



# 실전 ROS 시스템 구축

Python, Matlab을 이용한 ROS

Donghoon Park

Copyright © 2016 Donghoon Park

PUBLISHED BY PUBLISHER

SIGMA.SNU.AC.KR

GPL License

*First printing, June 2016*

# Contents

	Part1
<b>1</b>	<b>ROS 개요 및 환경설정 .....</b> <b>7</b>
1.1	ROS란 무엇인가 <b>7</b>
1.2	ROS를 사용하기 위해서는? <b>8</b>
1.3	Ubuntu 설치 <b>8</b>
1.4	ROS 설치 <b>10</b>
1.5	<b>Python Development Environment</b> <b>12</b>
1.5.1	Pycharm ..... <b>12</b>
1.5.2	Python Package Install ..... <b>13</b>
1.5.3	Atom Editor Install ..... <b>14</b>
1.6	<b>Matlab Install</b> <b>15</b>
<b>2</b>	<b>ROS Basics .....</b> <b>17</b>
2.1	<b>ROS Topic</b> <b>17</b>
2.2	<b>Topic Publisher</b> <b>18</b>
2.2.1	Rqt를 이용한 Topic Publisher ..... <b>18</b>
2.2.2	Python을 이용한 Topic ..... <b>19</b>
	<b>Bibliography .....</b> <b>21</b>
	<b>Books</b> <b>21</b>
	<b>Articles</b> <b>21</b>
	<b>Index .....</b> <b>23</b>



# Part1

<b>1</b>	<b>ROS 개요 및 환경설정 .....</b>	<b>7</b>
1.1	ROS란 무엇인가	
1.2	ROS를 사용하기 위해서는?	
1.3	Ubuntu 설치	
1.4	ROS 설치	
1.5	Python Development Environment	
1.6	Matlab Install	
<b>2</b>	<b>ROS Basics .....</b>	<b>17</b>
2.1	ROS Topic	
2.2	Topic Publisher	
	<b>Bibliography .....</b>	<b>21</b>
	Books	
	Articles	
	<b>Index .....</b>	<b>23</b>





## 1. ROS 개요 및 환경설정

### 1.1 ROS란 무엇인가

ROS는 Robot Operating System으로, 로봇과 관련된 여러 패키지를 제공하고, 특히 프로시저간의 안전한 통신 규약을 제공하는 프로그램입니다. Operating System이라는 이름 때문에 오히려 ROS의 개념에 대해 헷갈려하는 분들이 계신데, ROS는 OS가 하는 일을 일부 도맡아서 하는 프로그램으로 생각하시면 됩니다.

ROS는 2016년 6월 시점으로 ROS Kinetic 버전까지 나와있으며, ROS의 마스코트는 왜인지 모르겠지만 거북이이고, 레퍼런스 로봇은 거북기(turtlebot)이고, 예제도 거북이로 시작합니다. ROS는 설치하는 과정 자체가 기존 Linux유저 기준으로 어려운 편은 아니지만, 로봇을 만드는 사람들이 모두 Linux에 익숙한 것은 아니기 때문에 설치 난이도가 낮은 편은 아닙니다. 어떻게 보면 매우 어려운 설치 과정을 거치는데, 이런 고생을 하면서까지 왜 ROS를 사용해야 하는지에 대해 이야기해보도록 하죠.



ROS의 필요성에 대해 이야기하려면 ROS가 어떤 것인지부터 이야기해야 할 것입니다. 프로그래머의 입장에서 보면, ROS는 일종의 라이브러리입니다. Python, C++, Matlab, Javascript 등에서 돌아가는 라이브러리를 공식적/비공식적으로 제공하며, 프로그램 언어에서 쉽게 불러와서 사용할 수 있습니다. ROS 라이브러리가 제공하는 API는 상당히 단순합니다. 데이터를 저장하고 불러오는 두 가지 방식에 대한 함수(or Functional Block)를 제공할 뿐입니다. 여기서

의문이 들 수 있습니다. 리눅스에 조금 익숙하신 분이라면 알겠지만 리눅스에서는 변수 공유를 그다지 어려운 과정을 거치지 않고 할 수 있습니다. 왜 굳이 ROS를 써야 하는 것일까요? 그 해답은 ROS의 호환성에 있다고 할 수 있습니다. ROS는 다른 언어간에도 심지어 다른 하드웨어간에도 변수 공유를 위한 프로토콜을 제공합니다. 그리고 무엇보다도 프로그램을 패키지화 시키고 나면, 우리가 프로그램의 내부를 아예 모르고도 그 패키지에서 뽑아주는 변수만 사용할 수 있다는 점이 가장 큰 장점이죠. 사실상의 하드웨어 플러그 앤 플레이를 진정하게 가능하게 하는 것이죠. ROS 패키지를 이용할 때는 소프트웨어 패키지는 설치 후 실행만 하면 됩니다. 하드웨어 연결이 필요한 패키지는 동일한 하드웨어 구성 후 꽂고, 패키지를 실행하면 됩니다. 기존에 개발한 하드웨어에 대한 완벽한 재사용성을 확보할 수 있는 것이죠. 임베디드 환경에서 구르던 로봇 프로그래머들에게 조금 더 감을 잡을 수 있게 예시를 들어보자면, 다른 사람이 짠 아두이노 시스템을 컴퓨터에 꽂기만 하면 주요 변수를 ROS가 directory 형식으로 정리해서 보여준다는 말도 안 되는 일이 가능해집니다.

결론적으로, 하드웨어 및 소프트웨어의 모듈화 및 재사용성을 극한까지 추구할 수 있는 구조가 ROS라고 보시면 됩니다.

## 1.2 ROS를 사용하기 위해서는?

ROS를 사용하기 위해서는 먼저 시스템을 구성해야 합니다. ROS는 보통 Ubuntu라는 리눅스 배포판 위에 올려서 쓰는 것이 일반적입니다. Ubuntu 위에서는 ROS를 비교적 쉽게 명령줄 도구를 이용해서 설치할 수 있습니다. 또한, Ubuntu는 리눅스 배포판 중 매우 유명한 측면 속하며 ARM 보드를 상당히 잘 지원하기 때문에 소형 로봇 시스템에도 올릴 수 있다는 점에서 Ubuntu 위에서 ROS 구성 후 실습을 하시는 것을 권장합니다.

그리고, ROS에 접속하는 프로그램을 짜는 과정 또한 필요하기 때문에 개발용 프로그램 언어 환경 또한 설치해야 합니다. 보통 ROS 접속 프로그램을 짤 때는 Python, C++ 등의 언어를 사용합니다. 이 책에서는 Python을 주 개발 언어로 사용하는 과정에 대해 이야기하겠습니다. 보다 나은 성능을 원한다면 C++을 이용하는 것을 권장합니다. ros 기본 메뉴얼은 C++로 작성되어 있기 때문에, 기본 ros 매뉴얼을 따라서 한다면 C++ 프로그래밍은 어렵지 않게 할 수 있습니다. 이 교재에서 Python을 주 언어로 사용하는 이유는 스크립트 인터프리터를 이용한 실시간 코드 확인이 가능하기 때문입니다. 다시 한번 강조하지만 프로그램이 무거운 경우 C++에서의 개발을 권장합니다.

또한 이것은 필자의 개인적인 추천인데, Matlab을 활용할 수 있는 환경에 있고, 로봇의 Computing Power가 충분히 좋은 경우 Matlab의 Simulink를 사용하는 것을 추천합니다. Matlab Simulink는 Gui Block Programming Language로 시스템 구성에 있어서는 매우 탁월한 기능을 갖추고 있습니다. 특히, DSP Filter 구성에서는 따라올 대안이 없다는 것에는 대부분의 엔지니어가 동의할 것입니다.

따라서 본 교재를 전부 보기 위해서는 아래와 같은 환경을 구성하고 가는 것을 추천합니다.

1. Ubuntu 14.04 or Ubuntu 16.04
2. ROS Jade or Kinetic
3. Python 2.7 and Development Environment
4. Matlab 2016a with Simulink Robotics Communication Block
5. Arduino or Mbed

이제 Arduino를 제외한 (아두이노는 인간적으로 너무 쉬워요..) 해당 환경들을 설치하는 방법에 대해서 차근차근 설명해 보겠습니다.

## 1.3 Ubuntu 설치

이번 섹션에서는 Ubuntu를 설치하는 방법에 대해 이야기해 보겠습니다. Ubuntu는 앞서 말했듯이 리눅스 시스템으로, ROS를 올리기 좋은 환경을 제공해줍니다. 하드웨어를 연계해서 개발을 할 예정이니 Virtual Machine이 아닌 Native로 설치하는 것을 권장합니다. ARM 보드 위에 설치한 Ubuntu를 가지고 교재를 진행하는 것은 권장하지 않습니다. 왜냐하면 ARM에는 지원되지

않는 ROS 패키지가 많으며, 이 교재는 64비트 x86 CPU (일반적인 Intel, AMD CPU) 기준으로 쓰여졌기 때문입니다.

먼저 Ubuntu ISO Image를 다운로드 받습니다. 우분투 공식 배포판을 활용하는 것을 추천합니다.

#### Link 1.3.1 — <http://www.ubuntu.com/>. : Ubuntu 공식 홈페이지

Ubuntu 공식 홈페이지에 접속하셔서 Desktop 버전을 다운로드 받으세요. LTS(Long Term Support) 버전을 사용하는 것을 추천합니다.

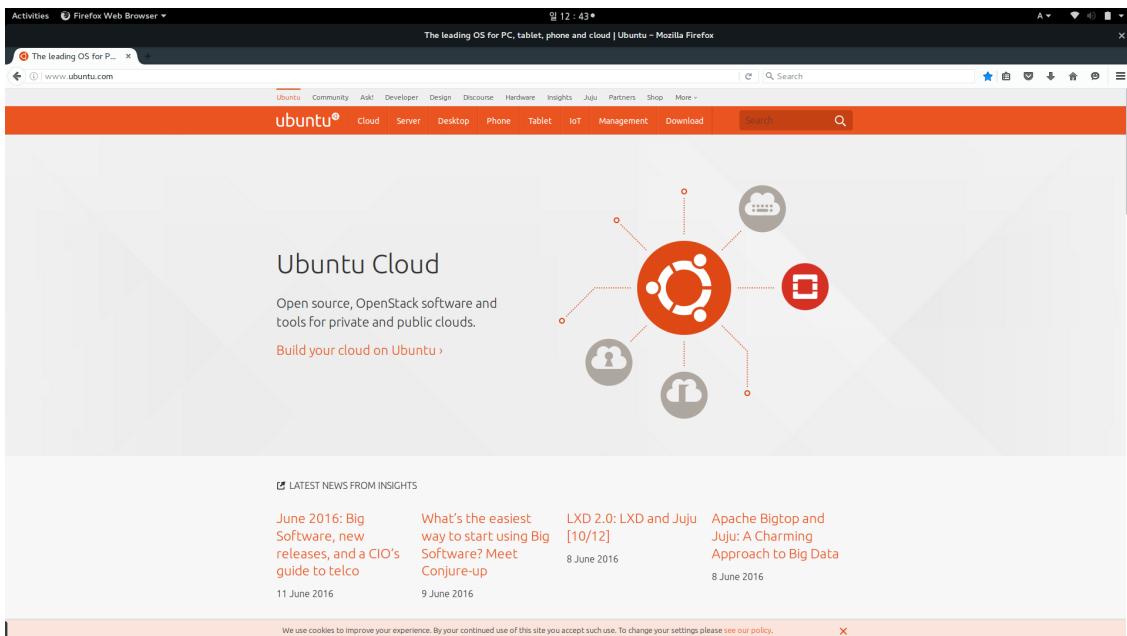


Figure 1.1: Ubuntu 공식 페이지

다운로드 받은 ISO 파일은 이제 usb에 넣어 부팅 usb를 만들고 설치해야 합니다. 해당 과정은 잘 정리되어 있는 블로그 링크가 있기에 아래에 첨부합니다.

#### Link 1.3.2 — <http://sergeswin.com/1178>. : Ubuntu 부팅 usb 제작과정

해당 블로그의 글을 보고 부팅 usb를 만드셨다면, 이제 컴퓨터 파티션에 공간을 30GB 정도 남긴 뒤(단순히 free space가 아니라 disk 관리로 들어가셔서 가용 공간을 확보하셔야 합니다.) usb로 부팅해서 해당 공간에 ubuntu를 설치하세요. 공식 홈페이지에서 튜토리얼을 제공하니 참고하셔서 설치하면 어렵지 않게 설치가 가능합니다.

#### Link 1.3.3 — <http://www.ubuntu.com/download/desktop/install-ubuntu-desktop>. : Ubuntu 설치 과정

사실 Ubuntu는 리눅스 중에서도 매우 사용자 친화적인 배포판이기 때문에 설치순서 자체가 어렵다기보다는, 설치 과정 중에 생기는 문제가 해결하기 까다롭습니다. 그러나 개인별 증상이 매우 다르고, 그 때마다 해결책도 다르기 때문에 모든 것을 매뉴얼로 만들기는 사실상 불가능 합니다. Ubuntu 설치 중에 문제가 생긴다면 아래 사이트를 참고해서 해결하도록 노력해보세요. Linux 환경에서 작업을 하기 위해서는 해당 배포판의 포럼을 계속해서 들락 날락 하는 노력이 필요하답니다.

#### Link 1.3.4 — <http://askubuntu.com/>. : AskUbuntu Forum

무사히 Ubuntu를 설치하셨다면 축하합니다! 다음 파트에서는 드디어 ROS를 설치해 보겠습니다.

## 1.4 ROS 설치

ROS는 2016.06 현재 ROS Kinetic Version 까지 나와 있습니다. ROS 또한 Ubuntu와 마찬가지로 여러 가지 배포판이 나와있는데, 일반적으로 비슷한 시기에 나온 Ubuntu와 ROS 배포판이 호환성이 좋습니다. Ubuntu 14.04를 설치해 온 분은 ROS Jade, Ubuntu 16.04를 설치해 온 분은 ROS Kinetic을 권장합니다. 혹시나 개발기간이 촉박해서 기존 패키지를 조합하는 방식의 개발을 원하는 분은 구버전인 ROS Indigo를 사용하셔도 좋습니다. ROS는 최신버전으로 올수록 apt-get을 이용해 간단히 설치할 수 있는 패키지가 줄어들기 때문에, 최신버전과 패키지의 숫자 사이에서 잘 저울질 해서 알맞은 버전을 선택하시길 바랍니다.



Figure 1.2: ROS Versions

ROS 설치를 위해서는 Ubuntu에서 커멘드 라인을 실행해야 합니다. 일반적인 환경을 구축했다면 Ctrl + Alt + T 를 누르면 아래와 같이 커멘드라인이 실행됩니다.

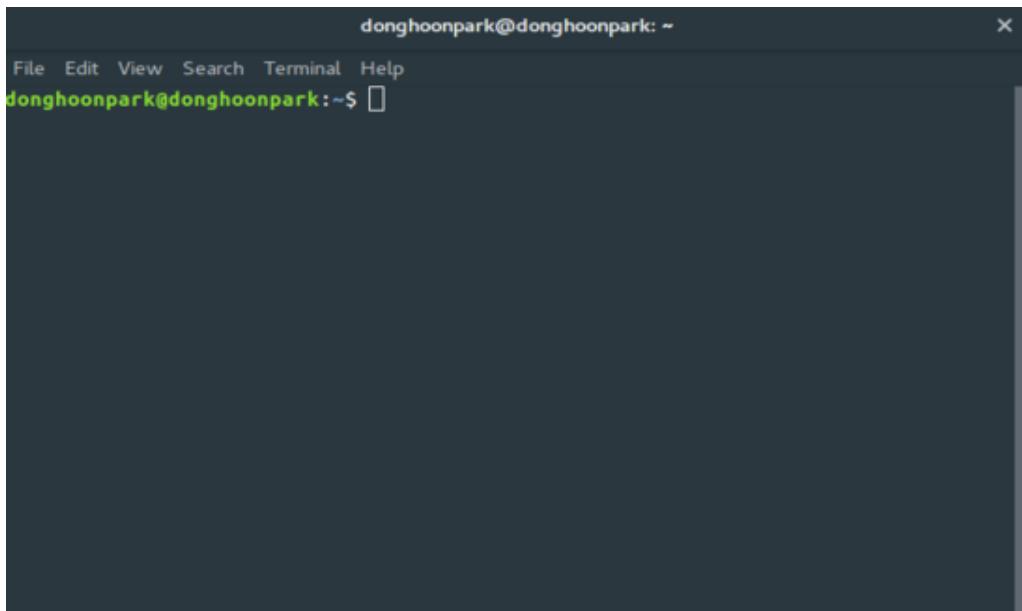


Figure 1.3: Terminal 실행 창

이제 이 창 위에 설치 명령어를 입력하여 ROS를 설치하게 됩니다. ROS 설치는 대부분 비슷한 과정으로 진행되지만, 버전별로 약간씩 상이하기 때문에 자신이 설치할 버전이 무엇인지 잘 알고 진행하시기 바랍니다. 혹시나 ARM Linux Board(Odroid/ Raspberry pi/ Beaglebone Black) 가 최종 Target System인 경우, 최신버전인 ROS Kinetic을 설치하지 않는 것을 권장합니다.

ROS의 버전별 설치 방법은 아래와 같습니다.

**Link 1.4.1 — <http://wiki.ros.org/indigo/Installation/Ubuntu>.** : ROS Indigo 설치과정

**Link 1.4.2 — <http://wiki.ros.org/jade/Installation/Ubuntu>.** : ROS Jade 설치과정

**Link 1.4.3 — <http://wiki.ros.org/kinetic/Installation/Ubuntu>.** : ROS Kinetic 설치과정

설치 과정에 따라 명령어를 입력하면 ROS가 설치됩니다. desktop-full package로 설치하는 것을 권장드리며, 최대 약 1시간까지도 걸리는 작업이므로 느긋하게 기다려 주세요. 1.1 - 1.8 까지 모든 명령어를 순서대로 입력하셔야 합니다.

간단하게 설치하고 싶으시다 혹은 리눅스 초보라 아무것도 모르겠다 하시는 분은 아래 명령어를 입력해서 ROS Indigo를 간단하게 설치하는 방법도 있습니다만 권장하지 않습니다. 해당 링크는 오픈소스 로보틱스 카페 표윤석 연구원님께서 만든 저장소와 스크립트입니다.

**Link 1.4.4 — <http://cafe.naver.com/openrt/13991>.** : wget [https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros\\_install.sh](https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh) && chmod 755 ./ros\_install.sh && ./ros\_install.sh catkin\_ws indigo

ROS를 모두 설치했다면 제대로 설치되었는지 roscore 프로그램을 실행시켜 봅시다. 터미널을 열고, roscore 명령어를 아래와 같이 입력하고 실행해 보세요. 정상적으로 작동한다면 ROS 설치에 성공한 것입니다. 아래 예시는 ROS Kinetic을 설치한 예시입니다.

```

roscore http://donghoonpark:11311/
File Edit View Search Terminal Help
donghoonpark@donghoonpark:~$ roscore
... logging to /home/donghoonpark/.ros/log/f81e68cc-3096-11e6-b92e-
3052cb70141d/roslaunch-donghoonpark-17659.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://donghoonpark:46744/
ros_comm version 1.12.0

SUMMARY
=====

PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.0

NODES

auto-starting new master
process[master]: started with pid [17672]
ROS_MASTER_URI=http://donghoonpark:11311/

setting /run_id to f81e68cc-3096-11e6-b92e-3052cb70141d
process[rosout-1]: started with pid [17685]
started core service [/rosout]

```

Figure 1.4: roscore 실행 창

## 1.5 Python Development Environment

이제 프로그래밍 언어인 파이썬을 사용하기 위한 개발환경을 갖추어 봅시다. 프로그래밍 언어 개발환경은 크게 세 가지로 구성됩니다. Editor, Compiler, Debugger. 여기에서는 IDE 한 가지를 추천하고, 혹시나 상황이 여의치 않을 경우 사용할 수 있는 오픈소스 개발환경 구축법을 소개하도록 하겠습니다.

### 1.5.1 Pycharm



Figure 1.5: pycharm

Pycharm은 JetBrains에서 만든 Python 전용 IDE입니다. 파이썬 패키지만 설치된 상태라면자동으로 모든 설정을 알아서 수행하며, 설치만 하면 바로 사용할 수 있는 상태가 됩니다. 아쉬운 점이라면 다른 환경에 비해 비교적 무겁다는 점과, 무료가 아니라는 점입니다. 그러나 학생인증을 하면 무료로 받을 수 있으므로 학생이라면 해당 IDE를 사용하는 것을 권장합니다. 개인적인 감상으로는 여타 개발 프로그램에 비해 비싼 편이 아니므로 구매하는 것도 상당히 좋은 선택일 것입니다.(필자는 위 회사와 아무런 관련이 없습니다.) Pycharm은 아래 링크로 가면 다운로드 받을 수 있습니다.

Link 1.5.1 — <https://www.jetbrains.com/pycharm/>. : Pycharm 공식 홈페이지

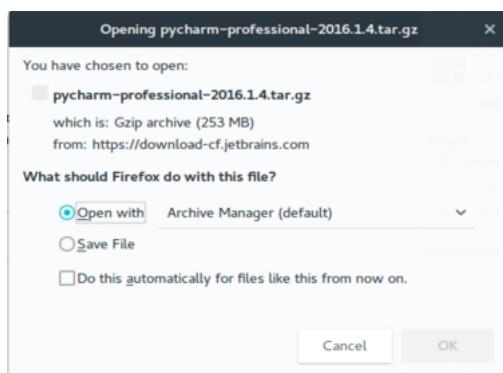


Figure 1.6: pycharm 다운로드

Pycharm을 다운로드 받고, 설치하는 방법에 대해 이야기하겠습니다. tar.gz 확장자를 가진 파일은 tar 명령어 혹은 파일 압축풀기 프로그램을 이용해 압축을 해제할 수 있습니다. 압축을 푸는 법은 검색해서 찾아보시고, 압축이 풀린 상태에서 아래 그림과 같이 커맨드라인에서 해당 경로로 이동 후 실행파일을 실행(.) 시켜주면 됩니다.

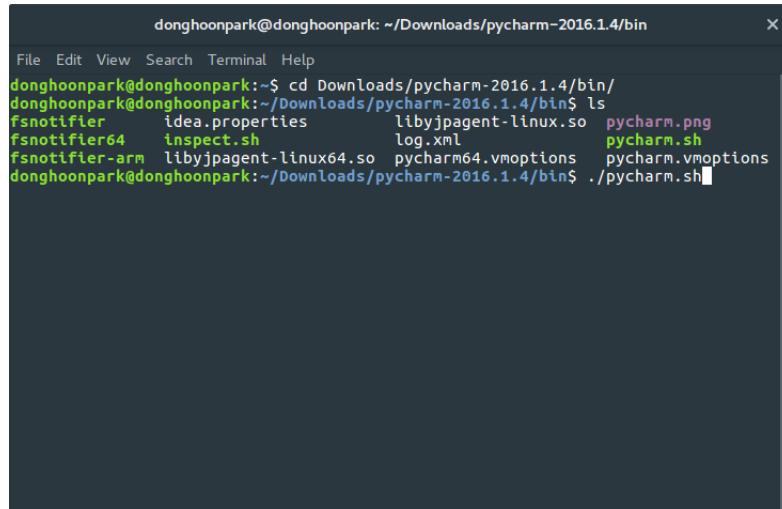


Figure 1.7: pycharm 설치



Figure 1.8: pycharm 실행화면

이제 new project를 누르고 프로그래밍을 시작하면 됩니다. Pycharm을 설치했다면 이후 단계를 생략해도 좋습니다.

### 1.5.2 Python Package Install

IDE를 사용하지 않는 경우에는 IDE에 해당하는 환경을 각각 구성해줘야 합니다. 각각의 환경이 또 파이선을 사용하는 경우가 많아 파이선 패키지 관리자를 먼저 설치하고 시작하도록 하겠습니다. 아래의 명령어 한 줄이면 Python Package 관리자인 pip이 설치됩니다.

```
# sudo apt-get install python-pip
```

이제 pip에서 아래의 패키지들을 명령줄 도구로 설치해봅시다.

```
# sudo pip install jedi
```

```
# sudo pip install pylama
```

jedi는 파이선 언어 자동완성 패키지이고, pylama는 파이선 언어 코드 형식 및 에러체크 패키지입니다.

### 1.5.3 Atom Editor Install

이제 에디터를 설치하고 위에서 설치한 파일 패키지들을 연결해봅시다.

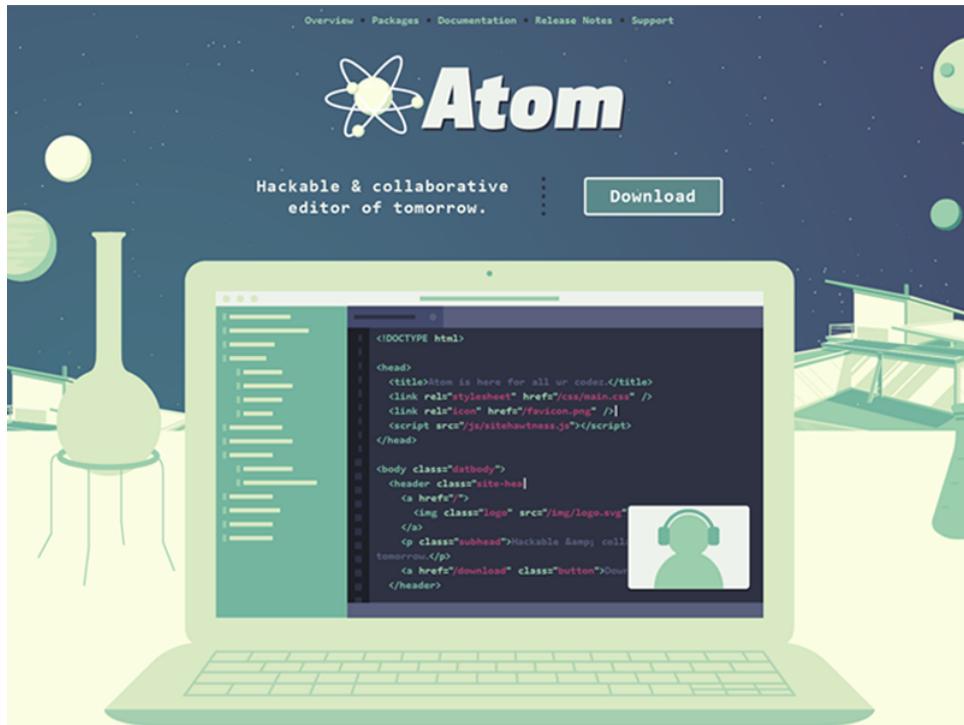


Figure 1.9: Atom Editor

Atom은 Github에서 만든 에디터로, 상당히 사용자 친화적인 인터페이스를 제공합니다. 물론 속도가 조금 느리다는 평이 있지만, 속도를 중요시하는 분들은 Vim이나 Emacs를 알아서 잘 사용하실 것으로 믿기에 에디터는 Atom 에디터만 다루고 넘어가도록 하겠습니다. 아래 링크로 들어가서 Atom editor 패키지를 받습니다.(.deb)

**Link 1.5.2 — [https://atom.io/.](https://atom.io/)** : Atom Editor 공식 홈페이지

.deb 패키지는 실행시키면 소프트웨어 센터에 연결되며 설치가 될 것입니다. 이제 위 파일 패키지들에 연결할 에디터 패키지를 설치해봅시다. atom을 실행한 후 Ctrl+, 를 누르면 세팅 메뉴가 열리고, 여기서 새로운 패키지들을 설치할 수 있습니다.

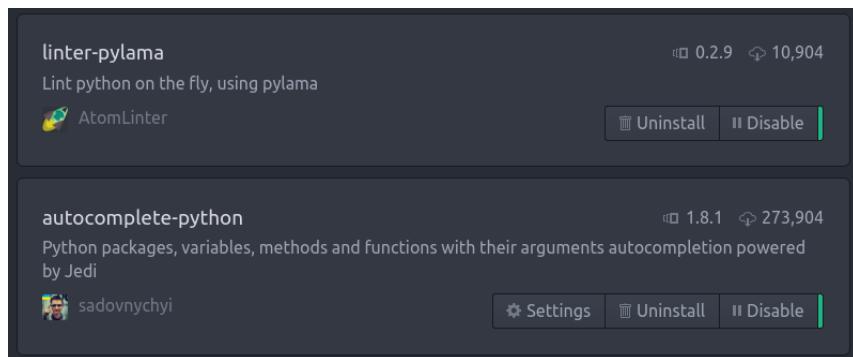


Figure 1.10: Atom Install Packages

위에 있는 두 패키지를 설치하면 됩니다. 이제 ROS를 손쉽게 사용할 수 있도록 autocomplete-

python package를 설정하겠습니다. 그 전에 ROS용 파일 패키지 설치 경로를 알아야 설정이 가능하므로 아래와 같이 Python 쉘을 실행시켜 ROS 모듈을 위치를 알아냅니다.

```
donghoonpark@donghoonpark:~$ python
Python 2.7.11+ (default, Apr 17 2016, 14:00:29)
[GCC 5.3.1 20160413] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import rospy
<module 'rospy' from '/opt/ros/kinetic/lib/python2.7/dist-packages/rospy/__init__.pyc'>
>>> 
```

Figure 1.11: ROS 모듈 위치 찾는법

아래 창에서 보면 rospy 모듈은 /opt/ros/kinetic/lib/python2.7/dist-packages에 있는 것을 알 수 있습니다. 해당 경로를 복사해서 jedi 패키지 설정의 Extra Paths for Packages에 넣어줍니다. 그리고 에디터를 재시작한 후 python 파일을 하나 만듭니다. 정상적으로 설치된 경우 아래와 같이 autocomplete과 linter가 잘 작동합니다.

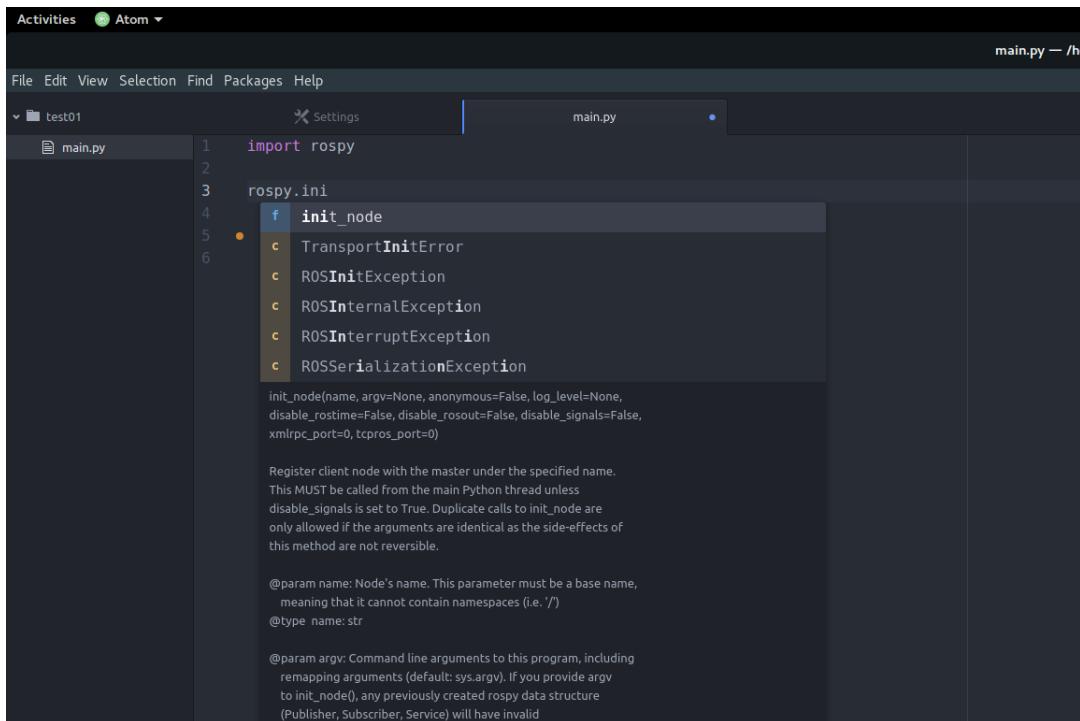


Figure 1.12: Atom 정상 실행 상황

파이선 개발용 패키지들은 이것 이외에도 유용한 것이 많으니 개인의 취향에 따라 선택해서 더 많은 패키지를 이용해 생산성을 높여보시기 바랍니다.

## 1.6 Matlab Install

마지막으로 Matlab을 설치합니다. (학교 혹은 기관 라이선스가 있는 경우에)

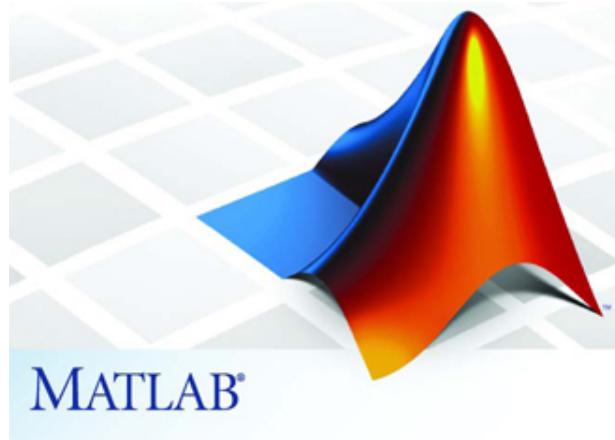


Figure 1.13: Matlab

Matlab 설치 과정은 Mathworks 사에 매우 자세하게 설명되어있기 때문에 생략하도록 하겠습니다. ROS 지원 블록을 설치하는 것을 잊지 마세요. 아래 링크를 이용해서 접속하시면 Matlab 을 받을 수 있습니다.

**Link 1.6.1 — [http://kr.mathworks.com/downloads/web\\_downloads/?s\\_tid=srchtitle](http://kr.mathworks.com/downloads/web_downloads/?s_tid=srchtitle) . :**  
Matlab 다운로드 페이지

설치가 완료되면 아래 한 줄 커멘드 라인으로 Matlab 우분투 지원 패키지를 설치하세요. Matlab을 시스템에 알맞게 Integration 시켜주고, 명령줄 상에서 Matlab을 실행할 수 있게 해주는 도구입니다.

```
#sudo apt-get install matlab-support
```

아래와 같이 커맨드라인에서 Matlab을 실행해보는 것으로 Chapter1을 마치겠습니다.

 A screenshot of a terminal window titled "donghoonpark@donghoonpark: ~". The window contains the following text:
 

```
donghoonpark@donghoonpark:~$ matlab -nodesktop
  < M A T L A B (R) >
  Copyright 1984-2016 The MathWorks, Inc.
  R2016a (9.0.0.341360) 64-bit (glnxa64)
  February 11, 2016

  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

  Academic License

>> 1+1
ans =
  2
>> █
```

Figure 1.14: Matlab Command Line Test



## 2. ROS Basics

### 2.1 ROS Topic

ROS에서 Topic은 비동기로 작동하는 시스템 변수로 볼 수 있습니다. 이 변수를 발행하는 곳(일반적으로 센서)을 Publisher라고 부르고, 이 변수를 받아들이는 곳(일반적으로 Actuator 등)을 Subscriber라 부르죠. 두 가지 기능을 동시에 하는 경우(Filter)도 존재할 수 있습니다. ROS 내용에 대한 보다 개념적인 자세한 설명은 아래 링크의 표윤석 수석연구원님의 책 53p-67p를 참고하기 바랍니다.

**Link 2.1.1 — [https://github.com/robotpilot/rosbook\\_kr/blob/master/pdf/ROS\\_Book\\_KR.pdf](https://github.com/robotpilot/rosbook_kr/blob/master/pdf/ROS_Book_KR.pdf).**  
: 로봇 프로그래밍 ROS로 시작하자! - 표윤석 수석연구원님 저

간단한 Line Follower Robot을 ROS로 구성한다고 했을 때 아래 그림과 같이 시스템을 구성해볼 수 있습니다.

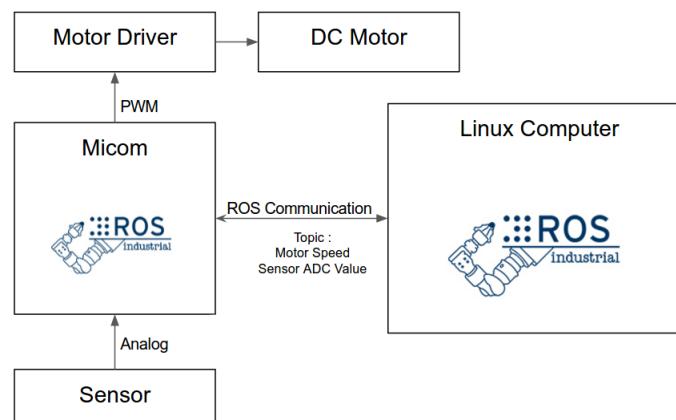


Figure 2.1: ROS System example

센서 데이터는 Micom(Arduino, Mbed, STM32 등등)에서 일차적으로 ADC로 읽어오게 됨

니다. 그리고 ADC로 읽은 데이터를 ROS에 Publish 하면 ROS 입장에서는 Data가 일정 주기로 계속 업데이트 되는 것으로 보이죠. 모터를 제어할 때도 마찬가지입니다. ROS에서 모터 출력에 대한 변수를 하나 Publish하고, 실제로 모터를 제어하는 Micom이 해당 데이터를 Subscribe하면서 업데이트하면 ROS에서 모터제어를 할 수 있습니다.

## 2.2 Topic Publisher

### 2.2.1 Rqt를 이용한 Topic Publisher

시뮬레이션 프로그램을 매우 간단하게 구동해 보면서 Topic을 Publish 한다는 것이 어떤 의미인지 감을 잡아봅시다. ROS는 시뮬레이션 모델에 대한 정보를 Topic으로 불러올 수 있습니다. 해당 Topic들은 시뮬레이션 모델의 물리 벡터를 가지고 있습니다.(Torque, Force 등) 이것들을 Control 하는 방법으로 시뮬레이션을 구동시킬 수 있죠. Topic이 어떤 형태로 나오는지 알아보기 위해서 Gazebo Robot Simulator를 구동해봅시다. 아래의 명령어를 입력하면 Gazebo를 ros에 연결된 상태로 구동할 수 있습니다.

```
#rosrun gazebo_ros gazebo
```

정상적으로 실행되면 아래와 같은 창을 보실 수 있습니다. 여기에서 컨트롤 할 대상인 Sphere를 하나 불러와 봅시다.

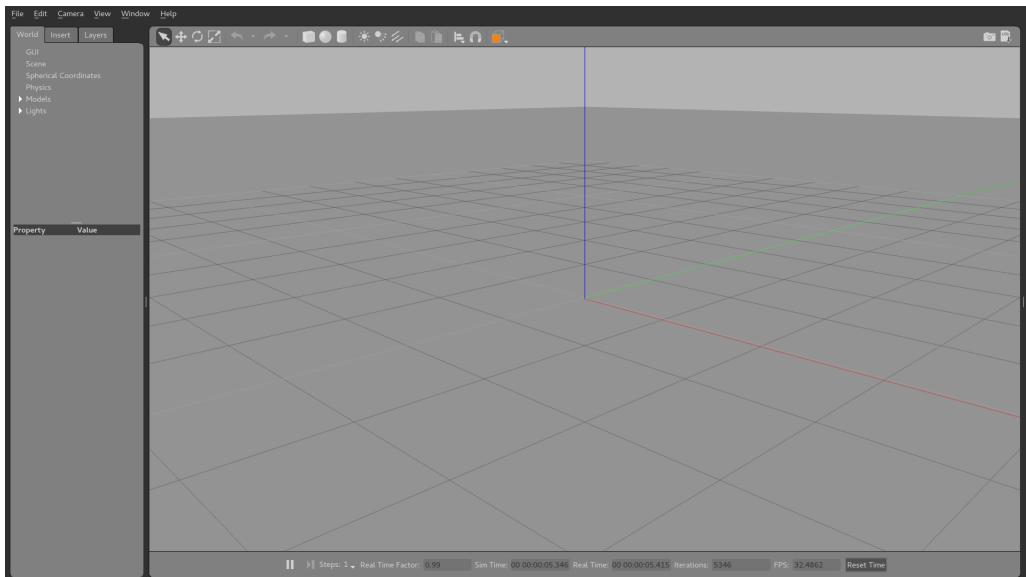


Figure 2.2: Gazebo 기본 화면

gazebo가 정상적으로 구동된 뒤에는 다른 터미널에서 rqt를 구동해봅니다. rqt에서는 기본 topic publisher를 제공합니다. Plugins -> Topics -> Message Publisher를 열어보세요.

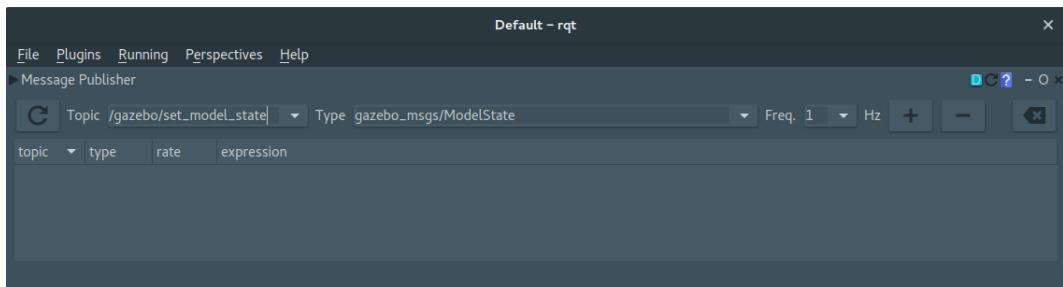


Figure 2.3: rqt topic message publisher

```
#rqt
```

여기에서 `/gazebo/set_model_state`라는 topic을 불러와 봅시다. 해당 topic은 Gazebo에서 Subscribe하고 있는 topic으로 해당 내용이 바뀌면 Gazebo는 프로그램에서 해당 모델의 상태를 바꾸어줍니다. 해당 topic을 publish하기 위해서 오른쪽의 버튼을 눌러 publisher를 추가합니다. 그리고 publish할 message를 아래와 같이 설정합니다. 아래와 같이 설정했을 때 i는 rate가 하루에 1번마다 원을 원점을 따라서 한 바퀴씩 움직이게 될 것입니다.

rqt는 topic 제어 이외에도 여러 기능을 가지고 있기 때문에 간단히 훑어보시는 것도 좋을 것입니다. 특히, Matlab을 사용하지 않는 경우 rqt의 plot 기능은 매우 유용하게 사용할 수 있습니다.

### 2.2.2 Python을 이용한 Topic



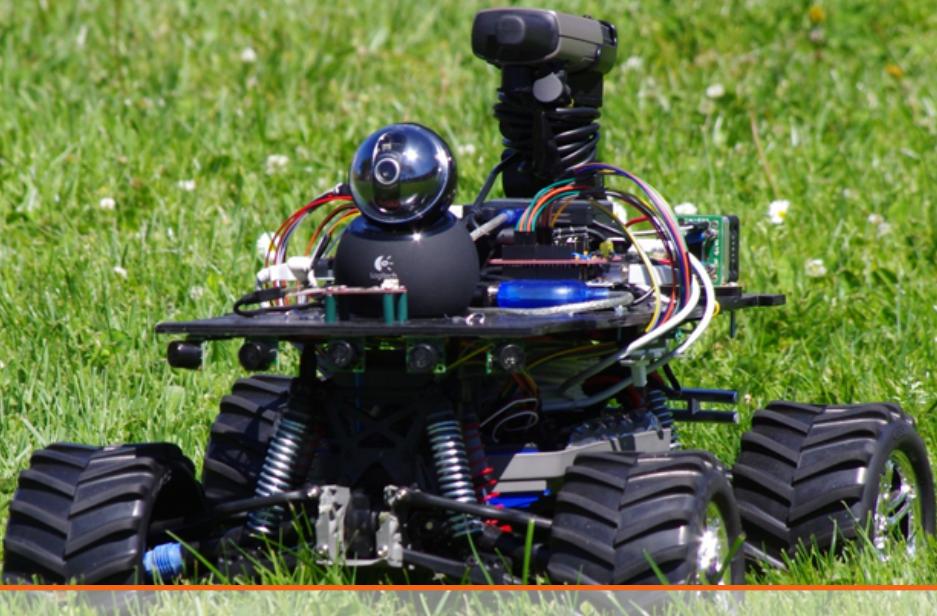


## Bibliography

**Books**

**Articles**





## Index

Atom Editor Install, 14

Matlab Install, 15

Paragraphs of Text, 7

Pycharm, 12

Python Development Environment, 12

Python Package Install, 13

ROS Topic, 17

ROS 설치, 10

ROS를 사용하기 위해서는?, 8