Pusan National University
Computer Science and Engineering
Technical Report 2022-32

인체의 움직임을 디지털 형태로 기록하는 모션 캡처 시스템 개발

저자 1 박해미

저자 2 박민수

지도교수 정상화

목 차

1. 서론 1

1.1. 연구 배경 1

1.2. 기존 문제점 1

1.3. 연구 목표 2

2. 연구 배경 3

2.1. 배경 지식 3

2.1.1. 관성 센서 (X-NUCLEO-IKS01A3) 3

2.1.2. roll, pitch, yaw 4

2.1.3. BLE 통신 (STM32WB55) 4

2.1.4. 소켓 통신 6

2.1.5. Quaternion 7

2.2. 개발 환경 7

3. 연구 내용 8

3.1. 데이터 수집 및 처리 8

3.2. 데이터 통신 및 Gateway 12

3.3. 데이터 시각화 13

4. 연구 결과 분석 및 평가 14

4.1. 설계 변경 내용 14

4.1.1. 네트워크의 형태 14

4.1.2. 디지털 형태로의 구현 범위 14

4.2. 연구 결과 15

5. 결론 및 향후 연구 방향 16
5.1. 결론 16
5.2. 향후 연구 방향 16
6. 개발 일정 및 역할 분담 17
6.1. 개발 일정 17
6.2. 역할 분담 18
7. 참고 문헌 19

1. 서론

1.1. 연구 배경

최근 여러 산업에서 모션 캡처 시스템이 유용하게 활용되고 있다. 영화나 게임에서 3D 모델링을 실감나게 구현하기 위해 인체의 움직임을 사용하거나 IT 업체에서 얼굴의 움직임을 파악해 이모티콘을 만드는 등 모션 캡처 시스템은 사회에 깊게 파고들고 있다. 이렇듯 모션 캡처를 이용한 시스템을 쉬이 볼 수 있지만, 이를 좀 더 보편화하여 일상에 적용하고 더 나아가 이를 실시간으로 디지털 형태로 기록할 수 있도록 적용해 보고자 하였다.

기존 모션 캡처 시스템은 캡처 대상에 마커를 사용하는 광학식과 관성 센서를 사용하는 방식으로 크게 나뉜다. 광학식은 많은 마커와 카메라를 요구하기 때문에 현실적으로 구현의 어려움이 많을 것으로 예상이 되어, 과제에서는 관성 센서를 이용하는 관성식 방식을 택하여 연구를 진행했다.

관성식 방식은 여러 센서를 연결하기 위해 각각의 무선 수신기를 달아 연결해야 하는 한계점이 있다. 이러한 문제로 인해 각각의 센서를 연결하기 위한 무선 통신 기술과 이 센서들의 정보를 받기 위한 여러 절차가 요구된다.

이러한 사실에 기반하여 관성식 방식에서 편의성과 성능을 향상시키는 방법에 대한 연구를 목표로 본 주제를 선정하였다.

1.2. 기존 문제점

관성식 모션 캡처 기술은 가속도 센서, 자이로 센서, 지자기 센서가 조합된 관성 센서를 신체에 부착하여 움직임, 회전, 방향을 읽어내는 방식이다. 광학식 모션 캡처 시스템과 달리 많은 카메라가 요구되지 않기 때문에 장소에 한정되지 않고 비교적 자유롭게 사용된다.

하지만, 정확도가 광학식에 비해 떨어지며 "연구 배경"에서도 설명했듯이, 기존 관성식 모션 캡처 방식은 센서 각각의 무선 송신기가 필요하며 이를 개별적으로 각각 연결하는 한계점을 지니고 있다.

1.3. 연구 목표

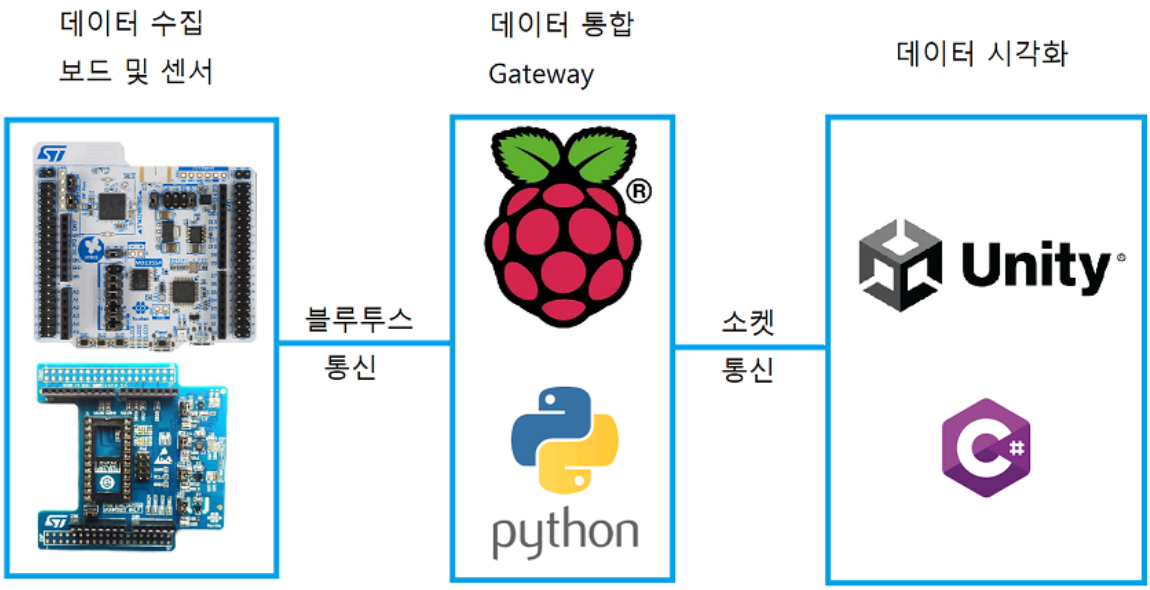
현재 모션 캡처 방식과 기술에는 한계점 및 제약사항들이 존재한다. 여러 사항들 중에서

(1) 편의성 및 성능을 향상시킨 무선 네트워크 시스템의 개선

(2) 해당 네트워크를 기반으로 시각화된 인체 모델의 디지털 형태의 움직임 확인

(3) 저가의 보급형 관성식 모션 캡처

이 세 가지 사항에 중점을 두고 연구의 방향을 설정하였다.



[그림 1] 전체 시스템 구성도

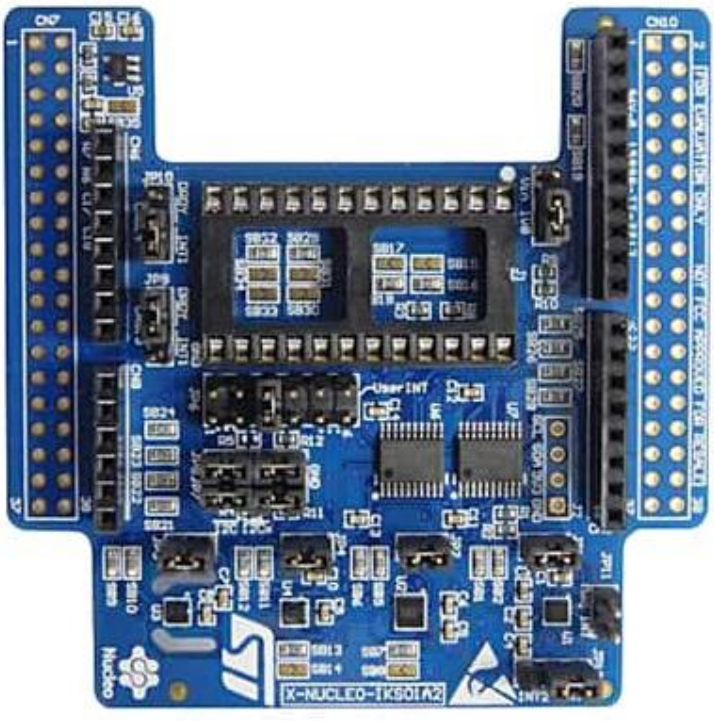
2. 연구 배경

2.1. 배경 지식

2.1.1. 관성 센서 (X-NUCLEO-IKS01A3)

관성 센서는 관성력을 측정하는 센서로 크게 6축과 9축으로 나뉜다. 연구에 사용한 관성 센서는 9축 관성 센서이며, 가속도계, 각속도계, 지자기계에 대해서 각각 3축, 총 9축으로 이루어져있다.

가속도계는 x, y ,z축에 대한 가속도를 측정하는 센서이며, 각속도계는 물체의 회전속도를 측정하는 센서이며, 지자기계는 지구의 자기장과 중력의 방향을 측정하는 센서이다. 세가지 종류의 센서들로부터 받아들인 데이터를 종합하여 위치에 대한 절대 위치(Absolute Reference of Orientation)를 구한다. 다만, 이 센서들은 높은 레벨의 노이즈를 포함할 수 있기에 노이즈에 대한 필터링을 거쳐야한다.

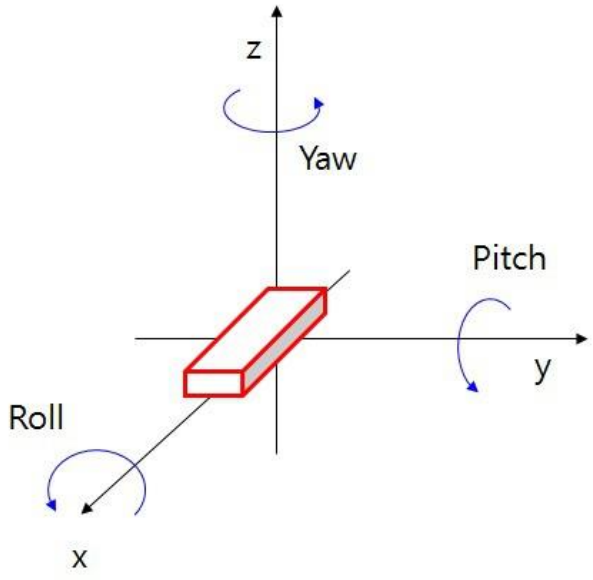


[그림 2] X-NUCLEO-ISK01A3

2.1.2. roll, pitch, yaw

연구의 최종적인 목적은 모션 캡처 데이터를 통한 3D 모델링 실시간 시각화이다. 인체의 구조상 신체의 특정 부위가 모습을 그대로 유지한 채 평행으로 3차원 공간에서 이동하지 않는다. 각각의 관절에 따라서 회전을 통해 움직임이 나타난다. 인체를 디지털 형태로 기록하기 위해서는 종축(x축), 횡축(y축), 수직축(z축)에 대한 회전값에 대한 이해가 필요하다.

roll는 x축에 대한 회전을, pitch는 y축에 대한 회전을, yaw는 z축에 대한 회전을 의미한다.



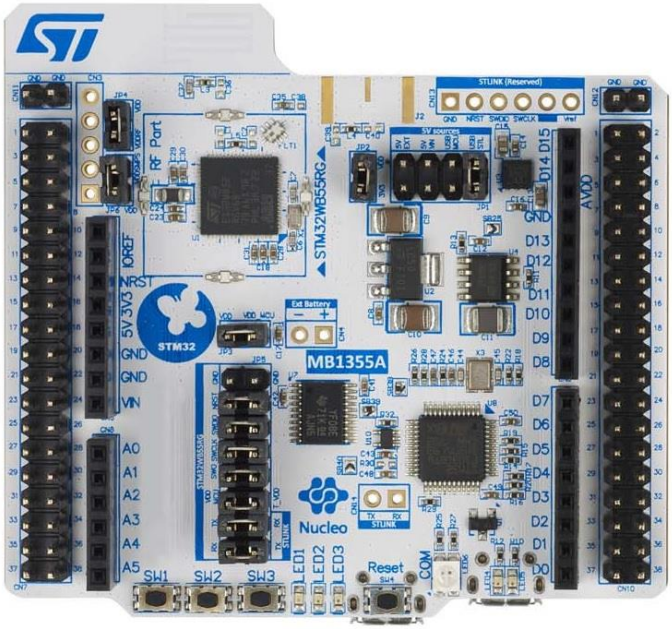
[그림 3] roll, pitch, yaw

2.1.3. BLE 통신 (STM32WB55)

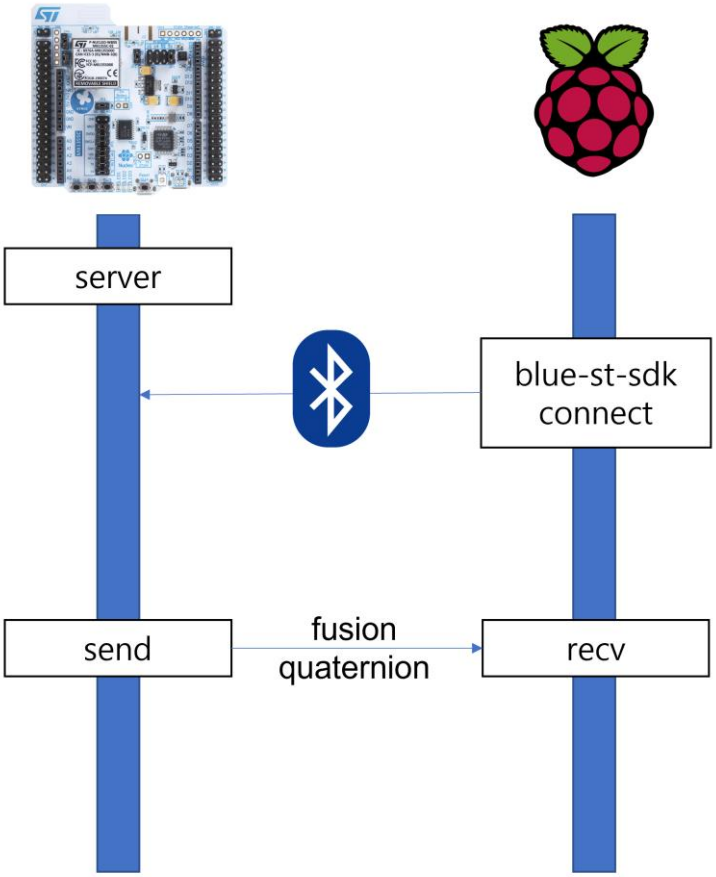
연구에 사용한 보드의 종류는 STM32WB55이다. 해당 보드는 Bluetooth Low Energy를 지원한다. 이를 이용하여 무선으로 데이터를 전송할 수 있도록 한다.

BLE의 연결 모드에는 크게 두 가지로 나뉜다. Advertising Mode, Connection Mode 둘로 나뉜다. 이 연구에서 사용할 모드는 Connecting Mode로 그 중에서도 다중 연결(Multiple Connection)을 이용할 것이다.

Master는 여러 대의 Slave와 연결이 가능하다. Slave장치는 오직 하나의 Master장치에만 연결이 가능하며, Slave 간의 통신은 불가능하다.



[그림 4] STM32WB55

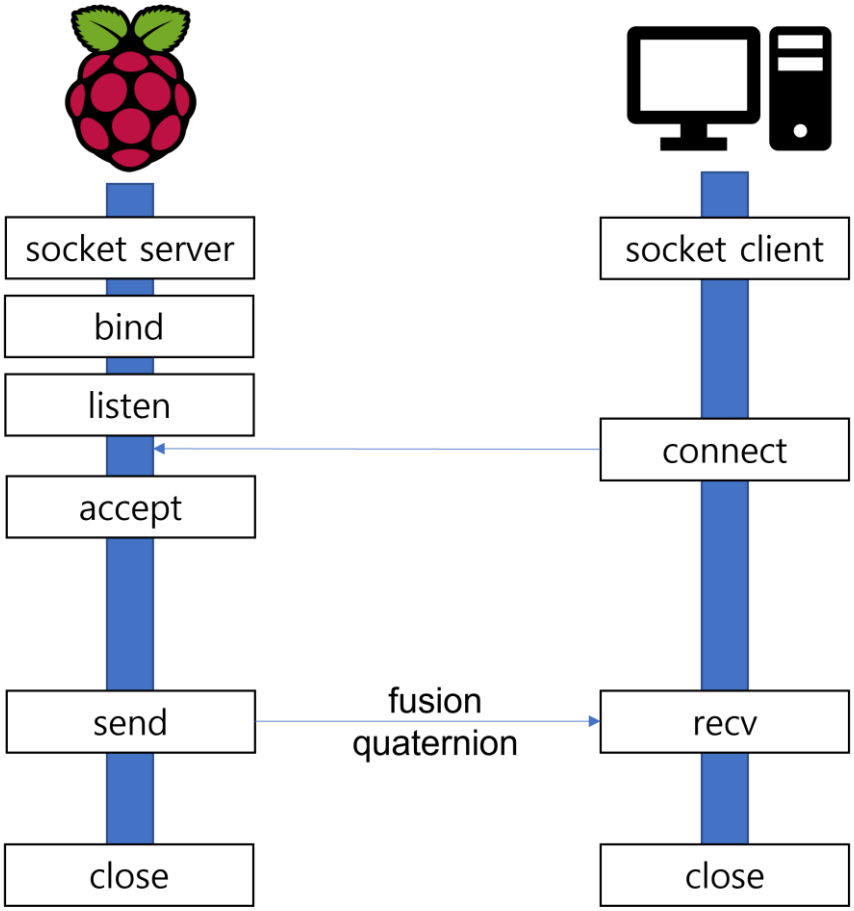


[그림 5] Bluetooth 통신

2.1.4. 소켓 통신

소켓은 데이터를 보내거나 받는 실질적인 창구 역할을 하며, 프로세스가 데이터를 보내거나 받을 때에 소켓을 열어서 데이터를 쓰거나 읽으며, 흔히 프로토콜, IP 주소, 포트 넘버로 정의된다.

소켓 통신은 서버와 클라이언트 양방향 연결이 이루어지는 통신으로, 클라이언트와 서버 모두 서로를 향해서 요청을 보낼 수 있다. 스트리밍이나 실시간 채팅과 같이 데이터를 서로 주고 받아야하는 경우에 주로 사용이 되며, 지속적으로 connection을 유지해야 하는 경우에 유용하다. 이 연구는 실시간으로 데이터를 주고받아야 한다는 점과 지속적으로 연결이 유지가 되어야하기에 소켓 통신을 선택한다.



[그림 6] 소켓 통신

2.1.5. Quaternion

3 차원 그래픽에서 회전을 표현할 때, 행렬 대신 사용하는 수학적 개념이다. 행렬에 비해서 연산 속도가 빠르기에 해당 데이터 형태로 나타낸다. 4 개의 수(x, y, z, w)로 이루어지며 각 성분은 축이나 각도를 의미하지 않는다. 하나의 벡터(x, y, z)와 하나의 스칼라값(w)를 의미한다. 회전의 원점과 특정 방향을 비교함으로 회전을 측정할 수 있도록 하는 구조다.

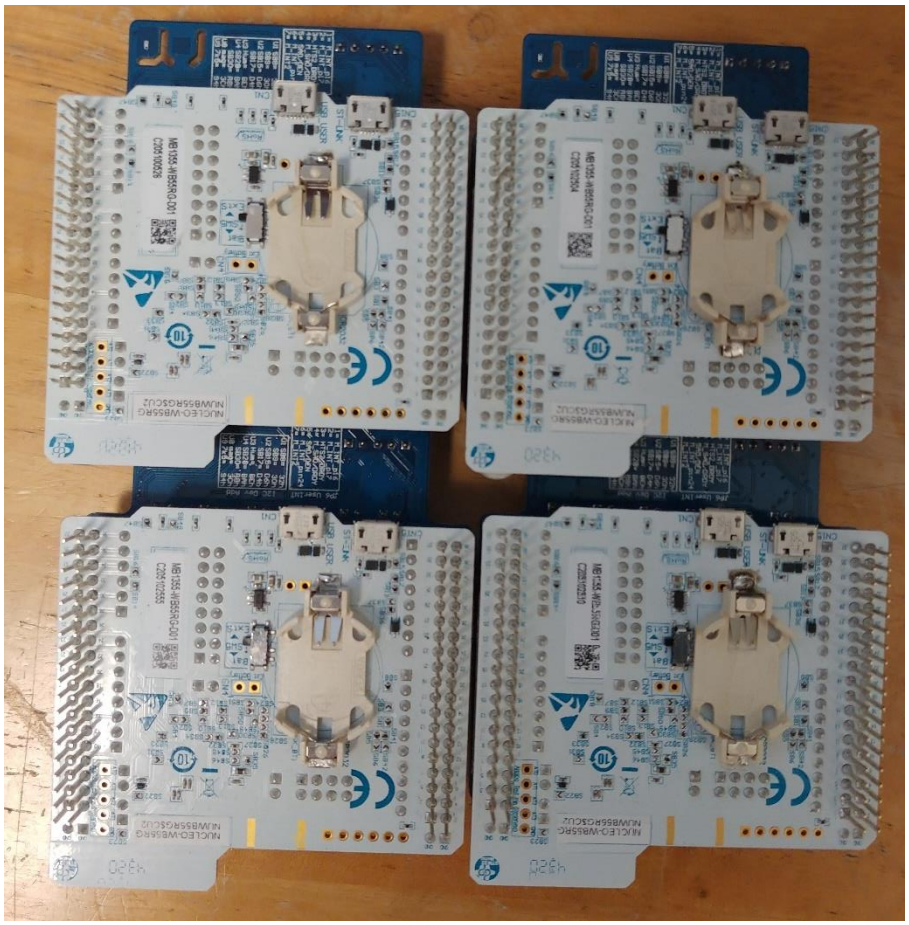
2.2. 개발 환경

윈도우 운영체제에서 실행이 가능하도록, 개발을 하였다. 개발 언어로는 C, Python, C#을 선택했다. C는 하드웨어 보드에 사용하였고, Python은 라즈베리 파이와 보드 간의 블루투스 통신, 라즈베리 파이와 유니티 간의 소켓 통신에 사용하였고, C#은 Unity 3D에 사용하였다.

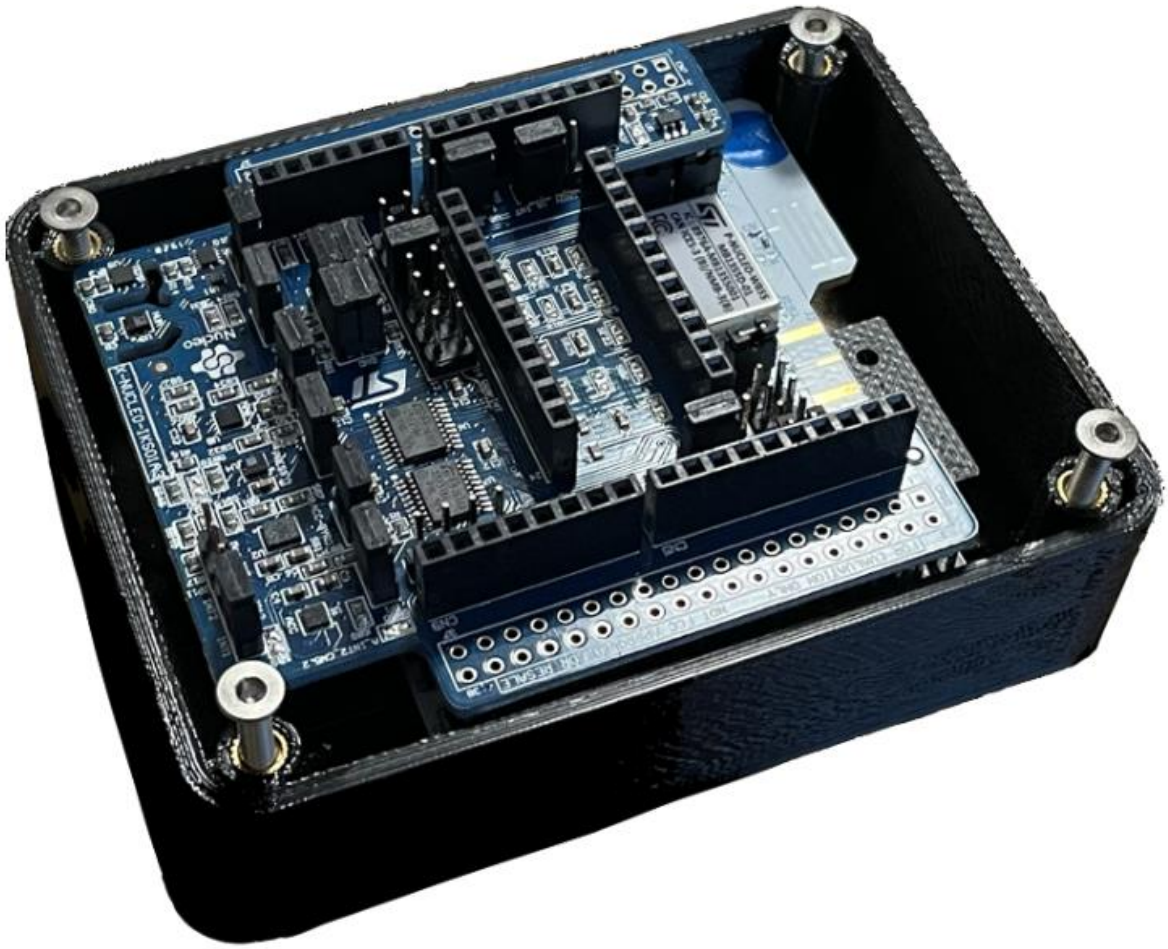
3. 연구 내용

3.1. 데이터 수집 및 처리

실제로 몸이 움직이는 것에 따른 데이터를 수집을 해야 하므로, 신체에 부착할 수 있는 형태로 데이터를 수집할 수 있는 장치를 만든다.



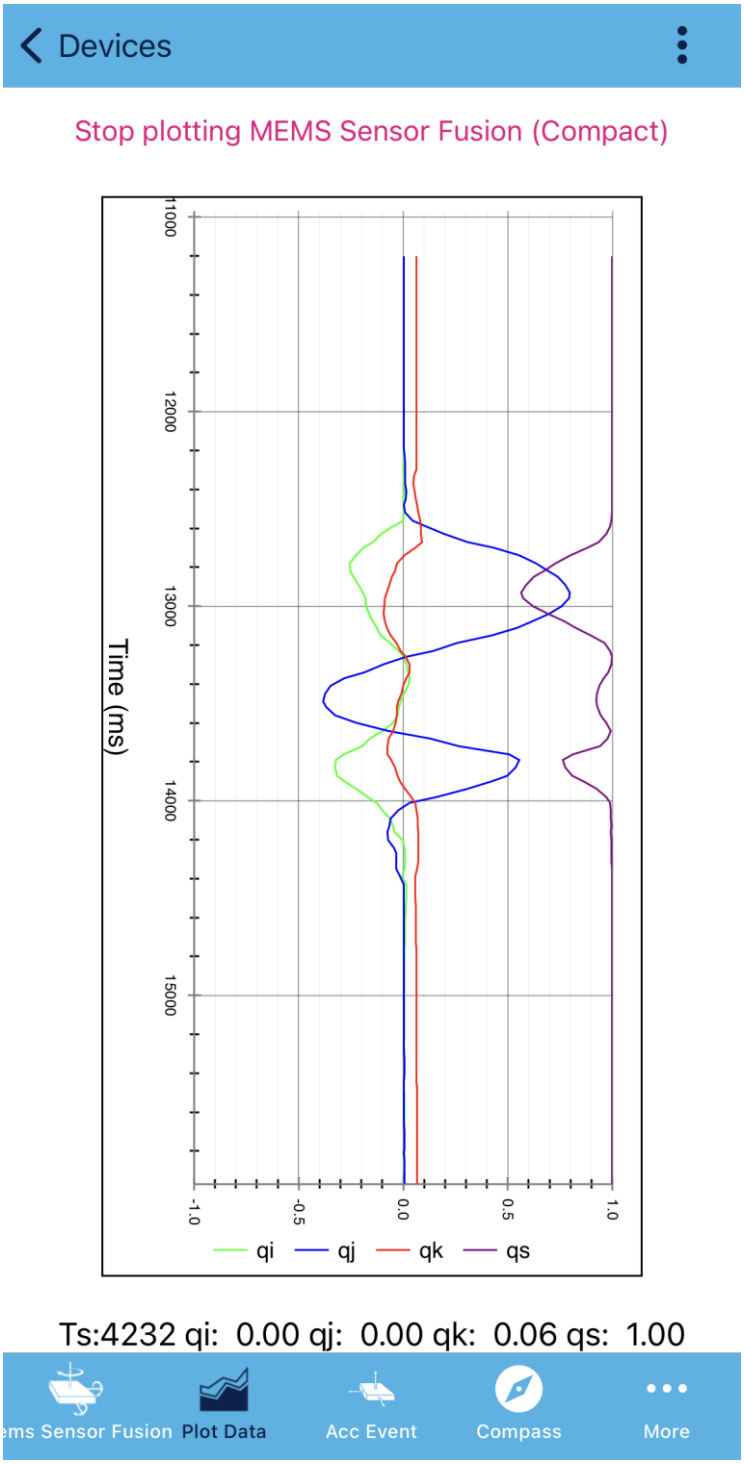
[그림 7] STM32WB55와 X-NUCLEO-IKS01A3



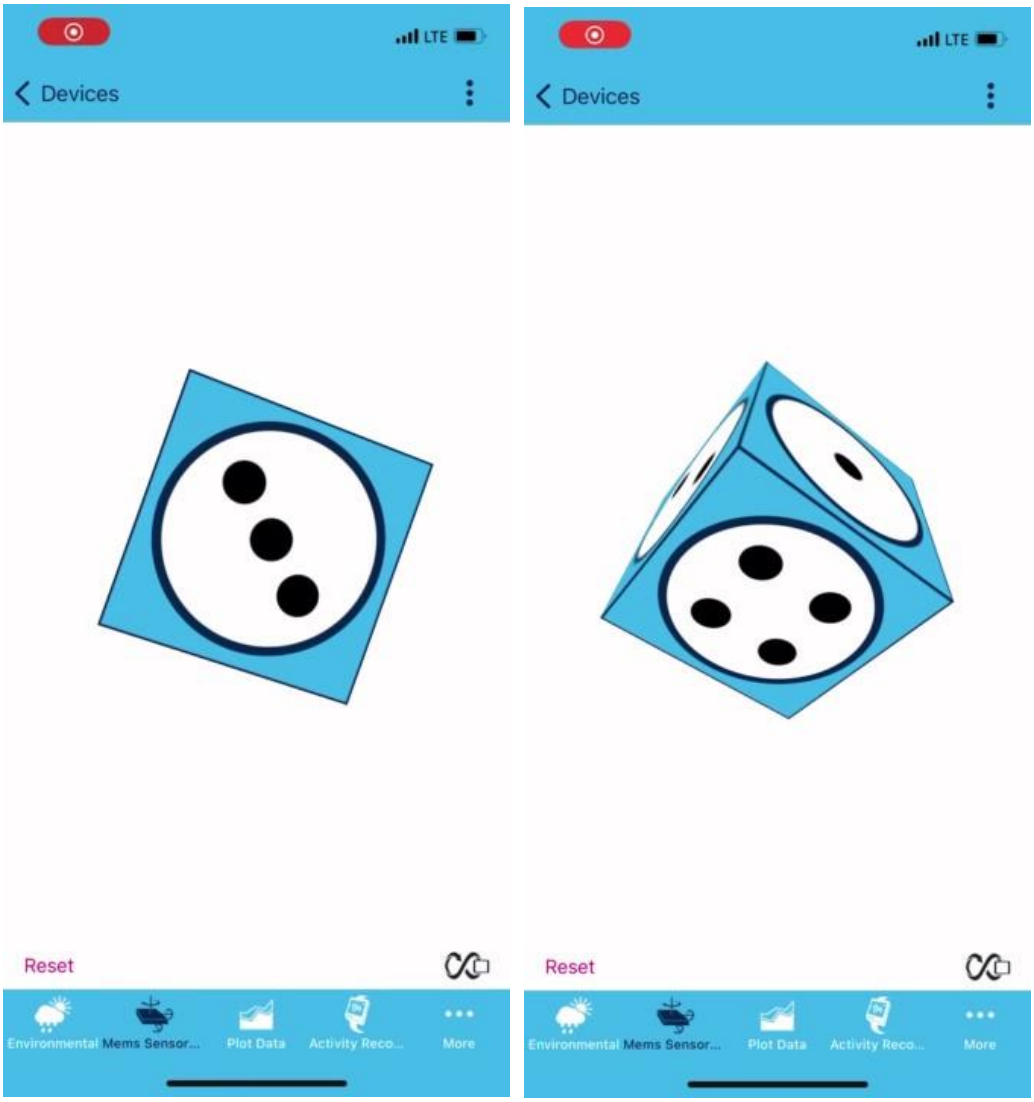
[그림 8] 그림 7의 장치를 담은 케이스

STM사에서 제공하는 Function Pack을 이용하여 가속도, 각속도, 지자계 센서의 데이터 값을 가공한다. 해당 데이터를 수신하는 과정에서 발생하는 노이즈를 자체적으로 필터링 한 후, MotionFX 파일 내의 함수를 이용하여 데이터를 Little Endian 타입의 Quaternion으로 만든다. 노이즈 및 데이터 값은 같은 회사에서 제공해주는 모바일 어플리케이션을 통해서 그래프로 확인이 가능하다.

이때 기존 Function Pack 의 센서 값 통신주기는 10ms 씩(100Hz) 총 3 번의 데이터를 보내며 이는 Gateway 에서 과부하가 생기는 문제가 발생한다. 이러한 이유로 기존 10ms 의 주기를 62.5ms(16.5Hz)로 수정한다.



[그림 9] 어플리케이션을 통해서 본 Quaternion 값



[그림 10] 하드웨어의 회전에 따라서 움직임을 보여주는 어플리케이션

a. 회전 전의 초기 모습

b. 회전 후의 모습

이후, 해당 데이터를 Bluetooth 무선 통신을 이용하여 송신해야하므로 통신을 True로 설정해준다. 이 때에 각각의 장치를 구분하기 위해서 ID를 다르게 부여해준다.

3.2. 데이터 통신 및 Gateway

3.1에서 전송되는 Little Endian 타입의 Quaternion 형태의 raw data를 Bluetooth 통신을 이용하여 Gateway로 수신한다.

이 때에 Gateway에서는 수신한 Little Endian 타입의 데이터를 float으로 변환하고, 변환한 데이터를 기반으로 qs를 계산한다.

$$qs = \sqrt{1 - (qi^2 + qj^2 + qk^2)}$$

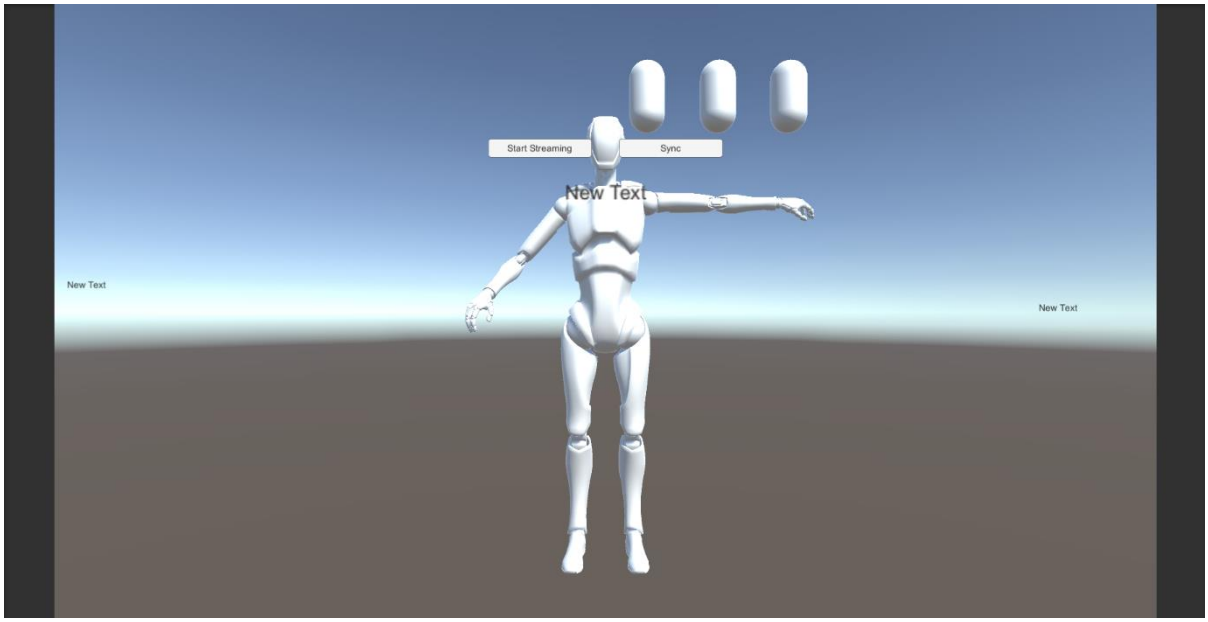
[그림 11] qs계산 공식

이후 계산한 qs와 더불어 float 형식으로 변환한 qi, qj, qk를 해당 장치의 ID와 timestamp와 함께 CSV 형태의 데이터를 소켓으로 write한다. 이후, 이 작성된 데이터를 소켓통신을 이용하여 Unity로 read한다.

```
P2PSRV2,MEMS Sensor Fusion (Compact)(6270),qi,0.001,qj,0.0045,qk,0.5835,qs,0.8121000554118932
P2PSRV2,MEMS Sensor Fusion (Compact)(6274),qi,0.001,qj,0.0045,qk,0.5835,qs,0.8121000554118932
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6275),qi,0.0011,qj,0.0045,qk,0.5835,qs,0.8120999261174698
P2PSRV2,MEMS Sensor Fusion (Compact)(6276),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact)(6277),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6278),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact)(6280),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact)(6281),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6282),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact)(6285),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact)(6286),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6287),qi,0.0011,qj,0.0046,qk,0.5835,qs,0.8120993658413974
P2PSRV2,MEMS Sensor Fusion (Compact)(6289),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6290),qi,0.001,qj,0.0046,qk,0.5835,qs,0.8120994951359101
P2PSRV2,MEMS Sensor Fusion (Compact)(6292),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact)(6291),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact): Unknown None sample
P2PSRV2,MEMS Sensor Fusion (Compact)(6293),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact)(6297),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact)(6294),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
P2PSRV2,MEMS Sensor Fusion (Compact)(6298),qi,0.001,qj,0.0047,qk,0.5835,qs,0.8120989225457697
```

[그림 12] 전송받은 Quaternion CSV 데이터

3.3. 데이터 시각화



[그림 13] 애플리케이션 기본화면

애플리케이션 실행 시, 모션 캡처할 모델 및 Start Streaming 버튼과 Sync 버튼을 확인할 수 있다. Start Streaming 버튼을 클릭할 경우 지정된 소켓 주소 및 포트번호로 연결을 시도하며 연결에 성공할 경우 "Connection Success"를 화면에 출력한다. 연결에 실패할 경우 "Connection Fail"을 화면에 출력한다. 소켓 연결이 성공하면 통신으로 데이터를 수신한다. 이후 Sync 버튼을 클릭하면 3.2 에서 read한 데이터에서 qi, qj, qk, qs를 추출하여 각 데이터를 유니티의 Quaternion으로 변환하여 지정된 각 부위를 회전시킨다.

4. 연구 결과 분석 및 평가

4.1. 설계 변경 내용

4.1.1. 네트워크의 형태

본 과제의 최초 설계 당시에는 Bluetooth Mesh라는 네트워크의 형태로 네트워크를 설계할 계획이었다. 하지만, 해당 네트워크를 이용하여 데이터를 송수신하기에는 데이터의 크기가 커서 송수신 간의 딜레이가 크게 발생할 것이라고 판단하였다. 하지만, 여전히 다수의 Bluetooth 통신이 필요하기에, Bluetooth Multiple Connection의 형태로 네트워크의 형태를 바꾸어 설계하였다.

4.1.2. 디지털 형태로의 구현 범위

본 과제의 최초 설계 당시에는 인체 전체를 디지털 형태로 모션 캡처할 계획이었다. 하지만, 하드웨어의 물량 부족으로 인한 인당 구매 제한과 하드웨어 구매에 대한 금전적 한계로 인하여 모션 캡처의 범위를 상반신, 그 중에서도 양팔로 제한하였다.

4.2. 연구 결과

아래 사진은 구현된 기기와 앱을 사용한 모습이다. 기기는 측정된 데이터를 무선 통신을 통해서 Gateway로 보내는 데에 사용되며, 앱은 Gateway로부터 받은 데이터를 시각화하는 역할을 한다.

[그림 14] 기기를 직접 착용한 모습

Bluetooth 통신을 통해서 받은 데이터를 Gateway에서 분류할 때의 모습이다. 타임스탬프와 여러 대의 센서 중에 어떤 센서에서 온 것인지를 확인하고, 그에 따라서 데이터를 분류한다. 이렇게 분류된 데이터를 Unity에서 해당하는 관절에 연동시킨다. 그 값에 따라서 아래와 같이 왼쪽 팔을 움직이면 앱에서의 모습 역시 왼쪽 팔이 움직이고, 오른쪽 팔을 움직이면 앱에서의 모습 역시 오른쪽 팔이 움직이며 디지털 형태로 인체의 움직임이 기록된다.

5. 결론 및 향후 연구 방향

5.1. 결론

연구에서는 총 6개의 장치를 이용하여 관성식 모션 캡처 시스템을 만들었고, 이를 Unity 3D를 통하여 디지털 형태로 시각화하고 기록했다. 시장에 상용화되고 있는 유료 모델들 대신 무료 Asset을 사용하였기에, 인간의 피부형태로 나타나지는 못 하며 매끄러운 형태로의 구현에는 미치지 못한다. 하지만, 그러한 구현들이 기존의 고가의 장비와 모델을 통해서 나왔으며, 본 연구에 사용된 장비가 저가의 장비임을 감안하면 만족스러운 결과가 나왔다고 판단한다.

관성식 모션 캡처라는 점에 있어서 광학식 모션 캡처에 비하면 확실한 한계가 존재한다.

5.2. 향후 연구 방향

현재 우리가 연구에서 사용한 장치는 총 6개였다. 하지만, 이는 상반신만을 구현할 때의 장치 개수였다. 만약 전신을 구현할 경우엔 추가적인 장비가 최소 9개는 더 필요할 것이라고 판단된다. 이때에도 통신의 속도가 여전히 지연없이 가능한지에 대한 연구가 더 필요하다.

연결되는 하드웨어가 늘어날수록 지연이 발생한다. 또한, 소켓을 내부로 연결하느냐, 외부로 연결하느냐에 따라서도 지연속도가 차이가 난다. 이를 개선할 방안을 찾는 것을 목표로 연구를 더 진행할 수 있을 것으로 보인다.

모션 캡처 장비의 일상으로의 상용화를 목표로 하는 만큼, 소형의 저가 모델로의 구현을 목표로 삼아야할 것이다.

6. 개발 일정 및 역할 분담

6.1. 개발 일정

<table>
<tr><th colspan="3">5월</th><th colspan="4">6월</th><th colspan="4">7월</th><th colspan="5">8월</th><th colspan="4">9월</th></tr>
<tr><th>3주</th><th>4주</th><th>5주</th><th>1주</th><th>2주</th><th>3주</th><th>4주</th><th>1주</th><th>2주</th><th>3주</th><th>4주</th><th>1주</th><th>2주</th><th>3주</th><th>4주</th><th>5주</th><th>1주</th><th>2주</th><th>3주</th><th>4주</th></tr>
<tr><td colspan="3">센서 보드 스터디</td><td colspan="3"></td><td></td><td colspan="4"></td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="2"></td><td colspan="3">무선 통신 스터디</td><td></td><td colspan="4"></td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="3"></td><td colspan="3">Unity 3D 스터디</td><td colspan="4"></td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="4"></td><td colspan="2">스터디 정리</td><td colspan="4"></td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="4"></td><td colspan="4">보드 BLE 구성</td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="2"></td><td>중간 보고서</td><td colspan="5"></td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="4">Gateway 구성</td><td colspan="3"></td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="4"></td><td colspan="5">센서와 Gateway 통신</td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="4"></td><td></td><td colspan="4">Unity와 Gateway 통신</td><td colspan="2"></td></tr>
<tr><td colspan="6"></td><td colspan="4"></td><td colspan="5"></td><td colspan="3">최종 점검</td><td></td></tr>
<tr><td colspan="6"></td><td colspan="4"></td><td colspan="5"></td><td colspan="2"></td><td colspan="2">최종 보고서, 발표 준비</td></tr>
</table>

6.2. 역할 분담

이름	역할
박해미	● Gateway를 통한 데이터를 Unity 3D로 시각화 ● IMU센서 예제를 이용해 Terminal 및 Fusion 분석
박민수	● BLE를 이용한 센서와 Gateway의 통신 ● 소켓 통신을 통한 Gateway와 Unity 간의 통신
공동	● 스터디 진행 ● 보고서 작성 ● 발표 및 시연 준비

7. 참고 문헌

[1] J. No, and H. Ye, "A Study on the Performance of Home Embedded System Using a Wireless Mesh Network" *Journal of KIISE : Computer Systems and Theory*, Vol. 34, No. 10, pp. 323-328, Sep. 2007. (in Korean)

[2] Bluetooth Documentation, Available: https://www.bluetooth.com/ko-kr/

[3] Unity Documentation, Available: https://docs.unity3d.com/kr/2021.2/Manual/index.html

[4] FP-SNS-MOTENV1, Available: https://www.st.com/en/embedded-software/fp-sns-motenv1.html

[5] BlueSTSDK_Python, Available: https://github.com/STMicroelectronics/BlueSTSDK_Python