


# 9주차 (4/29~5/5)



2024-04-30 (화) 오프라인 미팅 15:00 ~ 21:00

2024-05-01 (수) 오프라인 미팅 15:00 ~ 21:00

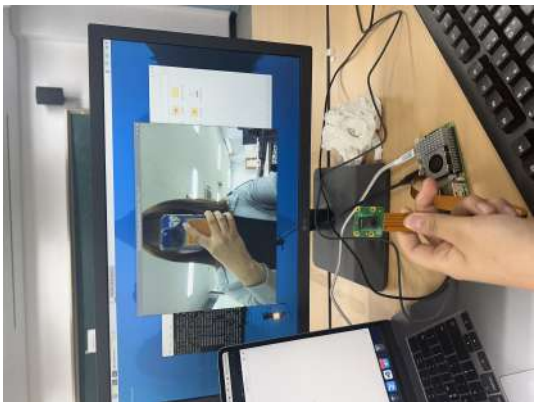
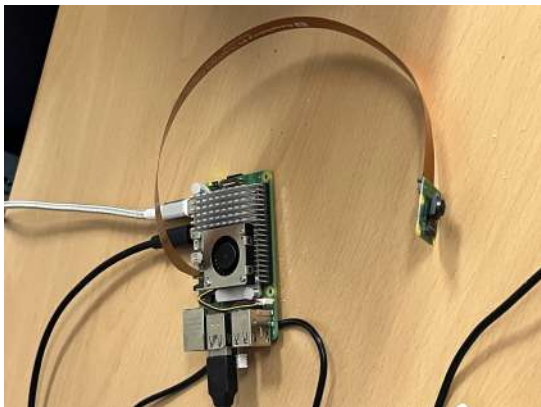
2024-05-02 (목) 오프라인 미팅 15:00 ~ 21:00

2024-05-03 (금) 오프라인 미팅 15:00 ~ 21:00

총 활동 시간 : 24시간

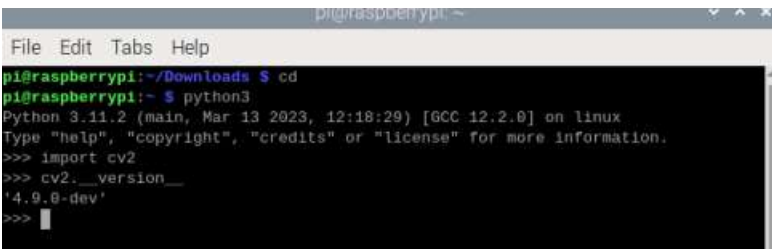
## (1) 라즈베리파이

- 라즈베리파이카메라 연결

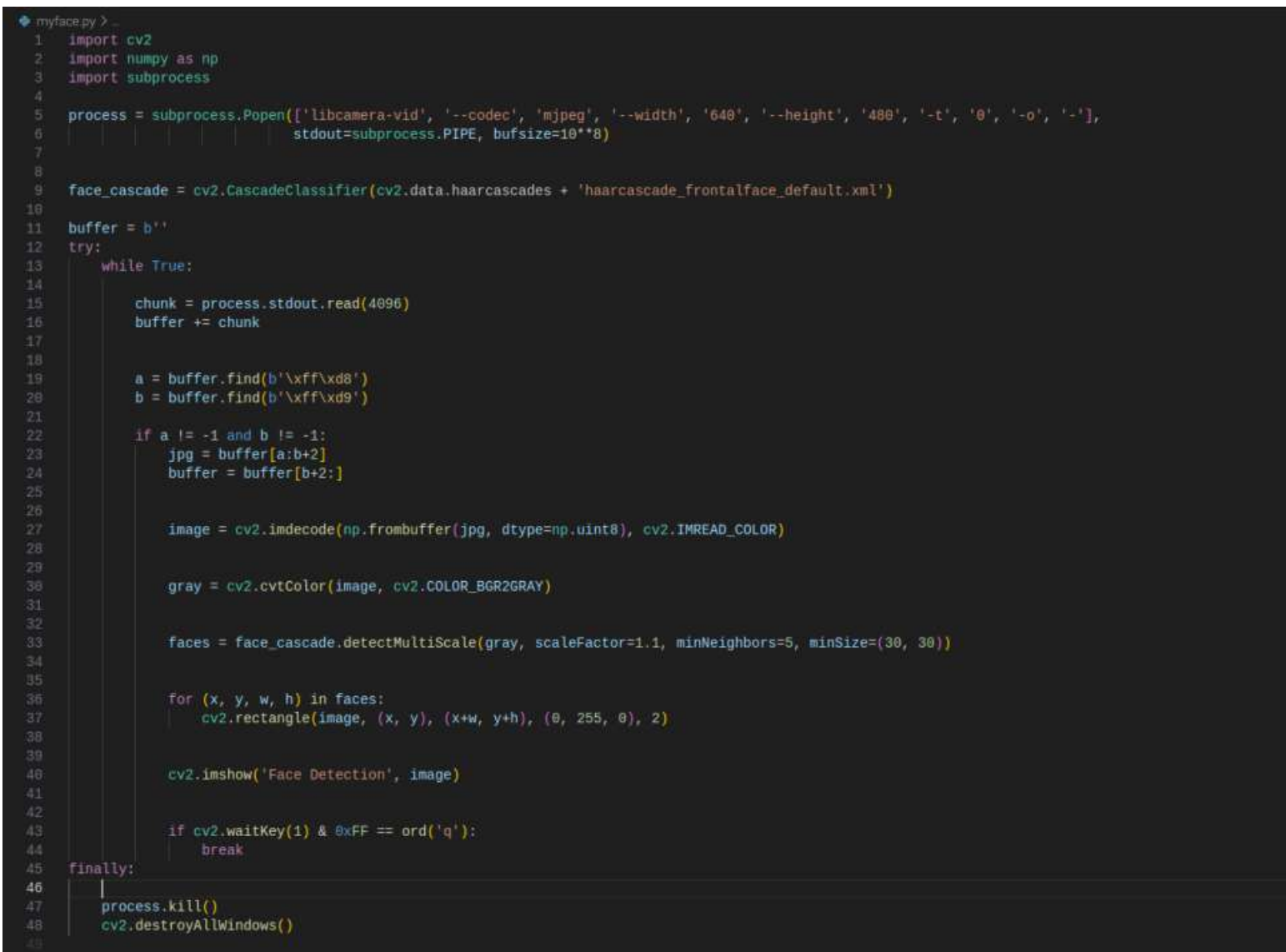


- opencv 설치

<https://qengineering.eu/install-opencv-on-raspberry-pi-5.html>



- 실시간 얼굴인식 알고리즘 : Haar Cascade



```

// 필요한 라이브러리 가져오기
import cv2
import numpy as numpy
import subprocess

// libcamera-vid 프로그램 실행
// MJPEG 코덱으로 너비 640, 높이 480인 비디오를 캡처
// -t 0 : 지속적으로 비디오 캡처
// -o : 결과를 표준 출력에 보내도록 지정
process = subprocess.Popen(['libcamera-vid', '--codec', 'mjpeg', '--width', '640', '--height', '480', '-t', '0', '-o', '-',
                             stdout=subprocess.PIPE, bufsize=10**8])

// 얼굴 감지를 위해 Haar Cascade 분류기를 사용
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

// 버퍼에 비디오 스트림을 임시로 저장
// 빈 문자열을 만들어서 버퍼를 초기화
buffer = b''

// 무한루프 시작 - 비디오 프레임을 계속해서 읽고 처리한다
try:
    while True:

        // 외부 프로세스로부터 비디오 프레임을 가져와 데이터를 버퍼에 추가
        chunk = process.stdout.read(4096)
        buffer += chunk

        // 버퍼에서 JPEG 이미지의 시작과 끝을 찾는다
        a = buffer.find(b'\xff\xd8') // 시작
        b = buffer.find(b'\xff\xd9') // 끝

        // 시작과 끝을 찾았는지 확인
        if a != -1 and b != -1:
            // 이미지를 추출하고 추출한 이미지 이후의 데이터를 버퍼에서 삭제
            jpg = buffer[a:b+2]
            buffer = buffer[b+2:]

            // JPEG 이미지 데이터를 numpy 배열로 변화하고 이미지 디코딩
            image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8), cv2.IMREAD_COLOR)

            // 이미지를 그레이스케일로 변환
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

            // detectMultiScale() 함수를 사용하여 이미지에서 얼굴 검출
            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

            // 검출된 얼굴 주변에 사각형을 그린다
            for (x, y, w, h) in faces:
                cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
                // 얼굴이 감지된 이미지 표시
                cv2.imshow('Face Detection', image)
                // 사용자가 q를 누를 때까지 대기하고 q를 누르면 무한루프 종료
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

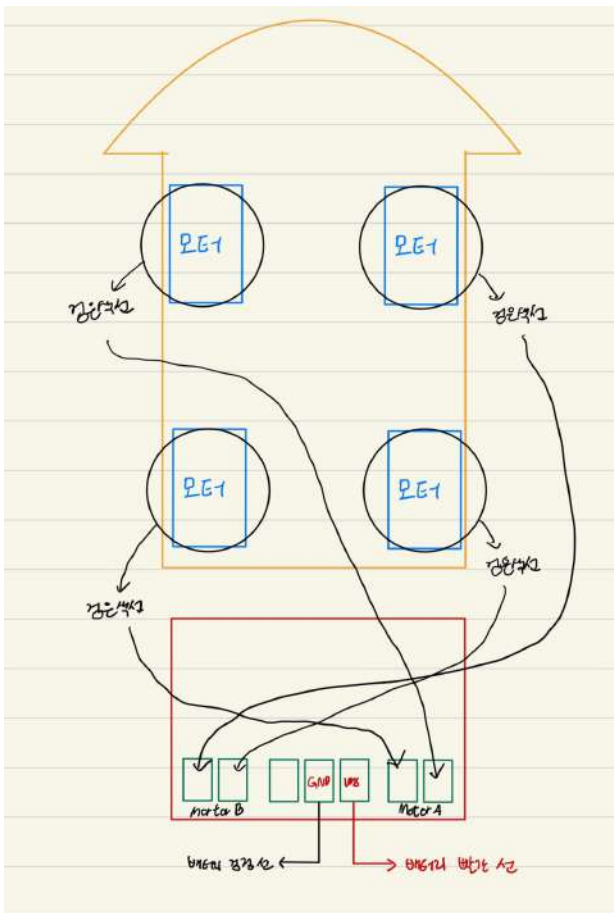
finally:
    // 루프가 종료되면 프로세스를 종료하고 윈도우를 닫는다
    process.kill()
    cv2.destroyAllWindows()

```



## (2) 로봇 하드웨어 제작

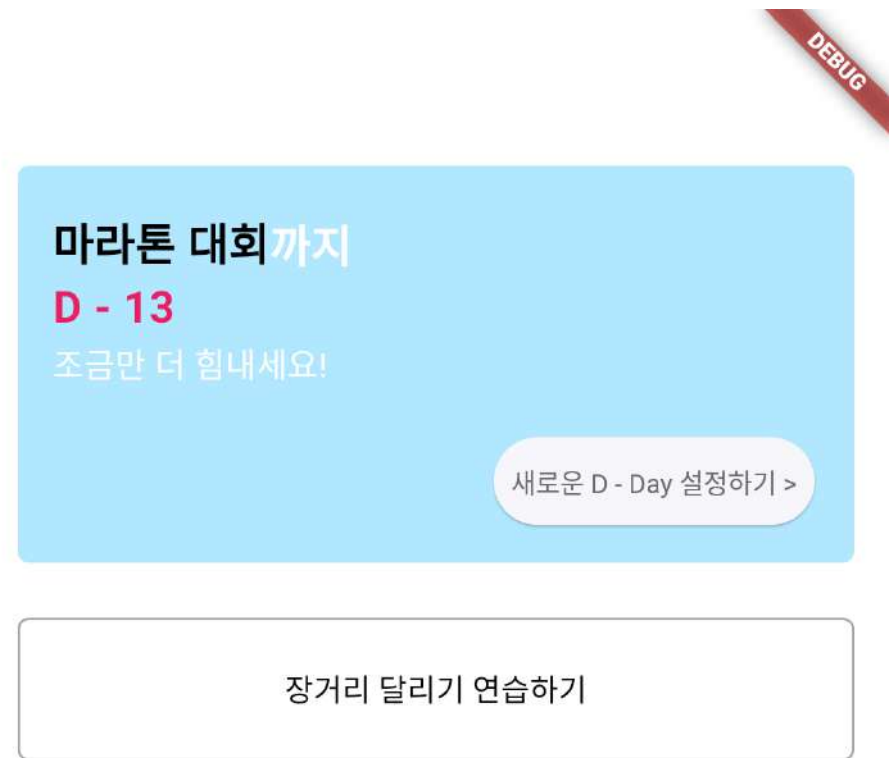
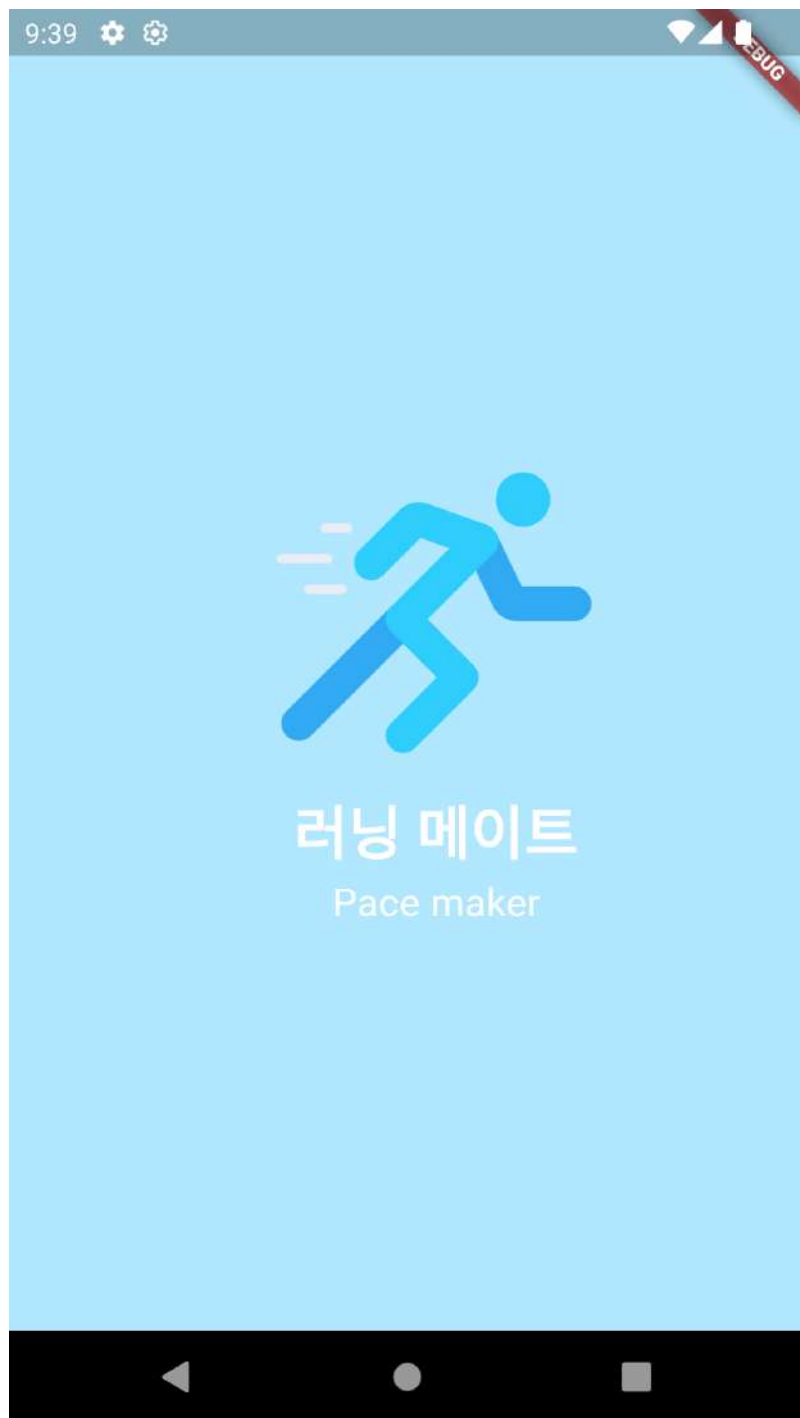
제작 도중 모터드라이버 고장 문제 발생 ⇒ 부품 교체 예정



로봇 구현도 그림으로 구상함.

1. 모터와 로봇의 첫 번째 본체 부분에 나사를 이용해 연결
2. 로봇의 본체를 뒤집어 모터 드라이버를 연결
3. 모터의 선을 째지어 꼬아준 후 모터 드라이버에 연결
4. 배터리 홀더를 모터 드라이버의 VMS와 GND에 연결
5. 아두이노를 로봇의 두 번째 본체 부분에 나사를 이용해 연결
6. 서보모터 브라켓을 조립한 후 초음파 센서와 케이블 타이로 고정
7. 6번째 과정에서 조립한 부품을 두 번째 본체에 연결
8. 4번째 과정에서 사용했던 배터리 홀더를 두 번째 본체와 연결
9. 로봇의 첫 번째 본체와 두 번째 본체를 나사를 이용해 전체 연결
10. 센서 쉴드를 아두이노 보드 위에 꽂아줌
11. 점퍼선을 이용하여 모터드라이버를 센서 쉴드에 연결
12. 점퍼선을 이용하여 서보모터와 초음파 센서를 센서 쉴드에 연결
13. 바퀴를 끼워줌

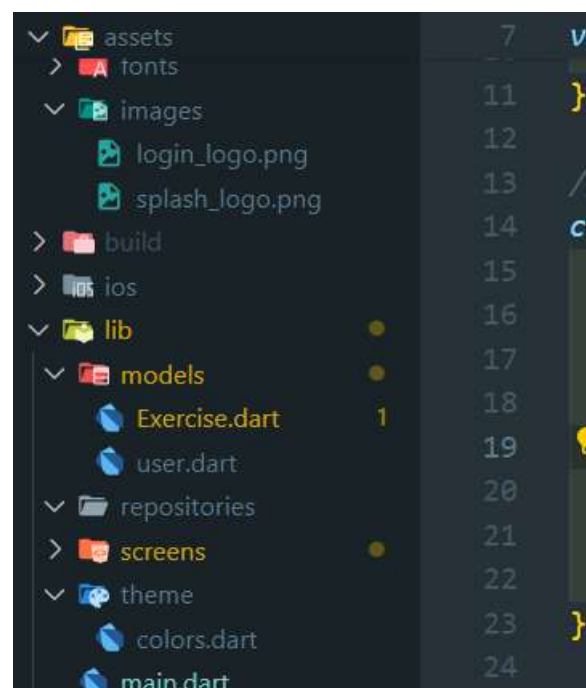
## (3) 화면 UI 제작



- 스플래시 화면: 5초 후 메인 화면으로 넘어가도록 코드 구현 완료하였습니다.
- 메인 화면: D - Day 설정 페이지, 달리기 연습 페이지로 넘어가는 코드까지는 구현했으나 아직 각 페이지UI를 완성하지 못했습니다. ⇒ 다음 주에 마저 다 완성할 예정입니다.

#### (4) 플러터 프로젝트 디렉토리 구조 셋팅

- 각 모듈별로 쪼개서 관리하기 위해 assets, screens, fonts등의 여러 폴더를 만들고 각 파일을 알맞은 폴더로 이동하였습니다.



#### (5) 기본 개념 공부

- 플러터 개발이 처음이어서, 유튜브 강의를 듣고 구글링을 하며 플러터의 기초 개념들을 공부하였습니다.