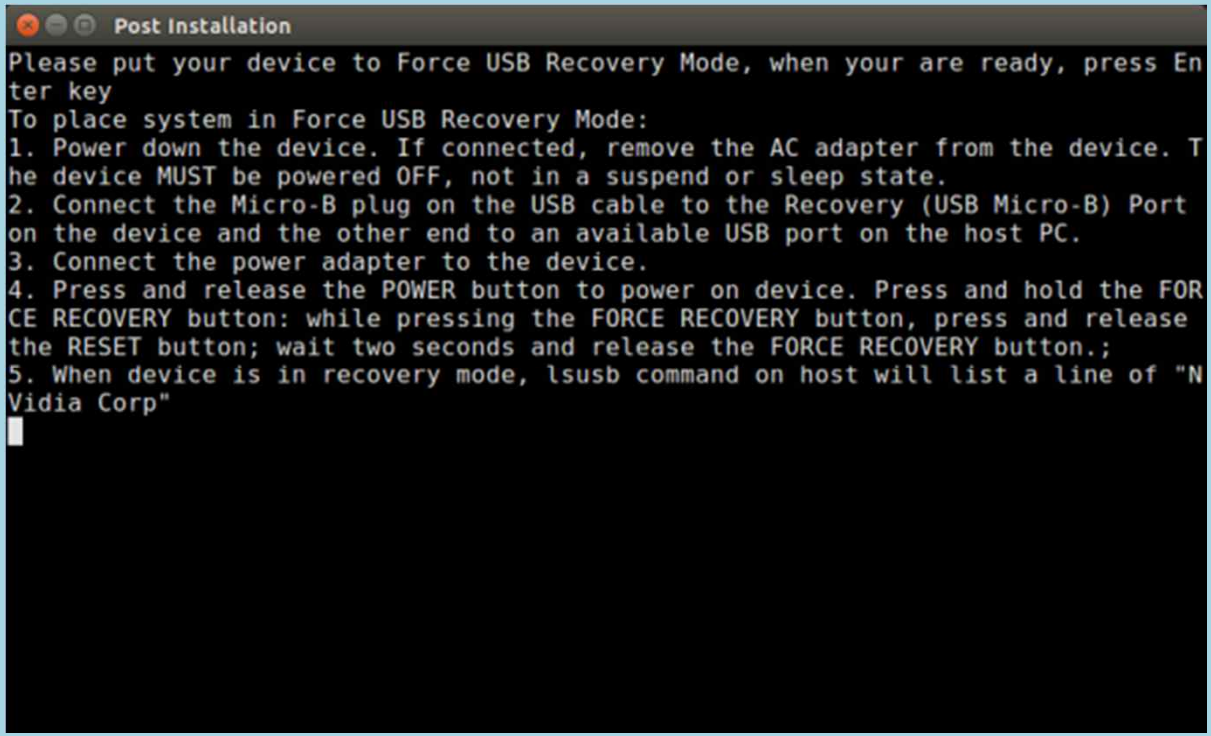
젊은세상

Jetpack 3.1 install



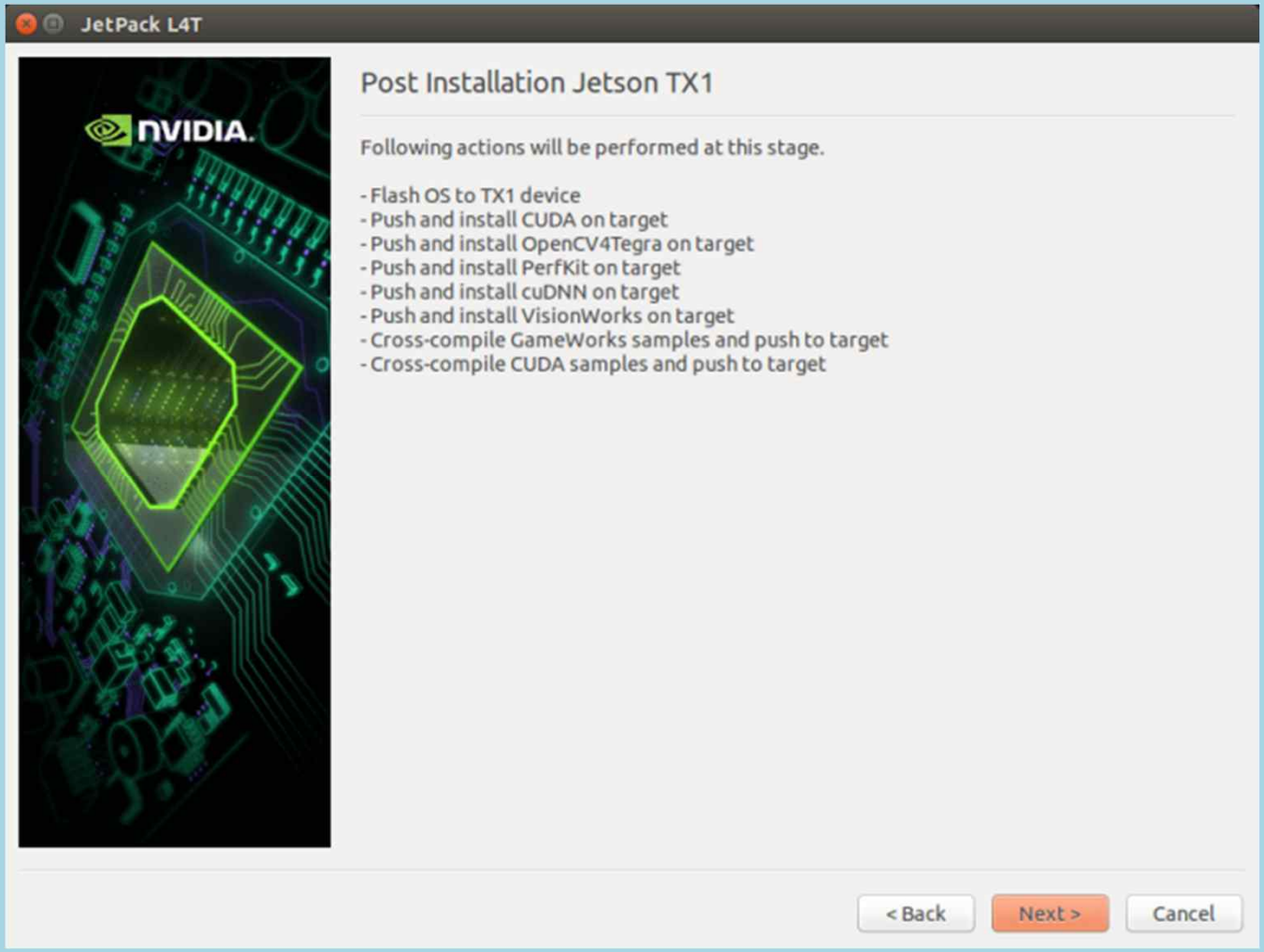
젊은세상

Jetpack 3.1 install



젊은세상

Jetpack 3.1 install



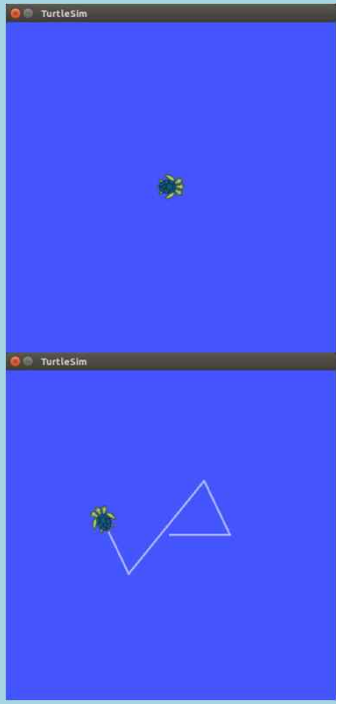
젊은세상

Jetson TX를 사용하는 이유

- 인공지능을 접목하기 위해서는 GPU가 필요하다.
  - PC의 확장슬롯을 이용한 GPU 이외에는 Jetson TX만 GPU core를 지원한다.
  - tensorflow및 caffe를 지원한다.
  - 1테라플롭스에 달하는 연산성능을 지원한다.
    - Jetson TK는 300기가플롭스
    - 인텔이 인수한 모비디우스는 100기가플롭스

젊은세상

ROS 실행해 보기



- ctrl-alt-t 키를 눌러 터미널 창을 연다
- roscore를 실행한다.
- ctrl-alt-t 키를 눌러 터미널 창을 연다
- rosrun turtlesim turtlesim_node
- ctrl-alt-t 키를 눌러 터미널 창을 연다
- rosrun turtlesim turtle_teleop_key
- 화살표키를 이용하여 화면의 거북이를 움직여 본다.

젊은세상

ROS 명령 #1

- rospack: ROS package management tool
  - rospack list
  - rospack find turtlesim
  - rospack depends turtlesim
  - rospack profile

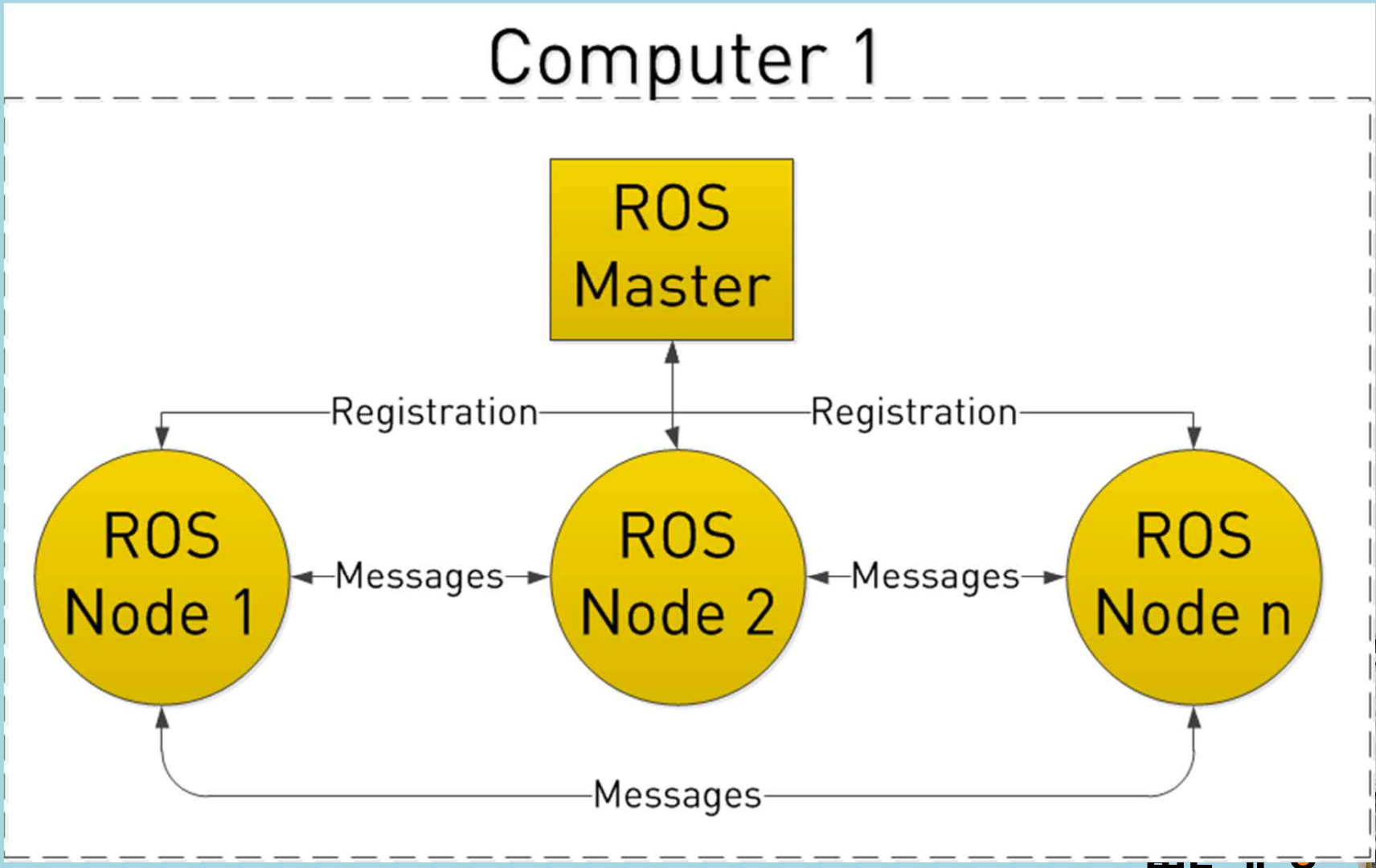
젊은세상

ROS 명령 #2

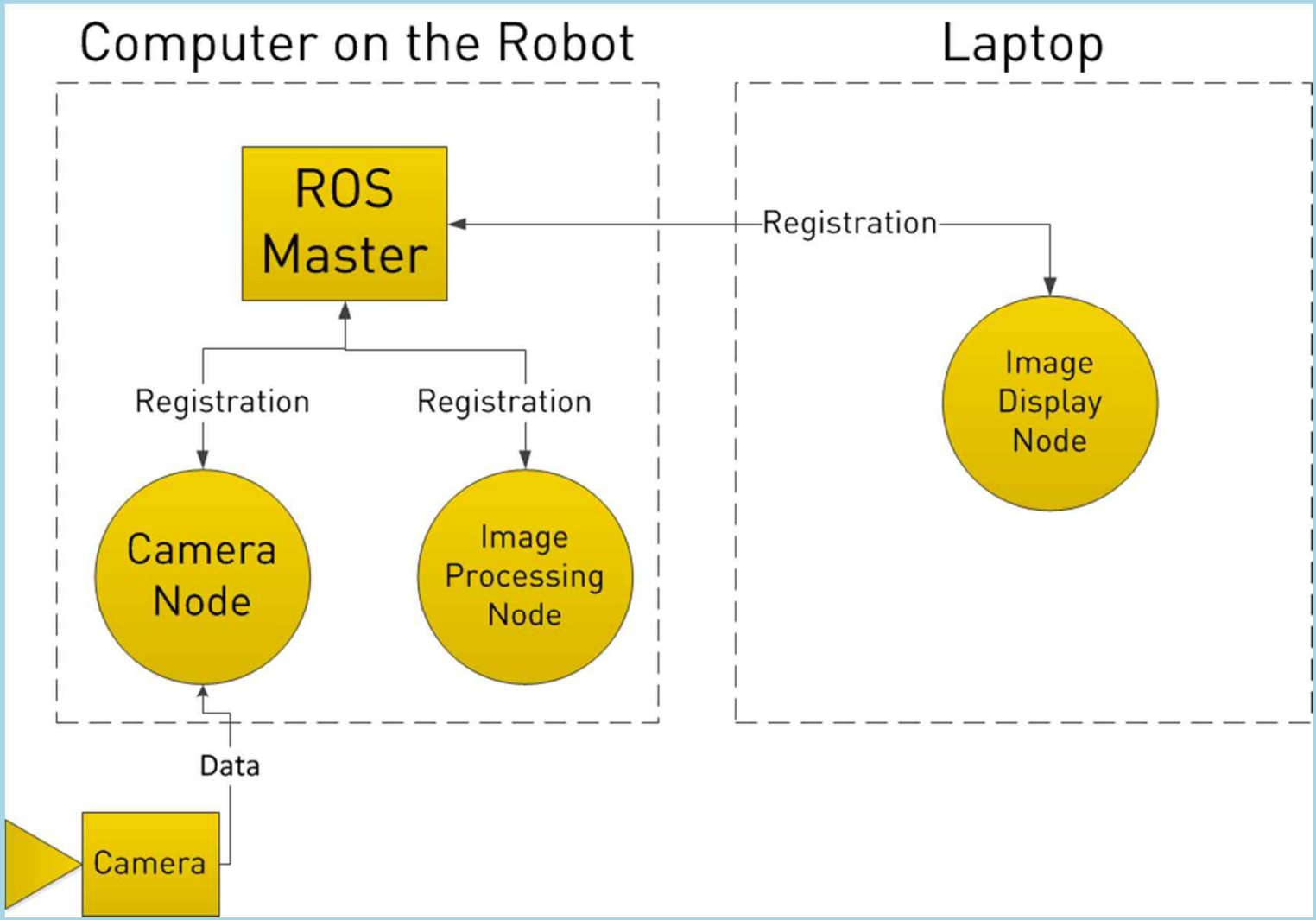
- roscd: change directory command for ROS
  - roscd
  - roscd turtlesim
- rosls: allows you to list the contents of a ROS package
  - rosls turtlesim

젊은세상

ROS 개념 #1

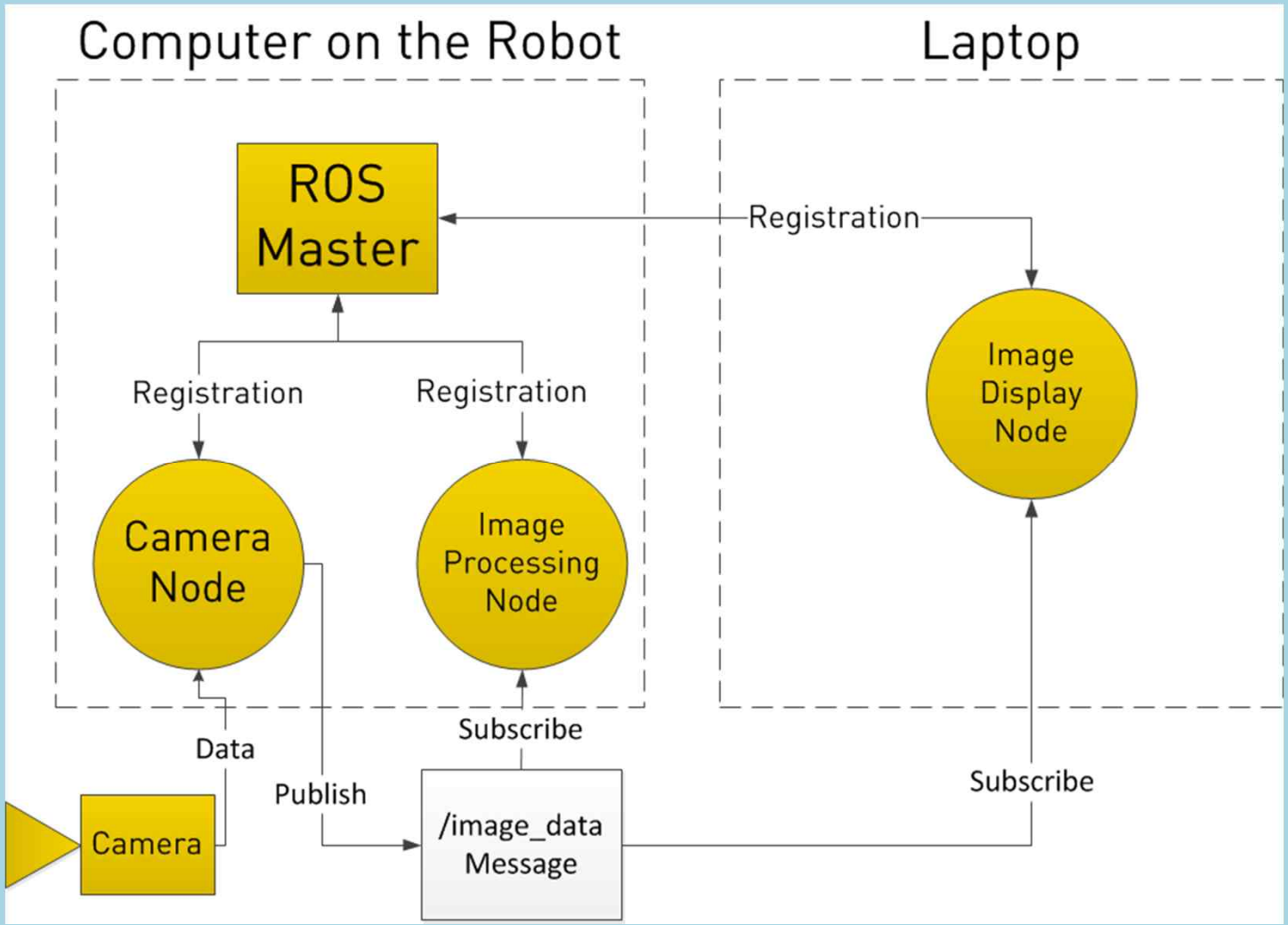

Computer 1
ROS Master
Registration
Registration
ROS Node 1
Messages
ROS Node 2
Messages
ROS Node n
Messages

ROS 개념 #2


Computer on the Robot
Laptop
ROS Master
Registration
Registration
Registration
Image Display Node
Camera Node
Image Processing Node
Data
Camera

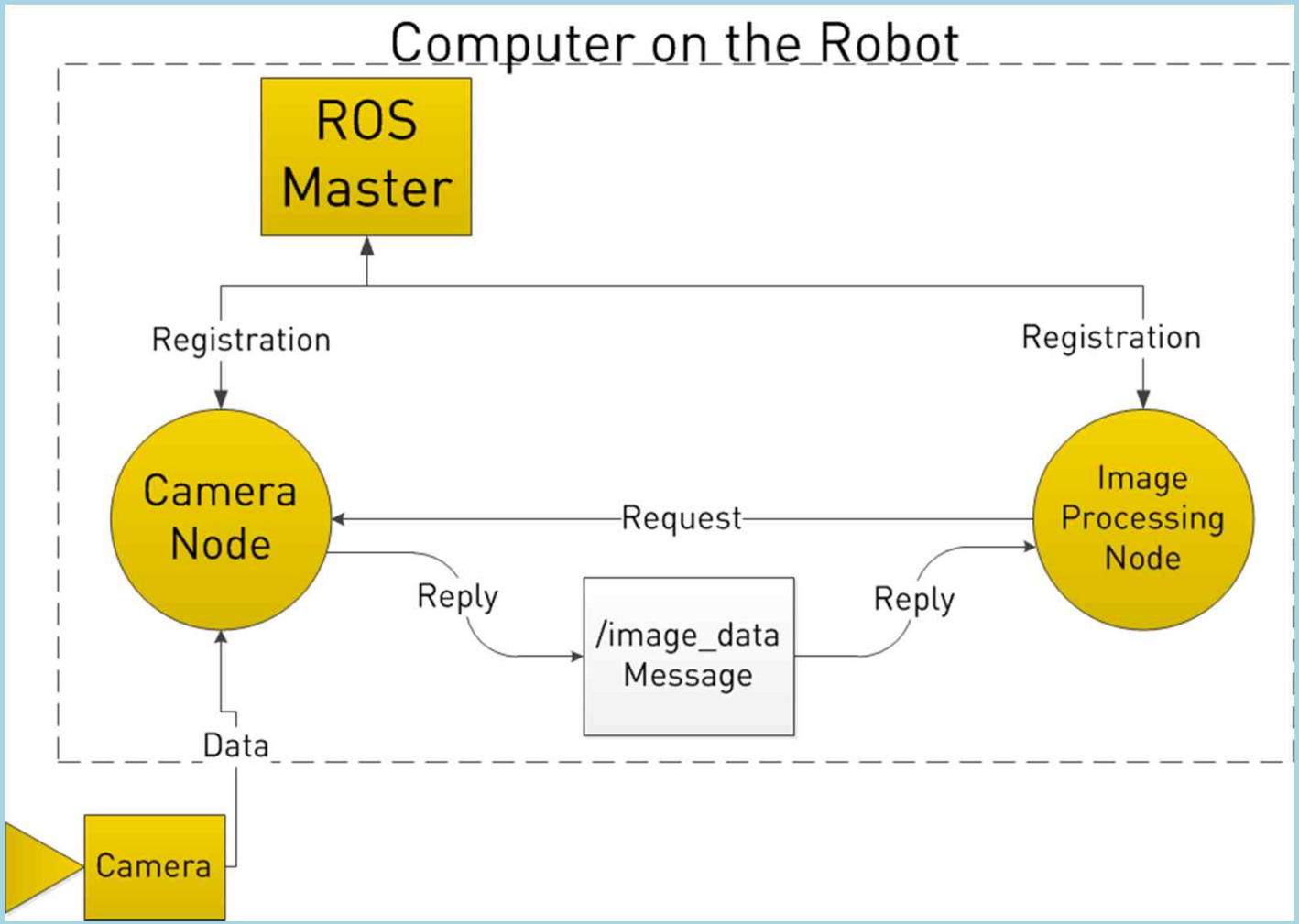
젊은세상

ROS 개념 #3


Computer on the Robot
Laptop
ROS Master
Registration
Registration
Registration
Image Display Node
Camera Node
Image Processing Node
Subscribe
Data
Publish
Subscribe
Camera
/image_data Message

젊은세상

ROS 개념 #4


Computer on the Robot
ROS Master
Registration
Registration
Camera Node
Request
Image Processing Node
Reply
/image_data Message
Reply
Data
Camera

젊은세상

ROS Master 실행

- gun@gun-Precision-M4600:~/catkin_ws$ roscore
- ... logging to /home/gun/.ros/log/5224135e-a1a1-11e7-8a4d-d067e537335b/roslaunch-gun-Precision-M4600-17845.log
- Checking log directory for disk usage. This may take awhile.
- Press Ctrl-C to interrupt
- Done checking log file disk usage. Usage is <1GB.

- started roslaunch server http://localhost:43499/
- ros_comm version 1.12.7

- SUMMARY
- ========

- PARAMETERS
- * /rosdistro: kinetic
- * /rosversion: 1.12.7

- NODES

- auto-starting new master
- process[master]: started with pid [17856]
- ROS_MASTER_URI=http://localhost:11311/

- setting /run_id to 5224135e-a1a1-11e7-8a4d-d067e537335b
- process[rosout-1]: started with pid [17869]
- started core service [/rosout]

젊은세상

ROS development tool

- ROS는 catkin 개발툴을 주로 지원하고 있다.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

- 개발을 위한 디렉토리를 만들고 초기화 한다.
- **catkin_make** 명령을 이용하여 전체 패키지를 빌드한다.

```
cd ~/catkin_ws
catkin_make
```

젊은세상

ROS development tool

- catkin_make를 수행하면 개발 작업디렉토리는 다음과 같이 구성된다.

```
catkin_ws/                -- WORKSPACE
  src/                    -- SOURCE SPACE
    ...
  build/                  -- BUILD SPACE
  devel/                  -- DEVEL SPACE
    setup.bash           \
    setup.sh             |-- Environment setup files
    setup.zsh            /
    etc/                  -- Generated configuration files
    include/              -- Generated header files
    lib/                  -- Generated libraries and other artifacts
      package_1/
        bin/
        etc/
        include/
        lib/
        share/
        ...
      package_n/
        bin/
        etc/
        include/
        lib/
        share/
    share/                -- Generated architecture independent artifacts
    ...
```

젊은세상

ROS development tool

- 소스개발을 위한 폴더는 다음과 같은 구조를 가진다.

```
workspace_folder/          -- WORKSPACE
  src/                     -- SOURCE SPACE
    CMakeLists.txt         -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt       -- CMakeLists.txt file for package_1
      package.xml          -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt       -- CMakeLists.txt file for package_n
      package.xml          -- Package manifest for package_n
```

- 개별 패키지 디렉토리는 다음과 같다.

Directory	Explanation
include/	C++ include headers
src/	Source files
msg/	Folder containing Message (msg) types
srv/	Folder containing Service (srv) types
launch/	Folder containing launch files
package.xml	The package manifest
CMakeLists.txt	CMake build file

젊은세상

ROS에서 C++, Python의 data type

Primitive Type	Serialization	C++	Python2	Python3
bool	unsigned 8-bit int	uint8_t		bool
int8	signed 8-bit int	int8_t		int
uint8	unsigned 8-bit int	uint8_t		int
int16	signed 16-bit int	int16_t		int
uint16	unsigned 16-bit int	uint16_t		int
int32	signed 32-bit int	int32_t		int
uint32	unsigned 32-bit int	uint32_t		int
int64	signed 64-bit int	int64_t	long	int
uint64	unsigned 64-bit int	uint64_t	long	int
float32	32-bit IEEE float	float		float
float64	64-bit IEEE float	double		float
string	ascii string	std::string	str	bytes
time	secs/nsecs unsigned 32-bit ints	ros::Time		rospy.Time
duration	secs/nsecs signed 32-bit ints	ros::Duration		rospy.Duration

젊은세상

example program

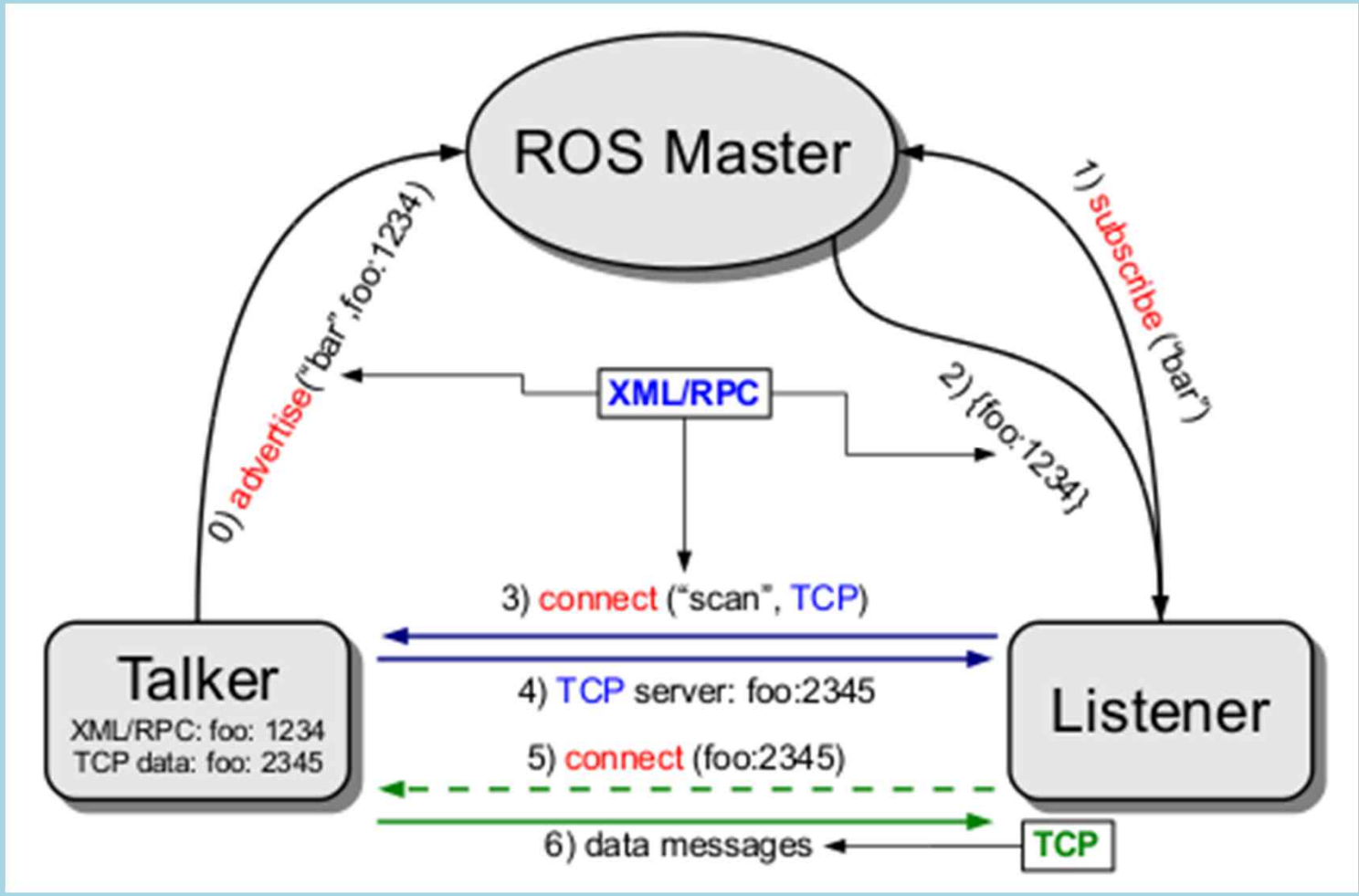
- $ cd ~/catkin_ws/src
- $ catkin_create_pkg example_pkg rospy roscpp std_msgs
- $ cd example_pkg
- $ cat CMakeLists.txt
- $ cat package.xml
- $ cd ~/catkin_ws
- $ catkin_make
- $ rospack find example_pkg

젊은세상

Topic

- 노드는 Topic을 퍼블리싱함으로써 메시지를 보낸다.
- Topic 타입은 퍼블리싱하는 메시지 타입에 의해 정의됩니다.
- 데이터가 필요한 노드는 데이터를 제공하는 Topic에 등록해야합니다
- 동일한 Topic에 대한 여러 퍼블리셔/구독자 허용한다.
- 단일 노드가 여러 Topic를 퍼블리싱 또는 구독 할 수 있습니다.
- 퍼블리셔와 구독자는 일반적으로 서로의 존재를 인식하지 못합니다.
- Publish / Subscribe 모델은 유연한 패러다임 (다대다, 단방향 전송)이며, 실행 순서가 필요하지 않습니다.

Topic


ROS Master
0) advertise("bar",foo:1234)
1) subscribe ("bar")
XML/RPC
2) {foo:1234}
3) connect ("scan", TCP)
Talker
XML/RPC: foo: 1234
TCP data: foo: 2345
4) TCP server: foo:2345
Listener
5) connect (foo:2345)
6) data messages
TCP

젊은세상

토픽 확인하기

- $ roscore
- $ rosrun turtlesim turtlesim_node
- $ rostopic list
  /rosout
  /rosout_agg
  /turtle1/cmd_vel
  /turtle1/color_sensor
  /turtle1/pose
- $ rosmsg show turtlesim/Pose
  float32 x
  float32 y
  float32 theta
  float32 linear_velocity
  float32 angular_velocity
- $

젊은세상

토픽 읽기

- 터틀봇의 Pose 토픽을 읽어 콘솔에 출력하는 프로그램을 만들어 본다.
- catkin_make를 사용하여 수행 가능한 패키지를 만든다.

Topic example program #1

- $ cd ~/catkin_ws/src
- $ catkin_create_pkg readtopic rospy roscpp std_msgs
- $ cd readtopic
- $ gedit CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(readtopic)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

catkin_python_setup()

catkin_package(
)

include_directories(
  ${catkin_INCLUDE_DIRS}
)
```

Topic example program #2

- $ gedit package.xml

```
<?xml version="1.0"?>
<package>
  <name>readtopic</name>
  <version>0.0.0</version>
  <description>The readtopic package</description>

  <maintainer email="gun@todo.todo">gun</maintainer>

  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <export>
  </export>
</package>
```

Topic example program #3

- $ touch include/readtopic/__init__.py
- $ gedit setup.py

```
setup_args = generate_distutils_setup(
    packages=['readtopic'],
    package_dir={'': 'include'},
)
setup(**setup_args)
```

- $ gedit src/subscriber_node.py

```
#!/usr/bin/env python
import rospy
from turtlesim.msg import Pose

def callback(data):
    rospy.loginfo("x=%f y= %f" % (data.x, data.y))

def readtopic():
    rospy.init_node('turtlesim_listener', anonymous=True)
    rospy.Subscriber("/turtle1/pose", Pose, callback)

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

if __name__ == '__main__':
    readtopic()
```

젊은세상

Topic example program #4

- $ cd ~/catkin_ws
- $ catkin_make
- $ cd src/readtopic
- $ rosrun readtopic subscriber_node.py

[INFO] [1506266665.656729]: x=7.591965 y= 7.206161
[INFO] [1506266665.673280]: x=7.591965 y= 7.206161
[INFO] [1506266665.688478]: x=7.591965 y= 7.206161
[INFO] [1506266665.704757]: x=7.591965 y= 7.206161
[INFO] [1506266665.721043]: x=7.591965 y= 7.206161
[INFO] [1506266665.737289]: x=7.591965 y= 7.206161
[INFO] [1506266665.752497]: x=7.591965 y= 7.206161
[INFO] [1506266665.770509]: x=7.591965 y= 7.206161
[INFO] [1506266665.784663]: x=7.591965 y= 7.206161
[INFO] [1506266665.801000]: x=7.591965 y= 7.206161
[INFO] [1506266665.816003]: x=7.591965 y= 7.206161
[INFO] [1506266665.832605]: x=7.591965 y= 7.206161
[INFO] [1506266665.849051]: x=7.591965 y= 7.206161

Topic example program #5

- ctrl-alt-t를 눌러 터미널 생성한다.
- $ rosrun turtlesim turtle_teleop_key

Reading from keyboard

Use arrow keys to move the turtle.

- 화살표 키를 움직인다.
- readtopic 패키지 프로그램 창의 데이터가 수정되는 것을 확인 할수 있다

젊은세상

젊은세상

Topic send example program #1

- $ cd ~/catkin_ws/src
- $ catkin_create_pkg writetopic rospy roscpp geometry_msgs std_msgs

Created file writetopic/package.xml

Created file writetopic/CMakeLists.txt

Created folder writetopic/include/writetopic

Created folder writetopic/src

Successfully created files in /home/gun/catkin_ws/src/writetopic. Please adjust the values in package.xml.

- $ cd writetopic

젊은세상

Topic send example program #1

- $ gedit CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(writetopic)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  geometry_msgs
  genmsg
)

catkin_python_setup()

add_message_files(
)

generate_messages(
   DEPENDENCIES
   std_msgs
)

catkin_package(
  CATKIN_DEPENDS message_runtime
)

include_directories(
  ${catkin_INCLUDE_DIRS}
)
```

젊은세상

Topic send example program #2

- $ gedit package.xml

```
<?xml version="1.0"?>
<package>
  <name>writetopic</name>
  <version>0.0.0</version>
  <description>The writetopic package</description>

  <maintainer email="gun@todo.todo">gun</maintainer>

  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>geometry_msgs</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <run_depend>geometry_msgs</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <export>
  </export>
</package>
```

Topic send example program #3

- $ touch include/writetopic/__init__.py
- $ gedit setup.py

```
setup_args = generate_distutils_setup(
    packages=['readtopic'],
    package_dir={'': 'include'},
)
setup(**setup_args)
```

- $ rostopic type /turtle1/cmd_vel

```
geometry_msgs/Twist
```

- $ rosmsg show geometry_msgs/Twist

```
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

Topic send example program #4

- $ gedit src/publisher_node.py

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

def move():
    # Initialize the node with rospy
    rospy.init_node('publisher_node')
    velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist,
queue_size=10)
    vel_msg = Twist()

    speed = input("Input your speed:")
    distance = input("Type your distance:")
    isForward = 1

    #Checking if the movement is forward or backwards
    vel_msg.linear.x = abs(speed)

    #Since we are moving just in x-axis
    vel_msg.linear.y = 0
    vel_msg.linear.z = 0
    vel_msg.angular.x = 0
    vel_msg.angular.y = 0
    vel_msg.angular.z = 0
    while not rospy.is_shutdown():

        #Setting the current time for distance calculus
        t0 = float(rospy.Time.now().to_sec())
        current_distance = 0

        #Loop to move the turtle in an specified distance
        while(current_distance < distance):
            #Publish the velocity
            velocity_publisher.publish(vel_msg)
            #Takes actual time to velocity calculus
            t1=float(rospy.Time.now().to_sec())
            #Calculates distancePoseStamped
            current_distance= speed*(t1-t0)
        #After the loop, stops the robot
        vel_msg.linear.x = 0
        #Force the robot to stop
        velocity_publisher.publish(vel_msg)
        rospy.spin()

if __name__ == '__main__':
    try:
        #Testing our function
        move()
```

젊은세상

Topic send example program #5

- $ cd ~/catkin_ws
- $ catkin_make
- $ cd src/writetopic
- $ rosrun writetopic publicher_node.py

Input your speed:1
Type your distance:1

- 연속해서 속도와 거리를 입력하면 turtlesim의 터틀 봇이 움직인다.

젊은세상