2016지능형모형차 경진대회 보고서

학 교	한양대학교
팀 명	A-TEAM
유 형	본경기
팀 장	김석원 (미래자동차공학과)
팀 원	권태준 (미래자동차공학과) 김기훈 (미래자동차공학과) 김상혁 (미래자동차공학과) 양진수 (융합전자공학부)

1. 개요

1.1 설계 배경

자동차 산업은 현대사회에서 가장 핵심이 되고, 주목을 받고 있는 산업이다. 그만큼 자동차 기술의 발달 방향과 성과에 따라서 사회에 큰 변화가 일어난다는 것이다. 전세계의 다수의 기업, 연구실 등에서 스마트카 개발에 앞다투어 참여하고 있고, 이미 사회에 변화가 나타나기 시작했다.

스마트카의 핵심 기술인 내장형 제어 시스템 설계를 목표로 하는 '2016 지능형 모형차 경진대회'는 대학생들에게 미래자동차에 한발 다가갈 수 있는 좋은 기회를 제공한다. 지능형 모형차 설계에는 임베디드 시스템을 기반으로, 학부에서 공부하는 신호처리, 제어, 역학 등의 전자, 전기, 기계적 지식이 필요하다. 우리는 이론적 지식을 토대로 내장형 제어 시스템을 이해하고 설계하여 미래형 자동차 산업에 한 발 다가가고자 이 대회에 참가하여 설계를 진행하였다.

1.2 설계 목표

미래형 자동차의 목표는 자율 주행 자동차 개발을 통해 삶의 질을 향상시키는 것이고, 여기서 가장 기본적인 전제가 되는 것이 바로 '안전'이다. 이번 대회의 목적도 차선 유지 제어와 AEB 등의 기술의 구현에 있고, 궁극적으로 '안전'한 주행에 있다. 우리는 주행선을 유지하면서 달리는 차선 유지제어(Lane Keeping Control)를 기본으로 하고, 장애물 인식 후 충돌 없이 정지하는 AEB 와 회피하는 Obstacle Avoidance에 중심을 두고 설계를 진행하였다.

Lane Keeping은 lane detecting 하여 얻은 차선에 대한 정보를 이용하여 차량의 조향각과 회전반경을 구하는 Pure Pursuit 경로 추종 알고리즘을 사용하였다. AEB의 경우 장애물 인식 후, 제어를 이용해 어떤 상황에서도 확실한 정지를 해내는 것을 목표로 하였다. 스쿨 존에서의 장애물 회피는 스마트카에 있어서 안전과 직결되는 부분으로, 이번 대회에서 가장 초점을 두는 기술이다. 따라서 미래에 우리의 자녀들의 안전을 지켜주겠다는 마음을 담아 설계에 심혈을 기울였다.

2. 설계 내용

2.1 하드웨어 구성

2.1.1 Circuit

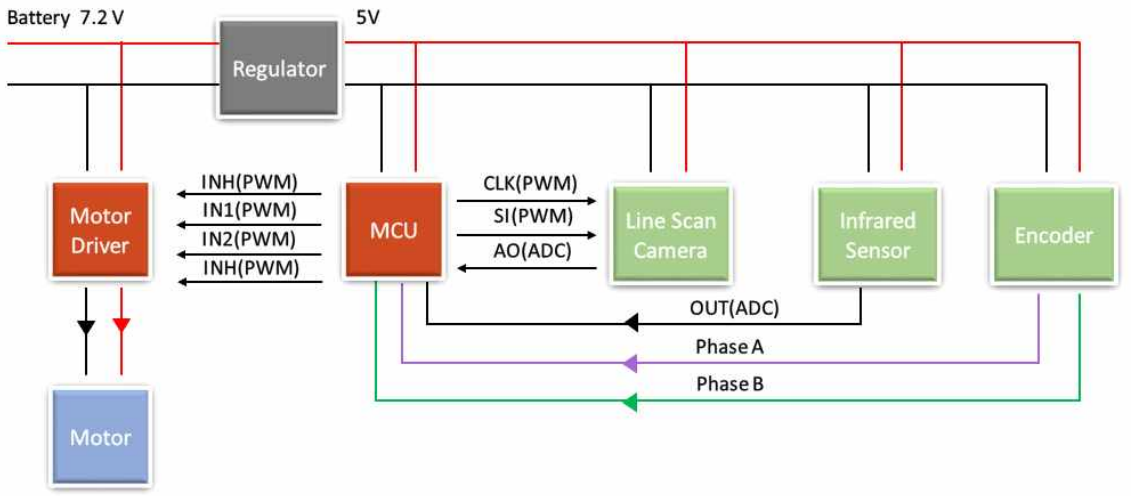


Fig. 1 circuit

배터리에서 나온 7.2V를 regulator를 사용하여 5V로 converting을 해주었고, 모터를 제외한 다른 센서부 등에 사용되었다.

2.1.2 MCU

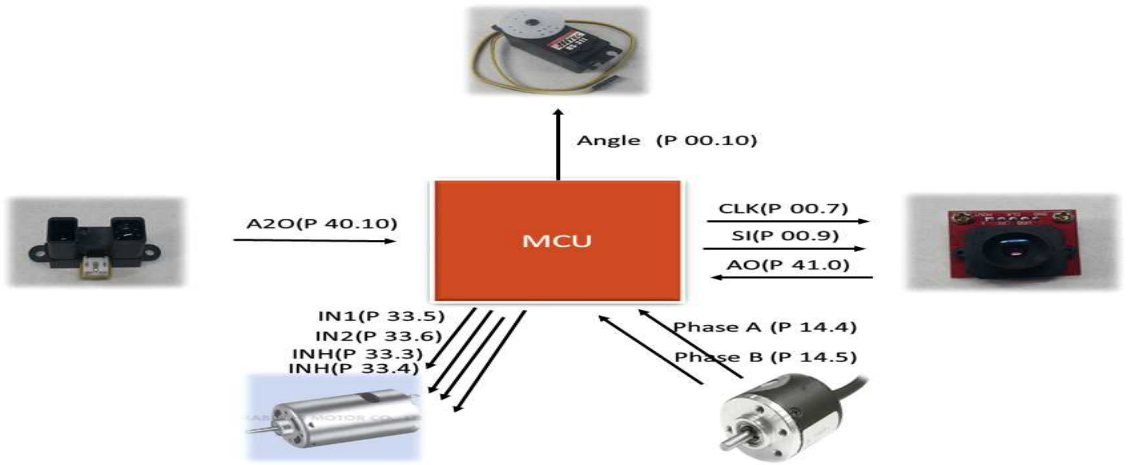


Fig. 2 MCU circuit

MCU와 신호를 주고 받는 것은 위 그림과 같은 5가지가 있다. 포트를 구성할 때에는 헷갈리는 일이 없도록 같은 장치에 쓰이는 것은 최대한 같은 곳으로 배치하

였다.

2.1.3 센서부

가) 라인 스캔 카메라

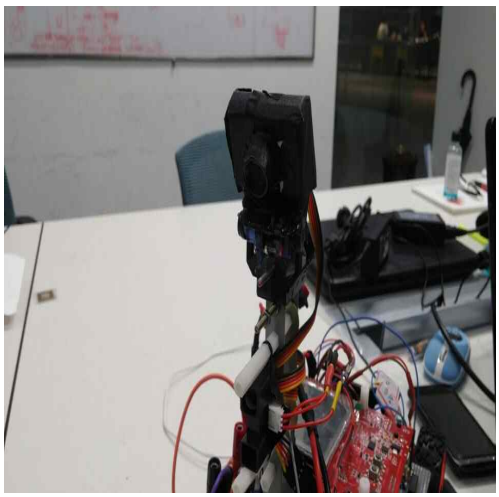


Fig. 3(a) 라인스캔 카메라

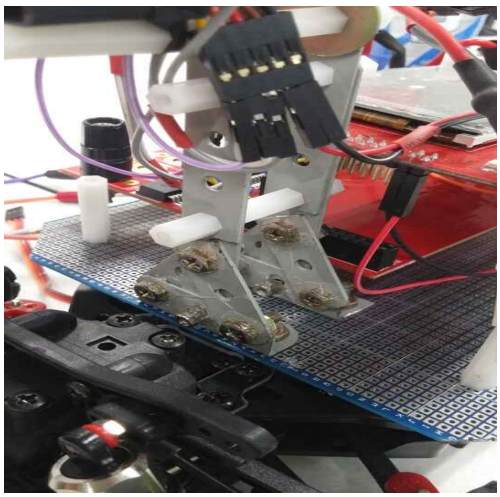


Fig. 3(b) 카메라 고정부(나사)

1) Linescan Camera spec

사용한 카메라는 freescale cup linescan camera(TSL-1401)이다. 128개의 Photo Diode array로 구성되어 128x1 개의 pixel을 가지고 있다. MCU에서 넣어주는 clock의 frequency 범위는 5-2000kHz까지 가능하다. 또한 Integration에 필요한 시간은 최소 0.0645ms 최대 100ms 이다. 카메라가 읽은 값은 0-4095까지의 아날로그 값으로 MCU로 전송해준다.

2) Camera Hardware 구성

지능형 모형차의 핵심이 되는 것은 카메라 값을 받아서 라인을 읽는 것이다. 그렇기 때문에 외부 충격(주행 시 떨림 등)에 대해 카메라가 움직이게 된다면 라인을 벗어나는 현상을 발생시킬 수 있다. 그렇기 때문에 카메라를 강하게 고정을 하고 카메라의 각도를 유지하는 것이 중요하였다. 우선 카메라의 각도를 유지하기 위해서는 서보모터를 이용하여 원하는 각도를 찾은 후 고정시켰다. 카메라 모듈 전체를 고정하기 위해서는 Figure. 3(b) 와 같이 과학상자와 나사 등을 이용하여 고정하였다.

나) 적외선 센서

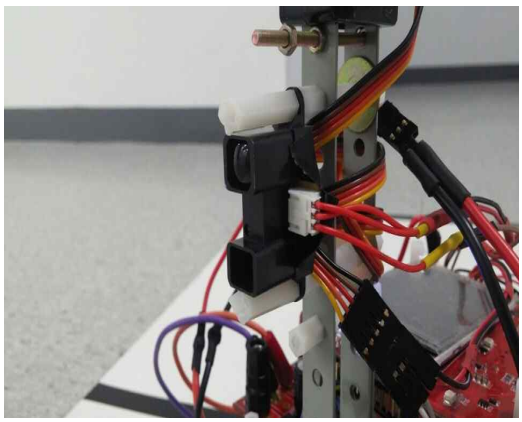


Fig. 4(a) 적외선 센서 (정면 우측)

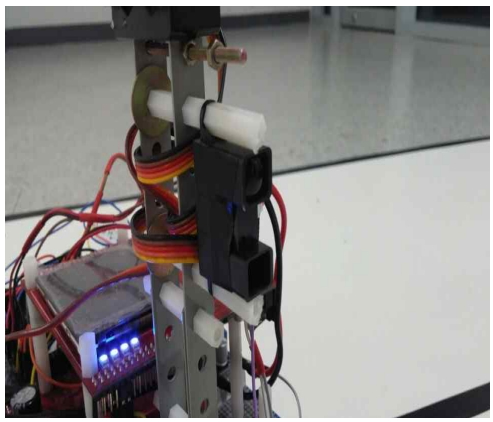


Fig. 4(b) 적외선 센서 (정면 좌측)

1) Infrared Sensor spec

적외선 센서로는 GP2Y0A2YK0F를 사용하였다. 측정 시간은 38.3ms±9.6ms이고, 측정 범위는 20 - 150cm이다.

2) 선정 이유

적외선 센서는 장애물을 보고 회피하거나 AEB 시에 사용된다. 만약 적외선 센서의 range가 너무 작으면 장애물을 가까이에서 인식하기 때문에 회피나 정지 시에 문제가 발생할 수 있다. 따라서 range가 20 - 150cm 인 적외선 카메라를 사용하였다.

2.1.4 구동부

가) Motor

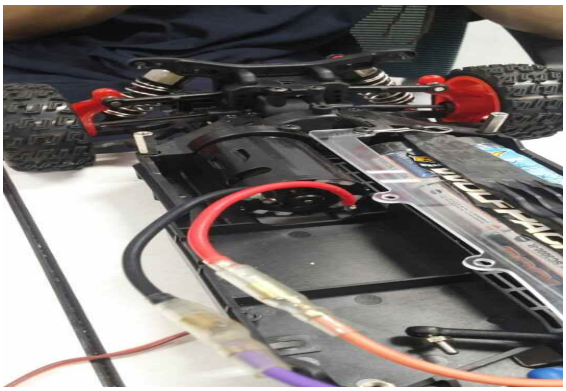


Fig. 5 DC Motor

DC Motor는 기존 차량에 부착된 모델을 그대로 사용하였다. 모터드라이브와의 연결은 커넥터를 이용하여 탈착이 편리하게 만들었다.

나) Encoder

엔코더는 바퀴의 속도를 재기 위하여 장착했다. PID 제어를 위해서는 현재 차량의 속도를 알아야 하기 때문에 엔코더가 필요하였다. 또한 엔코더는 차량의 기어에 연결되어야 하는데 겉으로 드러난 부분이 없었기 때문에 기어박스의 일부를 잘라서 기어를 드러나게 한 후에 엔코더를 연결하였다.

Fig. 6(a) 기어 박스 cutting

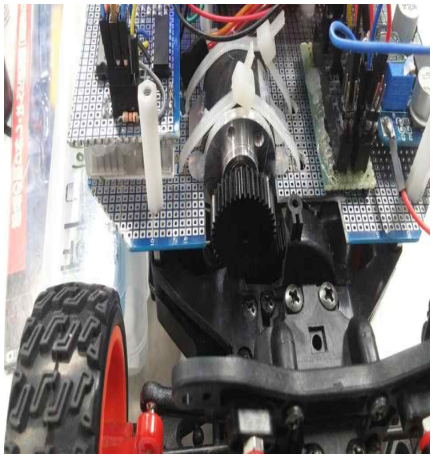


Fig. 6(b) 엔코더 - 기어 결합 후

1) Encoder spec

엔코더는 Autonics사의 E30S4-3000-3-N-5를 사용하였다. 1 회전당 3000pulse의 분해능을 가지고 있으며, 전압 출력을 위하여 출력부에 1.2KΩ의 저항을 연결해주었다.(NPN Open Collector type)

2) 엔코더 회로

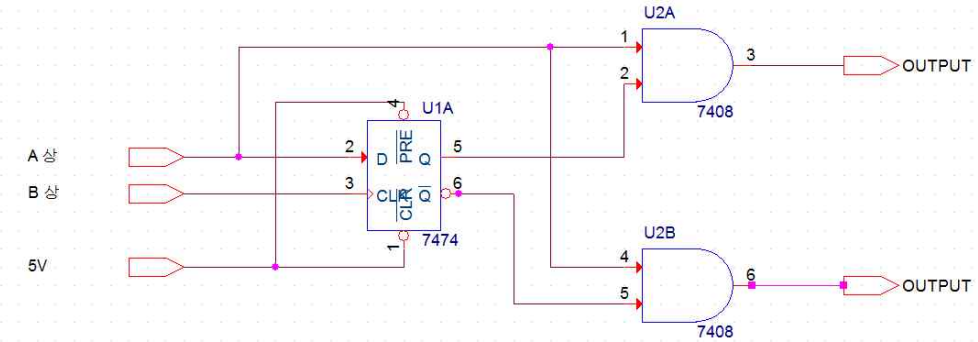


Fig. 7 정회전, 역회전을 구분하기 위한 엔코더 회로

PID 제어를 위해서는 정회전과 역회전을 구분 할 수 있어야 한다. 엔코더에서 나오는 Phase A 상의 출력과 Phase B상의 출력의 타이밍 차이로 정회전인지 역회전인지를 구분하는데 이를 위해서 D-Flipflop과 AND gate를 이용하여 회로를 구성하였다.

2.1.5 조향부

가) Servo Motor

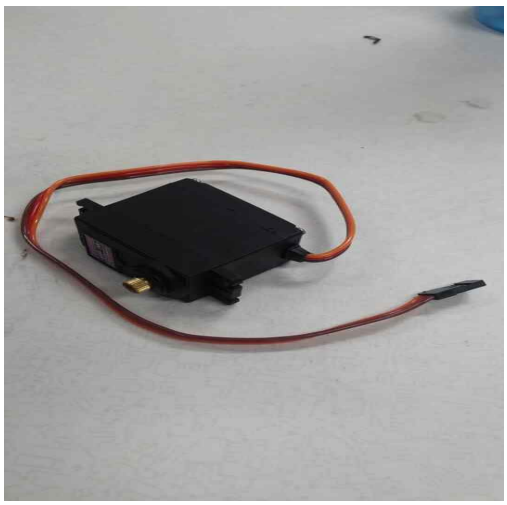


Fig. 7(a) Servo Motor

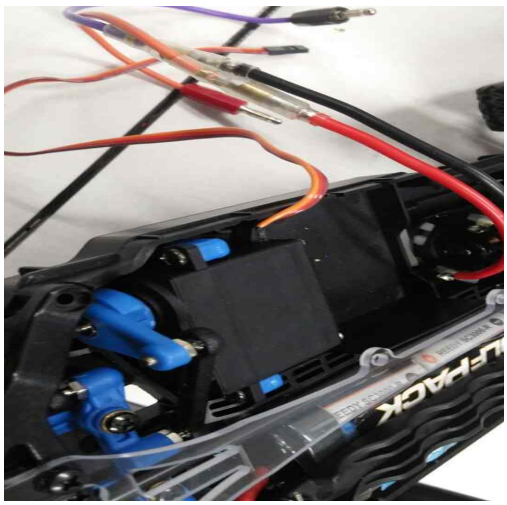


Fig. 7(b) Servo Motor 장착 모습

1) 서보 모터 SPEC

서보모터는 MG996R 모델을 사용하였다. 6V를 가해줬을 때 최대 토크가 나오고 running current는 500-900mA, stall current는 2.5A이다. operating speed는 6V를 가했을 때, 60도에 0.14s가 소요된다.

2) 선정 이유

처음에는 HS311 모델을 사용하였으나 차체 무게로 인하여 일정 범위 내의 각도는 HS311의 토크로는 움직이지 않는 문제가 발생하였다. 따라서 토크가 더 큰 MG996R 모델을 선정하였다.

3) 수정 사항

서보가 잘 동작함에도 불구하고 휠까지 움직임이 잘 전달되지 않는 문제가 발생하였다. 이 문제는 링크부가 플라스틱 재질로 되어 있어 서보가 움직임에도 링크 자체가 벌어지면서 유격이 발생한 것이었다. 따라서 링크가 벌어지는 부분에 본드로 고정을 하여서 해결하였다.

2.1.1.1 전체 하드웨어 구성

Fig. 8(a) 전체 외관 (바닥)

Fig. 8(b) 전체 외관 (정면)

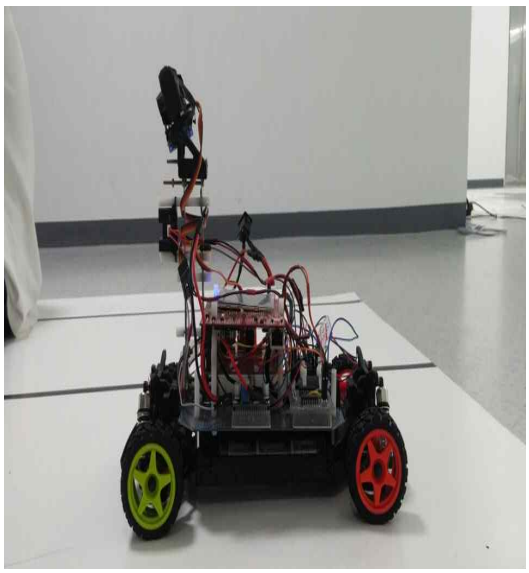


Fig. 8(a) 전체 외관 (우측면)

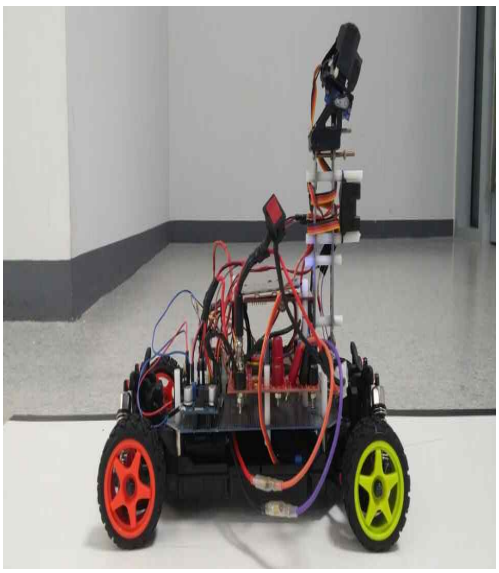


Fig. 8(b) 전체 외관 (좌측면)

차량의 하단부에 대해서는 제공된 모델과 거의 동일하다. 바뀐 점은 서보 모터와 기어박스의 수정, 그리고 배터리 고정부이다. 배터리 고정부에 대해서는 상단을 고정할 때 겹침 현상이 일어나서 튀어나온 부분을 갈아주었다. 하단부와 상단부는 스페이서를 통해서 연결하였다. 상단부 전체는 PCB 위에 놓인 형태를 하고 있다. PCB 판 위에 regulator 두 개와 Encoder 회로가 놓여있고 그 위로 모터드라이브가 스페이서를 이용하여 고정되어 있다. MCU 역시 스페이서를 이용하여 모터드라이브와 연결되어 있으며 라인스캔 카메라와 적외선 센서가 과학 상자 부품을 이용하여 앞쪽에 고정되어 있다.

2.2 소프트웨어 구성

2.2.1 Flow Chart

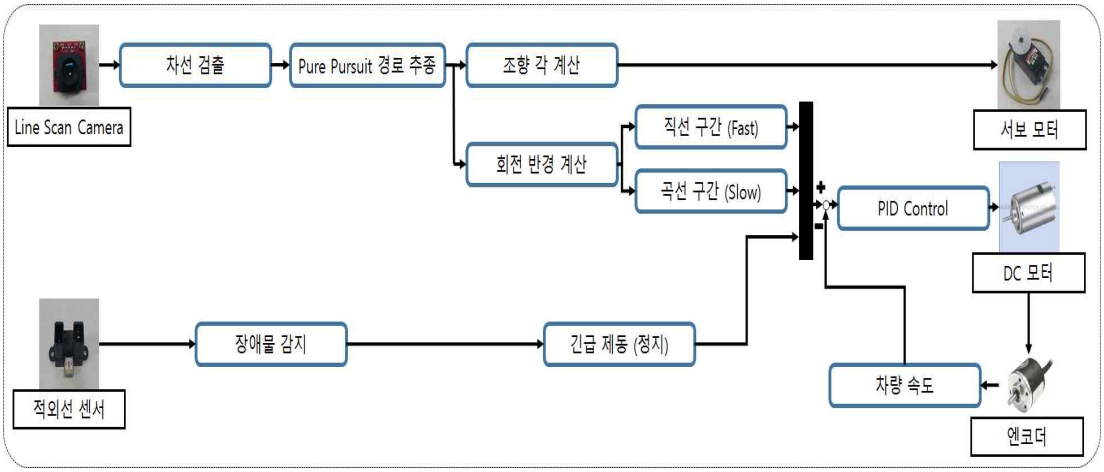


Fig. 9 Speed race 알고리즘

2.2.2 Lane detecting

Lane detecting을 위해서 Freescale cup line scan camera(TSL–1401)를 사용하였다.

가) median filter

처음 라인스캔 카메라만을 이용하여 값을 받았을 때는, 노이즈로 인해 튀는 값이 발생하였다. 이를 막기 위하여 median filter를 사용하였다. median filter란 여러 개의 값들을 크기 순으로 배열하여 중간값을 사용하는 filter이다. 이러한 filter는 noise가 발생하였을 경우에 noise의 영향을 줄여주는 역할을 한다.

나) Kernel filter

filter를 적용하기 전, 카메라의 결과값은 Fig. 10(a)의 그래프와 같다.

Fig. 10 (a) filtering 전 카메라 data

그래프를 보면, 검정색 라인과 흰색 배경의 값 차이가 2000 정도 차이가 남을 볼 수 있다. 하지만 조명 조건이 달라질 경우에 이 값의 차이가 더 줄어들었고 threshold값을 지정해 라인을 구분하는데 어려움이 있었다. 따라서 우리는 검정색 라인과 흰색 배경의 값 차이를 극대화시키기 위하여 kernel filter를 사용하였다. 영상처리에서 경계선을 명확하게 하는 데 쓰이는 filter를 1차원으로 변환시켜 적용해보았다. kernel filter란 증폭하고 싶은 값에 큰 값을 정해주고 양쪽 값에 대해서는 세 값의 합이 0이 되는 값들을 곱해줘서 증폭시키는 원리이다. 원래 kernel filter는 값을 증폭시켜주는 역할을 하는데 우리의 경우는 값을 반대로 증폭을 시켜줘야 했기 때문에 카메라 값을 4095에서 빼서 검정색 라인이 값을 가장 크게 만들고 시작하였다. 또한, 검정색 라인이 한 pixel에서만 나오는 것이 아니라 약 5개 pixel 에 걸쳐서 나왔기 때문에 증폭을 해줄 때 한 개의 값에 큰 값을 곱해주는 것이 아니라 5개의 값에 값을 곱해주었다. Fig, 10(b)는 kernel filter를 적용시킨 후의 카메라 data이다. filter를 적용시키기 전과 다르게 검정색 라인과 흰색 배경의 값 차이가 10000이상이 되면서 threshold를 안정적으로 정할 수 있는 범위가 넓어짐을 확인 할 수 있었다.

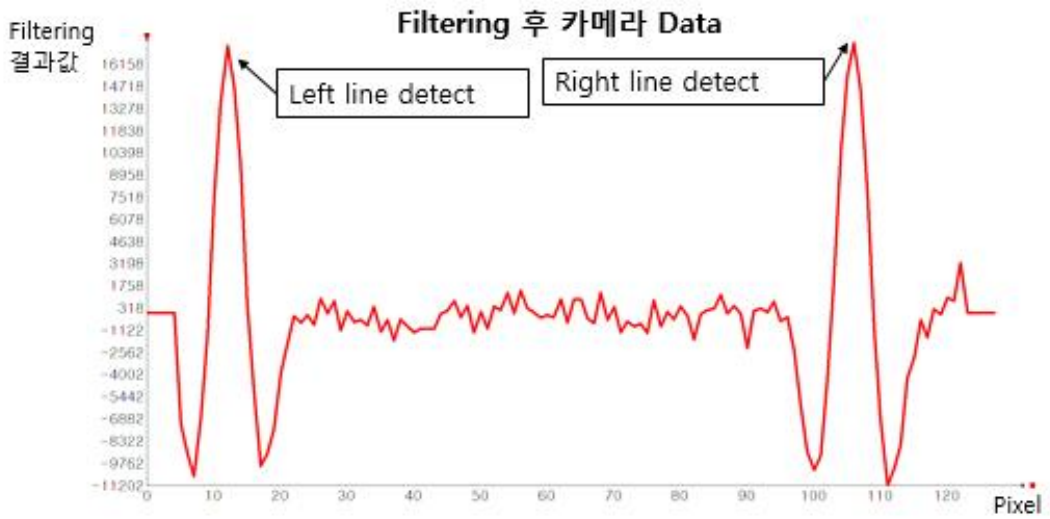


Fig. 10(b) Filtering 후 카메라 data

다) Algorithm(center value 이동)

필터를 거쳐서 받은 값들 중 어느 것이 라인인지 판별을 하기 위해서 maximum 값을 찾았다. maximum 값을 찾을 때는 128개 pixel의 중간인 64pixel을 기준으로 왼쪽에 있는 값들 중 가장 큰 값을 왼쪽 라인이라고 하고, 오른쪽에 있는 값들 중 가장 큰 값을 오른쪽 라인이라고 인지하였다. 또한 라인이 아닌 경우에도 maximum 값은 항상 존재하기 때문에 해당하는 pixel의 값을 살펴보았을 때, 우리가 지정한 threshold 값을 넘을 경우에만 라인으로 인지하도록 하였다. 뿐만 아니라, 카메라 pixel중 양 끝의 10개 정도의 pixel은 조명이 밝지 않은 경우 값이 제대로 나오지 않는 문제가 발생하여 7에서 120의 범위 안에 있는 경우만 라인으로 인지하였다.

이 알고리즘에서 발생할 수 있는 문제는 카메라가 앞쪽을 보고 있기 때문에 코너링 상

황에서 오른쪽 라인 값이 64보다 작아지는 경우가 생긴다는 것이다. 이 경우에는 왼쪽 라인으로 인식이 되기 때문에 반대방향의 조향이 되어서 라인을 벗어나는 경우가 생긴다. 따라서 왼쪽 라인이 일정 값 이상으로 커질 경우에는 64였던 center value를 더 크게 줘서 왼쪽 라인의 범위를 늘려주고 오른쪽 라인이 일정 값 이하로 떨어질 경우엔 center value를 줄여서 오른쪽 라인의 범위를 늘려주며 해결하였다.

2.2.3 Lane Keeping

가) Pure Pursuit 알고리즘 : 3.1.1에 설명

나) Motor duty control Algorithm

Motor는 MCU의 PWM 신호로 동작을 하는 것이기 때문에 PWM의 Period를 고정하고 Duty 값을 조정해 줌으로써 속도를 바꿔줄 수 있다. 일반 차량의 주행 시 코너에서 속도를 줄이는 것을 볼 수 있는데 이 점에 착안하여 왼쪽 라인값이 일정 수준 이하로 올라가거나 오른쪽 라인 값이 일정 수준 이하로 떨어질 경우 코너라고 인식하고 속도를 줄여주었다.

2.2.4 school zone

가) school zone detect

School Zone에 들어가게 되면 속도를 줄여야 하기 때문에 School Zone을 인식하는 것은 필수적이다. School Zone은 폭 10cm의 검정색 선으로 구성되어 있다. 검은색 선을 인지하는 것은 128개의 pixel 중 일정 개수 이상의 값이 검정색이라고 인식을 하면 검은색 선이라고 인식을 하였고, 그 이 후 다시 흰색이 되면 라인을 통과한 것이라고 생각하고 모터의 속도를 조절하였다. 바로 전에 detecting 된 라인 값과 현재 값을 비교할 경우에는 검은색이 연속적으로 인식 될 수 있기 때문에 3-4 타이밍 이전의 값과 비교를 하였다.

나) Obstacle Avoidance

앞에 장애물이 있을 때 장애물을 피하는 경우와 AEB 두 가지 경우가 있다. 장애물은 School Zone 안에만 있기 때문에 School Zone 안에 있다고 판단될 때 장애물을 볼 경우에는 장애물로 인식하여 피하게 하였다. 또한 장애물을 만났을 때 주행로에 있는지 추월로에 있는지에 따라 피하는 방향이 달라지기 때문에 처음 주행로 상태에서 피한 횟수를 이용하여 주행로인지 추월로인지를 판단하여 피할 방향을 정해주었다. 장애물을 인식하였을 경우에 일정시간동안 원하는 방향으로 차선을 인식하지 못할 때까지 회전을 해준 후에 반대쪽 라인의 값(주행로에서 피할 경우 추월로의 왼쪽 라인, 추월로에서 피할 경우 주행로의 오른쪽 라인)이 일정 값이 될 때까지 직진을 한다. 그리고 그 후에는 Lane Keeping Algorithm을 이용하여 라인을 따라가도록 하였다.

3. 주요 장치 이론 및 적용 방법

3.1 Lane Keeping Algorithm

3.1.1 Pure pursuit

Lane detecting의 결과로 나온 라인 값을 바탕으로 자동차의 경로 추종을 Pure Pursuit 알고리즘을 이용하여 설계하였다.

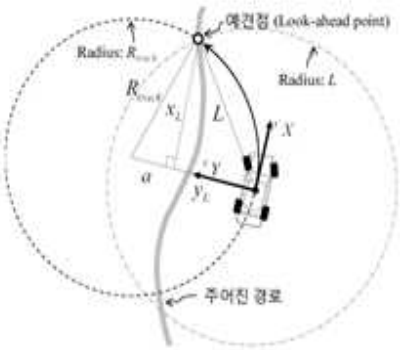
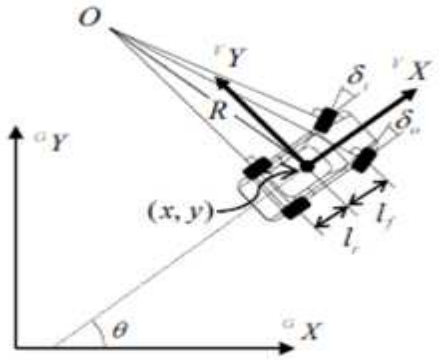


Fig.11(a) Determining the turning radius to follow the path



Fig, 11(b) coordinate systems and the kinematics model of the vehicle

$$x_L^2 + y_L^2 = L^2$$
$$a^2 + x_L^2 = R_{track}^2$$
$$a + y_L = R_{track}$$
$$R_{track} = \frac{L^2}{2y_L}$$

$$\tan(\delta) \cong \delta$$
$$\delta = \frac{\delta_o + \delta_i}{2} \cong \frac{l_f + l_r}{R} = l\kappa$$

Fig, 11(a) 에서 X_L 은 차량의 중심에서 카메라가 바라보고 있는 점까지의 위치이다. 카메라의각도가 너무 낮으면 고속 주행때 불안정해지고, 너무 크면 값이 불안정해지기 때문에 620mm를 기준으로 하였다. Y_L은 목표로 하는 경로상의 점, 즉 라인의 중앙에 위치한 점과 현재 차량 위치의 거리 차이다. Y_L 같은 경우 한쪽 차선을 기준으로 할 때 항상 235mm만큼 떨어져 있기 때문에 검출되는 라인 값을 이용하여 추정이 가능하였다. 그리고 이 값과 식들을 이용하여 원하는 점까지의 회전반경 R_{track}을 구할 수 있었다. 차량의 회전 반경이 정해지면, Fig. 11(b)의 식을 통해서 휠의 조향각을 찾아낼 수 있었다.

3.2 PID제어를 이용한 AEB

AEB를 구현하기 위해 먼저 MATLAB Simulink를 통해 PID controller를 설계하여 보았다. AEB에서 PID제어를 적용한 이유는 AEB에서 steady-state error가 있거나 response time이 길면 무조건 충돌할 수 밖에 없기 때문이다. Simulink의 Auto Tuning 기능을 통해서 적절한 gain값들을 찾아서 실험해 보았다.

실제 실험에 있어서는 주행환경에 따라 조금씩 오차가 발생하였다. 따라서 auto tuning으로 나온 gain을 바탕으로, 실험을 통해 확인하면서 조금씩 gain값을 조절하였다. P gain을 높이면 rise time이 감소하지만 oscillation이 증가하고, D gain을 높이면 overshoot는 감소하지만 settling time이 증가하며, I gain을 높이면 steady-state error는 감소하지만 overshoot가 증가한다는 점을 이용하였다. 이를 통해 steady-state를 거의 0으로 만들면서, rise time이 빨라서 즉각적으로 반응 할 수 있는 AEB시스템을 완성시킬 수 있었다.

4. 결론 및 토의

4.1 결론

4.1.1 Lane Keeping

Pure pursuit 알고리즘을 이용하여 lane keeping 알고리즘을 설계하였는데, 모형차의 곡선 진입 속도, 진입 각에 따라서 상황이 달라져서 직선속도와 곡선 속도에 각각에 적절한 값을 지정해주어야 했다. 또한 직선구간에서의 oscillation을 잡는데 있어서 PD제어를 통해 적절한 wheel gain을 조절해 주어야했다. motor의 PWM duty를 바꿔가며 실험적으로 가장 적합한 wheel gain, wheel angle, 그리고 직, 곡선에서의 속도를 구하고자 했다. 이 과정에서 경향성을 파악하면서 진행하였는데, 예상과 달리 주행을 할 때마다 어려움을 겪었다. 속도에 따라서도 경향성이 달라졌고, gain 값을 바꿀 때 작은 차이에도 결과가 판이하게 달라졌다. 결과적으로 적절한 gain값을 찾지 못했을 때, 오히려 더 큰 oscillation을 발생시킨다는 것을 알 수 있었다. 그와 동시에 진행 한 것이 직진일 때 Wheel angel에 제한을 두는 방법이었다. 직진에서는 Wheel angle이 커질수록 카메라가 인식한 라인 값의 변화가 커지는데 continuous 하지 않고 discrete한 시스템에서 급격한 라인 값의 변화는 oscillation을 일으키는 원인이 되었다. 따라서 직선에서는 Wheel angle의 값에 제한을 줌으로써 라인 값의 급격한 변화를 방지하였다. 하지만 이 부분에 있어서는 직선구간에 정확하게 진입해야 한다는 단점이 있기 때문에 코너 탈출 시 정확한 진입을 위한 방법이 필요하였다.

또한 Lane keeping을 위하여 최적의 서보모터를 찾는 과정도 필요했다. 처음에 선택한 서보모터가 0°, 즉 reference angle에 제대로 도달하지 않고 계속해서 흔들려서 직선에서의 oscillation을 잡는데 어려움을 겪었다. 문제를 해결하기 위해 서

보모터를 교체하였다.

4.1.2 Obstacle Avoidance

school zone에서 장애물을 인지하였을 때, 주행로를 변경하여 회피하는 알고리즘을 설계하였는데, school zone에서 나올 때 기존 주행로로 복귀해야하는 점을 해결하기 위해 여러 가지 방법을 고안하였다. 처음에 시도한 알고리즘이 오른쪽 장애물 회피 후 자동 복귀하는 알고리즘이었다. 오른쪽 장애물 회피 후에 오른쪽 차선의 변화를 인식하면서 회전각을 유지하여 다시 복귀하게 하였다. 하지만 왼쪽차선에 장애물이 먼저 나타나거나, 오른쪽 차선에 장애물이 연달아 있을 경우에 문제가 발생하여 완벽한 알고리즘을 다시 고안하였다. 회피할 때마다 counting을 하여 현재 차선을 판단하고, school zone에서 나갈 때 line scan camera로 인지하여 주행로로 복귀하는 알고리즘을 구현하였다.

4.2 토의

김석원: MCU manual을 보면서 레지스터, port 등을 찾아내는 것부터 시작해서 전반적인 주행, AEB, Obstacle Avoidance 등의 설계를 하면서 정말 힘들지만 즐겁게 준비할 수 있었다. 카메라, 엔코더, 서보모터 등을 coding을 통해 설계하면서, embedded system과 자동차에 대한 전반적인 이해를 할 수 있었다.

권태준: 지능형 모형차를 설계함에 있어서, 기계/전자적 지식과 역량이 모두 필요하다는 것을 몸소 느낄 수 있었다. 또한 납땜을 하며 회로를 제작하고, 코딩을 하면서 하드웨어/소프트웨어적 역량을 고루 갖춘 인재가 되어야겠다고 다시 한 번 다짐하였다.

김기훈: 그 어떤 주행보다 직선주행 알고리즘을 완벽하게 짜는 것이 가장 어려웠던 것 같다. 직선 oscillation을 잡으려 많은 알고리즘을 시도하고 실패하면서 많이 배울 수 있었다.

김상혁: school zone을 인식하는 완벽한 방법을 찾느라 시간을 많이 투자하였다. 여러 시행착오 끝에, 어떤 상황에서도 라인을 확실히 인식하도록 알고리즘을 설계하였다. 이번 대회를 통해 자율주행자동차에 대해 많이 느끼고 배울 수 있었다.

양진수: 이번 대회에서 가장 어려운 점은 주행에 있어서 어떤 상황에서도 똑같은 결과를 내는 차를 만들어야 한다는 것이었다. 제어를 좀 더 공부해야 할 필요성을 느꼈다.

Appendix – 부품 목록

제조사	부품명	수량	사용목적
TowerPro	MG996R	1	Servo Motor / steering
TAOS	TSL-1401	1	line scan camera / 차선 검출 및 school zone 인식
Autonics	E30S4-3000-3-N-5	1	Encoder / 차량 속도 검출
Sharp	GP2Y0A21YK	1	적외선 센서 / 장애물 거리 판단
Hitachi Semiconductor	HD74LS74AP	1	D flip-flop / NPN 엔코더 출력
NXP Semiconductor	74HC08N	1	AND gate / NPN 엔코더 출력
XLSemi	XL4005	1	DC-DC Step-down converter / 전압 분배
Eleparts	EPX37WJG	1	DC-DC Step-down converter / 전압 분배
Reedy	Reedy WolfPack 7.2V 3000mAh Ni-MH	1	차량 전원 공급
-	resistance	2	1.2k ohm / NPN 엔코더 출력