

# MRP-NRLAB02 Manual v1.3



## 문서 개정 이력

[illegible]

# 목차

1. 제품 소개 .....	7
2. 제품 확인 .....	8
2.1. 제품 구성 .....	8
2.2. 주의 사항 .....	9
3. 제품 설명 .....	10
3.1. MRP-NRLAB02 사양 .....	10
3.2. MRP-NRLAB02 외형 치수 .....	11
3.3. 내장 센서 .....	13
가. IR-PSD 센서 .....	13
나. AHRS 센서 .....	13
3.4. 제어 패널 .....	15
3.5. 비상스위치 및 제어기 연결 .....	16
가. 비상스위치 .....	16
나. 사용자 PC 연결 .....	16
4. 시작하기 .....	17
4.1. 기본 조이스틱 구동 .....	17
4.2. 배터리 충전 .....	19
5. 사용자 설정 .....	20
5.1. 로봇 플랫폼과 연결 .....	20
5.2. 로봇 플랫폼 IP 어드레스 설정 .....	21
6. 제어 라이브러리 .....	22
6.1. 사용자 제어 데이터 구성(사용자 송신) .....	23
6.2. 로봇 출력 데이터 구성(사용자 수신) .....	24
6.3. 제어 라이브러리 구성 .....	26
가. 제어 라이브러리 .....	26



나. 제어 라이브러리 함수 리스트 .....	26
7. 예제 프로그램 .....	28
7.1. Ubuntu .....	28
7.2. ROS Kinetic .....	29
가. MRP-NRLAB02 패키지 설치 .....	29
나. nrlab02 노드 실행 .....	30
다. Parameters .....	31
라. TOPICS .....	32
마. RVIZ .....	33
바. Gazebo .....	34
7.3. Windows .....	35
가. 예제 프로그램 실행 .....	35
나. 새 프로젝트 생성 시 .....	37
8. APPENDIX A – IP Configuration .....	39
8.1. Ubuntu 의 경우 .....	39
8.2. 변경된 로봇 IP 어드레스를 잃어버린 경우 .....	40
가. Windows 경우 .....	40
나. 리눅스의 경우 .....	40
다. IP 어드레스를 찾을 수 없는 경우 .....	41
9. APPENDIX B Library Function List .....	43
9.1. 동작 모드 정의 .....	43
9.2. 함수 동작 결과 정의 .....	43
9.3. 중요 파라미터 .....	44
9.4. 로봇 연결 제어 .....	44
9.5. 모터 제어 및 상태 읽기 .....	47
9.6. 로봇 제어 및 상태 읽기 .....	50
9.7. 센서 읽기 .....	52



## 그림 목차

그림 1. MRP-NRLAB02 이동 로봇.....	7
그림 2 IR-PSD 센서 배치 각도.....	12
그림 3 AHRS 좌표계.....	13
그림 4 AHRS 센서 장착 위치.....	14
그림 5 측면 제어 패널.....	15
그림 6 비상스위치 및 사용자 PC 연결 포트.....	16
그림 7 기본 조이스틱 구동.....	18
그림 8 로봇 플랫폼과 사용자 PC 연결.....	20
그림 9 제어 라이브러리 구조.....	22
그림 10 ROS Dynamic reconfigure – rqt_reconfigure.....	31
그림 11 Rviz RobotModel 디스플레이.....	33
그림 12 rviz & gazebo 시뮬레이션.....	34



## 표 목차

표 1. MRP-NRLAB02 내용물 구성 .....	8
표 2 MRP-NRLAB02 사양 .....	10
표 3 제어 라이브러리 사용환경과 컴파일 환경 .....	22
표 4 사용자 제어 데이터 구성 .....	23
표 5 사용자 수신 데이터 구성 .....	24
표 6 제어 라이브러리 구성 .....	26
표 7 제어 라이브러리 함수 리스트 .....	26



## 1. 제품 소개

- MRP-NRLAB02는 실시간 모션제어 네트워크인 이더넷을 기반으로 개발된 실내용 이동 로봇 플랫폼입니다.
- 주요 특징
  - 고성능 산업용 네트워크인 EtherCAT 적용
  - 2휠 기반의 차동 구동 메커니즘 적용
  - 14bit 엔코더 내장으로 보다 정밀한 위치 제어 가능
  - 제어 Controller가 내장되어 있어 사용자가 쉽게 사용이 가능
  - Linux와 Windows 개발 환경을 모두 제공하고 있어 다양한 방식의 어플리케이션 구성 및 개발이 가능



그림 1. MRP-NRLAB02 이동 로봇

## 2. 제품 확인

### 2.1. 제품 구성

제품의 내용물의 구성은 다음과 같습니다.

표 1. MRP-NRLAB02 내용물 구성

구분	항목	내용	수량
제품-로봇	MRP-NRLAB02	MRP-NRLAB02 이동 로봇 플랫폼 - BLDC Motor Driver x2 - BLDC Motor + Encoder - Sensor Module x1 - AHRS Sensor x1 - PSD Sensor x1	1EA
제품 CD	사용자 매뉴얼	사용자 설명서 (PDF)	1EA
	제어 Library v1.0	MRP-NRLAB02 제어용 라이브러리 및 샘플 프로그램	
액세서리	조이스틱	무선 조이스틱	1EA
	충전기	29.4V 5A 충전기	1EA
	케이블	LAN Cable	1EA





## 2.2. 주의 사항

제품을 사용하기 앞서 가급적이면 다음의 사항들을 유념하도록 합니다.

- 사용하기 앞서 주변 환경을 잘 정리한 후 사용하기 바랍니다.
- 제품을 너무 무리하게 다루지 마십시오.
- 전원이 켜진 상태에서 물을 쏟는 다거나, 전도성이 있는 물체로 장비를 건드리는 행위는 자체해 주십시오.
- 전원이 켜진 상태에서는 충전을 자제해 주십시오.
- 모든 케이블은 전원을 넣기 이전에 연결해 주시고, 사용 후에는 전원을 끈 상태에서 케이블들을 뽑아 주십시오.
- 케이블은 손잡이 부분을 잡고 천천히 뽑아 주십시오. 케이블을 잡고 무리하게 잡아 당길 경우 케이블에 손상을 주어 사용할 수 없게 될 수 있습니다.
- 사용하다가 갑자기 장비가 동작하지 않을 경우 케이블과 설정, 하드웨어 및 소프트웨어 설계에 이상이 없는지 다시 한 번 꼼꼼히 살펴보십시오.
- 어떤 특별한 목적이 있어 제품을 다른 장비 혹은 보드와 연결하여 사용할 시에는 문제가 없는지 다시 한 번 검토하고 실행해 주십시오.
- 장비를 사용한 후에는 전원이 끄고 케이블을 뽑아 잘 정리하여 케이블 보관함에 깔끔한 상태로 넣어 잘 보관해 주십시오.

### 3. 제품 설명

#### 3.1. MRP-NRLAB02 사양

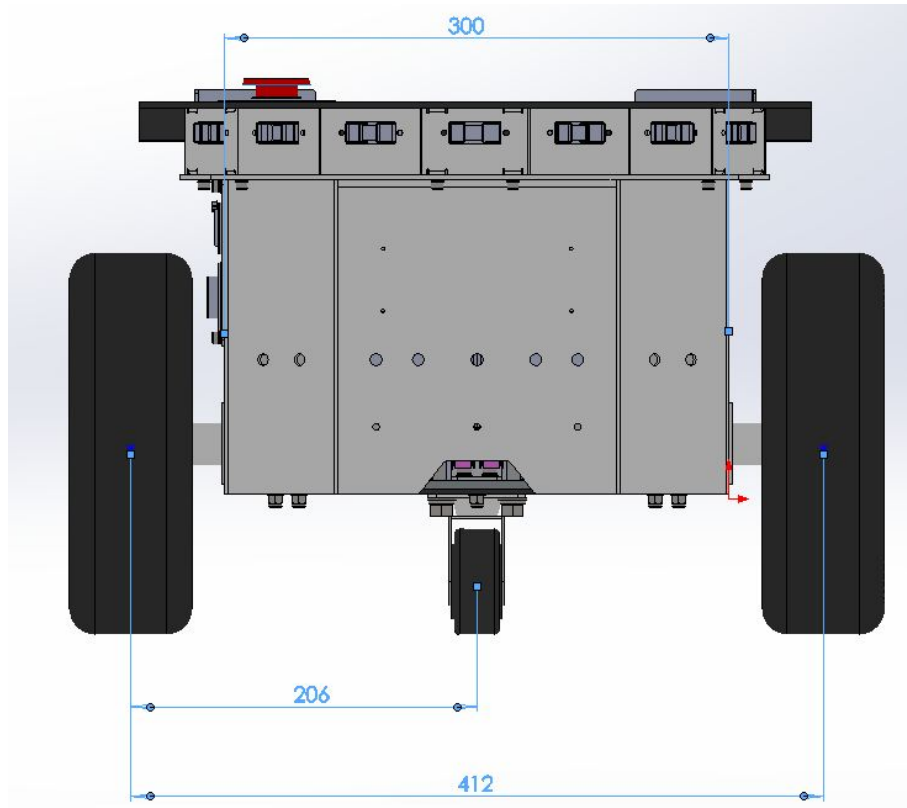
표 2 MRP-NRLAB02 사양

Physical	
Dimension	(L)610mm x (W)475mm x (H)320mm
Weight	27kg
Payload	Over 40Kg
Mobility	
Drive	2-wheel drive + real balancing caster
Wheel composition	Pneumatic rubber tire , hard caster
Steering	Differential
Gear ratio	25:1 (Planetary gear)
Translate speed	Max. 1.18 m/s
Power	
Battery	24V 12Ah Sealed, lead-acid
Continuous run time	2~5 hours
Extension Power	5V, 24V Extension power
Components	
Motor	24V, 4000rpm BLDC motor(x2)
Encoder	3.6V~5.5V, 2048PPR, 7500rpm
Sensor	Front IR-PSD Sensor (x7), Range : 20 ~150cm AHRS Sensor : Quaternion, Accelerometer, Magnetometer, Gyroscope
Joystick	Cordless Precision Joystick
Safety & Display	Emergency – Switch
Controller	Odroid-c1+ : Ubuntu 16.04 server
Network	
Network	EtherCAT
Transmission rate	100Mbps, 실시간 IEEE1588 동기화, 100Base TX
Cycle time	10ms

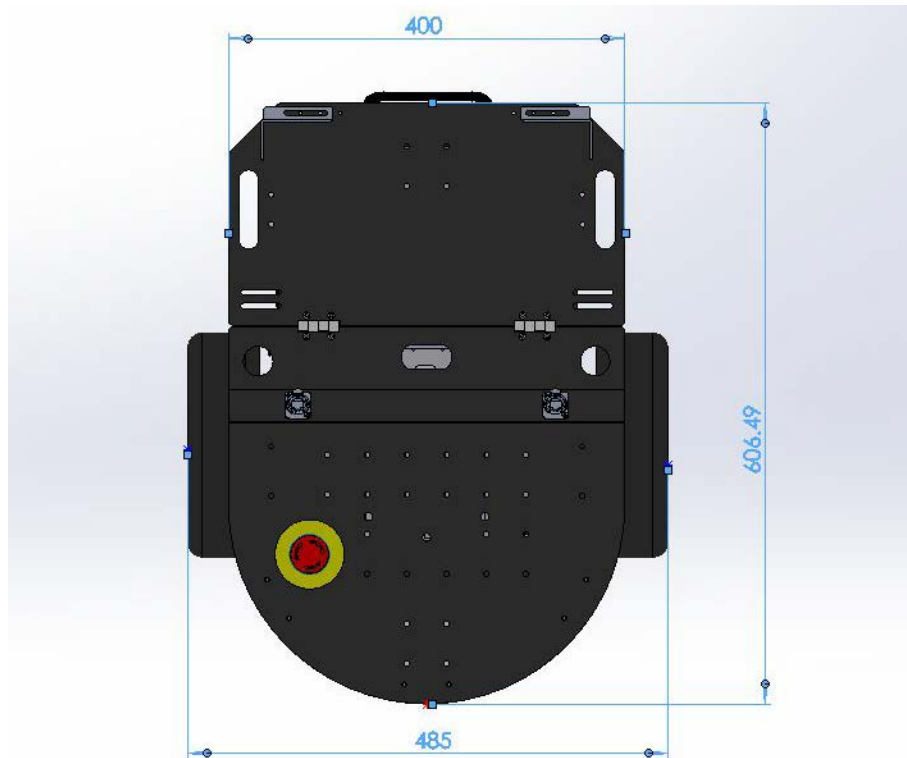


### 3.2. MRP-NRLAB02 외형 치수

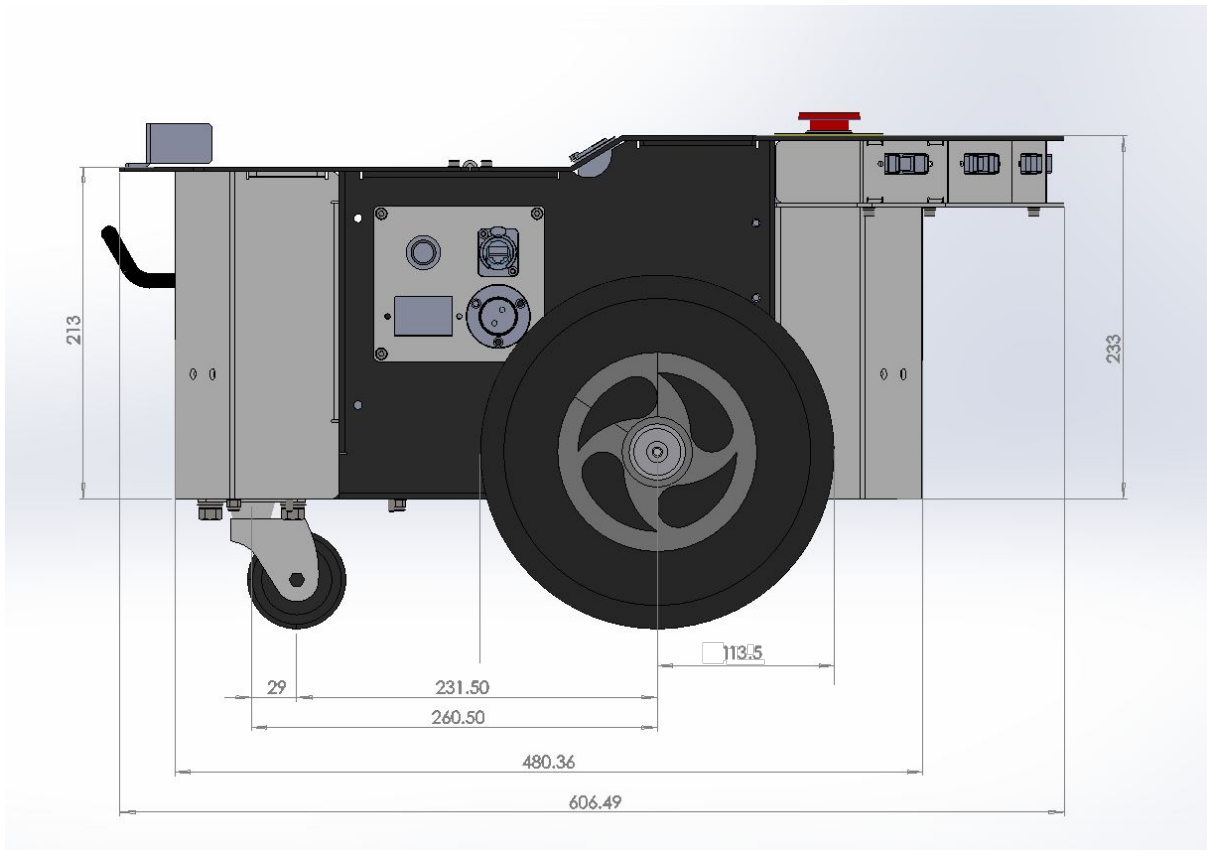
#### ■ 정면 뷰



#### ■ 탑 뷰



■ 측면 뷰



■ PSD 센서 배치 각도: 20도

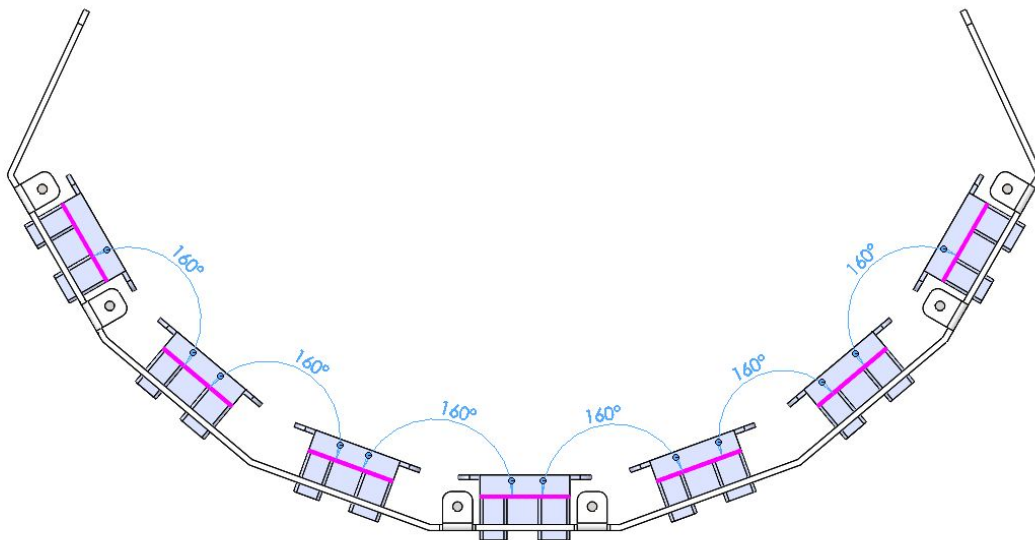


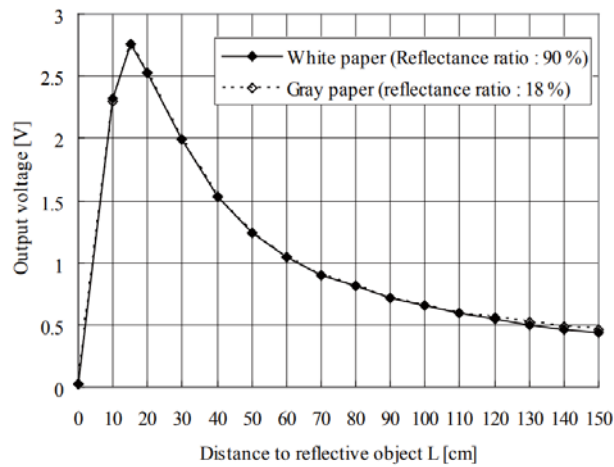
그림 2 IR-PSD 센서 배치 각도

### 3.3. 내장 센서

MRP-NRLAB02에 내장된 센서는 EtherCAT 모듈에 연결되어 있으며, 사용자는 제어 라이브러리를 통해 데이터를 수신할 수 있습니다.

#### 가. IR-PSD 센서

- 측정 거리: 20 ~ 150cm



#### 나. AHRS 센서

- 데이터 출력 주기: 100Hz
- 자세 출력: Quaternion
- 센서 출력: 3축 가속도(g), 3축 각속도(dps), 3축 지자기

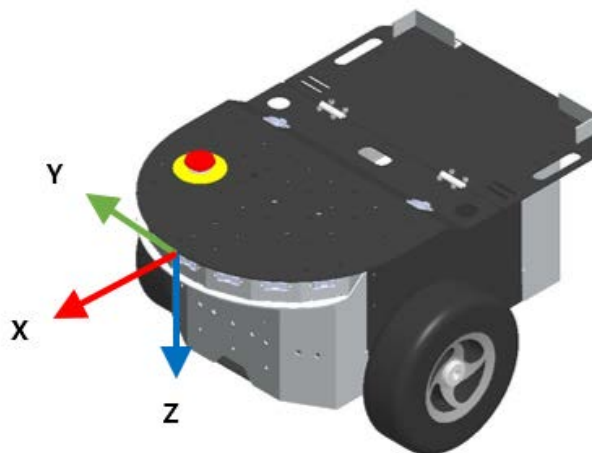


그림 3 AHRS 좌표계

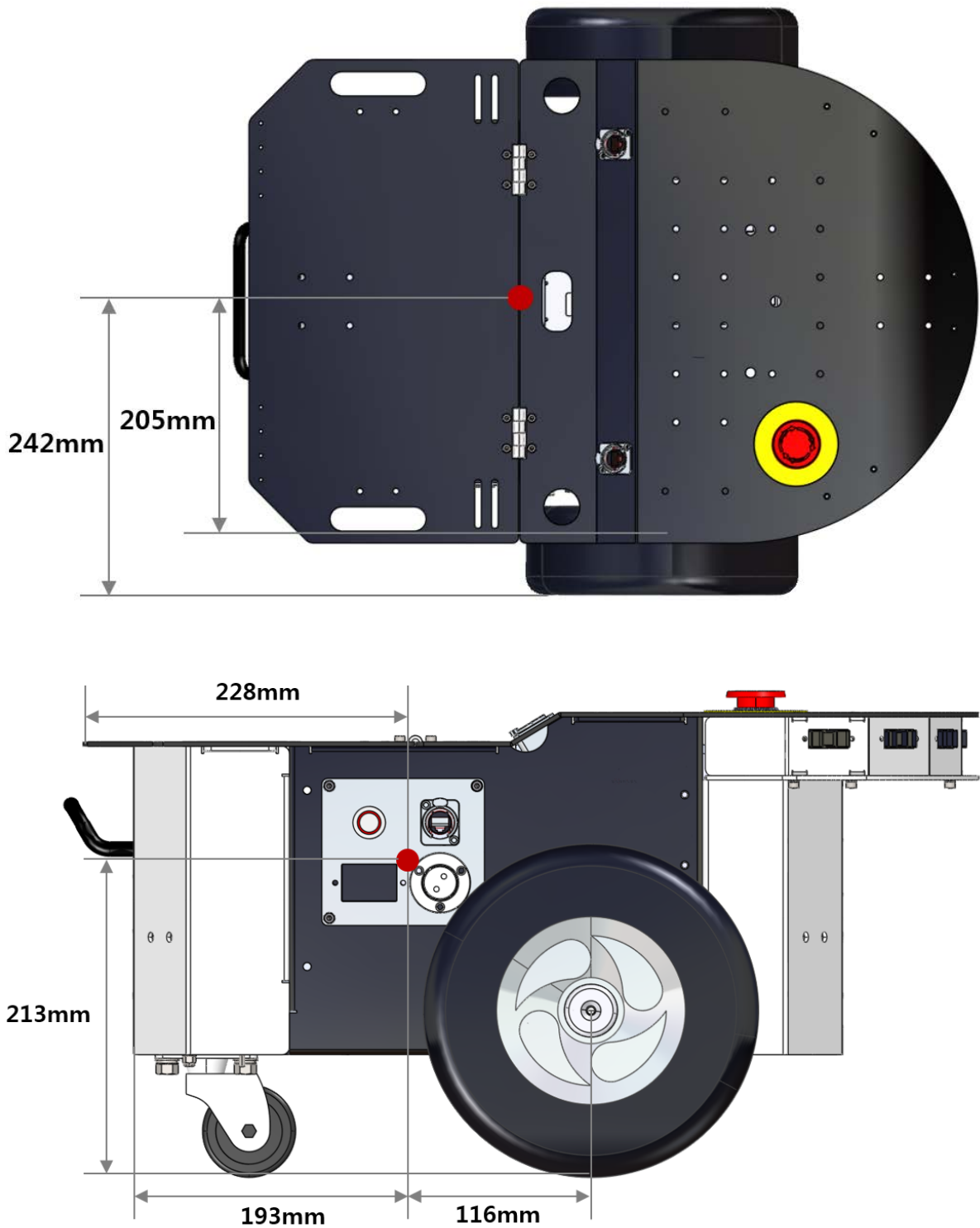


그림 4 AHRS 센서 장착 위치

### 3.4. 제어 패널

로봇 플랫폼의 오른쪽 측면에 아래 그림과 같은 제어 패널이 있습니다.

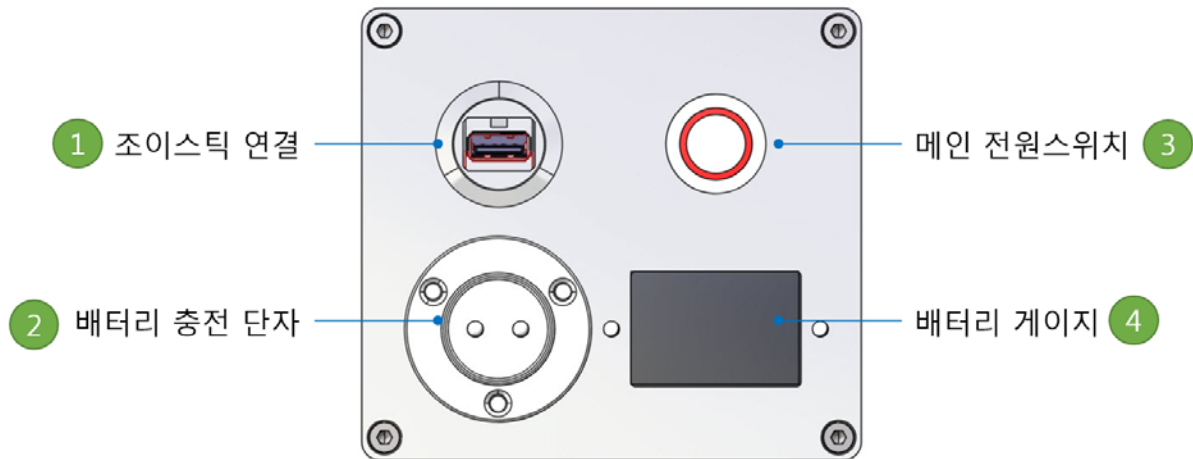


그림 5 측면 제어 패널

- ① USB 단자: 조이스틱 리시버를 연결할 때만 사용합니다.
- ② 배터리 충전 단자: 배터리 충전기를 연결하는 단자입니다.
- ③ 메인 전원스위치: Push-Lock 버튼으로 메인 전원을 On/Off 합니다.
- ④ 배터리 게이지: 현재 배터리 충전 상태를 표시합니다.

### 3.5. 비상스위치 및 제어기 연결

#### 가. 비상스위치

로봇 플랫폼의 상단에 비상스위치가 있습니다. 긴급 정지가 필요할 때 비상 스위치를 누르시면 전원이 차단되며 로봇이 정지하게 됩니다. 다시 해제할 때는 스위치를 시계방향으로 돌리면 풀립니다.

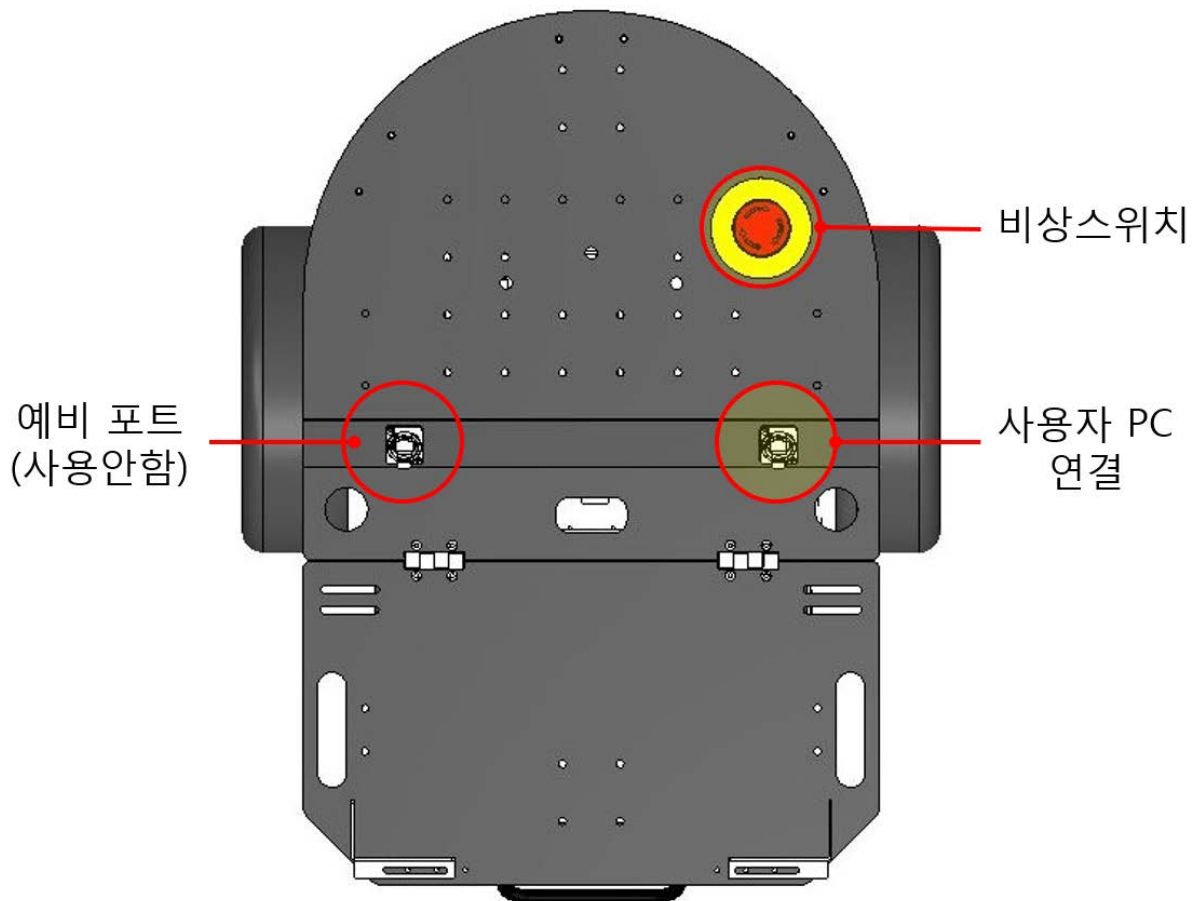


그림 6 비상스위치 및 사용자 PC 연결 포트

#### 나. 사용자 PC 연결

상단에 2개의 이더넷 포트가 있는데, 이 중에서 비상스위치가 있는 방향의 이더넷 포트를 사용합니다. 반대편 이더넷 포트는 연결되어 있지 않는 예비 포트입니다. 이더넷을 이용하는 장비를 추가하실 때 로봇 케이스 내부와 외부를 연결하는데 이용하실 수 있습니다.



## 4. 시작하기

### 4.1. 기본 조이스틱 구동

사용자 PC와의 연결 없이 내장 컨트롤러와 조이스틱을 이용하여 추가적인 설정 없이 바로 로봇을 구동할 수 있습니다. 먼저 로봇 플랫폼에 조이스틱 리시버를 연결합니다. 조이스틱 리시버는 조이스틱의 뒷면 배터리 커버를 제거하면 찾을 수 있습니다. 로봇 플랫폼의 측면 USB 단자에 이 리시버를 연결합니다.



조이스틱의 동작 모드를 **Direct X**로 설정되어 있는지 확인합니다.



로봇 플랫폼의 전원을 인가한 후 **약 15~20초 후에 조이스틱의 A 버튼을 클릭**합니다. 이 대기 시간은 내장 컨트롤러가 부팅된 후 프로그램이 실행되는 시간입니다. A 버튼은 내부 동작 모드를 조이스틱 제어 모드로 변경하는 것입니다. 그리고 조이스틱의 **MODE가 OFF**된 상태인지 확인합니다. 만약 ON이 되어 있다면 옆의 LED에 녹색 불이 들어온 상태이며, 이 때는 스틱이 아닌 패드가 동작합니다. 다시 MODE 버튼을 클릭하면 녹색 불이 꺼지면서 MODE가 OFF된 상태로 변경됩니다.



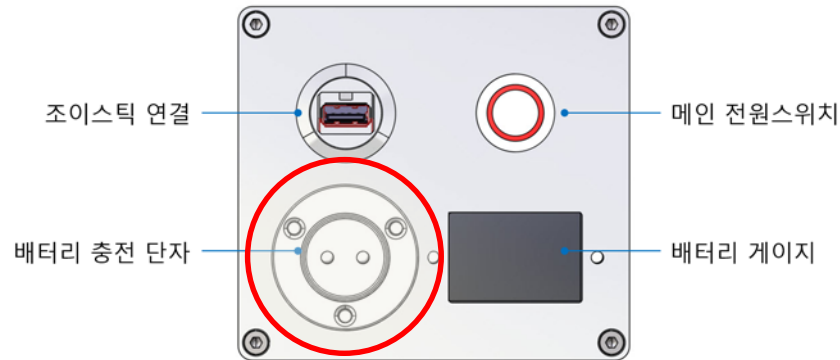
조이스틱 동작은 아래 그림과 같이 조작을 하면 로봇이 그대로 움직입니다. 두 스틱을 조합하여 곡률 주행을 만들 수 있습니다. **최대 선속도는 1.1 m/s로 제한**되어 있으며, **최소 값은 0.1m/s**입니다. 이 때 최대 각속도는 현재 설정된 **"최대 선속도\*2/윤거(0.41m)"** 입니다. 즉 최대 선속도가 1.0m/s일 때 최대 각속도는 4.878 rad/s 가 됩니다.



그림 7 기본 조이스틱 구동

## 4.2. 배터리 충전

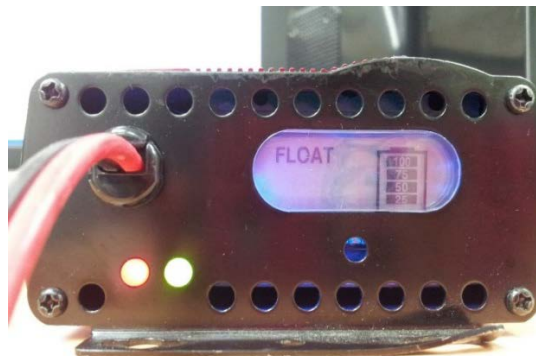
배터리 충전 단자에 홈이 있어 한 방향으로만 결합됩니다.



배터리 충전 단자와 연결한 후 충전기의 전원을 켭니다.



충전 중이면 주황램프가 켜지며 충전이 완료되면 초록램프가 켜집니다. FLOAT의 상태가 완충 상태입니다.



**주의: 로봇의 전원이 켜진 상태에서 충전하지 마시기 바랍니다. 이에 대한 보장은 하지 않습니다.**

## 5. 사용자 설정

### 5.1. 로봇 플랫폼과 연결

노트북과 같은 사용자의 PC와 로봇 플랫폼을 연결하기 위해서는 Ethernet 케이블이 필요합니다.  
아래 그림과 같이 연결하고 로봇의 전원을 켭니다.

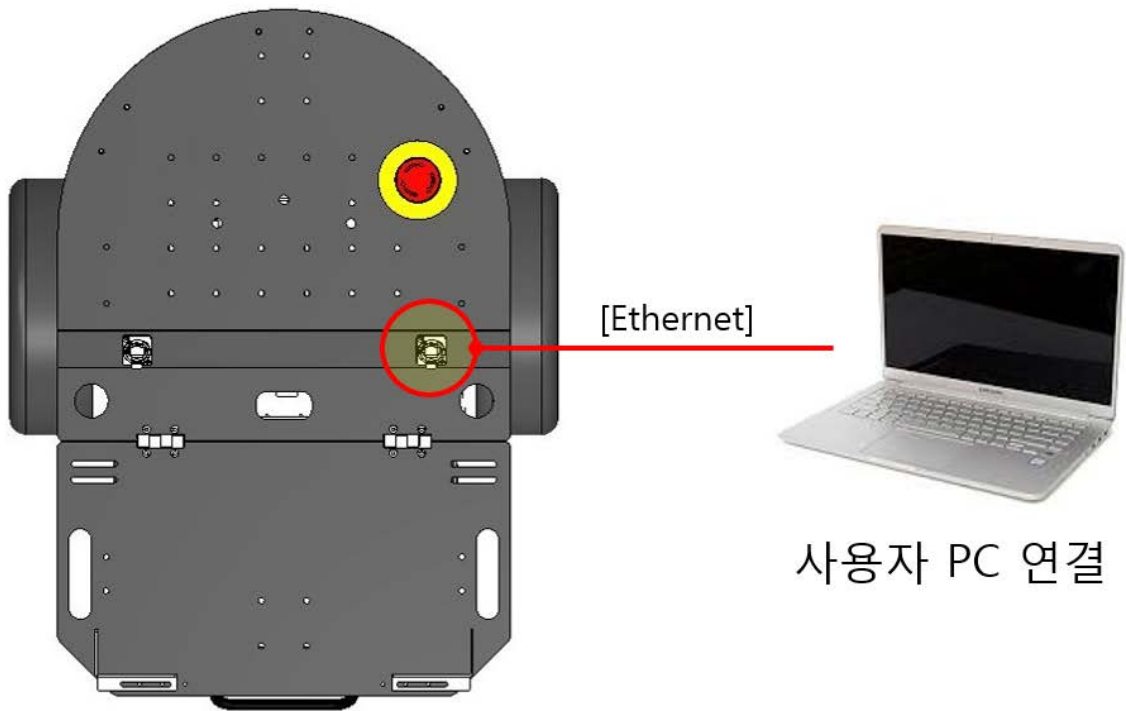


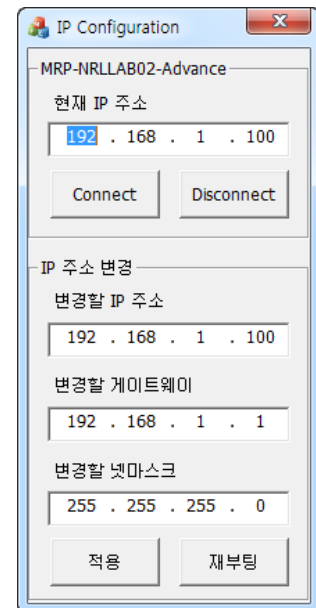
그림 8 로봇 플랫폼과 사용자 PC 연결

## 5.2. 로봇 플랫폼 IP 어드레스 설정

내장된 컨트롤러와 사용자 PC는 Ethernet기반의 TCP/IP 통신을 통해 연결됩니다. 내장 컨트롤러의 초기 IP 어드레스는 “192.168.1.100/24”로 고정 IP 어드레스가 설정되어 있습니다.

다음과 같은 과정을 통해 로봇의 IP 어드레스를 변경할 수 있습니다.

- ① 사용자 PC의 유선 이더넷을 192.168.1.xxx 로 고정 IP 어드레스로 변경합니다.
  - A. xxx에는 192.168.1.100을 제외한 1~254까지의 IP를 설정합니다.
- ② 로봇 플랫폼의 상단의 이더넷 포트와 사용자 PC의 이더넷 포트를 UTP 케이블(일반 LAN)을 이용해 서로 연결합니다.
- ③ 로봇의 전원을 켭니다.
- ④ 15~20초 후 제공된 IP Configuration을 실행합니다.
  - A. Connect 버튼을 클릭하여 로봇 내장 컨트롤러와 연결합니다.
  - B. IP 주소 변경 항목을 변경합니다.
    - i. 변경할 IP 주소: 변경할 IP 어드레스를 입력합니다.
      - Ex) 192.168.0.10
    - ii. 변경할 게이트웨이: 변경할 게이트웨이를 입력합니다.
      - Ex) 192.168.0.1
    - iii. 변경할 넷마스크: 변경할 넷마스크를 입력합니다.
      - Ex) 255.255.255.0
  - C. 적용 버튼을 클릭합니다.
    - i. 현재 설정한 내용을 내장 컨트롤러에 저장합니다.
  - D. 재부팅 버튼을 클릭합니다.
    - i. 변경된 사항은 재부팅을 해야 적용됩니다.
- ⑤ Ping 명령을 통해 IP 어드레스가 변경되었는지 확인합니다.



☞ 사용자 PC의 OS가 리눅스 계열이거나 변경된 IP 어드레스를 찾을 수 없을 때는 APPENDIX-A를 참조하시기 바랍니다.

## 6. 제어 라이브러리

사용자의 어플리케이션에 MRP-NRLAB02 로봇 플랫폼을 쉽게 적용할 수 있도록 제어 라이브러리를 제공합니다. 제어 라이브러리는 Windows, Linux 에서 사용 가능합니다.

표 3 제어 라이브러리 사용환경과 컴파일 환경

OS	컴파일러
Windows	Visual Studio 2015
Linux	g++ 4.2.1 이상, "-std=c++11" 옵션

제어 라이브러리는 내부적으로 소켓 통신이 구현되어 있습니다. TCP/IP 소켓 통신의 Client 기능으로 로봇 내장 컨트롤러에서 실행되고 있는 Server와 연결됩니다. 제어 라이브러리를 통해 로봇 플랫폼과 연결되면 10ms 주기로 데이터를 주고 받습니다.

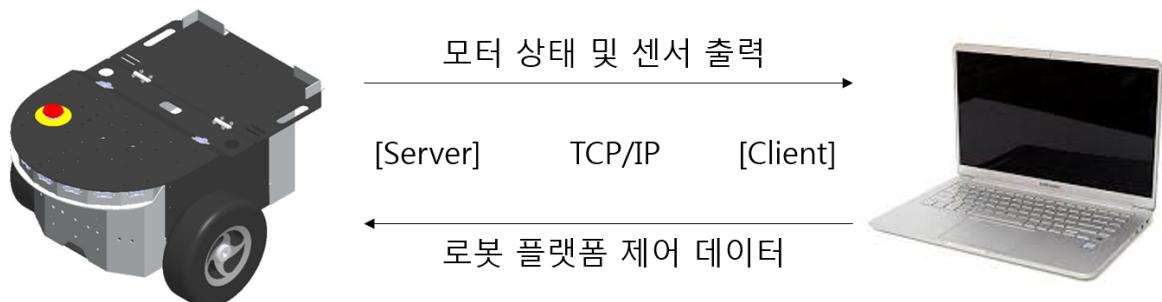


그림 9 제어 라이브러리 구조

## 6.1. 사용자 제어 데이터 구성(사용자 송신)

직접 TCP/IP 소켓 통신(포트 9911)을 구현할 경우 아래와 같은 데이터 구조를 사용자 입장에서 send 합니다.

```
typedef struct PACKED
{
    unsigned int marker;
    int controlMode;
    float maxLineSpeed;
    float tv;
    float rv;
    int TargetVelocity[2];
    int ProfileAccel;
}T_READ_DATA;
```

표 4 사용자 제어 데이터 구성

타입	변수명	설명	비고
uint	marker	시작점 marker: 0x75BD7E97	
int	controlMode	내부 동작 모드 설정 1: 조이스틱 동작 모드 2: Motor RPM 제어 모드(RPM 직접 제어) 3: Robot Motion 제어 모드(선속도, 각속도)	
float	maxLineSpeed	최대 선속도 설정	m/s
float	tv	동작모드 3일 때, 선속도 제어 값: $tv \leq \text{maxLineSpeed}$	m/s
float	rv	동작모드 3일 때, 각속도 제어 값: $rv \leq \text{maxLineSpeed} * 2 / \text{TREAD}$	rad/s
int	targetVelocity_Right	동작모드 2일 때, 오른쪽 모터 RPM 제어 값: -2500 ~ 2500	RPM
int	targetVelocity_Left	동작모드 2일 때, 왼쪽 모터 RPM 제어 값: -2500 ~ 2500	RPM
int	profileAccel	모터의 반응속도 설정: 1000 ~ 3000	

## 6.2. 로봇 출력 데이터 구성(사용자 수신)

직접 TCP/IP 소켓 통신(포트 9911)을 구현할 경우 아래와 같은 데이터 구조를 사용자 입장에서 read 합니다.

```
typedef struct
{
    unsigned int marker;
    float tv;
    float rv;
    int ActualPosition[2];
    int ActualVelocity[2];
    unsigned short psd[9];
    int x;
    int y;
    int z;
    int w;
    int ax;
    int ay;
    int az;
    int gx;
    int gy;
    int gz;
    int mx;
    int my;
    int mz;
    short temp;
    float curMaxLineSpeed;
    unsigned char devNum;
    unsigned char devState;
    int ProfileAccel;
}T_SEND_DATA;
```

표 5 사용자 수신 데이터 구성

타입	변수명	설명	비고
uint	marker	시작점 marker: 0x75BD7E97	
float	tv	현재 로봇 선속도	(m/s)
float	rv	현재 로봇 각속도	(rad/s)
int	actualPosition[2]	[0]: 오른쪽 바퀴 엔코더 [1]: 왼쪽 바퀴 엔코더	
int	actualVelocity[2]	[0]: 오른쪽 바퀴 RPM [1]: 왼쪽 바퀴 RPM	
uint16	Psd[9]	[0~6]: psd 센서 데이터	(cm)
int	x	쿼터니언 x	/10000.f



int	y	쿼터니언 y	/10000.f
int	z	쿼터니언 z	/10000.f
int	w	쿼터니언 w	/10000.f
int	ax	가속도 x	/10000.f (g)
int	ay	가속도 y	/10000.f (g)
int	az	가속도 z	/10000.f (g)
int	gx	자이로 x	/10000.f (dps*)
int	gy	자이로 y	/10000.f (dps)
int	gz	자이로 z	/10000.f (dps)
int	mx	마그네토 x	/10000.f
int	my	마그네토 y	/10000.f
int	mz	마그네토 z	/10000.f
Short	temp	온도	/10.f
float	curMaxLineSpeed	현재 설정된 최대 선속도 값	m/s
uint8	devNum	EtherCAT Slave 모듈 수 3: 정상 그 외: 에러	
uint8	devState	EtherCAT Slave 상태 0: 정상 상태 1 ~ 7: Slave 연결 에러 상태	
int	ProfileAccel	현재 설정된 모터 반응속도	

\* dps : degree per second

☞ 주의: 제어 라이브러리를 사용하지 않을 경우에는 센서 데이터의 스케일을 변환해 주어야 합니다. 제어 라이브러리의 헤더 파일에서 제공되는 센서 구조체는 스케일 변환 없이 그대로 사용하시면 됩니다.



### 6.3. 제어 라이브러리 구성

#### 가. 제어 라이브러리

사용할 플랫폼에 맞게 선택해서 사용하시면 됩니다.

표 6 제어 라이브러리 구성

지원 플랫폼	헤더 파일명	라이브러리 파일명
Windows 7, x86, x64	libnrlab02.h	libnrlab02d.lib (debug) libnrlab02.lib (release)
Ubuntu 16.04, x86_64	libnrlab02.h	libnrlab02.a

#### 나. 제어 라이브러리 함수 리스트

상세 설명은 **APPENDIX-B** 를 참조하시기 바랍니다.

표 7 제어 라이브러리 함수 리스트

구분	함수명	설명
로봇 연결	NWrite_EnableNetwork(char* ip)	로봇과 연결
	NWrite_DisableNetwork()	로봇과의 연결 종료
	NRead_RobotDeviceNum()	로봇 내장 모듈의 수 읽기 (고장 체크)
	NRead_RobotDeviceState()	로봇 내장 모듈의 상태 읽기 (고장체크)
모터 제어	MWrite_SetProfileAcceleration(int accel)	모터 반응속도 제어
	MRead_GetProfileAcceleration(int &accel)	모터 반응속도 읽기
	MWrite_MotorVelocity(int velL, int velR)	모터 RPM 직접 제어.
	MWrite_MotorStop()	모터를 정지하는 함수
로봇 제어	RWrite_SetMaxVelocity(float maxvel)	로봇 최대속도 제한
	RWrite_2W_Kinematics(float tv, float rv)	선속도, 각속도로 로봇을 직접 제어
	RWrite_StopDrive()	로봇을 정지하는 함수



로봇 상태	MRead_MotorVelocity(int &velL, int &velR)	모터 RPM 읽기
	MRead_MotorPosition(int &posL, int &posR)	모터 엔코더 읽기
	RRead_GetMaxLineSpeed(float &maxls)	현재 설정된 최대 선속도 읽기
	RRead_2W_Kinematics(float &tv, float &rv)	로봇의 현재 선속도, 각속도 읽기
	SRead_SensorData(T_SENSOR_DATA &data)	로봇 센서 읽기
<p>☞ 주의: <u>모터 직접 제어 함수</u>와 <u>로봇 제어 함수</u>는 동시에 동작하지 않습니다. 1cycle의 제어 주기에서 두 제어 함수 MWrite_MotorVelocity와 RWrite_2W_Kinematics 중 하나만 사용하시기 바랍니다. 동시에 사용할 경우 마지막에 실행된 제어 함수로 동작합니다.</p>		

## 7. 예제 프로그램

### 7.1. Ubuntu

매뉴얼 CD의 소프트웨어/Linux/samples 폴더에 샘플 프로그램이 있습니다.

폴더명	내용	
bin	샘플 프로그램의 빌드 결과	
libs	MRP-NRLAB02 제어 라이브러리	
include	MRP-NRLAB02 제어 라이브러리 헤더 파일	
samples	파일명	예제
	ecat_state.cpp	로봇 상태 점검 예제
	motor_rpm.cpp	모터 RPM을 직접 제어하는 예제
	vel_cmd.cpp	선속도, 각속도로 로봇을 제어하는 예제
	sensor_read.cpp	센서 데이터를 읽어오는 예제

예제 프로그램을 빌드하려면 samples 폴더에서 make를 입력하시기 바랍니다.

```
linux/samples$ make
```

예제를 실행하려면 bin 폴더로 이동 후 다음과 같은 방법으로 실행할 수 있습니다.

실행 파일	실행 방법 및 결과
ecat_state	./ecat_state 로봇 상태 정보 확인
motor_rpm	./motor_rpm 샘플 코드에 입력된 RPM 값으로 모터 동작
vel_cmd	./vel_cmd 선속도 0.1m/s에서부터 1.0 m/s 까지 로봇 직진 제어
sensor_read	./sensor_read 센서 데이터 출력



## 7.2. ROS Kinetic

### 가. MRP-NRLAB02 패키지 설치

매뉴얼 CD의 소프트웨어/Linux/ROS 에서 mrp\_nrlab02 패키지를 찾을 수 있습니다. 이를 사용자 PC의 ROS 빌드 시스템에 복사합니다.

먼저 의존성 패키지를 설치해야 합니다. 사용되는 ROS 패키지와 설치 방법은 다음과 같습니다.

```
$ sudo apt install ros-kinetic-dynamic_reconfigure
$ sudo apt install ros-kinetic-nodelet
$ sudo apt install ros-kinetic-sensor_msgs
$ sudo apt install ros-kinetic-tf
$ sudo apt install ros-kinetic-tf2_geometry_msgs
$ sudo apt install ros-kinetic-tf2_ros
$ sudo apt install ros-kinetic-urdf
```

만약 기본 빌드 경로인 ~/catkin\_ws/src 에 mrp\_nrlab02 패키지를 복사했다면 다음과 같은 명령을 통해 빌드합니다.

```
$ cd ~/catkin_ws
$ catkin_make
$ rospack profile
```

mrp\_nrlab02 패키지에는 메시지 파일과 dynamic reconfigure 파일도 있기 때문에 한 번에 빌드가 성공하지 못 할 수도 있습니다. 빌드 메시지를 살펴보고 메시지 파일이나 cfg 파일을 찾지 못했다는 메시지가 있다면 다시 빌드하시면 됩니다.



## 나. nrlab02 노드 실행

빌드가 성공적으로 완료되었다면 다음과 같은 명령을 통해 실행할 수 있습니다.

```
$ roslaunch mrp_nrlab02 mrp_nrlab02.launch
```

mrp\_nrlab02.launch 파일을 보면 세 가지 파라미터가 있습니다. 하나는 로봇 플랫폼의 IP 어드레스를 지정하는 파라미터와 초기 최대 선속도를 지정하는 파라미터가 있습니다. 이를 상황에 맞게 수정해서 사용하시기 바랍니다. 이 중에서 max\_line\_speed 와 motor\_profile\_acceleration 파라미터는 dynamic\_reconfigure가 가능하도록 되어 있습니다.

```
<launch>
  <node pkg="mrp_nrlab02" name="nrlab_node" type="nrlab_node" output="screen">
    <param name="robot_ip" type="str" value="192.168.1.100" />
    <param name="max_line_speed" type="double" value="0.3" />
    <param name="motor_profile_Acceleration" type="int" value="2000" />
  </node>
</launch>
```

```
msi@msi-GE63VR-7RF:~/catkin_ws$ roslaunch mrp_nrlab02 mrp_nrlab02.launch
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://msi-GE63VR-7RF:39245/

SUMMARY
=====

PARAMETERS
* /nrlab_node/max_line_speed: 0.5
* /nrlab_node/motor_profile_Acceleration: 2000
* /nrlab_node/robot_ip: 192.168.1.100
* /rostdistro: kinetic
* /rosversion: 1.12.12

NODES
/
  nrlab_node (mrp_nrlab02/nrlab_node)

auto-starting new master
process[master]: started with pid [21695]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 176dd352-43be-11e8-9a2a-9cb6d06a2c0b
process[rosout-1]: started with pid [21708]
started core service [/rosout]
process[nrlab_node-2]: started with pid [21712]
[ INFO] [1524134435.500207668]: Change parameters : 0.300000, 2000
[ INFO] [1524134435.501637532]: Got param: 192.168.1.100
[ INFO] [1524134435.502028855]: Got param: 0.500000
[ INFO] [1524134435.502430597]: Got param: 2000
[Socket] socket created!!
[Socket] try Connect ...
```



## 다. Parameters

rqt\_reconfigure를 실행하면 다음과 같이 MAX\_LINE\_SPEED와 Motor Profile Acceleration을 설정할 수 있습니다.

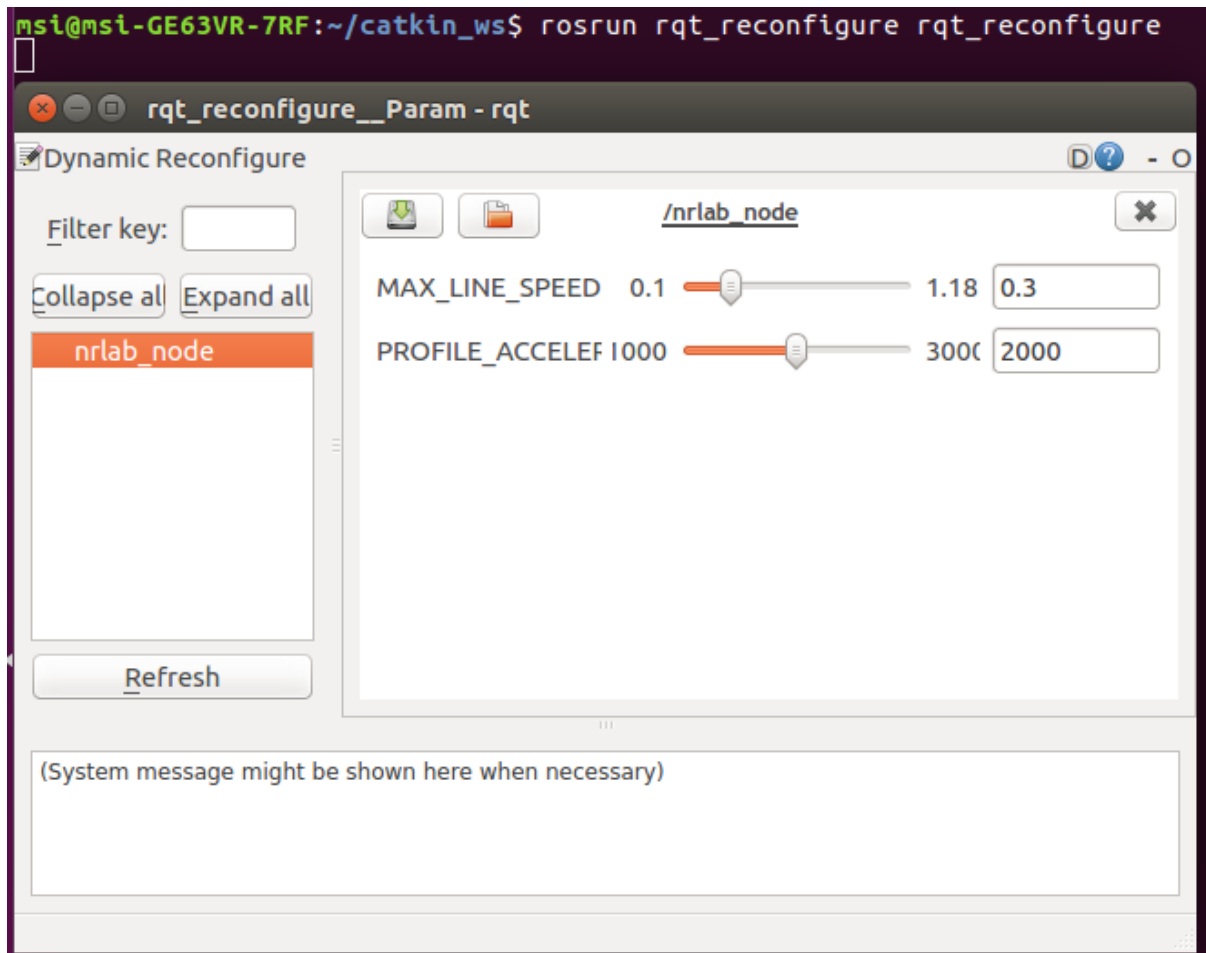


그림 10 ROS Dynamic reconfigure – rqt\_reconfigure

## 라. TOPICS

MRP-NRLAB02 패키지의 TOPIC은 다음과 같습니다.

### ■ Published topics

#### ■ [/nrlab/sensor](#) : MRP-NRLAB02의 센서 데이터

- msg 폴더에 정의된 메시지 입니다.
  - IR-PSD 센서
  - 엔코더 펄스 정보
  - 모터 RPM 정보
  - 로봇 현재 선속도(tv), 각속도(rv)
  - AHRS Quaternion
  - AHRS 3축 가속도 (g)
  - AHRS 3축 자이로 (dps)
  - AHRS 3축 마그네토

### ■ Subscribed topics

#### ■ [/nrlab/cmd\\_vel](#) : MRP-NRLAB02 모션 제어 (선속도, 각속도 입력)

- geometry\_msgs/twist

### ■ Topic list

```
msi@msi-GE63VR-7RF:~/catkin_ws$ rostopic list
/nrlab/cmd_vel
/nrlab/sensor
/nrlab_node/parameter_descriptions
/nrlab_node/parameter_updates
/rosout
/rosout_agg
/statistics
```



## 마. RVIZ

제공해드리는 Launch 파일은 Rviz나 Gazebo에 로봇 모델을 표시하는 기능만을 제공합니다.

아래 display.launch는 rviz에 로봇 모델링을 표시하는 기능만 있습니다.

```
$ roslaunch mrp_nrlab02 display.launch
```

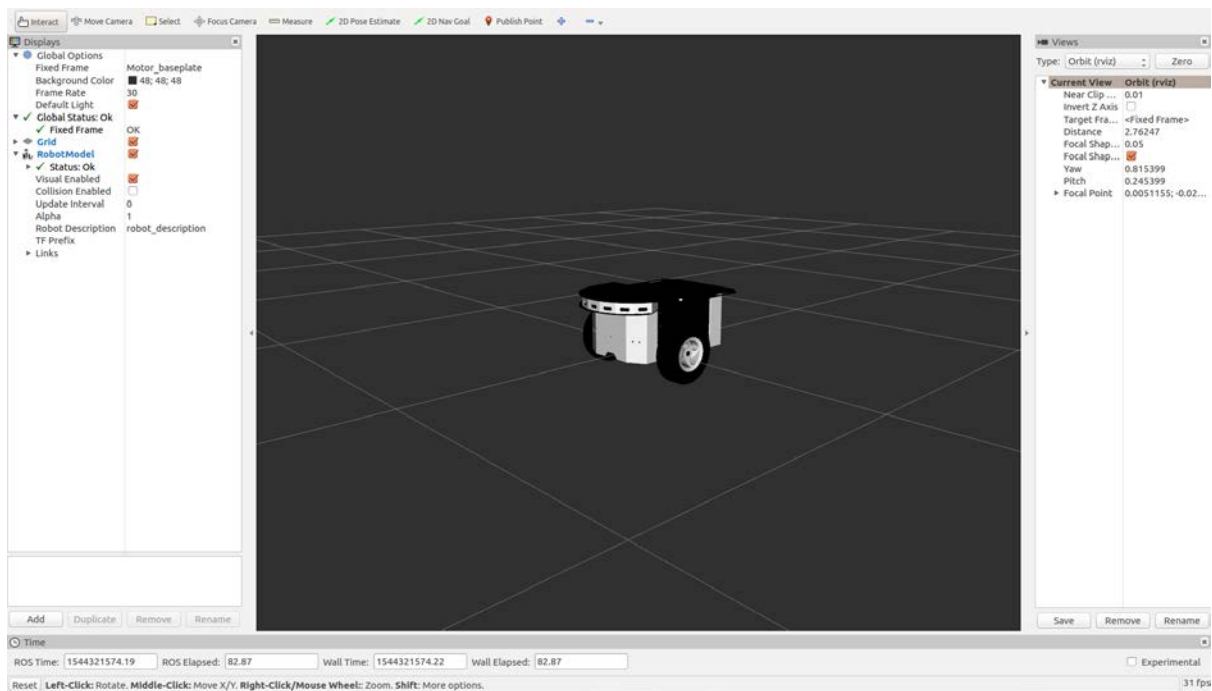


그림 11 Rviz RobotModel 디스플레이

## 바. Gazebo

Gazebo를 이용한 MRP-NRLAB02 시뮬레이션 기능을 제공합니다. Gazebo가 실행되면, Gazebo의 Differential Drive 플러그인이 적용되어 있어 /odom 토픽을 제공합니다. 로봇의 현재 위치와 같은 정보가 필요하시면 /odom 메시지 또는 /gazebo/model\_states 메시지를 수신하시면 됩니다. 그 외의 추가 기능은 직접 구현하시거나 문의 주시기 바랍니다.

다음과 같은 명령을 통해 실행할 수 있습니다.

```
$ roslaunch mrp_nrlab02 gazebo.launch
```

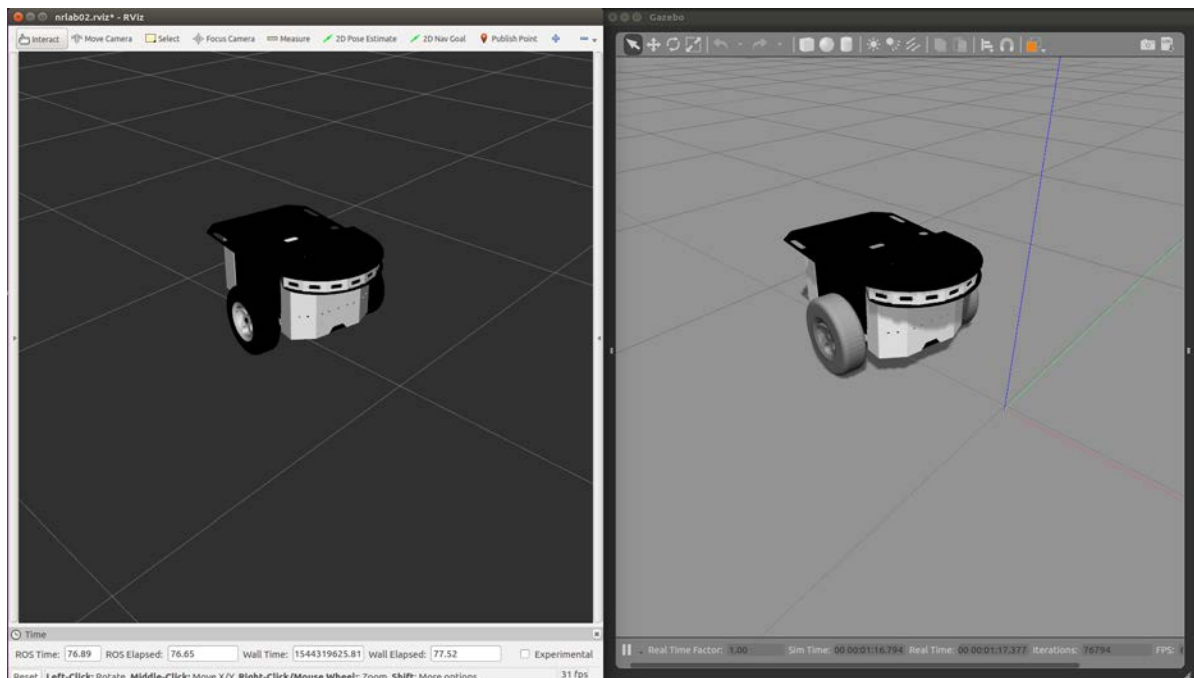


그림 12 rviz & gazebo 시뮬레이션

Gazebo의 로봇은 geometry\_msgs/Twist 타입의 /cmd\_vel 명령을 통해 이동시킬 수 있습니다. 빠른 테스트를 위해 아래 명령을 통해 키보드를 이용한 /cmd\_vel 토픽을 제공하는 노드를 실행시킬 수 있습니다.

```
$ roslaunch mrp_nrlab02 keyboard_teleop.launch
```

//실행이 안될 경우 아래 명령을 시도해보시기 바랍니다.

```
$ cd ~/catkin_ws/src/mrp_nrlab02/scripts/
```

```
$ chmod a+x diff_wheeled_robot_key
```

```
$ sudo apt install ros-kinetic-turtlebot-teleop
```





### 7.3. Windows

#### 가. 예제 프로그램 실행

매뉴얼 CD의 소프트웨어/Windows/samples 폴더에 샘플 프로그램이 있습니다. 리눅스와 동일한 예제 입니다.

폴더명	내용	
	폴더명	예제
examples	ecat_state	로봇 상태 점검 예제
	motor_rpm	모터 RPM을 직접 제어하는 예제
	vel_cmd	선속도, 각속도로 로봇을 제어하는 예제
	sensor_read	센서 데이터를 읽어오는 예제

examples 폴더의 예제 프로그램의 폴더로 이동하면 Visual studio 프로젝트 파일(.vcxproj)이 있습니다. 이 프로젝트 파일을 더블 클릭해서 Visual studio를 열고 빌드 및 실행을 합니다.

이름	수정된 날짜	유형	크기
 libnrlab02.h	2018-04-17 오후...	C/C++ Header	2KB
 libnrlab02d.lib	2018-04-17 오후...	Object File Library	327KB
 vel_cmd.cpp	2018-04-17 오후...	C++ Source	1KB
 <b>vel_cmd.vcxproj</b>	2018-04-17 오후...	VC++ Project	8KB

vel\_cmd 예제 프로그램의 실행 결과 입니다.

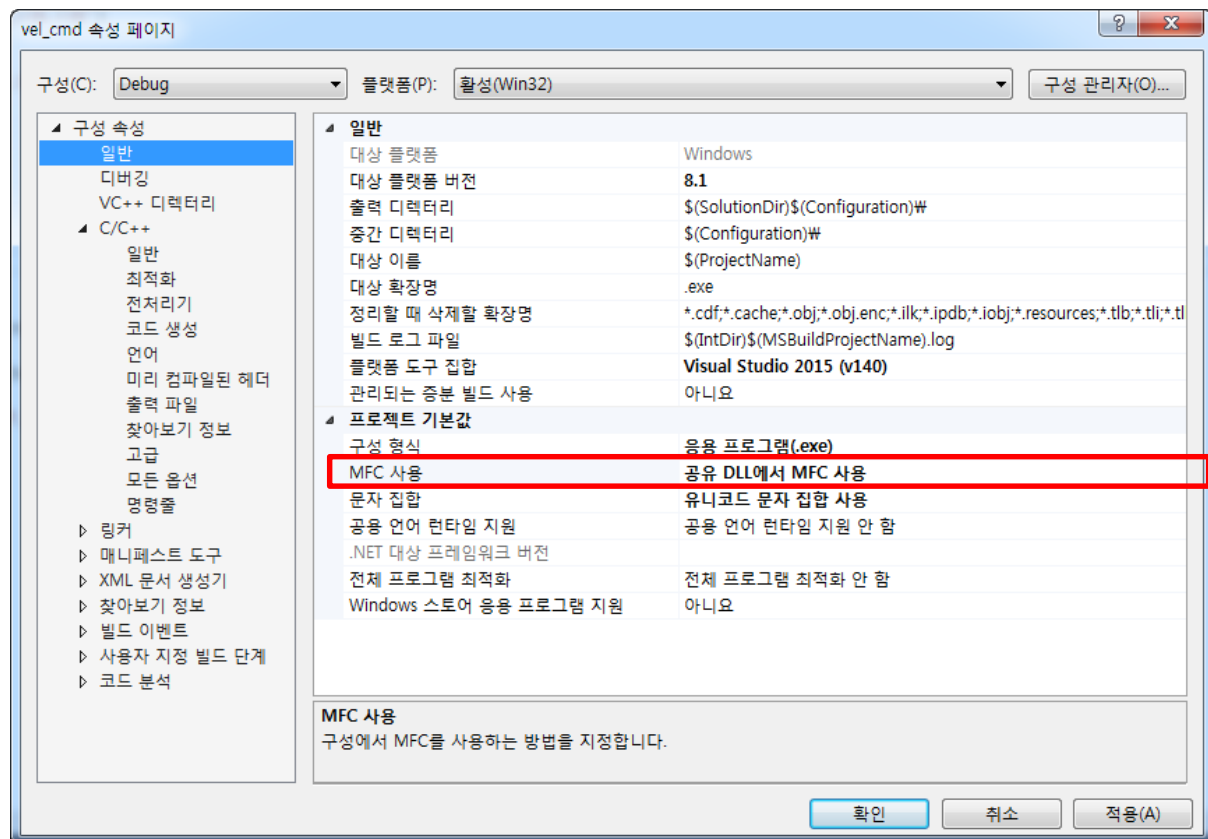
```
C:\Windows\system32\cmd.exe
set vel : 0.1256, 0.0000 / 0.1167, -0.0102
set vel : 0.1258, 0.0000 / 0.1147, -0.0195
set vel : 0.1260, 0.0000 / 0.1158, -0.0076
set vel : 0.1262, 0.0000 / 0.1158, -0.0127
set vel : 0.1264, 0.0000 / 0.1167, -0.0038
set vel : 0.1266, 0.0000 / 0.1154, -0.0195
set vel : 0.1268, 0.0000 / 0.1196, -0.0059
set vel : 0.1270, 0.0000 / 0.1229, 0.0017
set vel : 0.1272, 0.0000 / 0.1254, 0.0170
set vel : 0.1274, 0.0000 / 0.1276, 0.0229
set vel : 0.1276, 0.0000 / 0.1287, 0.0229
set vel : 0.1278, 0.0000 / 0.1315, 0.0399
set vel : 0.1280, 0.0000 / 0.1278, 0.0136
set vel : 0.1282, 0.0000 / 0.1288, 0.0119
set vel : 0.1284, 0.0000 / 0.1290, 0.0127
set vel : 0.1286, 0.0000 / 0.1292, 0.0000
set vel : 0.1288, 0.0000 / 0.1285, -0.0010
set vel : 0.1290, 0.0000 / 0.1288, -0.0102
set vel : 0.1292, 0.0000 / 0.1273, -0.0212
set vel : 0.1294, 0.0000 / 0.1287, -0.0164
set vel : 0.1296, 0.0000 / 0.1252, -0.0161
set vel : 0.1298, 0.0000 / 0.1245, -0.0144
set vel : 0.1300, 0.0000 / 0.1247, 0.0017
set vel : 0.1302, 0.0000 / 0.1226, -0.0119
```

## 나. 새 프로젝트 생성 시

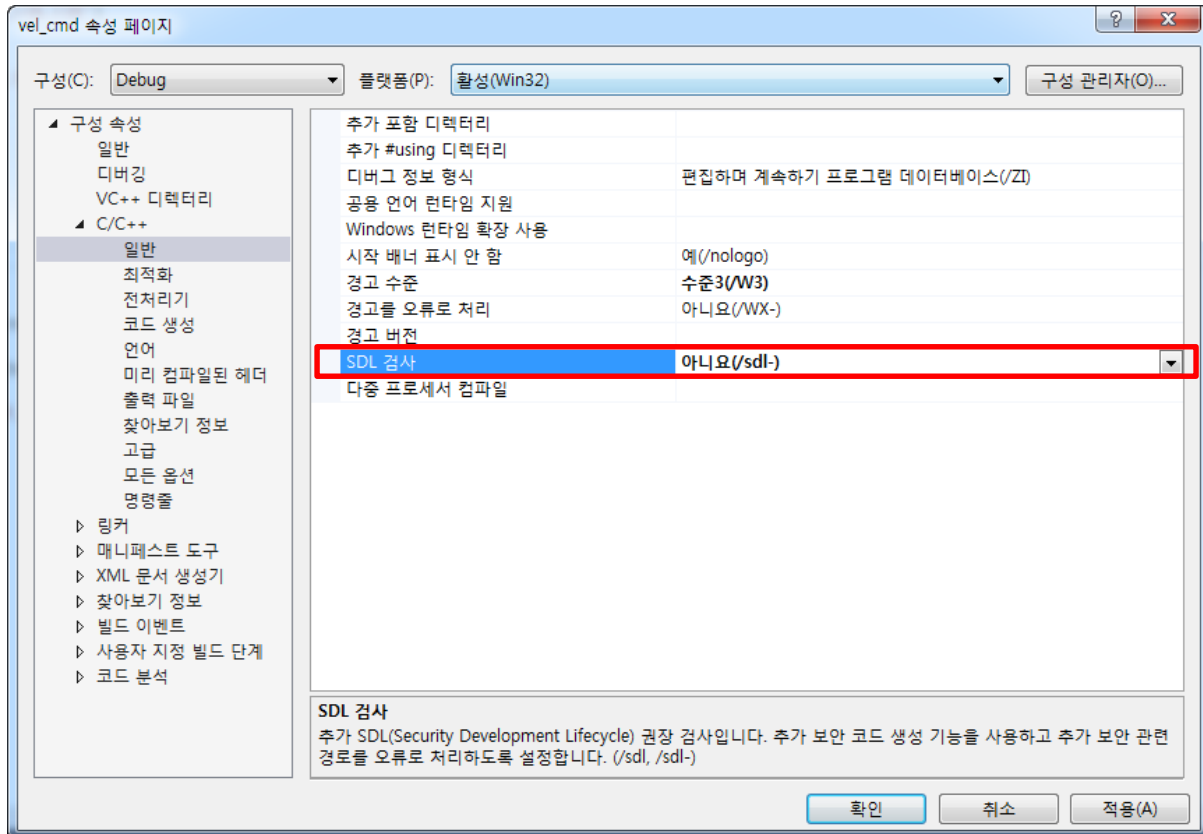
매뉴얼 CD의 “소프트웨어/Windows/include 폴더”와 “소프트웨어/Windows/lib 폴더” 에서 해당 플랫폼에 맞는 라이브러리를 프로젝트에 추가합니다.

새로운 프로젝트에 제어 라이브러리를 추가할 때는 다음과 같은 프로젝트 옵션을 수정해야 합니다.

- MFC 사용
  - 구성속성 -> 일반 -> MFC 사용
  - 공유 DLL에서 MFC 사용



- SDL 검사 사용 안함
  - 구성속성 -> C/C++ -> 일반 -> SDL 검사
  - 아니요(/sdl-)



## 8. APPENDIX A – IP Configuration

### 8.1. Ubuntu 의 경우

동봉된 매뉴얼 CD의 **Tools** 폴더에 IP configuration 툴이 있습니다. 다음과 같은 과정을 통해 IP를 설정합니다.

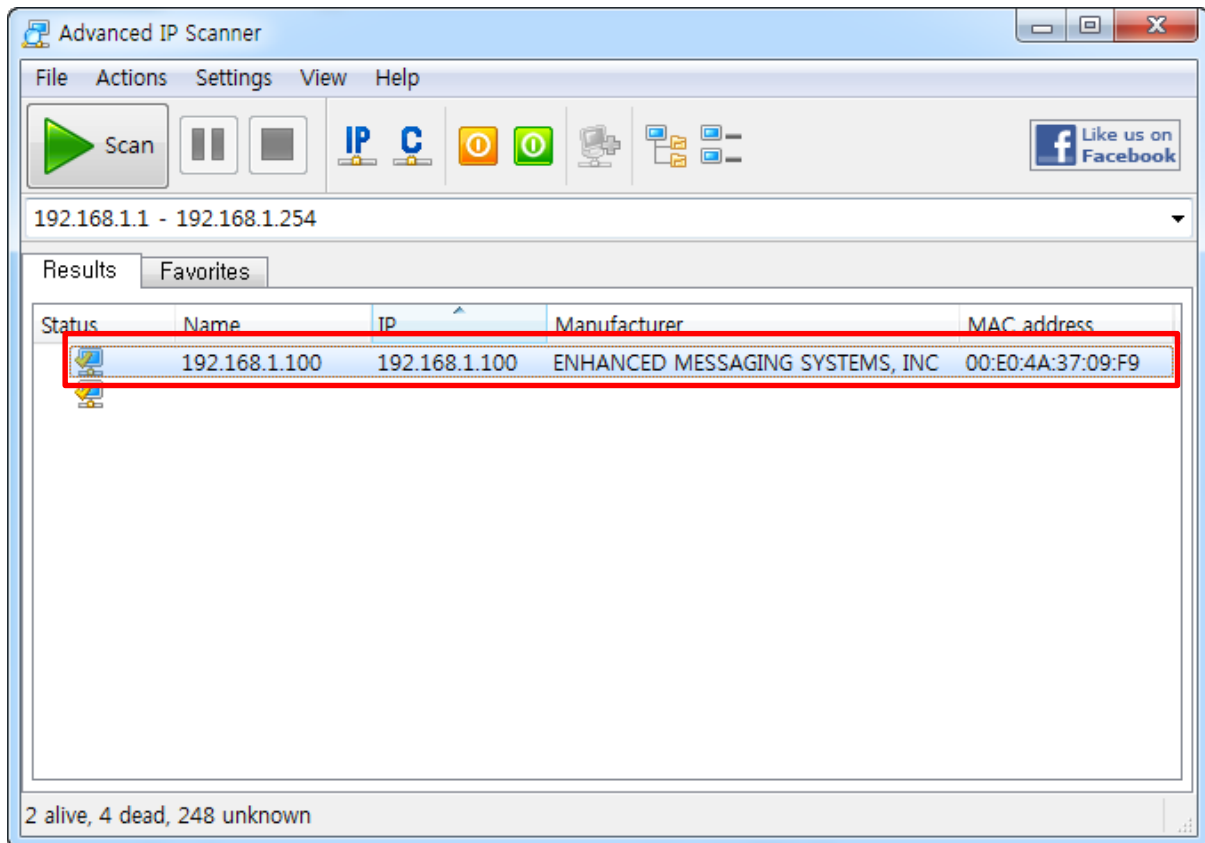
- ① 사용자 PC의 유선 이더넷을 192.168.1.xxx 로 고정 IP 어드레스로 변경합니다.
  - A. xxx에는 192.168.1.100을 제외한 1~254까지의 IP를 설정합니다.
- ② 로봇 플랫폼의 상단의 이더넷 포트와 사용자 PC의 이더넷 포트를 UTP 케이블(일반 LAN)을 이용해 서로 연결합니다.
- ③ 로봇의 전원을 켭니다.
- ④ 15~20초 후 제공된 매뉴얼 CD의 Tools 폴더로 이동합니다.
  - A. 사용자 PC로 ipChangeTool을 복사합니다.
    - i. `cp ./ipChangeTool ~/`
    - ii. `cd ~/`
    - iii. `chmod a+x ipChangeTool`
  - B. ipChangeTool을 실행합니다.
    - i. `./ipChangeTool [robot_ip]`
  - C. IP 주소 변경 항목을 입력합니다.
    - i. 변경할 IP 주소: 변경할 IP 어드레스를 입력합니다.  
 ■ Ex) 192.168.0.10
    - ii. 변경할 게이트웨이: 변경할 게이트웨이를 입력합니다.  
 ■ Ex) 192.168.0.1
    - iii. 변경할 넷마스크: 변경할 넷마스크를 입력합니다.  
 ■ Ex) 255.255.255.0
  - D. Yes를 입력하여 변경사항을 적용합니다.
    - i. 현재 설정한 내용을 내장 컨트롤러에 저장합니다.
  - E. Yes를 입력하여 재부팅 합니다.
    - i. 변경된 사항은 재부팅을 해야 적용됩니다.
    - ii. 또는 로봇 전원을 재인가 합니다.
- ⑤ Ping 명령을 통해 IP 어드레스가 변경되었는지 확인합니다.



## 8.2. 변경된 로봇 IP 어드레스를 잃어버린 경우

### 가. Windows 경우

무료 IP Scanner를 이용하는 방법이 있습니다.



### 나. 리눅스의 경우

nmap을 이용하여 스캔 할 수 있습니다.

```
$ sudo apt install nmap
```

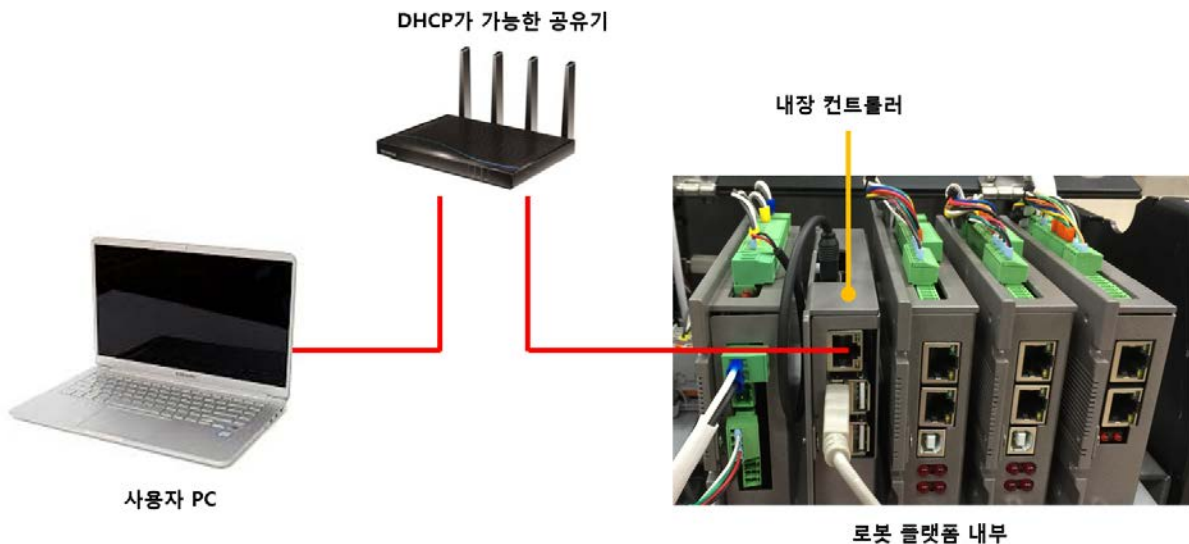
```
$ sudo nmap -sn x.x.x.0/24
```

```
msi@msi-GE63VR-7RF:~$ sudo nmap -sn 192.168.1.0/24
Starting Nmap 7.01 ( https://nmap.org ) at 2018-04-17 14:44 KST
Nmap scan report for 192.168.1.100
Host is up (0.00039s latency).
MAC Address: 00:E0:4A:37:09:F9 (ZX Technologies)
Nmap scan report for 192.168.1.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 3.74 seconds
```

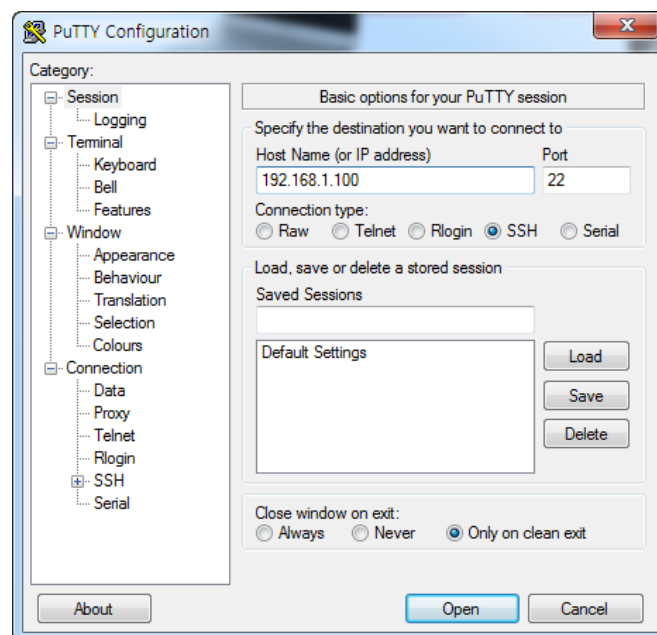


### 다. IP 어드레스를 찾을 수 없는 경우

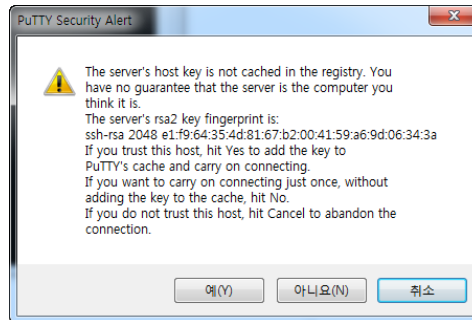
변경된 IP 어드레스를 찾을 수 없다면 직접 내장 컨트롤러에 접속해서 변경해주어야 합니다. 먼저 DHCP가 가능한 공유기가 필요합니다. 로봇 상단 커버를 열면 옆의 내장 컨트롤러를 볼 수 있으며, 랜 케이블이 바로 옆의 모듈과 연결된 것을 확인할 수 있습니다. 모듈과 연결된 랜 케이블을 아래 그림과 같이 다시 연결합니다. 그리고 로봇의 전원을 키고 공유기 관리 사이트로 들어가거나 앞서 IP 어드레스를 찾는 방법을 이용하여 내장 컨트롤러에 할당된 IP 어드레스를 찾습니다.



Putty와 같은 ssh 접속 프로그램을 활용하여 찾아낸 IP 어드레스를 이용하여 내장 컨트롤러에 접속합니다.

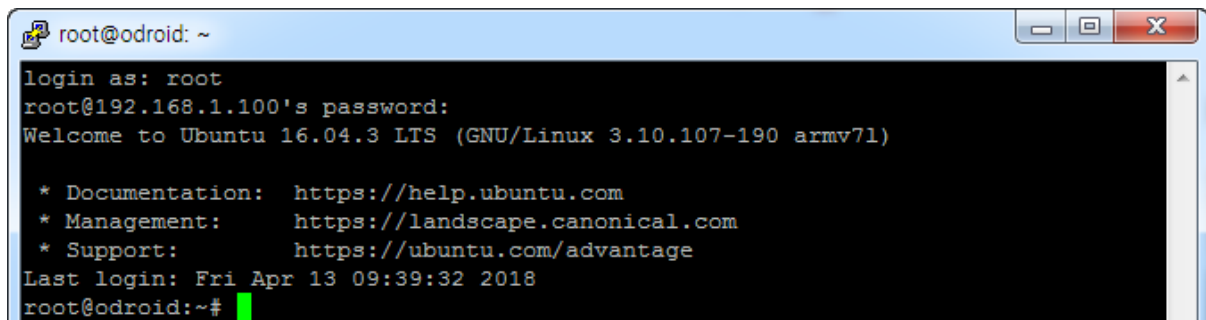


새로운 접속이기 때문에 예를 누릅니다.



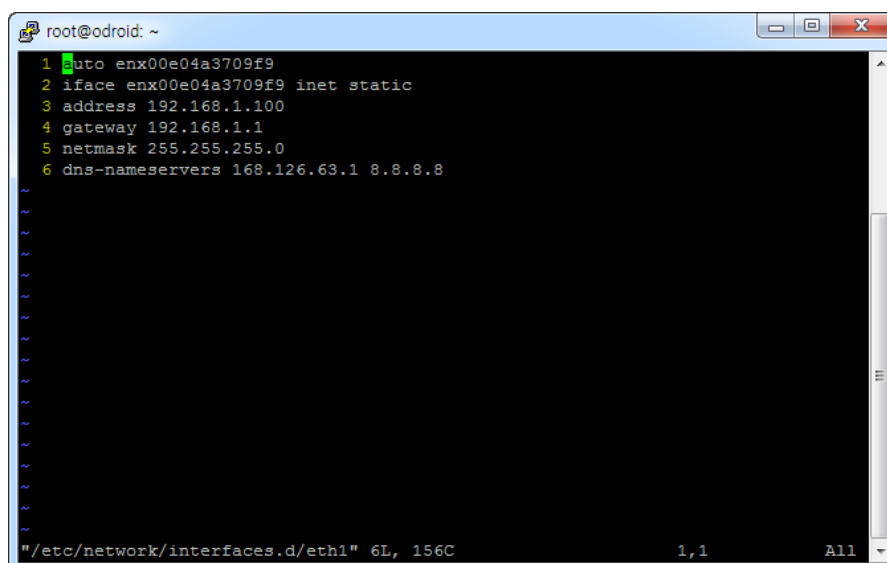
다음과 같은 로그인 정보를 입력합니다.

- 계정: root
- 비밀번호: nrlab



Vim 에디터를 이용하여 네트워크 설정 파일을 수정합니다. 이 중에서 Address, gateway, netmask 항목만 변경하시고 저장 후 재부팅 하시면 적용됩니다.

```
~# vim /etc/network/interfaces.d/eth1
```



## 9. APPENDIX B Library Function List

### 9.1. 동작 모드 정의

로봇에 내장된 컨트롤러는 아래 3가지 모드로 동작하고 있습니다. 제어 라이브러리 함수를 사용할 경우 제어 함수에 따라 동작모드를 자동으로 변경하기 때문에 동작 모드를 설정하는 함수는 별도로 제공하지 않습니다. 따라서 1cycle의 제어 주기에서 모터 제어 함수(MWrite\_MotorVelocity)와 모션 제어 함수(RWrite\_2W\_Kienmatics)를 동시에 사용할 경우 마지막에 실행한 함수에 따라 동작 모드가 변경됩니다.

동작 모드	실제 값	의미
조이스틱 동작 모드	1	조이스틱을 이용한 구동 <ul style="list-style-type: none"> <li>- 이 모드를 동작하려면 사용자 PC와 내장 컨트롤러의 통신(TCP/IP)이 연결되지 않은 상태여야 합니다.</li> <li>- 조이스틱의 A 버튼을 입력하면 해당 모드로 변경됩니다. (4장 시작하기 참조)</li> </ul>
MOTOR 직접 제어	2	모터의 RPM을 직접 제어 <ul style="list-style-type: none"> <li>- 범위 : -2500 ~ 2500</li> <li>- ex) 2500일 경우 1.18m/s</li> </ul>
로봇 모션 제어	3	로봇의 선속도 및 각속도를 입력하여 제어하는 모드 <ul style="list-style-type: none"> <li>- 선속도: maxLineSpeed에 설정한 값으로 제한</li> <li>- 각속도: maxLineSpeed*2/TRED 값으로 제한</li> </ul>

### 9.2. 함수 동작 결과 정의

모든 라이브러리에 포함된 함수는 실행 결과를 반환합니다. 5가지의 상태가 정의되어 있습니다.

상태 정의	실제 값	의미
SUCCESS	0	정상 동작
CONNECTIONSTATE_CONNECTED	1	로봇과 연결된 상태
CONNECTIONSTATE_FAULTED	2	로봇과 연결 실패 또는 연결되지 않은 상태
OUT_OF_RANGE	3	입력한 제어 값이 범위를 벗어난 경우
ETHERCAT_FAULTED	4	로봇 내부 EtherCAT 통신 에러 <ul style="list-style-type: none"> <li>- 자동 복구 기능이 동작</li> </ul>



### 9.3. 중요 파라미터

본 버전의 제어 라이브러리는 두 가지 중요한 파라미터를 제공합니다. 두 가지 파라미터는 로봇 플랫폼의 최대 선속도 제한과 모터의 반응 속도를 변경할 수 있습니다.

파라미터	설명	기본값	최소값	최대값	단위
MOTOR_PROFILE_ACCELERATION (INT32)	<ul style="list-style-type: none"> <li>- 모터 드라이브가 목표 속도에 도달하기 위한 가속도를 설정하는 파라미터</li> <li>- 값이 클수록 모터의 반응속도(가/감속)가 빨라짐</li> <li>- 관련 함수 <ul style="list-style-type: none"> <li>- RWrite_SetMaxLineSpeed</li> <li>- RRead_GetMaxLineSpeed</li> </ul> </li> </ul>	2000	1000	3000	-
MAX_LINE_SPEED (FLOAT)	<ul style="list-style-type: none"> <li>- 로봇 모션 제어 모드로 제어할 때 로봇의 최대 선속도를 제한하는 파라미터</li> <li>- 모터 RPM 직접 제어일 때는 영향을 주지 않음</li> <li>- 관련함수 <ul style="list-style-type: none"> <li>- MWrite_SetProfileAcceleration</li> <li>- MRead_GetProfileAcceleration</li> </ul> </li> </ul>	0.3	0.1	1.18	m/s

### 9.4. 로봇 연결 제어

함수 원형	NRLAB_State NWrite_EnableNetwork(char* ip);		
함수 설명	로봇과 연결하는 함수 1초에 한번씩 연결을 시도하며, 10초가 지나도록 연결이 되지 않으면 종료		
파라미터	In	ip	로봇 IP 어드레스 입력
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
<pre>char addr[25] = {"192.168.1.100"}; if(NWrite_EnableNetwork(addr) != SUCCESS)     exit(0);</pre>			



함수 원형	NRLAB_State NWrite_DisableNetwork();		
함수 설명	로봇과 연결을 종료하는 함수 현재 연결상태를 고려하지 않기 때문에 실패인 경우는 없음		
반환 값	성공	SUCCESS	
사용 예제 코드			
NWrite_DisableNetwork();			

함수 원형	NRLAB_State NRead_RobotDeviceNum(unsigned char &num)		
함수 설명	로봇 내부 모듈의 수를 읽는 함수로 연결된 EtherCAT Slave 모듈의 수를 반환 MRP-NRLAB02의 경우 3개의 모듈인 3의 값을 읽을 경우 정상		
파라미터	Out	num	현재 연결된 EtherCAT 모듈의 수
반환 값	성공	SUCCESS: 모듈의 수는 읽었지만 정상인지 에러상태인지는 알 수 없음	
	실패	CONNECTIONSTATE_FAULTED: 로봇과 연결되어 있지 않음	
		ETHERCAT_FAULTED: 로봇과는 연결되어 있지만 EtherCAT 통신 실패	
사용 예제 코드			
<pre>Unsigned char devNum = 0; if(NRead_RobotDeviceNum(devNum) != SUCCESS)     exit(0); else     printf("EtherCAT Slave Module : %d\n", devNum);</pre>			

함수 원형	NRLAB_State NRead_RobotDeviceState (unsigned char &state)		
함수 설명	로봇 내부 모듈의 상태 읽기 함수로 EtherCAT Slave 모듈의 상태를 반환 State의 값이 0일 경우 정상이며, 0 ~ 3bit의 값이 1일 경우 해당 모듈이 에러 상태임 1 – 첫 번째 모듈 에러 2 – 두 번째 모듈 에러 4 – 세 번째 모듈 에러		
파라미터	Out	state	현재 연결된 EtherCAT 모듈의 상태
반환 값	성공	SUCCESS: 상태는 읽었지만 정상인지 에러상태인지는 알 수 없음	
	실패	CONNECTIONSTATE_FAULTED: 로봇과 연결되어 있지 않음	
		ETHERCAT_FAULTED: 로봇과는 연결되어 있지만 EtherCAT 통신 실패	
사용 예제 코드			
<pre>Unsigned char devState = 0; if(NRead_RobotDeviceState (devState) != SUCCESS)     exit(0); if(devastate != 0)     printf("EtherCAT Slave Error: %d\n", devState);</pre>			

## 9.5. 모터 제어 및 상태 읽기

함수 원형	NRLAB_State MWrite_SetProfileAcceleration(int accel);		
함수 설명	모터의 반응속도를 설정하는 함수 값이 클수록 반응 속도가 빨라짐		
파라미터	In	accel	모터 가속도 : 1000 ~ 3000
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
		OUT_OF_RANGE	
사용 예제 코드			
if(MWrite_SetProfileAcceleration (2000) != SUCCESS) exit(0);			

함수 원형	NRLAB_State MRead_GetProfileAcceleration(int &accel);		
함수 설명	현재 설정된 모터의 가속도를 읽어오는 함수		
파라미터	Out	accel	현재 설정된 모터 가속도
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
int accel; if(MRead_GetProfileAcceleration (accel) != SUCCESS) exit(0);			



함수 원형	NRLAB_State MWrite_MotorVelocity (int vell, int velr);		
함수 설명	모터의 RPM을 직접 제어하는 함수 부호로 모터의 회전 방향이 결정됨		
파라미터	In	vell	왼쪽 모터 RPM 제어 값 -2500 ~ 2500
	In	velr	오른쪽 모터 RPM 제어 값 -2500 ~ 2500
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
		OUT_OF_RANGE	
사용 예제 코드			
if(MWrite_MotorVelocity (1000, -1000) != SUCCESS) exit(0);			

함수 원형	NRLAB_State MRead_MotorVelocity (int &vell, int &velr);		
함수 설명	현재 모터의 RPM을 읽어오는 함수		
파라미터	Out	vell	왼쪽 모터의 현재 RPM
	Out	velr	오른쪽 모터의 현재 RPM
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
int vell, velr; if(MRead_MotorVelocity(vell, velr) != SUCCESS) exit(0);			





함수 원형	NRLAB_State MRead_MotorPosition (int &posl, int &posr);		
함수 설명	현재 모터의 엔코더 펄스를 읽어오는 함수 - 8192 펄스 : 모터 1회전 - 8192 * 25(감속비) 펄스 : 휠 1회전		
파라미터	Out	posl	왼쪽 모터의 현재 엔코더 펄스
	Out	posr	오른쪽 모터의 현재 엔코더 펄스
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
int posl, posr; if(MRead_MotorPosition (posl, posr) != SUCCESS) exit(0);			

함수 원형	NRLAB_State MWrite_MotorStop ();		
함수 설명	모터 정지 함수		
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
if(NWrite_MotorStop() != SUCCESS) exit(0);			

## 9.6. 로봇 제어 및 상태 읽기

함수 원형	NRLAB_State RWrite_SetMaxLineSpeed (float maxls);		
함수 설명	로봇의 최대 선속도를 설정하는 함수 MWrite_MotorVelocity() 함수를 사용할 경우에는 무시됨		
파라미터	In	maxls	로봇의 최대 선속도 : 0.1 ~ 1.18 m/s
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
		OUT_OF_RANGE	
사용 예제 코드			
float maxlinespeed = 0.3; if(RWrite_SetMaxLineSpeed (maxlinespeed) != SUCCESS) exit(0);			

함수 원형	NRLAB_State RRead_GetMaxLineSpeed (float &maxls)		
함수 설명	현재 설정된 로봇의 최대 선속도 읽기		
파라미터	Out	maxls	현재 설정된 로봇의 최대 선속도 읽기
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
float maxlinespeed = 0; if(RWrite_GetMaxLineSpeed (maxlinespeed) == SUCCESS) printf("현재 최대 선속도 : %.2f m/s\n", maxlinespeed);			



함수 원형	NRLAB_State RWrite_2W_Kienmatics (float tv, float rv);		
함수 설명	모터가 아닌 로봇의 모션을 선속도와 각속도를 이용하여 제어하는 함수		
파라미터	In	tv	선속도 : 0.1 ~ MAX_LINE_SPEED m/s
	In	rv	각속도 : tv*2/TRED
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
		OUT_OF_RANGE	
사용 예제 코드			
float tv = 0.3f, rv = 1.2f; if(RWrite_2W_Kienmatics (tv, rv) != SUCCESS) exit(0);			

함수 원형	NRLAB_State RRead_2W_Kinematics (float &tv, float &rv);		
함수 설명	현재 로봇의 선속도와 각속도를 읽는 함수		
파라미터	Out	tv	로봇의 현재 선속도
	Out	rv	로봇의 현재 각속도
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
float tv = 0, rv = 0; if(RRead_2W_Kienmatics (tv, rv) == SUCCESS) printf("현재 로봇 선속도 : %.2f, 각속도 : %.2f \n",tv, rv);			



## 9.7. 센서 읽기

함수 원형	NRLAB_State SRead_SensorData (T_SENSOR_DATA &data);		
함수 설명	현재 로봇 센서 값 읽기 (단위 변환이 되어 있는 결과)		
파라미터	Out	data	센서 데이터 구조체
반환 값	성공	SUCCESS	
	실패	CONNECTIONSTATE_FAULTED	
사용 예제 코드			
<pre>T_SENSOR_DATA sensor; if(SRead_SensorData (sensor) != SUCCESS)     exit(0)</pre>			
<pre>//제어 라이브러리 헤더 파일 typedef struct {     unsigned short psd[7];    // [cm] psd distance     double x;                // Attitude Quaternion x     double y;                // Attitude Quaternion y     double z;                // Attitude Quaternion z     double w;                // Attitude Quaternion w     double ax;               // accel_x     double ay;               // accel_y     double az;               // accel_z     double gx;               // gyro_x     double gy;               // gyro_y     double gz;               // gyro_z     double mx;               // magnet_x     double my;               // magnet_y     double mz;               // magnet_z     double temp;             // temperature }T_SENSOR_DATA;</pre>			

