Pusan National University
Computer Science and Engineering
Technical Report 2024-10

IMU 기반 경량 재활 운동 자세 추론 및 보조 시스템 개발

201924603 하규승

201924446 김지훈

지도교수 백윤주

목 차

1. 서론 1

1.1. 연구 배경 1

1.2. 연구 목표 2

2. 연구 배경 2

2.1. 하드웨어 2

2.2. 통신 3

2.3. 기계학습 3

3. 연구 내용 4

3.1. IMU 데이터 수집 4

3.1.1. 유선으로 IMU 데이터 전송 4

3.1.2. 개발보드 변경 5

3.1.3. 배터리 도입 6

3.2. BLE 연결 8

3.2.1. BLE Characteristic 정의 8

3.2.2. 데이터 구조 정의 10

3.2.3. 데이터 동기화 구현 11

3.2.4. 게이트웨이 환경 이전 12

3.3. 데이터셋 수집, 자세 선정, 학습 13

3.3.1. 데이터셋 수집, 제작 13

3.3.2. 재활 운동 자세 선정 14

3.3.3. 학습 및 최적화 16

3.4. User Interface 구축 27

4. 연구 결과 분석 및 평가 30

5. 결론 및 향후 연구 방향 32

6. 참고 문헌 33

1. 서론

1.1. 연구 배경

인간은 나이를 먹으면서 점차 노화가 오고, 다양한 질병/질환을 겪게 된다. 노화로 인해 발생하는 대표적인 질환에는 오십견, 관절염과 같은 근골격계 질환이 있다. 이런 근골격계 질환은 지속적인 통증을 일으키면서 움직임에 제약을 걸어, 일상생활을 불편하게 만든다. 따라서 이를 빠르게 치료해 불편을 해소하는 것이 중요하다. 치료 방법에는 약물을 이용한 방법, 수술도 있으나 운동치료가 가장 많이 시행된다.

운동치료는 근골격계 손상으로 인해 움직임에 제한이 오거나 불편이 생기는 경우, 운동을 통해 긴장된 근육을 이완시켜 움직임이 부드러울 수 있도록 하는 재활 치료 방법이다. 운동 치료를 통해 근력을 강화하고 관절의 유연성과 가동 범위를 정상 수치에 가깝게 증가시킬 수 있으며 운동 장애를 개선할 수 있다.

운동치료는 환자의 (관절의) 움직임을 잘 파악해야 한다. 예를 들어 운동 치료 중 관절 가동성 운동은 가동 범위가 줄어든 관절을 지속적으로 움직이는 운동으로, 환자가 정해진 기준만큼 운동을 잘 수행하는지 파악하는 것이 중요하다. 이것은 보통 재활 치료사가 확인하는 부분이지만, 이 과정을 AI 와 같은 컴퓨팅 기술을 이용해 확인하는 연구가 수행되기도 하였다. Human Pose Estimation 도 그중 하나이다.

Human Pose Estimation(HPE)은 사람의 자세를 분석하고 예측하는 컴퓨터 비전의 한 분야로, 인체 사진을 분석하여 머리, 팔, 다리와 같은 관절의 위치와 회전 각도 등을 파악하고, 이를 통해 자세를 알아낸다. 운동 치료에 HPE 를 적용한다면, 카메라가 실시간으로 환자의 신체를 촬영하여 관절의 움직임을 파악하는 데 사용할 수 있다.

하지만 컴퓨터 비전을 이용해 HPE 를 수행하는 경우, 다음과 같은 문제점이 있다. 첫째로, 센서 장비가 매우 고가이다. 둘째로, 환자의 사진을 촬영해 처리하는 과정에서 개인 프라이버시가 침해될 수 있다. 따라서 비전과 달리 상대적 저가의 장비로 구성할 수

있고, 프라이버시가 침해될 우려가 없는 임베디드 환경에서 HPE 를 수행하고, 이를 운동 재활치료에 도입하면 이 문제를 해결할 수 있다.

1.2. 연구 목표

- BLE 를 통해 IMU 데이터를 게이트웨이로 전송하는 시스템 개발

  신체의 움직임을 파악하기 위해, 자세 측정이 필요한 부위에 측정 모듈을 장착한다. 모듈이 신체 움직임에 따른 IMU 데이터를 수집하고, 이를 게이트웨이에 전송할 수 있도록 한다.

- 전송받은 IMU 데이터들을 동기화(sync)하는 시스템 개발

  게이트웨이에서 여러 모듈을 통해 받은 IMU 데이터를 동기화하여 데이터셋 제작과 자세 추론에 이용할 수 있도록 한다.

- 재활 운동 자세 선정과 자세 추론을 위한 머신러닝 모델 학습

  많은 재활 운동 자세 중 자세 추론을 수행할 대표 자세와 학습 모델을 선정하고, 모듈을 신체에 부착하여 데이터셋을 수집해 이를 학습에 이용한다.

- 선정된 재활 동작 별 측정 모듈 개수의 최적화

  재활 운동 자세별로, 모델이 자세를 올바르게 추론할 수 있는 최적의 모듈 개수를 찾는다.

2. 연구 배경

2.1. 하드웨어

- ESP32는 Espressif systems 사에서 개발한 MCU 로, Wifi 와 Bluetooth 기능이 내장되어 있어 사물인터넷, 웨어러블 장치와 같이 인터넷 네트워크 연결이나 주변 장치들과의 무선 연결이 필요한 소형 기기에 자주 사용된다. 개발 언어로는 C 를 사용하고, 공식 개발 프레임워크인 ESP-IDF 를 이용하여 빌드, 플래시 등을 수행하게 된다. ESP32 내에도 다양한 제품군이 있고 제품군마다 사양이 조금씩 다른데, 이번 과제에서는 Bluetooth 5(BLE)가 지원되는 ESP32-C3, ESP32-S3를 사용하였다.

- IMU(Inertial Measurement Unit)는 관성 측정 장치로, 물체의 가속도나 회전 등을 측정할 수 있는 전자장비이다. IMU 는 보통 항공기나 우주선과 같이 물체의 움직임을 파악하는 것이 중요할 때 사용된다. 일반적으로 가속도를 측정할 수 있는 가속도계와 회전을 측정할 수 있는 자이로스코프 두 가지로 이루어져 있으며(이를 6축 IMU 센서라 함) 여기에 지자기 센서를 추가한 9축 IMU 센서도 존재한다.

- Jetson 시리즈는 NVIDIA 에서 개발한 임베디드 보드로, 임베디드 환경에서 AI 학습, 영상 처리 등을 수행할 수 있는 고성능의 임베디드 보드이다. 과제에서 사용할 Jetson Orin Nano 는 기존 Jetson 의 크기를 줄이고 경량화한 보드이다.

2.2. 통신

- BLE 란 Bluetooth Low Energy 의 약자로, 기존 블루투스의 전력 소모가 심하다는 단점을 해결하기 위해 Bluetooth 4.0부터 나온 통신 프로토콜이다. 전력 소모를 줄이면서도 기존 블루투스와 비슷한 성능을 내도록 설계되었다. 다만 저전력을 위해 설계되었기 때문에, 데이터 전송 속도가 다른 프로토콜에 비해 낮으며 통신 범위 또한 넓지 않다.

2.3. 기계학습

- SVM

  SVM(Support Vector Machine)은 지도 학습(Supervised learning)의 한 방법으로, N 차원 특성의 데이터를 잘 나눌 수 있는 가장 optimal 한 Hyperplane 을 찾는 분류 알고리즘이다. SVM 은 데이터를 나누는 Hyperplane 과 이것에 가장 가까운 데이터 샘플과의 거리인 Margin 을 최대화하는 방향으로 Hyperplane 을 결정한다. 또한 커널을 이용해 저차원의 비선형 데이터를 고차원으로 재배치시켜 분류하기 복잡한 데이터도 선형 분류를 가능하게 한다.

- RNN, LSTM

  RNN(Recurrent Neural Network)은 음성이나 문자열과 같이 순서가 있고 상호 독립적이지 않은 데이터(순차 데이터)를 학습하는 데 사용하는 딥러닝 모델이다. RNN 모델은 일반적인 feed-forward network 에서의 hidden unit 의 입력에 현재 시간(time step)의 입력뿐만 아니라 이전 시간에서의 hidden unit 의 출력까지 추가하여 이전의 정보 또한 학습하도록 하였다.

  하지만 단순 RNN 의 경우 순차 데이터가 길어질수록 초기에 입력된 정보가 사라지는 장기 의존성 문제를 가지고 있어, 긴 순차 데이터에 대해서는 정확도가 떨어진다는 단점이 있다.

  LSTM(Long Short Term Memory)을 이용하면 기존 RNN 이 가지고 있는 장기 의존성 문제를 방지할 수 있다. LSTM 은 이전의 정보를 기억할 수 있는 메모리 셀로 이루어져 있고, 상태 정보를 변경하는 3개의 게이트를 가지고 있다. 게이트는 각각 메모리 셀의 무한 성장을 막기 위한 삭제 게이트, Cell state 를 업데이트하기 위한 입력 게이트, 출력을 업데이트하기 위한 출력 게이트로 이루어져 있다.

- PCA

  PCA(Principal Component Analysis;주성분분석)는 차원 축소 기법 중 하나이다. 기존의 고차원 데이터에서 데이터를 잘 나타낼 수 있는 특성들을 뽑아 새로운 저차원의 데이터를 구성한다. 이번 과제에서는 IMU 데이터들을 2차원으로 시각화하기 위해서만 사용하였다.

3. 연구 내용

3.1. IMU 데이터 수집

3.1.1. 유선으로 IMU 데이터 전송

처음에 BLE 통신을 구현하고 IMU 센서로부터 데이터를 받아올 MCU 로 Nordic Semiconductor 사의 nRF52832 모듈을 이용할 계획이었으나, 개발 보드를 구하기 어렵게 되어 MCU 로 ESP32-C3 기반의 Geekble Mini 보드와 IMU 로 가속도, 자이로, 지자기 센서가 포함된 9축 센서(MPU-9250)를 사용하여 과제를 시작하였다. 먼저 브레드보드에

MCU 와 IMU 를 연결한 뒤, I2C 를 통해 MCU 가 센서값을 읽어올 수 있도록 하였다. 다음으로 시리얼 통신을 통해 유선으로 IMU 데이터를 PC 에 넘겨, IMU 데이터값이 정상적으로 들어오는지 확인하였다.

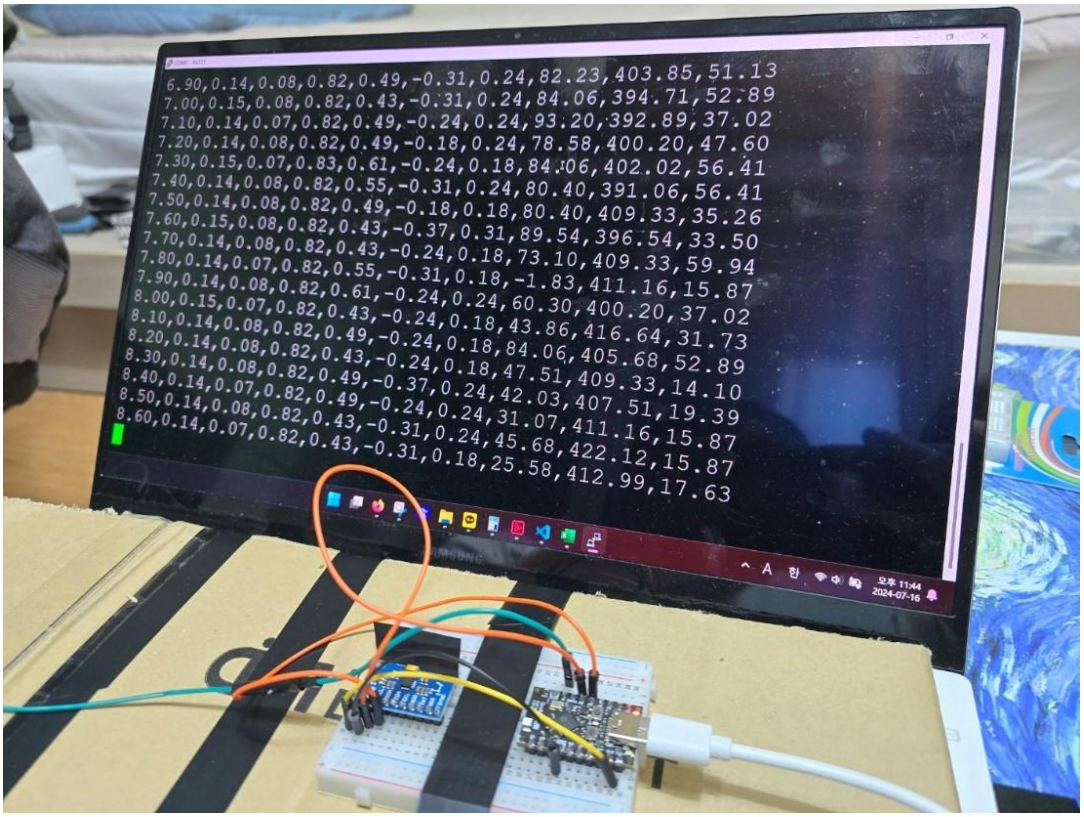

그림 1 MCU 와 IMU 를 연결하고 시리얼 통신으로 값을 전송

3.1.2. 개발보드 변경

이후 연구실에서 Gamba labs 의 개발 보드를 10개 제공받아 과제를 진행하게 되었다. ESP32-S3 기반 개발 보드이고, 기판에 버튼, LED 와 IMU 센서인 ICM-42670-P 가 내장되어 있어 추가 센서나 브레드보드 없이 보드만으로 IMU 데이터 수집과 BLE 통신이 가능하게 되었다. IMU 센서의 경우 가속도와 자이로를 측정할 수 있는 6축 센서로, 보드와는 I2C 를 통해 연결되어 있다.

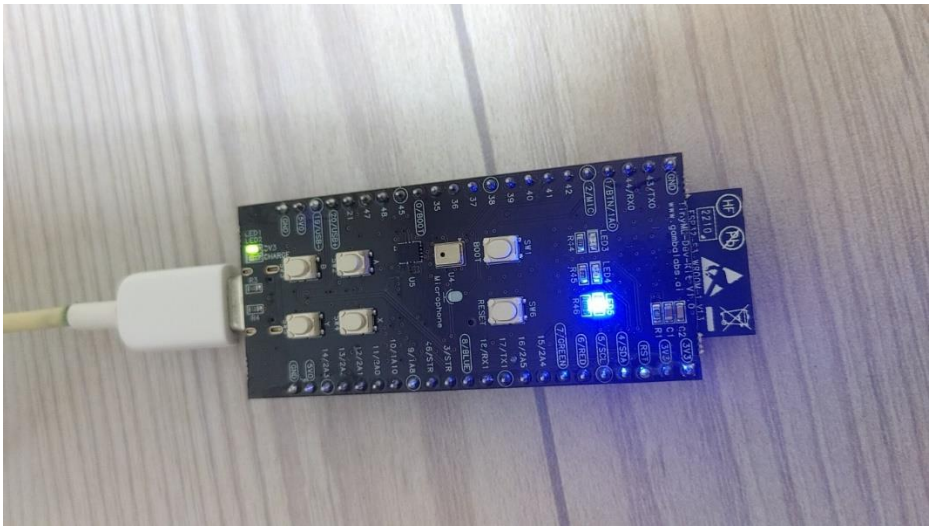

그림 2 ESP32-S3 기반 개발 보드

IMU 데이터를 가져오기 위해, 데이터시트를 참고하여 필요한 절차를 살펴보았다. 먼저 센싱을 시작하기 위해 파워 설정과 관련된 레지스터 PWR_MGMT0에 접근해 가속도계와 자이로스코프를 LN(Low Noise) 모드로 설정하여 두 측정계를 활성화시킨 뒤, 가속도와 자이로 감지 데이터가 담겨있는 레지스터에 접근해 값을 가져왔다.

3.1.3. 배터리 도입

몸을 자유롭게 움직이면서 IMU 데이터를 얻기 위해, 무선 통신으로 데이터를 수신해야 하며 또한 보드의 전원도 무선으로 연결될 필요가 있었다. 따라서 학과 졸업과제 제작 물품 지원을 통해 1,000mA 리튬 폴리머 배터리를 구매하였다. 보드에 딱 맞는 크기로 주문하여 핀 사이에 들어갈 수 있었고, 이를 절연테이프를 이용해 감아 보드에 떨어지지 않도록 부착하였다. 배터리 충전의 경우, 보드 내에 충전 회로가 내장되어 있었기에 따로 구매하여 장착할 필요는 없었다.

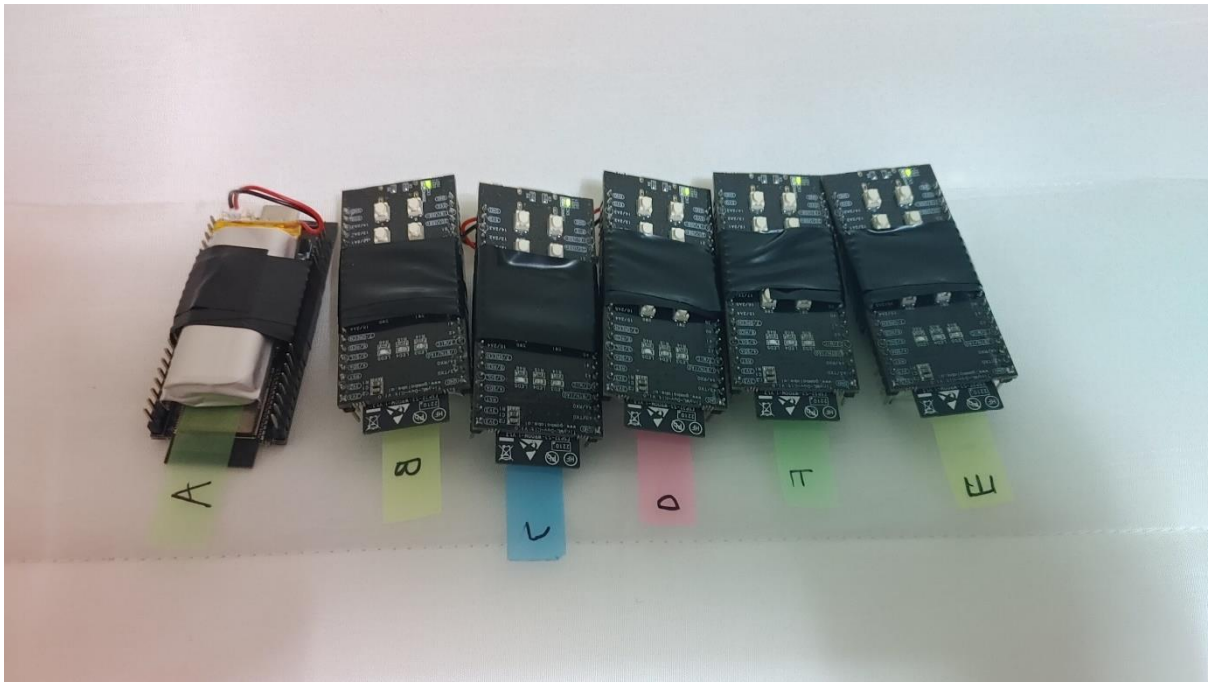

그림 3 절연테이프로 배터리를 부착한 개발 보드들

또한 배터리를 장착하여 보드를 사용하는 경우, 가만히 두어도 BLE 통신과 IMU 센서의 센싱, LED 의 사용 등으로 인해 전력을 많이 사용하게 된다. 따라서 일정 시간이 지나도 보드를 사용하고 있지 않으면, 스스로 Deep sleep mode 에 들어가게 하였다. ESP32는 다양한 절전 모드를 지원하는데, deep sleep 모드의 경우 CPU, RAM 과 같은 Core, 주변기기와 WiFi, Bluetooth 기능이 꺼지고 RTC 관련 장치만 켜져 있어 10μA 의 매우 적은 전류만 소비하게 된다. 다만 이 상태에 빠지면 다시 원격으로 보드를 제어할 수는 없어, 보드의 리셋 버튼을 눌러 깨워야 한다.

그리고 보드를 신체에 부착해 IMU 데이터를 수신해야 하므로, 몸에 잘 붙일 수 있는 도구가 필요했다. 이것은 아래와 같이 케이블타이용 탄성 벨크로 밴드를 몸에 감고 밴드 사이에 센서 보드를 고정하는 것으로 해결하였다.

그림 4 센서를 신체에 부착한 모습

3.2. BLE 연결

3.2.1. BLE Characteristic 정의

이제 MCU 에서 게이트웨이에서 얻은 IMU 데이터를 무선으로 송신하기 위해, BLE 통신을 보드에 구현하였다. 게이트웨이와 보드가 데이터를 교환할 Service 를 생성한 뒤, 게이트웨이로 IMU 데이터를 전송할 Read Characteristic 과 게이트웨이에서 보드를 제어할 명령을 보낼 Write Characteristic 을 정의하였다. 보드에 처음 전원이 들어오면 보드의 청색 LED 가 점멸되면서 Advertising 모드를 시작해 다른 장치가 검색할 수 있도록 하다가, 게이트웨이와의 연결이 수립되면 LED 의 점멸을 멈추고 게이트웨이로 IMU 데이터를 송신하게 된다. 이후 연결이 끊어지면, 보드는 다시 Advertising 모드를 시작하게 된다.

다음으로 게이트웨이에서 여러 개의 보드와 연결하여 IMU 데이터를 받아오기 위한 프로그램을 파이썬으로 작성하였다. 게이트웨이는 일단 PC 를 기준으로 작업하였고,

이후에 Jetson Orin Nano 로 환경을 옮겼다. 파이썬에서 BLE 통신을 가능하게 하는 bleak 모듈과 비동기적 작업을 위한 asyncio 모듈을 임포트하여 사용하였고, 사용할 수 있는 장치를 전부 스캔하는 보드 스캔 기능, 장치 목록을 출력하는 기능, IMU 데이터를 수신하여 학습용 데이터셋으로 만드는 기능을 구현하였다.

하지만 보드로부터 데이터를 수신하는 과정에서 한 가지 문제가 발행하였다. Read characteristic 을 이용하여 보드로부터 데이터를 수신하려면 먼저 클라이언트(게이트웨이)에서 서버(보드)에게 읽기 요청을 보낸 뒤, 보드가 클라이언트에게 데이터를 송신하게 된다. 이 과정이 생각보다 오래 걸리고 latency 가 일정하지 않게 측정되었다. 보드에서 10Hz, 즉 100ms 마다 게이트웨이로 데이터를 보내려고 하였을 때 게이트웨이에서 read 요청을 하고 읽어오는 데 걸리는 시간을 측정해 보았는데, 30~70ms 정도의 시간 지연이 있다가 어느 순간에는 100ms 를 훌쩍 넘기게 되는 경우도 발생하였다.

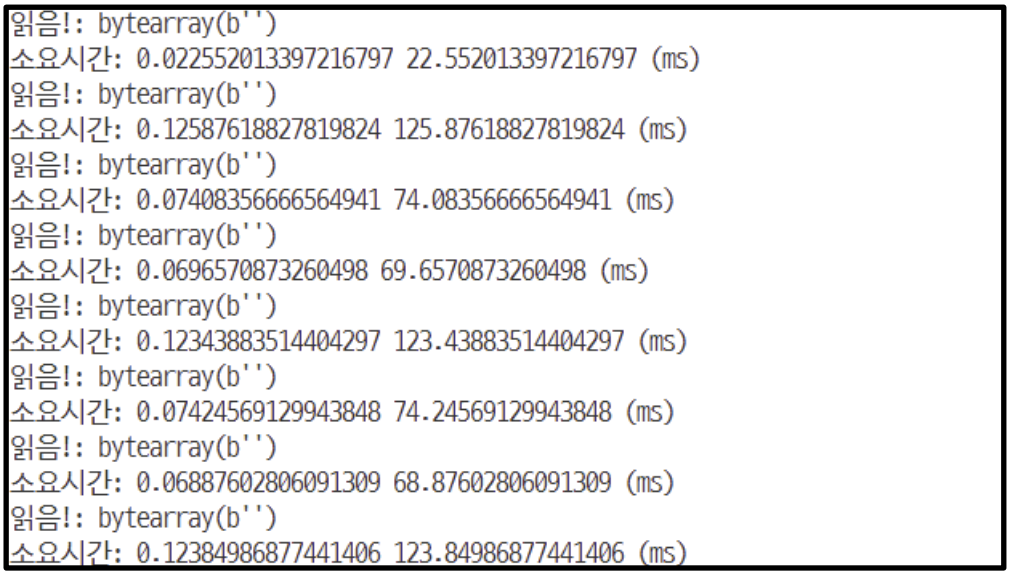

그림 5 Read Characteristic 에 실제 전송하는 데이터가 없는데도, 지연 시간이 큼.

데이터를 여러 개 묶어서 송신하는 방안도 고려해 보았다. 그러나 이 같은 경우에, 한 번에 보내는 데이터의 양도 덩달아 많아져서 지연 시간이 같이 올라가 해결 방안이 되지는 못하였다.

따라서 데이터를 게이트웨이에서 Read 하는 방식으로 가져오는 대신, Notify 방식으로 데이터를 가져오기로 하였다. BLE 통신에서 Notify 를 이용하게 되는 경우, 클라이언트(게이트웨이)가 서버(보드)에게 데이터를 수신하겠다는 Subscribe 요청을 처음 보내기만 하면 서버는 클라이언트의 추가적인 선행 요청(read) 없이 클라이언트에게 데이터를 보낼 수 있고, 데이터가 잘 받아졌는지 확인하는 작업 없이 다음 데이터를 송신할 수 있다.

Notify 를 이용해 데이터를 가져오기 위해 데이터 교환 Service 에 notify characteristic 을 추가하여 보드가 이 characteristic 으로 IMU 데이터를 notify 하도록 하였고, notify 가 발생할 경우 게이트웨이는 이 이벤트에 연결된 callback 함수를 통해 데이터를 처리하도록 하였다. 이렇게 하니 지연 없이 거의 바로 데이터를 얻을 수 있게 되었다.

3.2.2. 데이터 구조 정의

보드가 notify 를 통해 게이트웨이로 데이터를 보낼 때, 단순히 6개 IMU 데이터를 보내지 않고, 자신이 어떤 보드인지에 대한 정보(id)와 측정 시작 후 흐른 시간 정보(timestamp)도 같이 보내게 하여 게이트웨이에서 데이터의 sync 를 맞추고 하나로 통합하는 데 도움이 되도록 하였다. 즉 보드가 송신하는 데이터는 id, timestamp, data 의 세 가지인데, id 는 uint8_t 타입(부호 없는 8비트 정수;char), timestamp 는 uint32_t 타입(ms 단위), data 는 6개의 IMU 데이터를 가지는 float[6] 타입이다.

```
struct imu_data_line{
    uint8_t id;
    uint32_t timestamp;
    float data[6];
} line;
```

그림 6 게이트웨이에게 보내는 데이터의 구조

게이트웨이에서 보드로 보내는 제어 명령의 경우 Write characteristic 을 이용하여 보내게 하였다. 16비트의 부호 없는 정수 둘을 write 로 보내는데, 각각 명령의 종류를 구별하는

type 변수와 값을 저장하는 value 변수로 이루어져 있다. 명령은 다음과 같이 세 가지로 이루어져 있다.

1. 보드의 timestamp 를 0으로 초기화

   연결된 보드의 timestamp 를 초기화하여 sync 를 맞추는 데 사용한다.

2. 보드의 sampling rate 를 전달한 value 값으로 초기화

   sampling rate 를 낮추어 자세 추론이 필요할 때 사용한다.

3. 보드를 절전 모드로 변경

   보드를 절전모드(Deep sleep)로 변경하여 배터리 사용량을 줄인다.

```
typedef struct command_from_gateway{
    uint16_t type;
    uint16_t value;
} command;
```

그림 7 보드로 보내는 명령 데이터의 구조

3.2.3. 데이터 동기화 구현

학습용 데이터셋을 만들기 위해선, 게이트웨이에서 다수의 보드로부터 얻은 데이터를 같은 시간대에 묶고 정렬하는 작업이 필요하다. 이를 위해, 측정 시작과 동시에 모든 보드의 timestamp 를 0으로 만들도록 보드에 명령을 보낸다. 그리고 수신된 데이터로부터 id 와 timestamp 정보를 얻은 뒤, 동일한 timestamp 마다 보드로부터 얻은 IMU 데이터가 id 순서대로 정렬되게 하였다. 이때 여러 번 callback 함수가 호출되게 되는데, 동시에 데이터 변수에 접근하여 값의 Consistency 가 무너지는 것을 방지하기 위해, asyncio 모듈의 Lock 을 이용하여 Critical Section 을 보호하였다. 마지막으로 정해진 시간 동안 모든 데이터를 성공적으로 얻으면, 데이터 1행에 특성 이름(Ex. Agx = A 센서의 x 축 gyro 값)을 삽입하고 csv 형태의 파일로 저장하게 하였다.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ms	Aax	Aay	Aaz	Agx	Agy	Agz	Bax	Bay	Baz	Bgx	Bgy	Bgz
2	0	-0.89404	0.20459	0.415039	-0.18293	1.036585	-0.4878	-0.77637	-0.22949	0.590332	-0.60976	0.060976	0
3	50	-0.89453	0.207031	0.41748	-0.42683	1.158537	-0.42683	-0.77881	-0.22803	0.594238	-0.79268	0.182927	0.060976
4	100	-0.89355	0.207031	0.417969	-0.30488	1.219512	-0.36585	-0.77734	-0.22754	0.591797	-0.60976	0.304878	0.121951
5	150	-0.89404	0.206543	0.415527	-0.30488	1.463415	-0.73171	-0.77832	-0.22852	0.591309	-0.54878	0.304878	0.060976
6	200	-0.89258	0.206543	0.416992	-0.54878	1.463415	-0.60976	-0.77832	-0.23096	0.59082	-0.67073	0.121951	0.060976
7	250	-0.89648	0.205078	0.416504	-0.4878	1.341463	-0.73171	-0.7793	-0.23047	0.592773	-0.60976	0.121951	0
8	300	-0.89355	0.20752	0.416992	-0.30488	1.097561	-0.42683	-0.77637	-0.22998	0.592773	-0.73171	0.060976	0
9	350	-0.89404	0.204102	0.416992	-0.42683	1.036585	-0.36585	-0.77637	-0.23096	0.59082	-0.67073	0.060976	0
10	400	-0.89404	0.206543	0.417969	-0.36585	1.280488	-0.54878	-0.77686	-0.22852	0.594727	-0.73171	0.182927	0.121951
11	450	-0.89258	0.207031	0.416992	-0.36585	1.219512	-0.54878	-0.77686	-0.229	0.594238	-0.67073	0.182927	0.182927
12	500	-0.89551	0.20459	0.414551	-0.30488	1.280488	-0.54878	-0.77734	-0.22949	0.59082	-0.54878	0.121951	0.365854
13	550	-0.89746	0.20459	0.416016	-0.30488	1.158537	-0.30488	-0.78027	-0.22949	0.593262	-0.67073	-0.06098	0.060976
14	600	-0.89258	0.206543	0.414551	-0.4878	1.158537	-0.36585	-0.77637	-0.23047	0.586914	-0.67073	0.121951	0.182927
15	650	-0.89453	0.203613	0.416016	-0.30488	1.158537	-0.42683	-0.77832	-0.22852	0.59375	-0.54878	0.243902	0.182927
16	700	-0.89795	0.206543	0.416992	-0.30488	1.036585	-0.42683	-0.78076	-0.23291	0.594238	-0.67073	0.060976	0.121951
17	750	-0.89404	0.203613	0.416992	-0.60976	0.914634	-0.4878	-0.77344	-0.22852	0.59082	-0.42683	0.121951	0.121951

그림 8 완성된 데이터셋의 일부분 캡쳐

3.2.4. 게이트웨이 환경 이전

지금까지 PC(Windows 11 환경)를 게이트웨이로 두어 10개 센서들과 통신하고 데이터를 수신하였는데, 이제 이를 Jetson Orin Nano(Ubuntu 기반)로 옮겼다. 그리고 VNC 를 이용해 PC 에서 원격으로 접속하여 사용하였다.

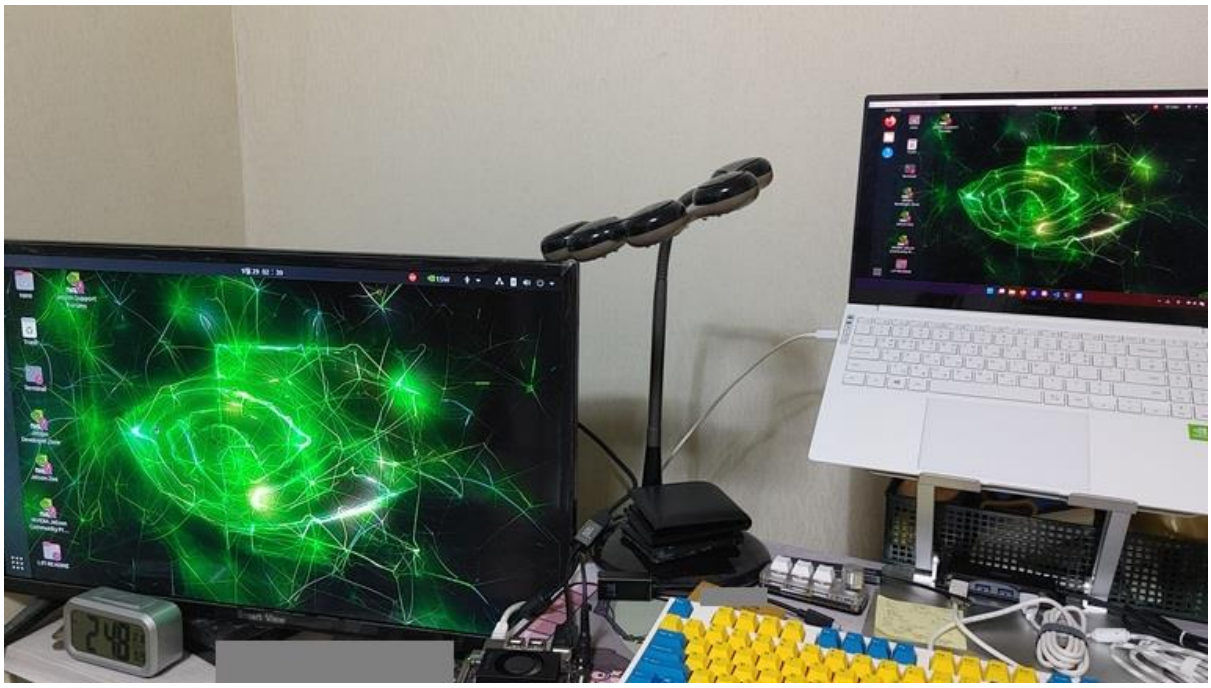

그림 9 젯슨 오린 나노와 VNC 로 연결

하지만 Jetson 으로 환경을 옮기니 예기치 못한 문제가 생기게 되었다. 기존의 Windows 노트북 환경에서 10개의 보드와 동시 연결을 수행할 때에는 아무런 문제가 없었으나, Jetson 에서 연결할 땐 8개를 넘어가는 보드와 연결을 수행할 때 소프트웨어 오류가 나며 연결이 중단되었다. 관련 내용을 찾아본 결과, BLE 의 경우 이론적으로는 많은 수의 장치와 연결이 가능하지만 대역폭 제한 등과 같은 현실적인 문제로 실제 사용 시에는 많은 수의 장치를 연결할 수 없기 때문에 동시 가능 연결 수가 하드코딩 되어 있다는 사실을 알게 되었다.

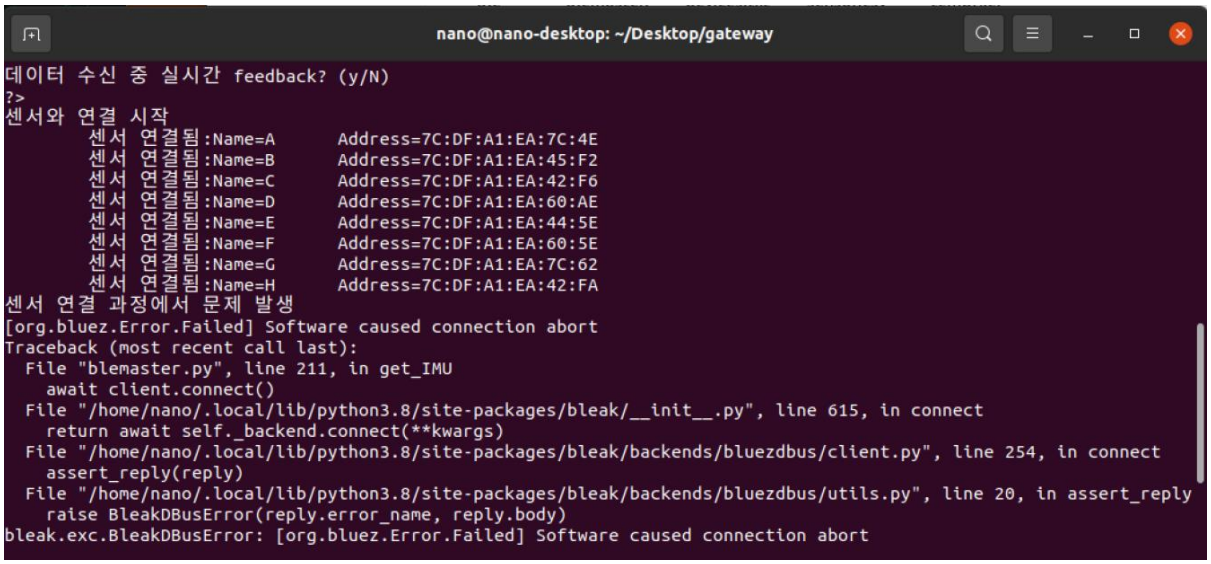

그림 10 8개 넘어가는 센서가 연결되려 하자 connection abort 가 발생

따라서 10개 센서를 연결하여 데이터를 수집할 필요가 있는 경우에만 기존 PC 에서 데이터를 수집하도록 하고, 최적화 작업 이후에는 동시에 연결할 센서의 수가 9개보다 줄어들 것이므로 이후 자세 추론은 Jetson nano 에서 수행하기로 결정하였다.

3.3. 데이터셋 수집, 자세 선정, 학습

3.3.1. 데이터셋 수집, 제작

먼저, 학습에 사용할 수 있는 공개 Dataset 이 있는지 Rehabilitation exercise, machine learning dataset 의 2가지 키워드를 활용하여 찾아보았다. 그렇게 검색하여 QMAR[1], UI-PRMD[2]과 같은 재활 운동 관련 공개 dataset 을 찾을 수 있었다. 하지만, 이런 대부분의 dataset 들은 Computer vision 을 통한 HPE 에 국한되어 있어 이번 과제에 사용하기

어려웠다. 또한 IMU 를 이용해 자세를 추론하는 공개 dataset 을 확인해 보았으나, 센서 부착 위치가 데이터셋마다 천차만별에다 대부분 전신 추론을 위하여 구성되었고, 특정 재활 운동에 대해 자세를 추론하는 다른 연구도 살펴보았으나 이 경우 연구를 위해 직접 데이터셋을 수집했다는 것을 확인할 수 있었다. 따라서 이번 과제에서는 재활운동 동작마다 신체에 부착된 IMU 센서로부터 raw 데이터를 받아 이것을 가지고 직접 데이터셋을 제작하여 학습에 이용하기로 하였다.

3.3.2. 재활 운동 자세 선정

데이터셋 수집 이전에 어떤 재활 운동을 가지고 무엇을 할지 결정해야 했다. 따라서 재활 운동을 수행할 신체 부위를 상지, 하지, 경추, 요추의 4가지 카테고리로 나누고, 카테고리마다 재활 운동을 다음과 같이 선정하였다.

- 상지: 팔, 어깨 부위
  - Assisted shoulder flexion, Stretching wrist flexors
- 하지 : 다리
  - Hamstring stretch, Sit to stand
- 경추 : 목
  - Neck side extension
- 요추 : 허리
  - Bridge stretch, 누워서 한쪽 다리 넘기기

7개가량의 재활 운동을 선정하였는데, 이후 자문을 통해 이 정도의 재활 운동 개수도 데이터 관점에서 많다는 것을 자문받았고 따라서 4개 카테고리별로 하나의 재활 운동만 선정하였다. (밑줄 친 운동)

또한 선정된 재활 운동으로 어떤 추론을 할지 의논한 결과, 운동 자세의 성질에 따라 추론을 아래와 같은 2가지의 방법으로 나누어 하기로 하였다.

1. 사용자가 지금 목표 자세를 취하고 있는지 판단
2. 사용자가 정해진 시간 동안 (반복적인) 운동 자세를 잘 수행하는지 판단

예를 들어, 상지의 Assisted shoulder flexion 은 양팔을 들어 올리고 내리는 동작, 하지의 Hamstring stretch 는 한쪽 다리를 들었다 내리는 동작으로 구성되어 있다. 이런 운동의 경우, 정해진 시간(10초) 동안 운동을 잘 수행하였는지 여부를 추론하기로 하였다. 또한 정해진 시간 동안 독립적이지 않은 연속적인 데이터(sequence)를 학습하는 것이므로 RNN 을 사용하기로 하였다.

반면 경추의 Neck side extension 은 한쪽 손으로 머리를 옆으로 잡아당기고, 요추의 Bridge stretch 는 누워서 브릿지 자세(엉덩이를 들어 올림)를 취하는 운동인데 이 경우 해당 자세를 취하고만 있는지를 추론하기로 하였다. 이 경우엔 모든 가능한 자세를 포함하는 일종의 "자세 공간"에서 특정 동작을 분류하면 되기 때문에 학습 모델로 SVM 을 이용하기로 하였다.

이제 앞서 설명한 펌웨어(IMU 센서와 게이트웨이)를 통해, 데이터 수집을 시작했다. 체형이 다른 조원 2명에 대해 각 운동 자세마다 바른 자세와 잘못된 자세를 5분씩 3~4번, 운동 자세마다 총 40분가량의 데이터를 수집하였고, 운동 자세를 취하지 않고 자유롭게 움직이는 데이터 또한 수집하였다. 데이터의 샘플링 속도는 20Hz 로 설정하였다. (50ms 마다 IMU 데이터를 수집하고 게이트웨이로 보내게 된다)

이때 처음에는 데이터셋 수집 시 자세마다 움직이는 관절 부위에만 센서를 착용하여 데이터를 수집하였으나, 이후 data 의 범주가 좁아진다는 지적을 수용하여 최종적으로 사용할 수 있는 센서 10개를 모두 착용한 상태로 새롭게 IMU 데이터를 수집하게 되었다. 아래 그림을 기준으로 센서를 착용하였다.

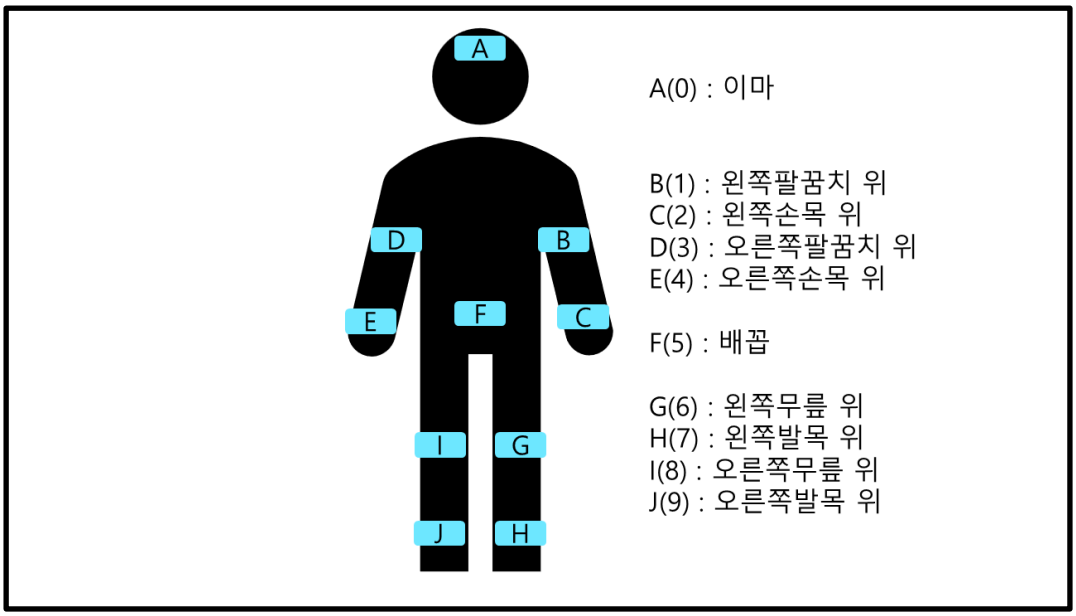

그림 11 센서 부착 위치

마지막으로 수집한 데이터를 학습에 사용하기 위해 데이터 라벨링을 수행하여 올바른 동작 데이터와 잘못된 동작 데이터를 구별하였다.

3.3.3. 학습 및 최적화

처음 연구실에서 개발 보드를 받지 않고 ESP32-C3 보드와 9축 IMU 로 과제를 진행할 때, 수집한 IMU 값을 학습에 이용하여 자세 추론에 사용할 수 있는지 확인해 보고자 관절 각도를 추론하는 실험을 수행하였다. 먼저 박스를 자르고 접어 팔꿈치처럼 동작하는 모형을 만든 뒤에, 관절 상/하 두 부위에 MCU-IMU 세트를 부착하였다. 두 MCU 와 PC 를 연결한 뒤, 4분 동안 모형의 관절 각도를 45°, 90°, 135°, 180° 씩 변화시키며 센싱 값을 기록하였고, 센싱을 마친 후 학습을 위해 데이터에 시간별 각도 데이터를 추가로 기재하였다.

그림 12 팔꿈치 박스 모형

두 개의 9축 IMU 를 사용하였고, 10Hz 의 Sampling rate 로 4분간 측정을 수행하였으므로 총 데이터셋의 개수는 2,400개, 특성 수는 18개이다. Scikit-learn 의 PolynomialFeatures 를 사용하여 데이터에 다항 특성(2차)을 추가하는 전처리 작업을 거친 뒤, Ridge Regression 을 통해 데이터셋을 학습시켰다.

모델 학습을 마친 후, 팔꿈치 모형을 직접 움직이면서 실시간으로 관절 각도를 아래 사진과 같이 추론해 보고 실제 각도와 비교해 보았다.

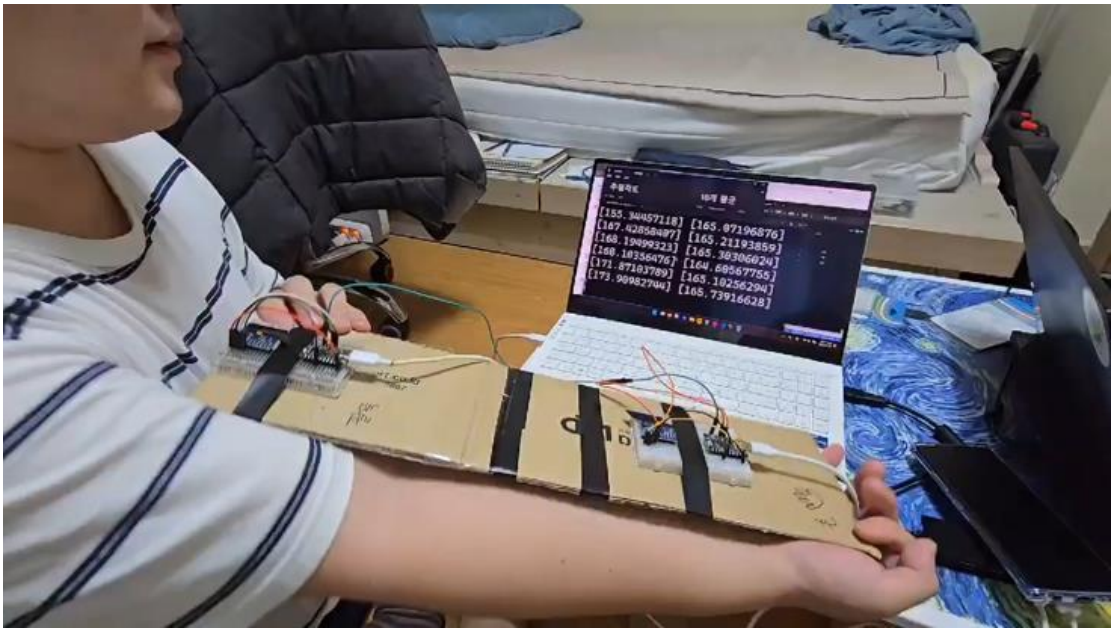

그림 13 팔꿈치 모형 각도 추론 결과 비교

각도 추론은 어느 정도 잘 이루어졌으나, 학습 모델이 선형이고 학습에 사용한 데이터셋이 적다는 한계로 인해 정밀한 추론은 이루어지지 않았다. 실제로도 학습 시에 IMU 센서의 지자기 데이터 특성에 대한 weight 가 컸는지, Yaw 축(바닥 면과 수직인 축)으로 팔꿈치 모형을 회전하니 추론 각도가 20도가량 편차가 생기는 것을 확인할 수 있었으며 전선과 같이 주변의 전도체에 민감한 지자기 센서의 특성으로 인해 Outlier 데이터가 발생하여 각도 추론에 영향을 주기도 하였다.

이제 다시 6축 IMU 가 내장된 ESP32-S3기반 개발 보드를 이용하여 학습을 수행하였다. 먼저, Neck side extension 과 Bridge stretch 자세에 대해 10개의 모든 센서를 착용하여 수집된 데이터셋에 대해서, 분류하기는 좋은지, 크게 튀는 outlier 는 없는지 확인하고자 하였다. 그러나 6축 IMU 센서가 10개 사용되었으므로 총 60차원의 데이터를 확인해야 하는데 이를 시각적으로 나타낼 방법이 없으므로, 2차원 공간에서 시각화하기 위해 PCA 를 통해 데이터 특성을 두 개로 줄이고 matplotlib 의 plot 기능을 이용하여 산점도를 그려 데이터 분포를 확인하였다. (가로축, 세로축은 각각 제1성분, 제2성분을 의미하며 파란색 x 마크가 옳은 자세, 빨간색 x 마크가 올바르지 않은 자세를 의미한다)

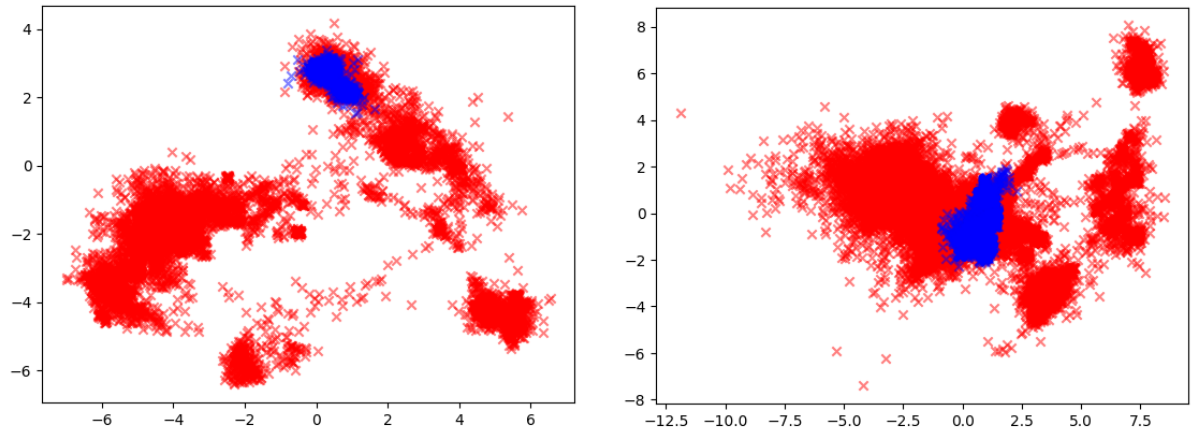

그림 14 Bridge stretch / Neck side extension 에 대한 데이터 시각화 결과

Correct / Wrong data 끼리 어느 정도 잘 구분되어 있는 것을 확인하였기 때문에, SVM 을 이용하여 자세 Classification 을 시도했다. 먼저 Dataset 의 30%를 test set 으로 분리하여 모델 평가에 사용하기로 하였다. 그리고 scikit-learn 의 SVC 모델을 불러온 뒤, 커널로는 가우시안 커널(rbf 커널)을 사용하였다. 규제 정도를 나타내는 C 값의 경우, 적절한 값을 선정하기 위해 아래와 같이 값을 0.05부터 5000까지 배로 증가시키며 학습 결과를 확인하였다.

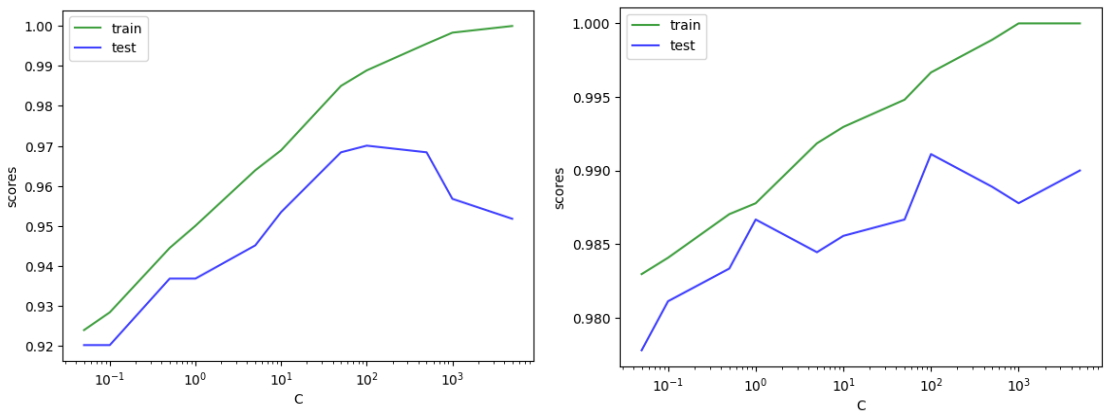

그림 15 Bridge stretch(좌), Neck side extension(우)에서 C 에 따른 train / test score 변화

규제가 커질수록 train set 에 대해 과적합(overfitting)이 일어나고 있음을 그래프에서 확인할 수 있다. train set 에 대한 점수가 높으면서도 test set 에 대한 점수가 더 떨어지지 않는 지점의 C 값을 선정하였다.

마지막으로 센서 개수를 줄이는 최적화 작업을 진행하였다. 즉 센서 개수를 줄이면서도 추론 정확도를 많이 낮추지 않는 최적의 센서 개수와 위치를 찾는 것이 필요하다. 이를 위한 한 가지 방법으로는 10개 센서의 부분집합(subset)을 모두 구해 부분집합마다 모델을 훈련시키고 그 결과를 비교하여 최적의 조합을 찾는 방법이 있다. 그러나 이 방식은 모든 부분집합의 개수인 $2^{10} = 1024$ 개의 조합을 확인해 봐야 해 매우 오랜 시간이 소요되게 된다.

따라서 이 과제에서는 Backward elimination 기술을 응용하였다. 모든 센서 조합에서 시작하여, 현재 센서 조합에서 센서를 하나씩 빼 보았을 때, 모델의 정확도가 가장 떨어지지 않는(혹은 loss 가 가장 오르지 않는) 센서를 제거한 후 다시 이 과정을 센서가 하나도 남아 있지 않을 때까지 반복하는 것이다. 그리고 과정 중간중간 모델의 학습 결과를 따로 저장해 적절한 성능을 보이면서도 필요한 센서 수가 적은 조합을 고르면 된다. 다만 이 방법은 한번 없어진 센서를 다시 가져오지 않으며, 모든 경우를 고려하지 않기 때문에 optimal solution 을 제공하지는 않는다.

또한 최적의 센서 조합을 구할 때, 총 10개의 센서 중에서 자세별로 전혀 관계없는 센서를 먼저 제외한 뒤 센서에 대한 Backward elimination 을 수행할 수 있다. 따라서 Neck side extension 의 경우 머리/상체의 움직임과는 상관 없는 하지 쪽의 센서(6,7,8,9번)을 고려하지 않았다.

2가지의 운동에서 센서를 하나씩 제거해가면서 best subset 을 기록하였고 그때의 모델 학습 결과를 그래프로 나타낸 결과는 아래와 같다.

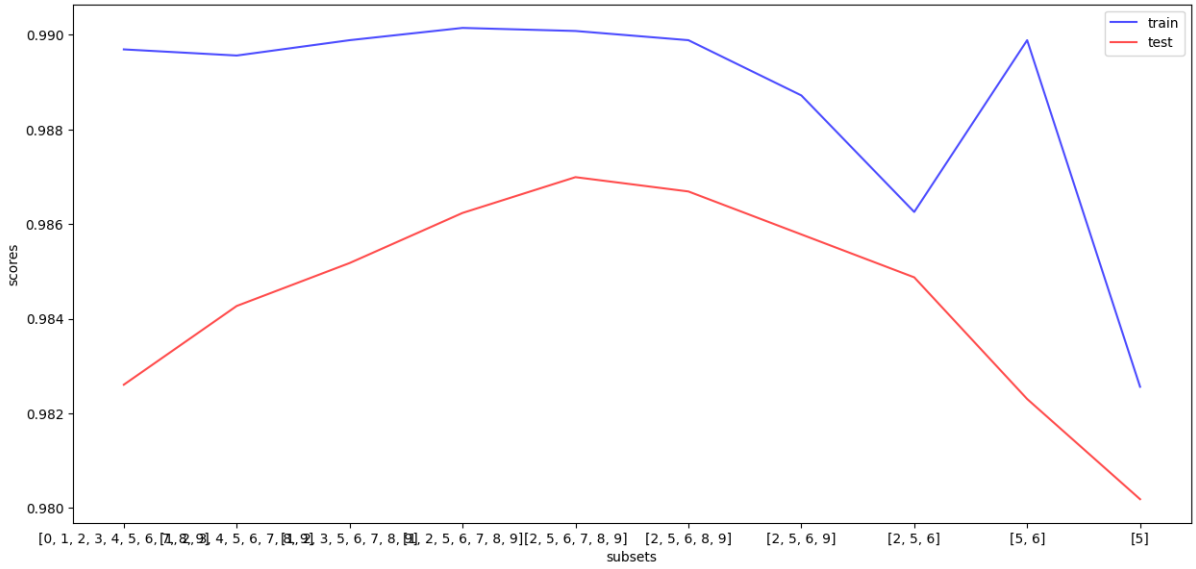

그림 16 Bridge stretch 에서 센서 조합에 따른 train / test set score

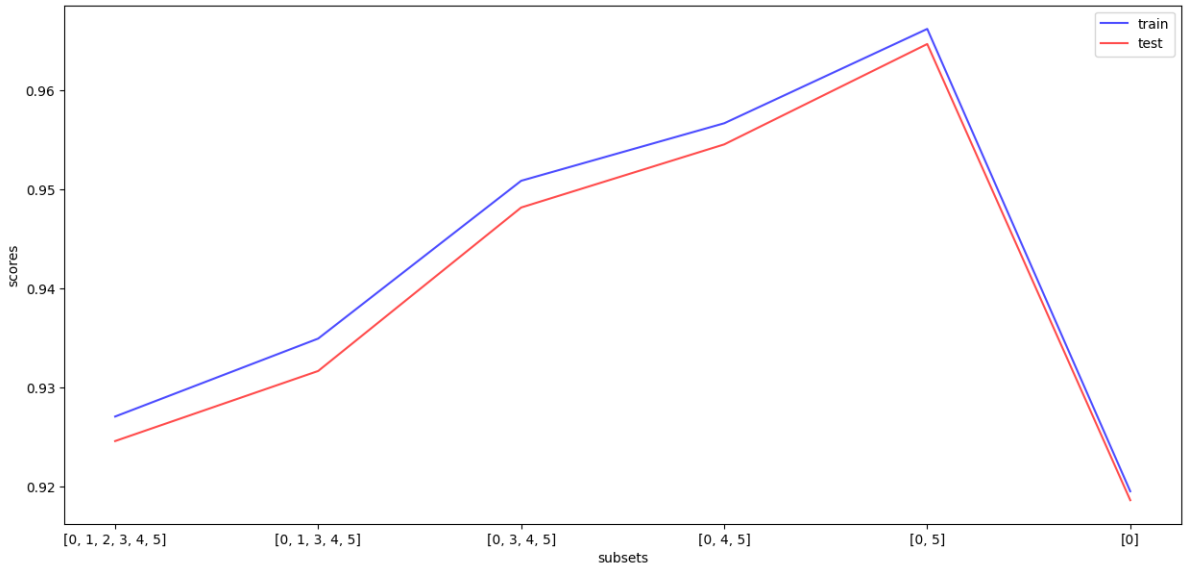

그림 17 Neck side extension 에서 센서 조합에 따른 train / test set score

그래프를 참고하여 알맞은 센서 조합을 선택하였고, 이 조합대로 모델을 다시 학습하여 학습 결과를 기록하고 모델을 따로 저장하였다.

- Bridge stretch : [5,6,7,8,9] (복부와 다리 전체)

- 그래프에 따르면 2,5,6,7,8,9의 조합이 가장 적절해 보이지만, 2번 센서(왼쪽 손목)는 Bridge 자세를 취하는 데 중요한 요소가 아니고 오히려 추론에 방해가 될 수 있는 부분이라 제외하였다.

- Neck side extension : [0,5] (머리와 복부)

이번엔, 반복 동작의 정상 수행 여부를 추론하는 자세인 Assisted shoulder flexion, Hamstring stretch 운동 자세 데이터셋을 이용해 추론 모델을 구성하고 학습한다. 이번 자세의 데이터는 Time-series data 이므로 순차적인 data 처리에 강점을 보이는 RNN 구조가 적합하다고 판단했다. 또한 다양한 RNN 의 종류 중에서 시간적 순서의 영향을 잘 잡아내는 LSTM 을 선택하여 진행했다. 모델은 다음과 같이 LSTM 레이어를 2층 쌓고 dropout 층과 Dense 층을 추가하여 구성하였고, 훈련은 100 epoch 동안 수행하였다.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 200, 50)	22,200
lstm_1 (LSTM)	(None, 50)	20,200
dropout (Dropout)	(None, 50)	0
dense (Dense)	(None, 50)	2,550
dense_1 (Dense)	(None, 2)	102

Total params: 45,052 (175.98 KB)
Trainable params: 45,052 (175.98 KB)
Non-trainable params: 0 (0.00 B)

그림 18 모델 구조 요약

그 후, SVM 을 이용하여 자세를 추론했던 것과 동일하게 여기에서도 센서를 Backward elimination 방식으로 하나씩 제거해가며 최적의 센서 조합을 찾아갔다. 먼저, Assisted shoulder flexion 의 경우 팔만 사용하는 운동이므로 하지의 센서(6,7,8,9)는 제외하고 시작하였다.

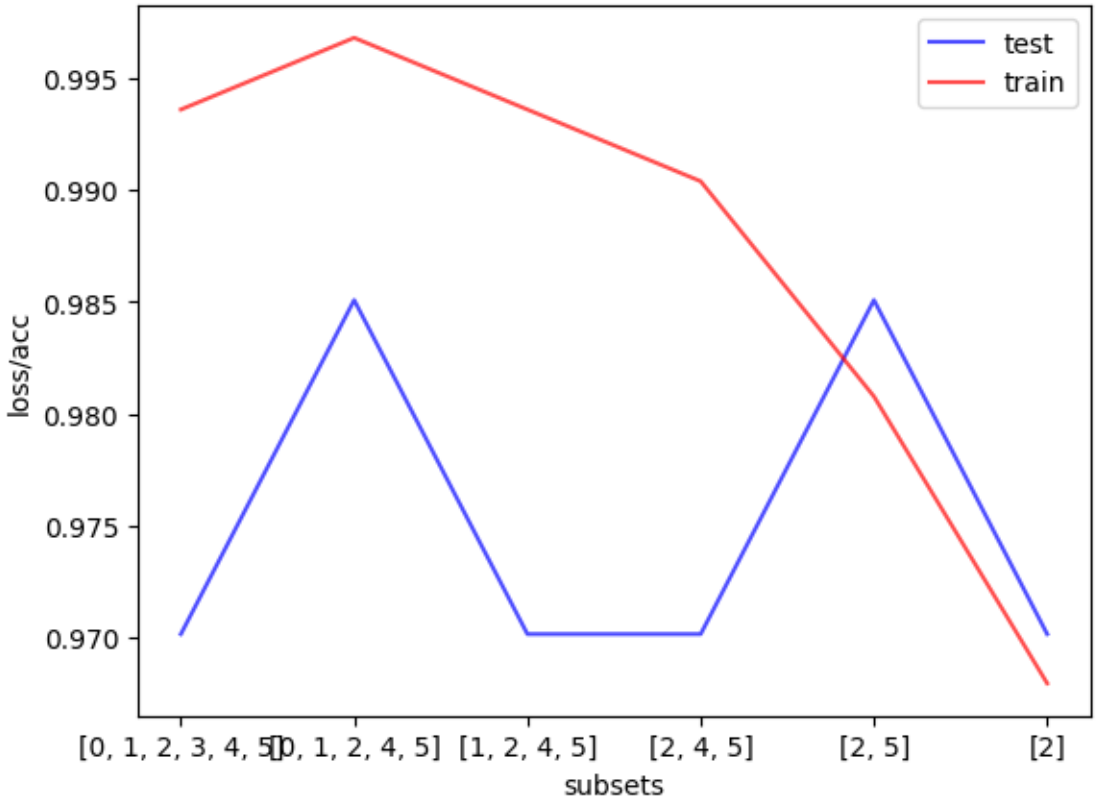

그림 19 Assisted shoulder flexion 에서 센서 조합에 따른 train / test accuracy

그래프를 확인해 2번, 4번, 5번의 센서를 선택하여 학습한 뒤 해당 센서를 착용해 실제로 추론이 잘 이루어지는지 확인해 보았다. 하지만 Neck side extension 이나 Bridge stretch 와는 다르게, 실제 추론 시 test dataset 에서 보인 성능과 실제 추론 결과에 대한 괴리가 심했다.

그 이유를 의논해 보았고 결국엔 데이터의 양이 아주 빈약하여 모델 학습에 지장이 갔다는 결론에 도달하였다. 10초 동안의 동작이 하나의 sequence data 를 구성하기 때문에, 한 시간 동안 데이터셋을 모아도 그 수는 3600(초)/10 = 360개의 sequence 에 불과하다는 사실을 깨닫게 되었다.

따라서 부족한 데이터로 인한 모델 학습의 문제와 실제 환경에서의 불안정한 추론 능력을 보완할 방법을 찾고자 데이터를 가지고 다양한 시도를 해보았다

1. 기존에 정한 IMU 센서의 sampling rate 인 20Hz 를 감소

추론하고자 하는 사람의 운동 자세는 단순히 10초 동안 관절 하나에 대한 왕복 움직임일 뿐인데, 그것보다 20Hz 의 sampling rate 는 조금 빠르다. 찰나의 움직임에도 민감하여 학습에 영향을 줄 수 있다고 생각했다. 따라서 10Hz, 5Hz 로 sampling rate 를 낮추었을 때 학습이 잘 이루어지는지 확인해 보았다. 이때 10Hz, 5Hz 처럼 0.5배로 속도를 줄이게 되면, 이 속도에 맞춰서 데이터셋을 새로 수집할 필요 없이, 기존 데이터셋에서 짝수 번째의 데이터 행을 지우기만 하면 된다.

2. 감소한 sampling rate 만큼 데이터 증가

또한 sampling rate 를 낮추게 되면 그만큼 기존에 수집한 데이터를 더 사용할 수 있게 된다. 만약 20Hz 에서 10Hz 로 sampling rate 를 줄였을 때, 짝수 번째의 데이터만 사용하게 되면 홀수 번째의 데이터를 새로 다시 구성할 수 있는 것이다. 즉 sequence 의 수가 2배가 되는 것이다. 당연하게도 짝수 번째 데이터와 홀수 번째 데이터는 같은 자세를 표현하고 있지만, 같은 데이터는 아니므로 이것을 단순 복제라고 볼 수는 없을 것이다.

이를 반영하여 Assisted shoulder flexion 자세에 대해 Backward elimination 방식으로 센서를 제거하며 모델 성능을 확인하였고 그 결과는 아래 그래프와 같다.

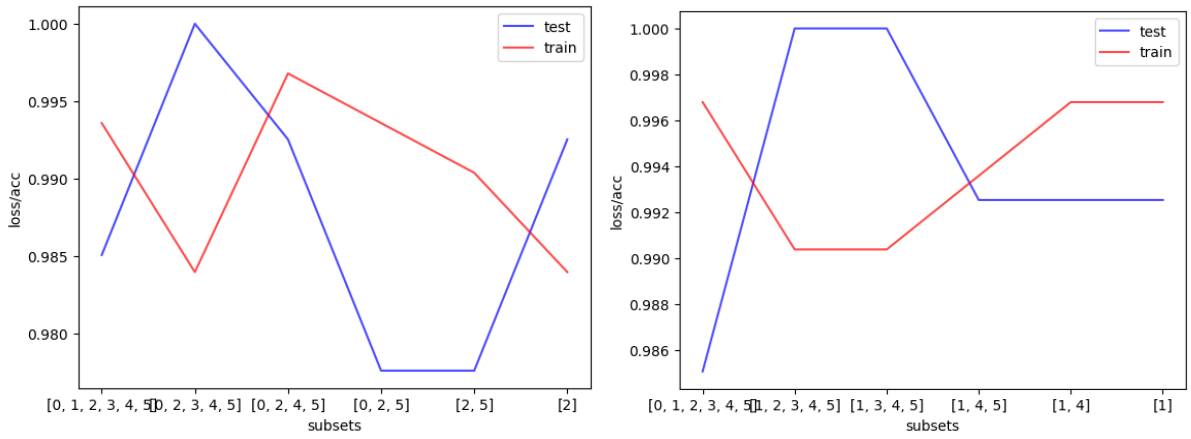

그림 20 Assisted shoulder flexion 에서 sampling rate 감소; 10Hz(좌), 5Hz(우)

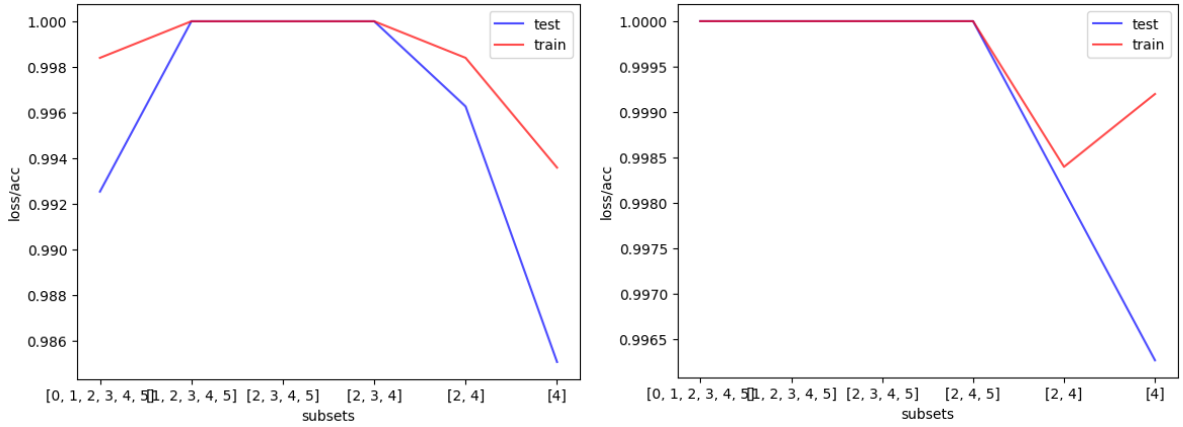

그림 21 Assisted shoulder flexion 에서 sampling rate 감소, 데이터 증가; 10Hz(좌), 5Hz(우)

그래프를 확인하면 다양한 조합이 눈에 띄어 보이지만, 그래도 [2,4,5] 세트의 센서를 사용하는 것으로 결정하였다. 착용해야 하는 센서가 좌우로 균형이 잡혀 있기 때문이다. (최종적으로 착용해야 할 위치를 고른다면 약간의 손실을 감안한다고 해도 보기에 균형 있게 고르는 것이 중요하다고 판단하였다)

Hamstring stretch 도 같은 방식으로 학습을 진행하였고 그 결과는 아래와 같다.
(Hamstring stretch 의 경우 상체의 센서 [0,1,2,3,4]를 먼저 제거하였음)

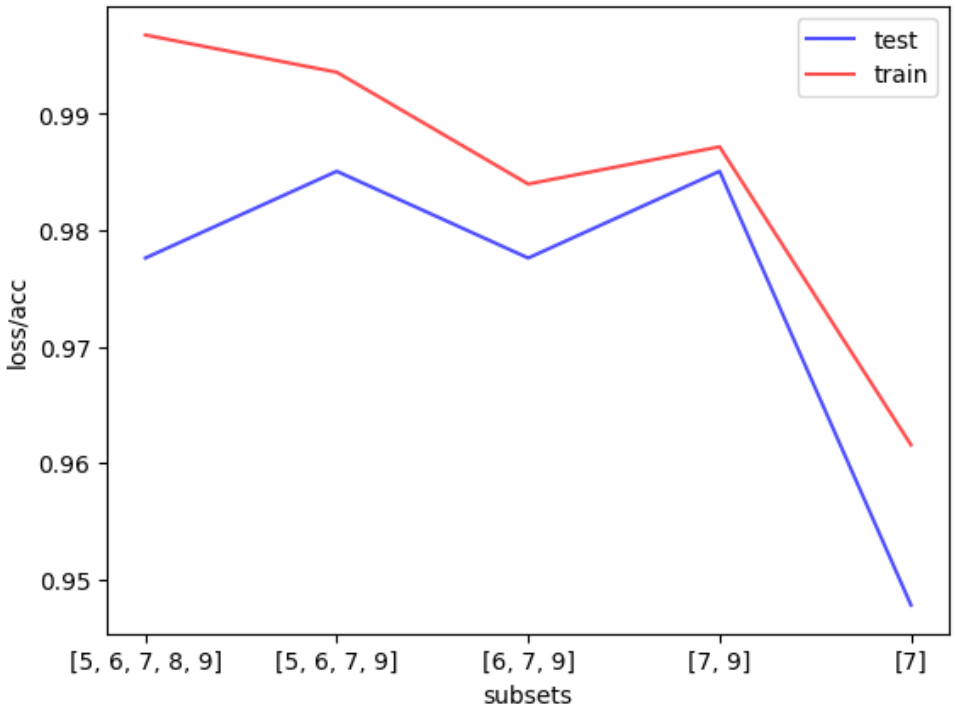

그림 22 Hamstring stretch 에서 센서 조합에 따른 train / test accuracy

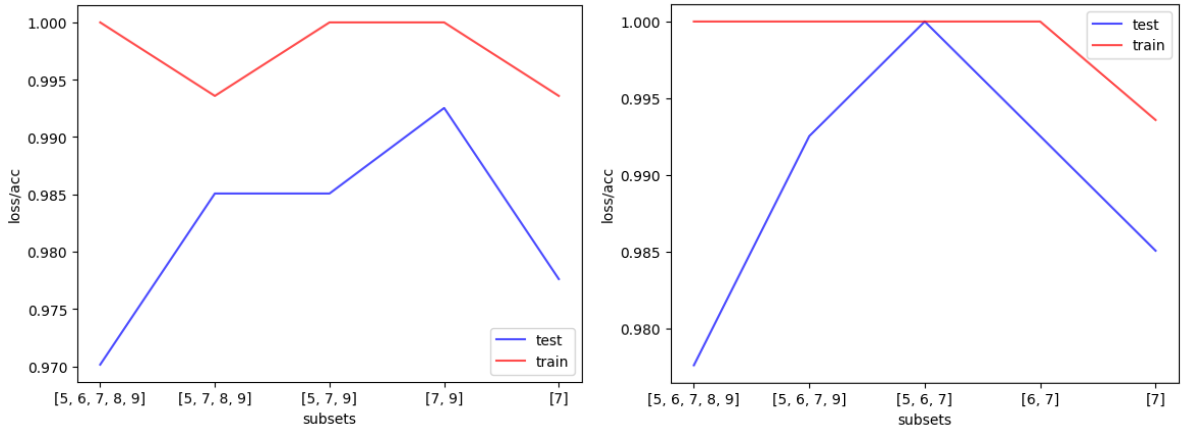

그림 23 Hamstring stretch 에서 sampling rate 감소; 10Hz(좌), 5Hz(우)

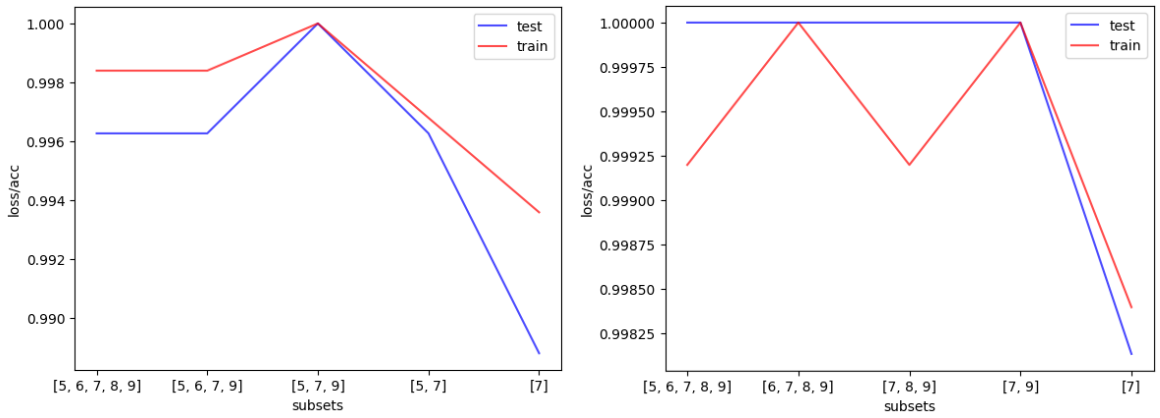

그림 24 Hamstring stretch 에서 sampling rate 감소, 데이터 증가; 10Hz(좌), 5Hz(우)

최종적으로 추론에 사용할 센서 위치는 [5,6,7]로 결정하였다.

3.4. User Interface 구축

기존에서는 아래와 같이 CLI 기반으로 보드와의 BLE 연결, 데이터 수집, 학습, 추론을 수행하였다. 그러나 일반 사용자 입장에서 CLI 환경은 조작하기 불편하기 때문에, 이것을 GUI 로 나타낼 필요가 있었다. 파이썬에서 GUI 환경을 구현하려면 PyQt5나 Tkinter 등을 이용할 수 있지만 이번 과제에서는 사용자에게 더욱 친숙한 웹 환경을 이용하여 GUI 를 구현해 보았다.

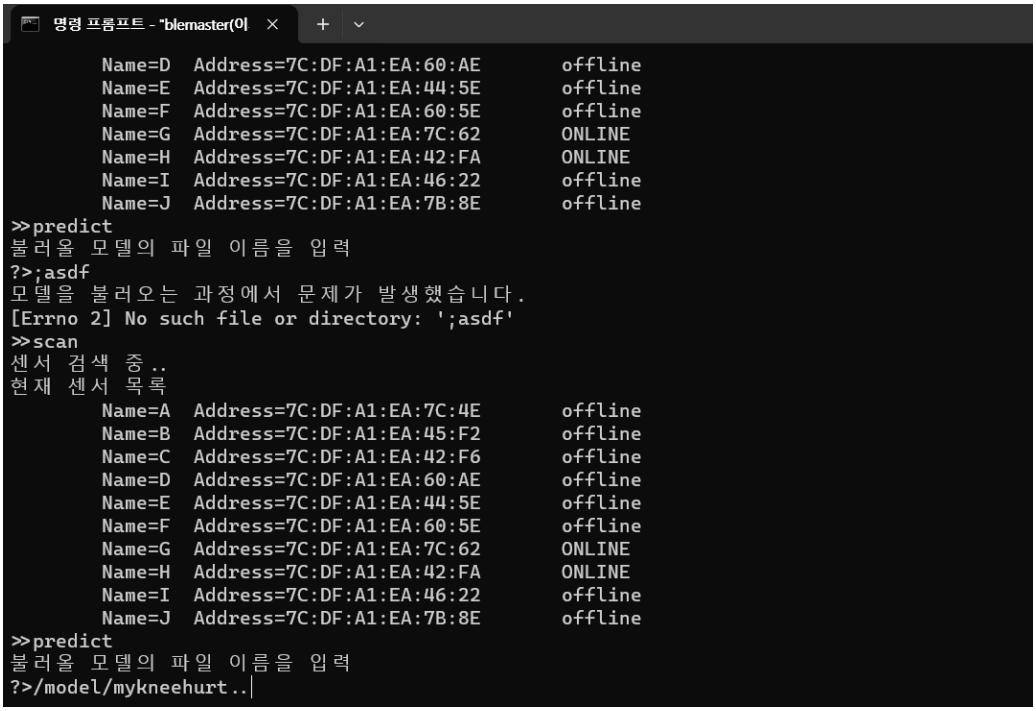

그림 25 CLI 환경에서 자세 데이터 수집, 자세 추론 수행

웹 환경에서 GUI 를 구현하기 위해, 먼저 HTML 과 CSS, JavaScript 를 통해 기본적인 디자인과 동작을 구현하였다(프론트). 다음으로는 기존 CLI 환경에서 장치 스캔, 모델 추론 등을 담당하는 파이썬 코드가 FastAPI 를 이용하여 웹 서버를 구동해, 프론트에서 Get 이나 Post 로 날아오는 요청에 반응해 그에 맞는 작업을 수행한 후 적절한 응답을 보내 주는 것으로 GUI 파트를 구현하였다.

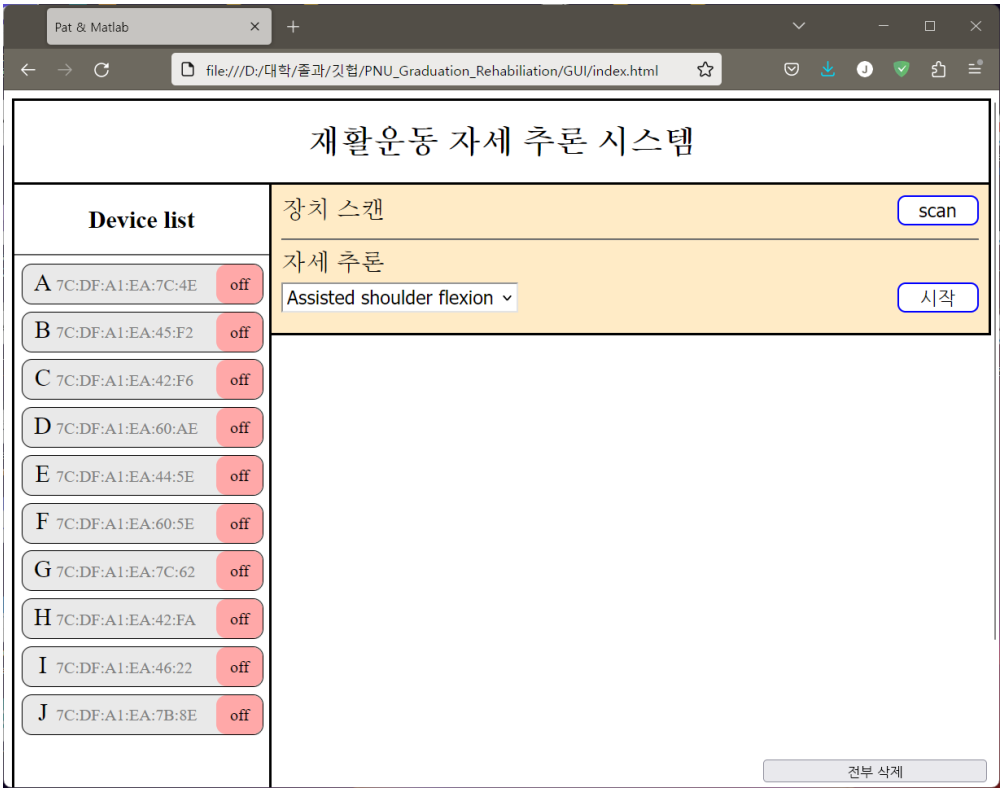

그림 26 기본 웹 화면

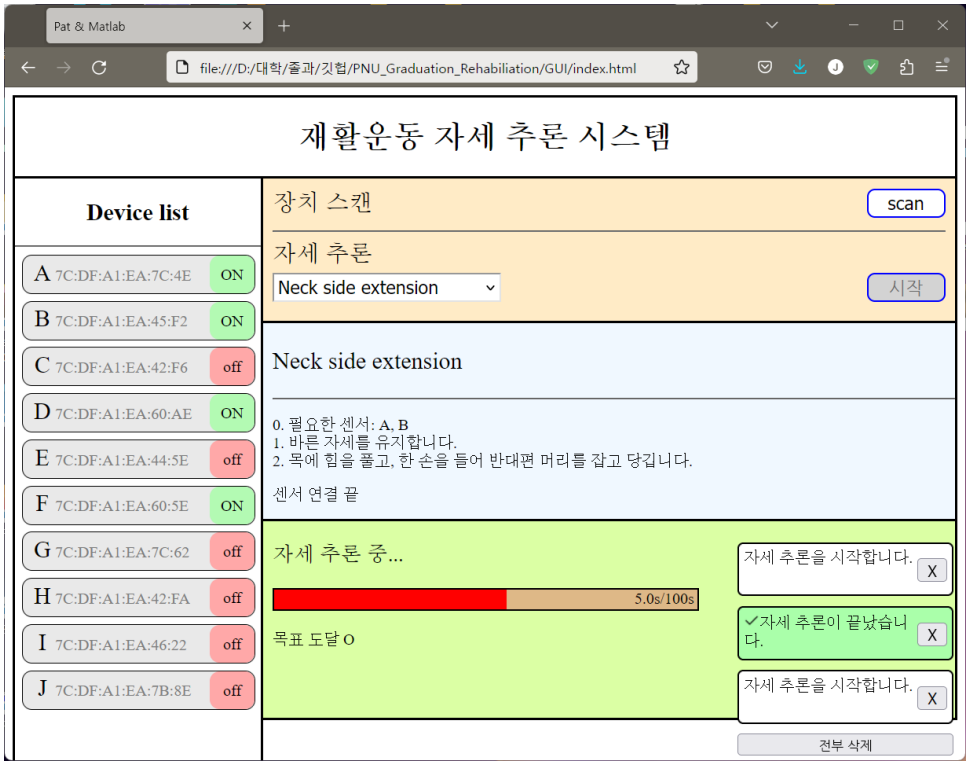

그림 27 자세 추론 중의 웹 화면

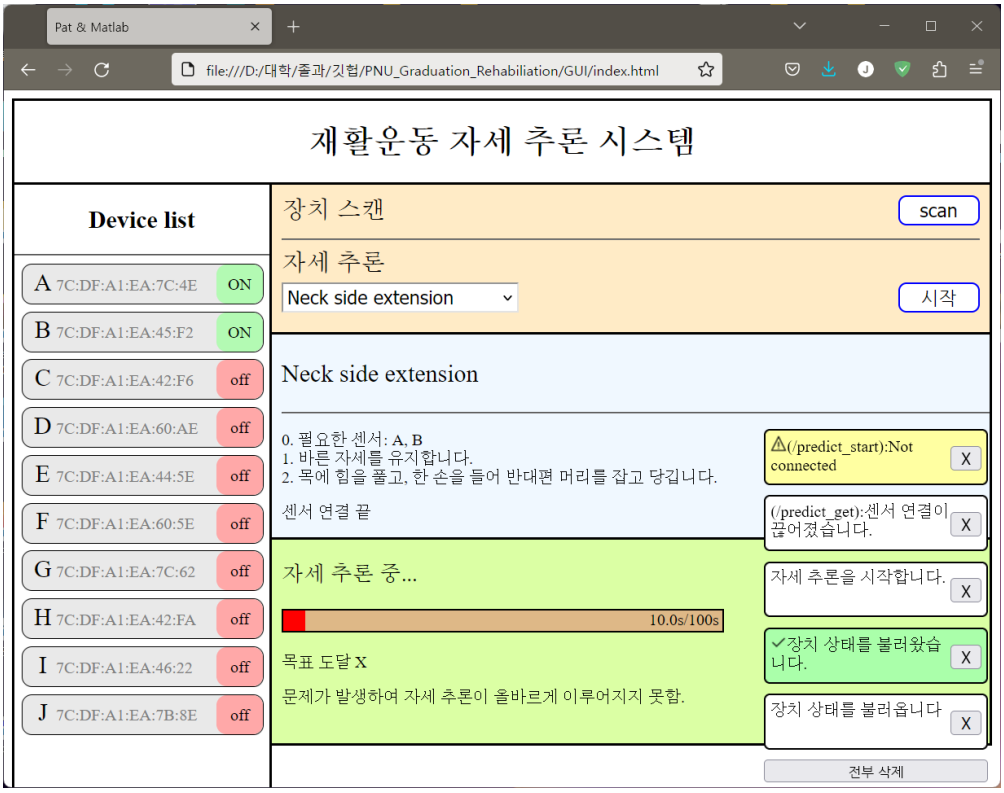

그림 28 추론 중 문제 발생 시 알림 발생

4. 연구 결과 분석 및 평가

SVM 과 LSTM 을 이용하여 4가지 재활운동 자세의 동작을 학습하였고, 최종 추론에 사용할 센서 수를 결정하였다. 결정된 센서 위치에 따른 학습 결과는 다음과 같다.

- Bridge stretch

하이퍼파라미터	kernel = "rbf", C = 100	
센서 조합	[5,6,7,8,9]	
훈련/평가 데이터셋 정확도	0.9930	0.9845

- Neck side extension

하이퍼파라미터	kernel = "rbf", C = 100	
센서 조합	[0,5]	
훈련/평가 데이터셋 정확도	0.9606	0.9508

- Assisted shoulder flexion

  사용한 센서 조합 = [2,4,5]

  사용할 Sampling rate : 10Hz

Sampling rate (Hz)	훈련 데이터셋 정확도	평가 데이터셋 정확도	비고
20	0.9871	0.9701	-
10	0.9903	0.9776	-
10	1.0	1.0	데이터 증가
5	0.9967	0.9925	-
5	1.0	1.0	데이터 증가

- Hamstring stretch (왼쪽 다리)

  사용한 센서 조합 = [5,6,7]

  사용할 Sampling rate : 10Hz

Sampling rate (Hz)	훈련 데이터셋 정확도	평가 데이터셋 정확도	비고
20	0.9839	0.9402	-
10	0.9871	0.9776	-
10	1.0	1.0	데이터 증가
5	0.9967	0.9850	-
5	1.0	1.0	데이터 증가

먼저, Bridge stretch 와 Neck side extension 운동 자세의 경우 SVM 을 이용하여 모델을 학습하였으며, 평가 데이터셋(30%)의 정확도가 각각 98.48%, 95.08%를 기록하였다. LSTM 을 이용한 두 운동(Assisted shoulder flexion, Hamstring stretch) 또한 평가 데이터셋의 정확도가 94% 이상으로 높게 측정되었다.

실제 센서를 정해진 위치에 착용하고 자세 추론 결과를 살펴보았는데, SVM 을 이용한 목표 자세 달성 여부 추론(Neck side extension, Bridge stretch)은 매우 잘 이루어짐을 확인할 수 있었으나, LSTM 을 이용해 특정 시간(10초) 동안의 반복 운동의 성공 여부를 추론하는 것의 정확도는 거의 60%-70%정도밖에 되지 않았음을 알게 되었다. 이는 평가 데이터셋의 정확도와는 매우 차이 나는 결과임을 알 수 있는데, 이전에 언급한 대로 데이터셋의 개수가 학습에 필요한 양만큼 충분하지 않아 일어난 일이라고 할 수 있다.

또한 센서를 착용하고 자세 추론을 할 때, 데이터셋을 수집할 때 당시의 환경이나 개인의 습관도 학습에 영향이 간다는 것을 확인할 수 있었다. 예시로 Hamstring stretch 를 수행할 때, 허리를 곧게 펴지 않고 앉거나 의자 높이가 다른 경우에도 추론 결과에 큰 영향을 끼치는 것을 확인할 수 있었다. 즉 무의식적으로 나오는 습관(허리 굽힘, 발 떨기, 가만히 있을 때의 팔다리의 위치 등)과 수집할 당시의 환경이 모델 학습에 영향을 받는다는 것을 알 수 있었는데, 이것도 결국엔 데이터를 더 많이 모으면 해결할 수 있다.

5. 결론 및 향후 연구 방향

본 과제에선 IMU 를 이용한 경량 재활 운동 자세 추론 시스템 개발을 위해, 먼저 BLE 통신으로 6축 IMU 데이터를 전송받을 수 있는 펌웨어를 개발하였다. 그리고 재활 운동 동작을 주요 신체 부위마다 하나씩 총 4개를 선정해 데이터셋을 수집한 뒤, 재활 운동 동작의 유형을 나눠 데이터셋을 머신러닝 모델에 학습하였다. 마지막으로 재활 운동 자세 별 센서 개수의 최적화를 거친 뒤, 실제 자세 추론을 시행하여 모델의 자세 추론 정확도를 확인하였다.

본 과제 수행 중 개발보드 및 배터리, 벨크로 밴드와 같이 과제를 수행하는 데 필요한 물품의 결정과 수령이 예상보다 늦어졌고, 후반부에 데이터셋 수집 방법의 변경으로

인해 과제를 수행하기 위해 주어진 시간을 여유롭게 사용하지 못하였다. 따라서 양적으로도, 질적으로도 좋은 데이터셋을 수집하지 못해 성능 높은 모델을 완성하기엔 무리가 있었고, 실제 자세 추론의 정확도도 훈련/평가 데이터셋의 정확도와 비교하였을 때 많이 떨어지게 되었다. 이 점이 이번 과제에서 가장 아쉬운 부분이다. 그래도 6축 IMU 데이터를 이용하여 적은 수의 센서로도 특정 자세의 추론이 가능하다는 것을 확인하게 되었다. 만약 더 많은 피험자를 모집하여 더 많은 데이터셋을 수집할 수 있다면, 좀 더 일반화된, 그리고 범용적으로 좋은 성능을 내는 모델 완성을 기대해 볼 수 있겠다. 향후 이를 진행할 수 있는 여유가 된다면 프로젝트로 진행해 볼 예정이다.

또한 다수의 보드와 BLE 연결을 통해 게이트웨이가 데이터를 받아 오게 되는데, 기기마다 동시에 연결이 가능한 수가 제한되어 있다는 것을 확인하였다. ESP32-S3정도면 충분히 처리 용량이 괜찮은데, 신체에 부착된 MCU 끼리 Mesh 와 같은 형태로 서로 연결되어 복잡한 계산이 필요한 자세 추론을 나누어서 담당하고, 추론 결과를 게이트웨이와 연결된 하나의 보드만 게이트웨이에 전달하게 되는 시스템의 개발 또한 향후에 연구해 볼만한 주제인 것 같다.

6. 참고 문헌

[1] QMAR: Quality of Movement Assessment for Rehabilitation, https://data.bris.ac.uk/data/dataset/1y37kc9a8y47y2cen7j907bpm7

[2] University of Idaho - Physical Rehabilitation Movements Data Set (UI-PRMD), https://www.webpages.uidaho.edu/ui-prmd/

[3] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, Gerard Pons-Moll, "Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time", ACM Transactions on Graphics (TOG), Vol. 37, Issue 6, No. 185, pp. 1-15, December 2018

[4] ICM-42670-P Datasheet, https://invensense.tdk.com/wp-content/uploads/2021/07/DS-000451-ICM-42670-P-v1.0.pdf

[5] ESP32 Bluetooth Low Energy Tutorial with ESP-IDF: Menuconfig and Code Implementation Explained, https://innovationyourself.com/esp32-bluetooth-low-energy-tutorial/

[6] Rehabilitation exercises, The Chartered Society of Physiotherapy, https://www.csp.org.uk/public-patient/rehabilitation-exercises

[7] Chenyu Gu, Weicong Lin, Xinyi He, Lei Zhang, Mingming Zhang, "IMU-based motion capture system for rehabilitation applications: A systematic review", Biomimetic Intelligence and Robotics, Vol. 3, Issue 2, 2023

[8] C. J. Lee, J. H. Kim, & J. K. Lee, "A Recurrent Neural Network for Magnetometer-free 3D Joint Angle Estimation based on 6-axis IMU Signals", Proceedings of KSPE 2023 Spring Conference,Vol. 40, No. 4, pp. 301-308, April 2023