

FUNNERS

주니어 임베디드SW 챌린저

-1차 기술 지원 교육-

2017. 07. 08

- Python 개발 환경 구축

- 프로그래밍 기초

- 리눅스 명령어
- 변수의 개념
- 연산자

- 로봇 주행 기초

- 로봇 움직이기
- 회전하기

- 컬러 센서 활용

- 컬러 모드
- 반사광 모드

- 라인트레이싱

- 기본 라인트레이싱 방법
- 원하는 종료지점에서 멈추기

- 미션 해결 전략 세우기

• 구축 순서

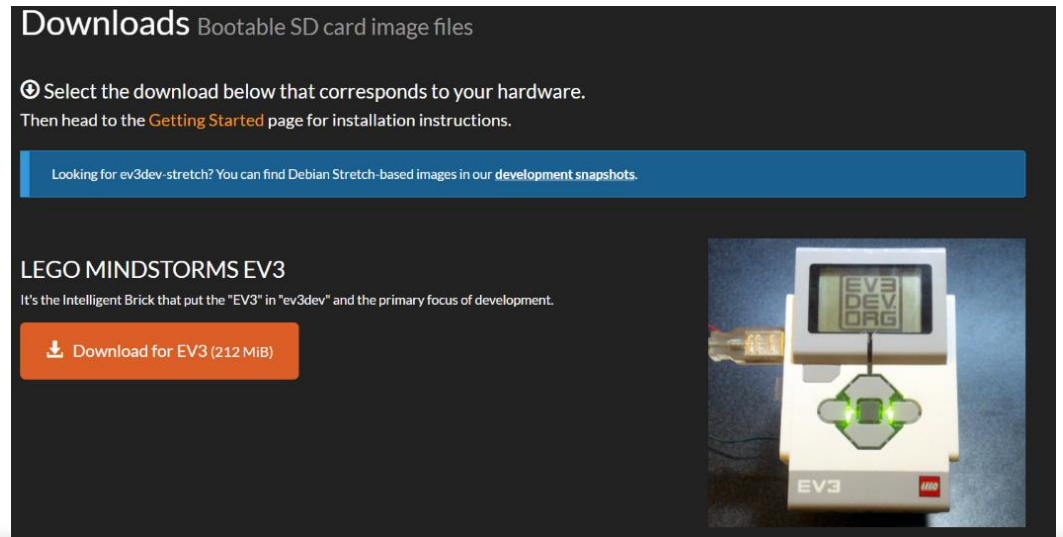
1. EV3Dev 이미지 파일 다운로드
2. 부팅 SD 카드 제작
3. EV3Dev 부팅
4. 네트워크 환경 설정
5. MobaXTerm 설치
6. SSH로 EV3와 연결

1. EV3Dev 이미지 파일 다운로드

EV3는 기본적으로 LME라는 펌웨어가 설치 되어있다.

Python을 이용해서 프로그래밍을 하려면 해당 이미지 파일을 받아 SD카드에 설치하고 SD카드를 이용하여 EV3를 실행해야 한다.

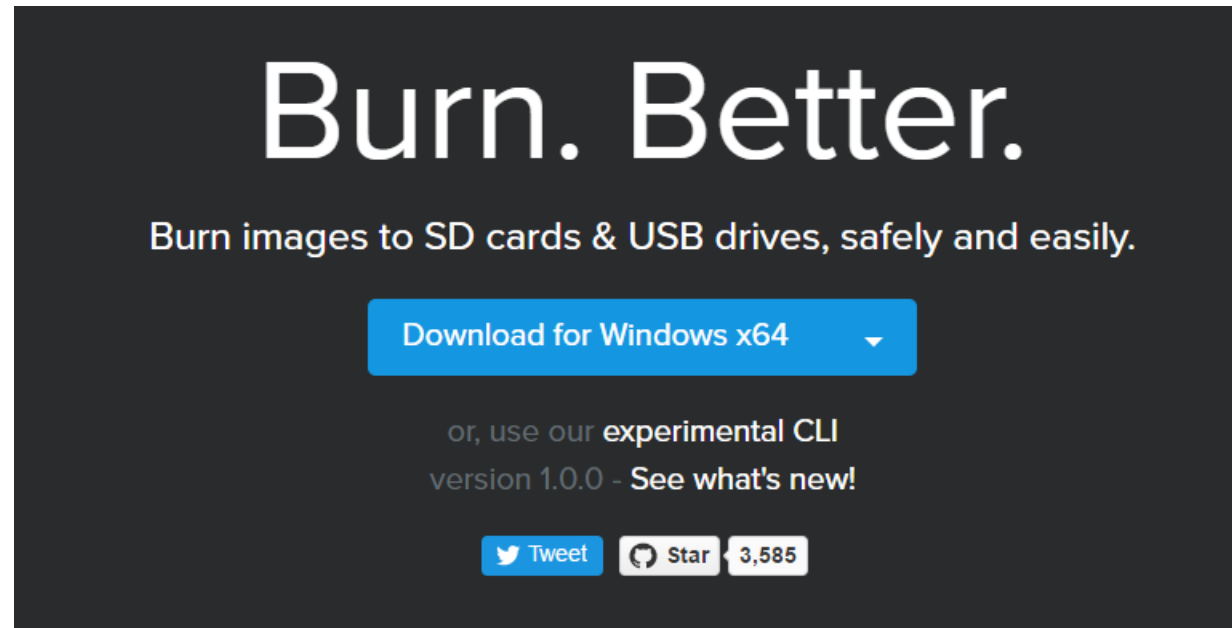
<http://www.ev3dev.org/download/>에 접속하여 EV3Dev를 다운로드한다.



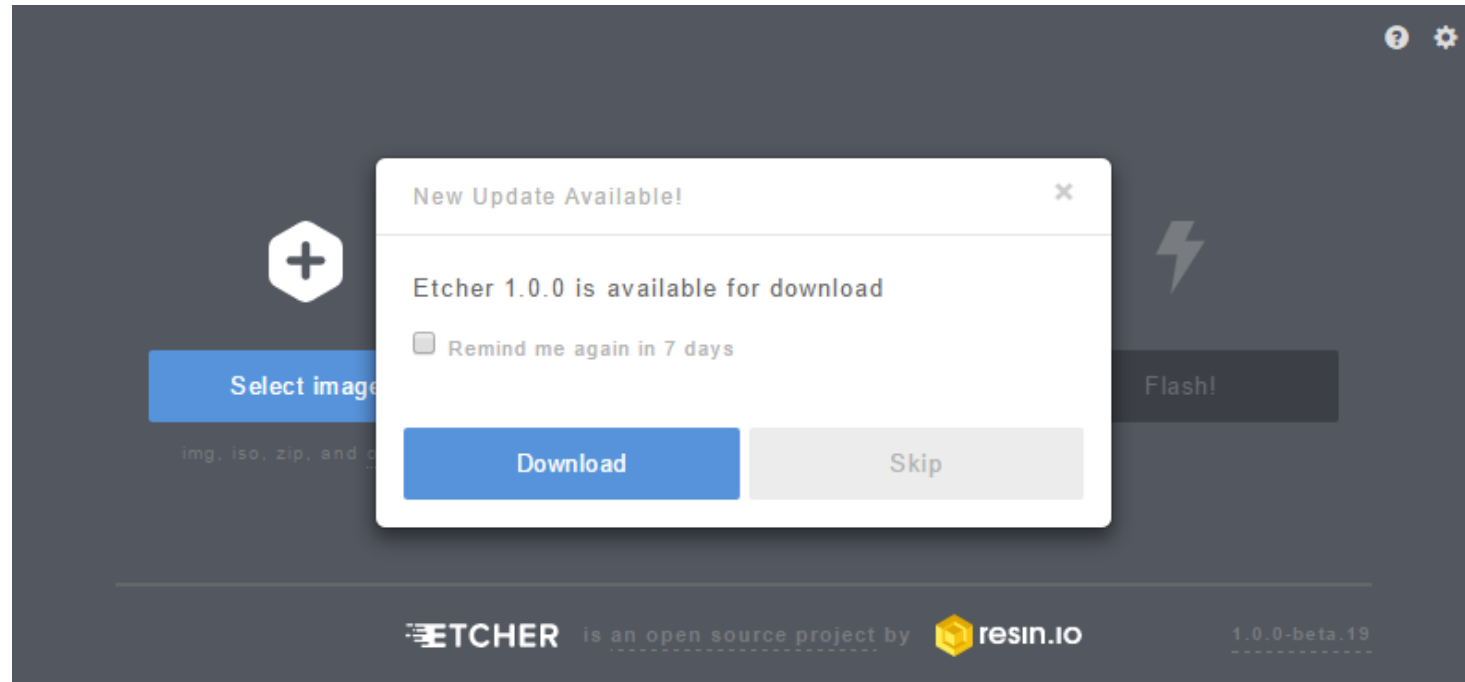
2. 부팅 SD카드 제작

부팅 SD카드를 제작하기 위해선 'Etcher'라는 프로그램을 설치해야한다.

- ① <https://etcher.io/>에 접속하여 다운로드한다.
- ② 다운로드된 Etcher-1.0.0-beta.19-win32-x64.exe를 실행하여 설치를 진행한다.

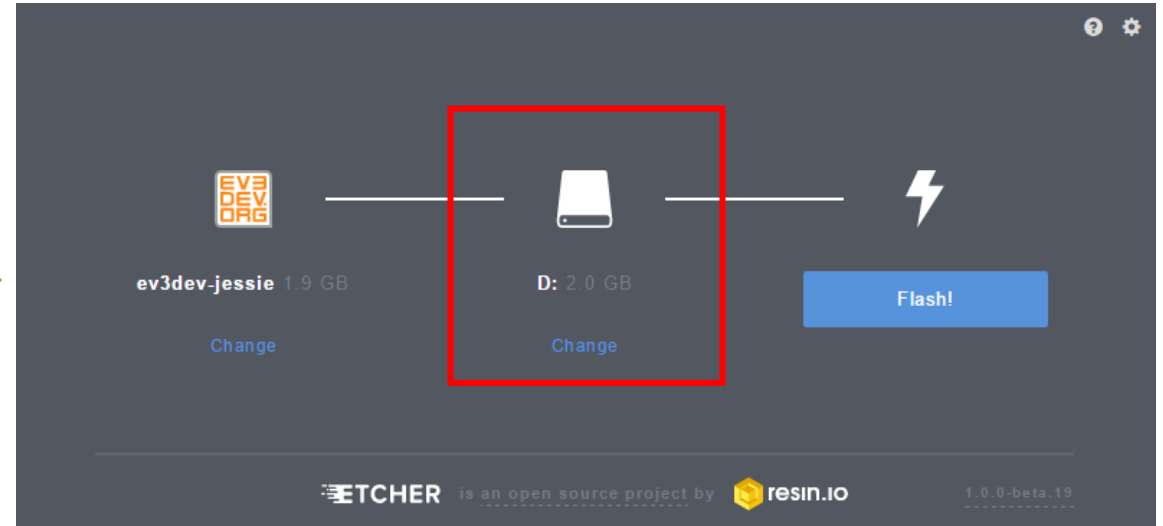
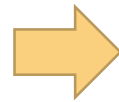
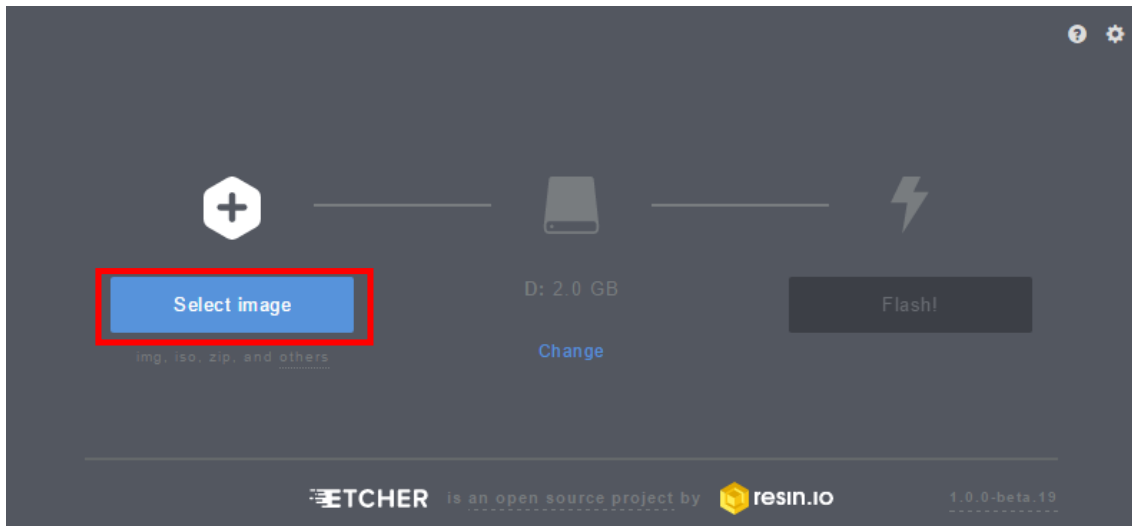


- ③ 설치가 완료된 'Etcher'를 실행한다.
- ④ 버전이 업데이트 되어 아래와 같은 창이 활성화 될 경우, Download를 클릭하여 새 버전을 다운로드하거나 Skip을 클릭하여 기존 버전을 계속 사용할 수 있다.

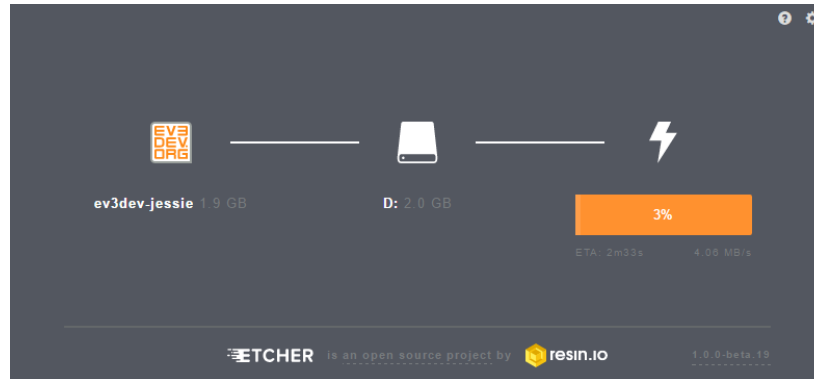
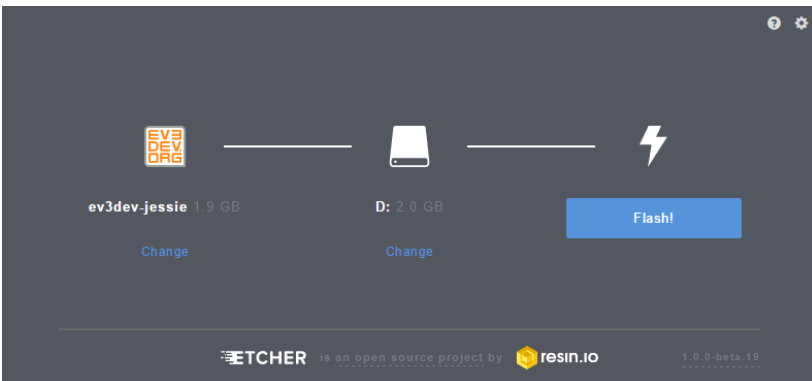
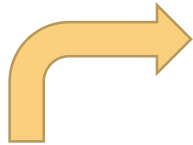


'Etcher'를 실행하였으면 아래와 같은 순서대로 진행한다.

- ① SD카드 리더기를 이용하여 SD카드(8Gb~32Gb)를 PC에 연결한다.
- ② Select image를 클릭하여 1에서 다운로드한 EV3Dev 압축 파일을 선택한다.
- ③ SD카드의 경로가 제대로 설정 되었는지 확인한다. 경로가 다를 경우 Change를 클릭하여 올바른 SD카드 경로를 설정해준다.



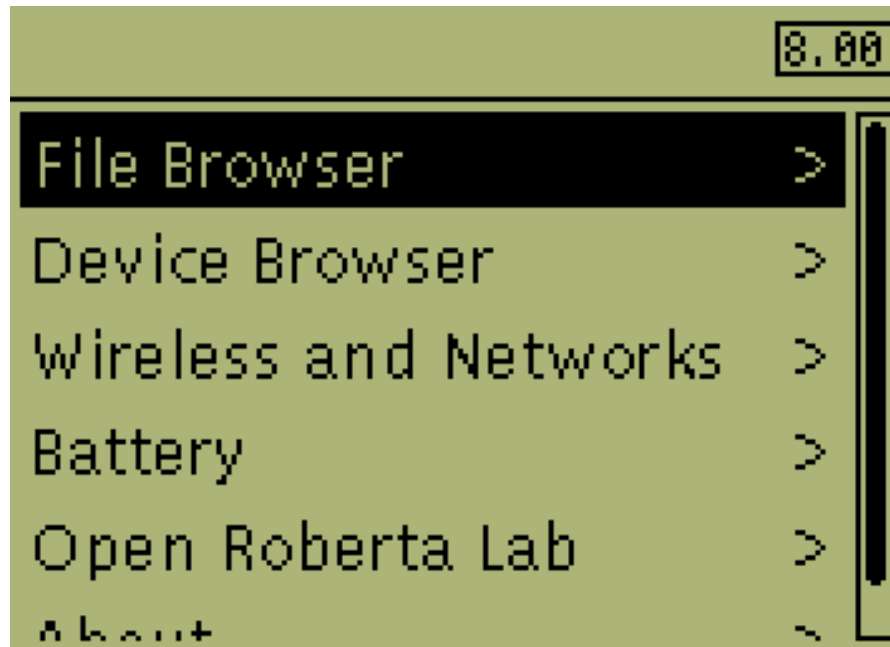
- ④ '경로를 설정하고 Flash!를 클릭한다.
- ⑤ 설치가 완료되면 Flash Complete!라는 화면으로 바뀐다.



3. EV3Dev 부팅

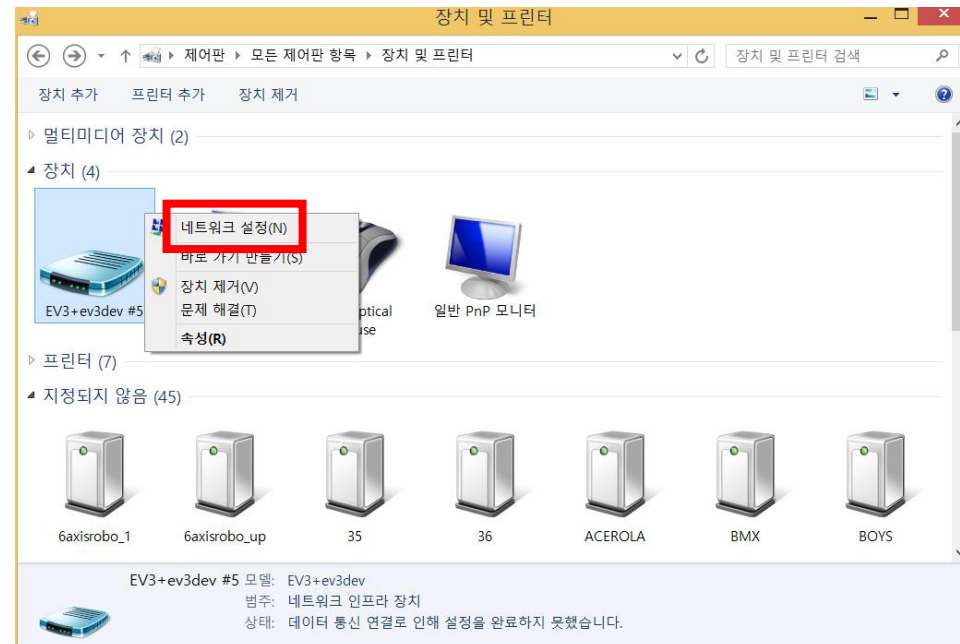
EV3 컨트롤러의 SD카드 슬롯에 만들어진 SD카드를 삽입하고, EV3 전원을 켜다.

부팅이 완료되면 아래와 같은 EV3 화면을 확인 할 수 있다.



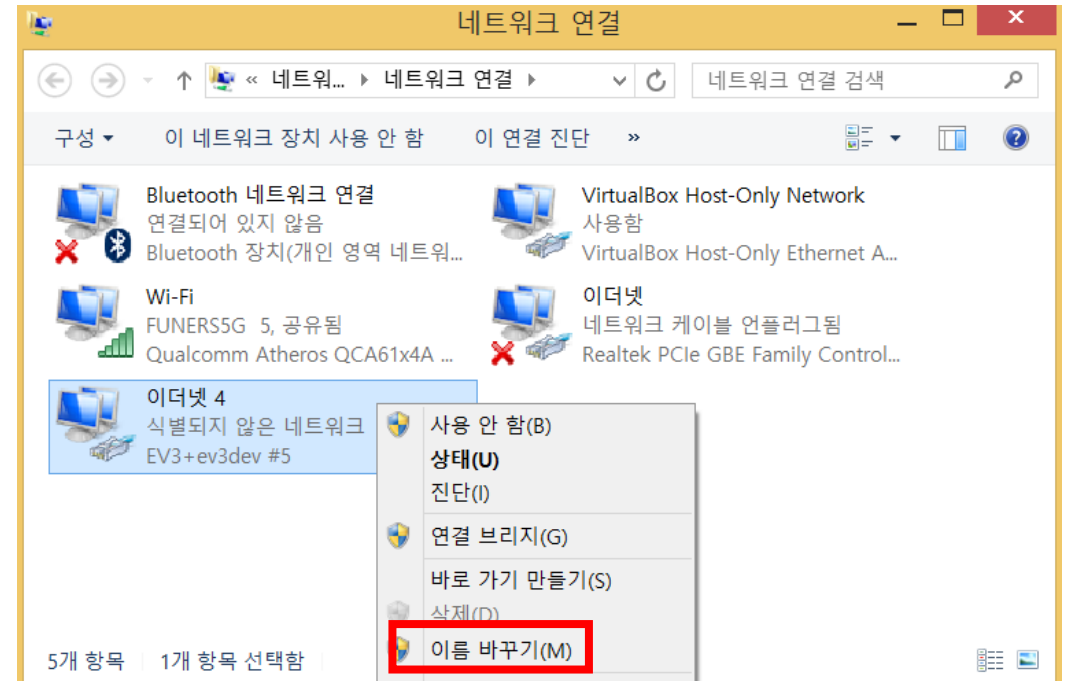
4. 네트워크 환경 설정

- ① 다운로드 케이블을 이용하여 EV3와 PC를 연결한다.
- ② 제어판 → 장치 및 프린터로 이동한다.
- ③ '장치'항목에 'USB Input Device' 또는 'EV3+ev3dev'를 우 클릭하여 '네트워크 설정'을 선택한다.

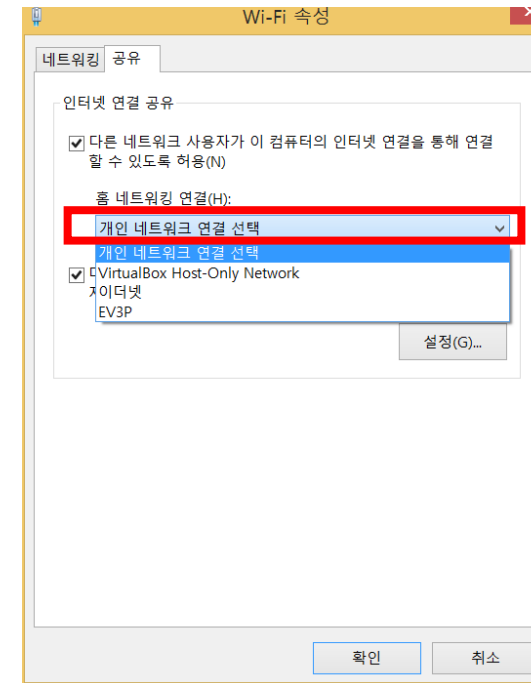
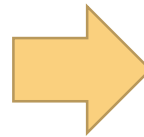
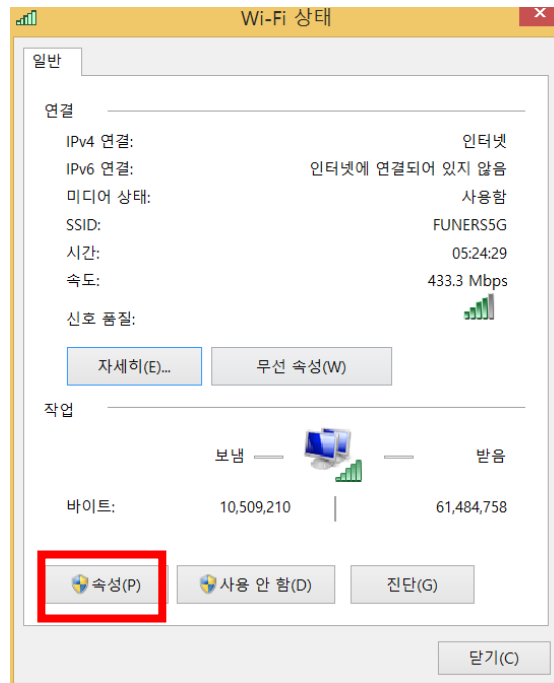


④ 좌측 '어댑터 설정 변경'을 선택한다.

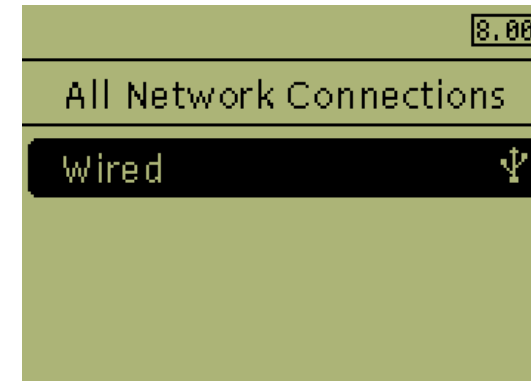
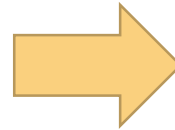
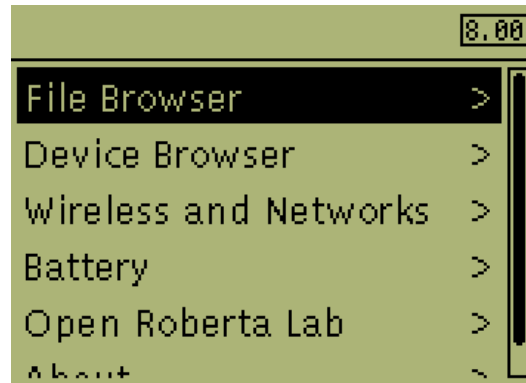
⑤ '네트워크 연결'에서 ③에서 나타난 이름을 가진 식별되지 않은 네트워크의 이름을 알아보
기 쉬운 이름으로 변경한다. (ex. EV3P)



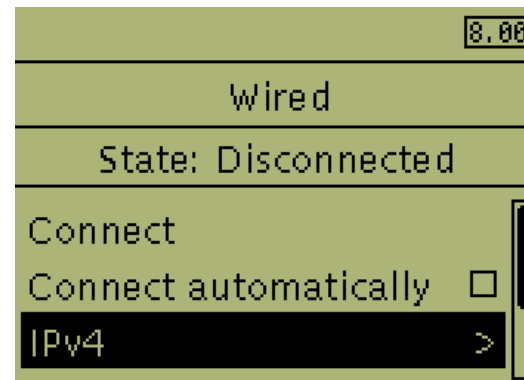
- ⑥ '네트워크 연결'에서 현재 연결된 인터넷(또는 WiFi)를 더블 클릭한다.
- ⑦ 'Wi-Fi 상태'창에서 속성을 클릭한다.
- ⑧ '공유'탭을 선택하고, '인터넷 연결 공유'를 두 곳 모두 체크 하고, '홈 네트워크 연결'에서 ⑤에서 설정한 이름을 선택한 후, 확인을 누른다.



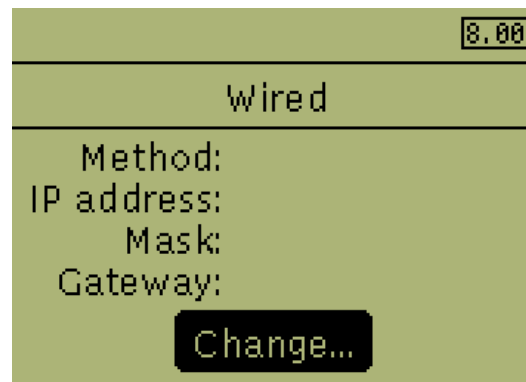
- ⑨ EV3 브릭에서 'Wireless and Networks' → 'All Network Connections' → 'Wired' 선택



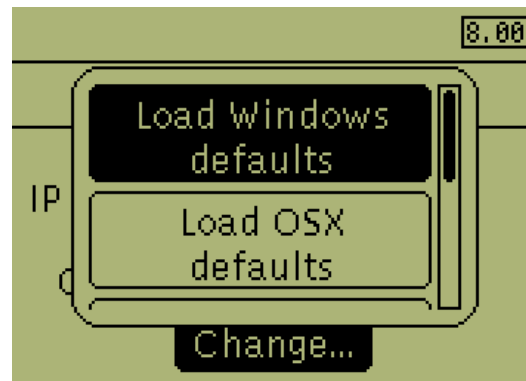
- ⑩ 'IPv4' 선택



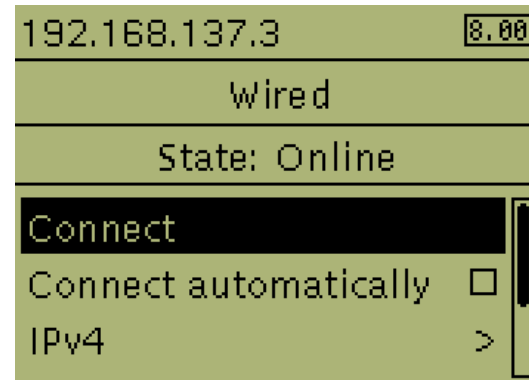
⑪ 'Change...' 선택



⑫ 'Load Windows defaults' 선택 후, EV3 브릭의 취소버튼을 이용하여 이전 화면으로 돌아감.



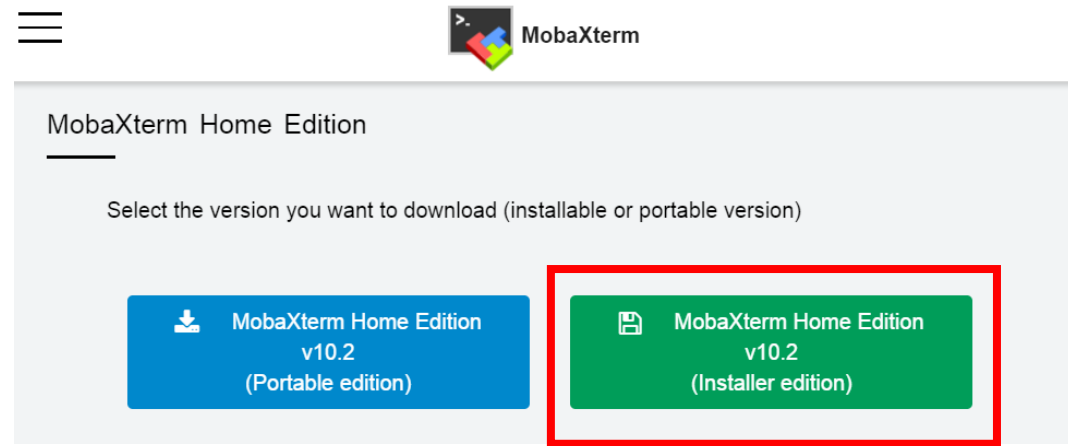
- ⑬ 'connect' 선택 후, EV3 브릭의 LCD 상단에 IP주소 생성 및 State : Online 상태 확인



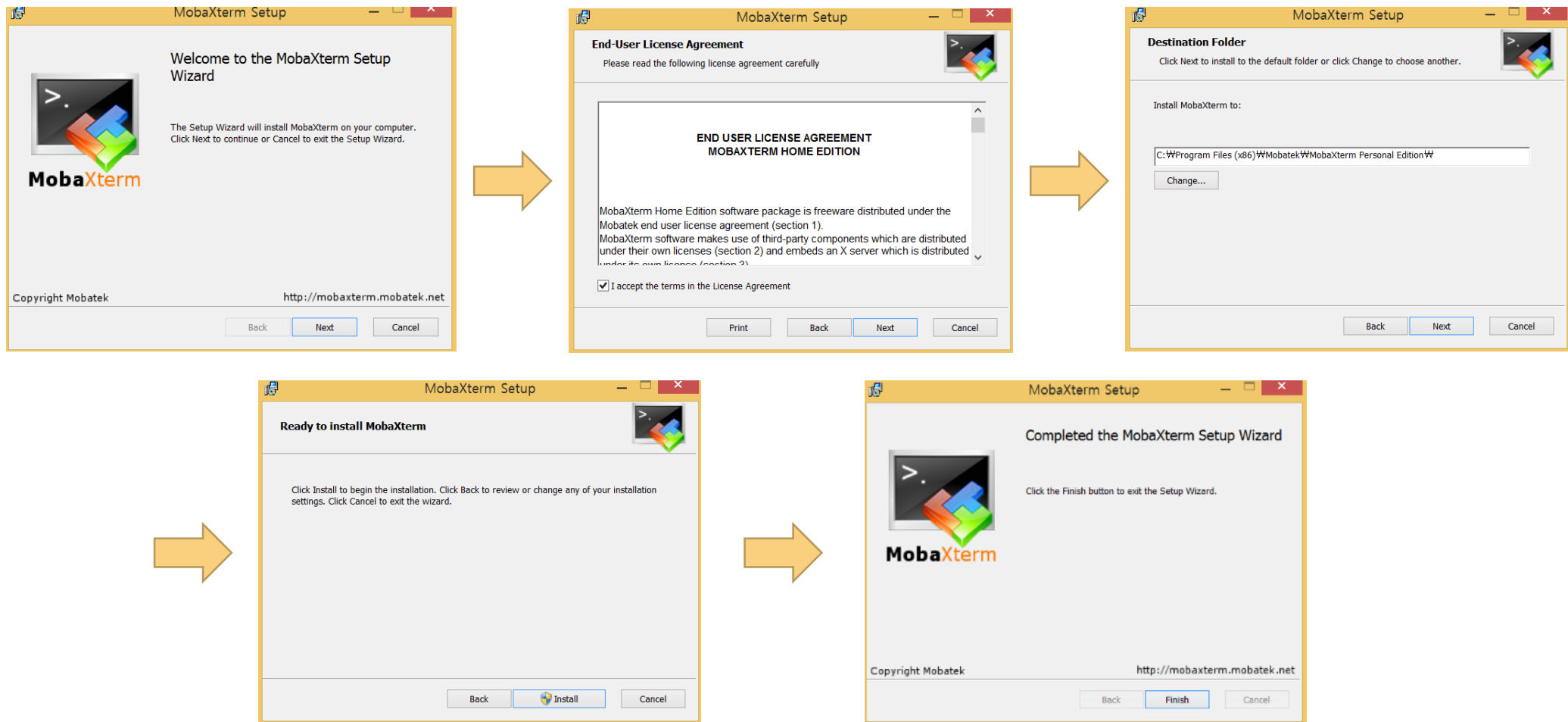
5. MobaXterm 설치

원격 작업 툴을 통해서 EV3에 프로그램을 주입해야한다. 여기서는 MobaXterm이라는 원격 작업 툴을 이용하여 진행한다.

- ① <http://mobaxterm.mobatek.net/download-home-edition.html>에 접속하여 다운로드 받는다.



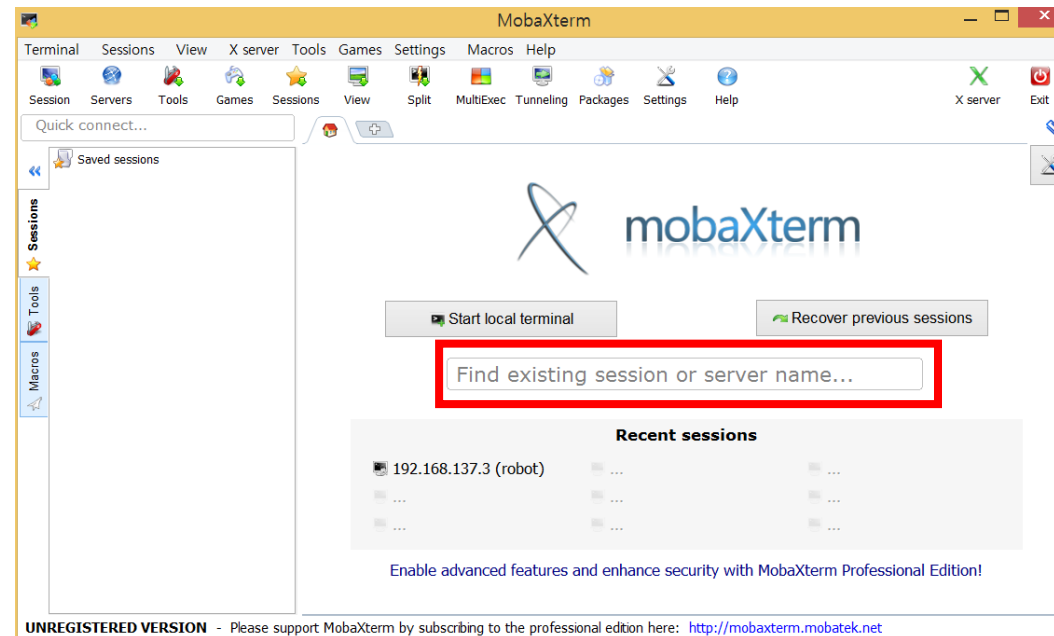
② 다운받은 설치파일을 아래 그림과 같은 순서로 설치를 진행한다.



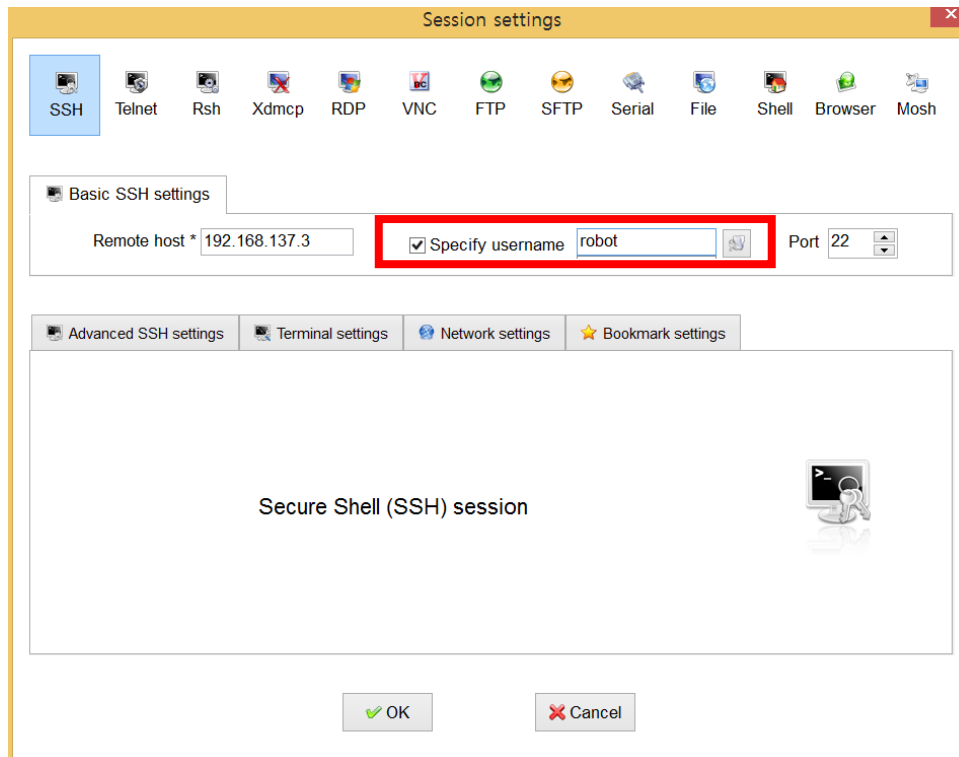
6. SSH로 EV3 연결

MobaXterm으로 EV3에 접근하기 위해 아래와 같이 진행한다.

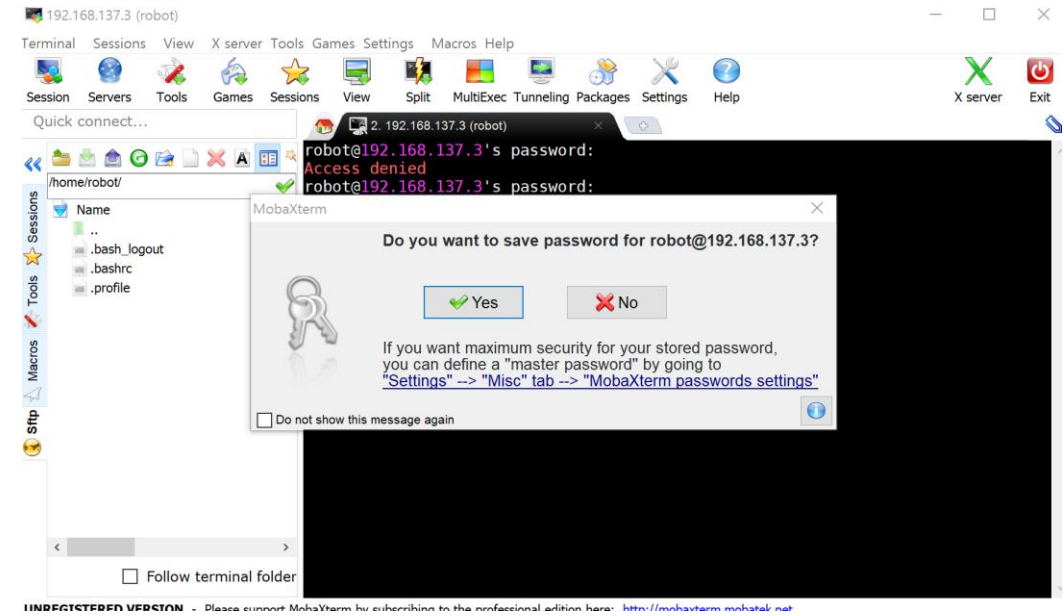
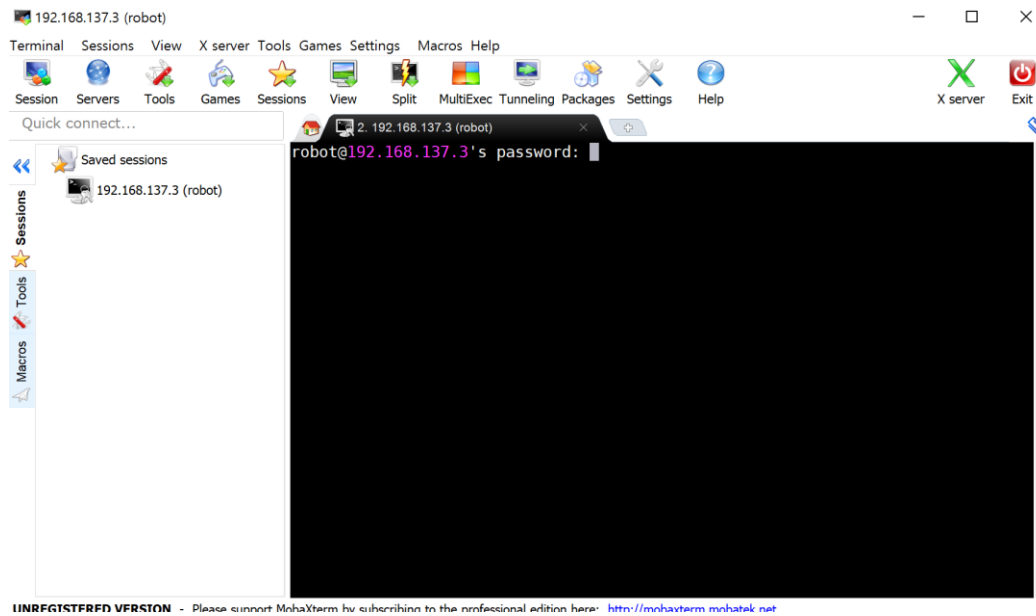
- ① MobaXterm 실행한 후, 'Find existing session or server name...'란에 EV3 LCD 화면 상단에 표시된 IP주소를 입력 후, Enter를 누른다.



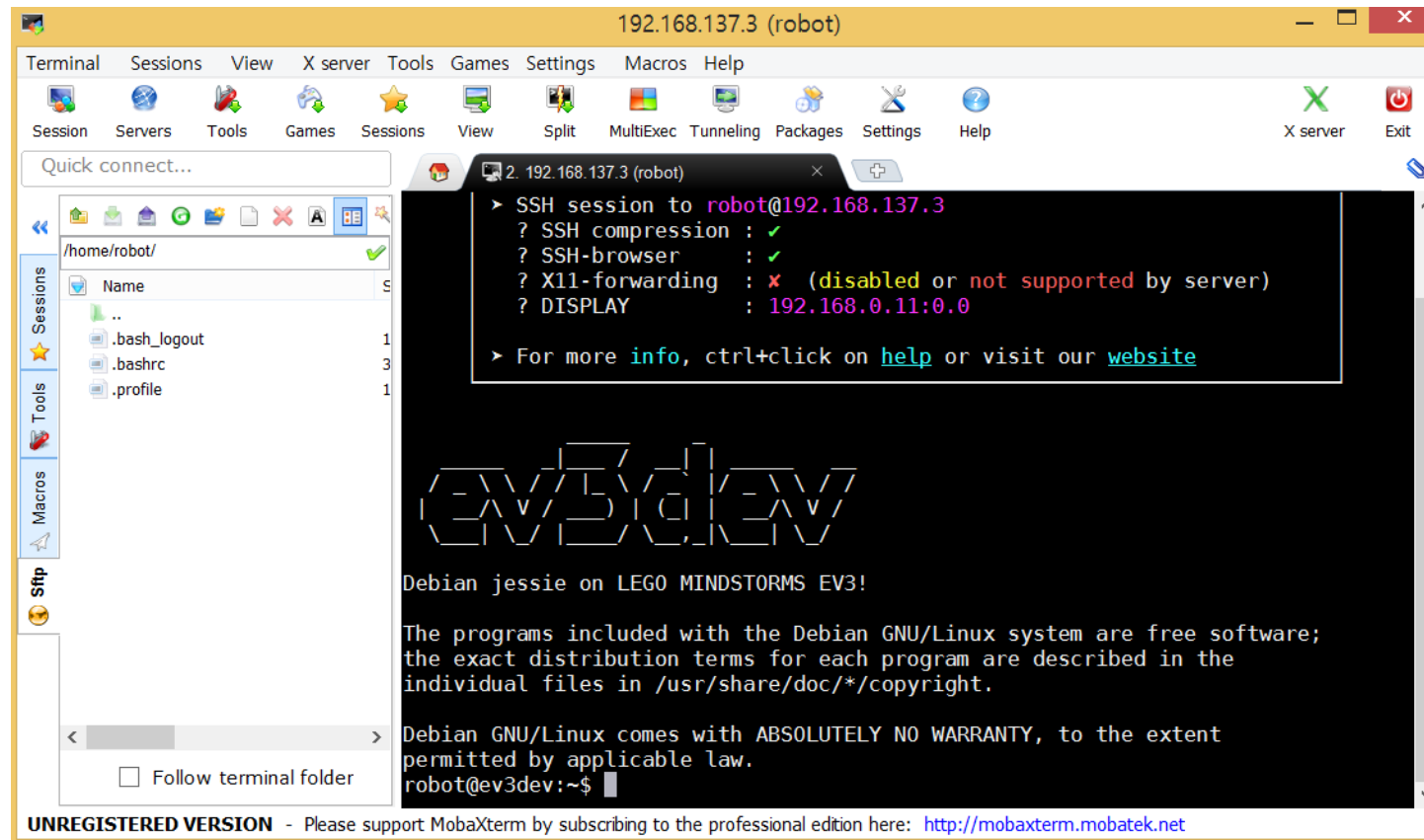
- ② 'Session settings'창이 자동으로 뜨면, 상단의 'SSH'를 클릭한 후, 'Specify username'을 체크한 후, robot을 입력하고 OK를 누른다. 만약 아래와 같은 경고창이 뜨면 Yes를 선택한다.



- ③ Password를 작성하는 창이 뜰 텐데, password는 "maker" 이다. "maker"를 입력하고 Enter를 누른 후, 'yes'를 클릭한다.



④ 아래와 같은 화면이 표시되면 정상적으로 연결된 것이다.



• 리눅스 명령어

- EV3dev는 리눅스 운영체제이기 때문에 프로그램을 편집 및 수정하기 위해서는 아래 표의 명령어를 알고 있어야 한다. 리눅스에는 다양한 명령어들이 있으나 이 과정에서는 프로그램을 작성하고, 편집할 때 필요한 필수 명령어만 다룬다.

명령어	동작
vi file_name	텍스트 파일 편집
:wq	파일을 저장하고 vi를 종료
:q	파일을 저장하지않고 vi를 종료
chmod	권한 변경 (r : 읽기 허용, w : 쓰기 허용, x : 실행 허용)
python3 file_name	파일 실행
ls -al	모든 파일 리스트 표시
rm file_name	파일 삭제

• 변수의 개념

- 데이터를 저장할 수 있는 메모리 공간이다.
- C언어, Java와는 달리 자료형에 상관없이 정수형, 문자열, 문자, 실수형을 알아서 인식한다.
- 변수 선언 시 유의 사항
 1. 변수명은 영문자(A-Z, a-z), 밑줄문자(_), 숫자를 조합하여 구성한다.
 2. 변수명의 첫글자는 반드시 영문자나 밑줄문자(_)로 시작해야 한다.
 3. 대소문자를 구분해서 사용해야 한다.
 4. 이미 다른 용도로 사용되고 있는 키워드 또는 예약어는 변수명으로 사용할 수 없다.

ex) LargeMotor, sleep 등

• 연산자

- 연산자를 활용하면 변수에 저장된 값을 다양하게 조작하여 사용할 수 있다.
- 프로그램 상에 괄호 없이 두 개 이상의 연산자를 한 문장에서 혼합하여 사용하게 되면, 연산자의 우선 순위에 따라 실행된다.

<산술 연산자>

연산자	의미	예시	연산 결과
+	덧셈	4 + 2	6
-	뺄셈	4 - 2	2
/	나눗셈	4 / 2	2
*	곱셈	4 * 2	8
%	나머지	4 % 2	0

<관계 연산자>

연산자	의미	예시	연산 결과
<	a가 b보다 작다	4 < 2	false
>	a가 B보다 크다	4 > 2	true
==	a와 b가 같다	4 == 2	false
!=	a와 b가 같지 않다	4 != 2	true
<=	a가 b보다 작거나 같다	4 <= 2	False
>=	a가 b보다 크거나 같다	4 >= 2	true

<대입 연산자>

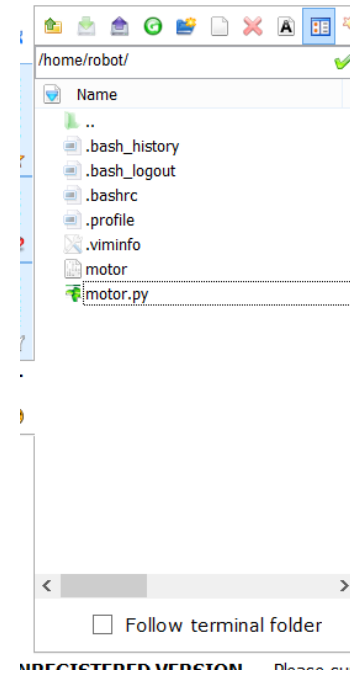
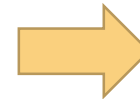
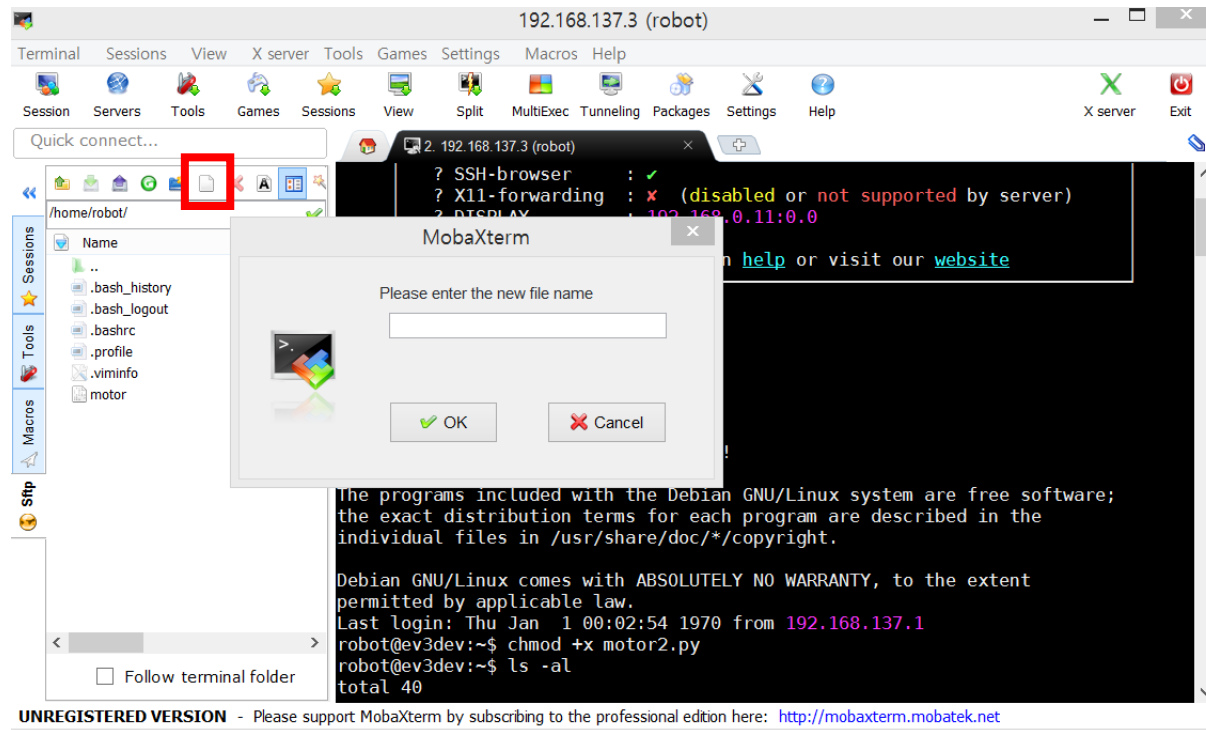
연산자	의미	예시	연산 결과
+=	덧셈 누적	a+=1	a=a+1
-=	뺄셈 누적	a-=1	a=a-1
/=	나눗셈 누적	a/=1	a=a/1
=	곱셈 누적	a=1	a=a*1

<논리 연산자>

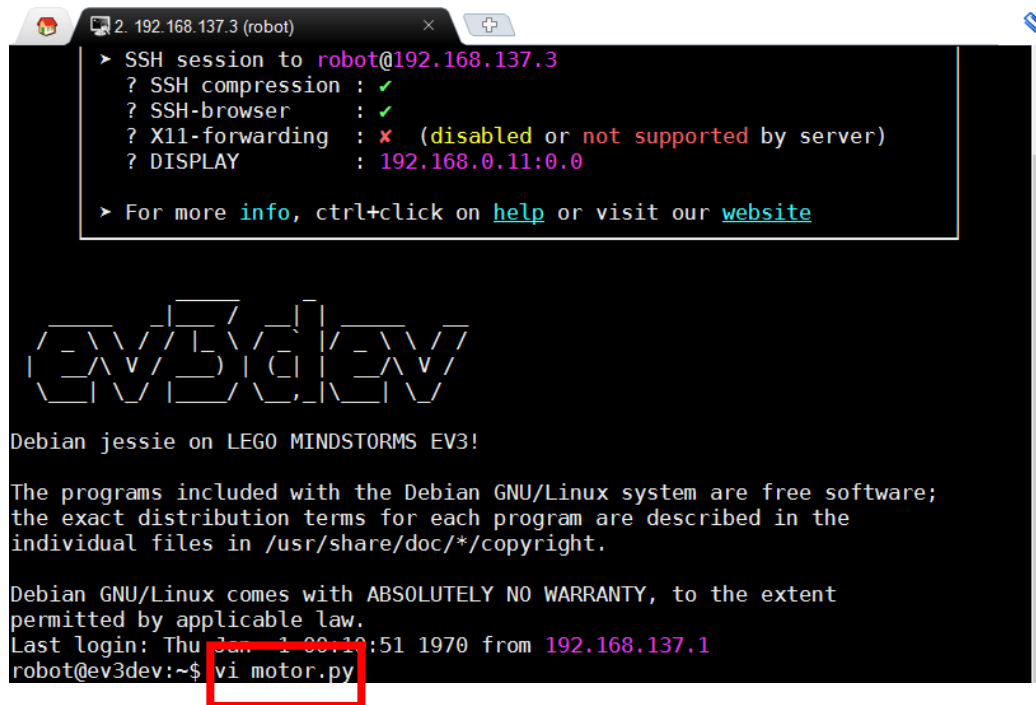
연산자	의미	예시	연산 결과
and	양쪽의 값이 모두 true인 경우 true	(2<4) and (5>8)	false
or	어느 한 쪽만 true인 경우 true	(2<4) or (5<8)	true
not	True면 false, false면 true	not(4<20)	false

• 프로그램 작성 방법

- ① MobaXterm의 왼쪽 SideBar에 Sftp항목에서 create new file을 클릭한다.
- ② File_name.py라는 형식으로 새로운 파일을 만들어준다. 예) motor.py



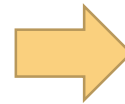
- ③ 이제 앞서 배운 명령어 중, vi(텍스트 파일 편집)를 이용하여 motor.py에 접근한다.



A terminal window titled '2. 192.168.137.3 (robot)' showing an SSH session. The output includes SSH status information, a 'robot@ev3dev:~\$' prompt, and the command 'vi motor.py' which is highlighted with a red box. The terminal also displays the 'robot@ev3dev:~\$' prompt and the command 'vi motor.py'.

```
> SSH session to robot@192.168.137.3
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✗ (disabled or not supported by server)
? DISPLAY          : 192.168.0.11:0.0
> For more info, ctrl+click on help or visit our website
```

robot@ev3dev:~\$ vi motor.py

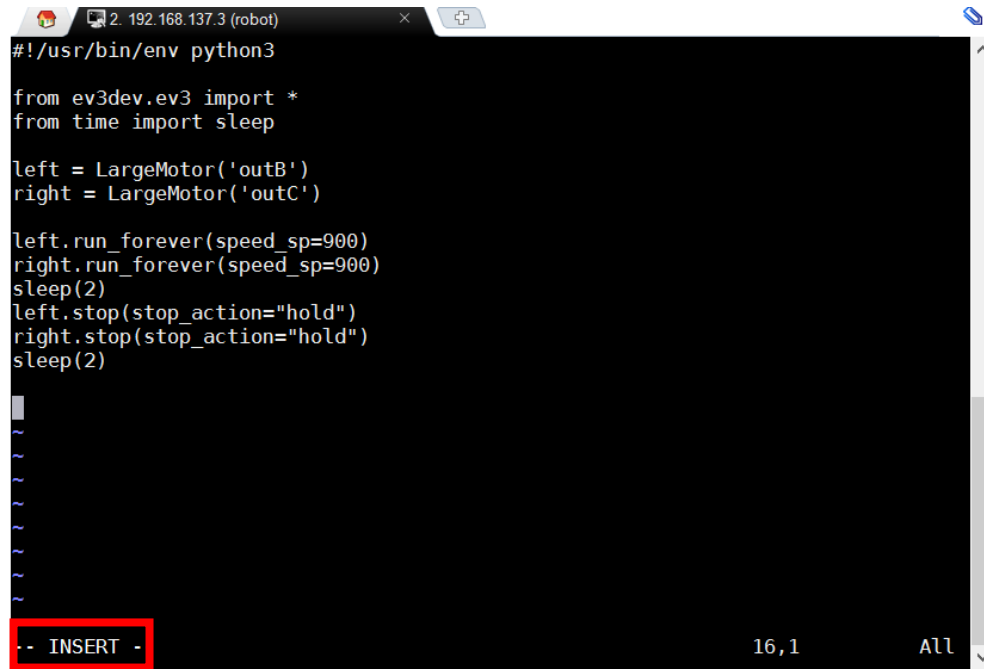


A terminal window titled '2. 192.168.137.3 (robot)' showing the vi editor interface. The editor is in normal mode, displaying the file 'motor.py' with line numbers 0, 0-1 and the cursor at the end of the first line. The status bar at the bottom shows 'port MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net'.

```
"motor.py" 0L, 0C
0,0-1 ALL
```

port MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

- ④ 키보드의 insert키(Ins)를 눌러 커서를 나타낸 후, 코드를 작성한다.
- ⑤ 작성이 완료되면, 키보드의 Esc버튼을 눌러 INSERT 작업을 마치고 :wq 명령어를 이용하여 저장한 후, 메인 화면으로 이동한다.

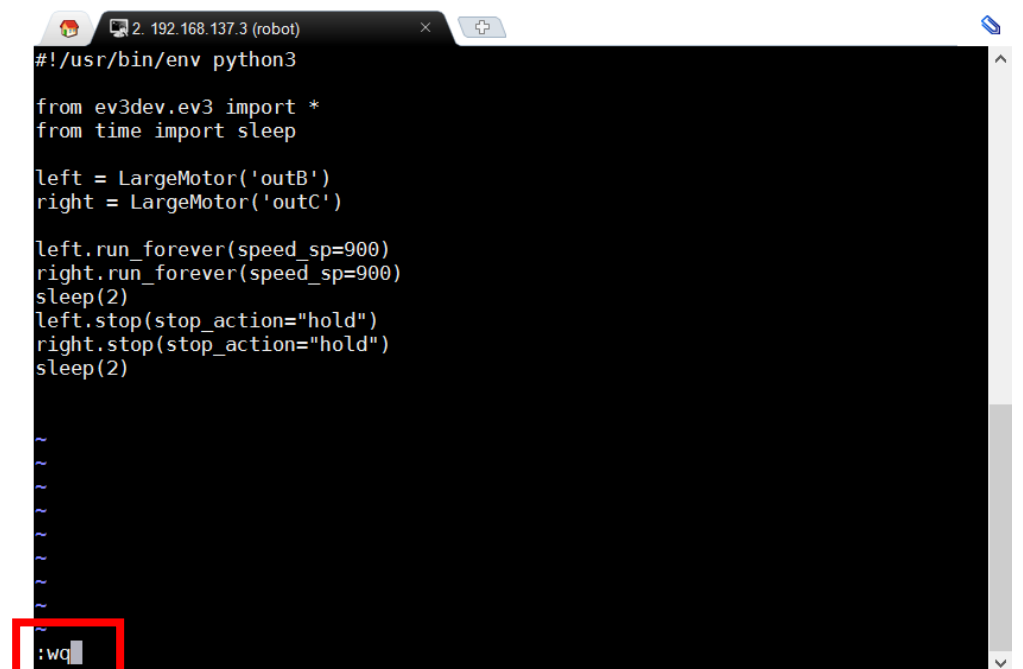


```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)

-- INSERT -
```



```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

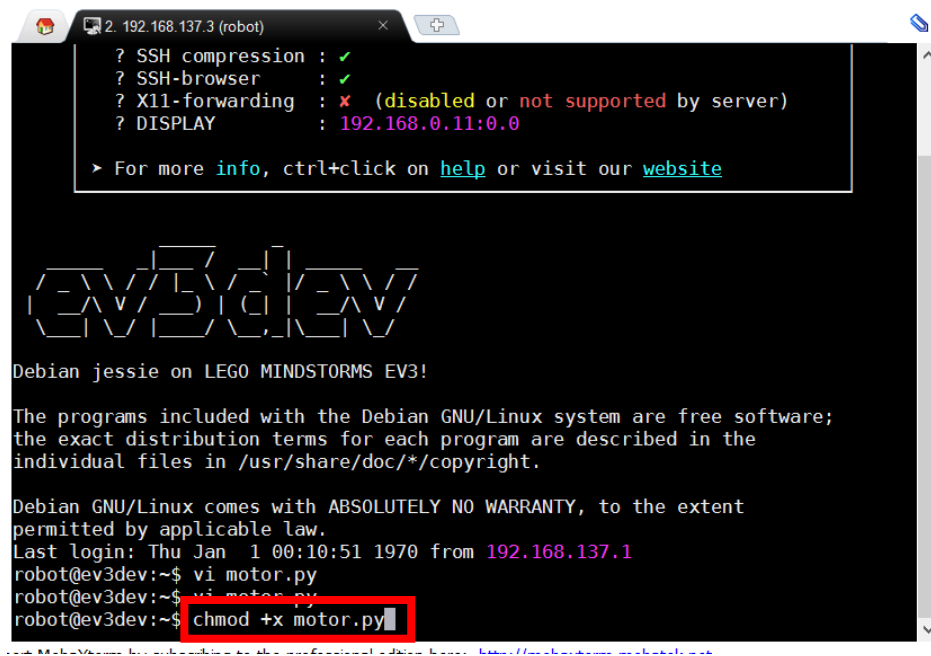
left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)

:wq
```

- ⑥ 메인화면에서 chmod 명령어를 이용하여 motor.py 파일을 실행 파일로 설정해준다.

chmod +x motor.py 입력 후, Enter를 누른다.

- ⑦ 이제 EV3 LCD화면에서 File Browser에 들어가서 만들어진 motor.py*파일을 실행시킨다.

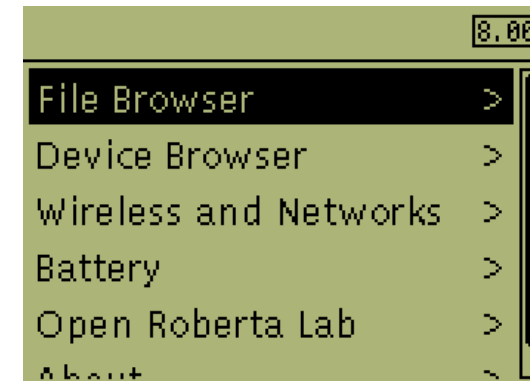


```
2. 192.168.137.3 (robot) x
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✗ (disabled or not supported by server)
? DISPLAY          : 192.168.0.11:0.0
> For more info, ctrl+click on help or visit our website

ev3dev
Debian jessie on LEGO MINDSTORMS EV3!

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan  1 00:10:51 1970 from 192.168.137.1
robot@ev3dev:~$ vi motor.py
robot@ev3dev:~$ vi motor.py
robot@ev3dev:~$ chmod +x motor.py
```



• 로봇 움직이기

- 프로그램 작성 방법에서 실행해 보았던 코드에 사용된 함수에 대해 알아본다.
- 기본 설정
 - `#!/usr/bin/env python3` : EV3에서 프로그램을 실행 시키기 위해서는 반드시 제일 첫 행에 입력해야한다.
 - `from ev3dev.ev3 import *` : EV3를 사용하기 위한 모든 public 속성/함수를 사용하기 위해 반드시 필요하다. 모든 프로그램의 첫 행에 입력한다.
 - `from time import sleep` : time 관련 함수를 사용하는 경우에 반드시 필요하다. sleep 함수를 사용하기 위해서 time 라이브러리의 sleep 함수를 추가한다.

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)
```

- 모터 구동 함수 및 Time 함수

- `m = LargeMotor('outX')` : 해당 출력 포트를 라지 모터 변수 `m`으로 설정한다.

Ex) `left.LargeMotor('outB')`

- `m.run_forever(speed_sp=000)` : 모터를 원하는 스피드(`speed_sp`)로 다른 명령이 있을 때 까지 동작 시킴. Ex) `left.run_forever(speed_sp=900)` (입력 가능한 범위 : -1050~1050)
- `m.stop(stop_action="xxx")` : 모터를 원하는 정지 방법(`stop_action`)으로 정지시킴.

"brake" : 모터에 전원이 제거되었다가 다시 전원이 들어옴.

"hold" : 모터에 전원이 제거되지 않고 현재 위치를 유지하려고 붙잡고 있음.

"coast" : 모터에 전원이 제거되어 자연스럽게 멈춤.

Ex) `left.stop(stop_action="brake")`

- `sleep(0)` : 원하는 시간(초) 동안 대기한다.

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)
```

- 다양한 모터 구동 함수

- `run_timed(speed_sp=000, time_sp=000, stop_action="xxx")` : 모터를 원하는 스피드(speed_sp)로, 원하는 시간(time_sp)만큼(ms) 움직여 원하는 정지방법(stop_action)으로 정지시킴.

Ex) `left.run_timed(speed_sp=500, time_sp=1000, stop_action="hold")`

- `run_to_abs_pos(speed_sp=000, position_sp=000, stop_action="xxx")` : 모터를 원하는 스피드(speed_sp)로, **지정된 위치(절대 위치)를 기준**으로 원하는 각도(position_sp)만큼 움직여 원하는 정지 방법(stop_action)으로 정지시킴. 사용하려면 먼저 position_sp 값을 0으로 실행하여 지정된 위치(절대 위치)를 파악해야함.

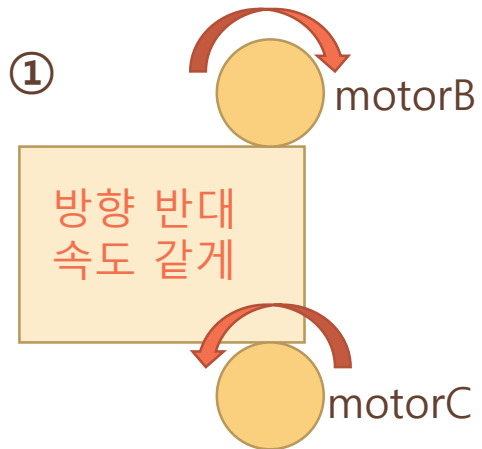
Ex) `left.run_to_abs_pos(speed_sp=500, position_sp=360, stop_action="hold")`

- `run_to_rel_pos(speed_sp=000, position_sp=000, stop_action="xxx")` : 모터를 원하는 스피드(speed_sp)로, **현재 위치를 기준**으로 원하는 각도(position_sp)만큼 움직여 원하는 정지방법(stop_action)으로 정지시킴.

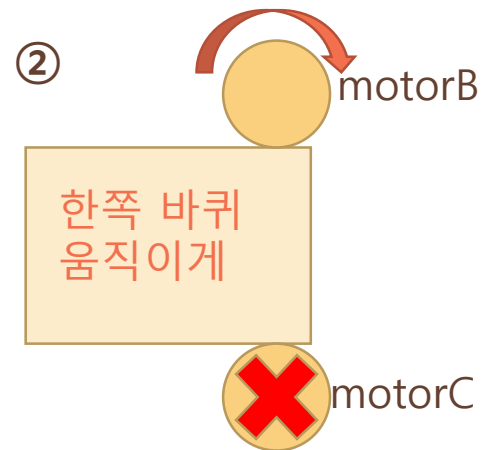
Ex) `left.run_to_rel_pos(speed_sp=500, position_sp=360, stop_action="hold")`

• 회전하기

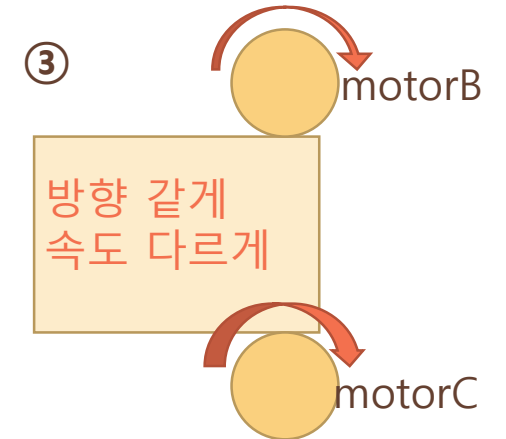
- 로봇을 회전시키는 방법



<point turn>



<swing turn>



<curve turn>

• 회전하기

- 로봇을 회전시키는 방법

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=300)
right.run_forever(speed_sp=-300)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(0.1)
```

<point turn>

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=300)
sleep(2)
left.stop(stop_action="hold")
sleep(0.1)
```

<swing turn>

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

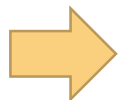
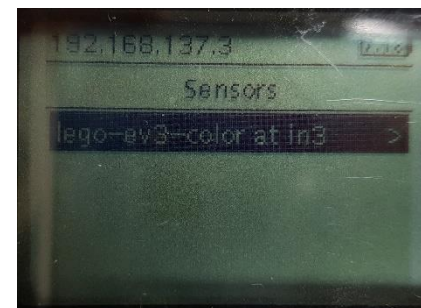
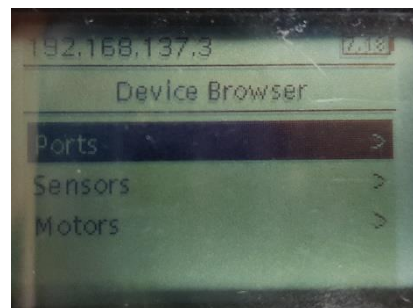
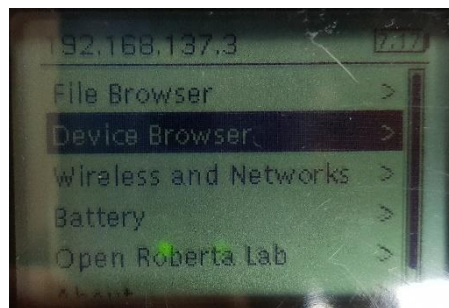
left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=300)
right.run_forever(speed_sp=100)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(0.2)
```

<curve turn>

• 컬러 센서 활용

- 컬러 센서로 색상의 값을 측정하는 방법.



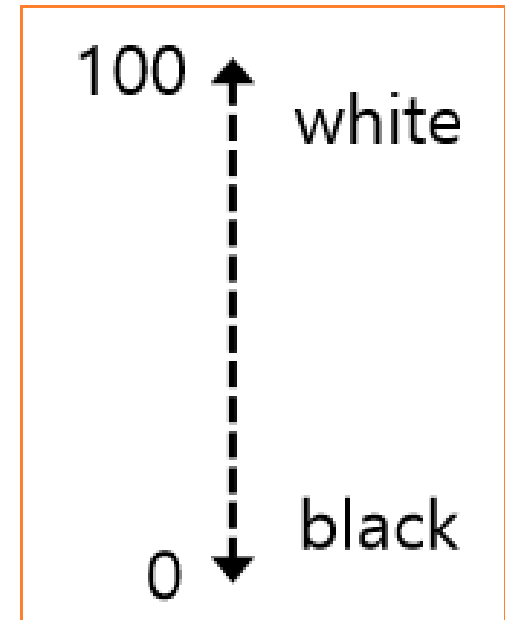
- 컬러 모드(COL-COLOR)

- 컬러 센서가 감지하는 표면의 색상에 따라 다른 숫자값이 반환된다.

숫자	색상	숫자	색상	숫자	색상	숫자	색상
0	None	1	Black	2	Blue	3	Green
4	Yellow	5	Red	6	White	7	Brown

• 반사광 모드(COL-REFLECT)

- 컬러 센서를 반사광 모드로 작동시키면, 센서는 현재 측정하고 있는 표면이 빛에 반사되는 정도를 감지하여 측정값을 0~100사이의 값으로 반환한다.
- 어두운 영역과 밝은 영역을 구분할 때는 두 영역의 경계로 판단할 수 있는 **문턱값(threshold)**을 정해야 한다.
 - 가장 일반적으로 문턱값을 정할 때는 **두 표면의 빛 값을 측정한 후 평균값을 계산하는 방법**을 활용한다.



- ColorSensor 함수

- `x = ColorSensor('in0')` : 컬러 센서가 연결되어 있는 입력 포트를 변수 `x`로 설정한다.

Ex) `Color=ColorSensor('in3')`

- `x.mode='COL-xxx'` : 컬러 센서를 원하는 모드로 설정한다.

'REFLECT' : 반사광 모드

'COLOR' : 컬러 모드

'AMBIENT' : 주변광 모드

Ex) `Color.mode='COL-REFLECT'`

- `x.value()` : 컬러 센서가 현재 읽고 있는 값을 의미한다.

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

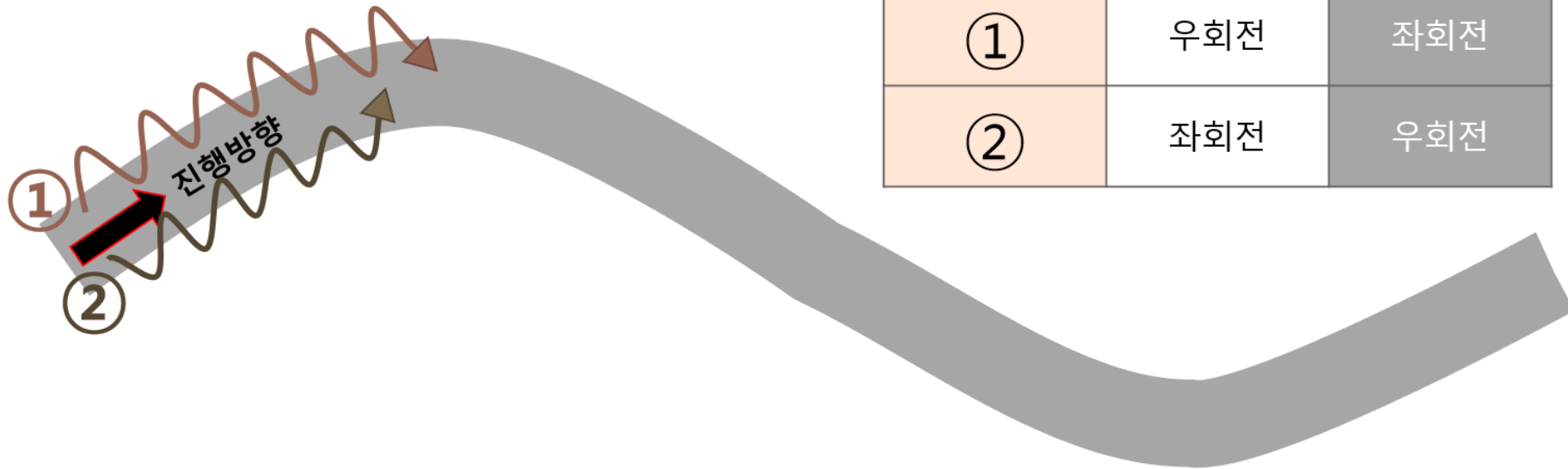
color = ColorSensor('in3')

color.mode='COL-REFLECT'
i=0

while i<10:
    print(color.value())
    sleep(0.5)
    i+=1
```

• 라인트레이싱

- 컬러 센서를 활용한 라인트레이싱



	white	grey
①	우회전	좌회전
②	좌회전	우회전

- Button 함수

- `b = Button()` : 버튼을 변수 `b`로 설정한다.

ex) `btn=Button()`

- `b.xxx` : 버튼의 상태를 설정한다.

상태 : up, down, left, right, enter, backspace

Ex) `btn.backspace`

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

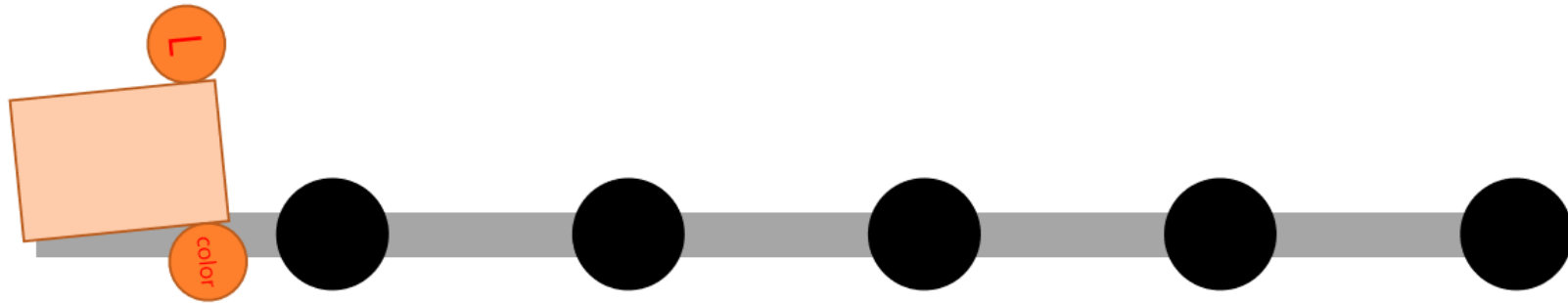
color = ColorSensor('in3')
color.mode='COL-REFLECT'

btn = Button()

while not btn.backspace:
    if color.value() > 50:
        left.run_forever(speed_sp=100)
        right.run_forever(speed_sp=200)

    else:
        left.run_forever(speed_sp=200)
        right.run_forever(speed_sp=100)
```


• 컬러 센서를 활용한 라인트레이싱 < 원하는 종료 지점에서 멈추기 >



- 로봇이 다섯 번째 검정색 원을 만날 때까지 라인트레이싱을 진행하려면 어떻게 해야할까?
 - 라인트레이싱 알고리즘에 로봇이 검정색 원을 인식하는 부분이 추가되어야 한다.
 - 검정색 원을 인식한 후에 다시 라인을 찾아가는 알고리즘이 필요하다.
 - 라인트레이싱의 종료 조건이 필요하다.

• 미션 해결 전략 세우기(예시)

- 맵의 색상정보 읽기(mapping)

- ① 효과적인 라인트레이싱 방법 개발하기
- ② 교차로에서 좌회전, 우회전, 유턴, 직진 반복적인 동작 개발하기
- ③ 색상 정보 읽어서 배열에 넣기

- 읽은 색상 정보로 알고리즘 개발하기

- ① 최단거리 알고리즘 개발하기
- ② 색상 정보가 없는 곳의 좌표 확인하기
- ③ 색상 정보가 없는 곳의 색상 알아내기

- 올바른 택배 상자를 정확하게 가져다 놓기

- ① 원하는 좌표로 이동하는 알고리즘 개발하기
- ② 택배 상자를 놓는 기계 구조 및 프로그램 개발하기