〈7조〉

임베디드 시스템
〈CNN, Tensorflow lite 기반 사용자 운동 상태에 따른 음악 추천 시스템〉

담당교수 : 조용범
실험날짜 : 2022. 11. 03
조 : 7 조
조원 : 201610925 정건희
201810528 고려욱
201810845 박종혁

1. Title

Final_Project

<CNN, Tensorflow lite 기반 사용자 운동 상태에 따른 음악 추천 시스템>

2. Name

7조

전기전자공학부 201610925 정건희

전기전자공학부 201810528 고려욱

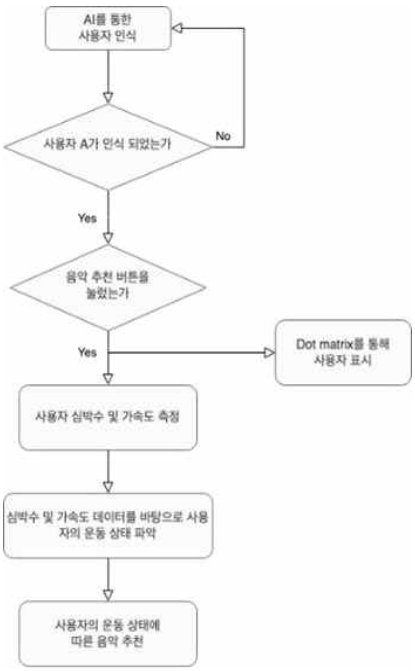
전기전자공학부 201810845 박종혁

3. Abstract

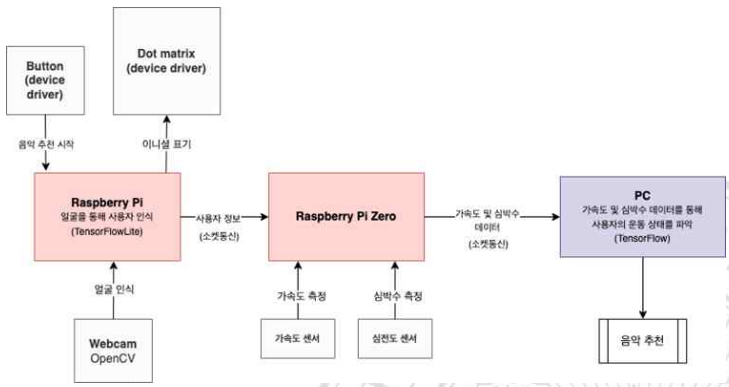
현재 음악 추천 시스템은 사용자가 선곡했던 기록, 혹은 음원 간의 유사도를 바탕으로 사용자에게 맞춤음악을 선곡해주고 있다.

우리는 이에 더해 사용자임을 기록하지 않더라도 얼굴 인식을 통해 누가 음악을 청취하는지 자동으로 인식하는 기능을 추가해보았다. 개인 기기를 통해서뿐 아니라 공용 기기로 음악을 들을 때, 가까운 미래에 cctv와 개인 인식 기술의 발달을 통해 보편화 될 것 같은 방식을 구현해보았다.

추가로 사용자의 운동 데이터를 웨어러블 기기를 통해 받아와 분석하여 사용자의 상황에 적합한 음악이 추천되도록 하는 가능성을 시험하였다.



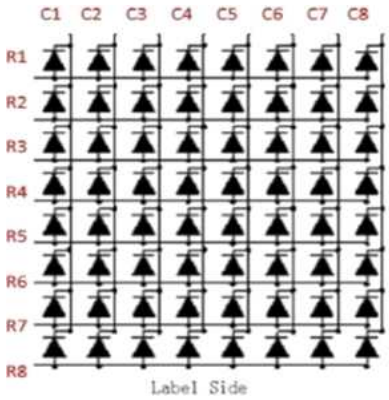
<시스템 흐름도>



<시스템 구성도>

4. Background

1) Dot-Matrix



Dot-Matix는 8개의 행과 8개의 열로 구성되어 총 64개의 Dot를 통해 원하는 정보를 Display할 수 있는 장치이다. Dot-Matrix의 각 행은 해당 행의 열 다이오드의 Anode가 각자 연결되어 있고 각 열은 해당 열의 행 다이오드의 Cathode가 각자 연결되어 있다.

2) Max30102 심박센서

MAX30102는 통합 맥박 산소 측정기 및 심박수 모니터 바이오 센서 모듈이다. 적색 LED 및 적외선 LED, 광 검출기, 광학 부품 및 주변 광 억제 기능이있는 저잡음 전자 회로로 구성되어 있다. 1.8V 전원 공급 장치와 내부 LED를위한 별도의 5.OV 전원 공급 장치를 갖추고 있다. 심박수 및 혈중 산소 획득을 위한 웨어러블 장치에 사용되며 손가락, 귓볼, 손목에 착용한다. 표준 I2C 호환 통신 인터페이스는 수집된 값을 마이크로 컨트롤러로 전송하여 심박수 및 혈액 산소 계산을 수행할 수 있다. 또한 칩은 소프트웨어를 통해 모듈을 종료할 수 있으며 대기 전류는 0에 가깝고 전원 공급 장치는 항상 유지된다. 뛰어난 성능으로 인해 이 칩은 삼성 Gala xy S 시리즈 휴대폰에 널리 사용되었다. 이전 세대의 MAX30100과 비교하여 이 칩은 유리 커버를 사용해 외부 및 내부 광 간섭을 효과적으로 제거하고 안정적인 성능을 제공한다.

3) 소켓 통신[1)]

소켓(Socket)이란 네트워크상에서 동작하는 프로그램 간 통신의 종착점(Endpoint), 접속의 끝부분이라 한다. 종착점인 Endpoint는 IP주소와 Port번호의 조합으로 이루어진 최종 목적지(PC, 핸드폰 등의 접속 연결부)를 나타낸다. 클라이언트(프로그램 1)와 서버(프로그램 2) 양쪽에서 서로에게 데이터 전달을 하는 방식의 양방향 통신으로, 소켓에서 서버는 요청을 기다릴 필요가 없다. 모든 연결은 2개의 앤드포인트로 유일하게 식별될 수 있다.

1) 참조 : https://bentist.tistory.com/35

소켓은 사전적으로 '전구 따위를 끼워 넣어 접속하게 하는 기구, 연결부'를 일컫는데 콘센트 구멍을 떠올리면 쉽다. 즉, 네트워크 상에서의 소켓은 멀리 떨어진 프로그램끼리 연결될 수 있게 만들어진 연결부인 것이다.

예를 들어, 휴대폰(프로그램 A)을 충전하기 위해 휴대폰 연결부 소켓과 보조배터리(프로그램 B)에 충전선을 꽂는 연결부 소켓이 있어야 한다. 또한 전기 충전이 제대로 작동되려면 220v라는 표준 규격에 맞게 만들어져야 하는 것처럼, 소켓 통신도 TCP, UDP 규약에 맞는 소켓을 미리 만들어 소켓에 꽂아 쓴다는 개념으로 프로그램 간에 데이터를 교환할 수 있다.

보통 스트리밍이나 실시간 채팅 등 실시간으로 데이터를 주고받아야 하는 경우 Connection을 자주 맺고 끊는 HTTP 통신보다 소켓 통신이 적합하다. 하지만, 소켓 통신은 계속해서 Connection을 들고 있기 때문에 HTTP 통신에 비해 많은 리소스가 소모된다.

3) TensorFlow

텐서플로는 구글에서 만든 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 라이브러리다. 텐서플로 자체는 기본적으로 C++로 구현 되어 있으며, Python, Java, Go 등 다양한 언어를 지원한다. 또한, 브라우저에서 실행 가능한 시각화 도우인 텐서보드(TensorBoard)를 제공하여, 딥러닝 학습 과정을 추적하는데 유용하게 사용된다.
TensorFlow에서 Tensor란 딥러닝에서 데이터를 표현하는 방식을 말한다. 즉, 텐서는 행렬로 표현할 수 있는 2차원 형태의 배열을 높은 차원으로 확장한 다차원 배열이다. TensorFlow에서 계산은 데이터 흐름 그래프로 이루어 진다. 즉, 텐서 형태의 데이터를 딥러닝 모델을 구성하는 연산들의 그래프를 따라 흐르면서 연산이 일어난다.

4) TensorFlow Lite
TensorFlow Lite은 Android, iOS, 리눅스 등 다양한 모바일 환경과 임베디드 시스템에서 머신 러닝 모델을 사용할 수 있게 하는 제품 단계 수준의 크로스 플랫폼 프레임워크이다. 2020년까지 전세계적으로 40억이 넘는 모바일 디바이스에서 TF Lite를 사용하고 있는 것으로 집계되었다. 구글의 대표적인 앱인 Photos, 유튜브, 구글 클라우드도 TFLite를 사용하고 있으며, 우버와 에어비엔비에서 역시 이 프레임 워크를 도입하여 이미지, 텍스트, 오디오, 콘텐츠 생성 등 다양한 어플리케이션에서 적용하고 있다.

TensorFlow Lite은 최적화된 모델을 다양한 하드웨어에서 돌아갈 수 있게 하는 TF Lite interpreter와 텐서플로우 모델을 인터프리터가 사용할 수 있는 효율적인 형태로 바꿔주고, 모델 용량을 줄이고 성능은 유지할 수 있도록 최적화 기능을 제공하는 TF Lite converter로 구성되어 있다.

5. Experimental Results

<Device Driver>

A. Source Code and Discussion

1. Button Device Driver

```
static ssize_t driver_read(struct file *File, char *user_buffer, size_t count, loff_t *offs) {
    int to_copy, not_copied, delta;
    char tmp[3] = " \n";

    /* Get amount of data to copy */
    to_copy = min(count, sizeof(tmp));

    /* Read value of touch */
    printk("Value of touch: %d\n", gpio_get_value(26));
    tmp[0] = gpio_get_value(26) + '0';

    /* Copy data to user */
    not_copied = copy_to_user(user_buffer, &tmp, to_copy);

    /* Calculate data */
    delta = to_copy - not_copied;

    return delta;
}
```

GPIO 26번 사용하여 driver_read함수만 활용하여 26번에 들어오는 신호 1/0을 읽어들인다.
사용하는 Buffer가 char형 배열이므로 해당 값에 '0'을 추가하여 int형을 char형으로 변경해줌

```
int main(int argc, char **argv) {

    char buff;
    char tmp;
    int flag = 1;
    char prev = 'r';
    int dev = open("/dev/my_gpio", O_RDWR); // if you want read-'O_RDONLY' write-'O_WRONLY', read&write-'O_RDWR'

    if (dev == -1) {
        printf("Opening was not possible!\n");
        return -1;
    }

    printf("Opening was successful!!\n");

    while(flag){
        read (dev, &buff, 1);
        printf("%c", buff);
        if(buff == '0'){
            printf("undetected\n");
        }
        else{
            printf("detected\n");
            flag = 0;
        }
    }

    close (dev);
    return 0;
}
```

반복문과 flag를 활용하여 버튼이 눌리는 것이 감지될 때까지 반복문 실행

2. Dot-Matrix Device Driver

```
static ssize_t driver_write(struct file* File, const char* user_buffer, size_t count, loff_t* offs) {
    int to_copy, not_copied, delta;
    unsigned int value = 0;

    /* Get amount of data to copy */
    to_copy = min(count, sizeof(value));

    /* Copy data to user */
    not_copied = copy_from_user(&value, user_buffer, to_copy);

    /* Row */
    if (value & (1 << 0)) {
        gpio_set_value(2, 1);
    }
    else {
        gpio_set_value(2, 0);
    }

    if (value & (1 << 1)) {
        gpio_set_value(3, 1);
    }
    else {
        gpio_set_value(3, 0);
    }

    if (value & (1 << 2)) {
        gpio_set_value(4, 1);
    }
    else {
        gpio_set_value(4, 0);
    }

    if (value & (1 << 3)) {
        gpio_set_value(27, 1);
    }
    else {
        gpio_set_value(27, 0);
    }

    if (value & (1 << 4)) {
        gpio_set_value(22, 1);
    }
    else {
        gpio_set_value(22, 0);
    }

    if (value & (1 << 5)) {
        gpio_set_value(10, 1);
    }
    else {
        gpio_set_value(10, 0);
    }
```

```
    if (value & (1 << 6)) {
        gpio_set_value(9, 1);
    }
    else {
        gpio_set_value(9, 0);
    }

    if (value & (1 << 7)) {
        gpio_set_value(11, 1);
    }
    else {
        gpio_set_value(11, 0);
    }
    /* Column */
    if (value & (1 << 15)) {
        gpio_set_value(14, 0);
    }
    else {
        gpio_set_value(14, 1);
    }
```

```
if (value & (1 << 14)) {
    gpio_set_value(15, 0);
}
else {
    gpio_set_value(15, 1);
}

if (value & (1 << 13)) {
    gpio_set_value(23, 0);
}
else {
    gpio_set_value(23, 1);
}

if (value & (1 << 12)) {
    gpio_set_value(24, 0);
}
else {
    gpio_set_value(24, 1);
}

if (value & (1 << 11)) {
    gpio_set_value(25, 0);
}
else {
    gpio_set_value(25, 1);
}

if (value & (1 << 10)) {
    gpio_set_value(8, 0);
```

```
else {
    gpio_set_value(8, 1);
}
if (value & (1 << 9)) {
    gpio_set_value(7, 0);
}
else {
    gpio_set_value(7, 1);
}
if (value & (1 << 8)) {
    gpio_set_value(18, 0);
}
else {
    gpio_set_value(18, 1);
}
```

8개 Row, 8개 Column에 대해 값을 Write하기 위해 GPIO 16개와 driver_write함수를 사용한다. 8개의 행은 Dot-Matirx의 구조상 다이오드의 Anode의 값들이 묶여 작동되므로 해당 행을 제어하여 불이 들어오도록 하고 싶다면 값 1을, 8개의 열은 Cathode의 값들이 묶여 작동되므로 해당 열을 제어하여 불이 들어오도록 하고 싶다면 값 0을 전달하여야 한다.

각 행과 열 모두의 정보를 사용자 buffer에서 값을 전달받아 동시에 제어하여야 하므로 데이터는 총 16bit의 데이터가 넘어와 저장될 수 있도록 하여야 한다. 따라서, unsigned int data type을 활용하여 16bit의 데이터를 저장할 수 있도록 하였다. 각 행들은 2^0 ~ 2^7 까지의 각 2의 제곱수 값과 bit연산을 통해 각 행이 켜져야 하는지를 결정한다. 각 열들은 2^15~2^8 까지의 각 2의 제곱수 값과 bit연산을 통해 각 열이 켜져야 하는지를 결정한다.

```
static struct termios init_setting, new_setting;

unsigned int dot_J[8] = { 0x3801,0x1002,0x1004,0x1008,0x1010,0x9020,0x6040,0x0080 };
unsigned int dot_K[8] = { 0x8801,0x9002,0xA004,0xC008,0xA010,0x9020,0x8840,0x0080 };
unsigned int dot_P[8] = { 0xF001,0x8802,0x8804,0xF008,0x8010,0x8020,0x8040,0x0080 };


int main(int argc, char** argv)
{
    //IO주차
    unsigned int data[8];
    int tmp_n = 0;
    int count = 1;
    int delay_time = 0;;
    char buff;

    //device IO주차
    int devIO = open("/dev/my_dot_matrix", O_RDWR);

    if (devIO == -1) {
        printf("OpeningIO was not possible!₩n");
        return -1;
    }
    printf("OpeningIO was successfull!₩n");

    if(argv[1][0] == 'p'){
        for(int k = 0; k < 8; k++){
            data[k] = dot_P[k];
        }
    }
    else if(argv[1][0] == 'k') {
        for(int k = 0; k < 8; k++){
            data[k] = dot_K[k];
        }
    }
    else if(argv[1][0] == 'j'){
        for(int k = 0; k < 8; k++){
            data[k] = dot_J[k];
        }
    }

    while(count){
        write(devIO, &data[tmp_n], 2);
        usleep(delay_time);

        tmp_n++;
        count++;
        if (tmp_n > 7) {
            tmp_n = 0;
        }
        if(count > 40000){
            count = 0;
        }
    }
```

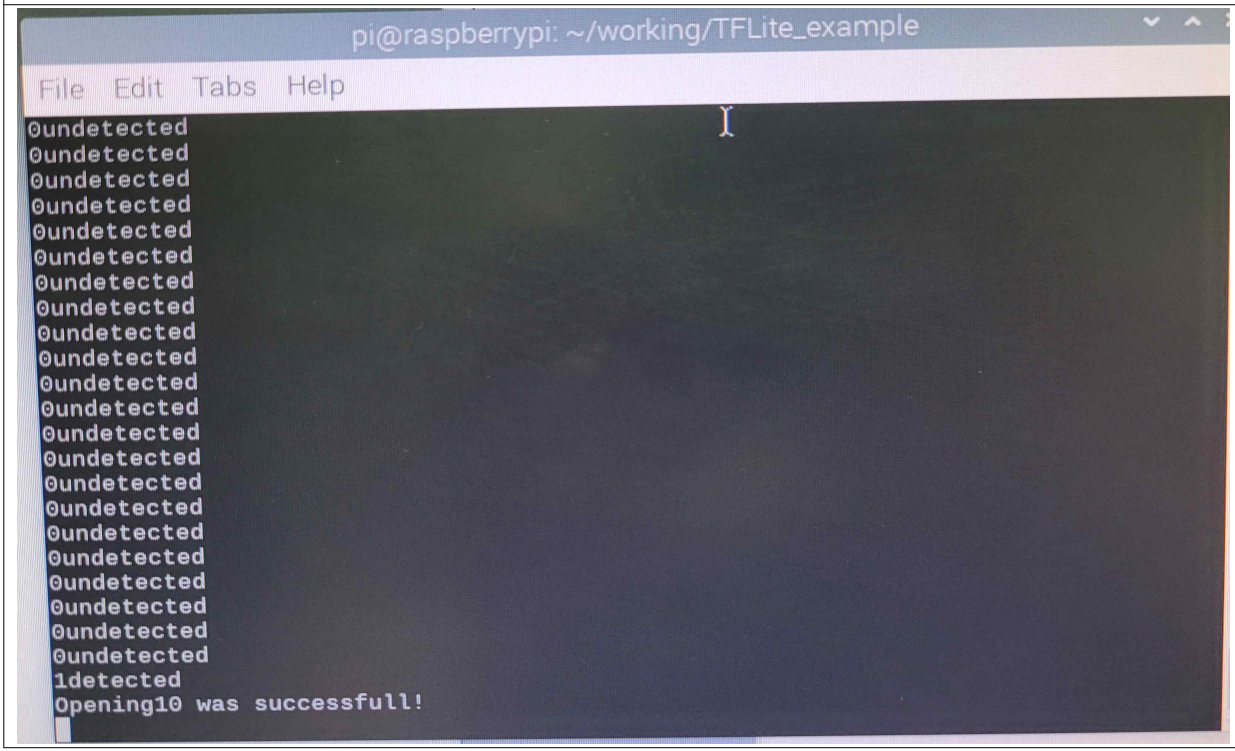
```
    write(devIO, 0x0000, 2);
    close(devIO);
    return 0;
}
```

표현하고자 하는 Alphabet 역시 unsigned int data type을 통해 미리 지정해 둔다. 16bit 중 앞 8bit는 열을 제어하는 데이터 값, 뒤 8bit는 순서대로 8개의 행이 켜져야 할 때를 알려주는 데이터 값으로 구성하여 delaytime없이 1행부터 8행까지 반복적으로 display를 해주게 된다.

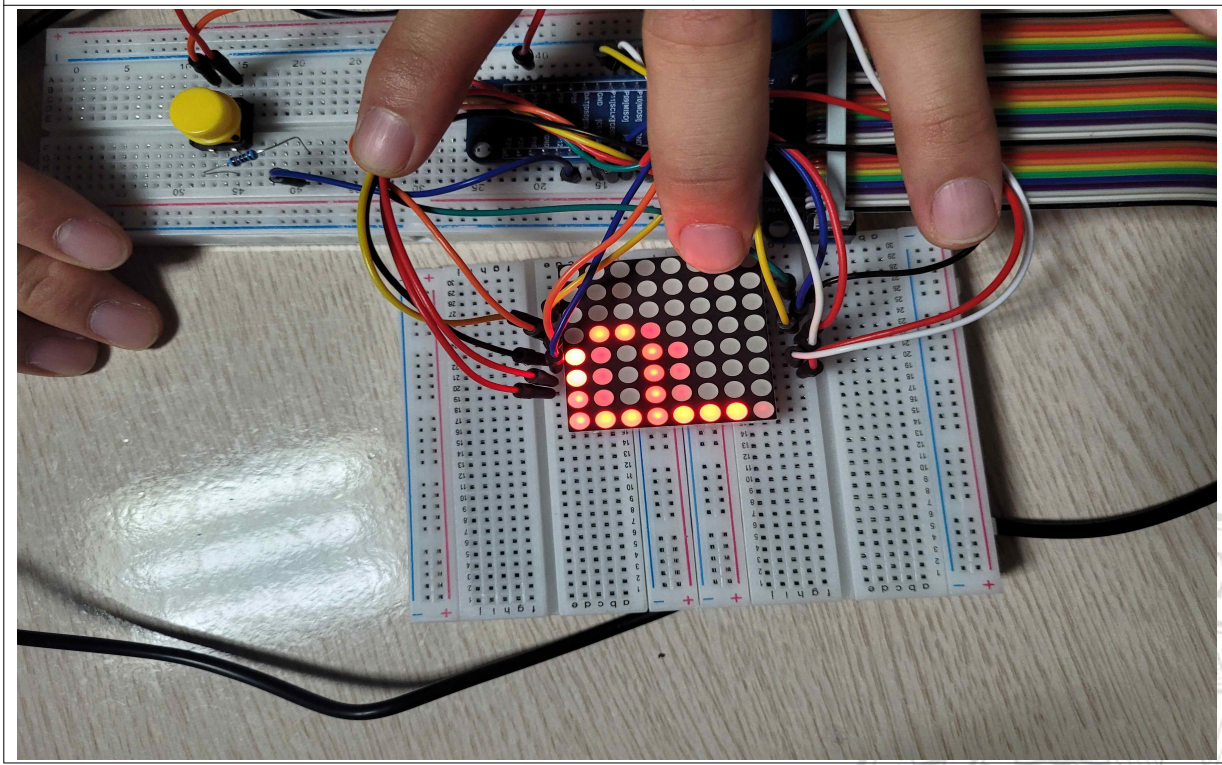
지정한 Alphabet은 실행파일 호출 시 argv를 통해 입력받게 되며 입력받은 알파벳에 해당하는 unsigned int 형 배열을 data배열에 저장하고 반복문을 통해 순차적으로 데이터를 device_driver에 넘겨주어 Dot-Matrix Display를 완성한다.

B. Data

Button Device 실행 시 Terminal 창



Button Device 실행 시 Terminal 창



<음악 추천 AI - Tensorflow>

A. Source Code and Discussion

1. hrcalc.py (심박 계산 파일)

```
# -*-coding:utf-8
import numpy as np
# 25 samples per second (in algorithm.h)
SAMPLE_FREQ = 25

# taking moving average of 4 samples when calculating HR
# in algorithm.h, "DONOT CHANGE" comment is attached
MA_SIZE = 4
# sampling frequency * 4 (in algorithm.h)
BUFFER_SIZE = 100

# this assumes ir_data and red_data as np.array
def calc_hr_and_spo2(ir_data, red_data):
    """
    By detecting  peaks of PPG cycle and corresponding AC/DC
    of red/infra-red signal, the an_ratio for the SPO2 is computed.
    """
    # get dc mean
    ir_mean = int(np.mean(ir_data))

    # remove DC mean and inver signal
    # this lets peak detecter detect valley
    x = -1 * (np.array(ir_data) - ir_mean)

    # 4 point moving average
    # x is np.array with int values, so automatically casted to int
    for i in range(x.shape[0] - MA_SIZE):
        x[i] = np.sum(x[i:i+MA_SIZE]) / MA_SIZE

    # calculate threshold
    n_th = int(np.mean(x))
    n_th = 30 if n_th < 30 else n_th  # min allowed
    n_th = 60 if n_th > 60 else n_th  # max allowed

    ir_valley_locs, n_peaks = find_peaks(x, BUFFER_SIZE, n_th, 4, 15)
    # print(ir_valley_locs[:n_peaks], ",", end="")
    peak_interval_sum = 0
    if n_peaks >= 2:
        for i in range(1, n_peaks):
            peak_interval_sum += (ir_valley_locs[i] - ir_valley_locs[i-1])
        peak_interval_sum = int(peak_interval_sum / (n_peaks - 1))
        hr = int(SAMPLE_FREQ * 60 / peak_interval_sum)
        hr_valid = True
    else:
```

```
    hr = -999  # unable to calculate because # of peaks are too small
    hr_valid = False

# ---------spo2---------
# find precise min near ir_valley_locs (???)
exact_ir_valley_locs_count = n_peaks

# find ir-red DC and ir-red AC for SPO2 calibration ratio
# find AC/DC maximum of raw

# FIXME: needed??
for i in range(exact_ir_valley_locs_count):
    if ir_valley_locs[i] > BUFFER_SIZE:
        spo2 = -999  # do not use SPO2 since valley loc is out of range
        spo2_valid = False
        return hr, hr_valid, spo2, spo2_valid

i_ratio_count = 0
ratio = []

# find max between two valley locations
# and use ratio between AC component of Ir and Red DC component of Ir and Red for SpO2
red_dc_max_index = -1
ir_dc_max_index = -1
for k in range(exact_ir_valley_locs_count-1):
    red_dc_max = -16777216
    ir_dc_max = -16777216
    if ir_valley_locs[k+1] - ir_valley_locs[k] > 3:
        for i in range(ir_valley_locs[k], ir_valley_locs[k+1]):
            if ir_data[i] > ir_dc_max:
                ir_dc_max = ir_data[i]
                ir_dc_max_index = i
            if red_data[i] > red_dc_max:
                red_dc_max = red_data[i]
                red_dc_max_index = i

        red_ac = int((red_data[ir_valley_locs[k+1]] - red_data[ir_valley_locs[k]]) * (red_dc_max_index - ir_valley_locs[k
         ]))
        red_ac = red_data[ir_valley_locs[k]] + int(red_ac / (ir_valley_locs[k+1] - ir_valley_locs[k]))
        red_ac = red_data[red_dc_max_index] - red_ac  # subtract linear DC components from raw

        ir_ac = int((ir_data[ir_valley_locs[k+1]] - ir_data[ir_valley_locs[k]]) * (ir_dc_max_index - ir_valley_locs[k]))
        ir_ac = ir_data[ir_valley_locs[k]] + int(ir_ac / (ir_valley_locs[k+1] - ir_valley_locs[k]))
        ir_ac = ir_data[ir_dc_max_index] - ir_ac  # subtract linear DC components from raw

        nume = red_ac * ir_dc_max
        denom = ir_ac * red_dc_max
        if (denom > 0 and i_ratio_count < 5) and nume != 0:
            # original cpp implementation uses overflow intentionally.
            # but at 64-bit OS, Pyhthon 3.X uses 64-bit int and nume*100/denom does not trigger overflow
            # so using bit operation ( &0xffffffff ) is needed
```

```
                ratio.append(int(((nume * 100) & 0xffffffff) / denom))
                i_ratio_count += 1

    # choose median value since PPG signal may vary from beat to beat
    ratio = sorted(ratio)  # sort to ascending order
    mid_index = int(i_ratio_count / 2)

    ratio_ave = 0
    if mid_index > 1:
        ratio_ave = int((ratio[mid_index-1] + ratio[mid_index])/2)
    else:
        if len(ratio) != 0:
            ratio_ave = ratio[mid_index]

    # why 184?
    # print("ratio average: ", ratio_ave)
    if ratio_ave > 2 and ratio_ave < 184:
        # -45.060 * ratioAverage * ratioAverage / 10000 + 30.354 * ratioAverage / 100 + 94.845
        spo2 = -45.060 * (ratio_ave**2) / 10000.0 + 30.054 * ratio_ave / 100.0 + 94.845
        spo2_valid = True
    else:
        spo2 = -999
        spo2_valid = False

    return hr, hr_valid, spo2, spo2_valid


def find_peaks(x, size, min_height, min_dist, max_num):
    """
    Find at most MAX_NUM peaks above MIN_HEIGHT separated by at least MIN_DISTANCE
    """
    ir_valley_locs, n_peaks = find_peaks_above_min_height(x, size, min_height, max_num)
    ir_valley_locs, n_peaks = remove_close_peaks(n_peaks, ir_valley_locs, x, min_dist)

    n_peaks = min([n_peaks, max_num])

    return ir_valley_locs, n_peaks


def find_peaks_above_min_height(x, size, min_height, max_num):
    """
    Find all peaks above MIN_HEIGHT
    """

    i = 0
    n_peaks = 0
    ir_valley_locs = []  # [0 for i in range(max_num)]
    while i < size - 1:
        if x[i] > min_height and x[i] > x[i-1]:  # find the left edge of potential peaks
            n_width = 1
            # original condition i+n_width < size may cause IndexError
```

```
            # so I changed the condition to i+n_width < size - 1
            while i + n_width < size - 1 and x[i] == x[i+n_width]:  # find flat peaks
                n_width += 1
            if x[i] > x[i+n_width] and n_peaks < max_num:  # find the right edge of peaks
                # ir_valley_locs[n_peaks] = i
                ir_valley_locs.append(i)
                n_peaks += 1  # original uses post increment
                i += n_width + 1
            else:
                i += n_width
        else:
            i += 1

    return ir_valley_locs, n_peaks


def remove_close_peaks(n_peaks, ir_valley_locs, x, min_dist):
    """
    Remove peaks separated by less than MIN_DISTANCE
    """

    # should be equal to maxim_sort_indices_descend
    # order peaks from large to small
    # should ignore index:0
    sorted_indices = sorted(ir_valley_locs, key=lambda i: x[i])
    sorted_indices.reverse()

    # this "for" loop expression does not check finish condition
    # for i in range(-1, n_peaks):
    i = -1
    while i < n_peaks:
        old_n_peaks = n_peaks
        n_peaks = i + 1
        # this "for" loop expression does not check finish condition
        # for j in (i + 1, old_n_peaks):
        j = i + 1
        while j < old_n_peaks:
            n_dist = (sorted_indices[j] - sorted_indices[i]) if i != -1 else (sorted_indices[j] + 1)  # lag-zero peak of autocorr i
             s at index -1
            if n_dist > min_dist or n_dist < -1 * min_dist:
                sorted_indices[n_peaks] = sorted_indices[j]
                n_peaks += 1  # original uses post increment
            j += 1
        i += 1

    sorted_indices[:n_peaks] = sorted(sorted_indices[:n_peaks])

    return sorted_indices, n_peaks
```

2. max30102.py (심박센서 드라이버)

```
from time import sleep
import RPi.GPIO as GPIO
from smbus2 import SMBus
I2C_WRITE_ADDR = 0xAE
I2C_READ_ADDR = 0xAF

# register address-es
REG_INTR_STATUS_1 = 0x00
REG_INTR_STATUS_2 = 0x01

REG_INTR_ENABLE_1 = 0x02
REG_INTR_ENABLE_2 = 0x03

REG_FIFO_WR_PTR = 0x04
REG_OVF_COUNTER = 0x05
REG_FIFO_RD_PTR = 0x06
REG_FIFO_DATA = 0x07
REG_FIFO_CONFIG = 0x08

REG_MODE_CONFIG = 0x09
REG_SPO2_CONFIG = 0x0A
REG_LED1_PA = 0x0C

REG_LED2_PA = 0x0D
REG_PILOT_PA = 0x10
REG_MULTI_LED_CTRL1 = 0x11
REG_MULTI_LED_CTRL2 = 0x12

REG_TEMP_INTR = 0x1F
REG_TEMP_FRAC = 0x20
REG_TEMP_CONFIG = 0x21
REG_PROX_INT_THRESH = 0x30
REG_REV_ID = 0xFE
REG_PART_ID = 0xFF

# currently not used
MAX_BRIGHTNESS = 255

class MAX30102():
    # by default, this assumes that physical pin 7 (GPIO 4) is used as interrupt
    # by default, this assumes that the device is at 0x57 on channel 1
    def __init__(self, channel=1, address=0x57, gpio_pin=7):
        print("Channel: {0}, address: {1}".format(channel, address))
        self.address = address
        self.channel = channel
        self.bus = SMBus(self.channel)
        self.interrupt = gpio_pin
```

```
        # set gpio mode
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(self.interrupt, GPIO.IN)
        self.reset()
        sleep(1)  # wait 1 sec

        # read & clear interrupt register (read 1 byte)
        reg_data = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_1, 1)
        # print("[SETUP] reset complete with interrupt register0: {0}".format(reg_data))
        self.setup()
        # print("[SETUP] setup complete")

    def shutdown(self):
        """
        Shutdown the device.
        """
        self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [0x80])

    def reset(self):
        """
        Reset the device, this will clear all settings,
        so after running this, run setup() again.
        """
        self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [0x40])

    def setup(self, led_mode=0x03):
        """
        This will setup the device with the values written in sample Arduino code.
        """
        # INTR setting
        # 0xc0 : A_FULL_EN and PPG_RDY_EN = Interrupt will be triggered when
        # fifo almost full & new fifo data ready
        self.bus.write_i2c_block_data(self.address, REG_INTR_ENABLE_1, [0xc0])
        self.bus.write_i2c_block_data(self.address, REG_INTR_ENABLE_2, [0x00])

        # FIFO_WR_PTR[4:0]
        self.bus.write_i2c_block_data(self.address, REG_FIFO_WR_PTR, [0x00])
        # OVF_COUNTER[4:0]
        self.bus.write_i2c_block_data(self.address, REG_OVF_COUNTER, [0x00])
        # FIFO_RD_PTR[4:0]
        self.bus.write_i2c_block_data(self.address, REG_FIFO_RD_PTR, [0x00])

        # 0b 0100 1111
        # sample avg = 4, fifo rollover = false, fifo almost full = 17
        self.bus.write_i2c_block_data(self.address, REG_FIFO_CONFIG, [0x4f])

        # 0x02 for read-only, 0x03 for SpO2 mode, 0x07 multimode LED
        self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [led_mode])
        # 0b 0010 0111
        # SPO2_ADC range = 4096nA, SPO2 sample rate = 100Hz, LED pulse-width = 411uS
        self.bus.write_i2c_block_data(self.address, REG_SPO2_CONFIG, [0x27])
```

```
        # choose value for ~7mA for LED1
        self.bus.write_i2c_block_data(self.address, REG_LED1_PA, [0x24])
        # choose value for ~7mA for LED2
        self.bus.write_i2c_block_data(self.address, REG_LED2_PA, [0x24])
        # choose value fro ~25mA for Pilot LED
        self.bus.write_i2c_block_data(self.address, REG_PILOT_PA, [0x7f])

    # this won't validate the arguments!
    # use when changing the values from default
    def set_config(self, reg, value):
        self.bus.write_i2c_block_data(self.address, reg, value)

    def read_fifo(self):
        """
        This function will read the data register.
        """
        red_led = None
        ir_led = None

        # read 1 byte from registers (values are discarded)
        reg_INTR1 = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_1, 1)
        reg_INTR2 = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_2, 1)

        # read 6-byte data from the device
        d = self.bus.read_i2c_block_data(self.address, REG_FIFO_DATA, 6)

        # mask MSB [23:18]
        red_led = (d[0] << 16 | d[1] << 8 | d[2]) & 0x03FFFF
        ir_led = (d[3] << 16 | d[4] << 8 | d[5]) & 0x03FFFF

        return red_led, ir_led

    def read_sequential(self, amount=200):
        """
        This function will read the red-led and ir-led `amount` times.
        This works as blocking function.
        """
        red_buf = []
        ir_buf = []
        for i in range(amount):
            while(GPIO.input(self.interrupt) == 1):
                # wait for interrupt signal, which means the data is available
                # do nothing here
                pass

            red, ir = self.read_fifo()
            red_buf.append(red)
            ir_buf.append(ir)
        return red_buf, ir_buf
```

3. Socket_raspzero.py (심박, 가속도센서 소켓통신)

```
import socket
import smbus                    #import SMBus module of I2C
import max30102
import hrcalc
import threading
import time
HOST = '192.168.137.247'
PORT = 9999
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, PORT))
server_socket.listen()
client_socket, addr = server_socket.accept()
print('receiving data...', addr)
while True:
        data = client_socket.recv(1024)
        if not data:
                print('Not receive', addr, data.decode())
                client_socket.close()
                server_socket.close()
                break;
        print('Received from',addr,data.decode())
        Received_data = data.decode()
        client_socket.close()
        server_socket.close()
        break;
#Socket interfacet setting
HOST = '192.168.137.1'
PORT = 9999
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
#define global variable for send data
global sAx, sAy, sAz, sGx, sGy, sGz, shr2, ssp2
#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47
def MPU_Init():
        #write to sample rate register
        bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)
```

```
        #Write to power management register
        bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)
        #Write to Configuration register
        bus.write_byte_data(Device_Address, CONFIG, 0)
        #Write to Gyro configuration register
        bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)
        #Write to interrupt enable register
        bus.write_byte_data(Device_Address, INT_ENABLE, 1)
def read_raw_data(addr):
        #Accelero and Gyro value are 16-bit
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)
        #concatenate higher and lower value
        value = ((high << 8) | low)
        #to get signed value from mpu6050
        if(value > 32768):
                value = value - 65536
        return value
bus = smbus.SMBus(1)    # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68   # MPU6050 device address
MPU_Init()
def Gyroscope():
        print (" Reading Data of Gyroscope and Accelerometer")
        #Setting global variable
        global sAx, sAy, sAz, sGx, sGy, sGz
        while True:
                #Read Accelerometer raw value
                acc_x = read_raw_data(ACCEL_XOUT_H)
                acc_y = read_raw_data(ACCEL_YOUT_H)
                acc_z = read_raw_data(ACCEL_ZOUT_H)
                #Read Gyroscope raw value
                gyro_x = read_raw_data(GYRO_XOUT_H)
                gyro_y = read_raw_data(GYRO_YOUT_H)
                gyro_z = read_raw_data(GYRO_ZOUT_H)
                #Full scale range +/- 250 degree/C as per sensitivity scale factor
                Ax = acc_x/16384.0
                Ay = acc_y/16384.0
                Az = acc_z/16384.0
                Gx = gyro_x/131.0
                Gy = gyro_y/131.0
                Gz = gyro_z/131.0
                now = time
                #data for sending to server
                sAx = Ax
                sAy = Ay
                sAz = Az
                sGx = Gx
                sGy = Gy
                sGz = Gz
                print ("Gx=%.2f" %Gx, u'₩u00b0'+ "/s", "₩tGy=%.2f" %Gy, u'₩u00b0'+ "/s", "₩tGz=%.2f" %Gz, u'₩u00b0'+ "/s
            ", "₩tAx=%.2f g" %Ax, "₩tAy=%.2f g" %Ay, "₩tAz=%.2f g" %Az)
```

```
            #f = open("data.csv", "a")
            #f.write(now.strftime('%Y-%m-%d  %H:%M:%S')+","+str(Gx)+","+str(Gy)+","+str(Gz)+","+str(Ax)+","+str(Ay)+","+
        str(Az)+"₩n")
            #f.close()
            time.sleep(1)
if __name__ == '__main__':
    t = threading.Thread(target=Gyroscope)
    t.start()
    #fd = open("heartbit_data.csv", "a")
    #fd.write("Date, Gx, Gy, Gz, Ax, Ay, Az₩n")
    #fd.write("Date, HeartBit, SPO2₩n₩n")
    #fd.close()
    global shr2, ssp2
    m = max30102.MAX30102()
    print (" Reading Data of MAX301002")
    while True:
        red, ir = m.read_sequential()
        hr,hrb,sp,spb = hrcalc.calc_hr_and_spo2(ir, red)
        if(hrb == True and hr != -999):
            if(spb == True and sp != -999):
                hr2 = int(hr)
                print("Heart Rate : ", hr2)
                sp2 = int(sp)
                print("SPO2       : ", sp2)

                #Setting global variable
                shr2 = hr2
                ssp2 = sp2
                now = time
                #fd = open("data.csv", "a")
                #fd.write(now.strftime('%Y-%m-%d  %H:%M:%S')+","+str(Gx)+","+str(Gy)+","+str(Gz)+","+str(Ax)+","
        +str(Ay)+","+str(Az)+"₩n","+str(hr2)+","+str(sp2)+"₩n")
                #fd.close()
                sendata = str(sGx)+' '+str(sGy)+' '+str(sGz)+' '+str(sAx)+' '+str(sAy)+' '+str(sAz)+' '+str(shr2)+' '+str
        (ssp2)+' '+str(Received_data)
                client_socket.sendall(sendata.encode())
                client_socket.close()
```

4. Socket_PC.py (PC에서 값 수신, 모델 실행)

```
import socket #소켓 통신
import os
from playsound import playsound

from sklearn import preprocessing
from sklearn.metrics import accuracy_score
import numpy as np #머신러닝 세팅
import sklearn
import pandas as pd
import tensorflow as tf
```

```
from tensorflow import keras
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import load_model

#소켓 통신
HOST = '192.168.137.1'
PORT = 9999

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, PORT))
server_socket.listen()
client_socket, addr = server_socket.accept()

print('생체 정보 받기 위해 라즈베리파이에 연결 중..', addr)

while True:
    data = client_socket.recv(1024)
    if not data:
        print('Not receive', addr, data.decode())
        client_socket.close()
        server_socket.close()
        break;
    print('Received from', addr, data.decode())
    Received_df = data.decode()
    client_socket.close()
    server_socket.close()
    break;


#모델 불러오기
model = load_model('Bodydata_CNN_model.h5')
print('model로드 완료')

#불러온 데이터 csv파일에 추가
Received_df = Received_df.split()
Name_data = Received_df[8]
Received_df = Received_df[0:8]
r_df = pd.DataFrame(Received_df)
r_df = r_df.T
r_df.columns = ['Gx', 'Gy', 'Gz', 'Ax', 'Ay', 'Az', 'Heartbeat', 'SPO2']
#
if Name_data == 'jgh':
    Name = '정건희'
elif Name_data == 'krw':
    Name = '고려욱'
elif Name_data == 'pjh':
```

```
    Name = '박종혁'
print('₩n',Name,'님에 대한 음악 추천')
#
print('₩n데이터 가공 완료')

df = pd.read_csv('finaldata.csv')
label = df[['label']]
df = df.drop(columns=['date','label'])
df = pd.concat([df,r_df], ignore_index = True)
df_scaled = sklearn.preprocessing.scale(df)
df = pd.DataFrame(df_scaled, columns=df.columns)

df = df.astype(np.float32)
ss = StandardScaler()
df = ss.fit_transform(df)
col_name_for_pred = ['Gx', 'Gy', 'Gz', 'Ax', 'Ay', 'Az', 'Heartbeat', 'SPO2']
new_df = pd.DataFrame(df, columns = col_name_for_pred)
print('₩n데이터 추가 후 예측 중..')
y_pred = model.predict(new_df.loc[[5763]])

np.set_printoptions(suppress=True, precision=5)
print('₩n------발라드 댄스 포크 힙합 랜덤 락에 대한 일치도--------')
print(y_pred)

prediction = np.argmax(y_pred[0])
#print(prediction)

if prediction == 0:
  print('₩n결과는 ballad')
  playsound("ballad.mp3")
elif prediction == 1:
  print('결과는 dance')
  playsound("dance.mp3")
elif prediction == 2:
  print('결과는 folk')
  playsound("folk.mp3")
elif prediction == 3:
  print('결과는 hiphop')
  playsound("hiphop.mp3")
elif prediction == 4:
  print('결과는 random')
  playsound("random.mp3")
else:
  print('결과는 rock')
  playsound("rock.mp3")

os.system('pause')
```

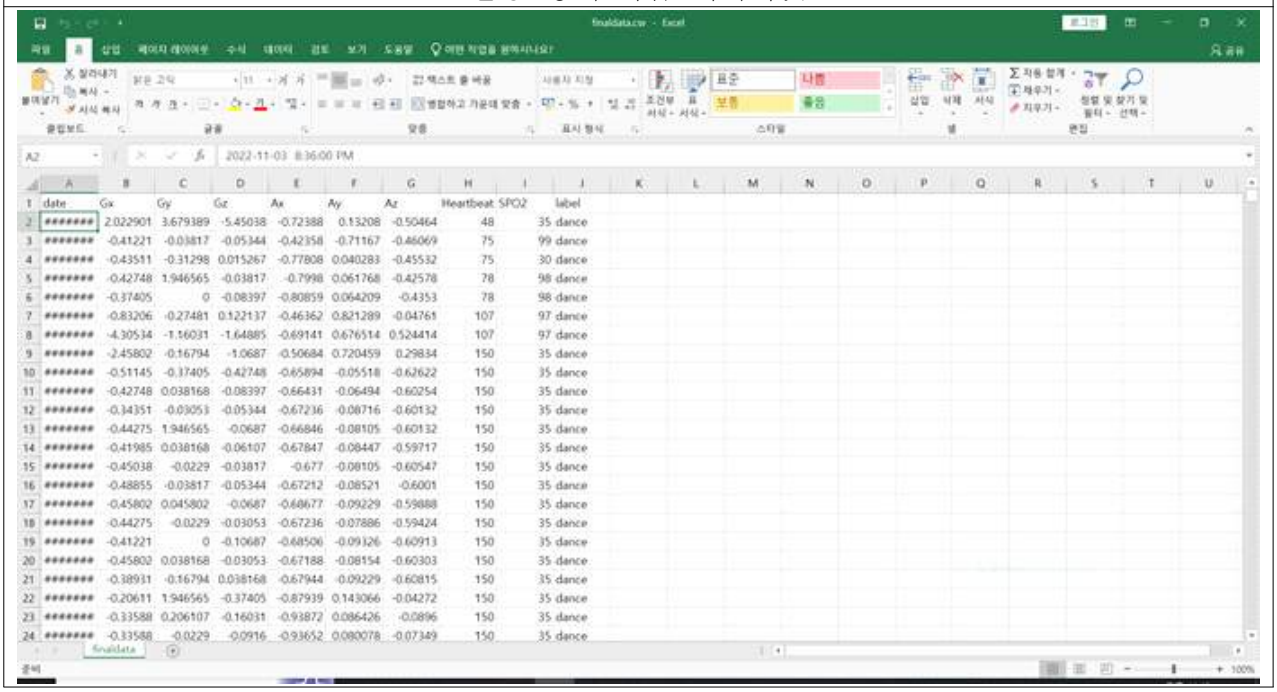
5. Bodydata.py (Colab 학습 코드)

```
#라이브러리 설치 및 준비
import numpy as np
import sklearn
import pandas as pd
import IPython
from sklearn import preprocessing

#필요시 구글드라이브 마운트
from google.colab import drive
drive.mount('/content/drive')

#Normalization csv파일 저장
df = pd.read_csv('drive/MyDrive/finaldata.csv')
label = df[['label']]
df = df.drop(columns=['date','label'])
df_scaled = sklearn.preprocessing.scale(df)
df = pd.DataFrame(df_scaled, columns=df.columns)
new_df=pd.concat([df,label],axis =1)
new_df.to_csv('./norm_df.csv')
new_df.head()

#파일에서 feature불러오기
new_df = pd.read_csv('drive/MyDrive/norm_df.csv')
new_df.drop(['Unnamed: 0'],axis=1, inplace = True)
new_df.head()

#데이터 학습 실시
# ============================ import lib, package ============================

import numpy as np
import pandas as pd
import tensorflow as tf

from tensorflow import keras
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# ============================ Function definition ============================
def get_data(): # load all data
  data = pd.read_csv("norm_df.csv");
  data.drop(['Unnamed: 0'],axis=1, inplace = True)
  return data

def seperate_x_y(data) : # Separate label data from all data.
  x = data.drop("label", axis = 1)
```

```
  y = data.iloc[:, -1]
  return x, y

def encode_label(data) : # Categorizing label data
    cvt = preprocessing.LabelEncoder()
    y = cvt.fit_transform(data)
    return y

def scale_data(data) : # Scaling feature data
  ss = StandardScaler()
  x = ss.fit_transform(np.array(data.iloc[:, :], dtype = float))

  return x

def seperate_train_test(x, y, test_size, random_state) : # train_test_split()
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = test_size, random_state = random_state)
  return x_train, x_test, y_train, y_test

def create_model(x_train, verbose) :#  create the model

  model=tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, activation='relu', input_shape=(x_train.shape[1],)),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(10,activation='softmax'),
  ])
  return model

def train_Model(model,x_train, x_test, y_train, y_test, epochs, optimizer, batch_size): # training the model
    model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics='accuracy')
    model.fit(x_train, y_train, validation_data = (x_test,y_test), epochs = epochs, batch_size = batch_size)
    return model

# ============================ Run and Check ============================
data_all = get_data()
x, y = seperate_x_y(data_all)#피쳐 빼지 않고 계산
y_encoded = encode_label(y)
x_scaled = scale_data(x)
x_train, x_test, y_train, y_test = seperate_train_test(x_scaled, y_encoded, test_size = 0.33, random_state = 42)
model = create_model(x_train, verbose = 1)
model_trained = train_Model(model, x_train, x_test, y_train, y_test, epochs = 150, optimizer = 'adam', batch_size = 128)
score = model.evaluate(x_test, y_test, verbose=1)
print(model.summary())
model.save('Bodydata_CNN_model.h5')# weight 담긴 모델파일 생성
```

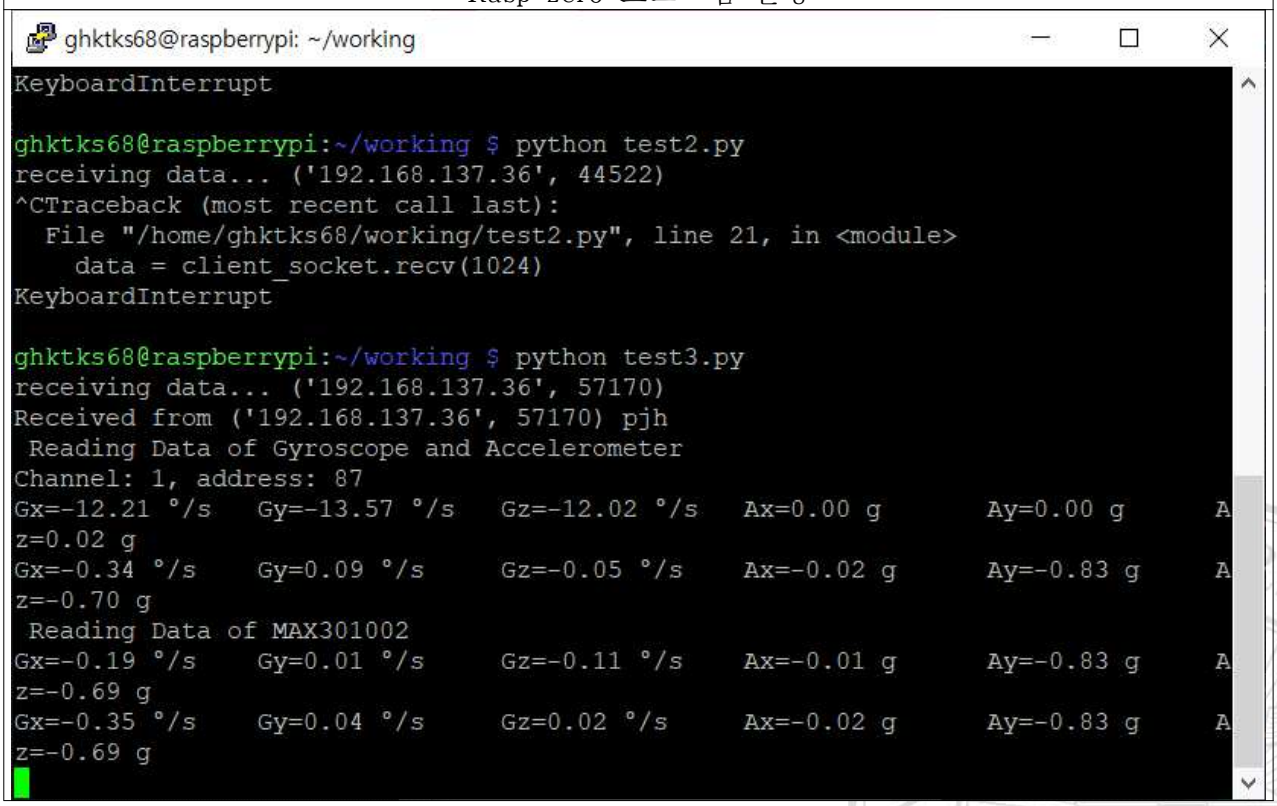
B. Data

운동 상태 학습 데이터셋



	A	B	C	D	E	F	G	H	I	J
1	date	Gx	Gy	Gz	Ax	Ay	Az	Heartbeat	SPO2	label
2	********	2.022901	3.679389	-5.45038	-0.72388	0.13208	-0.50464	48		35 dance
3	********	-0.41221	-0.03817	-0.05344	-0.42358	-0.71167	-0.46069	75		99 dance
4	********	-0.43511	-0.31298	0.015267	-0.77808	0.040283	-0.45532	75		30 dance
5	********	-0.42748	1.946565	-0.03817	-0.7998	0.061768	-0.42578	78		98 dance
6	********	-0.37405	0	-0.08397	-0.80859	0.064209	-0.4353	78		98 dance
7	********	-0.83206	-0.27481	0.122137	-0.46362	0.821289	-0.04761	107		97 dance
8	********	-4.30534	-1.16031	-1.64885	-0.69141	0.676514	0.524414	107		97 dance
9	********	-2.45802	-0.16794	-1.0687	-0.50684	0.720459	0.29834	150		35 dance
10	********	-0.51145	-0.37405	-0.42748	-0.65894	-0.05518	-0.62622	150		35 dance
11	********	-0.42748	0.038168	-0.08397	-0.66431	-0.06494	-0.60254	150		35 dance
12	********	-0.34351	-0.03053	-0.05344	-0.67236	-0.08716	-0.60132	150		35 dance
13	********	-0.44275	1.946565	-0.0687	-0.66846	-0.08105	-0.60132	150		35 dance
14	********	-0.41985	0.038168	-0.06107	-0.67847	-0.08447	-0.59717	150		35 dance
15	********	-0.45038	-0.0229	-0.03817	-0.677	-0.08105	-0.60547	150		35 dance
16	********	-0.48855	-0.03817	-0.05344	-0.67212	-0.08521	-0.6001	150		35 dance
17	********	-0.45802	0.045802	-0.0687	-0.68677	-0.09229	-0.59888	150		35 dance
18	********	-0.44275	-0.0229	-0.03053	-0.67236	-0.07886	-0.59424	150		35 dance
19	********	-0.41221	0	-0.10687	-0.68506	-0.09126	-0.60913	150		35 dance
20	********	-0.45802	0.038168	-0.03053	-0.67188	-0.08154	-0.60303	150		35 dance
21	********	-0.38931	-0.16794	0.038168	-0.67944	-0.09229	-0.60815	150		35 dance
22	********	-0.20611	1.946565	-0.37405	-0.87939	0.143066	-0.04272	150		35 dance
23	********	-0.33588	0.206107	-0.16031	-0.93872	0.086426	-0.0896	150		35 dance
24	********	-0.33588	-0.0229	-0.0916	-0.93652	0.080078	-0.07349	150		35 dance

Rasp zero 프로그램 실행

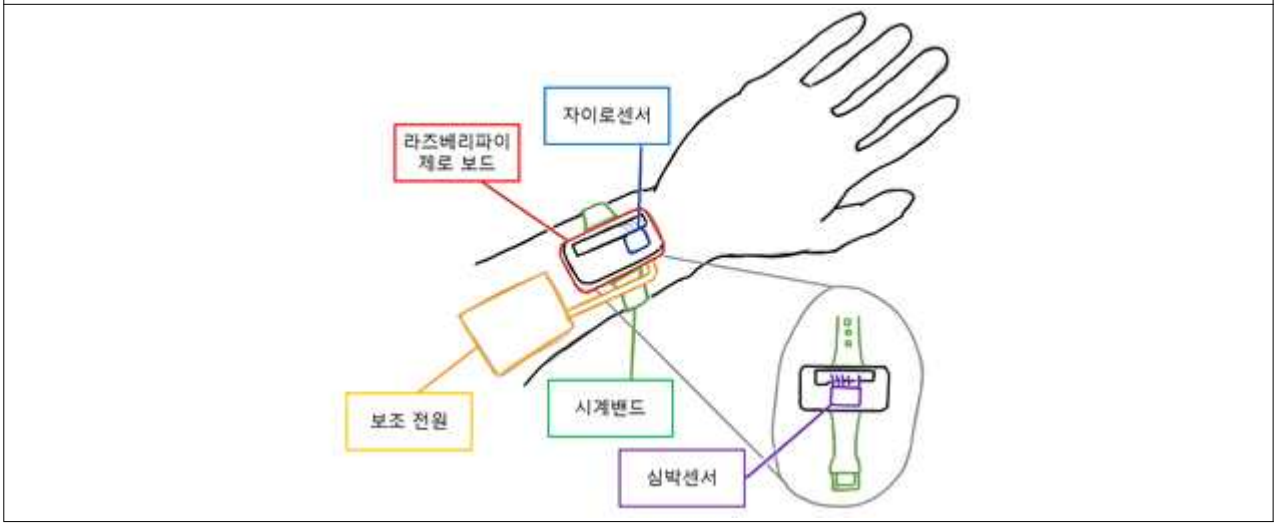


```
ghktks68@raspberrypi: ~/working
KeyboardInterrupt

ghktks68@raspberrypi:~/working $ python test2.py
receiving data... ('192.168.137.36', 44522)
^CTraceback (most recent call last):
  File "/home/ghktks68/working/test2.py", line 21, in <module>
    data = client_socket.recv(1024)
KeyboardInterrupt

ghktks68@raspberrypi:~/working $ python test3.py
receiving data... ('192.168.137.36', 57170)
Received from ('192.168.137.36', 57170) pjh
 Reading Data of Gyroscope and Accelerometer
Channel: 1, address: 87
Gx=-12.21 °/s   Gy=-13.57 °/s   Gz=-12.02 °/s   Ax=0.00 g      Ay=0.00 g      A
z=0.02 g
Gx=-0.34 °/s    Gy=0.09 °/s     Gz=-0.05 °/s    Ax=-0.02 g     Ay=-0.83 g     A
z=-0.70 g
 Reading Data of MAX301002
Gx=-0.19 °/s    Gy=0.01 °/s     Gz=-0.11 °/s    Ax=-0.01 g     Ay=-0.83 g     A
z=-0.69 g
Gx=-0.35 °/s    Gy=0.04 °/s     Gz=0.02 °/s     Ax=-0.02 g     Ay=-0.83 g     A
z=-0.69 g
```

PC 프로그램 실행

```
선택 Anaconda Prompt (anaconda3) - python  project_socketreceive.py
libifcoremd.dll     00007FFC92B0DF54  Unknown               Unknown  Unknown
KERNELBASE.dll      00007FFCD174AD63  Unknown               Unknown  Unknown
KERNEL32.DLL        00007FFCD1B47614  Unknown               Unknown  Unknown
ntdll.dll           00007FFCD3B426A1  Unknown               Unknown  Unknown

(base) C:₩Users₩User₩Desktop₩실행파일>python project_socketreceive.py
생체 정보 받기 위해 라즈베리파이에 연결 중.. ('192.168.137.247', 39948)
Received from ('192.168.137.247', 39948) 0.9770992366412213 -0.6259541984732825 -0.5648854961832062 -0.451171875 -0.7187
5 -0.615966796875 250 85 pjh
2022-12-19 13:31:27.801971: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-12-19 13:31:27.819511: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default
inter op setting: 2. Tune using inter_op_parallelism_threads for best performance.
model로드 완료

 박종혁 님에 대한 음악 추천

데이터 가공 완료

데이터 추가 후 예측 중..
1/1 [==============================] - 0s 435ms/step

------발라드 댄스 포크 힙합 랜덤 락에 대한 일치도-------
[[0.65718 0.00019 0.11487 0.03696 0.00001 0.00002 0.19076 0.      0.
  0.     ]]

결과는 ballad
```

웨어러블 디바이스 구성도

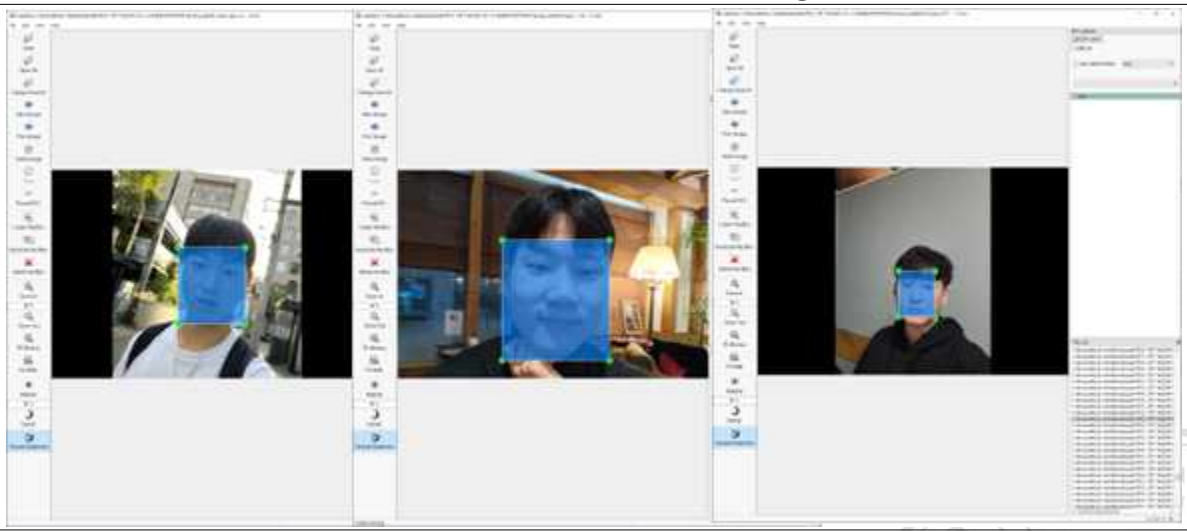


<Object Detection - TensorFlow Lite>

A. Source Code and Discussion

TensorFlow Lite Object Detection API 활용

1명 당 사진 1000장씩 총 3000장을 직접 Labeling 하여 dataset 생성



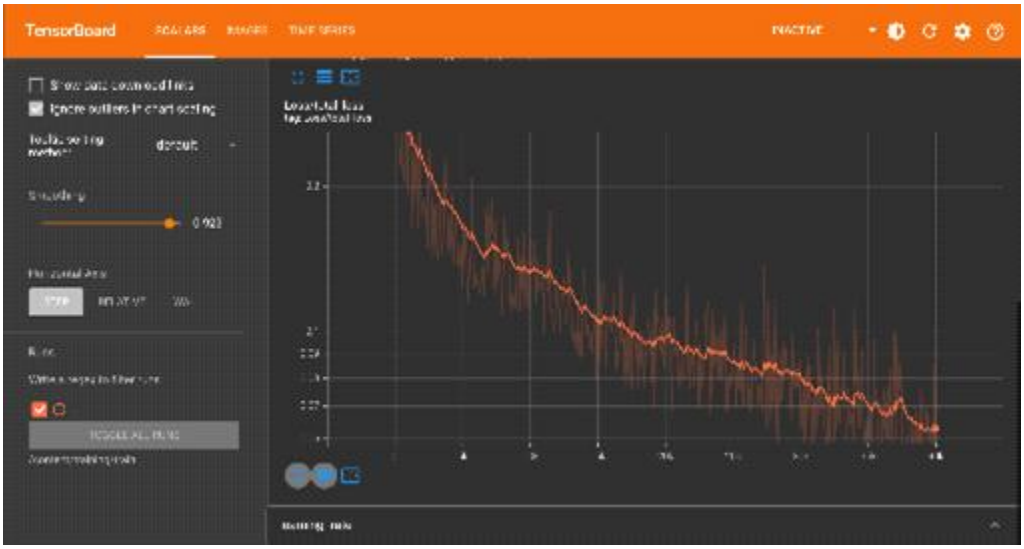
모델 : ssd-mobilenet-v2, step 횟수 : 8000, batch_size : 16

```
[ ] # Change the chosen_model variable to deploy different models available in the TF2 object detection zoo
    chosen_model = 'ssd-mobilenet-v2'

    MODELS_CONFIG = {
        'ssd-mobilenet-v2': {
            'model_name': 'ssd_mobilenet_v2_320x320_coco17_tpu-8',
            'base_pipeline_file': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.config',
            'pretrained_checkpoint': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz',
        },
        'efficientdet-d0': {
            'model_name': 'efficientdet_d0_coco17_tpu-32',
            'base_pipeline_file': 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config',
            'pretrained_checkpoint': 'efficientdet_d0_coco17_tpu-32.tar.gz',
        },
        'ssd-mobilenet-v2-fpnlite-320': {
            'model_name': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8',
            'base_pipeline_file': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config',
            'pretrained_checkpoint': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz',
        },
        # The centernet model isn't working as of 9/10/22
        #'centernet-mobilenet-v2': {
        #    'model_name': 'centernet_mobilenetv2fpn_512x512_coco17_od',
        #    'base_pipeline_file': 'pipeline.config',
        #    'pretrained_checkpoint': 'centernet_mobilenetv2fpn_512x512_coco17_od.tar.gz',
        #}
    }

    model_name = MODELS_CONFIG[chosen_model]['model_name']
    pretrained_checkpoint = MODELS_CONFIG[chosen_model]['pretrained_checkpoint']
    base_pipeline_file = MODELS_CONFIG[chosen_model]['base_pipeline_file']
```

```
[ ] # Set training parameters for the model
    num_steps = 8000

    if chosen_model == 'efficientdet-d0':
      batch_size = 4
    else:
      batch_size = 16
```

B. Data

학습하면서 Total Loss가 줄어드는 것을 확인



tflite 파일로 변환 후, raspberry pi 에서 실행
webcam으로 박종혁 학생을 두고 테스트 결과, 얼굴 인식도 99%

6. References

1) https://bentist.tistory.com/35

2) github에 코드를 정리해서 올려두었습니다.
https://github.com/KyleParkJong/embedded-system/tree/main/Final_Project