Final – Car Model

Date : 2019.06.15
Team : C
Student ID : 21400600
21500401
21500726
21600372
Name : Lee Ji Hun
Eom Hyun Sang
Choi Dae Yeol
Song Yoon Kyoung

1. Project 설명 및 목표

원하는 센서를 사용해서 두 개의 바퀴를 조작을 해 5 개의 트랙을 각각 완주하는 것이 목적입니다. 이때, 센서는 반드시 신체에 부착 또는 착용하여 신체 움직임 정보를 반영해야 합니다. 또한, 신속성과 정확성에 제약이 있기 때문에 두 조건을 모두 만족시킬 수 있도록 주행해야 합니다.

2. Overall System

2-1. Introducing the full system

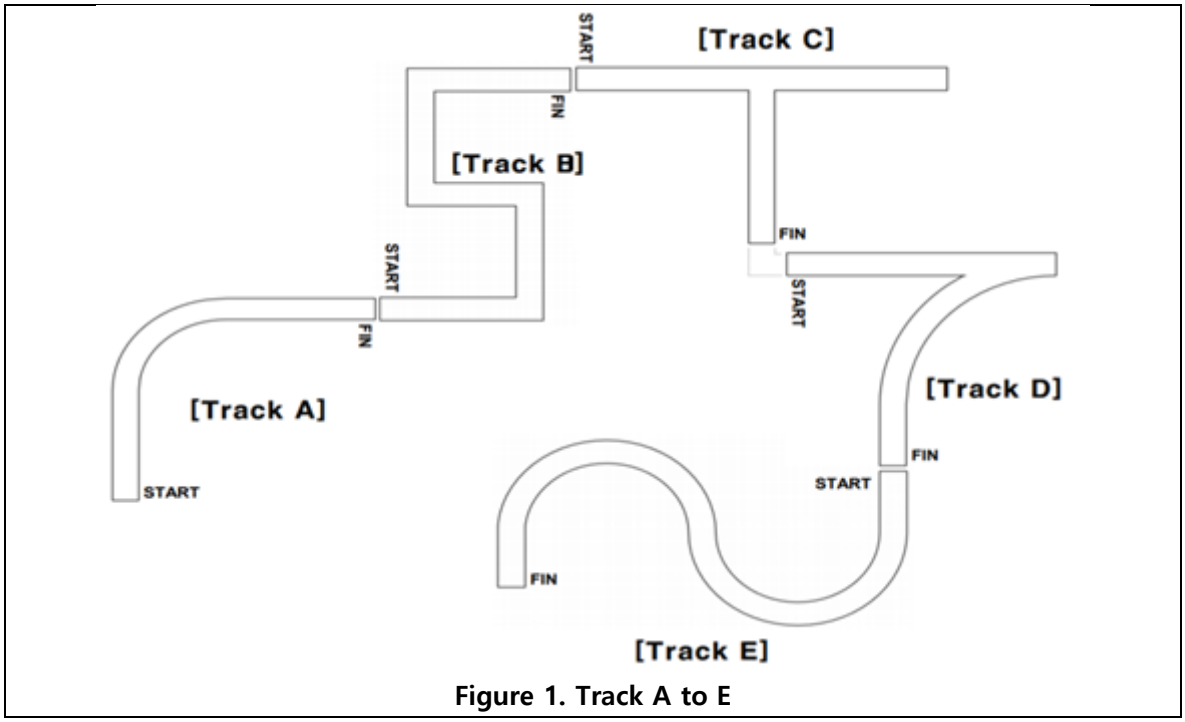

Figure 1. Track A to E

각각의 A,B,C,D,E Track 을 적절한 센서를 사용해 주행해서 완주하면 됩니다. 이 때, 트랙을 빠르게 완주하기 위해 차량을 어떻게 조절할지 신속성을 고려해야 하고, 트랙을 벗어나지 않고 얼마나 잘 제어하는지 정확성을 고려해야합니다.

2-2. Track driving strategy

자동차의 움직임은 총 3 가지 움직임으로 나눌 수 있는데, 왼 바퀴, 오른 바퀴의 움직임과 후진입니다. 우리 조는 모두 Bending sensor 로 움직였는데 이는 bending 센서가 가장 정확하고 움직임을 조절하기 쉽다고 판단했기 때문입니다.

Track A

A 코스는 직진성과 방향의 조절을 판단하는 코스입니다. 직진성에서는 자동차를 벽에 부딪치지 않고 정확히 앞으로 갈 수 있는지에 대해 측정하는 것입니다. 자동차 운전자가 운전의 기초인 직진에 능숙하게 할 수 있어야 다른 코스도 주행할 수 있기에 A 코스의 직진 구간에서는 실수를 하지 않는 것이 중요합니다. 다음으로 방향조절입니다. 완만한 90 도 회전이기 때문에 자동차의 방향을 조금씩 틀어도 충분히 움직일 수 있는 정도의 거리입니다.

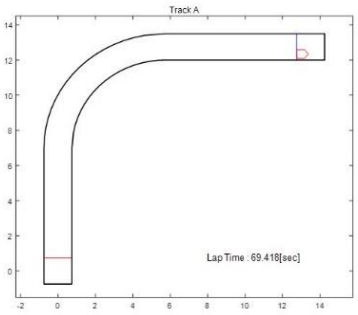

Figure 2. Track A

Track B

B 코스는 급격한 방향 회전에 대한 조절을 판단하는 코스입니다. A 코스와 다르게 급격하게 90 도를 회전하는 것은 자동차의 회전 작동 범위를 판단하는 것입니다. 실제 운전에서 좁은 주차장에 들어가게 되면 위와 같은 방향조절이 필요한 상황이 나옵니다. 첫째로 작은 차나 방향의 회전이 유리한 차의 경우 한번에 급격한 방향 전환이 가능하지만, 중형이상의 차이나 방향 전환이 좁은 각을 보이는 차량은 후진이라는 피드백을 통해서 조금씩 방향조절을 하는 것도 하나의 방법입니다. 저희 팀은 스피드에 중점을 두었기 때문에 후진을 통해서 조금씩 방향을 바꾸었습니다. 또한 대체적으로 이 코스에서 사람들이 방향전환에 시간을 쏟아 시간이 부족한 경우가 종종 발생하였습니다.

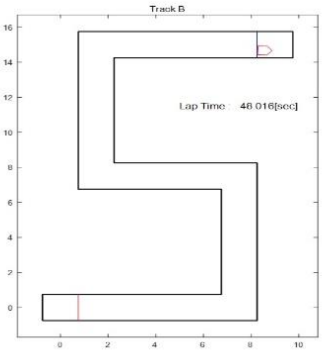

Figure 3. Track B

Track C

C 코스는 실제 시험에서 적용되는 T 자형 주차 코스입니다. 실제 시험에서와 동일하게 하되 전면, 후면 주차 모두 허용되어 사용자가 편한 방법으로 주차를 하도록 하였습니다. 위 코스에서는 후진이 얼마나 유연하게 되는지 그리고 후진 이후 방향조절에 대해 판단하였습니다. T 자에서 오른쪽 끝까지 차량을 이동한 후에 후진을 통한 중앙으로의 복귀 이후 오른쪽으로 90 도 틀어서 차량이 finish line 까지 가는 것을 목표로 하였습니다.

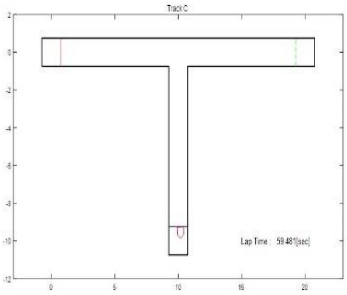

Figure 4. Track C

Track D

D 코스에서는 90 도 이상의 회전이 필요한 경우를 측정하였습니다. 일반 자동차의 경우 위와 같은 상황 시 start line 에서 직진해서 오른쪽 끝까지 가서 후진을 통해서 각도를 수정한 후에 방향을 트는 방법이 있습니다. 위와 같은 상황은 일반 주행상황에서는 나타나지 않으나, 협소한 골목길이나 시골길의 경우에 드물게 나타나는 코스입니다. 저희의 경우에는 조금씩 후진을 이용해서 방향을 수정하는 방법으로 했습니다.

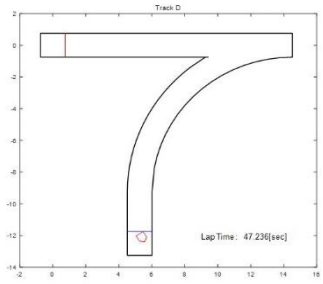

Figure 5. Track D

Track E

E 코스는 부드러운 회전을 요구하는 S 자 주행코스입니다. 실제 운전면허 시험장에서 쓰이는 코스이며 많은 초보 운전자가 애를 먹는 코스로 유명합니다. 위 코스의 특징으로는 조금씩 오랫동안 방향 조절을 해야 하기 때문에 운전자가 방향 조절에 미숙하다면 긴 거리의 방향조절에 애를 먹을 수도 있습니다. 우리 팀의 경우 여기에서 실수를 하였는데, 방향의 전환이 아니라 직진 구간에서 실수를 하여서 코스의 목표는 성공적으로 하였으나 기본이 부족한 모습을 보였습니다.

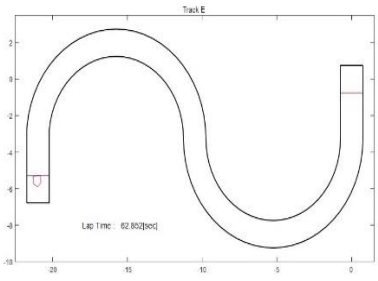

Figure 6. Track E

3. Hardware

3-1. Sensor

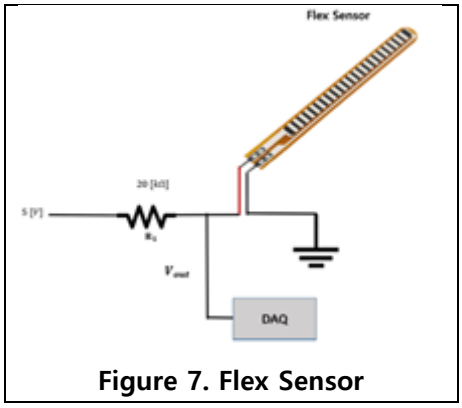

Figure 7. Flex Sensor

Flex 센서는 센서가 휘어짐에 따라서 저항 값이 변화됨으로 그 구부러짐을 측정할 수 있게 하는 센서입니다.

3-2. Circuit

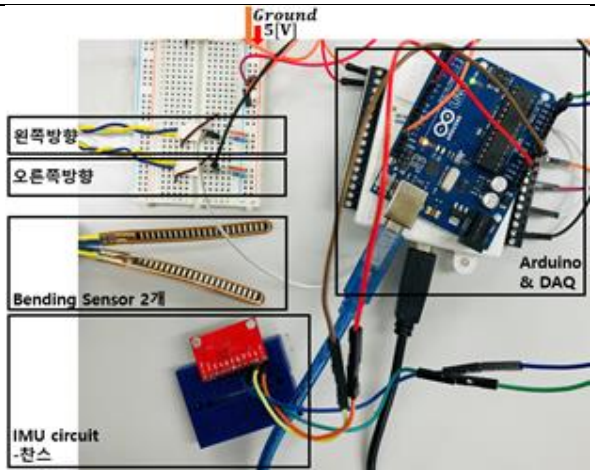

Figure 8. overall system

Figure 9. function of left & right hand

Bending sensor 3 개를 사용해 생체신호를 받았습니다. 왼쪽 손 검지는 후진방향, 오른쪽 손 검지는 왼쪽 방향, 약지는 오른쪽 방향으로 설정했습니다.

3-3. The way to use

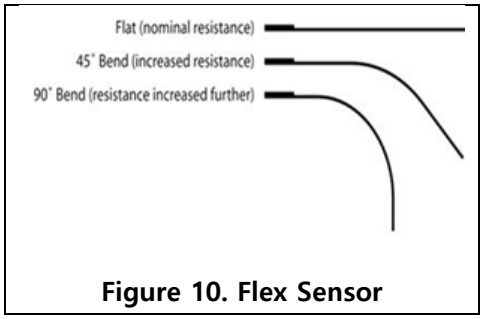

Figure 10. Flex Sensor

flex 센서는 구부러질수록 저항이 증가하는데, 그에 따라 인가되는 전압이 커집니다. 손가락의 굽혀짐에 따라 달라지는 전압을 이용해 일정값(우리의 경우 45 도에 해당하는 전압) 이상의 전압이 인가되면 그에 따라 바퀴의 각속도를 변화시켜주는 방법을 이용해 flex 센서를 사용하였습니다.

4. Software

4-1. Calibration method

Bending sensor 를 이용하여 손가락의 굽힘 정도를 측정하여 전압을 받으면 사람마다 그 굽힘의 정도가 달라 측정되는 전압이 달라진다. 따라서 사용자에 상관없이 시스템이 안정되게 하기위해 calibration 을 하였다. 사람이 주먹을 쥐었을 때와 손을 폈을 때의 전압을 각각 측정하면 bending sensor 로부터 각 사람이 받을 수 있는 최대전압과 최소전압이 측정된다. 이를 각도로 변환하고 1 차식으로 curve fitting 여 선형 일차식으로 모델링한다.

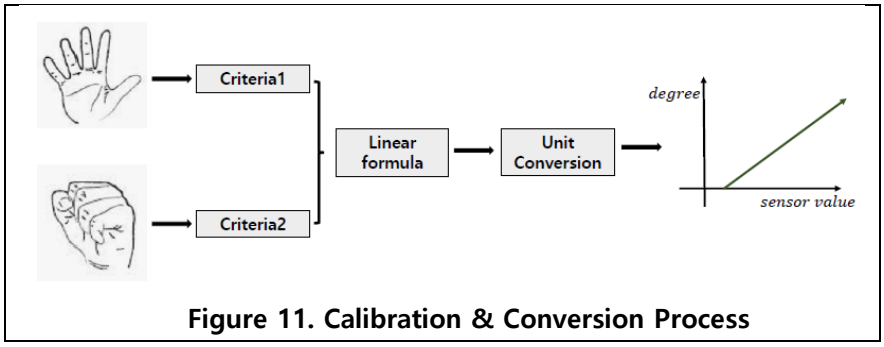

Figure 11. Calibration & Conversion Process

Bending sensor를 통해 들어온 주먹을 쥘 때와 펼 때의 전압(V_{close}, V_{open})을 먼저 측정한 후 1자유도 모델링으로 curve fitting하려 한다. 이번 프로젝트에서는 모델링의 정확성을 높이기 위해 V_{close}, V_{open}를 측정할 때 6개의 데이터의 평균값을 사용하였다. V_{close}가 인가되었을 때 PIP의 최대값을 β_{max}, V_{open}이 인가되었을 때 PIP의 최소값을 β_{min}이라 하면 2차평면에서 두 점의 좌표는 각각 (V_{close}, β_{max}), (V_{open}, β_{min})로 나타낼 수 있다. 두 좌표를 사용하여 기울기를 구한다. 따라서, 현재 bending sensor로 들어오는 전압을 V_{bend}, 손가락의 각도를 θ라 하면 다음과 같은 관계식이 도출된다.

$$\theta = \frac{\beta_{max} - \beta_{min}}{V_{close} - V_{open}} \times (V_{bend} - V_{open})[\text{degree}]$$

4-2. Driving algorithm

Basecode 에서 주어지는 차량 모델에 의해 차는 양쪽 바퀴의 각속도의 변화를 통해 차량의 움직임이 제어된다. 따라서, 우리는 3 개의 Bending sensor 를 이용해 각각의 센서로 오른쪽 바퀴의 각속도, 왼쪽 바퀴의 각속도, 그리고 방향을 세가지 자유도를 통해 자량을 제어하고자 한다. Matlab 에서 mcode 파일에 RunCarModel 함수코드로 daq 로부터 데이터를 불러오고, CmdCarModel 함수코드를 통해 차량의 움직임을 제어하였다.

각각의 bending sensor로부터 얻은 각도를 통해 제어하였는데, 조건문을 사용하여 bending sensor로부터 받은 각도가 최대각도에서 최소각도의 차에 2/3지점보다 커지면 명령을 수행하도록 코딩하였다. 기본적으로 차량은 2의 속도를 가지고 전진하도록 설정되어 있다. Bending sensor는 각각 Bending1 (첫번째 bending senor)는 왼쪽 바퀴, Bending2 (두번째 bending senor)는 오른쪽 바퀴, Bending3 (세번째 bending senor)는 후진을 의미하고 각각의 센서를 착용한 손가락이 손가락 굽힘을 수행하면 조건문의 명령을 수행한다. 따라서, Bending1이 굽혀지면 왼쪽 바퀴의 속도가 4로 빨라져 오른쪽으로 회전하고, Bending2가 굽혀지면 오른쪽 바퀴의 속도가 4로 빨라져 왼쪽으로 회전한다. Bending3이 굽혀지면 양쪽 바퀴가 방향이 바뀌며 후진하게 된다.

4-3. Matlab code (추가 및 변경)

- Mcode (including calibration code)

```
%% Program Initialization
instrreset; clear all; close all; clc;
fprintf('Program setting ');
global V_L V_R time_stack mydaq;              % DAQ
global FLAG_START Rate_Plot TYPE_TRACK;   % CarModel
global mydaq close_data1 close_data2 close_data3 open_data1 open_data2 open_data3 flt_data_1
flt_data_2 flt_data_3



Rate_Plot = 20;

mydaq = daq.createSession('ni');
mydaq.Rate = 1000;
mydaq.IsContinuous = 1;
mydaq.NotifyWhenDataAvailableExceeds = mydaq.Rate/Rate_Plot;

ch0 = addAnalogInputChannel(mydaq, 'Dev2', 'ai0', 'Voltage') ;
ch1 = addAnalogInputChannel(mydaq, 'Dev2', 'ai1', 'Voltage') ;
ch2 = addAnalogInputChannel(mydaq, 'Dev2', 'ai3', 'Voltage') ;

ch0.Range = [-10.0 10.0];     ch0.TerminalConfig = 'SingleEnded';
```

```
ch1.Range = [-10.0 10.0];     ch1.TerminalConfig = 'SingleEnded';
ch2.Range = [-10.0 10.0];     ch2.TerminalConfig = 'SingleEnded';

'finish'

%% calibration
f=440; d=1; fs=44100; n=d*fs;
t=(1:n)/fs; y=sin(2*pi*f*t);
sound(y,fs)
disp('calibration start')
disp('손을 펴세요')

pause(2)
data_bend1_calib=[];
data_bend2_calib=[];
data_bend3_calib=[];

for i=1:500
    data = inputSingleScan(mydaq);
    data_bend1_calib=[data_bend1_calib data(1)];
    data_bend2_calib=[data_bend2_calib data(2)];
    data_bend3_calib=[data_bend3_calib data(3)];
end

open_data1 = mean(data_bend1_calib);
open_data2 = mean(data_bend2_calib);
open_data3 = mean(data_bend3_calib);

sound(y,fs)
disp('주먹을 쥐세요')

pause(2)
data_bend1_calib=[];
data_bend2_calib=[];
data_bend3_calib=[];

for i=1:500
    data = inputSingleScan(mydaq);
    data_bend1_calib=[data_bend1_calib data(1)];
    data_bend2_calib=[data_bend2_calib data(2)];
    data_bend3_calib=[data_bend3_calib data(3)];
end
```

```
close_data1 = mean(data_bend1_calib);
close_data2 = mean(data_bend2_calib);
close_data3 = mean(data_bend3_calib);


[open_data1 open_data2 open_data3 close_data1 close_data2 close_data3]

%% Filter 초기값 지정 (6개단위)

ini_data = inputSingleScan(mydaq);

flt_data_1 = [];
for i=1:6
    data=inputSingleScan(mydaq);
    flt_data_1 = [flt_data_1 ini_data(1)]
end

flt_data_2 = [];
for i=1:6
    data=inputSingleScan(mydaq);
    flt_data_2 = [flt_data_2 ini_data(2)]
end

flt_data_3 = [];
for i=1:6
    data=inputSingleScan(mydaq);
    flt_data_3 = [flt_data_3 ini_data(3)]
end

%% 실시간 데이터 받기

lh = addlistener(mydaq, 'DataAvailable', @RunCarModel);
fprintf('DONE₩n₩nREADY₩n₩n');
load('TrackData_TrackA.mat');
load('TrackData_TrackB.mat');
load('TrackData_TrackC.mat');
load('TrackData_TrackD.mat');
load('TrackData_TrackE.mat');

%% Track Trial Start
% Available Track : A, B, C, D, E
TYPE_TRACK = 'E';
StartCarModel(mydaq,TYPE_TRACK);
```

```
%% Track Trial Finish
FinishCarModel(mydaq,TYPE_TRACK);
close all;
```

- RunCarModel

```
% ***** LAST DEVELOPER UPDATE : PWH 19.06.07   ***** %
function RunCarModel(src,event)

    global V_L V_R time_stack plot_var V_B

    V_L = event.Data(:,1);
    V_R = event.Data(:,2);
    V_B=event.Data(:,3);
    time_stack = event.TimeStamps ;

    [w_L w_R] = CmdCarModel(V_L(end), V_R(end),V_B(end));

    PlotCarModel(w_L, w_R);

End
```

- CmdCarModel

```
function [w_L w_R] = CmdCarModel(V_in_L, V_in_R, V_in_B)
    % Converting Input Signal(Sensor) to Command Signal(Car)
    global ypr open_data1 open_data2 open_data3 close_data1 close_data2 close_data3 flt_data_1
flt_data_2 flt_data_3 avg_data_1 avg_data_2 avg_data_3

    flt_data_1   = [flt_data_1(2:6) V_in_L];
    avg_data_1 = mean(flt_data_1)

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

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

    w_L = 2;
```

```
        w_R = 2;

        if(V_in_B < open_data3+(close_data3 - open_data3)/2)
            if(avg_data_1>(open_data1+close_data1)/2)
                w_L = 4;
            end

             if(avg_data_2>(open_data2+close_data2)/2)
                w_R = 4;
             end
        else
            w_L = -2;
            w_R = -2;
        end

    end
```

•

5. Conclusion

5-1. Concluding remark

저희 조는 움직임을 최소한 단순화 하여서 운전을 직관적으로 하려 했습니다. 또한 실전에 가면 손의 떨림과 차 움직임의 예상치 못한 움직임을 막기 위해서 threshold 로 차의 속도를 느리게와 빠르게 두 가지로 나누어서 주행하였습니다. 이는 calibration 에서 생기는 bending sensor 의 오류를 최소화 할 수 있었고 운전자가 가장 편하게 주행할 수 있다고 판단을 내린 방법이었습니다. 하지만 아쉬운 점은 각도의 조절이 한가지 각으로만 조절이 가능하고 속도를 높게 설정하다 보니 90 도를 회전하기 위해서는 후진을 통해서 각도를 조금씩 변경하는 방법으로 하였습니다. 물론 후진을 이용하여 각도를 조절하면 시간이 더 걸리는 단점이 있으나, 속도를 높게 잡아서 이를 보완했습니다. 실제로 차의 속도는 bending 을 굽힐 경우 7 이라는 숫자가 들어가게 설정하여 다른 팀의 속도보다 월등히 빠르게 설정하였고, 손을 피고 있을 때는 속도가 4 로 들어가서 미세한 컨트롤을 할 수 있도록 하였습니다. 한쪽의 바퀴만 구부리면 양바퀴의 속도가 차이가 나기 때문에 그 차이로 방향 회전을 하도록 하였습니다.

후진과 바퀴의 방향조절을 통해서 각도를 틀고 움직이는 데는 지장이 없었으나 결과론적으로 보자면 운전자의 실수가 있었습니다. 또한 만약 다른 팀처럼 bending 의 각도에 따라 속도를 조절하게 만들었더라면 코너링에서 더 부드럽게 운전이 가능 했을 것 같습니다. 하지만 그렇게 하기 위해서는 값을 계속 받아서 연산해야 하기 때문에 노트북으로 연습을 할 경우에는 손의 움직임과 화면 상의 자동차의 움직임에서 delay 가 있었습니다. 물론 좋은 컴퓨터로 demo 를 했기 때문에 이 문제점은 최소화 되었습니다. 다음에는 이를 보완하여 각도를 움직이는 것을 더 다양한 속도의 조절로 해결할 것 같습니다. 또한 후진도 각도를 움직여서 하는 것보다 일자로 뒤로 가는 것이 운전에 더 용의하다 판단하여 양 바퀴가 같은 속도로 움직이도록 설정하였습니다. 후진 중에 방향을 바꿀 수 있게 설정하였으나, 운전자가 후진은 전진에서 미처

방향을 조절하지 못한 것에 대한 최소한의 피드백으로 하면 좋겠다고 하여서 일자로 후진을 하도록 설정하였습니다.

자동차를 운전하는데 있어서 운전자가 원하는 방향으로 차가 움직이는 것이 가장 중요한 기능이라는 판단이 들었습니다. 하지만 운전자의 사용 미숙과 실수는 운전자가 보완하는 동시에 프로그램적으로 대처가 있었으면 더 좋았을 것이라는 생각이 들었습니다. 그래서 다음에 기회가 되면, 차선에 가까이 다가갈 경우 자동차가 중앙을 보도록 바퀴의 방향을 조절하는 기능을 추가하고 싶다는 생각이 들었습니다.

5-2. Picture(Track result picture)

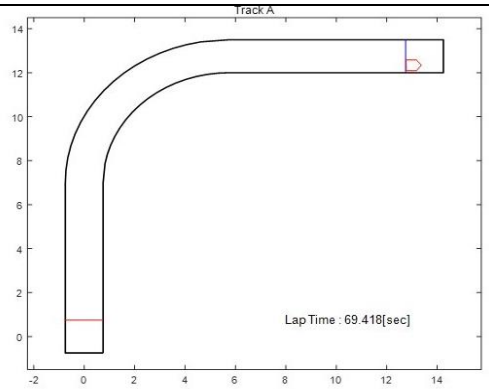

Figure 12. Track A

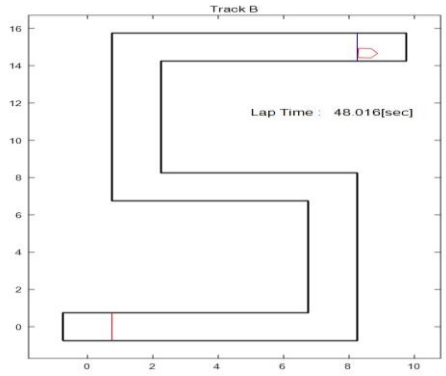

Figure 13. Track B

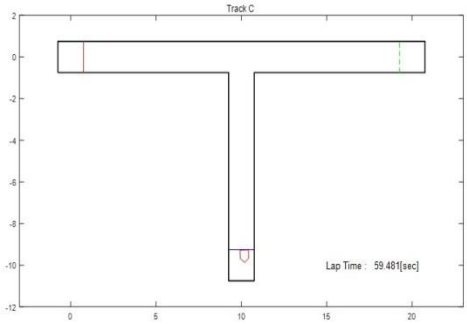

Figure 14. Track C

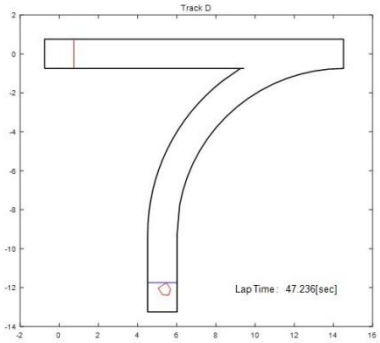

Figure 15. Track D

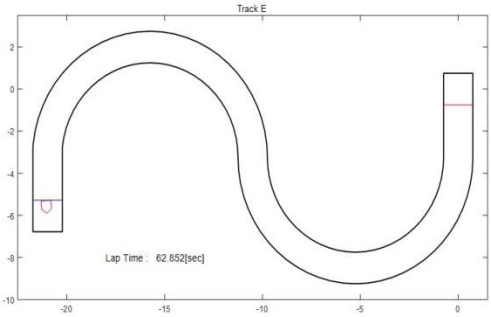

Figure 16. Track E