2019

**지능형모형차 경진대회 MATLAB활용 보고서**

|  |  |
| --- | --- |
| **학 교** | 건국대학교 |
| **팀 명** | 숑지나가요 |
| **팀 장** | 김승수 (전기전자공학부) |
| **팀 원** | 김은주(전기전자공학부)  민경민(전기전자공학부)  유진호(전자공학과)  김서정(전기전자공학부) |

1. 개요

1.1 설계 배경

오늘날 전 세계는 여가시간을 창출해 내고 앞으로의 인류를 위해 두 가지 키워드를 화두로 삼고 있다. 첫째로 5G, big data, IoT, deep learning, AI등의 4차 산업, 둘째로는 수소, 전기, 풍력 등의 친환경이다. 이들 중에서도 많은 관심을 두고 있는 자율주행이라는 단어는 이제 점점 우리에게 익숙한 단어가 되었고 이제는 생활 속에서 함께할 일만을 남기고 있다.

이런 세계적인 흐름을 따라 애플의 경우 핸드폰, 노트북 등의 기존의 주력 산업에서 자율주행 자동차 스타트업 기업을 인수했고 LG전자 역시 블랙베리와 함께 자율주행차 기술을 갖추는데 노력함으로써 그 흐름에 뛰어들었다. 그렇지만 우리의 생활 속에 완전히 녹아 들어오기에는 아직 많은 시간이 필요할 것으로 예상된다. 가장 최근의 사건으로 서울시에서 시연한 5G 자율주행 버스의 중앙선 침범이 있었으며 해외에서는 자율주행차 시범운전을 하던 중 사망사고가 발생한 경우가 있었기 때문이다. 사실 서울시에서 자율주행 버스를 시연하기 전까지는 대부분 사람, 가로등, 건물 등과 같은 장애물을 피하고 이 때 필요하다면 긴급 제동을 해야 하기에 이 두 가지에 큰 관심을 두었을 것이다.

그러나 이제는 가장 기본인 선을 인식하는 것에서 부족함이 나타나는 것을 확인했다. 따라서 우리 팀은 이 Line Scan이라는 기본을 단단하게 다진 후에 장애물을 회피하고 긴급제동을 하는 것을 쌓아 올리는 것을 목표로 삼고 지능형 모형차 대회에 참가하게 되었다.

1.2 설계 목표.

지능형 자동차는 자동차 스스로가 현재, 이전 모든 상황을 종합하여 앞으로의 주행상황을 예측한다. 따라서 차선 확인(Line Scan), 장애물 인식, 회피, 긴급 제동(AEB)의 크게 네 가지를 목표로 각각의 상황마다 속도와 방향을 조절해야 한다. 따라서 이를 가지고 세부 목표를 세워보면 다음과 같다.

* 언덕 등 주변 환경에 영향을 받지 않고 차량이 일정한 속도로 주행을 할 수 있도록 한다.
* Line Scan camera를 이용하여 차선을 인식함으로써 차량이 스스로 자신의 위치를 파악할 수 있도록 한다.
* IR 센서를 이용해 장애물, AEB, 언덕의 경우를 각각 알고리즘을 통해 판단한 후 각각의 경우에 맞춰 차량이 속도와 방향을 제어한다.

이 세가지 큰 목표를 통해 차량이 스스로 상황을 판단하고 결정을 내리는 자율주행 시스템에 대해 이해하고 설계를 해보고 직접 확인까지 하는 것이 목표이다.

2. MATLAB을 활용한 지능형모형차 설계 내용

2.1제어 이론 및 MATLAB을 이용한 제어 알고리즘 구현

2.1.1 제어 이론

2.1.1.1 Line Searching

우리 팀의 Line Scan Camera는 한번에 최대 두개의 Line을 검출할 수 있다. 양 쪽에 달린 카메라는 각각 한 개의 Line을 검출하고, 가운데에 달린 카메라는 좌 우 두개의 라인을 한꺼번에 검출하게 된다. 세 개의 TSL1401을 통하여 Pulse 신호를 받게 되면, CLK 주기에 따라 128Pixel 만큼의 Analog 값이 출력되고, 이 데이터를 ADC로 받아들인 후에 저장되게 된다. 이 Data를 UDE를 통해 분석한 결과, 라인이 있는 검은색 부분에서 값이 낮고, 흰색 부분에서 값이 크다. 라인 스캔 카메라는 카메라의 특성 상, 가운데 부분을 더 밝게 인식하고, 라인(검은색)을 인식하게 되면 급격히 값이 변화하는 것을 이용하여, Differential Filter를 이용하여 변화량이 큰 부분을 라인으로 검출하였다.

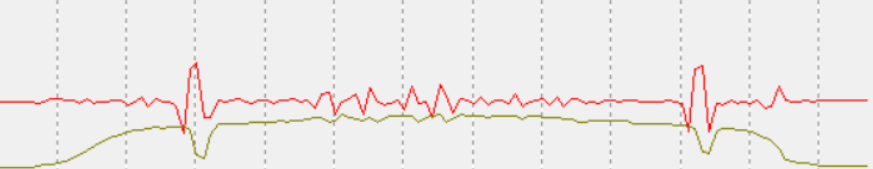


Figure 1. Raw Data(초록색) 와 Differential 필터 적용 후(빨간색)

하지만 이 경우에는, 노이즈가 심해 라인이 정확히 검출이 어려웠고, 이를 개선하기 위해 Gaussian Filter를 추가적으로 사용하여 노이즈를 개선시켰고, 효과적으로 라인을 검출할 수 있었다.



Figure 2. Raw Data(초록색) Differential필터, Gaussian필터 적용 후(빨간색)

2.1.1.2 Smart Cruise Control

임의의 PWM값을 DC 모터에 주입하여 인코더에서 검출되는 속도를 확인한 결과, 값이 일정하게 도출되지 않는 것을 확인하였다. 특히, 주행코스 중 하나인 높은 언덕코스는 어지간히 강한 PWM 값이 아니면 통과하지 못하였다. 지나치게 빠른 속도로 주행 시 라인 검출 및 주행 자체에 문제가 생겼고, 이를 해결하기 위하여 PID 제어로 정밀한 속도 제어를 하였다.



Figure 3. Simulink Discrete PID Controller Block Equation

PID 제어는 비례 제어(Proportional), 적분 제어(Integral), 그리고 미분 제어(Derivative)으로 제어하여 정확한 목표 값에 접근시킨다. 비례 제어는 에러 값에 비례하여 제어율을 변화시키며 시스템의 transient time 및 정상상태 오차를 감소시키나, overshoot의 증가를 가져온다. 적분 제어는 에러 값을 적분한 값을 가지고 제어하며, 정상상태 오차를 제거하고 상승 시간을 줄이는 반면, overshoot와 정정시간(setting time)이 증가된다. 마지막으로 미분 제어는 오차 값의 변화를 보고 제어한다. 미분 제어는 Overshoot 및 정정시간을 감소시킨다. 위의 식은 Simulink 내 이산 PID 블록의 공식으로 Ts는 Sample Time이며 z는 제어하게 될 대상 값이다. 현재 인코더의 속도를 대상 값으로 P, I, D의 값들을 적절히 제어하여 나온 결과값에 UDE에서 확인한 추측 인코더 스피드와 PWM 값의 비례 상수를 곱하여 목표 인코더 속도에 접근하기 위한 PWM 값을 주기적으로 도출하였다.

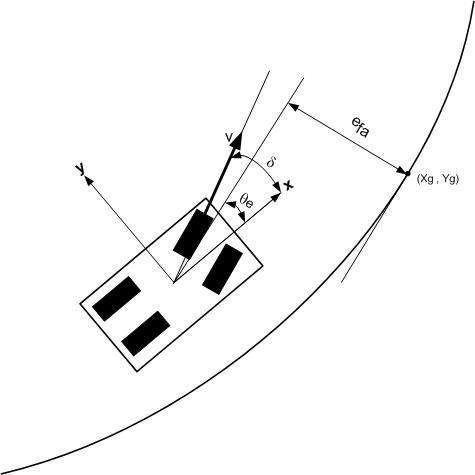


Figure 4. Lateral Controller Stanley

앞서 설명한 Line Searching에서 도출한 라인 값들의 위치 차를 이용하여 차량의 현재 가로축 위치를 도출해냈다. 이를 이용하여 차량이 도로의 중심을 유지하며 나아갈 수 있도록 Simulink의 Lateral Controller Stanley 블록을 사용하였다. 이 블록은 Stanley Method를 사용하여 주행 경로를 이탈하지 않도록 도와준다. 현재 자동차의 위치(x, y), 경로의 위치(Xg, Yg), 현재 자동차의 주행 방향(Θe)와 향후 경로 상의 주행 방향, 그리고 차량의 속도(v), 그리고 servo 모터의 최대, 최소 각을 계산하여 차량의 방향을 servo 모터의 값으로 도출해준다. (k는 속도 등으로 servo 모터에 가해지는 힘의 비례 상수)

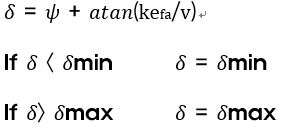


Figure 5. Stanley Method

* + 1. MATLAB을 이용한 제어 알고리즘 구현

2.1.2.1. Line Searching

Line Searching 알고리즘을 효과적으로 적용하기 위해서는 실시간으로 받아오는 Line Scan Camera의 데이터를 상황별로 분석하는 것이 중요하다. 따라서 우리 팀은 블루투스 모듈 HC-06을 연결하여 세 개의 Line Scan Camera 값을 MATLAB을 이용하여 실시간으로 받아들여 확인하였다. 대회 측에서 제공한 Serial Plot, 혹은 TeraTeram 프로그램을 이용하여 실시간으로 주기에 따른 카메라 값을 받아올 수 있었지만, 라인의 한 픽셀이 x축으로 고정된 상태에서 y축에 해당하는 라인 스캔 데이터 값을 볼 수 없어서, MATLAB을 이용하였다. ADC를 통해 받은 Raw 데이터를 받아오는 것이 아닌, 위에서 말한 Filter를 거친 값을 받아오게 하였고, 아래 사진 속 Figure 1은 세개의 Line Scan Camera 중 양쪽 끝에 있는 카메라 값을 한 개의 라인으로 모아서 출력한 그래프이고, Figure 2는 중앙에 있는 Line Scan Camera 값을 출력한 그래프로써, 명확하게 라인이 검출되는 것을 확인할 수 있었고, 이를 분석하여 효율적인 알고리즘을 구현할 수 있었다.

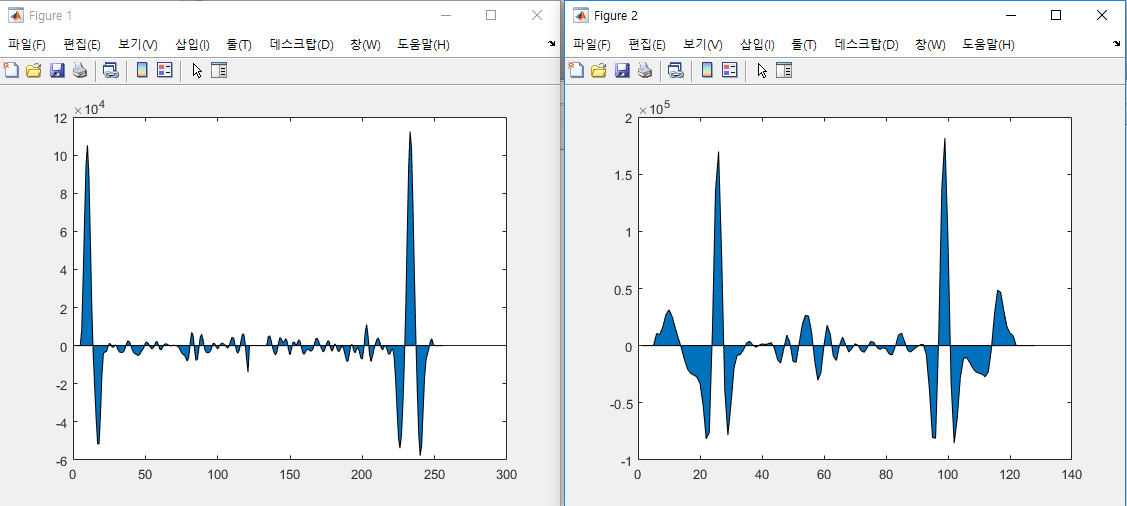


Figure 6. Line Searching

2.1.2.2. Smart Cruise Control

이론을 바탕으로 알고리즘을 구현하기 위하여 대회 측에서 제공하는 Aurix Simulink model예시를 활용, controller 부분을 수정하여 차량 속도의 정밀 제어와 Lane keeping control system을 구현하였다.

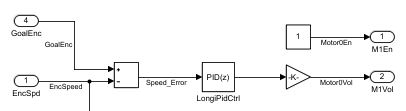


Figure 7. 속도 유지 시스템

속도 유지 시스템의 PID 모델은 기존 예시에 나와있던 종방향 모델을 수정하여 제작하였다. 실제 차량에 적용을 시킬 때는 차량 내부 함수에서 목표 인코더 속도 값을 받아와 현재 인코더 스피드 값과의 차이를 PID 제어를 통하여 목표 값에 접근하기 위한 PWM 값으로 변환하도록 설계하였다. 예시 모델과 우리 팀의 차량에는 값 환산에 차이가 있어, PID 값 자체에 앞서 말한 인코더와 PWM 값의 비례상수를 곱하였다. 초기 PID 값으로 테스트를 해본 결과, overshoot가 높아 차량이 급 발진하는 현상이 생겼고, 이에 overshoot를 낮추기 위하여 D 값을 조절하였다.

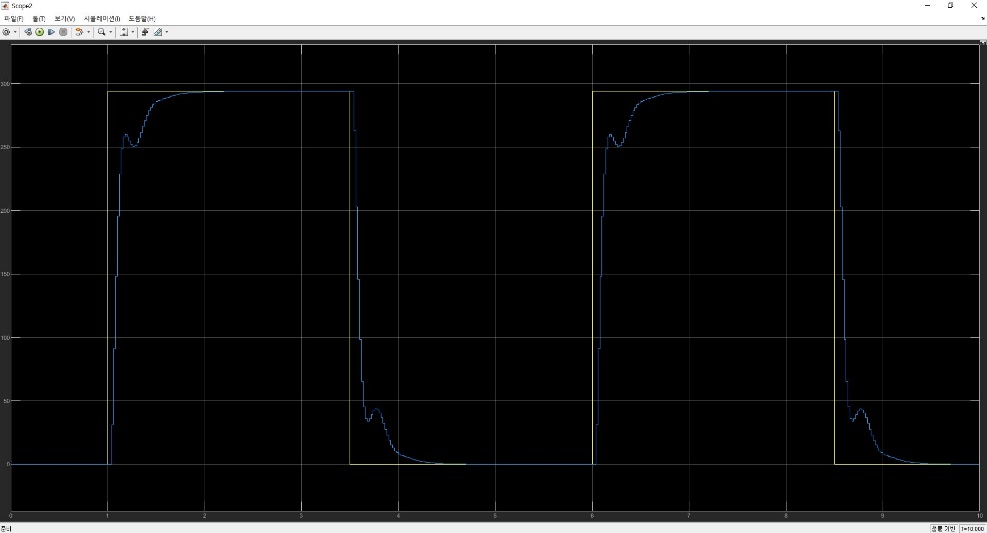
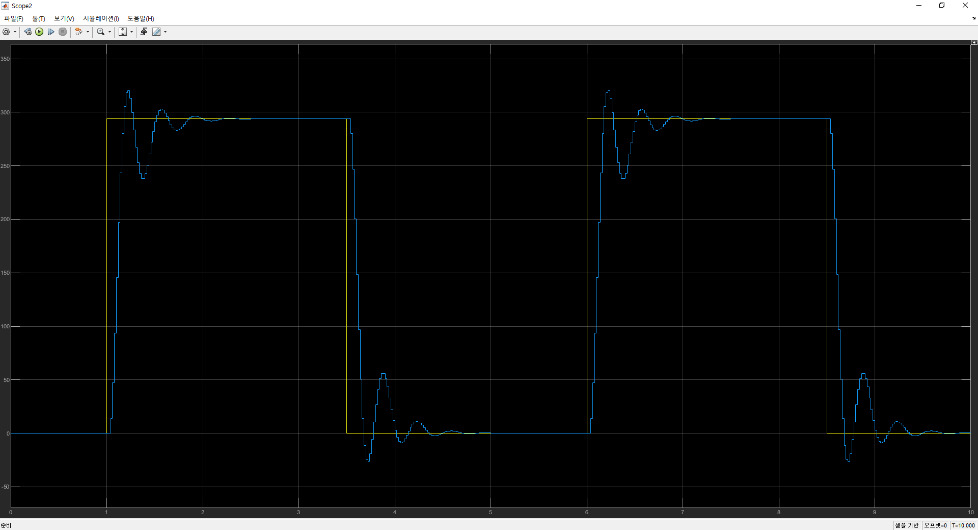


Figure 8. 속도 유지 시스템 Simulation 결과: D 값 조절 전(왼쪽) 후(오른쪽)

Lateral Controller Stanley 블록의 변수로서 현재 위치 값에는 x축은 0, y축은 Lane Offset(라인 간의 위치 차이)을 m로 계산하여 비례 상수를 곱하여 집어넣고, 차량의 방향에는 현재 servo 모터의 값을 받아왔다. 차량이 경로를 이탈하지 않게 가상의 라인을 형성해 주도록 lane keeping assist 블록을 참고하여 왼쪽에서 안 보일 때 lane 폭의 절반을 오른쪽에서 빼 주고, 오른쪽에서 안 보일 때는 lane 폭의 절반을 왼쪽에 더해주었다.

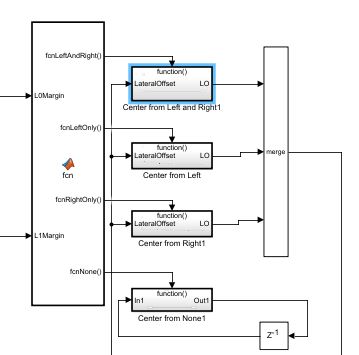
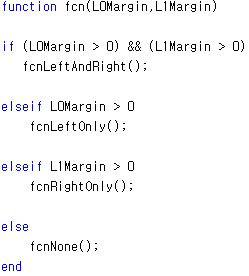


Figure 9. Lane이탈 방지 시스템

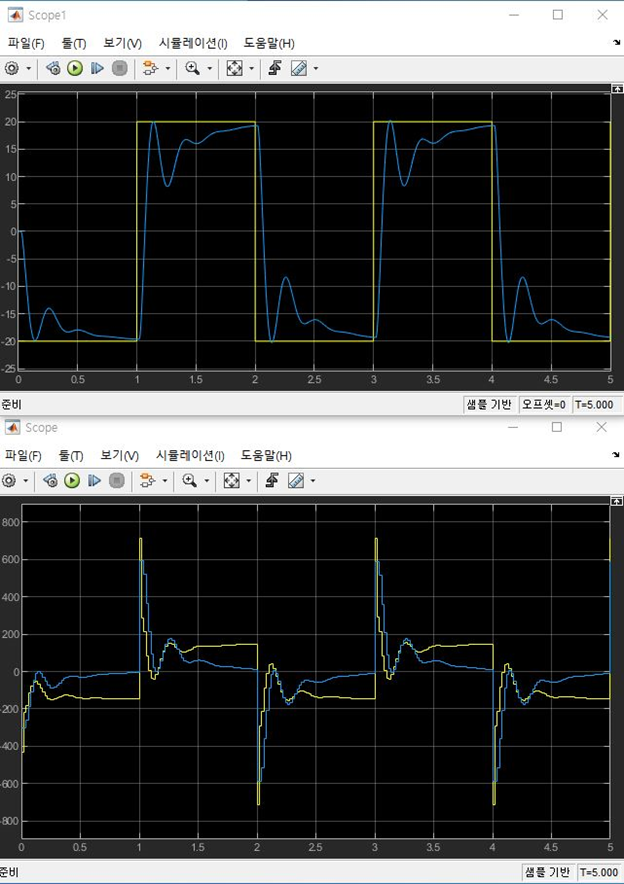
목표 위치 값의 x축은 0.01 (라인 카메라가 보고 있는 라인 지점이 차량으로부터 떨어진 거리 0.01m), y축은 0(라인 사이의 중심 지점), 차량의 방향에는 기존 예시에 나와 있던 횡방향 모델을 활용, Lane Offset을 PID 제어하여 라인 간의 중심으로 가기 위한 servo 값으로 변경하여 비례 상수를 곱한 값을 집어넣었다. 처음 예시 모델을 활용하였을 때 servo의 반응 속도가 느려 I와 D의 값을 최소한으로 조절하였다.

Figure 10. Lateral PID 결과

차량의 방향은 직진으로 설정하고, 그 차량 바퀴 간의 거리, servo 최대 각, position gain(Stanley Method의 비례 상수 k에 해당) 등을 입력하였다. 실제 주행을 해보았을 때, k의 값이 클수록 servo 모터의 각이 급격하게 변하고, 주행에 자체에 문제가 생기었다. 이에, k 값을 낮추며 적절한 k 값을 구하여 안정되게 주행이 가능하도록 하였다.

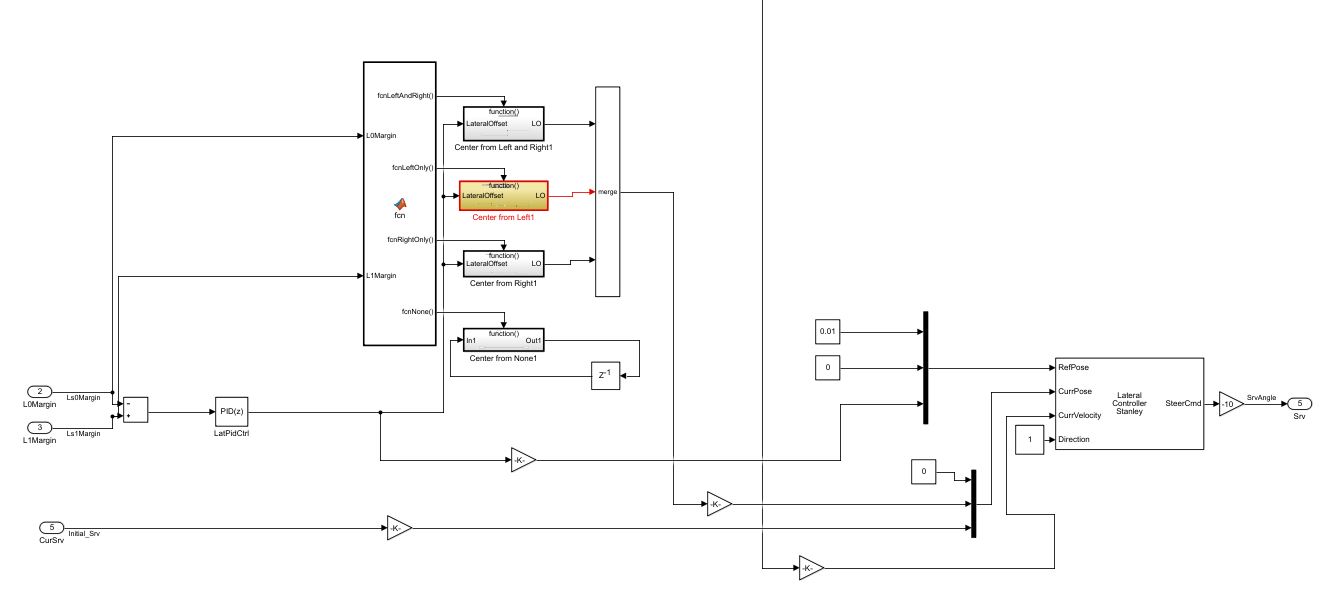


Figure 11. Lane Keeping Control 시스템

2.2 하드웨어 구성

2.1.1 센서부

|  |  |
| --- | --- |
| 전자기기, 회로이(가) 표시된 사진  자동 생성된 설명  Figure 12. 센서부(상) | 전자기기, 실외이(가) 표시된 사진  자동 생성된 설명  Figure 13. 센서부(하) |

2.1.1.1 Line Scan camera

|  |  |
| --- | --- |
| 실내, 전자기기, 벽, 사람이(가) 표시된 사진  자동 생성된 설명  Figure 14. Line Scan camera (Middle) | Figure 15. Line Scan camera (Left, Right) |

Left, Right에 쓰인 카메라는 왼쪽 오른쪽 차선을 판단하는 데에 쓰이며 가운데 카메라는 좀더 멀리 바라보며 차선을 한 번 더 인식한다. Line camera는 128개의 배열로 값을 받아오며 밝은 픽셀이 어두운 픽셀보다 값이 크다. 따라서 이를 이용해 라인을 탐지할 수 있다.

2.1.1.2 PSD(적외선 센서)

|  |
| --- |
| 실내, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 16. PSD(전면) |

이 센서는 5V를 공급해주고 입사각에 따라 전류가 변하며 출력전압을 내보낸다. 이 때 20cm~150cm를 감지할 수 있어 좀 더 일찍 차량이 장애물, AEB, 언덕 등을 판단할 것 같아 선택했다. 이 PSD의 출력 값을 MCU로 받아 일정 값 이상, 이하일 때를 나눠 AEB와 장애물을 판단한다.

2.1.1.3 엔코더

|  |  |
| --- | --- |
| 사람, 오토바이, 개체, 남자이(가) 표시된 사진  자동 생성된 설명  Figure 17. 엔코더(전면) | 실내, 오토바이, 검은색, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 18. 엔코더(상) |

Autonics 사의 E30S Series 로터리 엔코더를 이용하여 차량의 속도를 측정하는데 이용하였다.

2.1.2 전원부

|  |  |
| --- | --- |
| 전자기기, 회로, 테이블, 실내이(가) 표시된 사진  자동 생성된 설명  Figure 19. 전원부(전면) | 전자기기, 회로이(가) 표시된 사진  자동 생성된 설명  Figure 20. 전원부(후면) |

모터드라이버를 사용하게 되면서 보드에서 나오는 VCC를 직접적으로 사용하는 것이 불가 해져 배터리에서 나오는 7.2V를 레귤레이터를 이용해 각각 3.3V. 5V로 변환시킨 후 이를 센서부로 전달한다.

2.1.3 구동부

|  |
| --- |
| **Figure 21. 구동부(전면)** |

DC모터를 사용하여 차량이 앞, 뒤로 나아갈 수 있도록 함.

2.1.4 조향부

|  |  |
| --- | --- |
| 개체이(가) 표시된 사진  자동 생성된 설명  Figure 22. 조향부(상) | 머리장식, 헬멧, 의류이(가) 표시된 사진  자동 생성된 설명  Figure 23. 조향부(하) |
| 오토바이, 검은색이(가) 표시된 사진  자동 생성된 설명  Figure 24. 조향부(측면) | |

장애물이나 AEB등의 경우 각각의 상황에 맞게 바퀴의 방향이 바로 바뀔 수 있도록 하는 Servo Motor를 찾고 설치하였다.

2.1.5 전체외관

|  |  |
| --- | --- |
| 실내, 천장, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 25. 전면(1) | Figure 26. 후면 |
| Figure 27. 측면(1) | 실내, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 28. 측면(2) |
| 재봉틀, 기기이(가) 표시된 사진  자동 생성된 설명  Figure 29. 상부 | Figure 30. 전면(2) |

3D프린터를 이용한 틀에 보드와 모터드라이버, 전원부, 센서부를 쌓아 올리는 구조로 설계하였고 안정적인 동작을 위해 MCU 보드와 모터드라이버를 결합하였다. 또한 Line Scan camera와 PSD의 경우 기둥위에 3D프린팅을 붙여 위쪽의 판과 기둥에 각각 위치하도록 하였다.

3. 결론 및 토의

먼저 장애물 회피, AEB, 언덕의 경우 적외선 카메라를 3개를 달아 각자에 해당하는 알고리즘을 찾기 위해 특징을 찾아보았고 이론적인 값이 아닌 잠시 다른 값이 나올 수 있기 때문에 딜레이가 필요한 것을 함수를 두 번 실행시켜 확인하는 방법을 사용했다. 따라서, 장애물을 피해야 하는 경우 여러 번의 실험을 거친 servo motor의 조향 각 대로 움직이고, AEB의 경우 알고리즘에 충족된다면 서서히 속도를 줄여 멈추며 언덕의 경우 속도를 높이는 방향으로 차량의 속도와 방향을 조절하였다.

차선 인식의 경우 일반적인 1차선과, 2차선 그리고 횡단보도와 곡선으로 크게 나눠볼 수 있었다. 2차선의 경우 가운데 점선, 횡단보도는 굵은 선이 여러 개임을 곡선은 곡률이라는 각자의 특징들을 가지고 판단을 하는 조건으로 세웠다. 비록 조건을 구하는데 있어서 카메라의 값을 어떤 부분부터 참값이라고 해야 하는지에 대한 고민은 여러 번의 실험을 통해 점점 정확도를 높여갔다. 이 때 속도제한구역의 경우 볼록한 지점이 몇 번 나오는지를 계산한 후 해당 여부를 판단하는 형식으로 알고리즘을 만들었다.

알고리즘을 만들고 개선하면서, 대회에서 제공된 MATLAB이 상당한 도움이 되었다. 제공되는 프로그램인 UDE와 Shell 명령을 통한 Tera Term, SerialPlot을 통해 볼 수 있는 라인스캔 카메라 값은 실시간으로 변하는 과정에서 값을 확인하고 판단하기에 가독성이 매우 떨어졌지만, MATLAB과 블루투스 모듈 HC-06을 이용하여 라인 스캔 카메라의 데이터를 픽셀 별로 실시간으로 받아올 수 있었고, 대회 측에서 제공하는 Aurix Simulink Model을 이용한 PID 제어를 통하여 DC 모터를 더 정밀하게 제어할 수 있게 되었다. 학부 수준에서 다루는 보드보다 수준 높은 보드와, 실제 상황과 예상되는 상황에 맞춰 제어기를 수정해 나가고, 예상되지 않았던 상황에 대해 문제를 수정하고 진행하는데 많은 어려움이 있었지만, 이론과 실제의 차이를 줄여 나가는 과정에서 다 같이 고민하고 해결하면서 더 높은 자신감과 뿌듯함을 느낄 수 있었다. 현재 많이 발전되고 있는 지능형 자동차에 대해 많은 기술들을 미리 공부해볼 수 있었으며 자동차의 시스템이 돌아가는 전반적인 이해를 하게 되는 좋은 경험을 가질 수 있었다.

Appendix – 부품 목록

|  |  |  |  |
| --- | --- | --- | --- |
| **제조사** | **부품명** | **수량** | **사용목적** |
| TAOS | TSL1401-DB | 2 | Line Scan camera. Line을 탐지한다. |
| TAOS | TSL1401-CL | 1 | Line Scan camera. Line을 탐지한다. |
| Sharp | GP2Y0A02YK | 3 | IR 거리센서. 장애물, 언덕, AEB를 판단. |
| Tower-Pro | MG995 | 1 | 차량 조향조절. |
| Autonics | E30S Series | 1 | 엔코더, 차량의 속도 계산 |
| REEDY | AAK694 Wolf Pack 7.2V 3000mAh Ni-MH | 1 | 전원 공급 |
| Mabuchi motor | DC motor(IFX007T) | 1 | 차량 구동조절 |
| Infineon | Motor driver  (IFX007T) | 1 | Servo motor와 DC motor의 방향과 속도 제어를 위해 사용 |
| Infineon | TC275 | 1 | Main board로써 data를 읽고 분석등 담당. |
| Microchip Technology | LM2575- | 2 | 일정전압 유지 |
| Tower-Pro | MG995 | 1 | 서보혼 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |