Final Project - Brick Break

Date: 2019.06.15

Team: C

Student ID: 21600372

Name: Yoonkyoung Song

1. Introduction and Purpose Project

Brick break 프로젝트는 생체신호를 사용하여 벽돌깨기게임을 수행하는 프로젝트이다. IMU, EMG, Bending sensor 중 두 개 이상의 센서를 사용하여 패들을 제어하고, 게임을 수행하는 데 있어 유리한 찬스를 만들어 사용하고자 한다. 센서를 아날로그 회로와 연결하고 ADC과정을 거쳐 디지털로 변환된 데이터를 사용하여 MATLAB을 통해 제어한다. 데이터가 노이즈로부터 받는 영향을 제거하기 위해 필터를 사용하고 사용자에 의한 개인적인 특성을 일반화하기 위해 calibration을 적용하여 벽돌깨기게임을 수행한다.

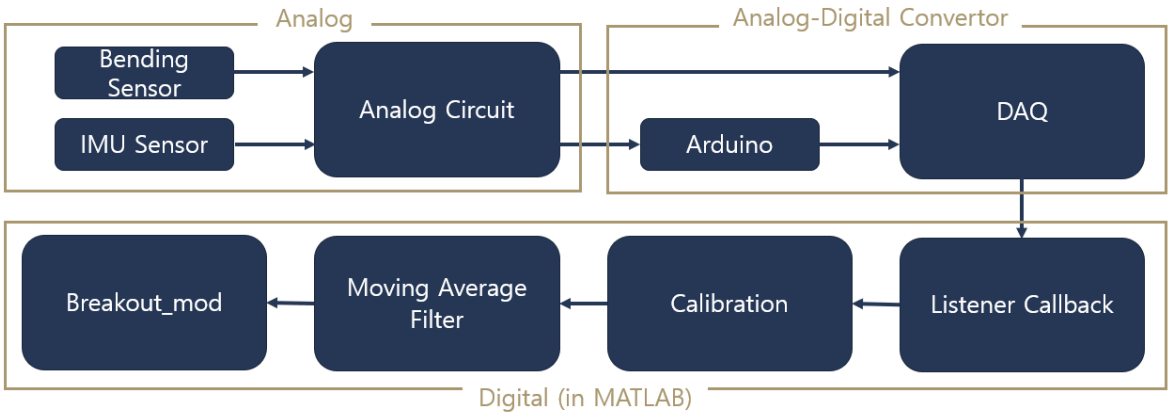


Figure 1. Overall Block Diagram of Brick Break System

2. Device

2-1. Software

Break Bricks는 paddle을 이용하여 벽돌을 부수는 게임이다. 게임의 상단에는 부수어야 하는 벽돌이 5줄에 걸쳐서 있고 패들로 공을 쳐서 벽돌을 부수는 방식으로 진행된다. 원래 게임은 마우스를 이용하여 패들을 움직이는 방식이었으나, 우리는 신체의 신호를 받아 패들을 움직이는 방식으로 게임을 재구성하였다. Break Bricks는 총 3가지의 matlab코드로 이루어진 시스템이다.

먼저 게임의 main인 break_out을 보면 게임의 진행을 파악할 수 있다. 게임을 처음 시작하면 게임의 창의 크기를 X=400, Y=300으로 설정한다. 이외 스코어, 점수, 목숨 등 게임의 화면에 나타나는 표시사항을 입력합니다. 처음 클릭을 하면 NewGame이라는 case를 시작한다. 게임에서 이용되는 ball은 X(가로)축과 Y(세로)축의 중간 지점에서 시작된다. Paddle은 가로는 X/6인 66.67, 세로는 Y/20인 15의 크기로 설정합니다.

패들은 bending sensor로 움직이는데, bending sensor의 신호는 voltage 값을 degree로 변환하고 normalization을 하여 paddle을 움직인다. 오른손의 검지와 약지에 센서를 부착하는데 시각적으로 인지하기 편하게 하기 위해서 왼쪽에 있는 검지를 굽히게 되면 paddle이 왼쪽으로, 오른쪽에 있는 약지를 굽히면 paddle이 오른쪽으로 움직이도록 설정하였다. 이때 손가락의 degree가 55도 이상 굽혔을 때 인지하도록 하였고, 한번 굽히면 X*0.04(=16) 만큼 가로축 방향으로 움직인다.

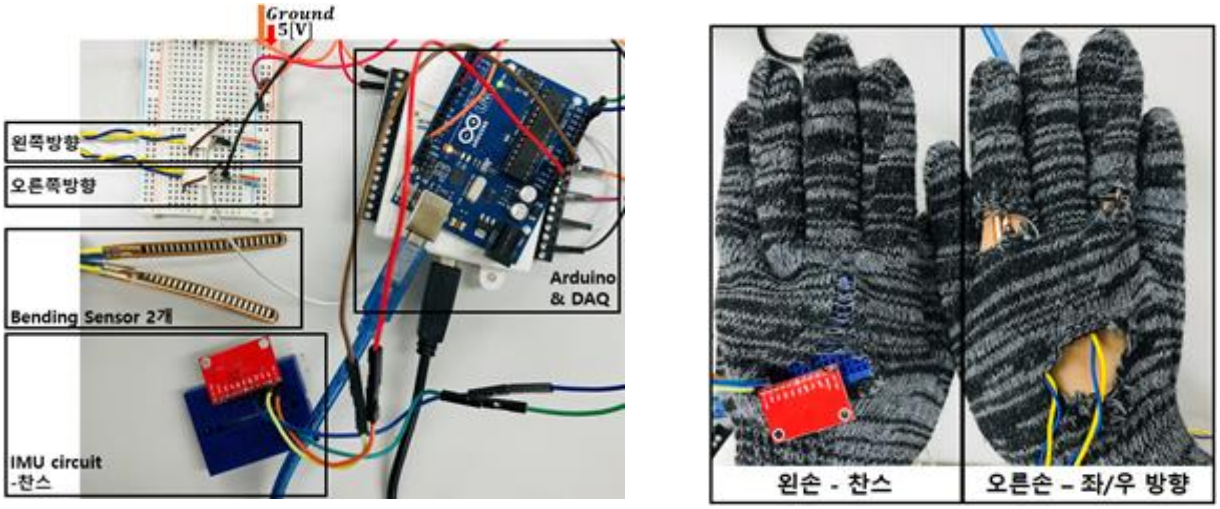
다음으로는 공의 위치에 따른 움직임의 변화를 보려 한다. 처음 시작하면 공은 세로 축으로 -5만큼 떨어지며 시작합니다. 이때 가로축으로 5만큼 움직이기 때문에 처음 공을 받을 때는 오른쪽에서 받게 된다. 공은 패들이나 벽, 또는 brick에 부딪치게 되면 반사각으로 움직인다. 패들과 brick은 X축의 반사각이므로 ballDX에 -1을 곱하여 방향을 바꿔주고 반대로 벽은 Y축의 반사각이므로 ballDY에 -1을 곱하여 방향을 바꿔준다.

패들에 공이 부딪치는 것을 인식하는 것은 게임을 진행하는데 가장 중요한 포인트이다. 공이 paddle의 높이에 있을 때, 패들의 위치안에 공이 있을 경우에는 공을 치고, 패들의 위치 안에 없을 경우에는 계속 진행시켜 땅에 닿게 한 다음 목숨이 줄어든다. Paddle의 위치는 paddlePos의 1번과 3번 열로 알 수 있습니다. 1번 열에는 패들의 왼쪽 끝 위치가 나와있고, 3번 열에는 패들의 길이가 나와있어서 오른쪽 끝은 1번과 3번 열의 합임을 알 수 있다. 패들의 왼쪽과 오른쪽의 숫자 안에 위치하면 부딪친 것이고 아니면 나간 것으로 간주한다.

다음은 점수의 표현이다. 천장을 기준으로 가까운 bricks부터 5,4,3,2,1점으로 점수가 매겨진다. 총 10개씩 50개의 bricks이 있어, 150점 만점으로 게임은 진행되고, 이때 맨 위의 벽돌을 부수고 천장에 닿는 경우는 paddle의 길이가 반으로 줄어서 난이도가 높아지게 된다.

2-2. Hardware

Bending sensor 2개, IMU 신호 중 pitch 1개를 사용해 생체신호를 받는다. 왼손은 찬스, 오른손 검지는 왼쪽 방향, 약지는 오른쪽 방향으로 설정하여 사용하였다.



(a) Actual circuit

(b) Sensor

Figure 2. Hardware

3. Sensor

3-1. 특성 설명 (Background knowledge of sensors)

- Bending sensor

bending sensor는 bending의 정도를 측정하고자 하는 표면에 부착되어 있으며, 센서 소자의 저항은 표면을 구부림에 따라 값이 변한다. 센서의 저항은 굽힘 정도에 비례하기 때문에 손가락을 완전히 굽히고 펼칠 때 사이의 값으로 사용된다.

- IMU sensor

IMU센서는 각 구성의 raw 데이터만 출력할 수 있으며, Roll, Pitch 및 Yaw 값은 반드시 계산하여 보정 필터를 통해 출력해야 한다. Roll, Pitch 값은 가속도 센서, 자이로 센서를 바탕으로 출력되고 Yaw 값은 자이로 센서, 자기장 센서를 바탕으로 출력이 된다. 이 세가지 값들을 계산해 3차원에서 회전각도에 대한 값들을 구할 수 있다.

- EMG sensor

EMG 신호는 골격근에서 방출되는 전기 신호로 이상적이지는 않지만, 운동 뉴런의 발화 속도를 합리적으로 반영했다고 볼 수 있다. EMG 신호를 얻는 종류에는 크게 2가지가 있습니다. 첫번째로, 바늘형태의 nEMG 방법입니다. 정확도는 높지만 고통이 크고 범위가 작으며 위생적인 문제가 발생한다. 두번째로, electrode를 표면에 붙이는 sEMG 방법입니다. nEMG의 단점을 보완하지만 정확도가 떨어지는 문제가 발생한다. 이렇게 어떠한 방법을 사용해도 문제가 있고 신호 자체가 비선형적 신호이기 때문에 선형적인 출력이 어렵다. 또한, 얻은 신호의 크기 및 주파수 특성은 피부 상태와 전극 부착 위치 같은 여러가지 요인과 함께 매우 민감하다.

3-2. 선정 이유 (Why did you apply this sensor to this command)

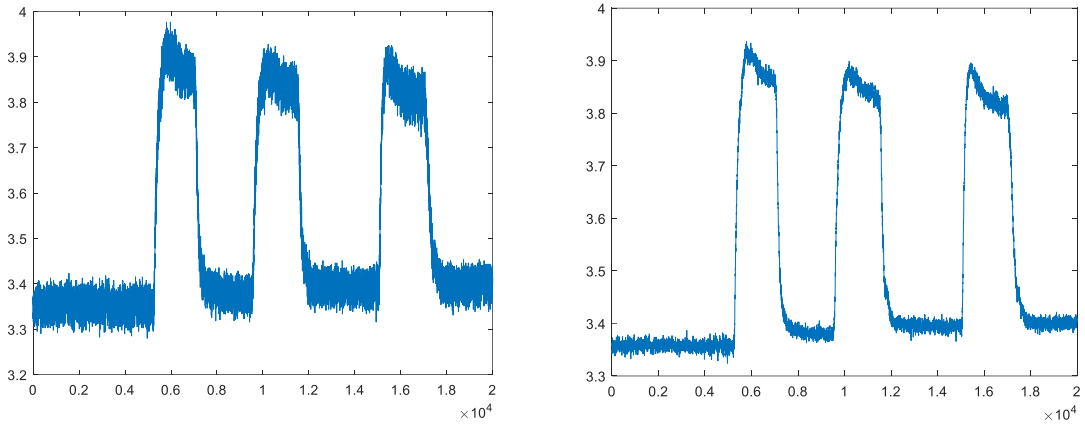
위의 3가지 sensor들을 가지고 다양한 조합으로 실험을 진행하였다. 패들의 움직임이 가장 민감한 센서였기 때문에 가장 선형성적으로 나오는 bend sensor로 측정한 값을 패들을 좌,우로 움직이는데 사용하였다. 또한, 찬스 같은 경우에는 기준만 잡아주면 되기 때문에 IMU, EMG 방법이 상관이 없었지만 EMG 같은 경우에는 피로누적이 심했기 때문에 IMU 찬스센서로 사용했습니다.

4. Filter

우리가 사용한 filter는 Moving average filter이다. Moving average filter는 입력신호로부터 몇 개의 데이터에 대해서 평균을 구해서 동작하는 filter이다. 즉, 오래된 데이터는 버리고 최근 값 만을 가지고 평균을 구하는 filter이다. 수식으로 표현해주면 다음과 같다.

$$\mathrm{y[i]} = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j]$$

이 식에서 x 가 input 이고, y 가 output, M 은 moving average 에서 평균을 구하기 위해 사용할 데이터의 개수이다. Moving average filter 는 random noise 의 크기를 줄일 수 있다는 장점이 있고, 오래된 데이터를 버리고 실시간으로 받는 데이터들을 계속해서 축적해서 그것들의 평균을 구하기 때문에 값이 계속해서 변하는 데이터들을 필터링하는데 장점이 있다고 판단하였다.



(a) Original Signal (b) After filtering Signal

Figure 3. Comparation of Using Filter

figure를 확인하면 Original signal에 비해 moving average filter를 적용한 signal의 노이즈가 확실히 줄었다는 것을 확인할 수 있다. 코드를 확인해보면 가장 오래 전에 입력 받은 데이터를 제거하고, 현재 받은 데이터를 추가한 뒤, 데이터들의 평균을 내는 방법을 사용하여 moving average filter를 Bending sensor의 데이터, IMU의 데이터에 적용하였다.

listener_callback 함수코드

```
function listener_callback(src,event)
    global ypr Pitch_value flt_data_1 flt_data_3 flt_data_2 avg_data_1 avg_data_2 avg_data_3

    ypr = event.Data;
    Bending_1 = mean(ypr(:,1));
    Bending_2 = mean(ypr(:,2));
    Pitch_value = (mean(ypr(:,3))*2*pi/5);

    % Moving Average Filter (size = 7)
    new_data = Bending_1;
    flt_data_1   = [flt_data_1(2:6) new_data];
    avg_data_1 = mean(flt_data_1);

    new_data = Bending_2;
    flt_data_2   = [flt_data_2(2:6) new_data];
    avg_data_2 = mean(flt_data_2);

    new_data = Pitch_value;
    flt_data_3   = [flt_data_3(2:6) new_data];
    avg_data_3 = mean(flt_data_3);

end
```

5. Calibration

총 2가지 센서를 이용하여 게임을 진행하였다. Bending sensor를 이용하여 paddle을 움직이고 IMU를 이용하여 2가지 종류의 찬스를 구현하고자 한다. 이 두가지 신호에 대한 calibration이 필요하다 판단이 되어 calibration 진행 후 게임을 실행한다.

5-1. Bending Sensor

손가락이 폈을 때와 구부렸을 때를 기준으로 Calibration을 진행하면 손가락을 폈을 때 bending sensor에서의 voltage 값을 open data로 받고, 손가락을 구부렸을 때를 close data로 받는다. Bending sensor은 구부리면 센서의 저항이 올라가 voltage가 상승한다. 그래서 close data를 max value로, open data를 min value로 볼 수 있다. 이 값으로 normalization을 하면 다음과 같다.

$$\text{normalization} = \frac{\text{recent data} - \text{open data}}{\text{close data} - \text{open data}}$$

손가락의 마디 중에서 손 끝을 기준으로 2번째 마디의 각도(PIP)는 약 100도 가량 움직이기 때문에 100을 곱하여 degree로 변환하였다. 이렇게 손가락의 움직임을 각도로 환산하여 1자유도 모델링한다.

$$\theta = \frac{100}{V_{close} - V_{open}} \times \left(V_{bend} - V_{open}\right)[\text{degree}]$$

5-2. IMU Sensor

IMU를 착용하고 있는 손의 기준으로 IMU가 위를 향할 때, 왼쪽을 향할 때, 오른쪽을 향할 때를 기준으로 calibration을 한다. 이 값들은 IMU sensor의 pitch value를 측정하여 손의 기울기를 측정한다. IMU센서는 찬스를 사용하는 센서로 설정을 하였고 찬스는 왼쪽으로 센서를 향하게 할 때, 오른쪽으로 향하게 할 때 사용하도록 만들었습니다. 각각 Pitch_middle, Pitch_left, Pitch_right로 pitch값을 받았다. Pitch는 왼쪽으로 기울어졌을 때 가장 작은 값을 갖고 middle, right로 갈수록 값이 커진다. 따라서, IMU의 값이 (Pitch_middle+Pitch_left)/2보다 작으면 왼쪽 찬스를 이용하고, (Pitch_middle+Pitch_right)/2보다 크면 오른쪽에 있는 찬스를 이용하도록 설정하였습니다.

5-3. Calibration을 하는 이유

이번 실험에서 Calibration은 꼭 필요한 과정이다. 우리는 사람 몸의 움직임을 신호로 받아 break bricks 게임을 한다. 하지만 사람마다 몸의 구조와 크기가 모두 다르기 때문에 개개인마다 받는 신호의 크기와 특징이 모두 다르다. 그래서 기준을 사용자의 maximum과 minimum으로 맞추어 주는 normalization을 하는 것이다. 그러면 신호에 따른 output이 사용자의 기준에 맞추어 제어할 수 있다.

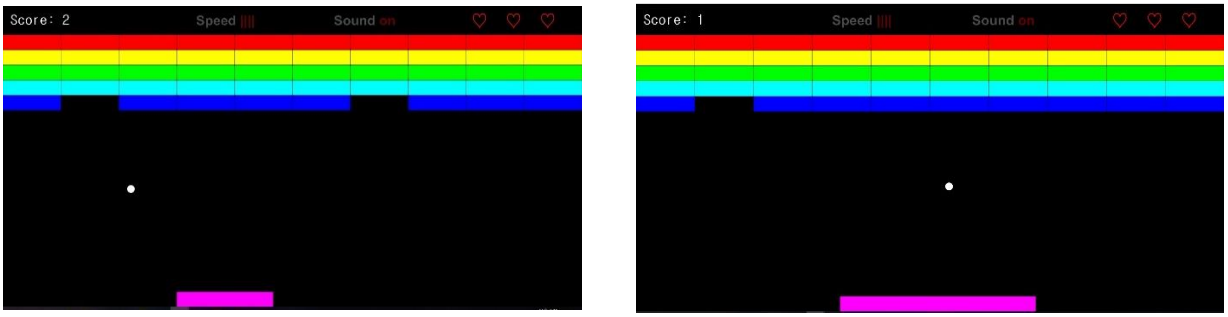
6. 찬스 사용방법 및 MATLAB 코드

찬스는 일정 시간동안 공의 속도가 줄어드는 찬스와 일정 시간동안 패들의 길이가 늘어나는 찬스 이 두가지 방식으로 진행하였다. 찬스는 각각 3 회씩 사용할 수 있으며 공속도는 5 초동안 패들의 길이는 3 초동안 유지되어야 한다. 찬스는 IMU 센서의 pitch value 를 사용하여 실행되며, IMU 센서를 왼쪽으로 기울이면 패들이 길어지고, 오른쪽으로 기울이면 공의 속도가 느려 진다. 두 찬스 코드는 break_mod 코드의 게임실행 반복문(while 문)안에 포함되어 있으며 변수 및 초기값 설정은 게임을 새로 시작할 때마다 다시 설정해준다.

pitch value 를 사용하여 현재 IMU 로부터 받아오는 값이 calibration 할 때 측정된 가운데 값과 각방향으로 최대로 기울였을 때의 값의 평균값보다 크면 실행되도록 코딩하였다. 찬스의 횟수를 세는 변수를 설정하여 3 번이상 실행되지 않도록 제한하였고, bending sensor 가 굽혀져 있는 순간 계속해서 들어오는 값으로 인한 중복실행을 방지하기 위해 찬스가 실행되는 동안에는 bending sensor 의 변화를 감지하는 조건문을 실행하지 않도록 설정하였다. 또한, 찬스가 실행되는 순간부터의 시간을 측정한 후, etime 함수로 현재시간과 차이를 비교하여 지정된 시간이 지나면 원래의 상태로 돌아오도록 설정하였다. 이때 현재시간과 찬스시작시간의 차이가 정수로 딱 떨어져 나오지 않기 때문에 범위로 설정하였다.

6-1. Paddle Chance

패들 찬스는 IMU 센서의 왼쪽방향을 사용하였다. 현재 IMU로부터 받아오는 값이 IMU의 가운데 값과 왼쪽으로 최대로 기울였을 때의 값의 평균값보다 크면 실행되고 현재시간과 찬스의 시작 시간이 3초가 되는 지점에서 돌아오도록 설정하였다. 패들 찬스가 실행되면 패들의 길이가 2배로 길어진다.



(a) Normal (b) Using Chance

Figure 4. Result of Using Paddle Chance

break_mod() 함수코드

```
% paddle chance cord 초기값 설정 (반복문 밖)

    t1=[2020 0 0 0 0 0];      %찬스가 실행되는 시간을 재는 변수 및 초기값
    speedchance=0;            %찬스가 실행되는 횟수를 세는 변수 및 초기값
    IMU1sign=0;               %찬스가 실행되는 중인지 확인하는 변수

% paddle chance cord (반복문 안)
  if paddlechance<3
        etime(clock,t1)
        if IMU1sign==0
            if cp < (data_Pitch_Left + data_Pitch_Middle)/2   % 기울임이 평균값보다 크면
                  IMU1sign=1;
                  paddlePos(3) = paddlePos(3)*2;      % 패들의 길이증가
                  t1=clock;
            end
        end
        if etime(clock,t1)>2.90 && etime(clock,t1)<3 % 3초가 되면
            paddlePos(3) = paddlePos(3)/2; % 원래길이로 돌아옴
            IMU1sign=0;
            t1=[2020 0 0 0 0 0];
            paddlechance=paddlechance+1;
        end
  end
```

6-2. Ball speed Chance

공 속도 찬스는 IMU 센서의 오른쪽방향을 사용하였다. IMU의 가운데 값과 오른쪽으로 최대로 기울였을 때의 값의 평균값보다 크면 실행되고 현재시간과 찬스의 시작 시간이 5초가 되는 지점에서 돌아오도록 설정하였다. 공 속도 찬스는 사진으로 확인이 불가능하여 결과를 첨부하지 않았다.

break_mod() 함수코드

```
% ball speed cord 초기값 설정 (반복문 밖)

  t2=[2020 0 0 0 0 0];   %찬스가 실행되는 시간을 재는 변수
  speedchance=0;         %찬스가 실행되는 횟수를 세는 변수
  IMU2sign=0;            %찬스가 실행되는 중인지 확인하는 변수

% ball speed cord (반복문 안)
  if speedchance<3
      if IMU2sign==0
          if cp > (data_Pitch_Right + data_Pitch_Middle)/2   % 기울임이 평균값보다 크면
              IMU2sign=1;
              ballDX =   ballDX/2;    % 공의 x방향 속도감소
              ballDY =   ballDY/2;    % 공의 y방향 속도감소
              t2=clock;
          end
      end

      if etime(clock,t2)>4.95 && etime(clock,t2)<5    % 5초가 되면
          ballDX = ballDX*2;      % 원래속도로 돌아옴
          ballDY = ballDY*2;      % 원래속도로 돌아옴
          IMU2sign=0;
          t2=[2020 0 0 0 0 0];
          speedchance=speedchance+1;
      end
  end
```

7. 논의 및 결론

Brick brake 프로젝트를 통해 생체 신호를 통하여 게임을 제어하였다. 생체신호를 사용하는 것은 다양한 논의점이 발생한다. 어떤 센서를, 어느 신체부위를 사용하여 생체신호를 받을 것인지, 비선형적 특성을 가진 생체신호를 어떻게 처리하여 원하는 신호 얻어낼 것인지에 대한 논의가 필요하다.

패들을 제어할 때 고려한 사항은 어떤 방식으로 패들을 제어하는 것이 직관적으로 쉬울 것인가였다. 생체신호를 사용하는 가장 큰 이유는 물체를 제어하기 위해 또 다른 인터페이스를 사용하지 않는 것이다. 인터페이스가 하나 더 존재하게 되면 그 인터페이스를 학습하는데 시간이 소요된다. 따라서 생체신호를 사용하는 궁극적인 목표는 신체에서 주어지는 신호를 통해 직접적으로 물체의 제어하는 것이다. 그렇기 때문에 이 논의점이 이번 프로젝트에서 가장 중요한 고려사항이라 생각했고, 우리 스스로가 쉽게 제어할 수 있는 손가락의 구부림을 통해 패들을 제어하였다.

다른 고려사항은 생체신호의 처리이다. 우리는 신호를 선형화하여 물체를 제어하는데, 생체신호는 특히 비 선형적인 요소가 크게 작용하고 센서를 통한 측정 자체만으로도 많은 노이즈를 동반하기 때문에 신호의 처리과정이 반드시 필요하다. 이번 프로젝트에서는 디지털필터를 사용하였고 그 중 이동평균필터를 사용하여 노이즈를 제거하고자 하였다.