
자율 프로젝트

기업연계프로젝트 - 삼성SDI

# 포팅 매뉴얼

A507

권경민 김유민 문요성 이지수 정진아

# 목차

# 1. FrontEnd

## 1. 버전

package.json 에 명시

## 2. 패키지 설치

```
npm i
```

## 3. 로컬호스트 시작

```
npm start
```

# 2. BackEnd

## 1. 버전

- aws ubuntu 20.04
- mysql 8.0.31
- django 3.2.16

## 2. 설치

### a. 젠킨스/도커

## 방화벽 설정

- 사전 준비: AWS EC2
- ubuntu 계정 접속
    - Session - SSH
    - Remote host: host 주소 입력
    - Specify username: ubuntu 입력
    - Advanced SSH settings
    - Use private key: pem 파일 등록
- 방화벽 설정

```
sudo ufw default deny incoming
sudo ufw default allow outgoing

sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https

sudo ufw allow 8080
sudo ufw allow 9090  # jenkins

...
```

- 방화벽 설정 확인

  sudo ufw status

  Status: inactive
    - 기본적으로 비활성화 된 상태
- sudo ufw enable

  Command may disrupt existing ssh connections. Proceed with operation (y|n)? y

  Firewall is active and enabled on system startup
    - 방화벽 활성화
- sudo ufw status

  Status: active

  | To | Action | From |
  | --- | --- | --- |
  | 22/tcp | ALLOW | Anywhere |
  | 80/tcp | ALLOW | Anywhere |
  | 443/tcp | ALLOW | Anywhere |
  | 8080 | ALLOW | Anywhere |
  | 9090 | ALLOW | Anywhere |
  | 8000 | ALLOW | Anywhere |
  | 3000 | ALLOW | Anywhere |
  | 3306 | ALLOW | Anywhere |
  | 3306/tcp | ALLOW | Anywhere |
  | 22/tcp (v6) | ALLOW | Anywhere (v6) |
  | 80/tcp (v6) | ALLOW | Anywhere (v6) |
  | 443/tcp (v6) | ALLOW | Anywhere (v6) |
  | 8080 (v6) | ALLOW | Anywhere (v6) |
  | 9090 (v6) | ALLOW | Anywhere (v6) |
  | 8000 (v6) | ALLOW | Anywhere (v6) |
  | 3000 (v6) | ALLOW | Anywhere (v6) |
  | 3306 (v6) | ALLOW | Anywhere (v6) |
  | 3306/tcp (v6) | ALLOW | Anywhere (v6) |

# Docker 설치

- 사전 패키지 설치

```
sudo apt update
sudo apt-get install -y ca-certificates ₩
  curl ₩
  software-properties-common ₩
  apt-transport-https ₩
  gnupg ₩
  lsb-release
```

- gpg 키 다운로드

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /etc/apt/keyrings/docker.gpg
```

- Docker 설치

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose
```

# 젠킨스 설치 및 계정 생성

- docker-compose 이용하여 젠킨스 컨테이너 생성

```
vim docker-compose.yml
version: '3'

services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "9090:8080"
```

```
    privileged: true
    user: root
```

    - volumes
        - aws `/var/run/docker.sock`와 컨테이너의 `/var/run/docker.sock` 연결
        - aws `/jenkins`와 컨테이너의 `/var/jenkins_home` 연결
    - ports
        - aws 9090 포트와 컨테이너의 8080 포트 연결
    - user
        - 젠킨스에 접속할 유저 계정
    - 작성 후 `esc` + `:wq` + `enter`
- 컨테이너 생성

  `sudo docker-compose up -d`
- 컨테이너 확인

  `sudo docker ps`
    - django와 react는 젠킨스 설정 후 추가됨
- 젠킨스 계정 생성 및 플러그인 설치
    - `{ip 주소}:9090` 으로 접속
    - Administrator password 확인 및 입력

      `sudo docker logs jenkins`
    - `Install suggested plugins` 클릭
    - 플러그인 설치 모습
    - 젠킨스 계정 생성 `Save and Continue` 클릭
    - `Save and Finish` 및 `Start using Jenkins` 클릭
    - `Jenkins 관리` 탭 클릭
    - `플러그인 관리` 클릭
    - `gitlab` 검색 후 표시된 플러그인 체크
    - `Install witout restart`
    - `docker` 검색 후 표시된 플러그인 체크
    - `Install witout restart`
    - `SSH` 검색 후 표시된 플러그인 체크
    - `Install witout restart`
- 모든 설치 완료

# 젠킨스 프로젝트 생성, 웹훅 설정, 자동 빌드 테스트

- 사전 준비: 깃랩 레포지토리 생성
    - 구성: `BE`(장고 프로젝트), `FE/bom`(리액트 프로젝트)
- 젠킨스 프로젝트 생성
    - `+ 새로운 Item` 클릭
    - 프로젝트 이름 작성 (`thundervolt`), `Freestyle project` 클릭 및 `OK`
    - `구성 - 소스 코드 관리`에서 `None`을 `Git`으로 변경
    - `Repository URL`에 깃랩 레포지토리 URL 입력
    - 에러메시지 출력 정상
    - `Credentials - Add - Jenkins` 클릭
    - Kind: Username with password
    - Username: 깃랩 아이디
    - Password: 깃랩 비밀번호
    - ID: Credential 구별할 텍스트
    - 입력 후 `Add` 클릭
    - 생성된 Credentials 를 클릭했을 때 오류가 사라지면 성공
    - `Branches to build` 빌드할 브랜치 설정
    - dev 브랜치를 빌드하기 위해 `*/dev` 작성
    - `빌드 유발` 탭에서 `Build when a change is pushed to GitLab` 체크
    - `고급` 클릭
    - `Secret token`에서 `Generate` 클릭하여 토큰 생성
    - 이 토큰을 통해 깃랩 웹훅 연결하기 때문에 저장해두기
    - `Build Steps` 탭에서 `Add build step` 클릭
    - `Execute shell` 선택
    - 명령어 입력 칸이 생성됨
    - 연결 테스트를 위해 간단한 명령어만 작성
    - `저장` 클릭
    - 프로젝트 화면에서 `지금 빌드`를 클릭하여 수동 빌드 진행
    - 완료 표시가 뜨면 성공

    - Console Output을 통해 콘솔 출력 확인 가능
    - 빌드에 성공한 콘솔 창
- 깃랩 웹훅 설정 및 자동 빌드 테스트
    - 깃랩 레포지토리에서 설정 - 웹훅 클릭
    - URL: 연결할 젠킨스 프로젝트 주소
    - Secret token: 젠킨스 프로젝트 생성할 때 저장한 토큰 값
    - Trigger: Push events와 Merge request events 체크
    - 대상 브랜치는 dev
    - Add webhook 클릭
    - 테스트 클릭, Push events 선택
    - 코드 200 확인
    - 젠킨스에서 빌드 성공 확인
    - 젠킨스와 깃랩 연결 끝

# 젠킨스 도커 이미지 빌드

젠킨스에서 도커 빌드를 하기 위해 젠킨스 컨테이너 안에 도커 설치

EC2에 도커 설치할 때와 동일하게 진행

- Jenkins bash shell에 접근
  sudo docker exec -it jenkins bash
- 사전 패키지 설치
  apt update
  apt-get install -y ca-certificates ₩
    curl ₩
  software-properties-common ₩
  apt-transport-https ₩
  gnupg ₩
  lsb-release
    - 루트 계정이기 때문에 명령어에 sudo 넣지 않기

- gpg 키 다운로드
  `mkdir -p /etc/apt/keyrings`
  `curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg`
- Docker 설치
  `apt update`
  `apt install docker-ce docker-ce-cli containerd.io docker-compose`
- 프로젝트에 Dockerfile 작성
  - 파일명 `Dockerfile`
  - 확장자 없음
  - 프로젝트가 위치한 곳에 작성
- `# Django`

  ```
  FROM python:3.9.15
  ENV PYTHONUNBUFFERED 1

  RUN apt-get update && apt-get install default-libmysqlclient-dev

  WORKDIR /var/jenkins_home/workspace/thundervolt/BE

  COPY . .
  RUN pip install --upgrade pip
  RUN pip install --upgrade setuptools

  RUN pip install -r requirements.txt

  CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
  ```

  ```
  # React

  FROM node:16.16.0 as build-stage
  ENV WDS_SOCKET_PORT=443
  WORKDIR /var/jenkins_home/workspace/thundervolt/FE/bom

  COPY package*.json ./
  ```

```
RUN npm install

COPY . .
RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]
```

- 젠킨스에서 Dockerfile을 이용하여 도커 이미지 생성
    - 젠킨스 프로젝트에서 구성 클릭
    - Build Steps로 이동하여 기존의 명령어 삭제
- docker image prune -a --force
  mkdir -p /var/jenkins_home/images_tar

  cd /var/jenkins_home/workspace/thundervolt/BE/
  docker build -t django .
  docker save django > /var/jenkins_home/images_tar/django.tar

  cd /var/jenkins_home/workspace/thundervolt/FE/bom/
  docker build -t react .
  docker save react > /var/jenkins_home/images_tar/react.tar

  # ls /var/jenkins_home/images_tar (확인용)
    - 위 명령어로 변경 후 저장
    - 지금 빌드를 통해 수동 빌드
    - 빌드 성공 확인

# 젠킨스에서 SSH 명령어를 통해 저장한 도커 이미지로 컨테이너 생성

- 사전 준비: EC2 root, ubuntu 비밀번호 설정

  ```
  sudo passwd root
  New password:  # 비밀번호 입력
  Retype new password:  # 비밀번호 재입력
  passwd: password updated successfully  # 설정 완료

  sudo passwd ubuntu
  New password:  # 비밀번호 입력
  Retype new password:  # 비밀번호 재입력
  passwd: password updated successfully  # 설정 완료
  ```

- 젠킨스 SSH 설정
  - 젠킨스 홈페이지에서 Jenkins 관리 클릭
  - 시스템 설정 클릭
  - Publish over SSH에서 SSH Servers의 추가 버튼클릭
  - Name: 이름
  - Hostname: EC2 IP
  - Username: EC2 접속 계정 이름 (ubuntu)
  - 고급버튼 클릭
  - Use password authentication, or use a different key 체크
  - Key에 pem 파일 값 입력
    - pem 파일을 VSCode 로 열기
    - 전체 복사하여 붙여넣기
  - Test Configuration 클릭, Success가 나오면 성공
  - SSH 연결 완료, 저장 버튼 클릭
- 젠킨스 빌드 후 조치로 SSH 명령어 전송 (EC2에 도커 컨테이너 생성)
  - 젠킨스 프로젝트 페이지에서 구성 클릭
  - 빌드 후 조치 추가클릭, Send build artifacts over SSH 선택
  - Source files: 컨테이너에서 AWS로 파일을 전송하는 부분 (중요X)

- sudo docker load < /jenkins/images_tar/django.tar
  sudo docker load < /jenkins/images_tar/react.tar

  if (sudo docker ps | grep "django"); then sudo docker stop django; fi
  if (sudo docker ps | grep "react"); then sudo docker stop react; fi

  sudo docker run -it -d --rm -p 8000:8000 --name django django
  # echo "Run BE" (확인용)
  sudo docker run -it -d --rm -p 3000:3000 --name react react
  # echo "Run FE" (확인용)
    - Exec command에 명령어 입력
    - sudo docker load < /jenkins/images_tar/django.tar
      - django.tar를 압축 해제하여 docker 이미지로 등록
    - if (sudo docker ps | grep "django"); then sudo docker stop django; fi
      - django 컨테이너가 동작중이면 멈춤
    - sudo docker run -it -d --rm -p 8000:8000 --name django django
      - django 이름으로 컨테이너 생성, 8000번 포트로 연결
    - 저장 클릭
    - 지금 빌드를 통해 빌드, 콘솔 창에서 결과 확인

## b. mysql

- mysql
  - 설치

    ```
    sudo apt-get install mysql-server
    ```

  - 초기 비밀번호 설정

    ```
    mysqladmin -u root -p password  [비밀번호]
    ```

  - 스키마(DB)생성

    ```
    mysql
    create database [DB이름];
    ```

  - 유저 생성 및 권한 부여

    ```
    create user '[user_name]'@'%' identified by '[password]';
    grant all privileges on [DB].* to '[user_name]'@'%';
    ```

### c. my_settins.py

- .gitignore 등록 파일

```
SECRET_KEY = '{SECRET_KEY}'

# LOCAL
# DATABASES = {
#   'default': {
#     'ENGINE': 'django.db.backends.mysql',
#     'NAME': '{DB_NAME}',
#     'USER': '{USER_NAME}',
#     'PASSWORD': '{USER_PASSWORD}',
#     'HOST': '{HOST}',
#     'PORT': '3306',
#   }
# }

# AWS
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': '{DB_NAME}',
    'USER': '{USER_NAME}',
    'PASSWORD': '{USER_PASSWORD}',
    'HOST': '{HOST}',
    'PORT': '3306',
  }
}

AWS_S3_ACCESS_KEY_ID = '{ACCESS_KEY_ID}'
AWS_S3_SECRET_ACCESS_KEY = '{SECRET_ACCESS_KEY}'
AWS_STORAGE_BUCKET_NAME = '{BUCKET_NAME}'
```

# 3. Embedded

## 1. 버전

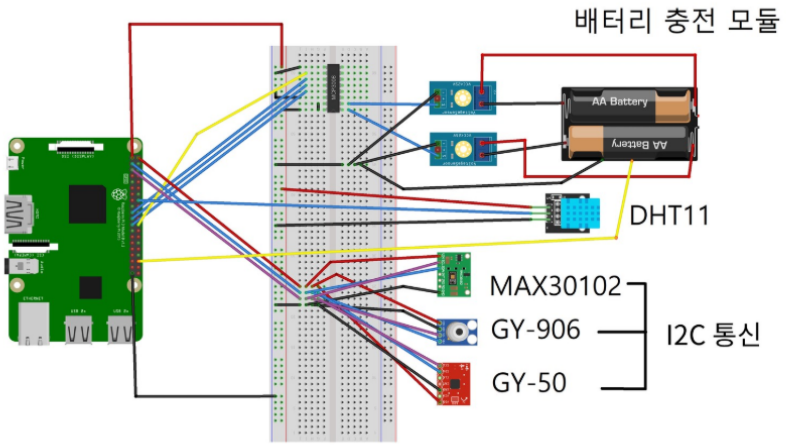
- python 3.9.2
- PySide2
- numpy 1.23.4
- RaspberryPi4 B

## 2. 설치

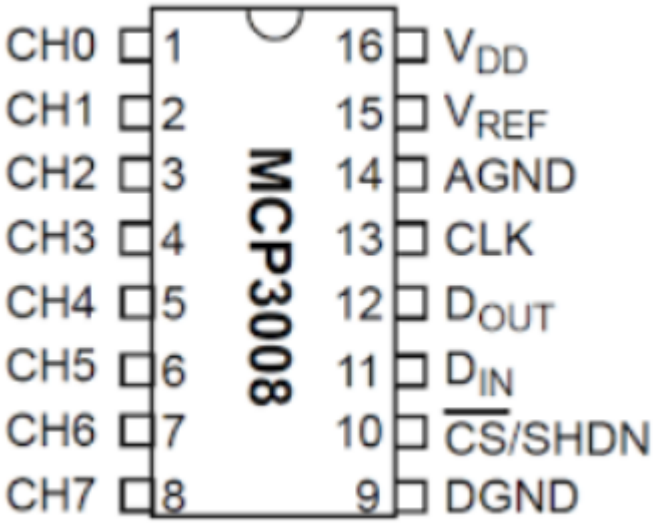
### a. 부품목록

- adc - mcp3008
- 심전도 센서 - max30102
- 비접촉식 온도 센서 - gy-906
- 온습도 센서 - dht11
- 자이로 센서 - gy-50

### b. 회로도



- mcp3008




| RaspberryPi 4 | mcp3008 |
|---|---|
| 3.3V | Vdd |
| 3.3V | Vref |
| GND | AGND |
| SCLK (GPIO 11) | CLK |
| MISO (GPIO 9)) | Dout |
| MOSI (GPIO 10) | Din |
| CE 0 (GPIO 8) | CS/SHDN |
| GND | DGND |

- CH0, CH1 에 배터리 전압센서 연결

| mcp3008 | 전압센서 |
|---|---|
| CH0, CH1 | S |
| GND | GND ( - ) |

- I2C 3개 사용
    - 심전도, 자이로, 비접촉온도
    - VIN - 5V로 연결, SDA - GPIO 2 , SCL - GPIO 3 연결
- 온도센서(BMS)용 DATA - GPIO 23번

## c. 라즈베리파이 초기설정

- vim 설치

```
sudo apt-get install vim
```

- 시간 설정 - 한국 (DB update를 위하여)

```
sudo apt-get install rdate
sudo rdate -s time.bora.net
sudo vi /etc/rc.local
(파일에 추가 ) sudo rdate -s time.bora.net
```

- 한글 글씨 설정

```
sudo apt-get install ibus
sudo apt-get install ibus-hangul
sudo apt-get install fonts-unfonts-core
```

- python3.9.2 확인

```
python3 --version
```

- 마우스 이용하여 screen configure에서 1280*720 설정 및 작업 표시줄 설정

## d. 모듈 및 필요 패키지 설치

- MAX30102 (심전도)
  pip install numpy
- DHT11
  - python3-dev (apt-get)
  - setuptools wheel (pip)

```
sudo apt-get install python3-dev
pip install setuptools wheel
```

  - Adafruit_DHT 라이브러리

```
git clone
https://github.com/adafruit/Adafruit_Python_DHT.gi
```

  - detect_platform 에서 BCM=”2711” 에 return 3 추가

```
    if not match:
        # Couldn't find the hardware,
        return None
    if match.group(1) == 'BCM2708':
        # Pi 1
        return 1
    elif match.group(1) == 'BCM2709':
        # Pi 2
        return 2
    elif match.group(1) == 'BCM2835':
        # Pi 3 or Pi 4
        return 3
    elif match.group(1) == 'BCM2837':
        # Pi 3b+
        return 3
    else:
        # Something else, not a pi.
```

  - 111번 째 줄 이후 다음 코드 추가

```
elif match.group(1) == 'BCM2711':
            return 3
```

- sudo python3 setup.py install (라이브러리 설치)

```
sudo python3 setup.py
```

※ 반드시 detect_platform.py 변경 후 설치해야 함

Adafruit_DHT 가 더 이상 업데이트가 되지 않으면서 라즈베리파이4B와 호환불가인 문제 발생. 이를 해결하기 위하여 직접 보드 지정

- GUI
  - mysql-connector-python

```
pip install mysql-connector-python
```

  - pyside2

```
sudo apt-get install pyside2-tools
sudo apt-get install pyside2.*
sudo apt-get install python3-pyside2.*
```

  - pygraph

```
pip install pygraph
```

- 이외

```
sudo apt-get install libatlas-base-dev
```

  - conf.py의 db정보 변경 필수