포팅 매뉴얼

1. 테스트 환경

- AWS EC2
- Ubuntu 20.04 LTS
- Raspberry Pi 4
- USB Webcam
- USB Mic
- LCD Display (1024x600)

2. Requirements

- Docker >= 20.10.0
- Mosquitto >= 2.0.15
  - with TLS encryption, port 8883
- RPi4
  - Python == 3.7.3
  - OpenCV == 4.6.0
  - cvlib == 0.2.7
  - dlib == 19.24.0
  - tensorflow == 2.4.0
  - google-cloud-speech == 2.16.2
  - pvporcupine == 2.1.4
  - pvrecorder == 1.1.1
  - NodeJS

3. 빌드 및 실행

3-1. 백엔드 서버

루트 .env 파일

빌드 및 운영에 관여하는 모든 시크릿 정보가 담긴 파일

- .env.template를 .env로 복사한 후 내용을 작성
  - `SERVICE_HOST`: 서비스의 도메인 주소 (ie. j7a704.p.ssafy.io)
  - `MYSQL_###`: MariaDB 관련 변수
    - `MYSQL_HOST`: 빌트인 MariaDB를 사용하는 경우 기본값 db를 유지해야 함 (docker-compose.yml에 정의됨)
  - `OAUTH2_###`: 소셜로그인을 위한 ID, SECRET
    - Google
    - Kakao
  - `LOGIN_###`: 로그인 후 리다이렉트 될 주소
    - `LOGIN_SCHEME`: http 또는 https
    - `LOGIN_HOST`: {SERVICE_HOST}와 동일하게 설정
  - `BROKER_URL`, `BASE_TOPIC`: MQTT 관련 설정
    - `BROKER_URL`: MQTT 서버의 URL (TLS를 사용할 경우 ssl://로 시작)
    - `BASE_TOPIC`: 기본적으로 사용할 MQTT Topic

SSL 설정

- docker-compose.yml 수정
  - `nginx:` 블록의 `volumes:` {cert_path}:/etc/letsencrypt 형태로 수정
- nginx/nginx.conf.template 수정
  - `ssl_certificate` 및 `ssl_certificate_key`를 알맞게 설정 (기본값: `/etc/letsencrypt/live/${SERVICE_HOST}/...`)

빌드 및 서버 실행

```
$ cd {PROJECT_ROOT}
$ docker compose up -d --build
```

3-2. 하드웨어

.env 파일

- `hw/.env.template`을 `hw/.env`로 복사 후 내용을 작성
  - WEBSOCKET_PORT : 사용할 웹소켓 port 번호
  - MQTT_HOST : MQTT 서버의 웹 주소
  - MQTT_PORT : MQTT port 번호
  - MQTT_BASE_TOPIC : MQTT의 BASE 주제
  - DEVICE_ID : 라즈베리파이의 시리얼 번호 (`cat /proc/cpuinfo | grep Serial | awk '{print $3}'`)
  - PICOVOICE_KEY : Picovoice 사이트에서 로그인 후 얻은 ACCESS KEY
  - KEYWORDS_PATH : 사용할 wakeup word ppn 파일 경로
  - MODEL_PATH : 사용할 wakeup word의 언어 pv 파일 경로

- AUDIO_INDEX : USB 마이크가 장착된 audio index
  - `./Voice-recognition/show_audio.py` 실행시키기
  - 사용중인 마이크 명에 일치하는 audio index 확인
- RECORD_PATH : 녹음 후 저장되는 경로, 어느 경로든 상관없음

음성인식 설정

USB 마이크 1개를 이용하여 wakeup word와 google stt를 이용하여 명령어를 입력

1. Picovoice 설정 "하이 빅스비"와 같은 wakeup word 설정
   - Picovoice 사이트에서 로그인 후 `Access Token` 발급
   - 라이브러리 설치

     ```
     pip3 install pvporcupinedemo
     pip3 install pvrecorder
     ```

   - 테스트

     `hw/Voice-recogntion/Test_Code/porcupine_demo_mic`를 통해 만들어둔 본인의 access_key를 넣고 실행

     ```
     porcupine_demo_mic --access_key ${ACCESS_KEY} --keywords picovoice
     ```

     picovoice라고 마이크에 말 했을 때, Hotword detected라는 메세지 출력하면 정상
2. Google STT 설정
   - `프로젝트 생성 > 서비스 계정 & Access 키(.json 파일) 발급` 참고 사이트
   - 생성한 Access 키(.json 파일)를 환경변수로 설정

     ```
     GOOGLE_APPLICATION_CREDENTIALS="KEY_PATH"
     ```

   - 라이브러리 설치

     ```
     pip3 install --upgrade google-cloud-speech
     pip3 uninstall grpcio
     pip3 uninstall grpcio-status
     pip3 install grpcio==1.44.0 --no-binary=grpcio
     pip3 install grpcio-tools==1.44.0 --no-binary=grpcio-tools
     ```

   - 테스트

`hw/Voice-recogntion/Test_Code/google_stt_test.py`를 실행

```
python3 google_stt_test.py
```

`Transcript: how old is the Brooklyn Bridge` 출력 시 정상

자세인식 설정

OpenCV와 Tensorflow를 이용하여 USB 카메라에 찍힌 사진을 통해 자세 인식

1. OpenCV 설정
    - 라이브러리 설치

    ```
    pip3 install opencv-contrib-python==4.1.0.25
    ```

    - 테스트

    ```
    python3 -c "import cv2; print(cv2.__version__)"
    ```

    에러 없을 시 정상

2. dlib 설정
    - 라이브러리 설치

    ```
    pip3 install dlib
    ```

    - 테스트

    ```
    python3 -c "import dlib; print(dlib.__version__)"
    ```

    에러 없을 시 정상

3. cvlib 설정
    - 라이브러리 설치

    ```
    pip3 install cvlib
    pip3 install <https://github.com/lhelontra/tensorflow-on-
    arm/releases/download/v2.1.0/tensorflow-2.1.0-cp37-none-
    linux_armv7l.whl>
    ```

- 테스트

```
python3 -c "import cvlib; print(cvlib.__version__)"
```

에러 없을 시 정상

스마트싱스 설정

- Install packages
- Build device library for smartThings

```
cd ~/rpi-st-device
./sdkbuildsetup
cd ~/st-device-sdk-c
make
```

- Register Device in Developer Workspace

1. Make device workspace
2. Make device Profile and Onboarding
3. Write Product info
4. Create raspberry pi device key
5. Enroll your device key into your SmartThings in workspace
6. download Onboarding.json and copy into raspberryPi directory
7. make qrcode for device

- Build your device

```
cd ~/st-device-sdk-c/example
rm example
make
```

- Setting Raspberry Pi for AP mode
  - Check your Pi have AP mode

```
iw phy0 info
```

  - Check if you have with Wifi(2.4GHz) in your Area
    - If there no Wifi(2.4GHz) you can't connect your device in your phone

- Install packages for SoftAP services

```
sudo apt-get install hostapd
sudo apt-get install dnsmasq
```

- Disable SoftAP service to prevent start without your purpose

```
sudo systemctl unmask hostapd
sudo update-rc.d hostapd disable
sudo update-rc.d dnsmasq disable
```

- Copy and change dhcpcd_ap code

```
sudo cp /etc/dhcpcd.conf /etc/dhcpcd_ap.conf
```

```
# /etc/dhcpcd_ap.conf

interface wlan0
static ip_address = 192.168.2.1/24
nohook wpa_supplicant
```

- Change hostapd.conf check your channel with `iw wlan0 info`

```
# /etc/hostapd/hostapd.conf

contry_code=KR
interface=wlan0
channel={channel}
```

- Change code dnsmasq.conf

```
# /etc/dnsmasq.conf

interface=wlan0
dhcp-range=192.168.2.2,192.168.2.10,255.255.255.0.12h
domain=wlan
address=/gw.wlan/192.168.2.1
```

- Enable AP mode

```
cd ~/rpi-st-device
sudo ./testhostapd
```

- Connect your device in your phone application

    1. Use qrcode
    2. Find devices that can be registered nearby

디스플레이 프론트엔드 설정

- 프론트엔드 코드 빌드 및 실행

```
cd front
npm install
npm run build
npx live-server build --port=3000 --no-browser --entry-file=build/index.html
```