



Porting Manual

사용 도구 및 버전

[협업 도구]

[개발 도구]

[개발 환경]

TIZEN 포팅 매뉴얼

OS 설치

개발을 위해서는 SDB나 별도의 설정 필요

PC - rasp(tizen) SDB 연결 및 wifi plug-in설치

1. SDB연결

2. Wifi plug in 설치

만약, 다음날 nmap이 안 잡힌다면?

App 포팅

Arduino 설정

프론트엔드 포팅메뉴얼

node.js LTS version 설치(v20.11.0)

주요 라이브러리 및 플러그인

개발 도구 및 환경 설정

Vite 구성

빌드 및 배포

Docker Compose

Jenkinsfile

Backend Dockerfile

Frontend Dockerfile

Machine learning Dockerfile

nginx.conf

사용 도구 및 버전

[협업 도구]

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost, Webex, Discord

- 디자인 : Figma
- CI/CI : Jenkins

[개발 도구]

- Visual Studio Code
- IntelliJ : 2023.3.2
- Tizen Studio : 5.5

[개발 환경]

Backend

Java=17

JKD=17.0.10

Spring boot = 3.2.3

Gradle = 8.6

Packaging = Jar

Tizen os = 8.0

Frontend

- 언어 : TypeScript
- 프레임워크 : React
- 패키지 관리자 : npm

Server

AWS EC2 Ubuntu

Service

Mysql=8.3.0

NginX

Jenkins

TIZEN 포팅 매뉴얼

OS 설치

tizen docs : <https://docs.tizen.org/platform/developing/flashing-rpi/>

1. 리눅스 환경 준비

- tizen image를 sd카드에 flash하기 위해서는 linux 18.04 LTS 이상의 환경이 필요
- WSL은 sd card 등의 device를 인식하지 못하므로 멀티부트를 통한 리눅스 환경이나 vm을 이용
 - ▼ vm 환경 세팅 방법

virtualbox 홈페이지 → downloads :

<https://www.virtualbox.org/wiki/Downloads>

VirtualBox 7.0.14 platform packages

- [Windows hosts](#)
- [macOS / Intel hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)
- [Solaris 11 IPS hosts](#)

The binaries are released under the terms of the GPL version 3.

See the [changelog](#) for what has changed.

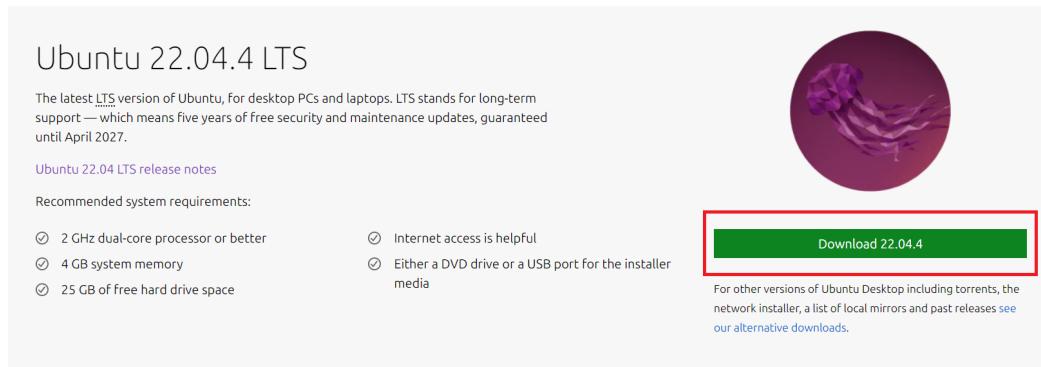
You might want to compare the checksums to verify the integrity of downloaded ;
checksums should be favored as the MD5 algorithm must be treated as insecure!

- [SHA256 checksums](#), [MD5 checksums](#)

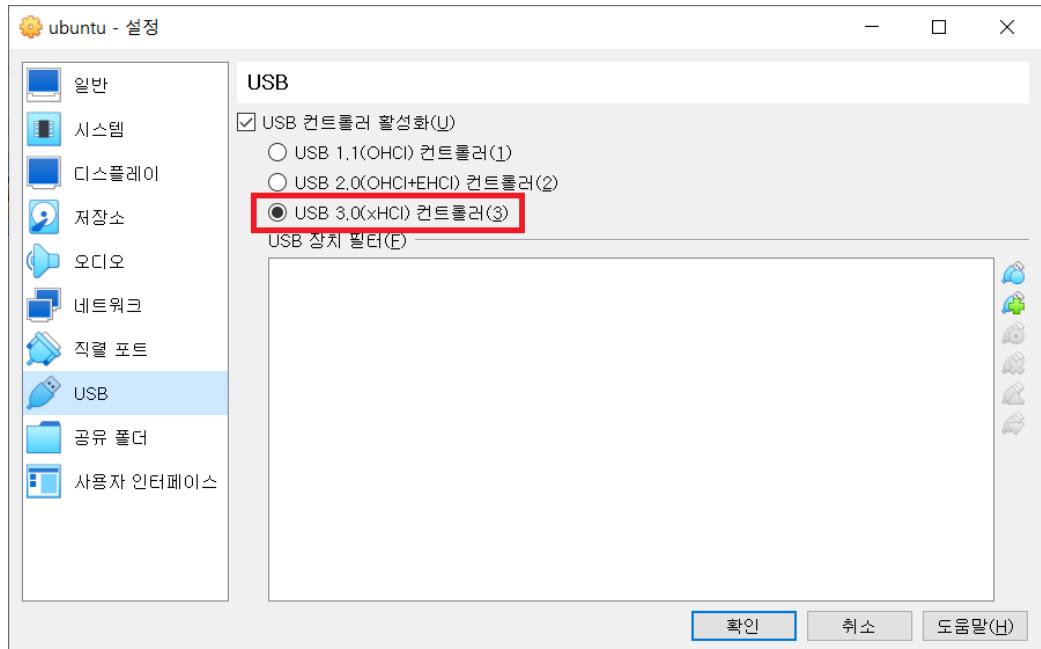
Note: After upgrading VirtualBox it is recommended to upgrade the guest additio

VirtualBox 7.0.14 Oracle VM VirtualBox Extension Pack

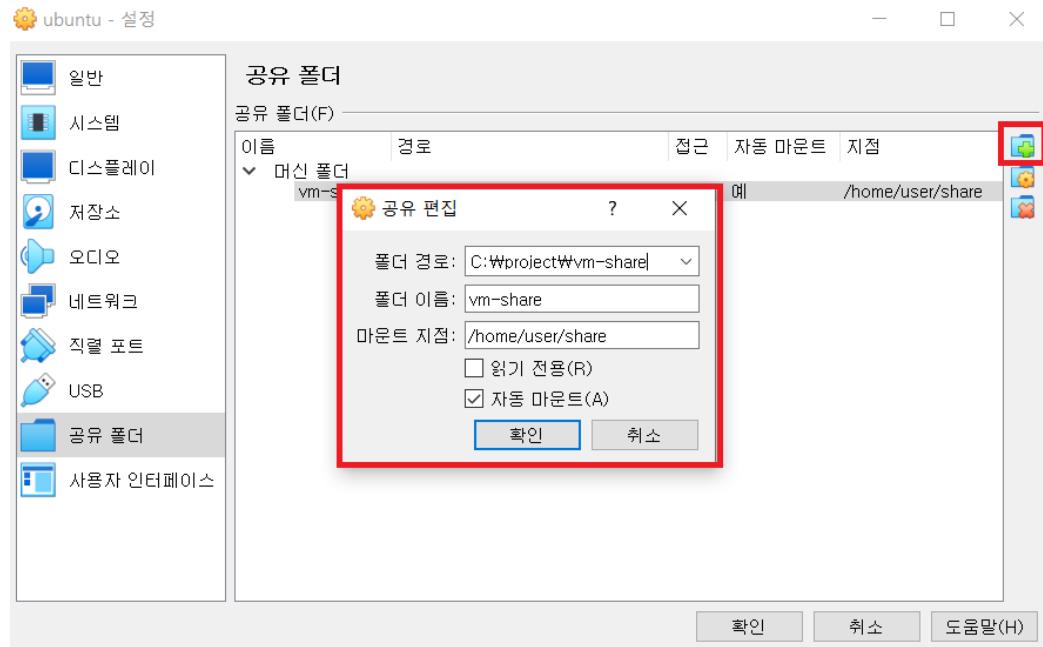
- [All supported platforms](#)



- virtualbox 다운 프로그램 실행 후 iso를 이용하여 ubuntu 설치
- extension pack도 설치

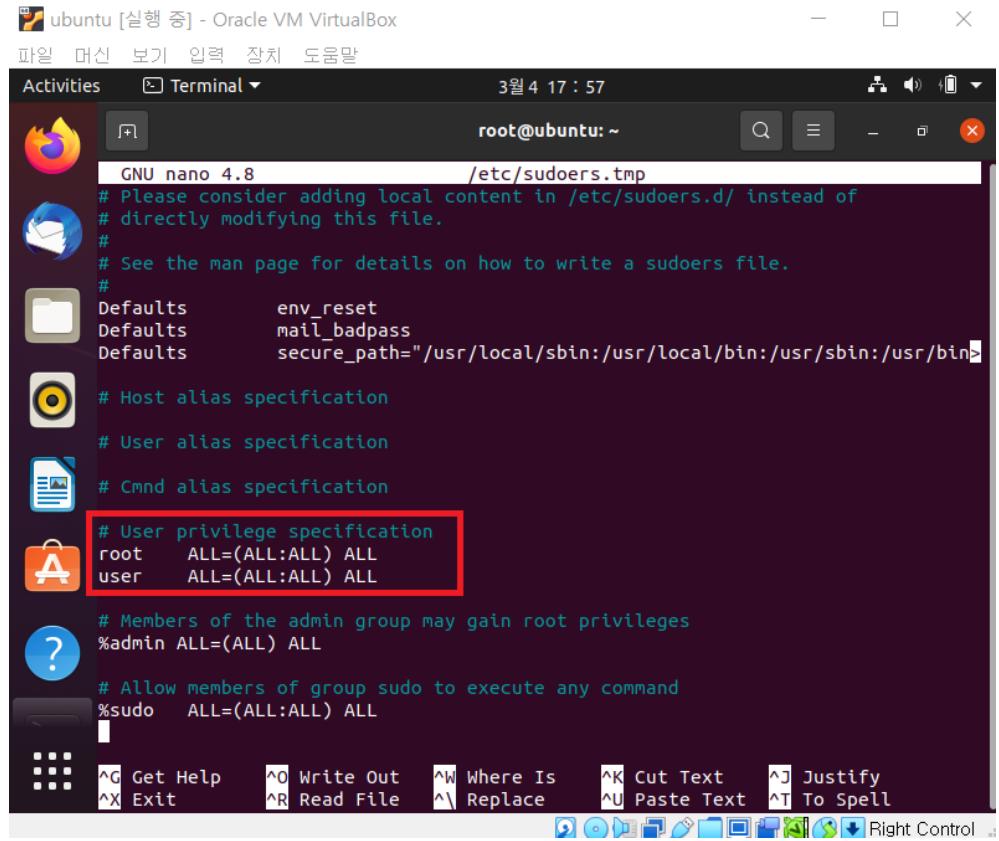


- virtualbox에서 설정 → usb → usb 3.0 컨트롤러 선택 후 저장



- 설정 → 공유폴더 → 추가
- 폴더 경로 : host 컴퓨터의 경로
- 폴더 이름 : path 최종 directory name
- 마운트 지점 : vm 내의 경로, /home/username/{원하는폴더이름}
- vm 내에서 해당 폴더 명으로 직접 생성
- 자동 마운트 체크

▼ sudo 권한 없을 때

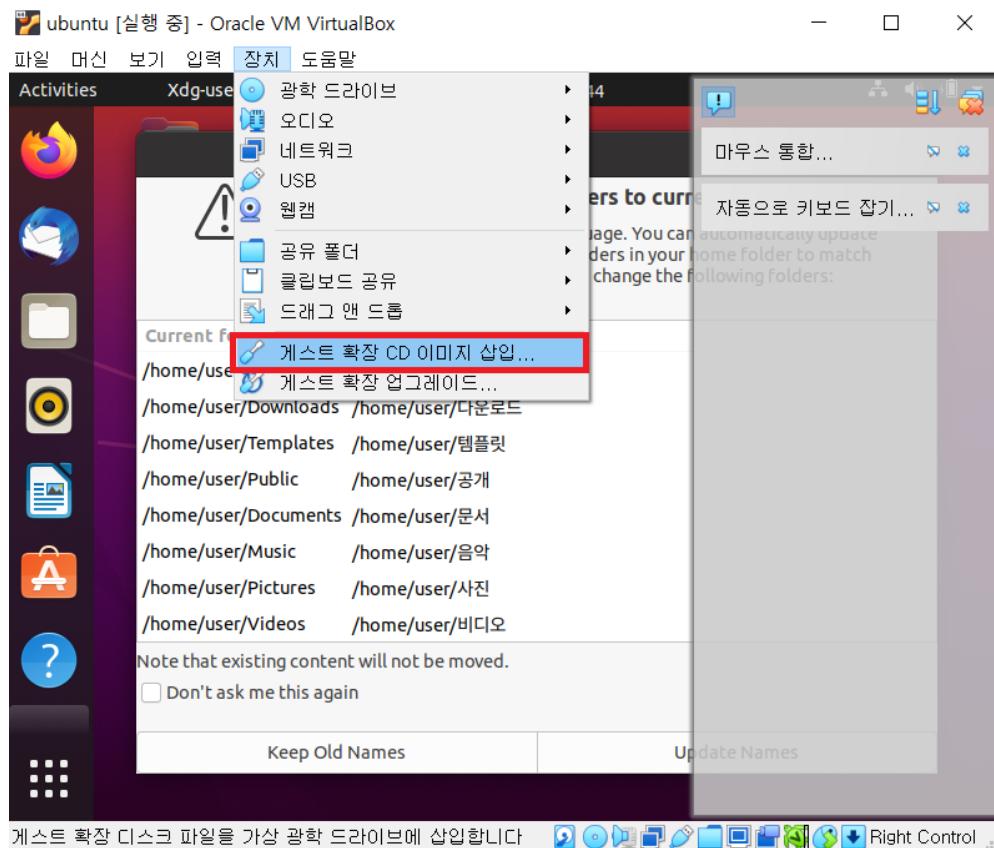


```
GNU nano 4.8 /etc/sudoers.tmp
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin"
#
# Host alias specification
#
# User alias specification
#
# Cmnd alias specification
#
# User privilege specification
root    ALL=(ALL:ALL) ALL
user    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

- su root로 root 계정으로 접속 후 visudo /etc/sudoers
- 해당 부분의 root 아래에 가상환경 유저 이름 ALL=(ALL:ALL) ALL 작성 및 저장



- 게스트 확장 CD 이미지 삽입 → 설치 후 재부팅
- 공유 폴더를 사용할 수 있게 된다

2. 바이너리 다운로드

- 디스플레이가 없는 제품 개발 시 headless, 디스플레이가 있을 시 headed인 부팅, 플랫폼 이미지를 다운로드
- vm 환경일 경우 호스트에서 다운받은 부팅 이미지를 공유폴더를 이용하여 복사

3. 이미지 Flash

```
sudo apt-get install pv

wget https://git.tizen.org/cgit/platform/kernel/u-boot/pla
chmod 755 sd_fusing_rpi3.sh
```

- pv 패키지 설치

- rpi용 스크립트 설치(rpi3, rpi4는 동일)

▼ SD 카드 장치 노드 확인

```
ls -al /dev/sd*
```

- SD 카드 삽입 전 상태 확인
- SD 카드 삽입 후 동일한 명령어 실행하여 노드 확인

```
# SD 카드 삽입 전
```

```
brw-rw---- 1 root disk 8, 0 9 18 09:08 /dev/sda
brw-rw---- 1 root disk 8, 1 9 18 09:08 /dev/sda1
brw-rw---- 1 root disk 8, 2 9 18 09:08 /dev/sda2
brw-rw---- 1 root disk 8, 5 9 18 09:08 /dev/sda5
```

```
# SD 카드 삽입 후
```

```
brw-rw---- 1 root disk 8, 0 9 18 09:08 /dev/sda
brw-rw---- 1 root disk 8, 1 9 18 09:08 /dev/sda1
brw-rw---- 1 root disk 8, 2 9 18 09:08 /dev/sda2
brw-rw---- 1 root disk 8, 5 9 18 09:08 /dev/sda5
brw-rw---- 1 root disk 8, 16 9 22 14:59 /dev/sdb
brw-rw---- 1 root disk 8, 17 9 22 14:59 /dev/sdb1
brw-rw---- 1 root disk 8, 18 9 22 14:59 /dev/sdb2
brw-rw---- 1 root disk 8, 19 9 22 14:59 /dev/sdb3
```

- 위와 같은 경우 SD카드의 노드는 /dev/sdb 가 된다.

```
# 코드 양식
```

```
sudo ./sd_fusing_rpi3.sh -d <SD card device node> --format
sudo ./sd_fusing_rpi3.sh -d <SD card device node> -b <Boot
```

```
# 예시
```

```
sudo ./sd_fusing_rpi3.sh -d /dev/sdb --format
sudo ./sd_fusing_rpi3.sh -d /dev/sdb -b tizen-unified_2022
```

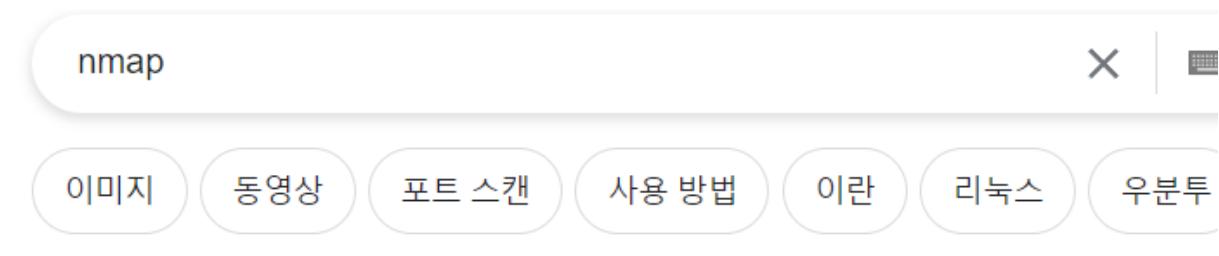
- 위의 양식에서 본인에게 해당하는 노드 번호, 부팅/플랫폼 파일을 입력한 후 실행
- format 후 설치까지 진행됨

개발을 위해서는 SDB나 별도의 설정 필요

PC - rasp(tizen) SDB 연결 및 wifi plug-in설치

1. SDB연결

1. nmap설치. 아래 사진들처럼 진행하여 nmap을 설치한다.



검색결과 약 13,500,000개 (0.24초)



Nmap: the Network Mapper - Free Security Scanner

Nmap ("Network Mapper") is a free and open source utility for network discovery and auditing. Many systems and network administrators also find it ...

[Download the Free Nmap...](#) · [Reference Guide](#) · [Documentation page](#) · [Zenmap](#)



Nmap.org

Npcap.com Seclists.org Sectools.org Insecure.org

Site Search

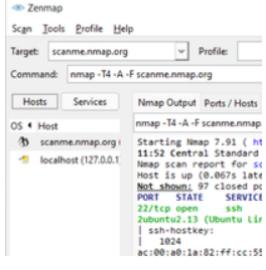
Download Reference Guide Book Docs Zenmap GUI In the Movies

Downloading Nmap

Get the latest Nmap for your system:

- [Windows](#)
- [macOS](#)
- [Linux \(RPM\)](#)
- [Any other OS \(source code\)](#)

Microsoft Windows binaries



Please read the [Windows section](#) of the Install Guide for limitations and installation instructions for the Windows version of Nmap. It's provided as an executable self-installer which includes Nmap's dependencies and the Zenmap GUI. We support Nmap on Windows 7 and newer, as well as Windows Server 2008 R2 and newer. We also maintain a [guide for users who must run Nmap on earlier Windows releases..](#)

Note: The version of Npcap included in our installers may not always be the latest version. If you experience problems or just want the latest and greatest version, download and install [the latest Npcap release.](#)

Latest stable release self-installer: [nmap-7.94-setup.exe](#)

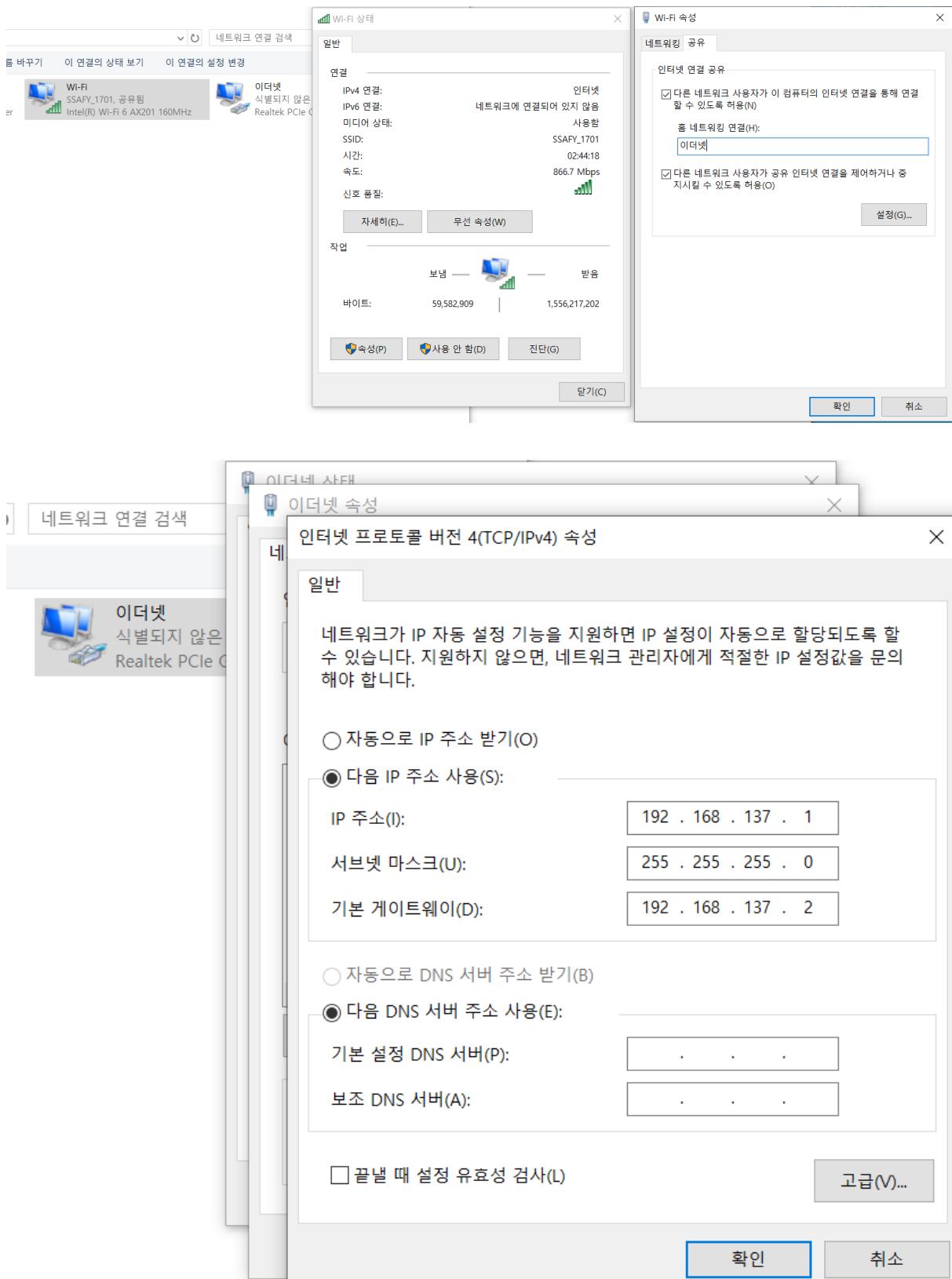
Latest Npcap release self-installer: [npcap-1.79.exe](#)

We have written [post-install usage instructions](#). Please [notify us](#) if you encounter any problems or have suggestions for the installer.

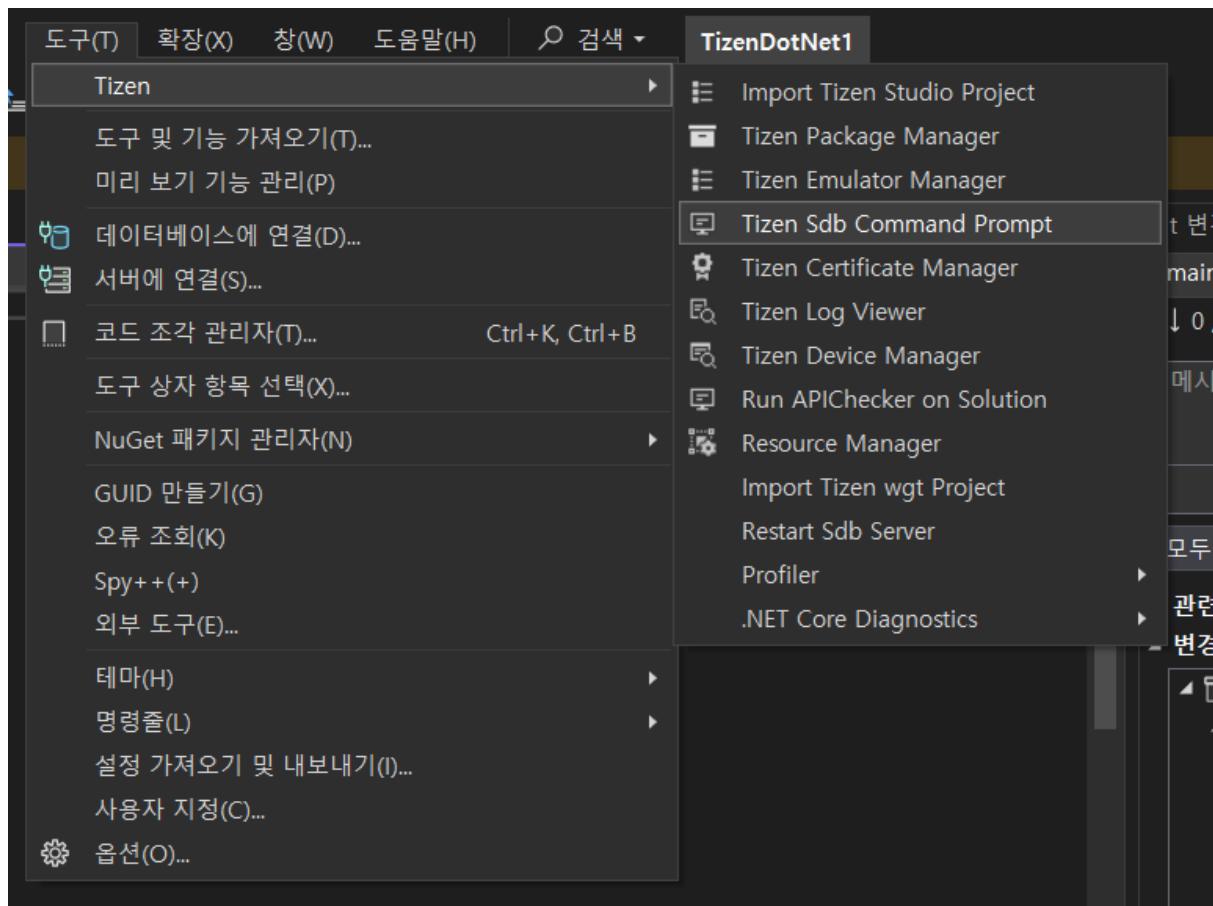
다운 받은 설치 파일을 설치한다.

2. 라즈베리파이와 노트북을 LAN선을 이용해 연결한다.

3. PC에서 인터넷 설정을 아래와 같이 진행한다.



4. sdb.exe가 있는 위치에 명령 프롬프트 창을 연다.



5. 원도우 키 + R을 눌러서 cmd를 입력하고, cmd창을 연다.

nmap을 이용해 PC에서 라즈베리파이의 IP를 찾는다.

```
관리자: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SSAFY>nmap -sP 192.168.137.1/24
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-12 12:51 대한민국 표준시
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify
servers with --dns-servers.
Nmap scan report for 192.168.137.132
Host is up (0.00048s latency).
MAC Address: D8:3A:DD:53:D6:D7 (Unknown)
Nmap scan report for 192.168.137.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 9.85 seconds
C:\Users\SSAFY>
```

6. 찾은 IP를 기반으로 sdb연결을 한다.

```
관리자: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Tizen\tools>sdb.exe connect 192.168.137.132
connecting to 192.168.137.132:26101 ...
connected to 192.168.137.132:26101

C:\Tizen\tools>
```

2. Wifi plug in 설치

참고자료

<https://docs.tizen.org/iot/get-started/rpi3-5.0/>

sdb명령어 모음

<https://docs.tizen.org/application/tizen-studio/common-tools/smart-development-bridge/>

tizen을 설치하고, wifi를 켜면 안 잡히는 문제가 있다.

이를 해결하기 위해 plug in을 sdb를 통해 설치해야한다.

1. 먼저 plug-in을 다운 받는다.

<https://developer.samsung.com/tizen/TizenDeviceFirmware.html>

회원가입을 필수로 진행해야 한다.

라즈베리파이에 설치한 Tizen버전에 맞는 plug-in을 설치 받는다.

(여기선 Tizen8.0을 설치했다)

2. sdb명령어를 통해 다운로드한 zip파일을 라즈베리파이로 옮긴다.

```

# 먼저 제한이 없게 root권한을 부여 받는다.
C:\Tizen\tools> sdb root on

# 다운로드 받은 폴더에서 cmd를 켜고, 라즈베리파이로 파일을 옮긴다.
C:\경로> sdb push [다운받은 파일] [라즈베리파이 경로]

#예시) sdb push RPI3n4_plugin_tizen-8.0.zip /opt/usr/home/own

# 타이젠의 shell로 이동, 이때 root: ~>가 떠야한다.
C:\경로> sdb shell
cd /opt/usr/home/owner
unzip RPI3n4_plugin_tizen-8.0.zip
cd RPI3n4_plugin_tizen-8.0
chmod 755 ./install_on_target.sh
install_on_target.sh

```

이제 라즈베리파이의 전원을 빼고, 다시 연결한다.

wifi가 정상 작동하는 것을 볼 수 있다.

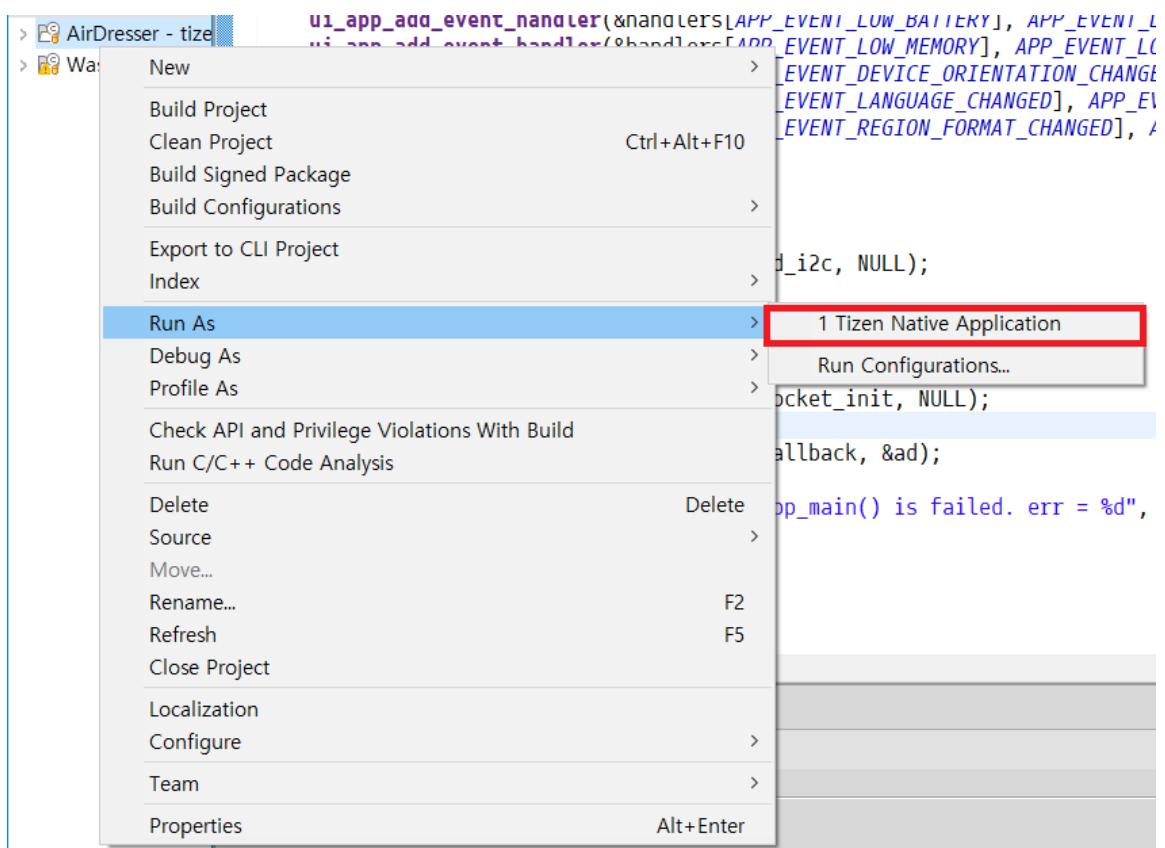
만약, 다음날 nmap이 안 잡힌다면?

다시 랜을 연결하고 nmap을 통해 IP를 찾으면 192.168.137.1만 나올 수 있다.
이런 경우, 네트워크 설정을 다시 해주어야한다.

1. 네트워크 변경에서 WIFI를 클릭한다.
2. 속성 > 공유 >
3. 첫번째 박스인 “다른 네트워크 사용자~”를 해제하고 확인을 누른다.
4. 다시 속성 > 공유까지 진행한다.
5. 해제한 박스를 다시 활성화 한다. 이때 홈 네트워킹 연결은 이더넷으로 바꿔준다.
6. 네트워크 변경에서 이더넷을 클릭한다.
7. 속성 > 네트워킹 > TCP/IP4를 선택한다.
8. 게이트 웨이를 전과 동일하게 세팅한다.
9. 다시 nmap을 해보자

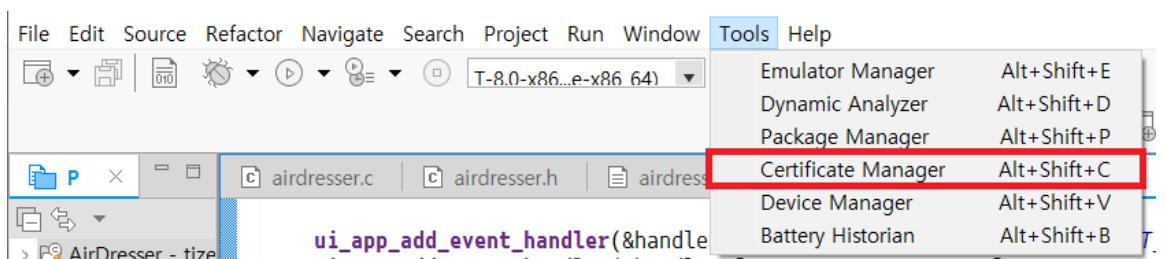
App 포팅

1. Tizen studio에서 tizen project를 open
2. sdb로 raspberry pi에 연결
3. 프로젝트 우클릭 -> Run As → Tizen Native Application



4. 설치 및 실행이 진행된다.

▼ 실행이나 설치가 되지 않을 경우 인증서 발급이 필요할 수 있다.





Preferences

You can create, select, remove and edit certificate profiles.

You can also change the certificate information of the certificate profile.

App Signing

Certificate Profile



TStest

aa

Author Certificate

a

Verified by: Tizen Developers CA



Distributor Certificate

tizen-distributor-signer-new

Verified by: Tizen Studio



Close

Create Certificate Profile

Certificate Profile Author Certificate ③ Distributor Certificate

A distributor issues a certificate of its own to developers. And the distributor checks if an application submitted to its app store includes its distributor certificate.

Use the default Tizen distributor certificate

The Tizen Studio offers a default Tizen distributor certificate, which you can use to submit your application to the Tizen Store.

Privilege level Platform ▾

You can use public, partner, and platform level APIs.

Version New - Default ▾

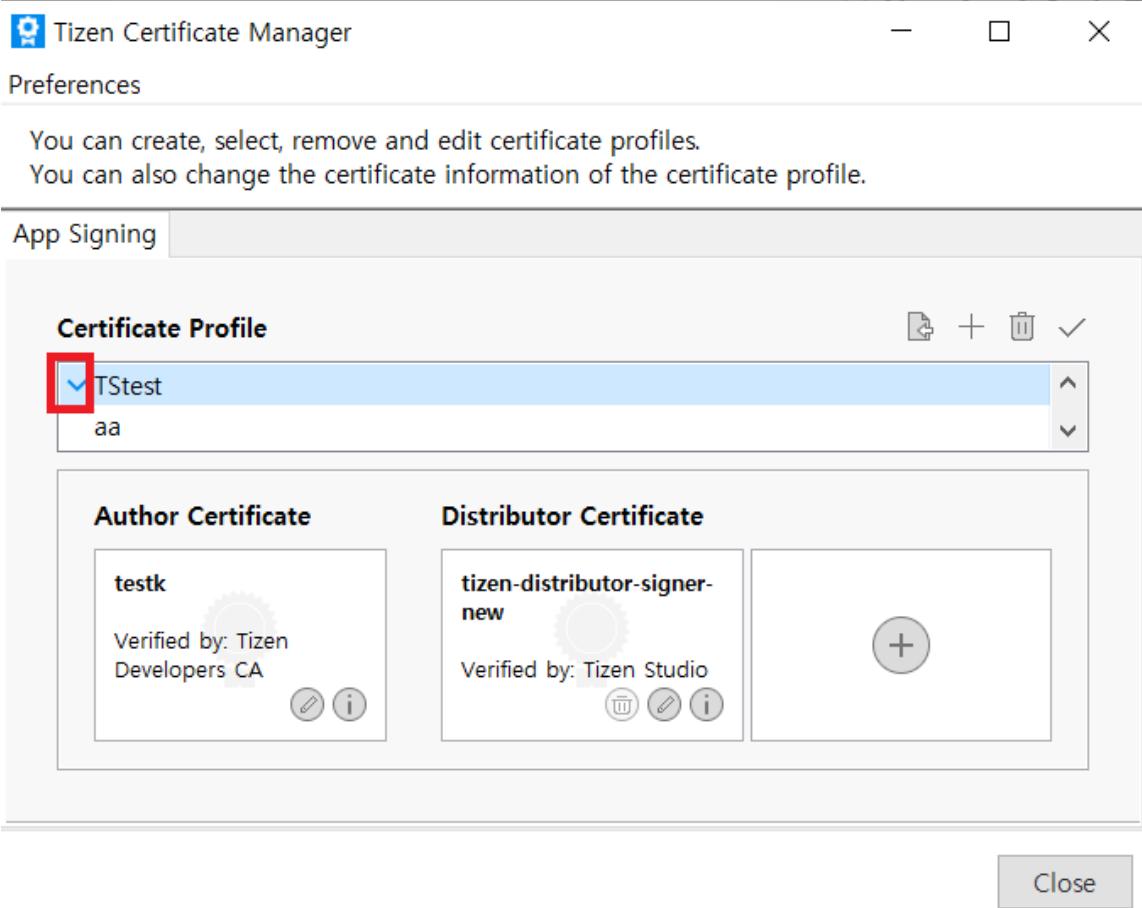
You can install your app only on Tizen 8.0 devices or the higher versions.

Select a distributor certificate for an another app store

If you own a distributor certificate for another app store, and want to submit your application to that store, add that distributor certificate to the certificate profile.

< Back Next > **Finish** Cancel

발급 진행 중 해당 화면에서 Privilege level을 Platform으로 설정해야 peripheral이 가능하다.



발급 후 더블클릭으로 인증서를 선택하여 체크 표시 활성화 필요

Arduino 설정

1. 아두이노 코드

```
#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>

#define SLAVE_ADDR 0x08
#define SS_PIN 10
#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;
```

```

        uint8_t a = 0x12;
        uint8_t b = 0x13;
        uint8_t c = 0x23;
        uint8_t d = 0x21;

        uint8_t nuidPICC[4];

void setup() {
    Wire.begin(SLAVE_ADDR);
    // Wire.onReceive(receiveData);
    Wire.onRequest(sendData);
    Serial.begin(9600);

    SPI.begin();
    rfid.PCD_Init();

    for(byte i = 0; i < 6; i++) {
        key.keyByte[i] = 0xFF;
    }

    nuidPICC[0] = 0x01;
    nuidPICC[1] = 0x01;
    nuidPICC[2] = 0x01;
    nuidPICC[3] = 0x01;
}

void loop() {
    // Reset the loop if no new card present on the sensor/read
    if ( ! rfid.PICC_IsNewCardPresent()){
        return;
    }

    // Verify if the NUID has been readed
    if ( ! rfid.PICC_ReadCardSerial())
}

```

```

    return;
}

Serial.print(F("PICC type: "));
MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid);
Serial.println(rfid.PICC_GetTypeName(piccType));

// Check is the PICC of Classic MIFARE type
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Class"));
    return;
}

if (rfid.uid.uidByte[0] != nuidPICC[0] ||
    rfid.uid.uidByte[1] != nuidPICC[1] ||
    rfid.uid.uidByte[2] != nuidPICC[2] ||
    rfid.uid.uidByte[3] != nuidPICC[3] ) {
    Serial.println(F("A new card has been detected."));

    // Store NUID into nuidPICC array
    for (byte i = 0; i < 4; i++) {
        nuidPICC[i] = rfid.uid.uidByte[i];
    }

    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    printHex(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
    Serial.print(F("In dec: "));
    printDec(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
}
// else Serial.println(F("Card read previously."));

// Halt PICC
rfid.PICC_HaltA();

```

```

// Stop encryption on PCD
rfid.PCD_StopCrypto1();

delay(5);
}

// char receivedByte;
// void receiveData(int byteCount) {
//   while (Wire.available()) {
//     receivedByte = Wire.read();
//     Serial.print("Received: ");
//     Serial.println(receivedByte, HEX);
//   }
// }

void sendData() {

    Wire.write(lowByte(nuidPICC[0]));
    Wire.write(lowByte(nuidPICC[1]));
    Wire.write(lowByte(nuidPICC[2]));
    Wire.write(lowByte(nuidPICC[3]));
    for (byte i = 0; i < 4; i++) {
        nuidPICC[i] = 0x01;
    }
}

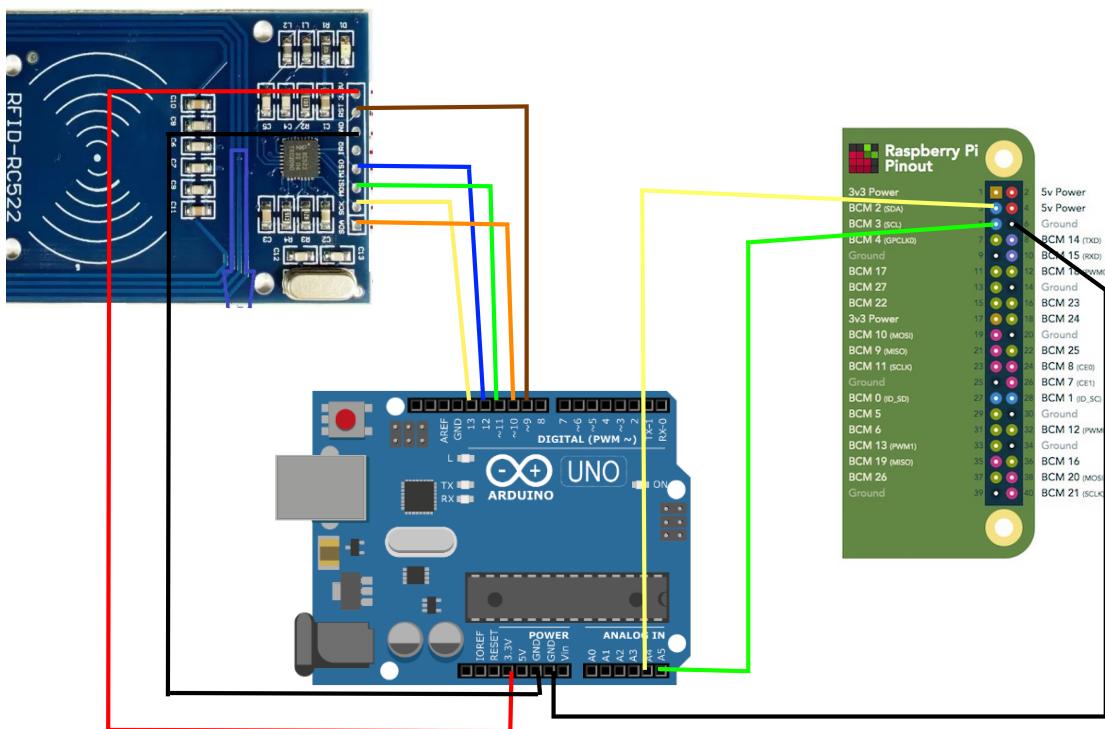
void printHex(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}

/**
 * Helper routine to dump a byte array as dec values to Se
 */

```

```
void printDec(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(' ');
        Serial.print(buffer[i], DEC);
    }
}
```

2. 파이 - 아두이노 연결 이미지



프론트엔드 포팅메뉴얼

node.js LTS version 설치(v20.11.0)

Node.js — Download

주요 라이브러리 및 플러그인

- 상태 관리: ZustandHTTP
 - 클라이언트: Axios
 - 날짜 관리: Moment
 - 라우팅: React Router DOM

- 스타일링: Styled ComponentsUI
- 컴포넌트: Swiper, React Calendar
- 그래픽스: Konva, React Konva
- 데이터 캐싱 및 관리: @tanstack/react-query
- PWA 지원: Vite Plugin PWAFirebase: Firebase SDK

개발 도구 및 환경 설정

- 코드 포맷팅 및 린팅: ESLint
 - 플러그인: eslint-plugin-react-hooks, eslint-plugin-react-refresh
- 정적 타입 검사: TypeScript
- 코드 컴파일러: @vitejs/plugin-react-swc
- 경로 별칭 관리: vite-tsconfig-paths

Vite 구성

- 플러그인
 - `vite-tsconfig-paths`
 - 프로젝트 내에서 TypeScript의 경로 별칭을 기반으로 모듈 해석을 도와준다.
 - `VitePWA`
 - Progressive Web App(PWA)을 위한 플러그인으로, 오프라인 지원, 웹 앱 설치 등의 기능을 제공한다.

빌드 및 배포

Docker Compose

```
version: "0.0.0"

services:
  supertizen_frontend:
    container_name: supertizen_frontend
    build:
      context: ./FE/vite-project
```

```

dockerfile: Dockerfile
image: supertizen_frontend_img
volumes:
  - /etc/letsencrypt/:/etc/nginx/ssl/
  - /home/ubuntu/clothes_images/:/usr/share/nginx/images/
  - /home/ubuntu/profile_images/:/usr/share/nginx/profile_
  - /home/ubuntu/outfit_images/:/usr/share/nginx/outfit_i
ports:
  - "80:80"
  - "443:443"
networks:
  - supertizen_net
environment:
  - TZ=Asia/Seoul

supertizen_backend:
  container_name: supertizen_backend
  build:
    context: ./BE/smartclothing
    dockerfile: Dockerfile
  image: supertizen_backend_img
  volumes:
    - /home/ubuntu/outfit_images/:/app/outfit_images/
  ports:
    - "8080:8080"
    - "65432:65432"
  networks:
    - supertizen_net
  environment:
    - TZ=Asia/Seoul

supertizen_machine_learning:
  container_name: supertizen_machine_learning
  build:
    context: ./ML/machinelearning
    dockerfile: Dockerfile
  image: supertizen_machine_learning_img
  ports:

```

```
- "8000:8000"
networks:
  - supertizen_net
environment:
  - TZ=Asia/Seoul

networks:
  supertizen_net:
```

Jenkinsfile

```
pipeline {
    agent any

    stages {
        stage('Shutdown Docker Compose') {
            steps {
                script {
                    // Docker Compose를 내리는 단계
                    sh 'docker-compose down'
                }
            }
        }
        stage('Start Docker Compose') {
            steps {
                script {
                    // Docker Compose를 시작하는 단계
                    sh 'docker-compose up --build -d'
                }
            }
        }
    }
}
```

Backend Dockerfile

```

# OpenJDK 17 이미지를 베이스로 사용
FROM openjdk:17-jdk-slim

# 애플리케이션을 빌드할 소스 코드 및 리소스 복사
COPY . /app

# 작업 디렉토리 설정
WORKDIR /app

# Gradle Wrapper에 실행 권한 부여
RUN chmod +x ./gradlew

# Spring Boot 애플리케이션 빌드
RUN ./gradlew clean bootJar

# JAR 파일을 /app 디렉토리로 복사
RUN cp build/libs/*.jar /app/app.jar

# Spring Boot 애플리케이션 실행을 위한 명령 설정
ENTRYPOINT ["java", "-jar", "/app/app.jar"]

```

Frontend Dockerfile

```

# 기본 이미지로 Node.js 버전 20.11.1 사용
FROM node:20.11.1 as build

# 작업 디렉토리 설정
WORKDIR /usr/src/app

# package.json 및 package-lock.json을 복사하여 종속성 설치
COPY package*.json ./

# 종속성 설치
RUN npm install

# 나머지 애플리케이션 코드 복사
COPY . .

```

```
# .env 파일 변경
#COPY .env-dev .env

# 프론트엔드 코드 빌드
RUN npm run build

# NGINX 이미지 생성
FROM nginx:latest

# NGINX에서 작업 디렉토리 설정
WORKDIR /usr/share/nginx/html

# 기본 NGINX 정적 콘텐츠 제거
RUN rm -rf /*

# Node.js 빌드 단계에서 빌드된 프론트엔드 코드 복사
COPY --from=build /usr/src/app/dist/ .

# 추가 NGINX 구성 파일 복사
COPY nginx.conf /etc/nginx/conf.d/default.conf

# NGINX를 시작
CMD ["nginx", "-g", "daemon off;"]
```

Machine learning Dockerfile

```
# 베이스 이미지 설정
FROM python:3.11.7

# 작업 디렉토리 설정
WORKDIR /app

# 필요한 파일 복사
COPY . /app

# 필요한 라이브러리 설치
```

```

RUN python -m pip install --upgrade pip

RUN pip install -r requirements.txt

# opencv 사전준비
RUN apt-get update && apt-get install -y libgl1-mesa-glx

# 데이터베이스 마이그레이션
RUN python manage.py makemigrations
RUN python manage.py migrate

# 서버 실행
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

```

nginx.conf

```

# HTTP 리다이렉션을 HTTPS로
server {
    listen 80;
    server_name j10s006.p.ssafy.io;

    return 301 https://$host$request_uri;
}

# HTTPS 서버 설정
server {
    listen 443 ssl;
    server_name j10s006.p.ssafy.io;

    ssl_certificate /etc/nginx/ssl/live/j10s006.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/live/j10s006.p.ssafy.io/privkey.pem;
    ssl_trusted_certificate /etc/nginx/ssl/live/j10s006.p.ssafy.io/cert.pem;

    # 프론트엔드
    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}

```

```

        try_files $uri $uri/ /index.html;
    }

# 백엔드 프록시
location /api {
    proxy_pass http://j10s006.p.ssafy.io:8080;    # 백엔드 서버 주소
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

# ML서버 프록시
location /ML-api {
    proxy_pass http://j10s006.p.ssafy.io:8000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

# 옷 이미지 프록시
location /images {
    alias /usr/share/nginx/images;
    autoindex on;
    try_files $uri $uri/ =404;
}

# 프로필 이미지 프록시
location /profile {
    alias /usr/share/nginx/profile_images;
    autoindex on;
    try_files $uri $uri/ =404;
}

# 스케줄 코디 이미지 프록시
location /outfit {
    alias /usr/share/nginx/outfit_images;
}

```

```
    autoindex on;
    try_files $uri $uri/ =404;
}
}
```