Smart Sensors and IoT Devices Final Project

Date: 2023.06.10
Student ID: 21900031
Name: 곽 진

1. Introduction

① 실험 개요 및 목표

본 실험은 스마트 센서와 IoT 디바이스 최종 프로젝트로 습득한 센서 계측 및 신호 처리 및 제어 지식을 통해 센서에 가하는 자극과 센서의 전압 관계에 대하여 파악하고 전압 값을 계측하여 자유롭게 동체를 원하는 위치로 움직이는 것이다. 본 프로젝트는 Flex sensor를 계측하여 전압을 Matlab 상의 동체 양쪽 바퀴의 각속도로 변환하여 동체를 맵 상에서 이동시킨다. 하지만 센서 계측에 있어 필연적으로 존재하는 불확실성(노이즈와 같이 실제 값과의 오차), 계측 오차 등이 있기 때문에 이를 줄이는 방안에 대하여 논의하고, 시뮬레이션 프로그램의 여러 계수를 조절하여 안정적인 동체의 운행이 가능하게 설계하는 것이 프로젝트 목표이다.

② 실험 방법

실험 구성품은 National Instruments(이하 NI)사의 NI USB-6009, Flex Sensor 2 개, $20k\Omega$ 저항 2 개로 구성되어 있다. 이를 회로로 표현하였을 때 다음과 같이 표현할 수 있다. Fig 1 의 전압공급원과 멀티미터는 실제 회로에서 NI 사 DAQ 로 구현한다. 본 프로젝트에 가용하는 프로그램은 Matlab 으로 Addon 은 Data Acquisition Toolbox 이 필요하다.

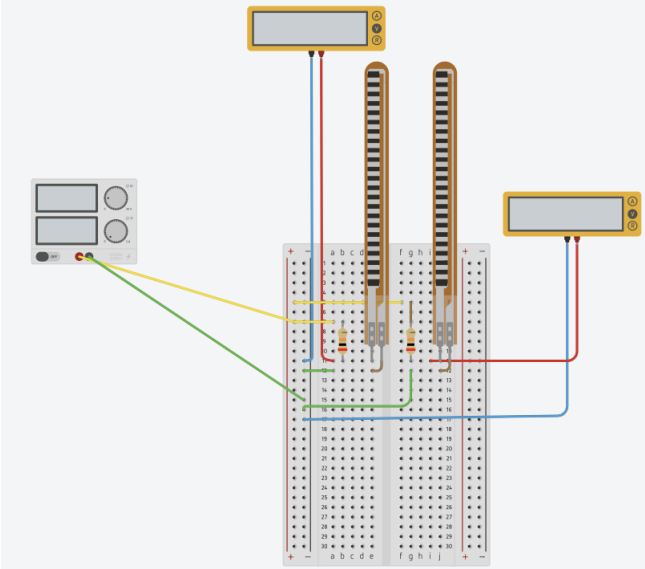

Figure 1. 프로젝트의 회로도(Tinkercad)

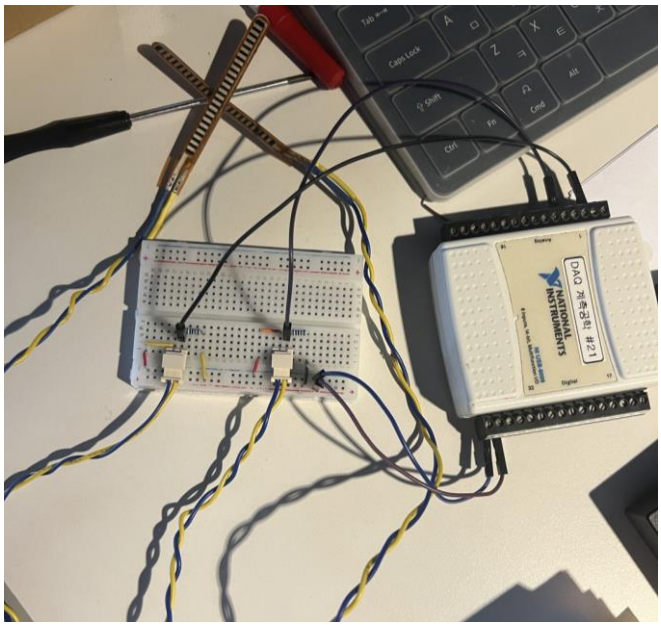

Figure 2. 프로젝트 실제 회로도

본 프로젝트의 Flex Sensor는 휘어지는 정도에 따라 저항 값이 변하는 가변 저항이다. Figure 3처럼 두 Flex Sensor의 전압을 DAQ를 통해 받아오고 두 전압 조건에 따라서 매트랩 프로그램으로 동체의 조향 명령을 달리하여 동체를 이동시켜 결승선까지 빠르고 정확하게 도착하는 것이 목표이다. 또한 계측된 전압 그대로 사용할 경우, 발생할 수 있는 오차 및 불확실성이 존재하기 때문에, 이에 대한 견실성을 확보하기 위해, 이동

평균 필터(moving average filter 이하 MAF), Median Filter 등을 적용해 안정성을 확보한다. 따라서 본 프로젝트에서 동체를 잘 제어하기 위해서는 매트랩 시뮬레이션의 실시간성과 적당한 안정성을 모두 확보해야 한다.

2. Main subject

2.1 Hardware

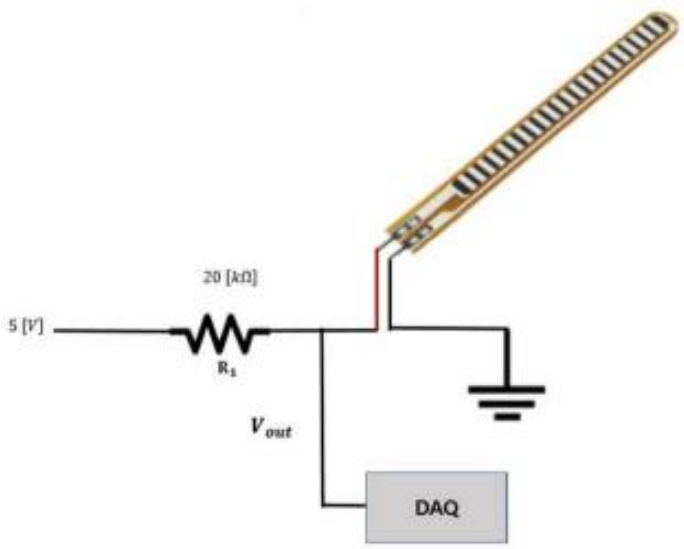

Figure 3. Flex Sensor 하나의 회로도

본 실험에서 사용하는 Flex Sensor는 가변 저항의 계측과 같은 방법을 사용한다. 센서는 20킬로옴저항과 직렬로 연결 되어있기 때문에 전압 분배 법칙을 따라 전압 값을 갖게 된다. 이를 식으로 표현하면 다음과 같다.

$$V_{flex} = V \times \frac{R_{flex}}{R_1 + R_{flex}} = 5 \times \frac{R_{flex}}{20 \times 10^3 + R_{flex}}$$

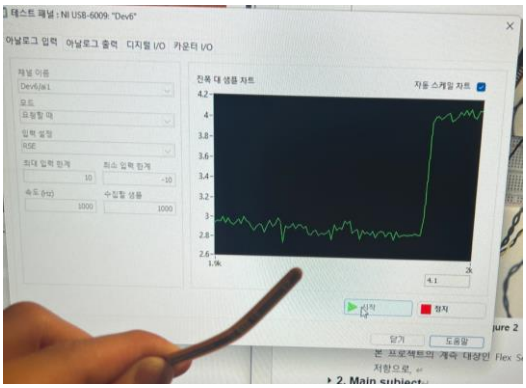

Figure 4. Flex Sensor를 굽힌 상태에서 폈을 때

Figure 5. Flex Sensor를 편 상태에서 굽혔을 때

Fig 4, Fig 5를 관찰한 결과 Flex sensor를 굽혔을 때 작은 전압이 걸리고, 폈을 때 큰 전압이 걸리는 것을 확인할 수 있다. 이는 굽힌 상태에서 저항이 적다는 것을 의미한다. 전압원 및 전압 관측은 컴퓨터에 연결된 DAQ로 실행하였다. 또한 Flex Sensor는 Fig 9처럼 손가락 관절에 장갑과 함께 결합되어 손가락을 구부림에 따라 센서에 자극이 가해지는 것으로 구현하였다.

2.2 Software

1. 기본 Software

시뮬레이션은 매트랩으로 진행하였다. Matlab Data Acquisition Toolbox 를 통해 DAQ 를 연동하여, 전압 계측 및 실험을 진행할 수 있다. 전체 시뮬레이션의 알고리즘은 다음과 같다. Program Initialization 상태에서 센서 전압 계측을 위한 DAQ 에 사용될 기본 변수 및 세팅을 활성화한다. 센서 계측 단계에서는 손을 폈을 때, 꽉 쥔 상태 10 초 간 1000 번의 전압 데이터를 받는다. 또한 각 상태의 데이터의 평균을 구하여 손의 Open 상태, Close 상태의 평균 전압에 대한 값을 얻는다. 다음 트랙 세팅 단계에서는 다른 Callback function 인 RunCarModel 을 활성화 시키며, 트랙이 끝나기 전까지 전압 값을 계속 받아오는 mydaq.IsContinuous 또한 활성화 시킨다. 또한 이 시점에 트랙을 선정하고 시작한다. 끝나면 자동으로 몇 번 벽에 부딪혔는지, 몇 초 만에 완주하였는지에 대한 정보를 반환한다.

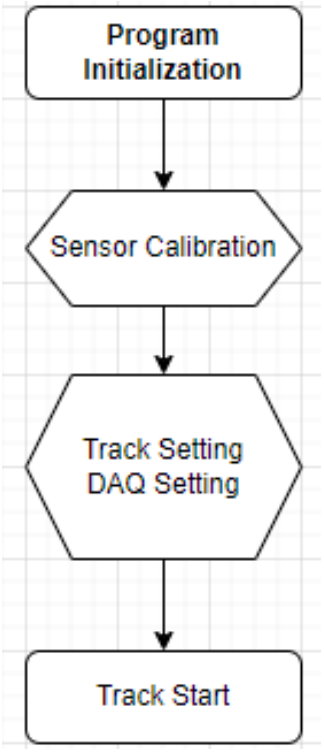

Figure 6. 전체 시뮬레이션 흐름도

2. 조향 명령 알고리즘

본 실험은 두개의 Flex Sensor의 전압 Threshold를 기준으로 4가지 조건에 따라 시뮬레이션 상의 조향을 바꾸는 것이다. 따라서 이 Threshold를 선정할 때 필요한 것이 센서 계측이다. Threshold선정은 10초간 최대로 Flex Sensor를 펴있는 상태(Open data)의 평균 값과 10초간 Flex Sensor를 구부린 상태(Close data)의 평균 값으로 그 중간 값으로 Threshold를 정의한다. 같은 센서라도 센서의 연식, 습도, 온도, 회로 상태 등에 따라 그 값이 상이할 수 있기 때문에 오른쪽, 왼쪽 센서 각각 평균 값을 측정한다. Fig7은 RunCarModel 함수를 통해 측정된 두 전압에 대해 그 값을 CmdCarModel 함수 내에서 Threshold와 비교하며, 그 비교한 값에 따른 조향 명령을 조건에 따라 반영하여 PlotCarModel 함수로 전달하는 것이다.

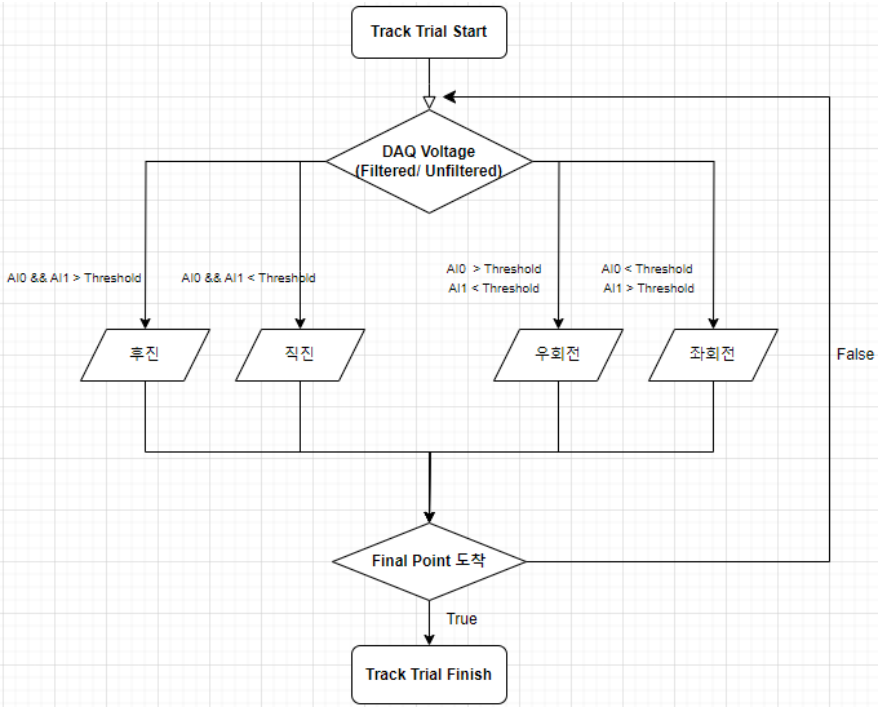

Figure 7. 측정된 전압에 따른 시뮬레이션 조향 명령

3. PlotCarModel 원리 및 알고리즘

PlotCarModel로 전달된 오른쪽, 왼쪽 각속도는 차량 속도로 변환되며, 그 방향 또한 각속도와 초기 위치에 의해 산출된다. PlotCarModel 함수에서 동체의 변하는 위치를 고정된 맵 위에서 실시간으로 표현하는 방식은 선형 변환(회전) 행렬과 속도 및 각도의 오일러 적분으로 설명할 수

있다. 먼저 선형 변환 행렬에 대하여 설명하려면 좌표계에 대한 정의가 필요하다. 동체의 움직임은 질점(particle)과 강체(rigid body)로 구분할 수 있지만, 왼쪽 오른쪽 바퀴의 각속도(X, Y축 각속도)를 통해 Z축을 기준으로 회전을 하기 때문에 본 프로젝트는 강체로 표현한다. 따라서 이를 동체 좌표계로 표현한다. 동체의 중심을 원점으로 원점의 코(횡) 방향(X_B)과 옆(종) 방향(Y_B)으로 표현하는 동체 좌표계가 있다. 이 좌표계에서 ω_L, ω_R 가 표현된다. 그리고 위치, 자세각 등이 정의되는 고정 좌표계(reference coordinate)가 있다. 두 직교 좌표계는 대표적인 선형변환(Linear Transformation) 행렬인 회전 행렬을 통하여 좌표계 통일을 할 수 있다. 이 행렬의 가장 큰 특징은 왜대칭행렬(skew-symmetric matrix)행렬로 역행렬이 전치행렬과 동일하며, Determinant의 크기가 1이라는 점이다. 이는 고유 벡터의 값이 바뀌지 않으며, 그 크기 또한 바뀌지 않는 것을 의미한다. 회전 행렬은 다음 식과 같다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

동체 좌표계에서 정의되었던 자세각 변화량(이하 Turn rate), 횡/종 방향 운동을 회전 행렬을 통하여 고정 좌표계 위에 표현한다. 또한 이로 산출되었던 값은 동체 중앙의 좌표이기 때문에, 이를 차량 동역학 조건에 맞추어 연산해준다.

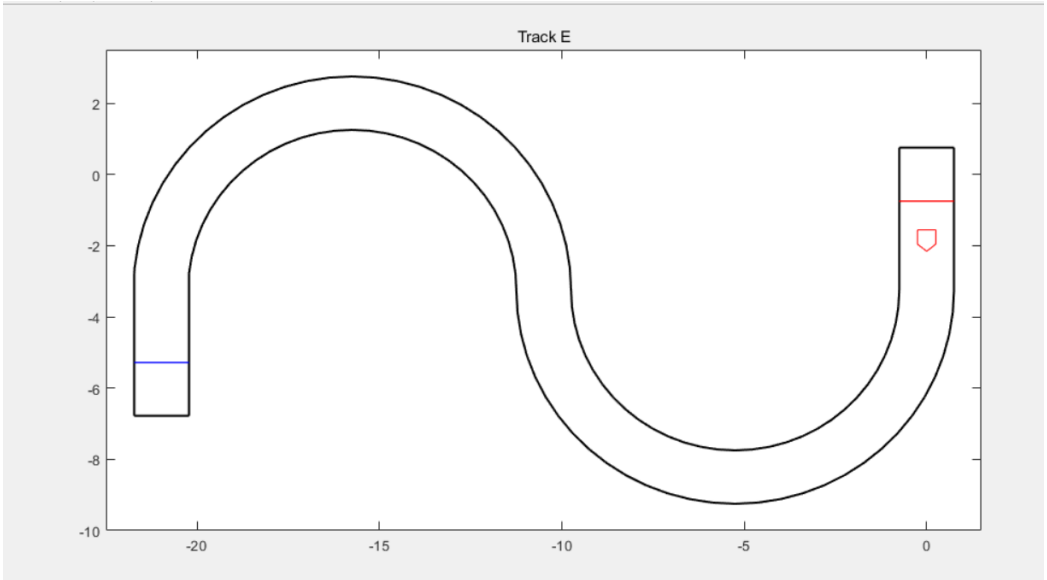

Figure 8. Matlab Track 'E'

디지털 상의 업데이트는 현재 위치, 자세각, 속도 정보가 있다면 다음 위치, 자세에 대하여 정의할 수 있다. 속도는 각 바퀴의 각속도의 평균과 동체 바퀴의 반지름으로 계산한다.

$$v = r\omega$$

자세 각은 시계방향을 기준으로 현재 자세각에 두 각속도를 적분하는 방식으로 계산할 수 있다.

$$\theta_{k+1} = \theta_k + \frac{r\omega}{w} \times dt$$

이 두가지는 속도(거리 변화량)의 크기와 방향에 대한 정보이기 때문에 가용정보를 이용하여 다음 위치를 X축 Y축에 대하여 표현할 수 있다.

$$x_{k+1} = x_k + v \times \sin(\theta)$$
$$y_{k+1} = y_k + v \times \cos(\theta)$$

다른 방법으로는 Runge-Kutta method 등이 있지만, 연산량이 더 적은 오일러 수치적분 기법을 사용하여 간단한 1차 미분방정식에 대한 다음 위치를 계산할 수 있다.

4. Filter

본 프로젝트의 조향 명령은 전압 값에 비례하지 않고, 역치 값(이하 Threshold) 이상 또는 이하를 판별하여 고정된 조향 명령(유도 이득)을 내는 알고리즘이기 때문에, 엄밀한 정확성이 필요하지는 않다. 따라서 Callback function(RunCarModel.m)이 한번 재생될 때마다 저장된 event.Data의 평균을 취하고, 평균 값으로 조향 명령을 수행하였다. 이를 검증하기 위해 Flex Sensor를 굽혔을 때, 폈을 때를 10초동안 1000개의 데이터로 받아서 평균과 표준 편차에 대해 나타내 보았다.

Table 1. Status에 따른 평균 전압 및 표준편차

Status	Standard deviation		Average Voltage [V]	
	왼쪽	오른쪽	왼쪽	오른쪽
Open	0.0073	0.0075	4.396	3.599
Close	0.0569	0.0375	3.859	2.976

역치 값은 Open, Close 상태에서의 평균 값의 중간 값인 4.128V과 3.288V인데 Open, Close 상태에서 표준편차가 크지 않기 때문에 역치 값을 넘어서 제어 알고리즘에 이상이 생길 확률은 적다고 판단할 수 있다. 다만 주먹을 꽉 쥐고 있는 상태는 직접 손가락을 의도해서 구부려야 하는데, 이 때 힘이 풀리거나 힘을 더 주는 상황으로 인하여 표준 편차가 더 크게 나오는 것을 확인할 수 있지만 유효한 값은 아니었다. 평균 값을 취하기 전의 Raw data 1000개의 데이터(오른쪽, 왼쪽, Open, Close 상태 총 4000개의 데이터) 중에 이런 역치 값을 넘는 데이터는 없었다. 표준 편차와 평균을 통해 정규 분포로 계산하였을 때에 마찬가지로 역치 값을 넘지 않는 것을 확인할 수 있었다.

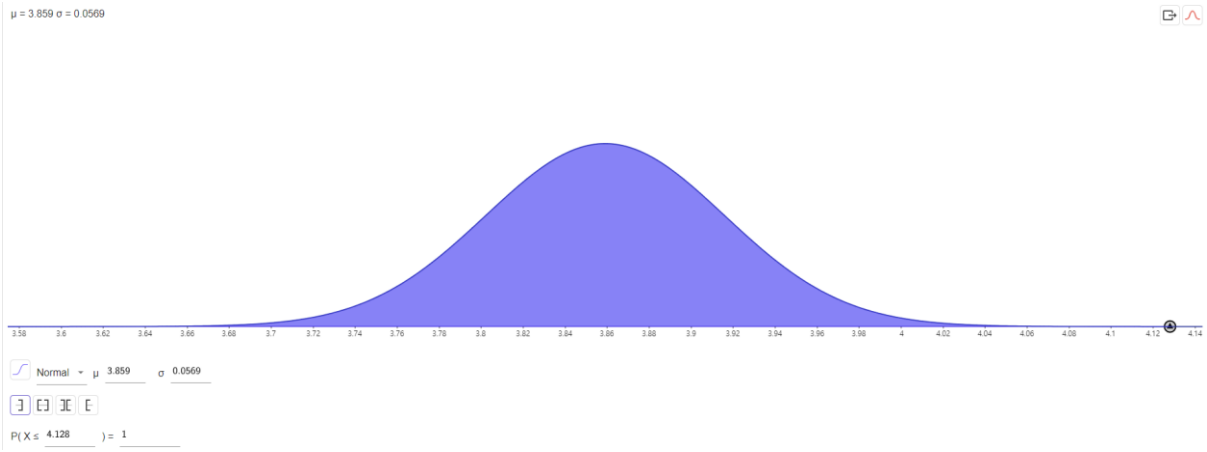

Figure 9. Close data의 평균과 표준편차로 평균값을 넘지 않을 확률 계산

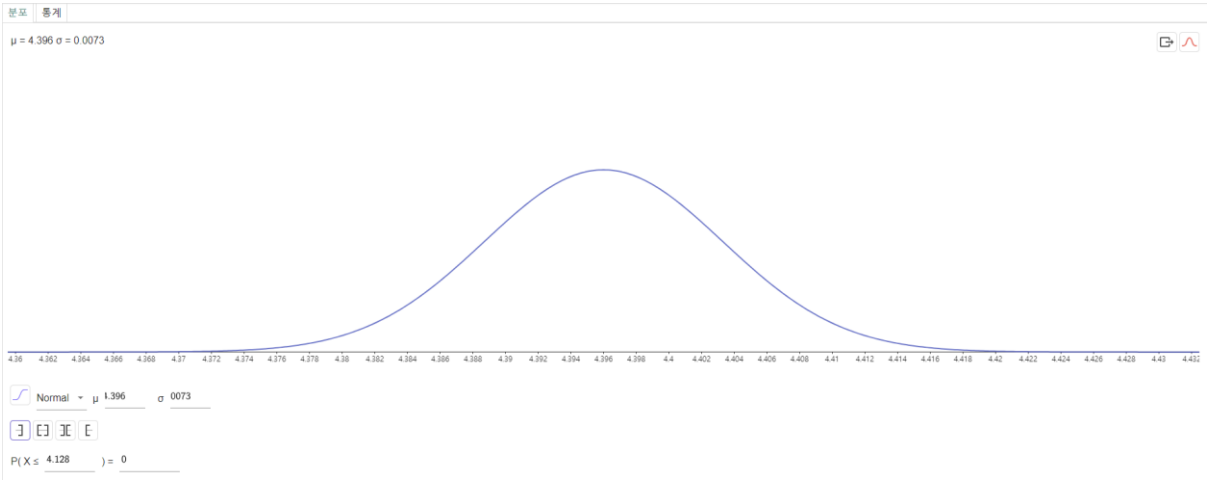

Figure 10. Open data의 평균과 표준편차로 평균 값을 넘을 확률 계산

2.3 Experimental results

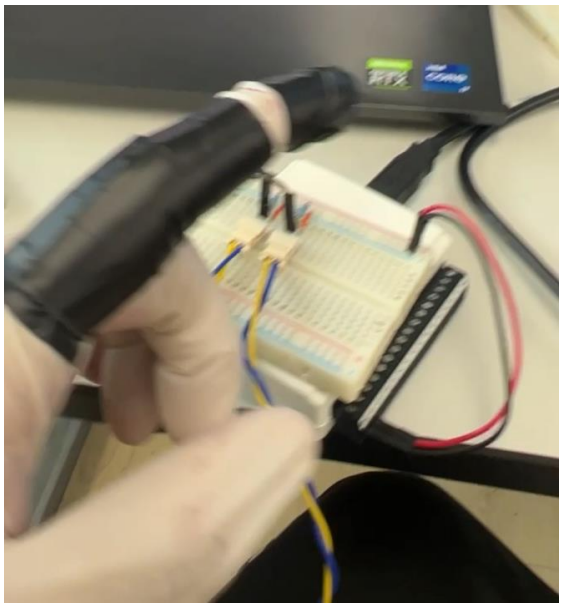

Figure 11. Flex Sensor 손가락 관절 부착

전 챕터에서 기술하였던 알고리즘으로 주행하였을 때, 컴퓨터 포트에 일정한 전압이 되지 않거나, Breadboard, 점프선 결함과 같은 하드웨어로 인해 전압 저하가 된 상황을 제외하고 제어 성능 저하는 확인되지 않았다.

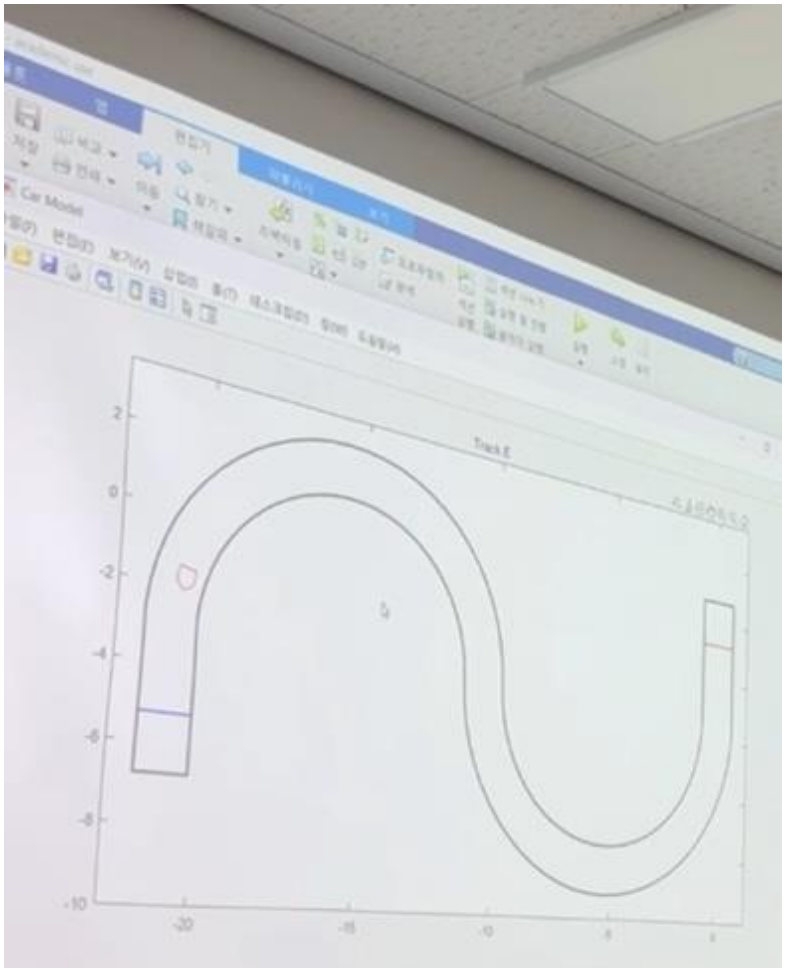

Figure 12. 본 알고리즘으로 제어하는 주행 시뮬레이션

3. Discussions

본 프로젝트를 진행하였을 때 계측을 잘 시행하였으나, 빈번하게 시뮬레이션을 돌릴 때 전압 값을 제대로 인지하지 못하여 실패하였었다. 이는 Flex Sensor의 전선이 길고, 전선을 구부려 움직이기 때문에 이로 인한 센서와 회로 연결이 헐거워지는 현상을 문제점으로 파악하였다. 이에 대한 보완점으로 PCB 기판을 사용하거나 절연 테이프로 회로에 고정되는 방법을 제안할 수 있다.

만약 조향각 명령이 전압 값에 선형적으로 비례하게 입력된다면, 더 엄밀한 전압 값 보정을 통한 각속도 제어가 필요하다. 즉, 전압의 외란 및 불확실성에 대한 안정성 확보가 되어야 한다. 하지만 기준 Sampling time은 0.001초로 매순간 전압 값을 측정하는데, Matlab의 내장함수 특성상,

연산량이 많아서 지연이 생기게 되고 이럴 경우에 상대 안정도 관점의 위상 여유(Phase margin)에 문제가 생길 수 있기 때문에 지양된다. Batch 식의 Moving average filter(이하 MAF)로 실험하였는데, 연산량이 늘어나게 되며, 여러 반복문 및 조건문이 있기 때문에, 이에 대한 안정성을 확보할 수 없어 정상적인 주행이 불가했다. 다만 이를 C 언어 환경에서 구현하여 연산 속도 및 실시간성을 확보하고 Batch 식을 Recursive 방식으로 표현하여 MAF 연산량을 줄여 이런 불가피한 문제를 해결할 수 있다. 다른 해결 방안으로는 Sampling time을 천천히 조절하여 Phase에 대한 안정성을 확보할 수 있다. MAF를 통해 노이즈로 인하여 역치 값을 넘는 이상점(Outlier)에 대한 안정성 또한 확보될 수 있다. 하지만 0.05초 동안 받아온 50개의 데이터의 평균이 역치 값에 대해 유의미한 값을 갖지 않기 때문에 실시간 제어에 주안을 두고 실현하지 않았다.

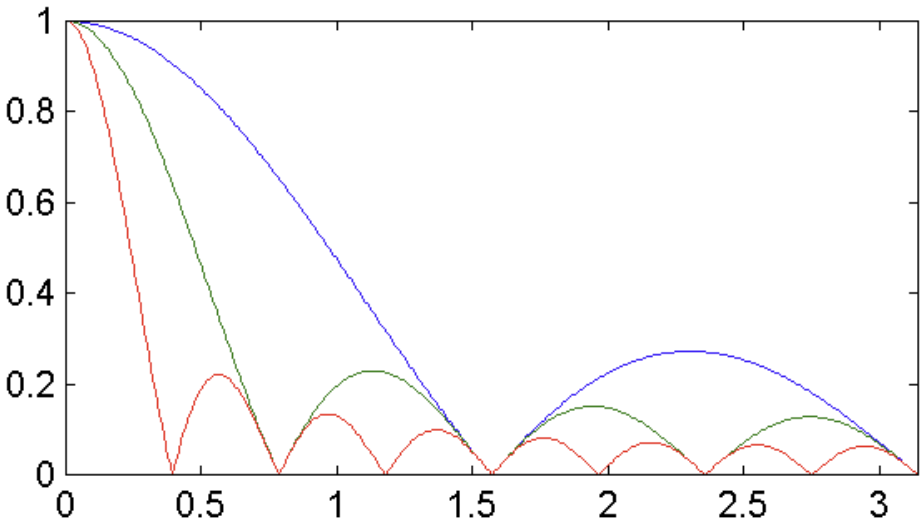

Figure 13. Window Size에 따른 L-sampling MAF frequency response

Fig 11처럼 MAF는 Window size에 관계없이 항상 LPF 성향을 갖기 때문에 급변하는 전압에 대한 응답 성능이 빠르지 않다. 이 또한 실시간 제어에 영향이 있다.

비교한 필터 중 Median filter보다 Moving average filter가 비교적 실시간성 확보에 유용한 것을 발견하였다. Median filter는 matlab 내장함수인 median을 사용하였다. 연산량 차이(실시간성)를 확인하기 위해 시간 복잡도(Time complexity)를 비교해 보았을 때 MAF는 $O(n)$, Median Filter는 소팅 알고리즘에 따라 최대 $O(n^2)$까지 그 복잡도가 증가한다. Sorting 시에 필요한 최소 반복문이 2중이며 그 안에 조건문이 필요하기 때문이다. 따라서 본 프로젝트와 같이 Sampling time이 작을

때는 그나마 MAF가 더 적합하다.

4. Conclusion

센서를 그대로 측정한 값만으로는 그 신뢰성과 정확성의 문제로 실용적으로 사용하기 어렵다. 본 계획은 이를 필터링하고 측정한 값을 보정하여 신뢰 가능한 데이터로 변환하는 계획을 수립하였으나 이미 50개의 데이터를 평균 값과 다음 50개의 평균 값에 대한 Moving average filter를 사용한다면 위상 지연으로 인하여 실시간성이 저하되는 단점이 있었다. 따라서 두 센서의 전압을 역치 값에 비교하여 직진, 후진 외 이분법적으로 조향 명령을 내리는 알고리즘에서 50개의 데이터의 평균으로만 조향 명령 알고리즘을 수립하였고 이에 대한 성능 저하가 관찰되지 않았다. 필터는 관찰하는 전압 값을 원하던 출력의 꼴로 안정하게 만드는 장점이 있지만, 안정성을 확보하기 위해서 연산량, 정밀성 등이 감소하게 되는 Tradeoff관계가 있다. 상황에 따라 어떤 필터를 쓸 것인지에 대하여 적절한 결정이 필요하다.

5. Appendix and matlab code

Moving average filter Batch expression:

$$\bar{x_n} = \frac{(x_{n-i+1} + x_{n-i+2} + \cdots + x_n)}{n}$$

Moving average filter recursive expression:

$$\bar{x_k} = \bar{x}_{k-1} + \frac{x_k - x_{k-n}}{n}$$

프로젝트에 사용된 매트랩 코드

Mcode.m

```
%% Program Initialization
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07    ***** %
clear all; close all; clc;
% Index finger modeling(initial setting)
% Initial setting before main code starts
global data_stack time_stack

mydaq = daq.createSession('ni');
mydaq.Rate = 100;
mydaq.DurationInSeconds=10.0;
```

```
mydaq.NotifyWhenDataAvailableExceeds = mydaq.Rate/20;
ch(1) = addAnalogInputChannel(mydaq,'Dev6',0,'Voltage');       %Left Flex Sensor
ch(2) = addAnalogInputChannel(mydaq,'Dev6',1,'Voltage');       %Right Flex Sensor
ch(1).Range = [-5 5];
ch(1).TerminalConfig = 'SingleEnded';
ch(2).Range = [-5 5];
ch(2).TerminalConfig = 'SingleEnded';
lh= addlistener(mydaq,'DataAvailable', @listener_callback_final);
%% Sensor calibration
% Left and Right finger modeling(calibration)
% Smart Sensors and IoT devices 23-1
% Name/ID: Jin Kwak/21900031

global time_stack data_stack mean_L mean_R
f = 440; d = 1; fs = 44100; no = d*fs;
t = (1:no)/fs; y = sin(2*pi*f*t);
sound(y, fs)                                %buzz
disp('Calibration start')
disp('손을  펴세요')

pause(2);

startForeground(mydaq);
open_data = [mean(data_stack(:,1)) mean(data_stack(:,2))];                    % Open set data of
Left(:,1) and Right(:,2)
sound(y, fs)
disp('주먹을  쥐세요')
pause(2);
startForeground(mydaq);
data_stack = data_stack((mydaq.Rate*mydaq.DurationInSeconds)+1:end,:);        % Picking up
latest set(Close) of data
close_data = [mean(data_stack(:,1)) mean(data_stack(:,2))];                   % Mean calculation

sound(y, fs)
disp('Calibration finish')

stop(mydaq)
Left_data = [close_data(1,1) open_data(1,1)]      ;                           % Left Flex
Sensor Close Open
Right_data = [close_data(1,2) open_data(1,2)]     ;                           % Right Flex
Sensor Close Open
mean_L = (Left_data(1)+Left_data(2))/2              ;                         % Left Sensor
threshold
mean_R = (Right_data(1)+Right_data(2))/2            ;                         % Right
Sensor threshold

%%   Noise(Attenuation) and its stdev
 startForeground(mydaq);
 % Picking up the latest mydaq data
 data_stack = data_stack((mydaq.Rate*mydaq.DurationInSeconds)+1:end,:);
 % # of data
 leng=length(data_stack);
 % Mean
 meanData = mean(data_stack);
 Sumdata =zeros(1,2);
 Stdev     =zeros(1,2);
 for i= 1:leng
       Sumdata(1) = Sumdata(1) + (meanData(1) -data_stack(i,1))^2;          %Left   Sensor StDev
       Sumdata(2) = Sumdata(2) + (meanData(2) -data_stack(i,2))^2;          %Right Sensor StDev
 end
 %Standard Deviation of Each Voltages obtained from DAQ
 Stdev = sqrt(Sumdata/(leng-1))
%% Before track Start
%Before starting this main function
% Look at Left_data & Right_data
```

```
% to make sure close & open data varies
global V_L V_R time_stack mydaq                                                    % DAQ
global FLAG_START Rate_Plot TYPE_TRACK mean_L mean_R avgV_R avgV_L flag            % CarModel
medR & medL or avgV_R & avgV_L should be commented
fprintf('Program setting ')          ;
Rate_Plot = 20                       ;
mydaq = daq.createSession('ni')      ;
mydaq.Rate = 1000                    ;
mydaq.IsContinuous = 1               ;                                             % System is in
progress until clear/clc/close
mydaq.NotifyWhenDataAvailableExceeds = mydaq.Rate/Rate_Plot      ;                 % 50 data at once

ch0 = addAnalogInputChannel(mydaq, 'Dev6', 'ai0', 'Voltage')     ;                 %Left
ch1 = addAnalogInputChannel(mydaq, 'Dev6', 'ai1', 'Voltage')     ;                 %Right

ch0.Range = [-10.0 10.0];    ch0.TerminalConfig = 'SingleEnded'   ;
ch1.Range = [-10.0 10.0];    ch1.TerminalConfig = 'SingleEnded'   ;
lh = addlistener(mydaq, 'DataAvailable',@RunCaRrModel)           ;
fprintf('DONE\n\nREADY\n\n')                                     ;
%% Track Trial Start
% Available Track : A, B, C, D, E
TYPE_TRACK = 'C'                     ;
StartCarModel(mydaq,TYPE_TRACK) ;
%% Track Trial Finish
FinishCarModel(mydaq,TYPE_TRACK);
close all;
```

Listener_callback_final.m

```
%===============Smart Sensors and IoT device ===============%
%===============       Flex Sensor calibration    ===============%
% Author       : Jin Kwak
function listener_callback_final(src,event)
    global data_stack time_stack

    if isempty(data_stack)
        data_stack = event.Data;
        time_stack = event.TimeStamps;
    else
        data_stack = [data_stack; event.Data]; %Voltage
        time_stack = [time_stack; event.TimeStamps]; %Time
    end
end
```

RunCarModel.m

```
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07    ***** %
% ============================Jin Kwak changed on 2023/06/05============================ %
function RunCarModel(src,event)
    global V_L V_R plot_var mean_R mean_L

    V_L= mean(event.Data(:,1));              %Left Voltage is mean of 50 data set
    V_R= mean(event.Data(:,2));
    [w_L, w_R] = CmdCarModel(V_L,V_R);    %Right, Left Voltage is put in the CmdCarModel function and the
angular velocity of wheels of left/right vehicle is obtained
    PlotCarModel(w_L,w_R);                   %Movement of Vehicle is only dependent on the wheels' angular
velocity
```

FinishCarModel.m

```
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07    ***** %
function FinishCarModel(NAME_DAQ, TYPE_TRACK)
    stop(NAME_DAQ);
    fprintf('Track %c Finished\n\n', TYPE_TRACK);
```

```
    ShowLapTime(TYPE_TRACK);
end
```

CmdCarModel.m

```
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07    ***** %
function [w_L w_R] = CmdCarModel(V_in_L, V_in_R)
    % Converting Input Signal(Sensor) to Command Signal(Car)
    % Binary decision
    global mean_R mean_L

    if   V_in_L > mean_L && V_in_R > mean_R                    % Both Open data
        w_L = -3;
        w_R = -3;                                              % Reverse

    elseif V_in_L > mean_L && V_in_R < mean_R                  % Left Open Right Close
        w_L = 1;
        w_R = 5;                                               % Right turn
    elseif V_in_R > mean_R && V_in_L < mean_L                  % Right Open Left Close
        w_L = 5;                                               % Left Turn
        w_R = 1;
    else
        w_L = 8;                                               % Both Close
        w_R = 8;                                               % Go Straight
    end
end
```

PlotCarModel.m

```
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07    ***** %
function PlotCarModel(w_in_L, w_in_R)
    persistent r w l dt Track Line_S Line_F Line_R;
    persistent pi_p_k x_p_k y_p_k AxisLimit TextTitle;
    global FLAG_START FLAG_TIMER Rate_Plot TYPE_TRACK Fig;

    if(FLAG_START == 0)
    %-------------------- Setting ---------------------------------------
        % car info
        r = 0.2;            % 0.2[m] = 20[cm] : radius of both wheels
        w = 0.5;             % 0.5[m] = 50[cm] : width of the car
        l = 0.4;            % 0.4[m] = 40[cm] : length of the car

        % program info
        dt = 1/Rate_Plot;       % [sec]

    %-------------------- Calculation ----------------------------------
    % State Computation
        x_p0 = 0;                   % initial position of x_p
        y_p0 = 0;                   % initial position of y_p

        switch TYPE_TRACK           % initial steering angle of pi_p       %Initial attitude
            case 'A'
                pi_p0 = 0;
            case 'B'
                pi_p0 = pi/2;
            case 'C'
                pi_p0 = pi/2;
            case 'D'
                pi_p0 = pi/2;
            case 'E'
                pi_p0 = pi;
            otherwise
                pi_p0 = 0;
        end
```

```
        v_p_k = (w_in_L + w_in_R)*r/2;                  % Average Velocity depends on the angular velocity of
wheels
        pi_p_k = (w_in_L - w_in_R)*r*dt/w + pi_p0;    % Turn rate(Clockwise) calculation
        x_p_k = v_p_k*sin(pi_p_k)*dt + x_p0;            % x axis displacement update
        y_p_k = v_p_k*cos(pi_p_k)*dt + y_p0;            % y axis displacement update

        % Show Figure
        TrackFile = load(sprintf('TrackData_Track%c.mat',TYPE_TRACK));
        Track = TrackFile.TrackData;
        Line_S = TrackFile.LineData_Start;
        Line_F = TrackFile.LineData_Fin;
        if(TYPE_TRACK=='C'), Line_R = TrackFile.LineData_Ret; end
        Fig = figure('Name','Car Model','NumberTitle','off');
        set(Fig, 'OuterPosition', TrackFile.ValueOutPos);
        AxisLimit = TrackFile.ValueAxiLim;
        TextTitle = TrackFile.TextTitle;
        FLAG_START = 1;
    else
        v_p_k = (w_in_L + w_in_R)*r/2;                  % Average Velocity depends on the angular velocity
of wheels
        pi_p_k = (w_in_L - w_in_R)*r*dt/w + pi_p_k;     % Turn(Clockwise) calculation
        x_p_k = v_p_k*sin(pi_p_k)*dt + x_p_k;           % x axis displacement update
        y_p_k = v_p_k*cos(pi_p_k)*dt + y_p_k;           % y axis displacement update
    end
    SttVec = [ v_p_k; pi_p_k; x_p_k; y_p_k ];           %State-vector [velocity, attitude, x-position, y-position]

    % Orientation Computation
    pi_Rot = 2*pi - pi_p_k;
    RotMat = [ cos(pi_Rot) -sin(pi_Rot) ; sin(pi_Rot) cos(pi_Rot) ];    % Direction Cosine Matrix
    OrtVec = [ w/2 ; l/2 ];                                             % half-width half-length vector
    X_fr = RotMat * [    1 0 ; 0    1 ] * OrtVec + [ x_p_k ; y_p_k ];   % Front Right calculation
    X_fc = RotMat * [    0 0 ; 0    2 ] * OrtVec + [ x_p_k ; y_p_k ];   % Front Center calculation
    X_fl = RotMat * [ -1 0 ; 0    1 ] * OrtVec + [ x_p_k ; y_p_k ];     % Front Left calculation
    X_br = RotMat * [    1 0 ; 0 -1 ] * OrtVec + [ x_p_k ; y_p_k ];     % Back Right calculation
    X_bl = RotMat * [ -1 0 ; 0 -1 ] * OrtVec + [ x_p_k ; y_p_k ];     % Back Left calculation

    % Shape Assignment
    X_car = [ X_fr(1) X_fc(1) X_fl(1) X_bl(1) X_br(1)    X_fr(1) ];
    Y_car = [ X_fr(2) X_fc(2) X_fl(2) X_bl(2) X_br(2)    X_fr(2) ];

    %-------------------- Checking --------------------------------------
    % Line Crossing Detection
    [SttVec, X_car, Y_car, FLAG] = CheckTrackCrossing(SttVec, X_car, Y_car);
    pi_p_k = SttVec(2);    x_p_k = SttVec(3);    y_p_k = SttVec(4);

    %------------------- Plotting ---------------------------------------
    plot(Track(:,1),Track(:,2),'k','linewidth',1.5);
    hold on;
    plot(Line_S(:,1),Line_S(:,2),'r','linewidth',1);
    plot(Line_F(:,1),Line_F(:,2),'b','linewidth',1);
    if(TYPE_TRACK=='C'), plot(Line_R(:,1),Line_R(:,2),'g--','linewidth',1); end
    plot(X_car, Y_car, 'r');
    title(TextTitle); axis(AxisLimit);
    hold off;
    drawnow

    %-------------------- Checking --------------------------------------
    % Start & Finish Detection
    if(FLAG_TIMER==0), CheckDriveStart(X_car, Y_car, Line_S);
    else,                    CheckDriveFinish(X_car, Y_car, Line_F); end

end
```

moving_average_filter.m

```
global V_L V_R time_stack plot_var mean_R mean_L avgV_L avgV_R flag n
   if flag==0                                          %If isempty(V_L)
      for i= 1 : n-1
            avgV_L(i,1) = event.Data(i,1);                              % filtered voltage before window size
            avgV_R(i,1) = event.Data(i,2);
            [w_L, w_R] = CmdCarModel(avgV_L(i,1), avgV_R(i,1));
            PlotCarModel(w_L, w_R);
      end
      for i= n : length(event.Data)
            avgV_L(i,1) = mean(V_L(i-n+1:i)) ;                      % After window size, average filter is adjusted
with batch expression
            avgV_R(i,1) = mean(V_R(i-n+1:i)) ;
            [w_L, w_R] = CmdCarModel(avgV_L(i), avgV_R(i)) ;        % f(v) = angular velocity of Vehicle's
left/right wheel
            PlotCarModel(w_L, w_R);                                 % Unite each coordinate into Fixed frame
and update vehicle position
      end

   else
         for i= length(V_L)-length(event.Data)+1:length(V_L)
               avgV_L(i,1)= mean(V_L(i-n+1:i,1)) ;
               avgV_R(i,1)= mean(V_R(i-n+1:i,1)) ;
               [w_L, w_R] = CmdCarModel(avgV_L(i,1), avgV_R(i,1));
               PlotCarModel(w_L, w_R);
         end
   end
```

median_filter.m

```
global V_L V_R time_stack plot_var mean_R mean_L flag medR medL n
len= length(event.Data);
      if flag == 0
            for i=1:n-1
                  medL(i,1) = event.Data(i,1);
                  medR(i,1) = event.Data(i,2);
                  [w_L, w_R] = CmdCarModel(medL(i,1), medR(i,1));
                  PlotCarModel(w_L, w_R);
            end
            for i=n:length(event.Data)
                medL(i,1)= median(event.Data((i-n+1):i,1));
                medR(i,1)= median(event.Data((i-n+1):i,2));
                [w_L, w_R] = CmdCarModel(medL(i,1), medR(i,1));
                PlotCarModel(w_L, w_R);
            end
      else
            tempL = V_L(length(V_L)-len-n+2:length(V_L)-len,1);
            tempR = V_R(length(V_R)-len-n+2:length(V_R)-len,2);
            for i= 1:n-1
                  temp_medL(i,1)= median(tempL);

                  temp_medR(i,1)= median(tempR);

                  tempL=[tempL(2:end,1); event.Data(i,1)];
                  tempR=[tempR(2:end,1); event.Data(i,2)];
                  [w_L, w_R] = CmdCarModel(temp_medL(i,1), temp_medR(i,1));
                  PlotCarModel(w_L, w_R);
            end
            for i = n:len
```

```
            temp_medL(i,1) = median(event.Data(i-n+1:i,1));
            temp_medR(i,1) = median(event.Data(i-n+1:i,2));
           [w_L, w_R] = CmdCarModel(temp_medL(i,1), temp_medR(i,1));
            PlotCarModel(w_L, w_R);
        end
        medL= [medL; temp_medL];
        medR= [medR; temp_medR];
    end
```

6. Reference

[1] https://www.newtc.co.kr/index.php

[2] https://ptolemy.berkeley.edu/eecs20/week12/freqResponseRA.html