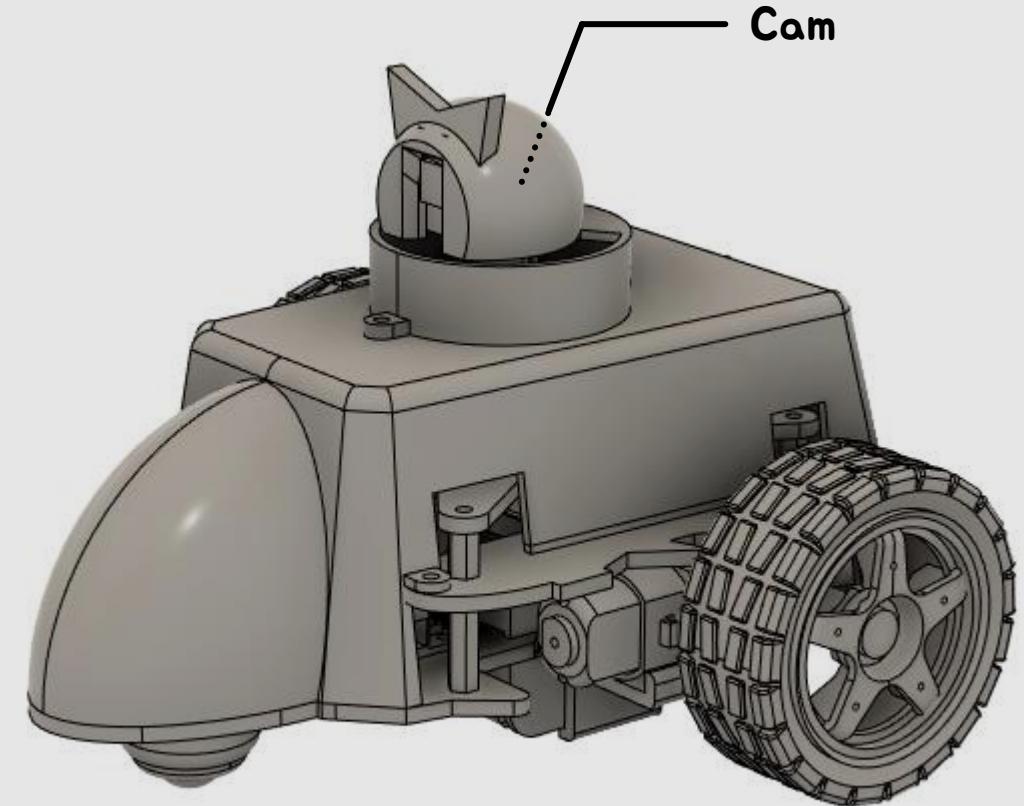


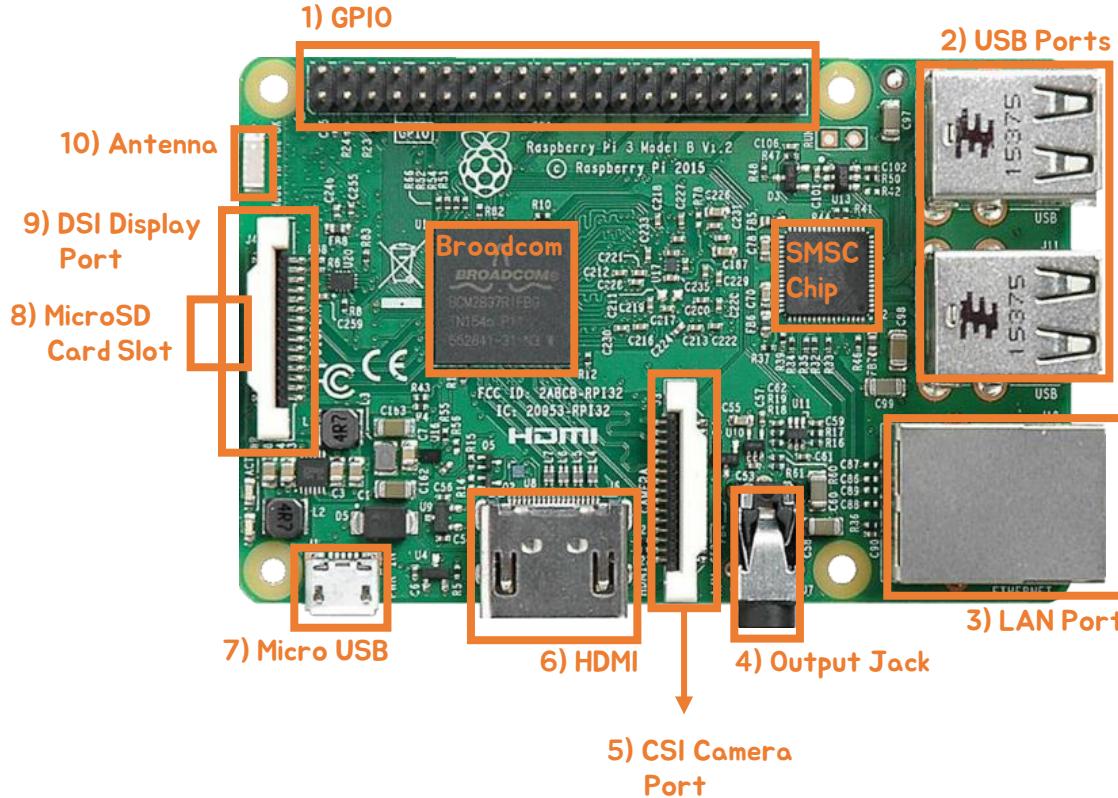
Raspberry Pi IoT CCTV RC Car



Intro

라즈베리파이란?

2012년 2월 영국의 라즈베리파이 재단에서 교육적 목적으로 만든 싱글 보드 컴퓨터(SBC)



라즈베리파이 3B Diagram

- 1) **GPIO**: 디지털 값의 입력/출력에 사용
- 2) **USB Ports**: USB 장치를 연결할 수 있는 포트
- 3) **LAN Port**: 이더넷 포트
- 4) **Output Jack**: 오디오 연결 단자
- 5) **CSI**: 카메라 연결 인터페이스
- 6) **HDMI**: HDMI 단자
- 7) **Micro USB**: 전원 단자
- 8) **Micro SD Card Slot**: 마이크로 SD카드 장착 슬롯
- 9) **DSI**: 디스플레이 장치 인터페이스
- 10) **Antenna**: 와이파이 안테나

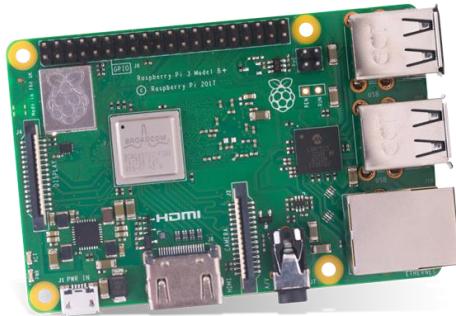
Intro

라즈베리파이 SPEC

모델명	4 MODEL B	3 MODEL B+	3 MODEL B	ZERO
출시일	19.06.24	18.03.14	16.01.29	15.11.25
SOC	Broadcom BCM2711	BCM2837B0	BCM2837	BCM2835
CPU	Quad core Cortex-A72(ARM v8) 64-bit SoC @1.5GHz	Quad Cortex-A53 @1.4GHz	Quad Cortex-A53 @1.2GHz	ARM11 @1GHz
RAM	1GB, 2GB, 4GB, 8GB LPDDR4-3200 SDRAM (depending on model)	1GB SDRAM	1GB SDRAM	512MB SDRAM
Storage	Micro-SD			
Ethernet	Gigabit Ethernet	Gigabit	10/100	None
Wireless	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0 BLE	WLAN/Bluetooth 4.2	WLAN/Bluetooth 4.0	None ZERO W의 경우 Wireless 지원
GPIO	40	40	40	40

Intro

준비물



라즈베리파이 3B 이상



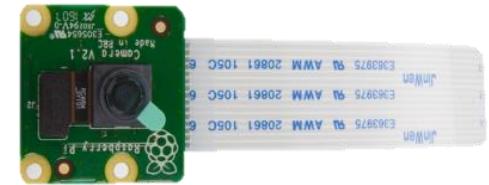
마이크로SD Card
16GB 이상



마이크로SD Card
리더기



마이크로 USB



라즈베리파이용 카메라



L9110 모터 드라이버



DC모터 x 2



서보 모터



라커 스위치



배터리 홀더



리튬 배터리



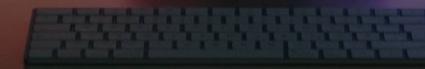
리튬 배터리 충전모듈

+ 조립용 물품

+ WIFI 공유기 관리자 ID/PWD

+ 제어할 PC (라즈베리파이를 PC로 사용시에 필요없음)

Install



Install

Raspberry OS

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. [Watch our 45-second video](#) to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Windows](#)

[Download for macOS](#)

[Download for Ubuntu for x86](#)

To install on **Raspberry Pi OS**, type
`sudo apt install rpi-imager`
in a Terminal window.



참고 링크:

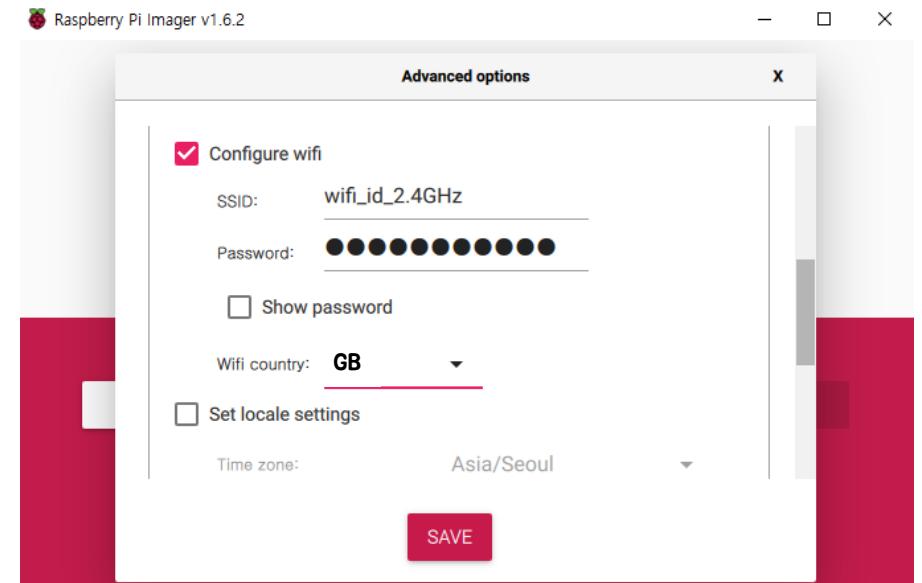
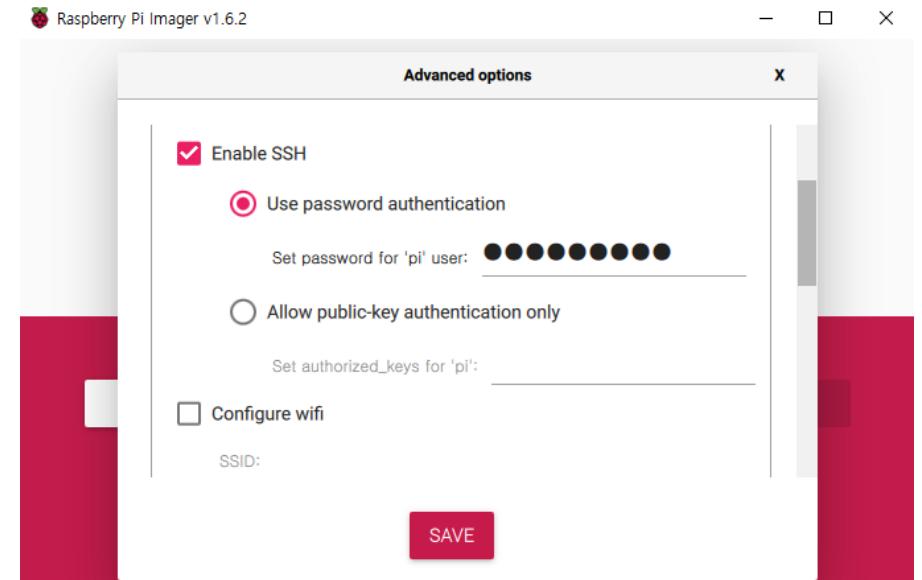
<https://www.youtube.com/watch?v=ntaXWS8Lk34>

Install

Raspberry OS



Ctrl + Shift + x



CHOOSE OS → Use custom

(최신 버전에서 카메라 호환의 문제가 있는 것으로 판단)

구버전: <http://downloads.raspberrypi.org/raspbian/images/raspbian-2020-02-14/>

Install

Raspberry OS



Install

program



Visual Studio Code



Install

Python

<https://www.python.org/>

1991년 귀도 반 로섬이 발표한 고급 프로그래밍 언어

언어 이름은 귀도가 좋아하는 코미디 Monty Python's Flying Circus에서 따옴

인터프리식 언어, 객체지향적, 동적 타이핑 대화형 언어

비영리 파이썬 소프트웨어 재단이 관리하고 있으며, 사실상 표준은 C언어로 구현된 Cython



귀도 반 로섬

*인터프리식 언어: 코드를 한줄 한줄 읽어가며 명령을 처리

**객체지향적: 파이썬은 Object(객체)로 이루어져 있으며, 객체 지향 프로그래밍을 추구하고 있다.

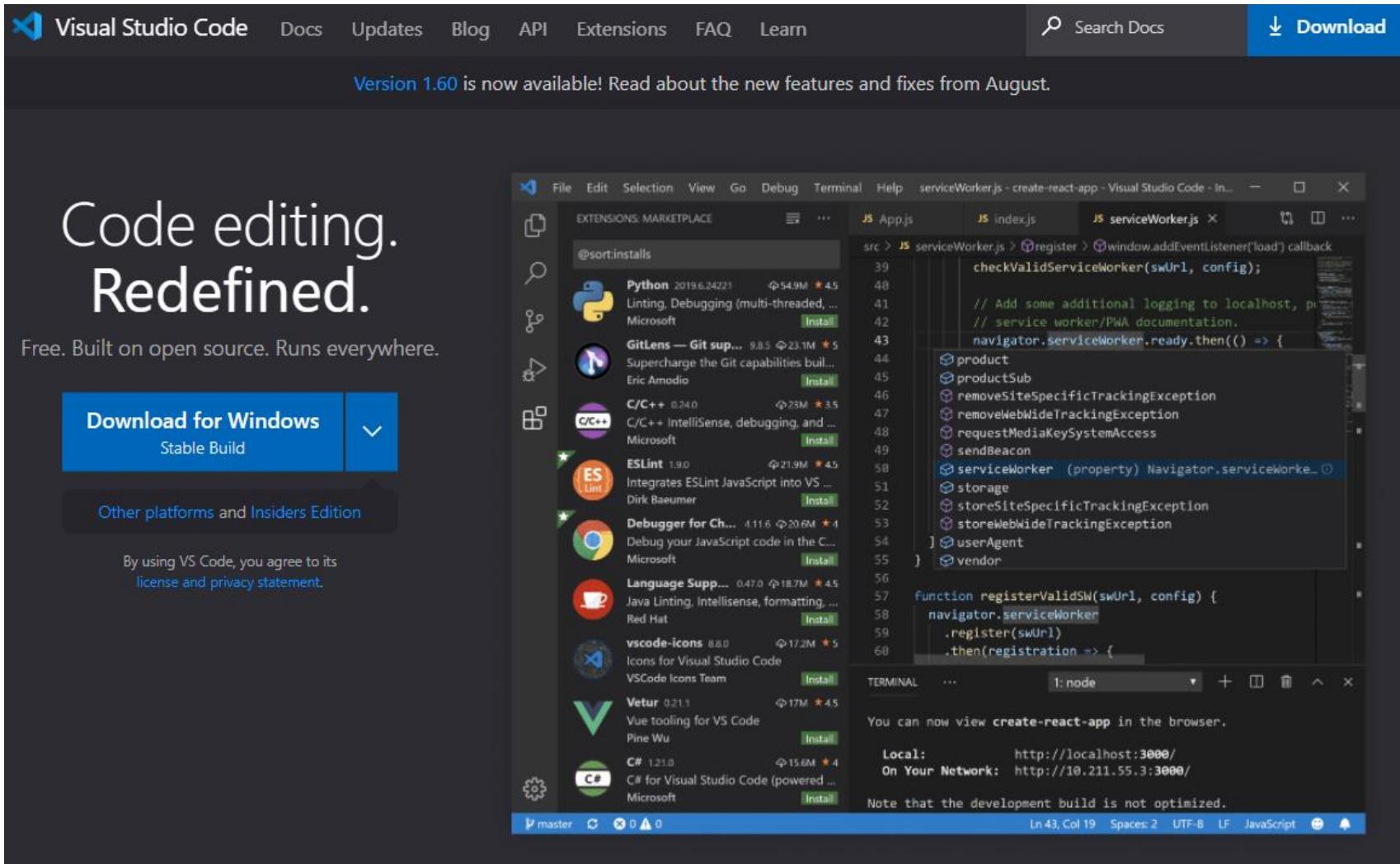
OOP(Object Oriented Programming, 객체 지향 프로그래밍): 프로그래밍에 필요한 데이터를 추상화하여 상태와 행위를 가진 객체를 만들고, 객체들 간의 유기적 상호작용을 통해 로직을 구성하는 방법

***동적 타이핑 대화형 언어: 파이썬은 변수를 만들 때 변수의 타입을 먼저 지정해주지 않습니다. 그리고, 프로그램 자체가 영어 친화적으로 대화하듯 되어 되어 있습니다.

Install

코드 편집툴: VS Code

<https://code.visualstudio.com/>



Install

Raspberry Pi, IP 확인

The screenshot shows the configuration interface for an ipTIME A3008-MU router. The left sidebar contains a tree view of settings, with '내부 네트워크 설정' (Internal Network Settings) selected. The main panel displays the following information:

- 내부 IP주소:** 192.168.0.1 (MAC address: 70-5D-CC-01-28-9B)
- 서브넷 마스크:** 255.255.255.0
- 내부 게이트웨이:** (Fields for gateway IP and '수동 지정' checkbox)
- 내부 기본 DNS:** (Fields for primary DNS and '수동 지정' checkbox)
- 내부 보조 DNS:** (Fields for secondary DNS and '수동 지정' checkbox)

Below these settings is a section titled "사용중인 IP 주소 정보" (List of Used IP Addresses) which shows 21 entries. At the bottom of the main panel, there is a summary table:

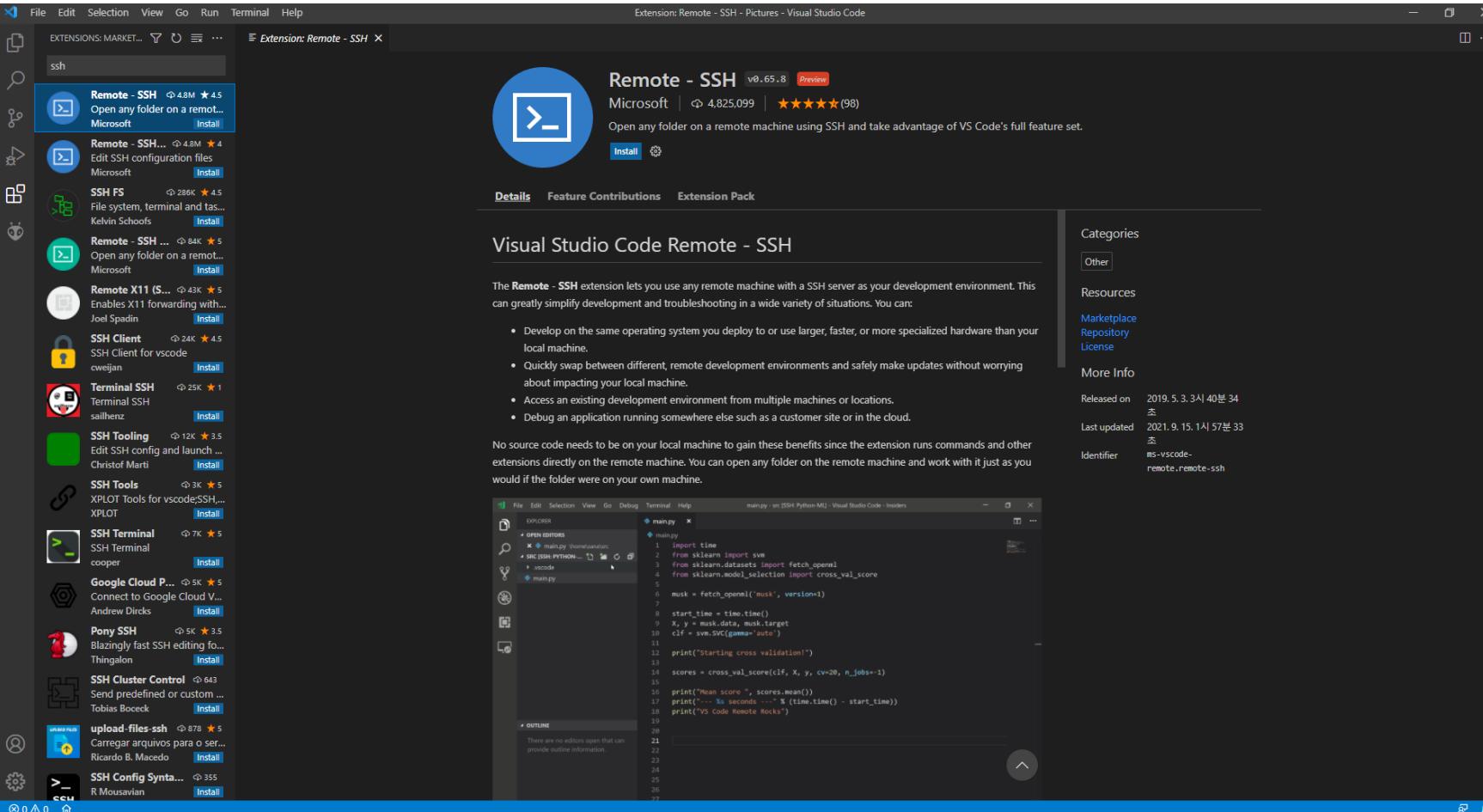
192.168.0.53	B8-27-EB-BF-98-4D	raspberrypi	무선:자동할당
--------------	-------------------	-------------	---------

Raspberrypi ip주소 확인

Install

VS Code SSH Extension

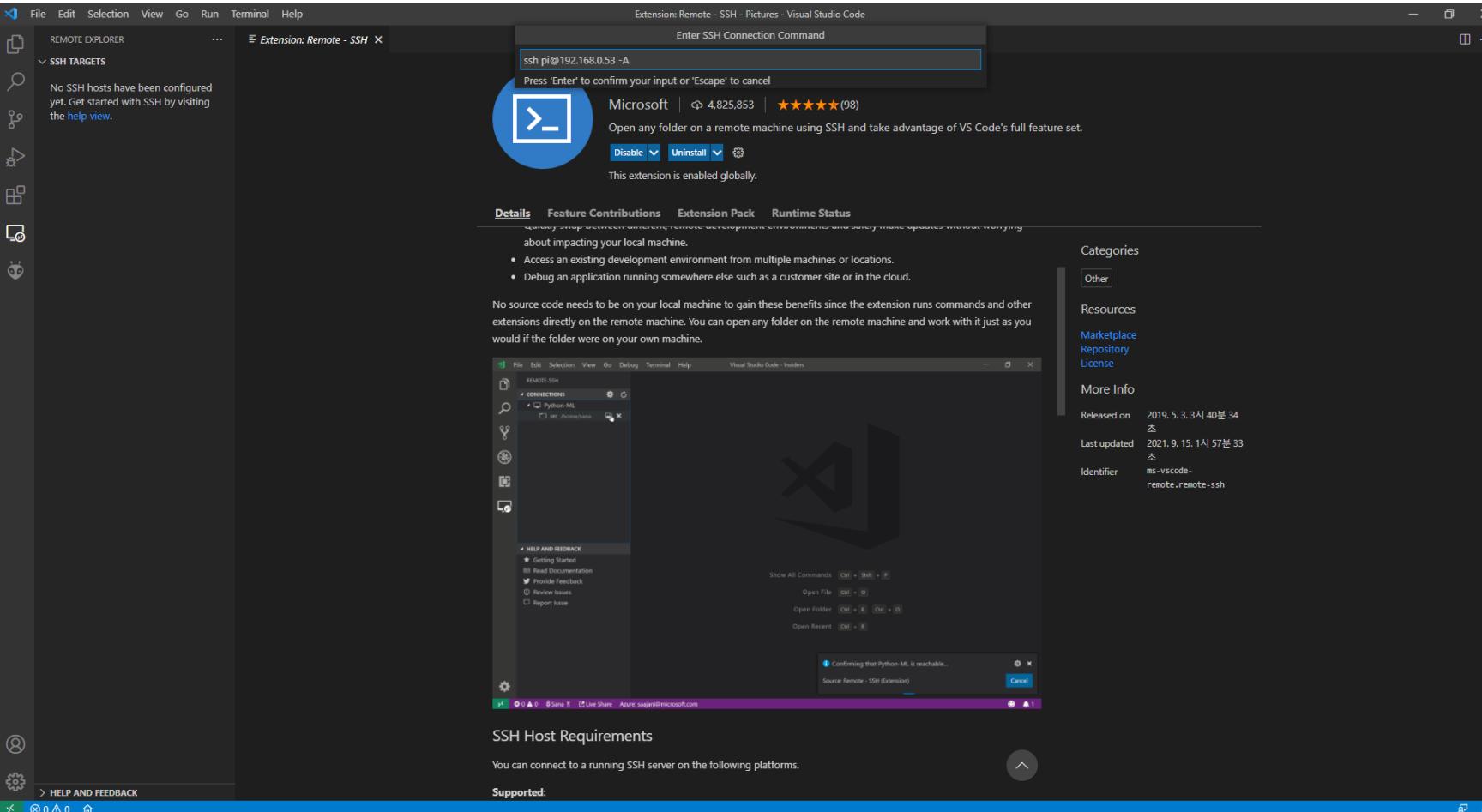
VS Code Extension에 'Remote -SSH'를 설치하여 라즈베리파이와 연결



Install

VS Code SSH Extension

ssh id@ip주소 -A

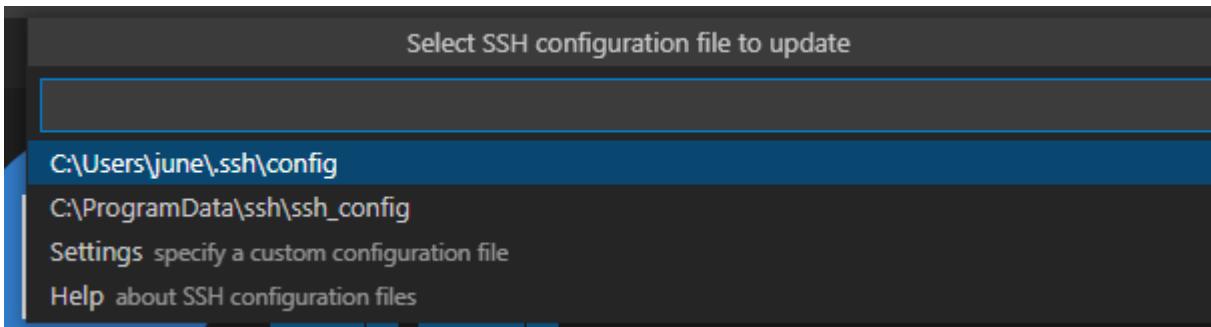


Install

VS Code SSH Extension

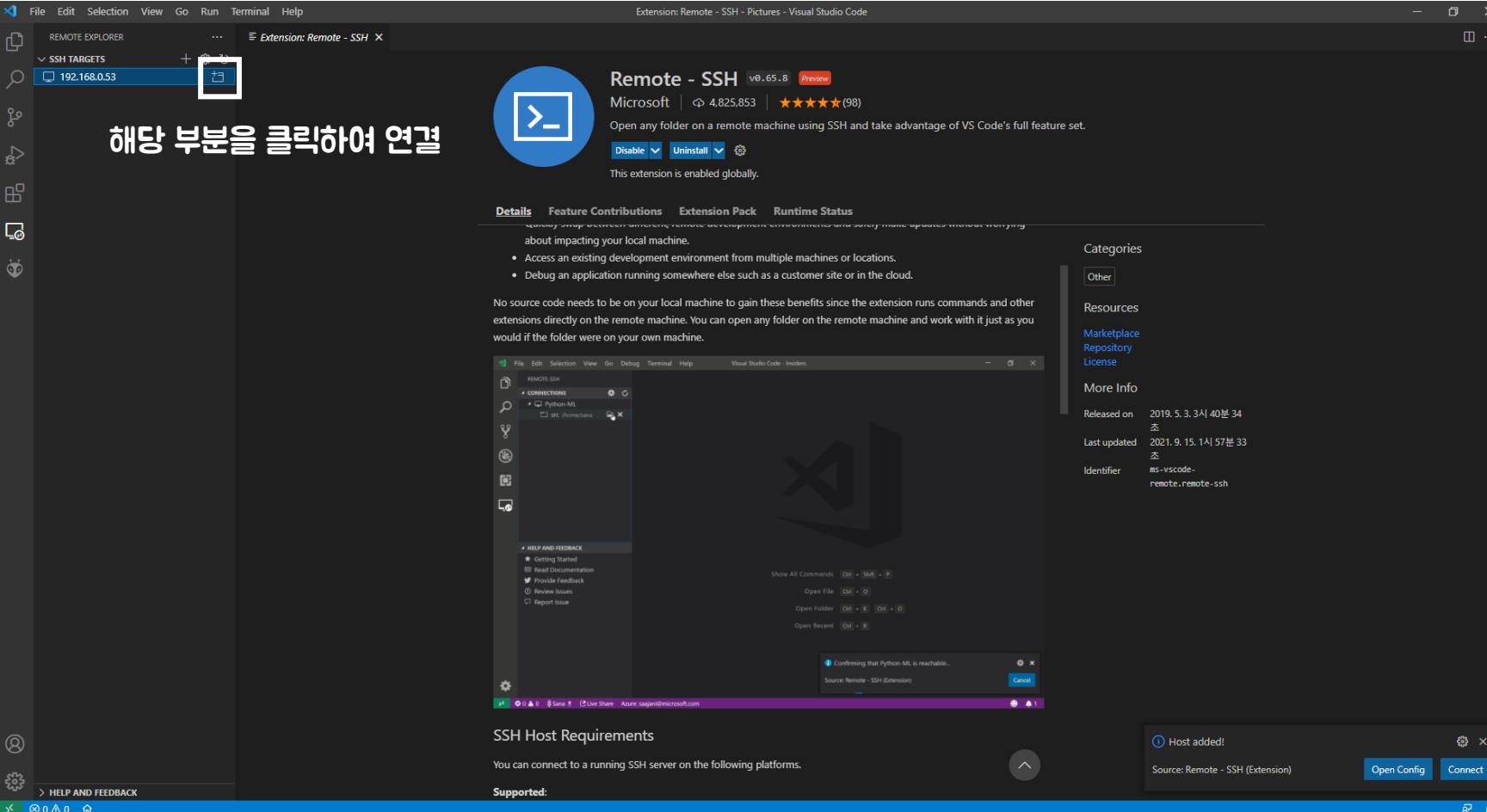
설정할 파일 위치 저장

기본 처음으로 설정해서 진행함



Install

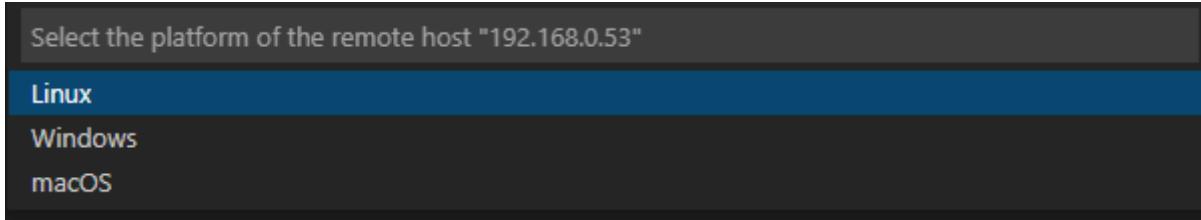
VS Code SSH Extension



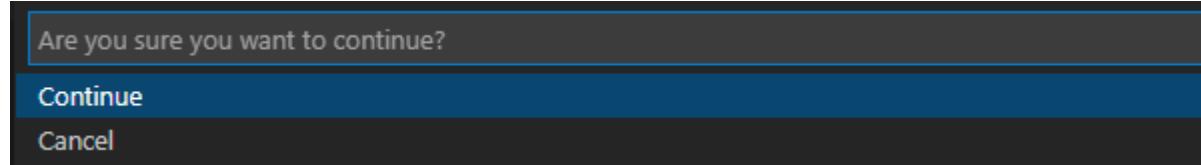
Install

VS Code SSH Extension

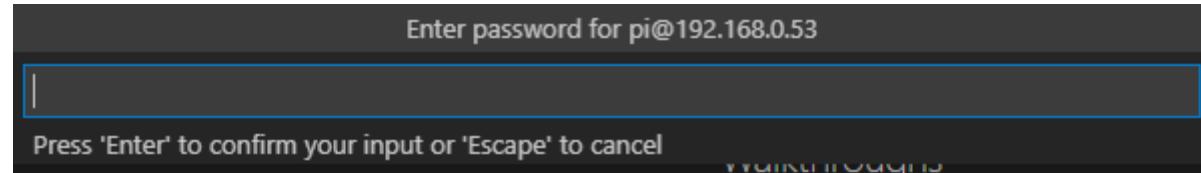
Linux로 설정



Continue



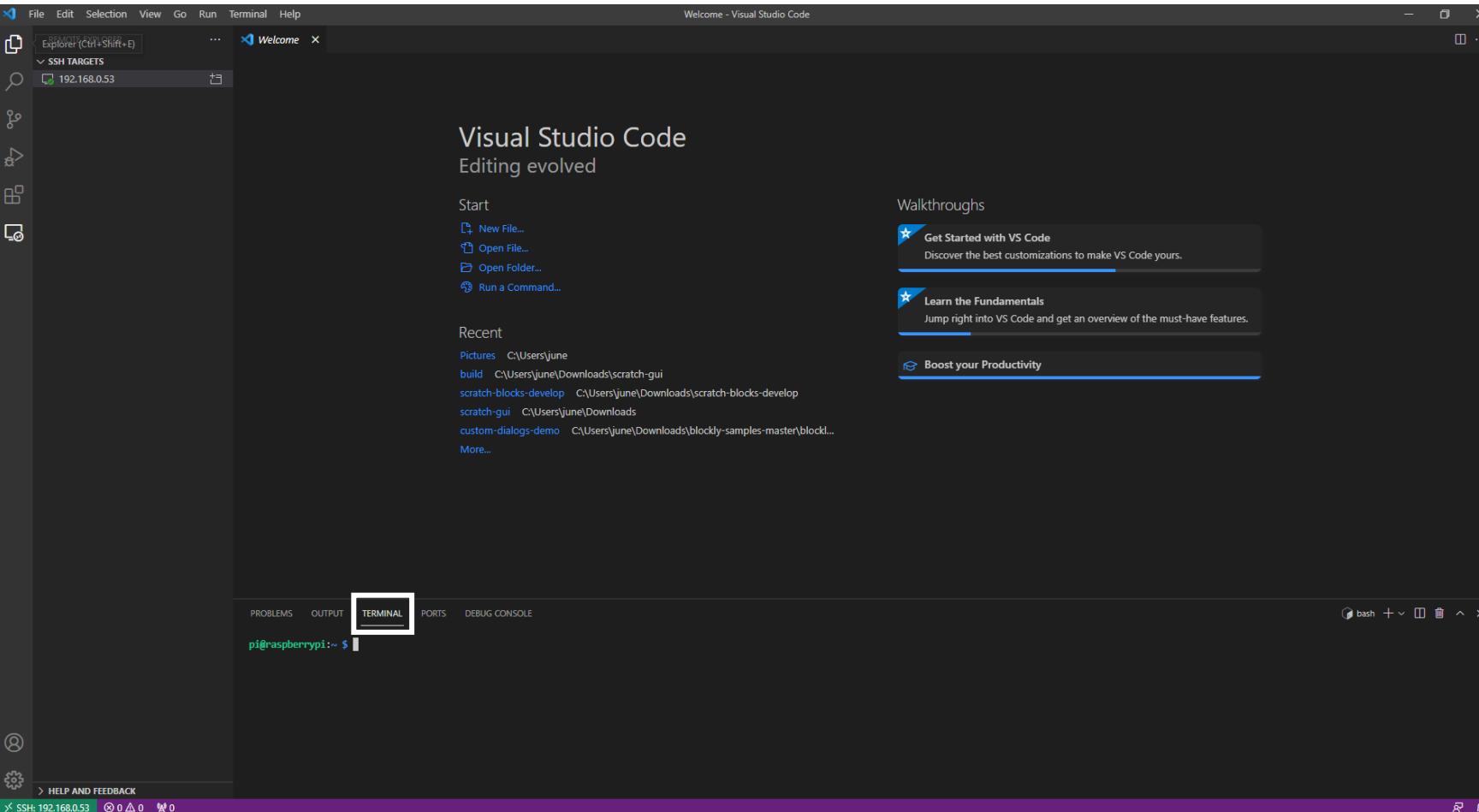
비밀번호 작성: 초기 | 비밀번호: raspberry



Install

VS Code SSH Extension

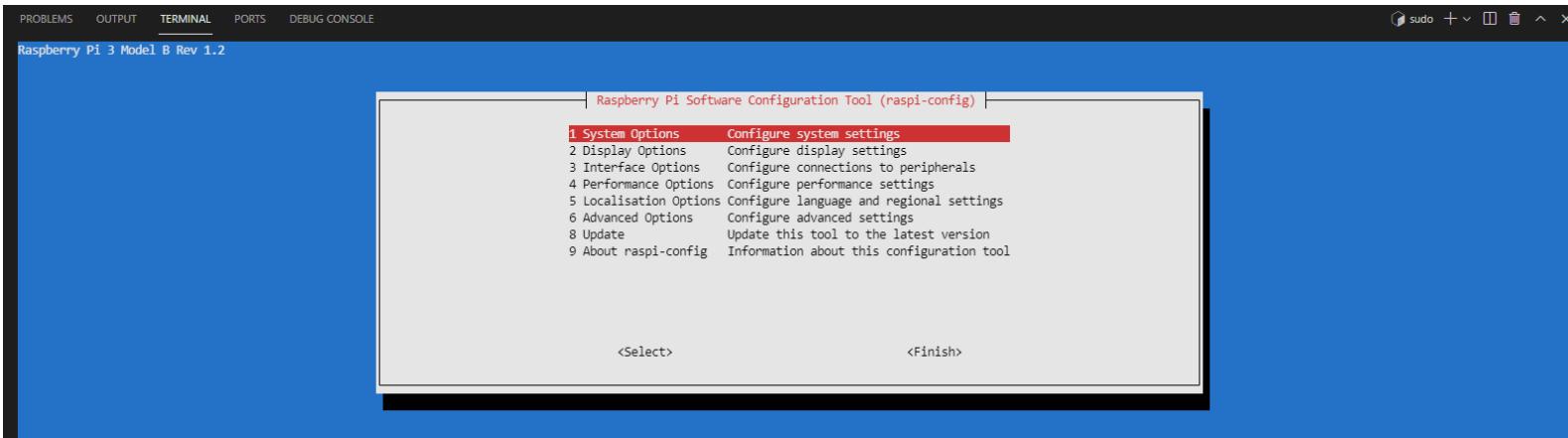
터미널을 열면 SSH로 제어 가능



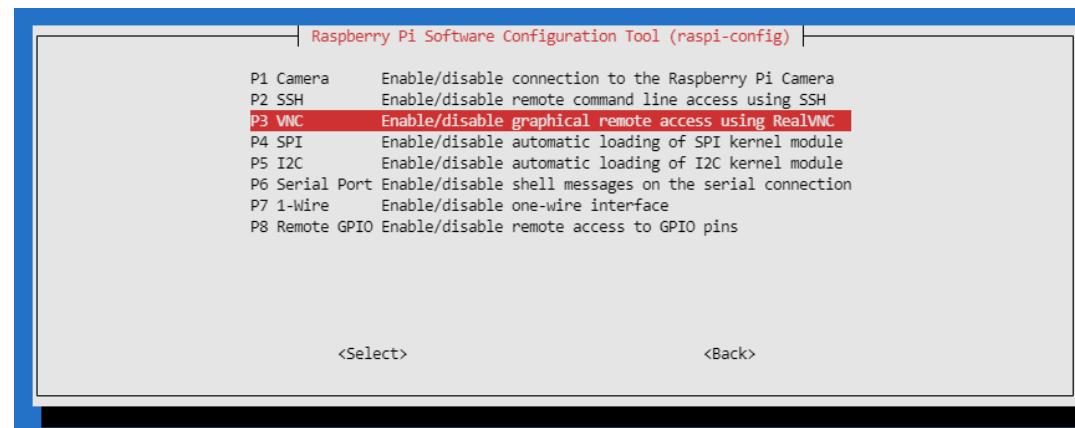
Install

Raspberry Pi VNC ON

터미널에 'sudo raspi-config' 작성시에 해당 설정이 나옴



해당 부분에서 '3 Interface Options' 클릭시에 아래와 같이 나오며



VNC를 눌러서 'YES'를 눌러서 사용 가능하게 설정 → 이와 같이 하면 VNC Viewer에서 접근 가능

Install

VNC Viewer

<https://www.realvnc.com/en/connect/download/viewer/>

VNC® Connect consists of VNC® Viewer and VNC® Server

Download VNC® Viewer to the device you want to control from, below. Make sure you've [installed VNC® Server](#) on the computer you want to control.



[Download VNC Viewer](#)

SHA-256:

EXE x86/x64 ▾

[Looking for VNC® Server?](#)

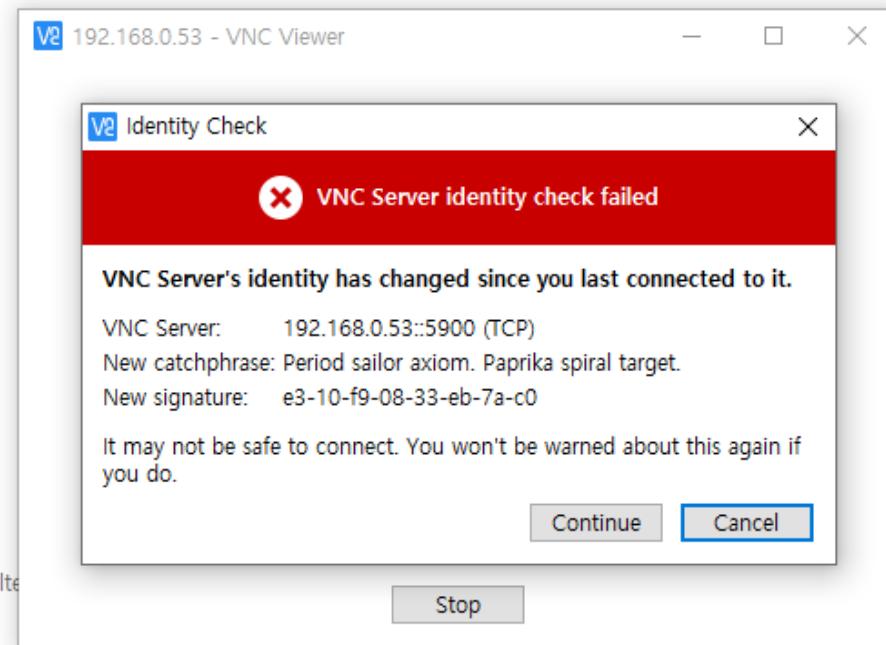
VS Code SSH로 연결하여 진행할 시에 VNC Viewer 필요없이 텍스트 상으로만 제어하실 수도 있습니다.

Install

VNC Viewer



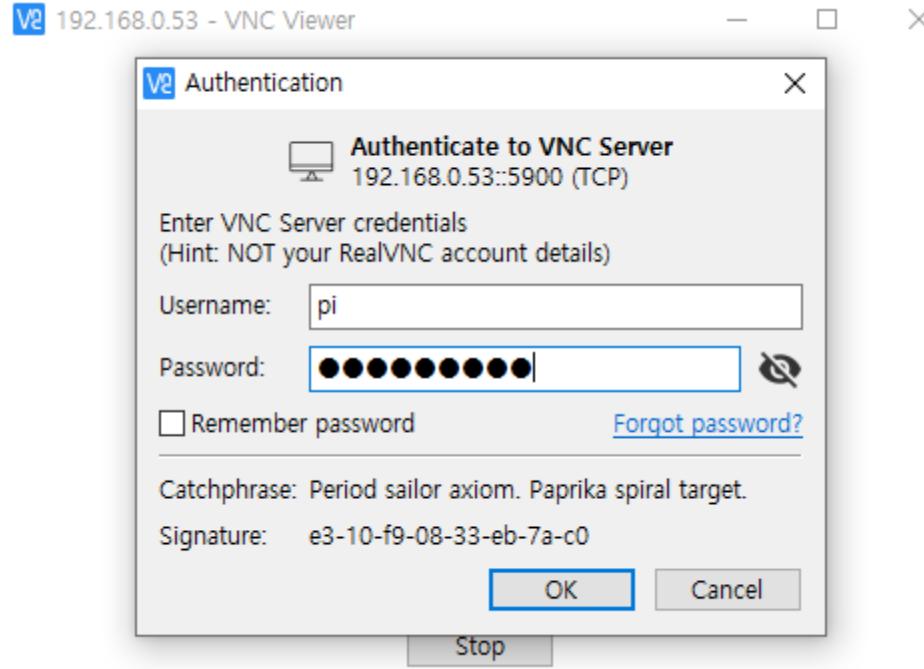
라즈베리파이 ip 주소 작성을 하고 엔터를 누르면



해당 창이 나오고, continue를 눌러서 진행하면 됩니다.

Install

VNC Viewer



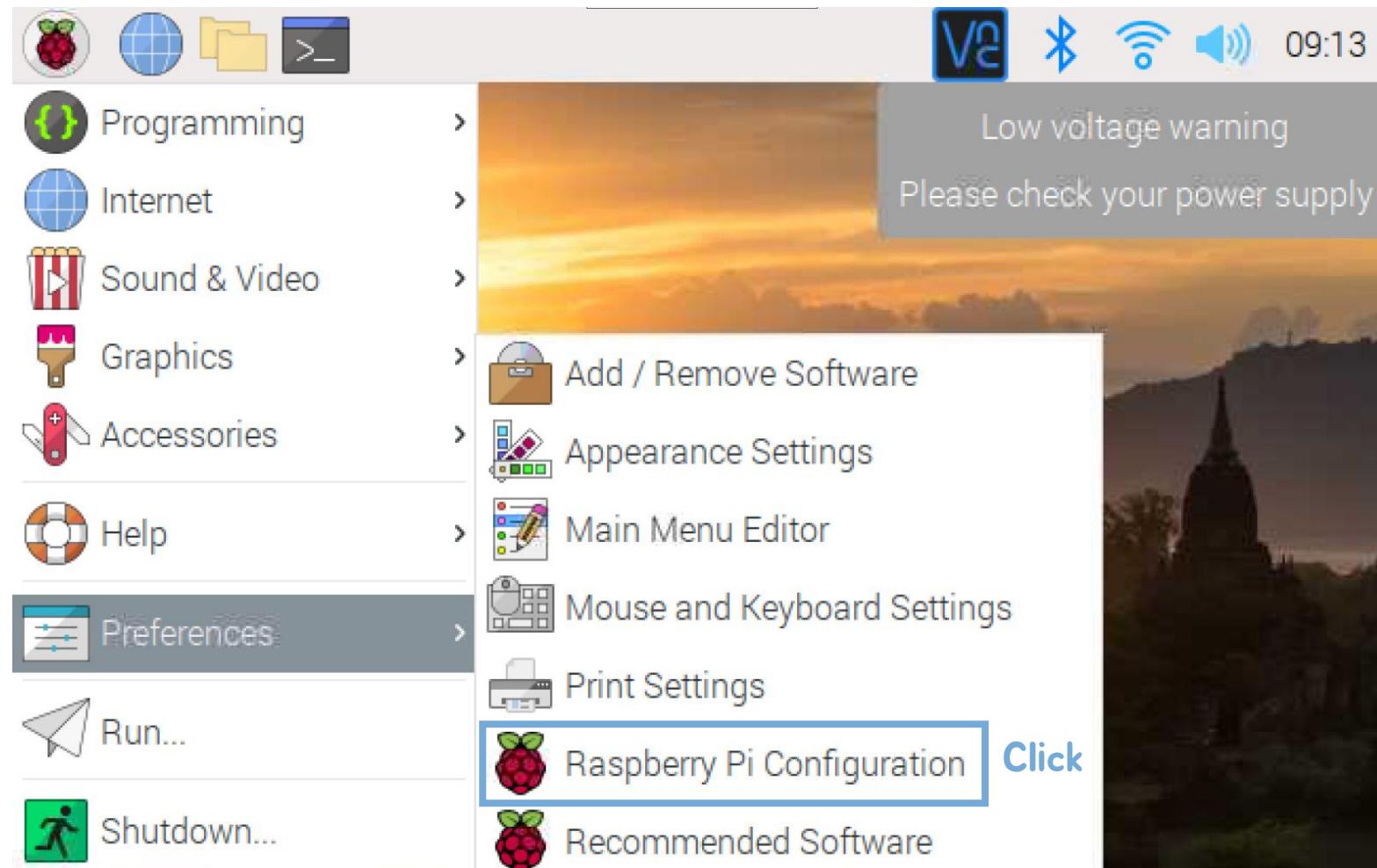
Username과 password를 작성하고 'OK'을 클릭하면 됩니다.

기본 Username: pi

기본 Password: raspberry

Install

VNC Viewer

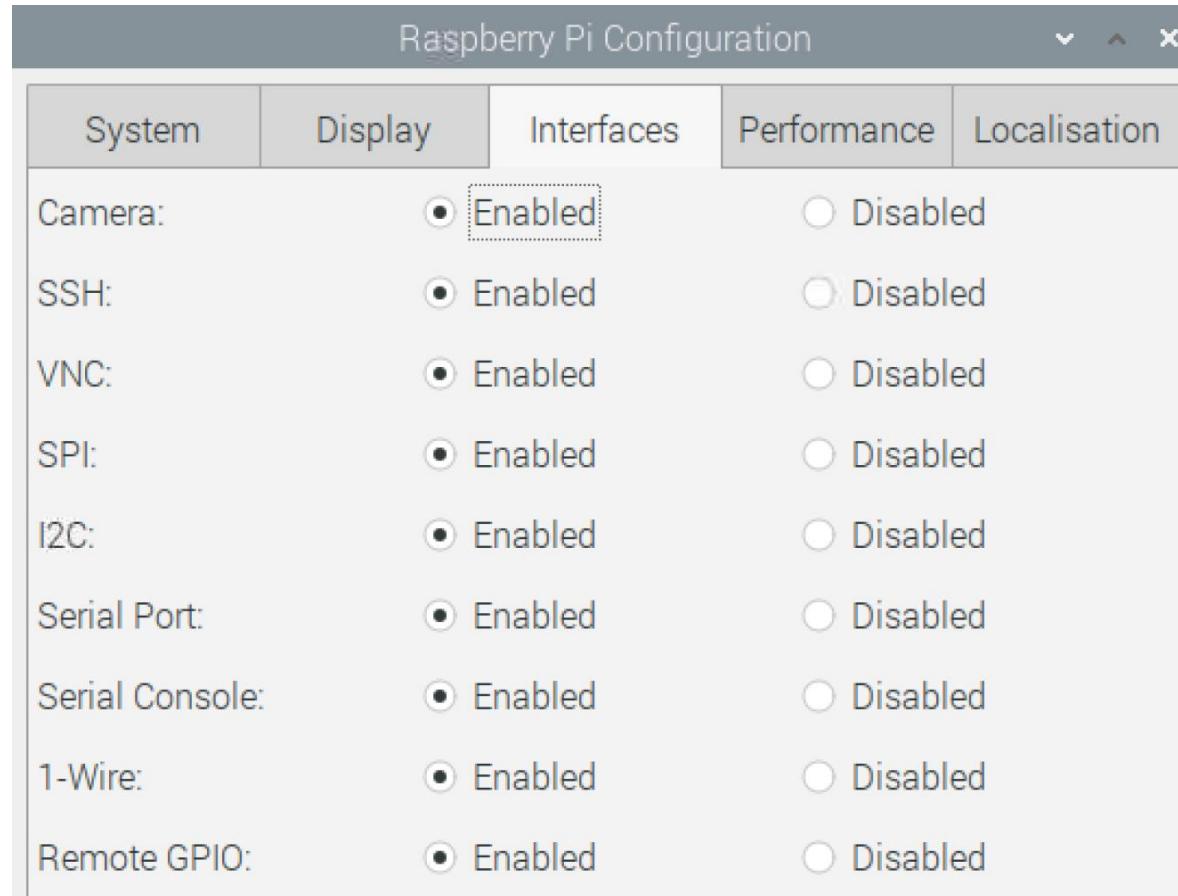


Install

VNC Viewer

Interfaces 부분에 Enabled로 다 두어서 사용 가능하게 하겠습니다.

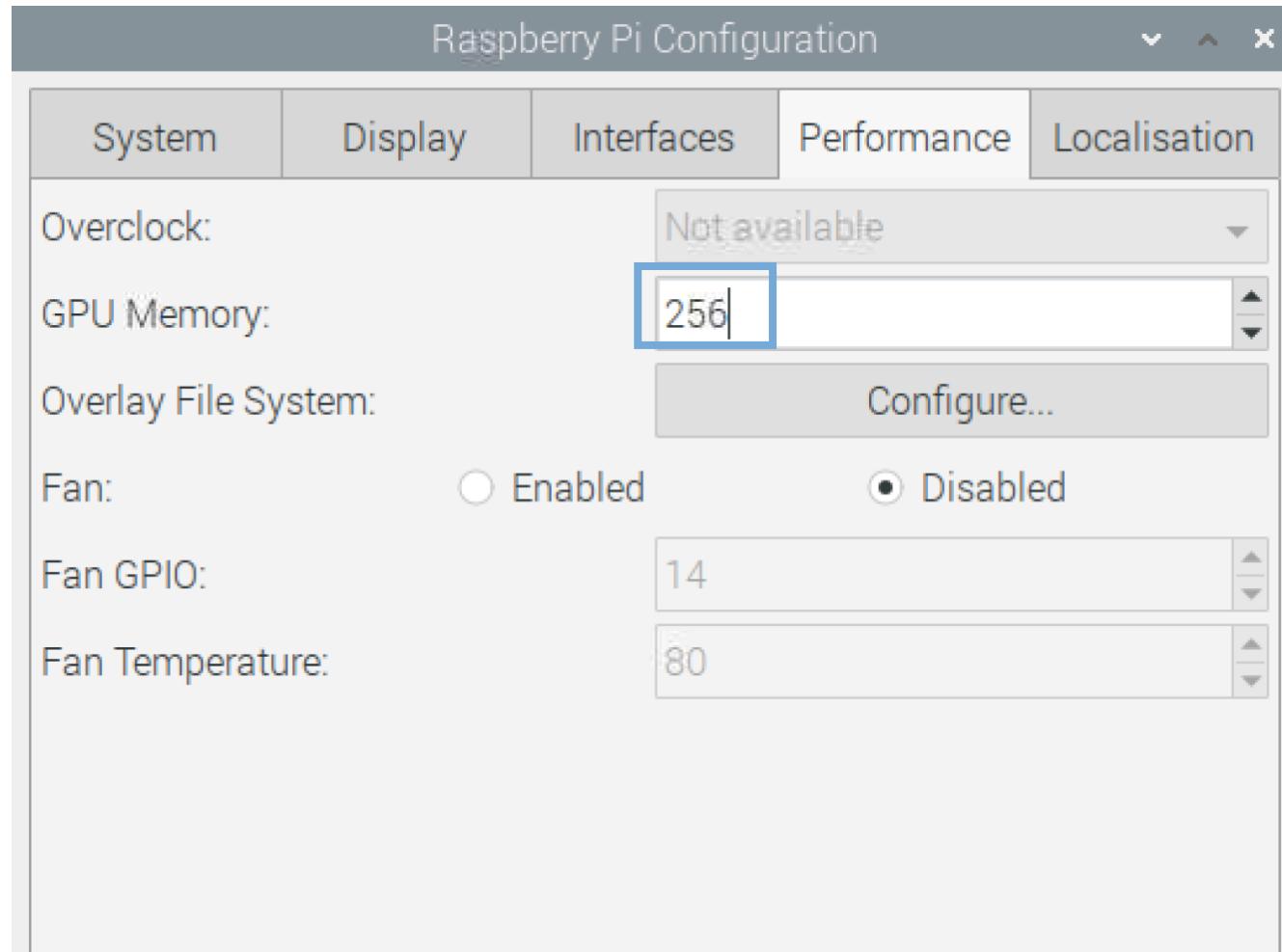
본래, 사용 목적에 따라 설정해야 하지만, 편의를 위해 다 가능하게 설정



Install

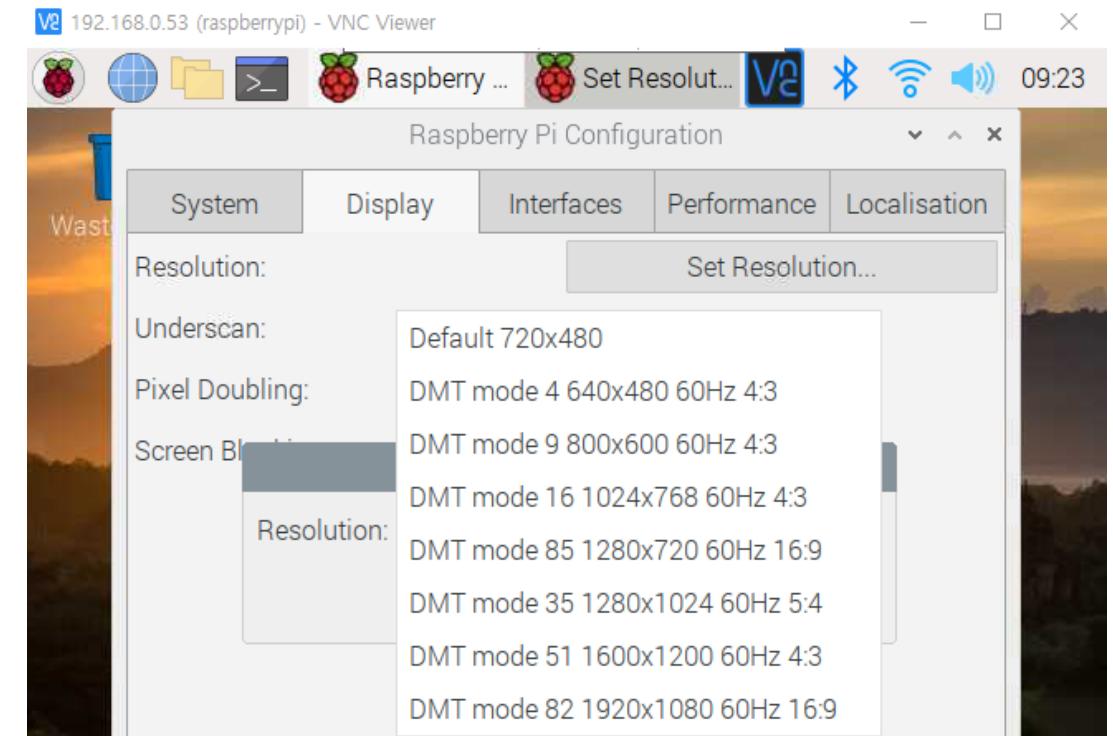
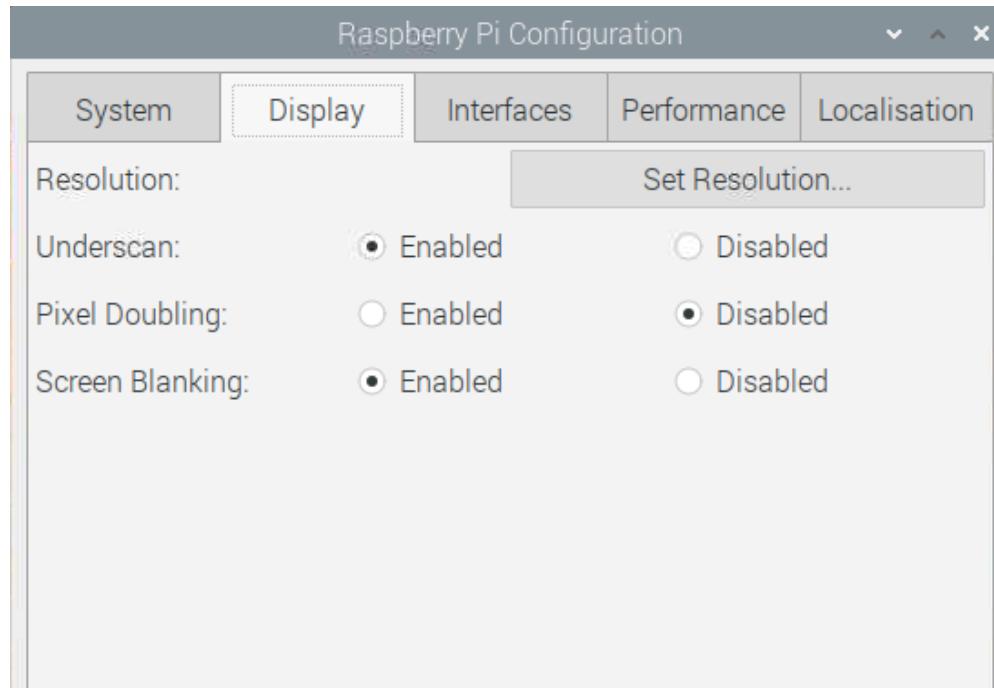
VNC Viewer

GPU Memory를 256 정도로 올려주도록 하겠습니다.



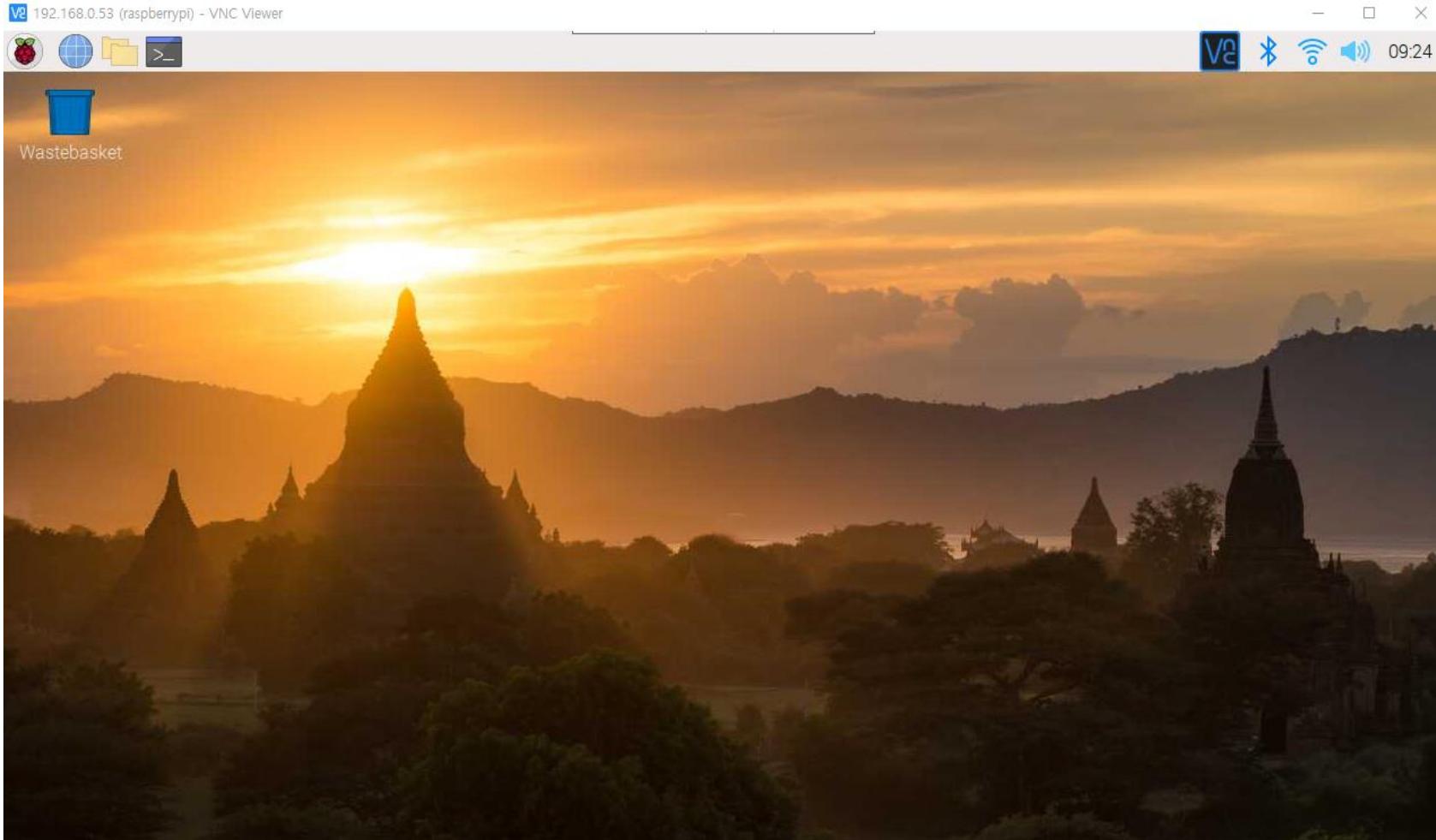
Install

VNC Viewer



Install

VNC Viewer



참고) 라즈베리파이 VNC Port: 5900

해당 프로젝트를 진행하면서 사용할 기본적인 리눅스 명령어만 알아보겠습니다.

sudo: Super User Do

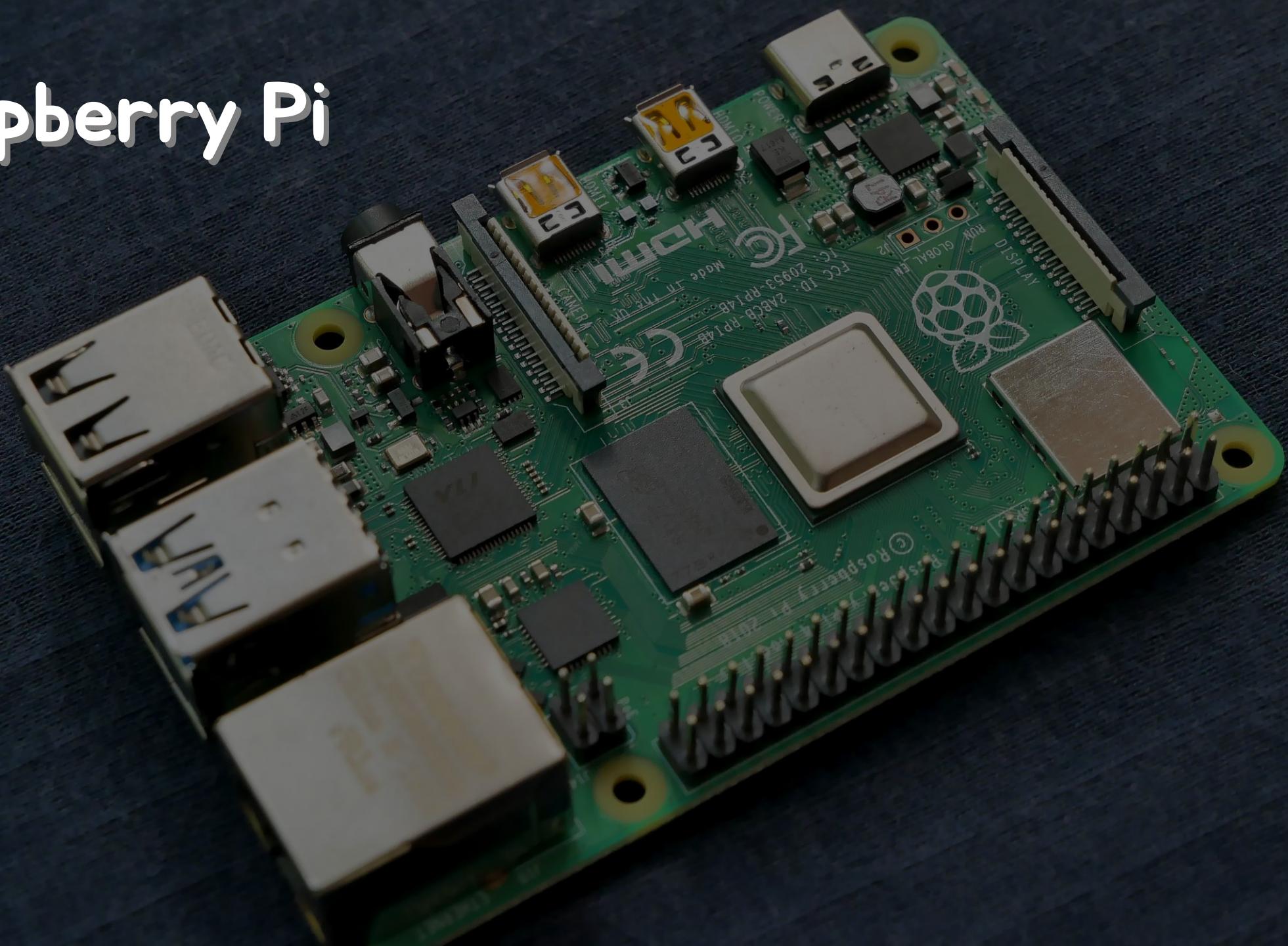
ls : List

cd : Change Directory

mkdir: Make Directory

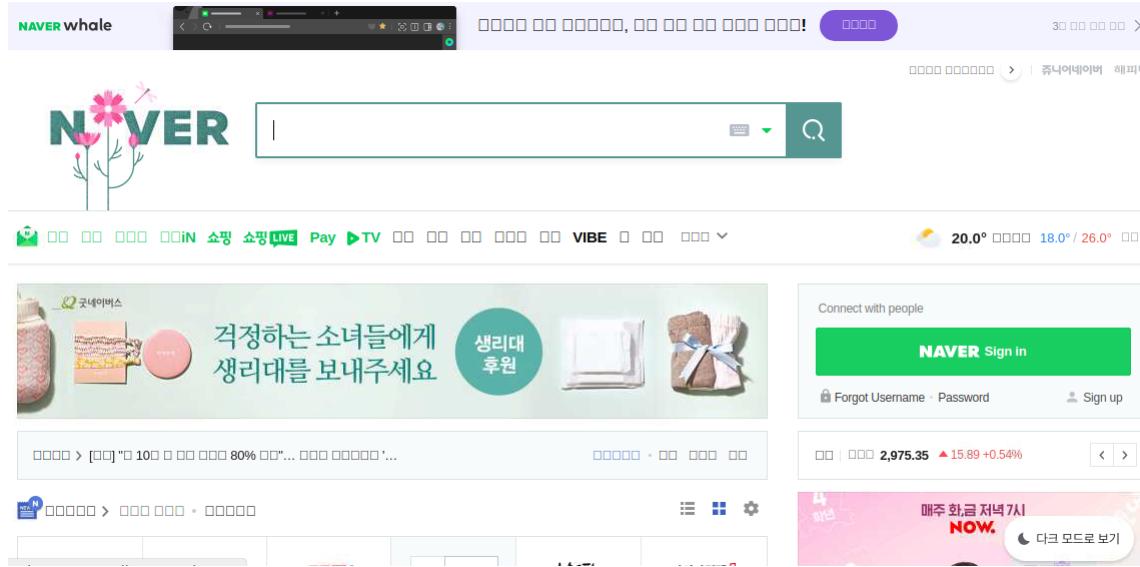
touch: Create File

Raspberry Pi



Raspberry Pi 3B +

한글화



초기에는 한글이 깨져서 나오는 것을 확인 가능

먼저, 터미널을 열고

sudo apt-get update

sudo apt-get upgrade

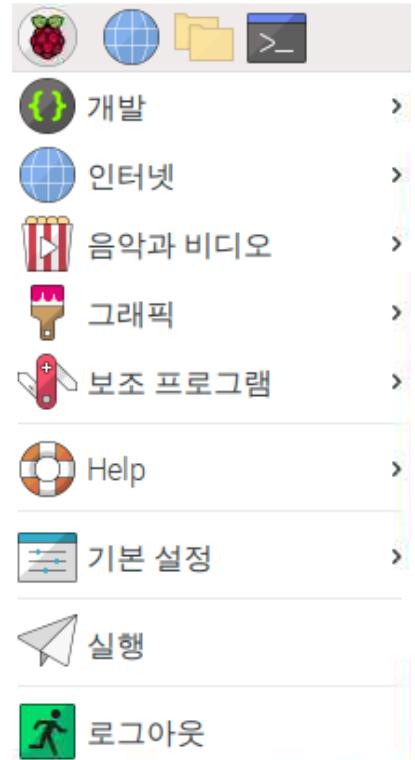
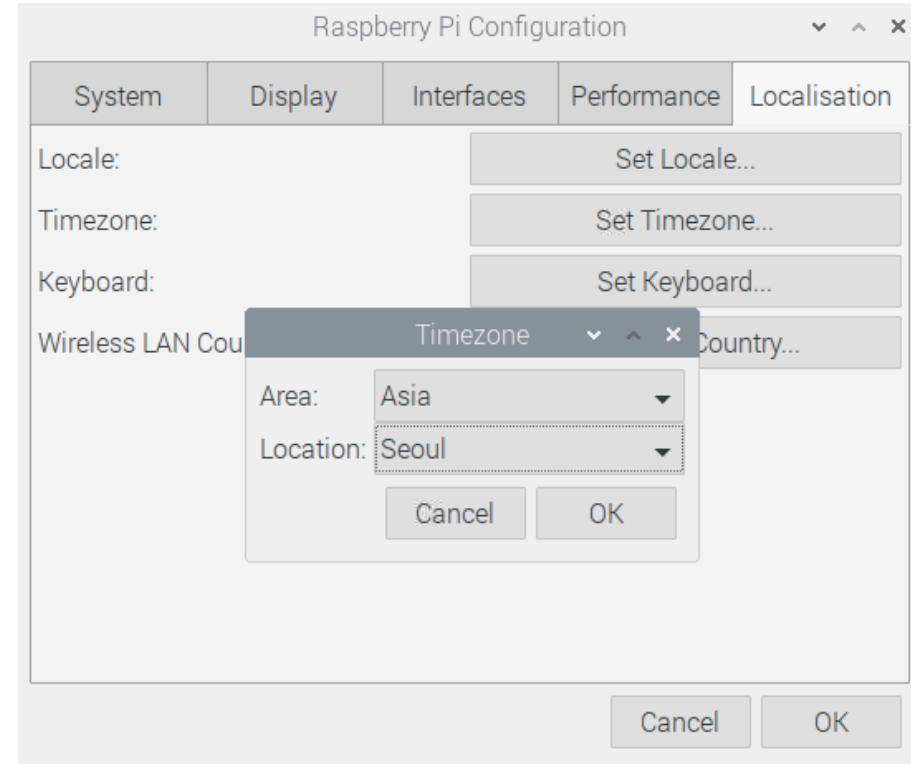
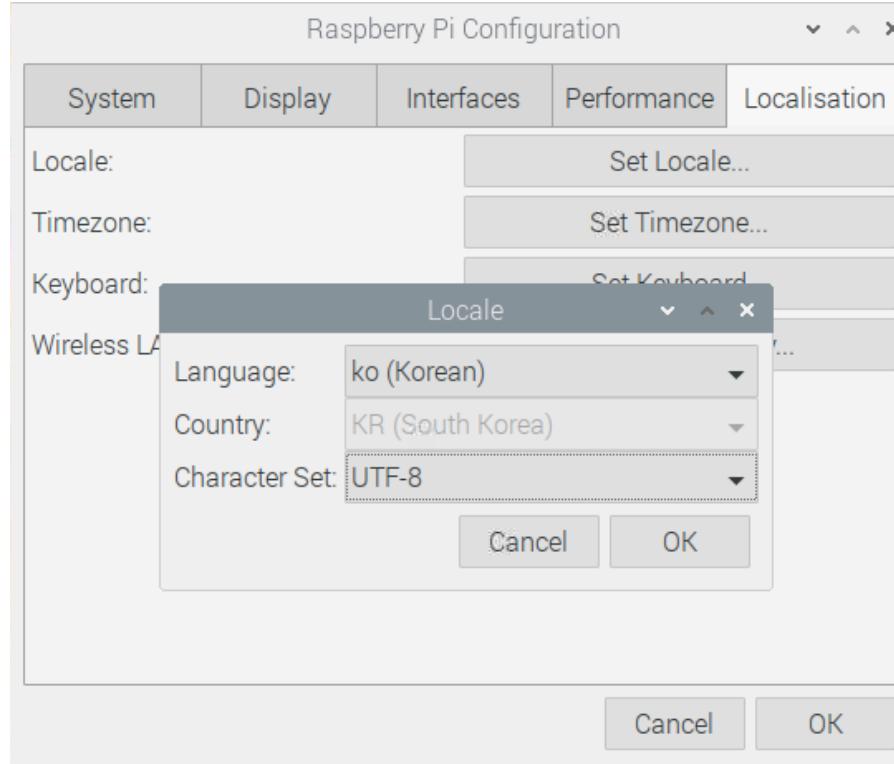
Apt 패키지를 업데이트 해줍니다.

Advanced Packaging Tool(apt)은 리눅스에서 쓰이는 패키지 관리 명령어 도구입니다.

sudo apt install fonts-unfonts-core -y

Raspberry Pi 3B +

한글화



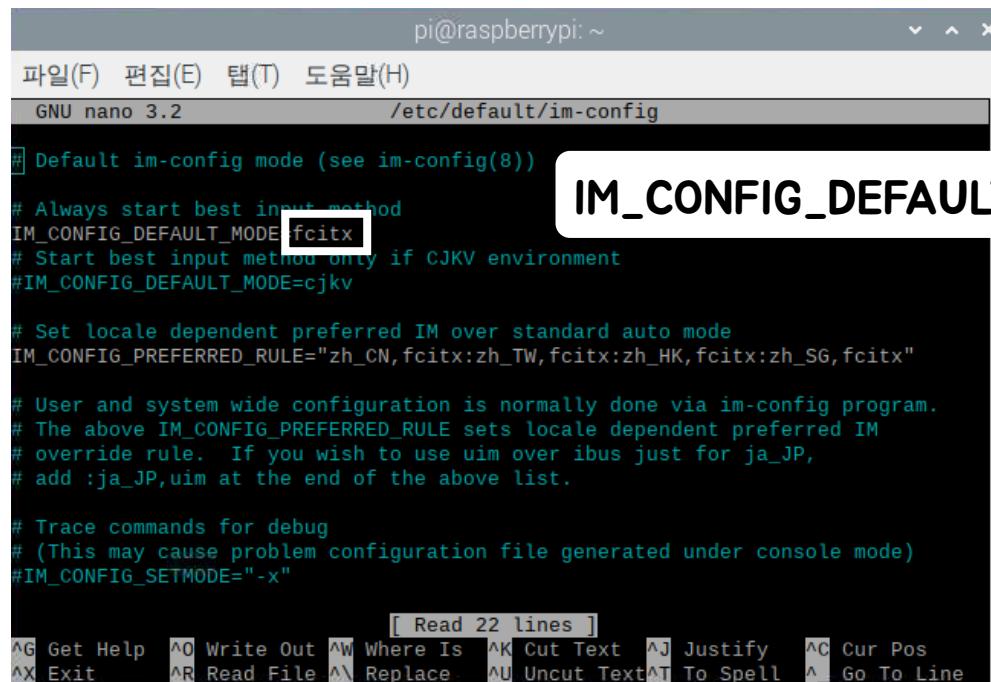
이와 같이 한글로 나오는 것을 확인할 수 있습니다.

Raspberry Pi 3B +

한글화

```
sudo apt remove ibus ibus-hangul  
sudo apt install fcitx fcitx-hangul -y  
sudo nano /etc/default/im-config
```

기존에 패키지를 삭제하고
필요한 패키지 설치
환경 설정을 위해 파일 OPEN

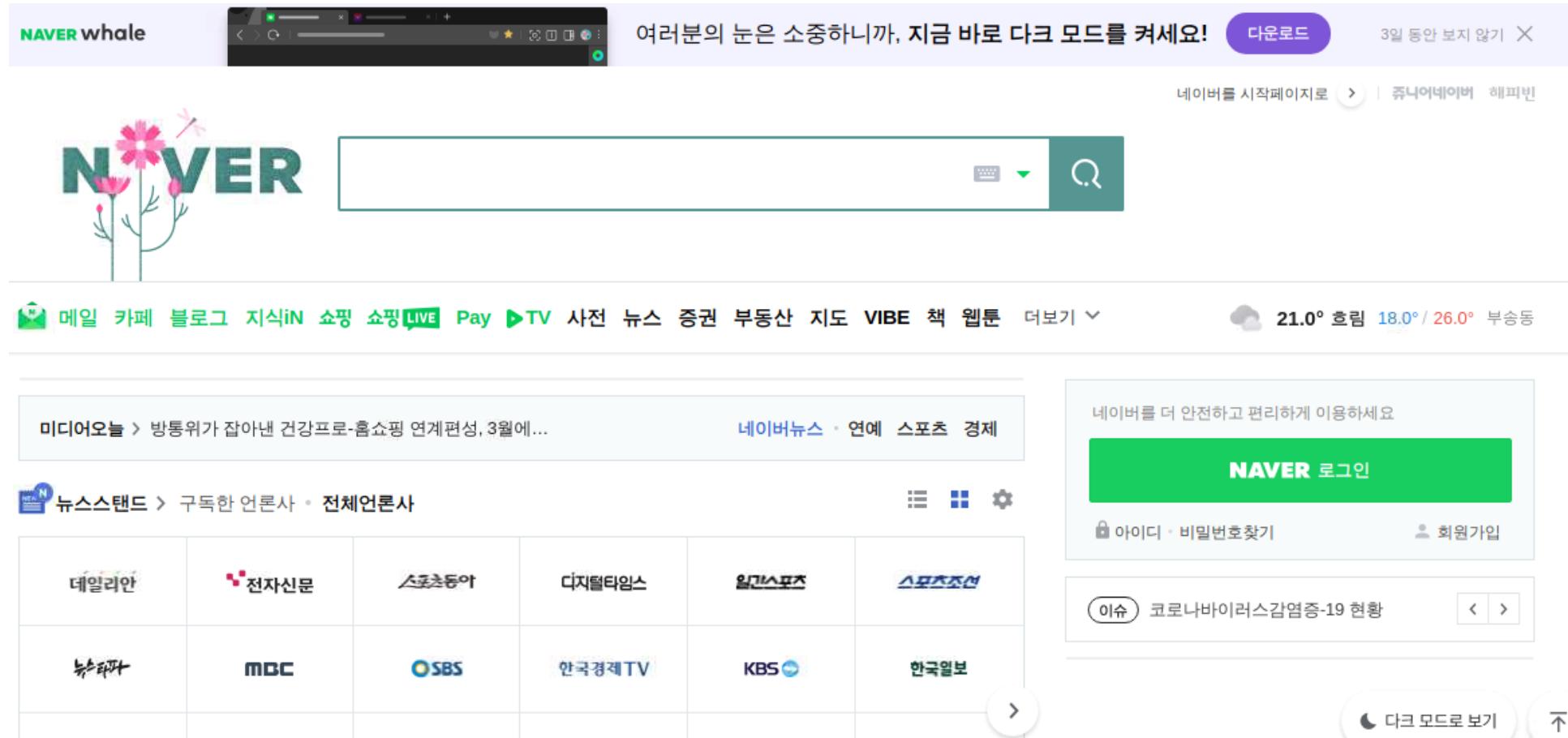


```
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
GNU nano 3.2          /etc/default/im-config  
# Default im-config mode (see im-config(8))  
# Always start best input method  
IM_CONFIG_DEFAULT_MODE=fcitx  
# Start best input method only if CJKV environment  
#IM_CONFIG_DEFAULT_MODE=cjkv  
  
# Set locale dependent preferred IM over standard auto mode  
IM_CONFIG_PREFERRED_RULE="zh_CN,fcitx:zh_TW,fcitx:zh_HK,fcitx:zh_SG,fcitx"  
  
# User and system wide configuration is normally done via im-config program.  
# The above IM_CONFIG_PREFERRED_RULE sets locale dependent preferred IM  
# override rule. If you wish to use uim over ibus just for ja_JP,  
# add :ja_JP,uim at the end of the above list.  
  
# Trace commands for debug  
# (This may cause problem configuration file generated under console mode)  
#IM_CONFIG_SETMODE="-x"  
  
[ Read 22 lines ]  
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  
^X Exit     ^R Read File  ^\ Replace   ^U Uncut Text  ^T To Spell  ^_ Go To Line
```

IM_CONFIG_DEFAULT_MODE=fcitx
해당 부분 수정 후, 'ctrl+x' → 'Y' 엔터
터미널에 sudo reboot로 재실행

Raspberry Pi 3B +

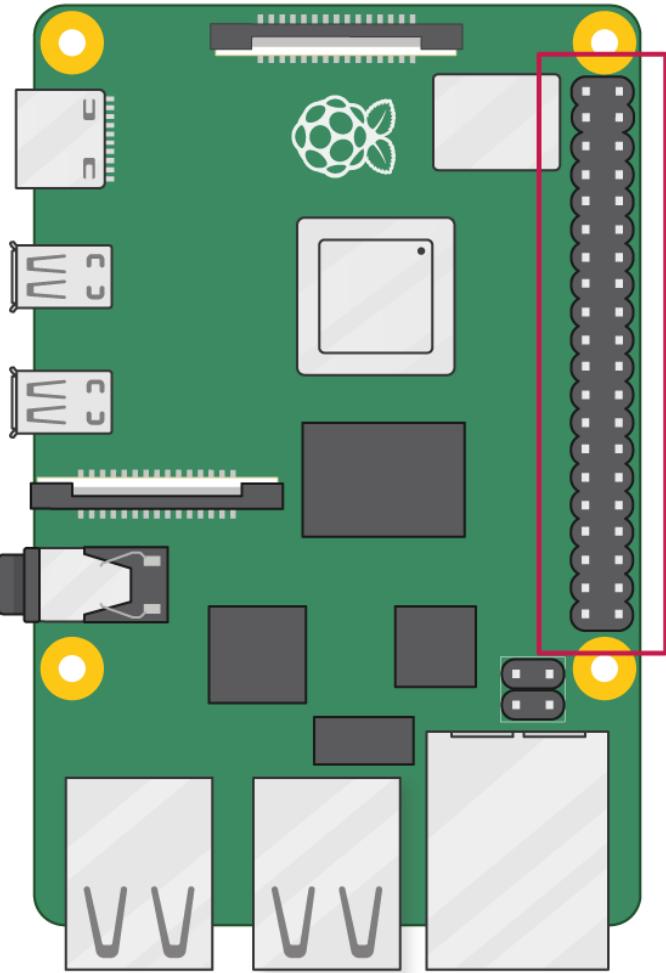
한글화



'ctrl + space'를 누르면 한/영키 변환이 가능

Raspberry Pi 3B +

GPIO(BCM) 핀맵



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

GPIO 제어 with Python

LED ON / OFF

```
1 import RPi.GPIO as GPIO  
2  
3  
4 GPIO.setmode(GPIO.BCM)  
5  
6 PIN_LED = 26  
7  
8 GPIO.setup(PIN_LED, GPIO.OUT)  
9  
10 GPIO.output(PIN_LED, True)  
11 time.sleep(5)  
12 GPIO.output(PIN_LED, False)  
13  
14 GPIO.cleanup()
```

LED를 켜고 5초 후에 꺼는 CODE

Import로 외부 라이브러리 사용 설정

- RPi.GPIO 라즈베리 파이 GPIO제어 라이브러리 **as** → GPIO 별칭 설정
- time: 시간 관련 라이브러리

GPIO.setMode → BOARD와 BCM 설정

- BOARD: 1~40번
- BCM: GPIO 번호

LED 핀번호 GPIO 26 설정

GPIO.setup → 해당 핀 번호, 출력/입력 설정

GPIO.output → 해당 핀, 전기 + / - 설정

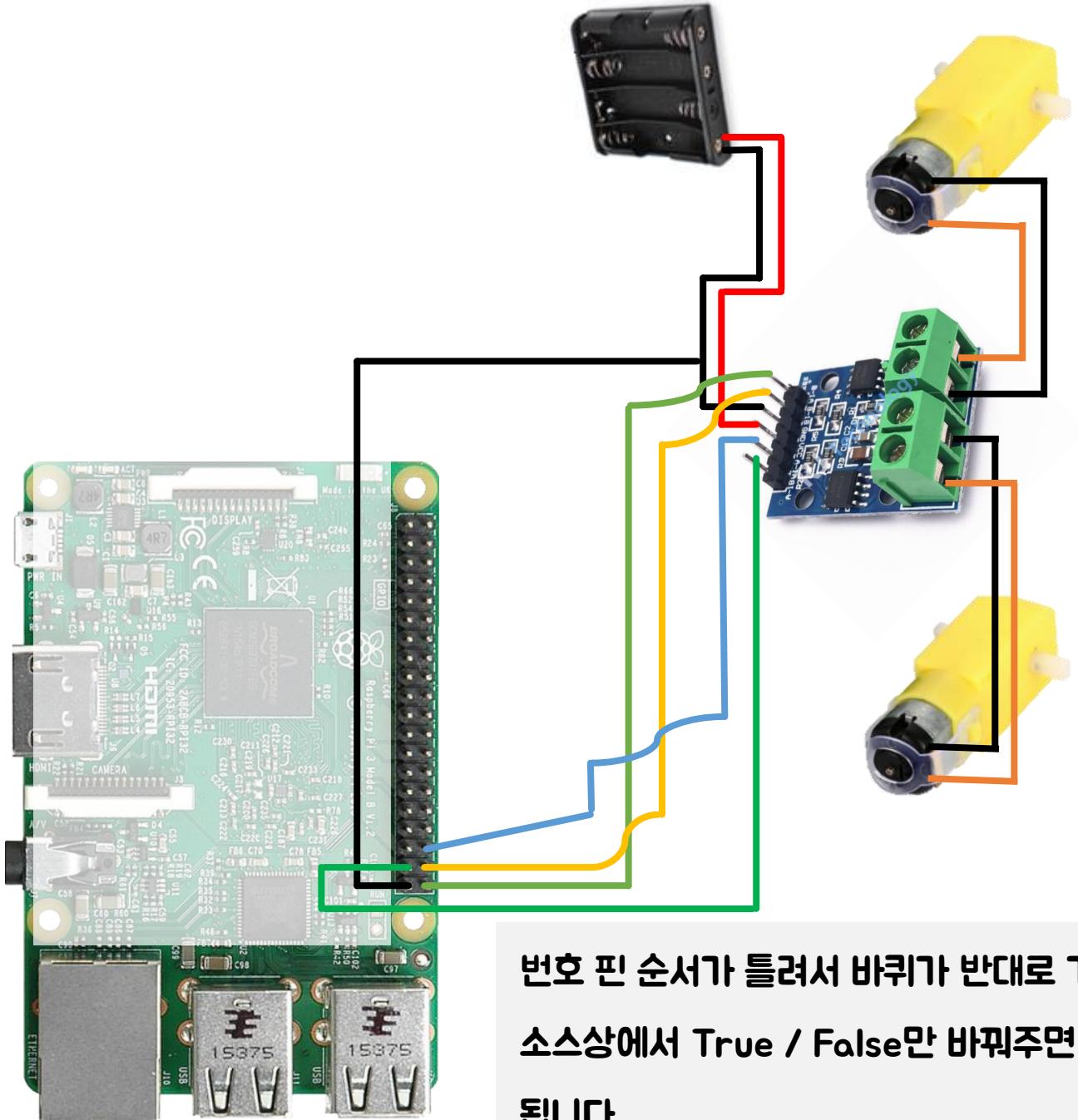
time.sleep(숫자) : 숫자 '초'만큼 멈추기

GPIO.cleanup() → GPIO 설정 핀 clean

GPIO 제어 with Python

DC Motor

```
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setmode(GPIO.BCM)
5
6 PIN_LEFT_MOTOR_FW = 20
7 PIN_LEFT_MOTOR_BW = 21
8 PIN_RIGHT_MOTOR_FW = 16
9 PIN_RIGHT_MOTOR_BW = 26
10
11 GPIO.setup(PIN_LEFT_MOTOR_FW, GPIO.OUT)
12 GPIO.setup(PIN_LEFT_MOTOR_BW, GPIO.OUT)
13 GPIO.setup(PIN_RIGHT_MOTOR_FW, GPIO.OUT)
14 GPIO.setup(PIN_RIGHT_MOTOR_BW, GPIO.OUT)
15
16 GPIO.output(PIN_LEFT_MOTOR_FW, True)
17 GPIO.output(PIN_LEFT_MOTOR_BW, False)
18 time.sleep(3)
19 GPIO.output(PIN_LEFT_MOTOR_FW, False)
20 GPIO.output(PIN_LEFT_MOTOR_BW, True)
21 time.sleep(3)
22 GPIO.output(PIN_LEFT_MOTOR_FW, False)
23 GPIO.output(PIN_LEFT_MOTOR_BW, False)
24 time.sleep(1)
25 GPIO.output(PIN_RIGHT_MOTOR_FW, True)
26 GPIO.output(PIN_RIGHT_MOTOR_BW, False)
27 time.sleep(3)
28 GPIO.output(PIN_RIGHT_MOTOR_FW, False)
29 GPIO.output(PIN_RIGHT_MOTOR_BW, True)
30 time.sleep(3)
31 GPIO.output(PIN_RIGHT_MOTOR_FW, False)
32 GPIO.output(PIN_RIGHT_MOTOR_BW, False)
33 time.sleep(1)
34
35 GPIO.cleanup()
```



번호 핀 순서가 틀려서 바퀴가 반대로 가면
소스상에서 True / False만 바꿔주면
됩니다.

GPIO 제어 with Python

DC Motor

```
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setmode(GPIO.BCM)
5
6 PIN_LEFT_MOTOR_FW = 5
7 PIN_LEFT_MOTOR_BW = 6
8 PIN_RIGHT_MOTOR_FW = 22
9 PIN_RIGHT_MOTOR_BW = 23
10
11 GPIO.setup(PIN_LEFT_MOTOR_FW, GPIO.OUT)
12 GPIO.setup(PIN_LEFT_MOTOR_BW, GPIO.OUT)
13 GPIO.setup(PIN_RIGHT_MOTOR_FW, GPIO.OUT)
14 GPIO.setup(PIN_RIGHT_MOTOR_BW, GPIO.OUT)
15
16 def motor_ctrl(is_left_front, is_right_front, is_left_stop, is_right_stop):
17     if is_left_stop:
18         GPIO.output(PIN_LEFT_MOTOR_FW, False)
19         GPIO.output(PIN_LEFT_MOTOR_BW, False)
20     else:
21         GPIO.output(PIN_LEFT_MOTOR_FW, is_left_front)
22         GPIO.output(PIN_LEFT_MOTOR_BW, not is_left_front)
23
24     if is_right_stop:
25         GPIO.output(PIN_RIGHT_MOTOR_FW, False)
26         GPIO.output(PIN_RIGHT_MOTOR_BW, False)
27     else:
28         GPIO.output(PIN_RIGHT_MOTOR_FW, is_right_front)
29         GPIO.output(PIN_RIGHT_MOTOR_BW, not is_right_front)
30
31 motor_ctrl(True, False, False, True)
32 time.sleep(3)
33 motor_ctrl(False, False, False, True)
34 time.sleep(3)
35 motor_ctrl(False, False, True, True)
36 time.sleep(1)
37 motor_ctrl(False, True, True, False)
38 time.sleep(3)
39 motor_ctrl(False, False, True, False)
40 time.sleep(3)
41 motor_ctrl(False, False, True, True)
42 time.sleep(1)
43
44 GPIO.cleanup()
```

Python에 함수 문법을 이용하여
제어를 편하게 사용하기 위해 변경된 CODE

Python에서 함수는 'def'로 표현

함수 참조 링크

- <https://github.com/EduProgramming/Python/blob/develop/15def.ipynb>

- <https://wikidocs.net/24>

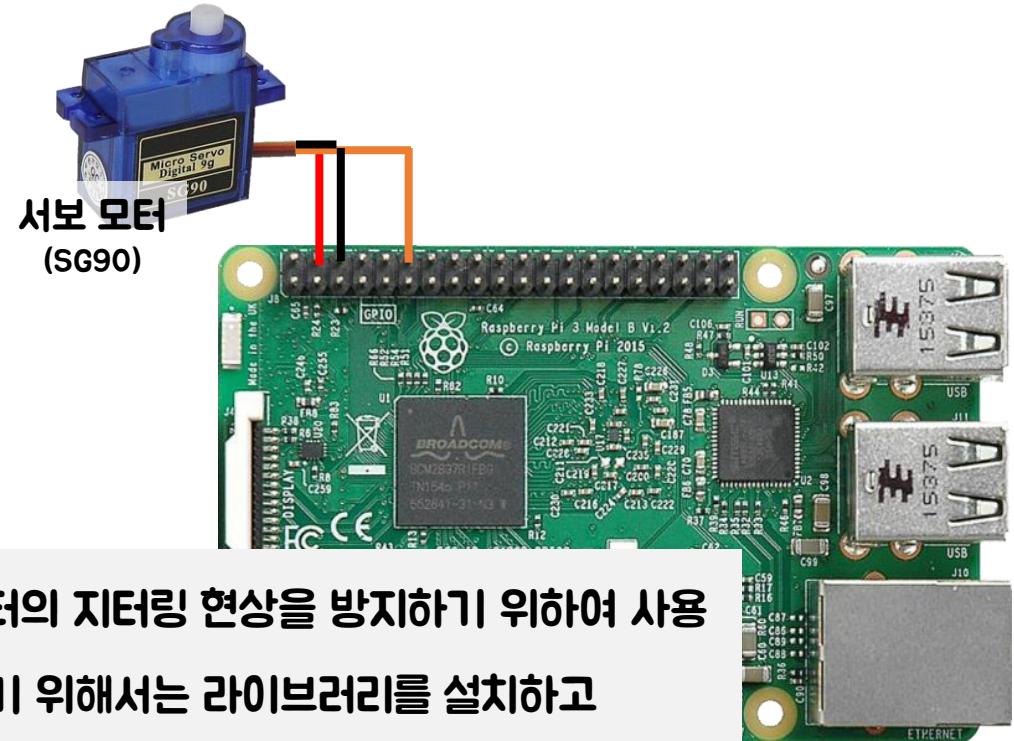
GPIO 제어 with Python

Servo

```
sudo apt-get install pigpio python-pigpio python3-pigpio
```

sudo pigpiod

```
1 import pigpio  
2 import time  
3  
4 PIN_SERVO = 18  
5  
6 servo = pigpio.pi()  
7 servo.set_servo_pulsewidth(PIN_SERVO, 1500)  
8 time.sleep(1)  
9 servo.set_servo_pulsewidth(PIN_SERVO, 2400)  
10 time.sleep(1)  
11 servo.set_servo_pulsewidth(PIN_SERVO, 600)  
12 time.sleep(1)  
13 servo.set_servo_pulsewidth(PIN_SERVO, 1500)  
14 time.sleep(1)
```



pigpiod는 서보모터의 지터링 현상을 방지하기 위하여 사용
pigpiod를 사용하기 위해서는 라이브러리를 설치하고
pigpiod 서버를 실행하면 됩니다.

Pulse 범위: 500 ~ 2500

참고 링크: <http://abyz.me.uk/rpi/pigpio/python.html>

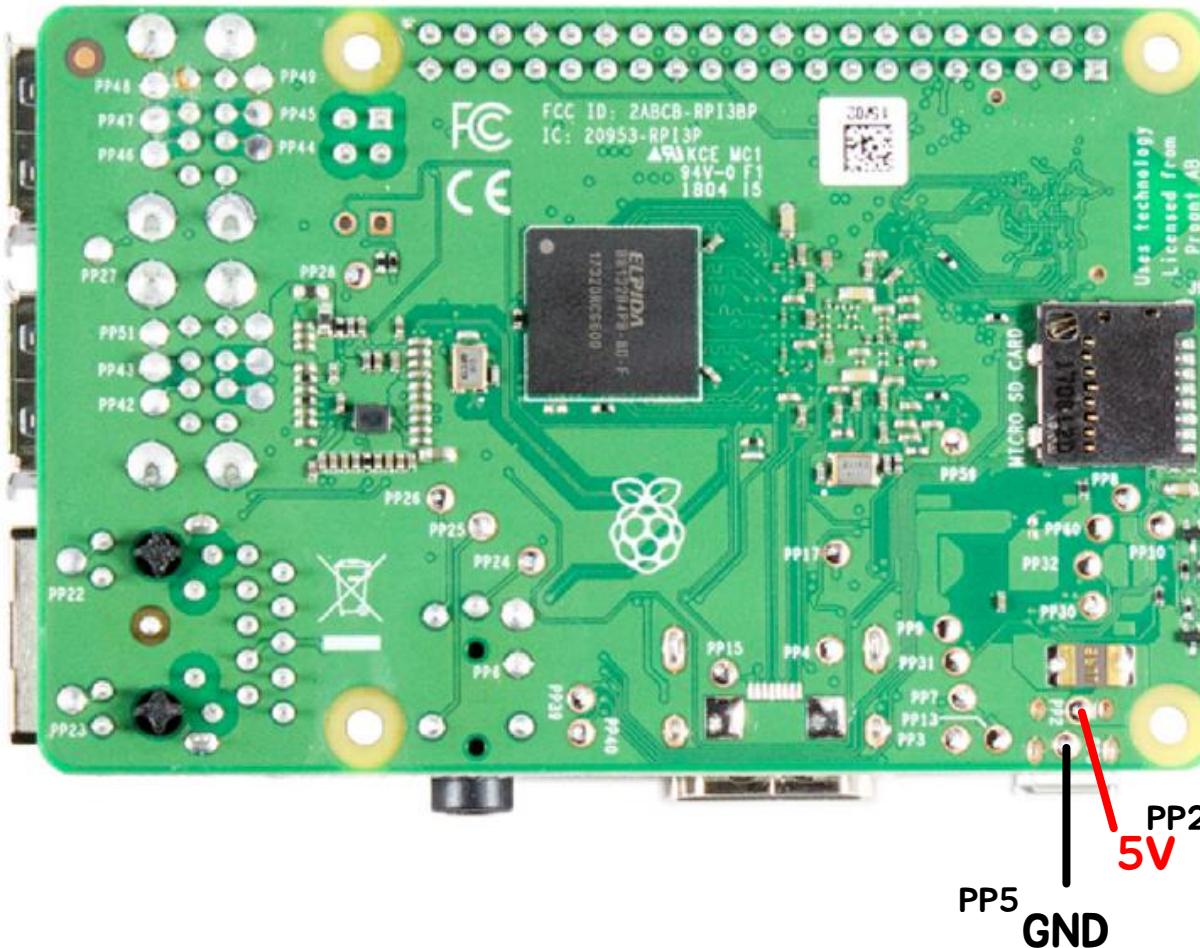
Raspberry Pi 3B +

Camera Module



외부 전력 공급 방법

PP2 / PP5

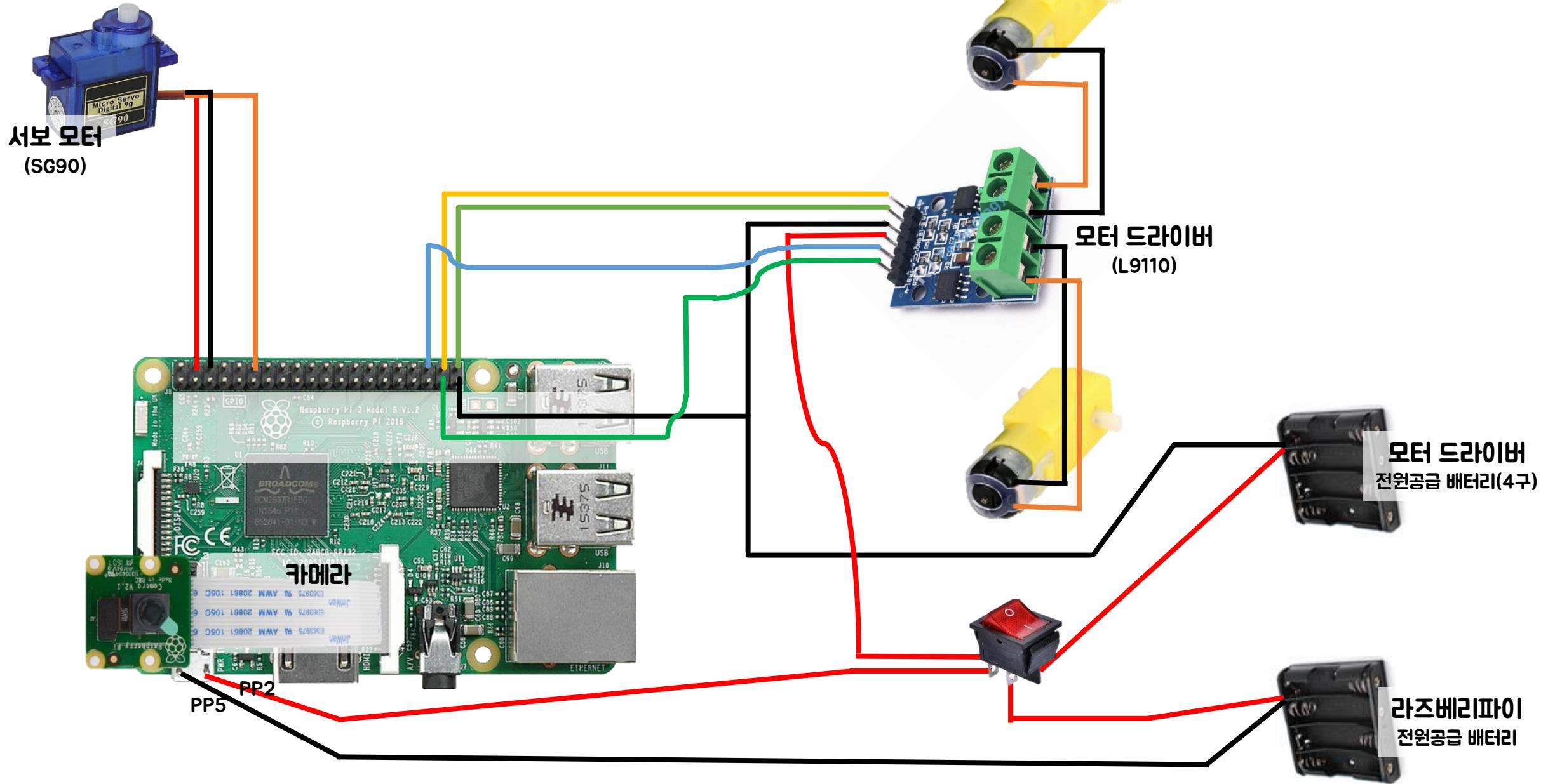


PP2에 **5V** 전원단

PP5에 GND

연결하여 외부 전원으로 작동 시킬 수 있습니다.

전체 회로도



CODE

Flask

파이썬 웹 프레임워크 중 하나로써, 웹서버 역할을 합니다.

프레임워크: 뼈대 역할을 하는 것(간단히, 클래스 + 라이브러리 형태)

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', threaded=True)
```

라즈베리파이 IP:5000 브라우저에 주소를 치면

localhost:5000도 VS Code 진행시 가능했음

Hello, World!

이렇게 나오는 웹 페이지가 나옵니다.

참고링크: <https://flask.palletsprojects.com/en/2.0.x/>

Flask

flask의 render_template을 이용하여 다른 HTML을 열게 설정

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

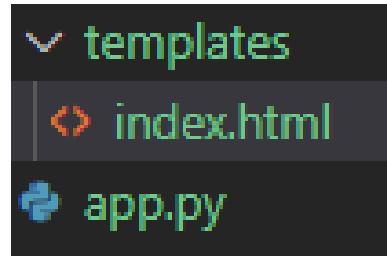
if __name__ == '__main__':
    app.run()
```

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    Hello, RaspberryPi
</body>
</html>
```

Flask

여기서 주의할 점은 HTML 파일의 위치인데 Flask는 기본적으로
'templates' 폴더를 기반으로 html파일을 찾습니다.



App.py 파일이 있는 곳에 'templates' 폴더가 있으며,
해당 폴더 안에는 index.html 파일이 있습니다.

`sudo python app.py`

터미널에 작성시에 서버가 실행됩니다.

127.0.0.1:5000으로 이동하게 되면 아래와 같이 나오게 됩니다.

127.0.0.1:5000

Hello, RaspberryPi

해당 주소는 서버를 키 `local`에서만 사용 가능

이외의 주소에서는 접근이 불가능합니다.

```
import RPi.GPIO as GPIO
import time
from flask import Flask, render_template

app = Flask(__name__)
GPIO.setmode(GPIO.BCM)

PIN_LED = 12

GPIO.setup(PIN_LED, GPIO.OUT)

@app.route('/')
def index():
    return "INDEX"

@app.route('/<int:value>')
def get_int(value):
    return f"숫자로 인식{value}"

@app.route('/<string:state>')
def get_string(state):
    if state.lower() == 'on':
        GPIO.output(PIN_LED, True)
    elif state.lower() == 'off':
        GPIO.output(PIN_LED, False)
    return state

if __name__ == '__main__':
    try:
        app.run()
    except KeyboardInterrupt:
        GPIO.cleanup()
```

GPIO 제어 부분과 Flask를 합쳐서 다뤄보도록 하겠습니다.

Route로 3가지의 경우를 나눠줬습니다.

1) 기본 /: 메인이라고 생각하시면 됩니다.

2) 숫자 /숫자: /뒤에 숫자만 붙으면 get_int 함수 내용을 실행합니다.

3) 문자열 /문자열: /뒤에 문자열이 붙으면 get_string 함수 내용을 실행합니다.

대다수의 경우가 이 영역에 속한다고 보시면 됩니다.

get_string 함수 내용을 보시면 `state.lower()`라는 것은 문자열의 값을 소문자 취급하는 것입니다.

해당 부분을 통해 문자열이 on이라고 오면 GPIO 12번의 LED를 ON하고, off라고 문자열이 오면 GPIO 12번의 LED를 OFF하고 있습니다.

*Route란?

'길'이라는 뜻으로 네트워크 상에서도 어떤 출발지에서 특정 목적지로 가는 길이 여러 갈래가 있습니다.

'@app.route'를 통해서 route를 만들어 주고 만든 길을 통해서 해당 함수를 수행하는 방식입니다.

Final

CODE

app.py

```
1 from importlib import import_module
2 from flask import Flask, render_template, Response, request, jsonify
3 import RPi.GPIO as GPIO
4 import time, json
5 import os
6
7 if os.environ.get('CAMERA'):
8     Camera = import_module('camera_' + os.environ['CAMERA']).Camera
9 else:
10     from camera_cctv import Camera
11
12 app = Flask(__name__)
13
14 GPIO.setmode(GPIO.BCM)
15
16 # Pin 번수 설정
17 LEFT_MOTOR_FW = 20
18 LEFT_MOTOR_BW = 21
19 RIGHT_MOTOR_FW = 16
20 RIGHT_MOTOR_BW = 26
21
22 speed = 0
23
24 # PinMode 설정
25 GPIO.setup(LEFT_MOTOR_FW, GPIO.OUT)
26 GPIO.setup(LEFT_MOTOR_BW, GPIO.OUT)
27 GPIO.setup(RIGHT_MOTOR_FW, GPIO.OUT)
28 GPIO.setup(RIGHT_MOTOR_BW, GPIO.OUT)
29
30 # Global 변수 설정
31 angle = START_ANGLE = 1500
32
33 left_fw_speed = GPIO.PWM(LEFT_MOTOR_FW, 50) #핀번호, 주파수 50Hz
34 left_bw_speed = GPIO.PWM(LEFT_MOTOR_BW, 50) #핀번호, 주파수 50Hz
35 right_fw_speed = GPIO.PWM(RIGHT_MOTOR_FW, 50) #핀번호, 주파수 50Hz
36 right_bw_speed = GPIO.PWM(RIGHT_MOTOR_BW, 50) #핀번호, 주파수 50Hz
37
38 left_fw_speed.start(0)
39 left_bw_speed.start(0)
40 right_fw_speed.start(0)
41 right_bw_speed.start(0)
```

from importlib import import_module

: 동적으로 모듈을 import 하는 방법입니다.

현재 같은 경우는 os.environ.get('CAMERA')의 값을 받아야하는

상황이기에 변화된 값에 적용하기 위해 사용된 import 라이브러리

방식입니다.

from flask import Response, request, jsonify

- **Response**: 응답으로 데이터를 주는 함수

- **request**: 데이터 값을 받기 위해 요청하는 함수

- **jsonify**: 데이터를 json형식으로 치환하기 위한 함수

import json

: json과 관련된 라이브러리

import os

: os(운영체제)와 관련된 라이브러리

*운영체제란? (위키백과 내용 중 일부)

시스템 하드웨어를 관리할 뿐 아니라 응용 소프트웨어를 실행하기 위하여
하드웨어 추상화 플랫폼과 공동 시스템 서비스를 제공하는 시스템 소프트
웨어입니다.

CODE

app.py

```
1  from importlib import import_module
2  from flask import Flask, render_template, Response, request, jsonify
3  import RPi.GPIO as GPIO
4  import time, json
5  import os
6
7  if os.environ.get('CAMERA'):
8      Camera = import_module('camera_' + os.environ['CAMERA']).Camera
9  else:
10     from camera_cctv import Camera
11
12 app = Flask(__name__)
13
14 GPIO.setmode(GPIO.BCM)
15
16 # Pin 변수 설정
17 LEFT_MOTOR_FW = 20
18 LEFT_MOTOR_BW = 21
19 RIGHT_MOTOR_FW = 16
20 RIGHT_MOTOR_BW = 26
21
22 speed = 0
23
24 # PinMode 설정
25 GPIO.setup(LEFT_MOTOR_FW, GPIO.OUT)
26 GPIO.setup(LEFT_MOTOR_BW, GPIO.OUT)
27 GPIO.setup(RIGHT_MOTOR_FW, GPIO.OUT)
28 GPIO.setup(RIGHT_MOTOR_BW, GPIO.OUT)
29
30 # Global 변수 설정
31 angle = START_ANGLE = 1500
32
33 left_fw_speed = GPIO.PWM(LEFT_MOTOR_FW, 50) #핀번호, 주파수 50Hz
34 left_bw_speed = GPIO.PWM(LEFT_MOTOR_BW, 50) #핀번호, 주파수 50Hz
35 right_fw_speed = GPIO.PWM(RIGHT_MOTOR_FW, 50) #핀번호, 주파수 50Hz
36 right_bw_speed = GPIO.PWM(RIGHT_MOTOR_BW, 50) #핀번호, 주파수 50Hz
37
38 left_fw_speed.start(0)
39 left_bw_speed.start(0)
40 right_fw_speed.start(0)
41 right_bw_speed.start(0)
```

os.environ

: 운영체제에 있는 환경변수값을 가져올 수 있습니다.

만약 CAMERA라는 환경변수값이 있으면 camera_ 카메라환경변수.Camera라는 클래스를 이용하고 아니면, camera_cctv.py에 있는 Camera라는 Class를 이용하라고 되어 있습니다.

Class(클래스)의 개념은 프로그래밍 문법에 하나이며, 해당 설명은 이전에 내용들을 알아야 수월하기에 생략하도록 하겠습니다.

<https://wikidocs.net/28> 위키독스의 점프 투 파일 클래스 부분을 참조 하시는 것을 권장 드립니다.

가능하면 이전의 내용부터 아시는 것이 좋습니다.

CODE

app.py

```
43 @app.route('/')
44 def index():
45     global angle
46     angle = START_ANGLE
47     os.system(f'sudo python3 servo_ctrl.py {START_ANGLE}')
48     return render_template('index.html')
49
50 def gen(camera):
51     """Video streaming generator function."""
52     while True:
53         frame = camera.get_frame()
54         yield (b'--frame\r\n'
55                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
56
57
58 @app.route('/video_feed')
59 def video_feed():
60     """Video streaming route. Put this in the src attribute of an img tag."""
61     return Response(gen(Camera()),
62                     mimetype='multipart/x-mixed-replace; boundary=frame')
63
64 @app.route('/motor_forward')
65 def motor_forward():
66     left_bw_speed.ChangeDutyCycle(0)
67     right_bw_speed.ChangeDutyCycle(0)
68     left_fw_speed.ChangeDutyCycle(speed)
69     right_fw_speed.ChangeDutyCycle(speed)
70     return jsonify({"result": True})
71
72 @app.route('/motor_turn_left')
73 def motor_turn_left():
74     left_fw_speed.ChangeDutyCycle(0)
75     right_bw_speed.ChangeDutyCycle(0)
76     left_bw_speed.ChangeDutyCycle(speed)
77     right_fw_speed.ChangeDutyCycle(speed)
78     return jsonify({"result": True})
```

os.system

: 터미널에 명령을 내리듯 os의 system 함수를 통해 명령을 내릴 수 있습니다.

gen 함수

: 해당 함수는 만든은 함수로서, 해당 내용은 카메라의 프레임을 가져와서 Image 값을 순차적으로 보여주는 역할을 한다고 보시면 됩니다.

video_feed 함수에서 해당 부분을 통해서 요청하는 곳에 이미지를 보내고 있습니다.

우리가 보는 것은 카메라 동영상 같지만, 실제로는 이미지들의 결과입니다.

(이 방법을 사용하지 않고 RTSP 통신을 통하는 방식, 다른 방식들도 많이 있습니다.)

CODE

base_camera.py

base_camera와 cctv_camera 2개의 파일은 오픈소스에서 가져와서 사용한 부분입니다. 실행이 되게 수정만 조금한 소스입니다.

```
1 import time
2 import threading
3 try:
4     from greenlet import getcurrent as get_ident
5 except ImportError:
6     try:
7         from thread import get_ident
8     except ImportError:
9         from _thread import get_ident
```

import threading

: Thread를 제어하는 라이브러리

from greenlet import getcurrent as get_ident

: python2 버전에서 사용되던 기법으로 넘어가겠습니다.

from thread import get_ident

: python3에서는 _thread를 사용하며 혹여나 몰라 해당 부분을 추가한 것으로 보입니다.

from _thread import get_ident

: get_ident() → 현재 스레드의 ID를 리턴해주는 함수

_thread는 저수준의 Thread 라이브러리입니다.

CODE

base_camera.py

```
12 class CameraEvent(object):
13     """An Event-like class that signals all active clients when a new frame is
14     available.
15     """
16     def __init__(self):
17         self.events = {}
18
19     def wait(self):
20         """Invoked from each client's thread to wait for the next frame."""
21         ident = get_ident()
22         if ident not in self.events:
23             # this is a new client
24             # add an entry for it in the self.events dict
25             # each entry has two elements, a threading.Event() and a timestamp
26             self.events[ident] = [threading.Event(), time.time()]
27         return self.events[ident][0].wait()
28
29     def set(self):
30         """Invoked by the camera thread when a new frame is available."""
31         now = time.time()
32         remove = None
33         for ident, event in self.events.items():
34             if not event[0].isSet():
35                 # if this client's event is not set, then set it
36                 # also update the last set timestamp to now
37                 event[0].set()
38                 event[1] = now
39             else:
340                 # if the client's event is already set, it means the client
341                 # did not process a previous frame
342                 # if the event stays set for more than 5 seconds, then assume
343                 # the client is gone and remove it
344                 if now - event[1] > 5:
345                     remove = ident
346         if remove:
347             del self.events[remove]
348
349     def clear(self):
350         """Invoked from each client's thread after a frame was processed."""
351         self.events[get_ident()][0].clear()
```

CameraEvent 클래스에서는

이벤트가 발생하면 해당 Thread를 잠시 기다리게 하고, 지정하고 삭제하고 비워주는 역할을 하고 있습니다.

CODE base_camera.py

```
54 class BaseCamera(object):
55     thread = None # background thread that reads frames from camera
56     frame = None # current frame is stored here by background thread
57     last_access = 0 # time of last client access to the camera
58     event = CameraEvent()
59
60     def __init__(self):
61         """Start the background camera thread if it isn't running yet."""
62         if BaseCamera.thread is None:
63             BaseCamera.last_access = time.time()
64
65             # start background frame thread
66             BaseCamera.thread = threading.Thread(target=self._thread)
67             BaseCamera.thread.start()
68
69             # wait until frames are available
70             while self.get_frame() is None:
71                 time.sleep(0)
72
73     def get_frame(self):
74         """Return the current camera frame."""
75         BaseCamera.last_access = time.time()
76
77         # wait for a signal from the camera thread
78         BaseCamera.event.wait()
79         BaseCamera.event.clear()
80
81         return BaseCamera.frame
82
83     @staticmethod
84     def frames():
85         """Generator that returns frames from the camera."""
86         raise RuntimeError('Must be implemented by subclasses.')
```

`__init__` 생성자 함수에서 `Thread`을 생성하고 시작합니다.

그리고 `get_frame` 함수에서 마지막 접근 시간을 담고, `event`를 멈추고 `clear`하는 행위를 여기서 하고 있습니다.
`return BaseCamera.frame`인가 보니

```
def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

`app.py`에서 있던 위의 부분에 `frame`이 해당 요소인 것을 우리는 알 수 있습니다.

`frames` 함수와 `_thread` 함수는 크게 이해하지 않고 넘어가도 괜찮을 거 같습니다.
넘어가도록 하겠습니다.

CODE

camera_cctv.py

```
1 import io
2 import time
3 import picamera
4 from base_camera import BaseCamera
5
6
7 class Camera(BaseCamera):
8     @staticmethod
9     def frames():
10         with picamera.PiCamera() as camera:
11             # let camera warm up
12             time.sleep(2)
13
14             stream = io.BytesIO()
15             for _ in camera.capture_continuous(stream, 'jpeg',
16                                              use_video_port=True):
17                 # return current frame
18                 stream.seek(0)
19                 yield stream.read()
20
21             # reset stream for next frame
22             stream.seek(0)
23             stream.truncate()
```

import io

: Thread를 제어하는 라이브러리

import picamera

: 라즈베리파이 카메라를 사용하는 라이브러리입니다.

with picamera.PiCamera() as camera: 라고 하여서

with문을 써서 해당 indent(들여쓰기) 영역이 끝나면 알아서 close 됩니다.

as를 써서 해당 부분의 별칭을 camera라고 해서 사용하였고, 캡처한 이미지를 jpeg
로 만들어 사용하였습니다.

From base_camera import BaseCamera

위에서 다뤘던 base_camera.py에 BaseCamera Class를 사용하겠다고 가져왔습니다.

CODE

servo_ctrl.py

```
1 import pigpio  
2 import sys  
3  
4 PIN_SERVO = 18  
5 cmd = int(sys.argv[1])  
6 servo = pigpio.pi()  
7 servo.set_servo_pulsewidth(PIN_SERVO, cmd)
```

Servo 지터링을 없애기 위해서 사용한 pigpio 방식이니 넘어가도록 하겠습니다.

이외의 소스적인 부분은 웹 프로그래밍에 대해서 알아야 이해할 수 있는 부분입니다.

해당 부분들을 하나하나 소개하기에는 요소가 많아서 사용한 언어와 라이브러리가 무엇인지 소개하도록 하겠습니다.

[언어]

- HTML: 웹 사이트를 만들 때 골격을 세우는 역할을 한다고 보시면 됩니다.
- CSS: 웹 사이트에 디자인을 꾸미는 요소입니다.
- JavaScript: 웹 사이트에 액션을 주게 됩니다.

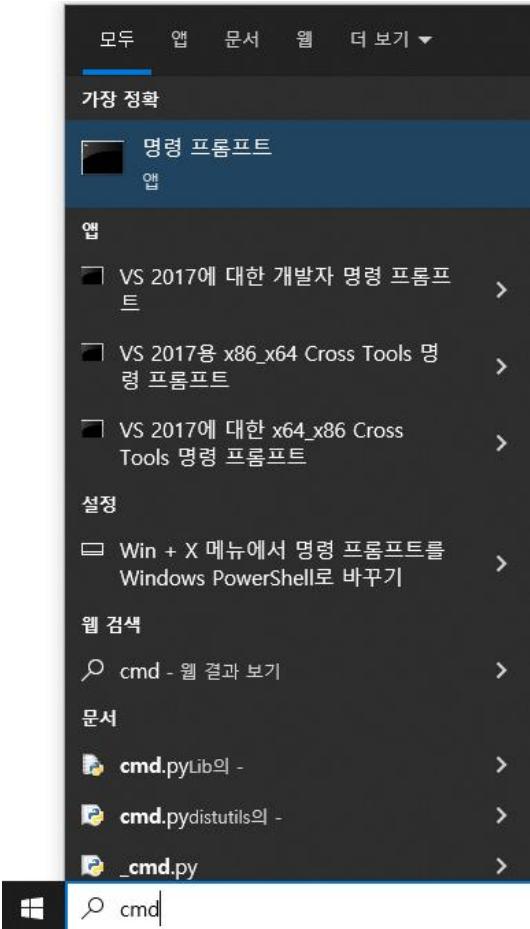
[라이브러리]

- Bootstrap4: CSS를 편리하게 하기 위한 라이브러리
 미리 정해져 있기에 해당 방안으로 진행하면 됩니다.
 단, 정해져 있기에 필요 없는 부분이 겹칠 수 있습니다.
- Axios: 비동기 통신을 편하게 하기 위해 사용하는 라이브러리

Port Forwarding

포트포워딩의 예시로 **iptime** 공유기를 사용하겠습니다.

다른 공유기도 동일하나, 사용하는 공유기마다 다른 부분은 존재합니다.



먼저, 명령 프롬프트를 켜주세요.

'cmd' 라고 검색하면 명령 프롬프트가 나오게 됩니다.

A screenshot of a Windows Command Prompt window titled '명령 프롬프트'. The window shows the output of the 'ipconfig' command. The output includes information about network adapters, such as '이더넷 어댑터 이더넷:' and its details: '연결별 DNS 접미사', '링크-로컬 IPv6 주소 : fe80::99cb:87b5:4aed:748%10', 'IPv4 주소 : 192.168.0.7', '서브넷 마스크 : 255.255.255.0', and '기본 게이트웨이 : 192.168.0.1'. The word '기본 게이트웨이' is highlighted with a red rectangle.

명령 프롬프트에 'ipconfig'라고 작성 후에 엔터를 누르면

연결된 인터넷 관련 정보가 나오게 됩니다.

해당 정보에서 기본 게이트웨이 주소를 인터넷에 작성해주시면 됩니다.

Port Forwarding

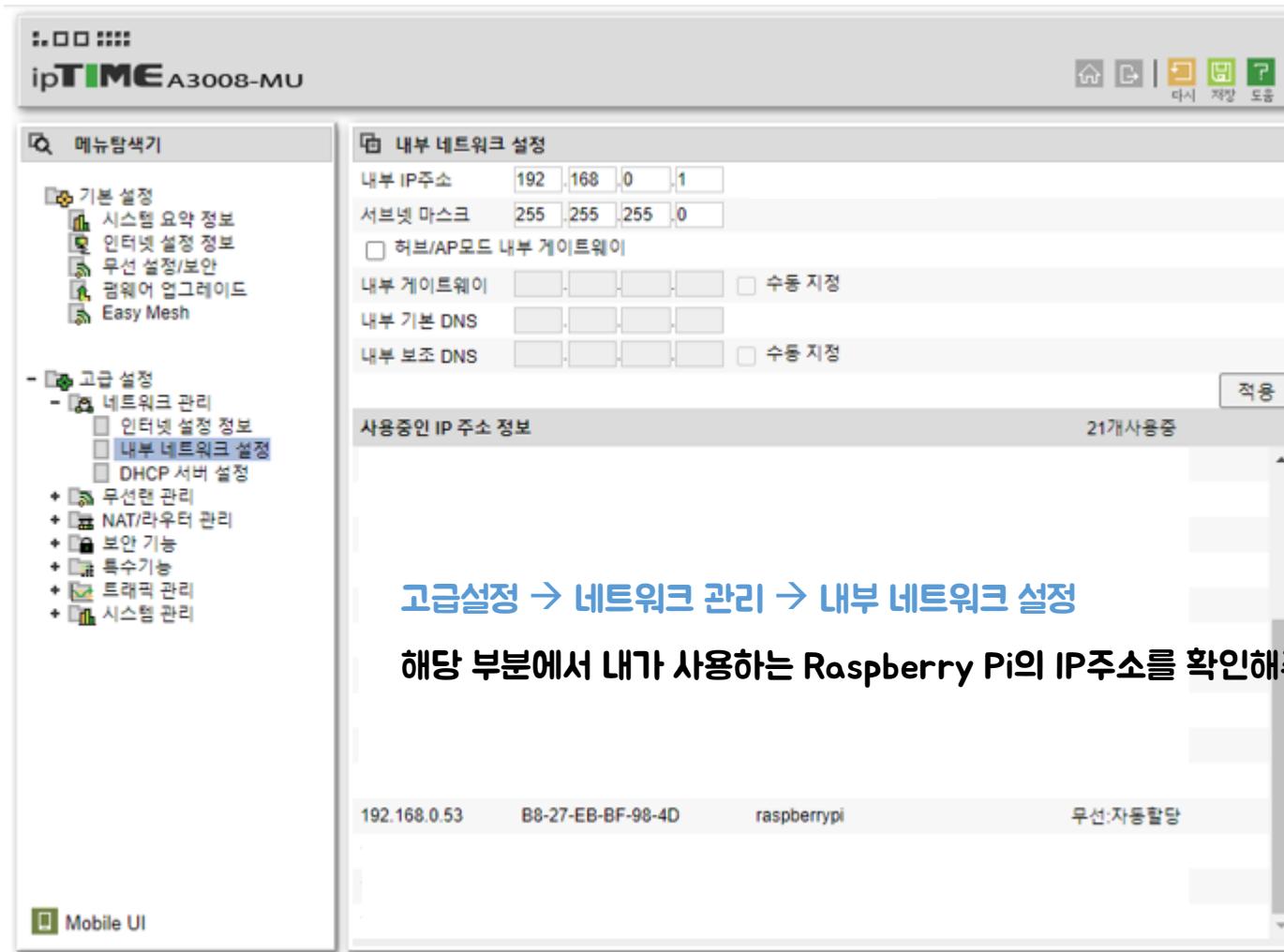


주소를 작성하였더니 공유기 관리자 계정으로 들어갈 수 있게 되었습니다.
관리자 계정과 비밀번호를 작성하여 주세요.
초기 iptime의 경우에는 계정: [admin](#) 암호: [admin](#) 입니다.



로그인을 하게 되면 위와 같은 화면이 나오게 됩니다.
(사용하시는 iptime 마다 상이할 순 있습니다.)
관리도구로 들어가주세요.

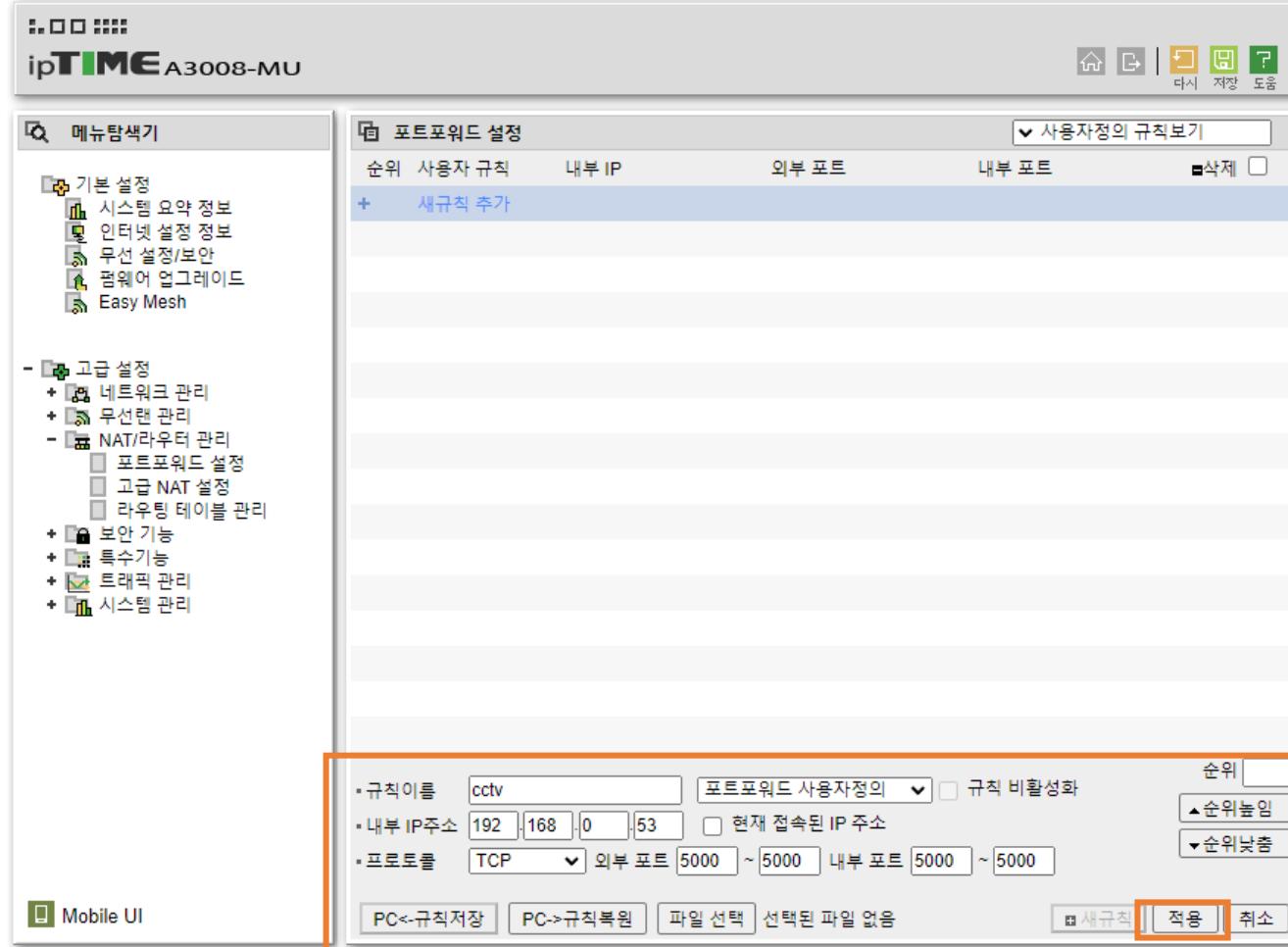
Port Forwarding



고급설정 → 네트워크 관리 → 내부 네트워크 설정

해당 부분에서 내가 사용하는 Raspberry Pi의 IP주소를 확인해주세요.

Port Forwarding



고급설정 → NAT/라우터 관리 → 포트포워드 설정

아래에 규칙이름과 내부 IP주소, 외부 포트, 내부포트를 다 채워주세요.

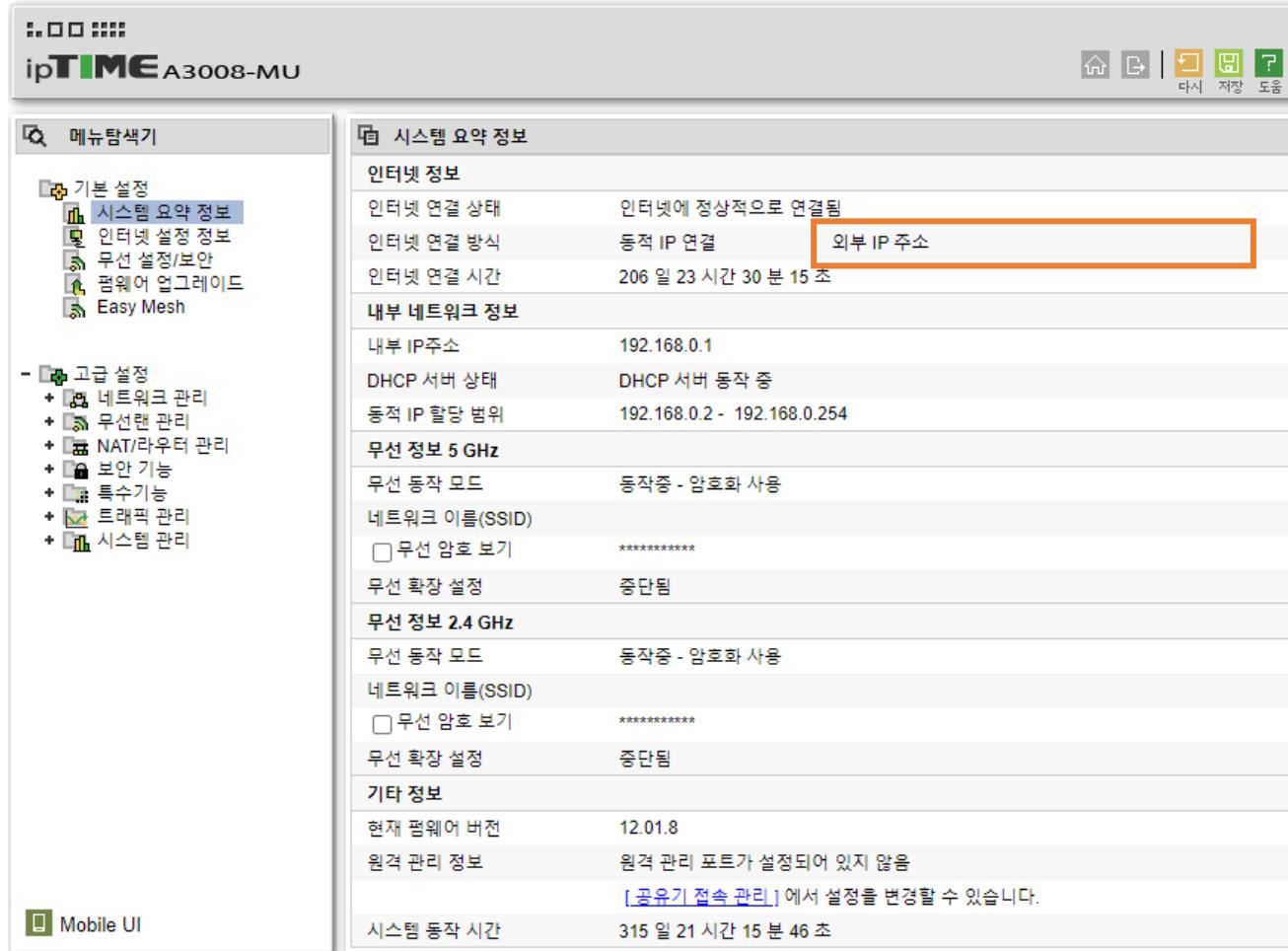
내부 포트의 경우에는 라즈베리파이에서 웹 서버 실행되는 포트이며,

외부 포트는 외부에서 접근할때의 포트입니다.

보안을 위해서는 내부 포트 주소와 외부 포트 주소는 다르게 설정하시는
것이 좋습니다.

작성을 다 하셨으면 적용을 눌러 주세요.

Port Forwarding



기본 설정 → 시스템 요약 정보

외부 IP 주소를 통해서 접근하시면 됩니다.

지금까지는 라즈베리파이 내부IP:5000 으로 접근하였는데

이제는 외부 IP주소를 작성한 후에 :외부포트 를 작성하시면 됩니다.

예를 들어 제 외부 IP주소가 111.111.111.1110이라고 하였을 때,

포트포워딩 할때 외부 포트 주소가 1234라고 한다면

111.111.111.111:1234 라고 검색할 곳에 작성하게 되면

기존에 했던 웹 서버에 접속되게 되며 진정한 IoT로 사용될 수 있습니다.

참고적으로 외부 IP주소는 타인에게 공개하지 않으시는 것을
권장 드립니다.