2019**지능형모형차 경진대회 보고서**

|  |  |
| --- | --- |
| **학 교** | 연세대학교 |
| **팀 명** | SBTM |
| **유 형** | 본 경기 |
| **팀 장** | 이민혁 (전기전자공학부) |
| **팀 원** | 서규식 (생활디자인학과)  신재호 (전기전자공학부)  하진호 (컴퓨터과학과) |

1. 개요

1.1 설계 배경

인간이 생활의 편의를 추구함에 따라, 자율주행차에 대한 관심은 증가하고 있다. 미래에는 자동차가 단순한 교통수단을 뛰어넘어 움직이는 생활공간으로 자리 잡을 것이라는 예측도 있다. 이에 SBTM팀은 자율주행차에 크게 매료되었고, 카메라와 센서를 이용해 차선과 장애물을 감지하고, 컴퓨터를 이용하여 차량을 무인으로 제어하는 자율주행차량을 직접 제작해 보고 그 무궁한 가능성을 느껴보고자 이번 경진대회에 참가하였다.

1.2 설계 목표

본 대회의 목표는 자율형 주행 모형차가 트랙을 따라 주행하고, 주행 중 주어진 미션들을 수행하면서 트랙을 완주하는 것이다. 이에 SBTM팀은 다음과 같은 설계 목표를 가지고 설계에 임하였다.

1. MCU를 통해 카메라 영상 값을 받아온다.
2. Line sensing camera로 얻은 행렬 데이터를 처리하여 라인들을 검출하고, 이를 통해 차량이 라인의 중앙을 따라 자율 주행하도록 서보 모터에 신호를 준다. 이 과정에서 PID control를 통해 error를 최대한 줄인다.
3. 속도제한구간의 시작 라인을 검출하면 DC모터를 조작하여 속도를 0.7m/s ~ 1m/s로 늦추고, 끝 라인을 검출하면 다시 속도를 높인다.
4. 적외선 센서를 이용해 차량 전방의 장애물을 인지하고 상황에 맞게 정지 혹은 회피 미션을 수행한다.
5. 장애물회피 시에는 현재 차량이 위치하고 있는 차선을 파악하고, 그에 따라 회피방향을 결정한다.

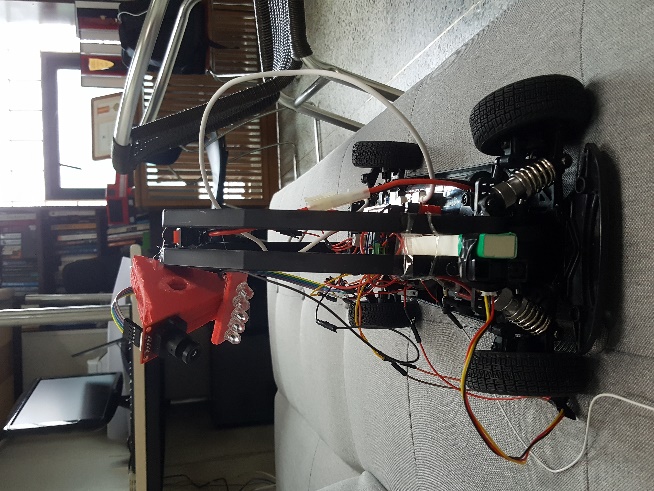
2. 설계 내용

2.1 하드웨어 구성

2.1.1 하드웨어 시스템 구성

하드웨어 시스템 구성은 크게 MCU, 조향부, 센서부, 구동부, 전원부로 구성되어 있다. 모든 부는 MCU에서 제어되며 센서부에서는 적외선 센서를 통한 장애물 감지나 카메라를 통한 Line sensing 작업을 하고 센서부에서 받은 값을 통해 MCU에서 연산을 통해 조향부가 기체의 방향을 결정한다. 구동부에서는 Motordriver가 모터의 출력을 조절하여 차체의 속도를 조절한다. 그리고 전원부에서 모터, MCU, 센서에 전원을 공급한다.

2.1.2 전체 외관 사진



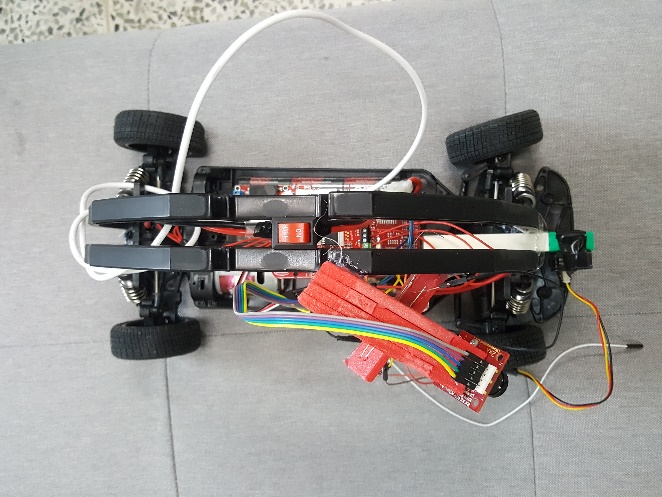


Figure 1. 정면 Figure 2. 윗모습

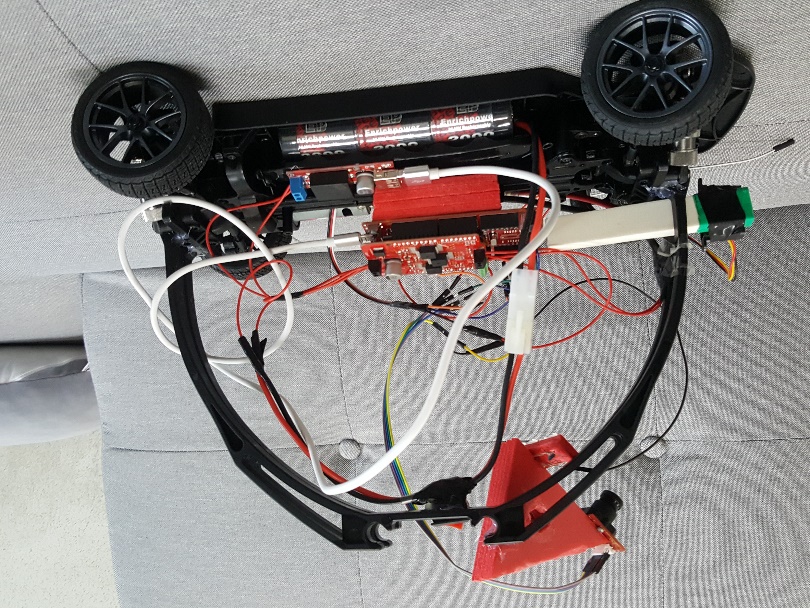


Figure 3. 측면

2.1.3 전원부

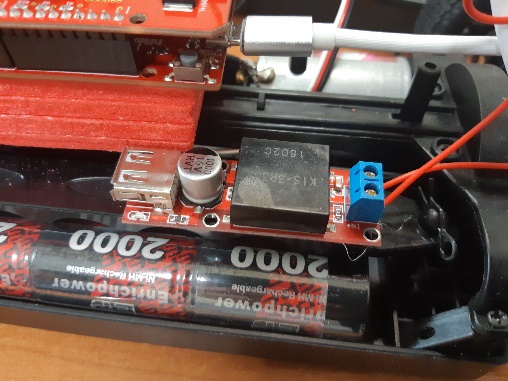


Figure 5. 5V 스텝다운 컨버터와 NiMH배터리

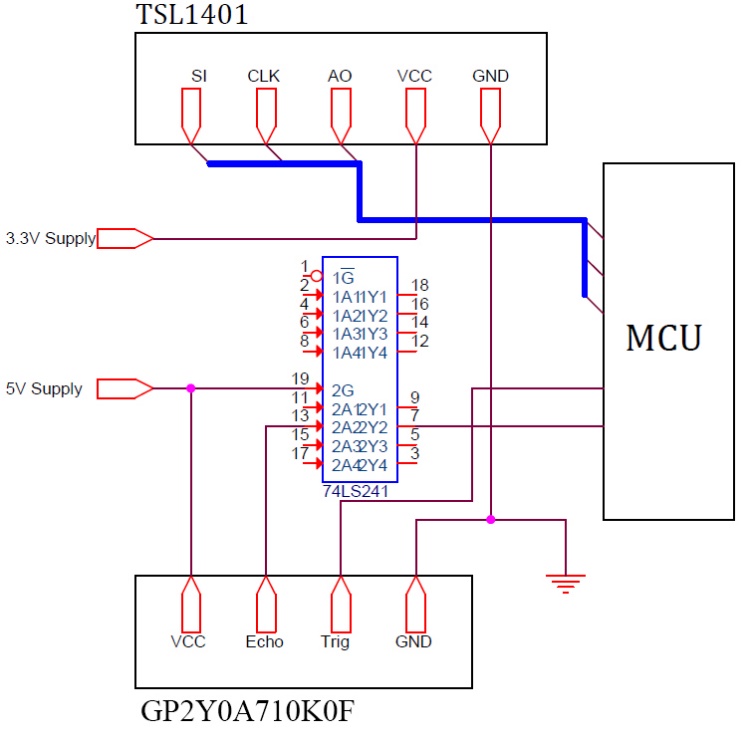
****

Figure 6. 전원 공급 회로

전원부의 전원은 NiMH 2000mAh 7.2V 배터리를 사용하였다. 7.2v를 제공된 모터드라이버에 공급해주고 그것을 5v step down converter를 이용하여 MCU에게 5V의 전압을 공급하였다. 또한, 점퍼 세팅을 통해 조향부에 3.3V의 전압을, 센서부에 5V의 전압을 공급하였다. MCU는 입력과 출력이 5V이다. 센서부의 카메라(TSL1401)는 입출력이 MCU와 5V로 동일하므로 MCU에 직접 연결한다. 그리고 모터드라이버 보드와 MCU 보드, 3개의 센서의 전원과 신호 연결을 깔끔하게 정리하기 위해서 전원 보드를 만들었다.

2.1.4 센서부

센서부는 크게 장애물 탐지를 위한 적외선 센서와 Line sensing을 위한 카메라로 이루어져 있다.

Figure 7. 적외선 센서 Figure 8. 카메라

장애물 탐지를 위한 적외선 센서는 GP2Y0A710K0F 를 이용하였으며 차체의 앞부분에 달아 장애물을 감지하기 쉽게 하였다. 최대 측정 거리는 5.5m로 물체와의 거리가 멀수록 펄스폭이 늘어나고 거리가 가까울수록 펄스폭이 감소하는 것을 이용하여 이를 MCU에서 감지하여 장애물과의 거리를 알 수 있게 하였다.

Line sensing은 카메라가 담당하는데 카메라는 TSL1401을 이용하였다. TSL1401은 라인센싱 카메라로 카메라의 화각을 128등분 해서 128\*1의 배열로 값을 전송해준다. 카메라를 MCU와 연결하여 카메라의 화각을 알아내고 그에 맞게 차체의 높이를 고려하여 카메라를 설치하였다. 카메라에서 나오는 배열값을 이용하여 Line sensing 알고리즘을 작성하였다.

2.1.5 조향부

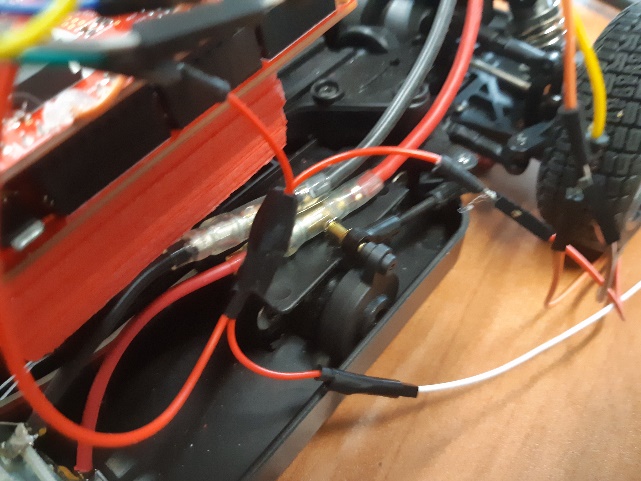


Figure 9. 서보모터

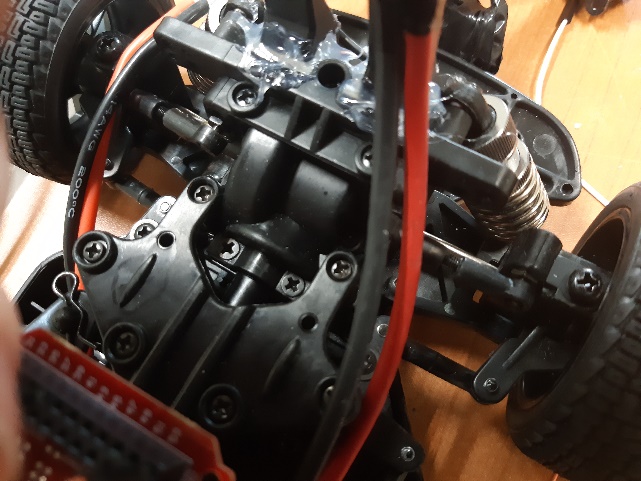


Figure 10. 조향부

차체의 조향을 담당하는 Servo Motor는 제공된 서보모터를 이용하였고3.3V를 인가해 주었다.

2.1.6 구동부

구동부는 Motor driver와 Motor로 이루어져 있다. Motor driver 같은 경우 제공된 모터드라이버 보드를 사용하였다.

2.2 소프트웨어 구성

2.2.1 Logic Blocks

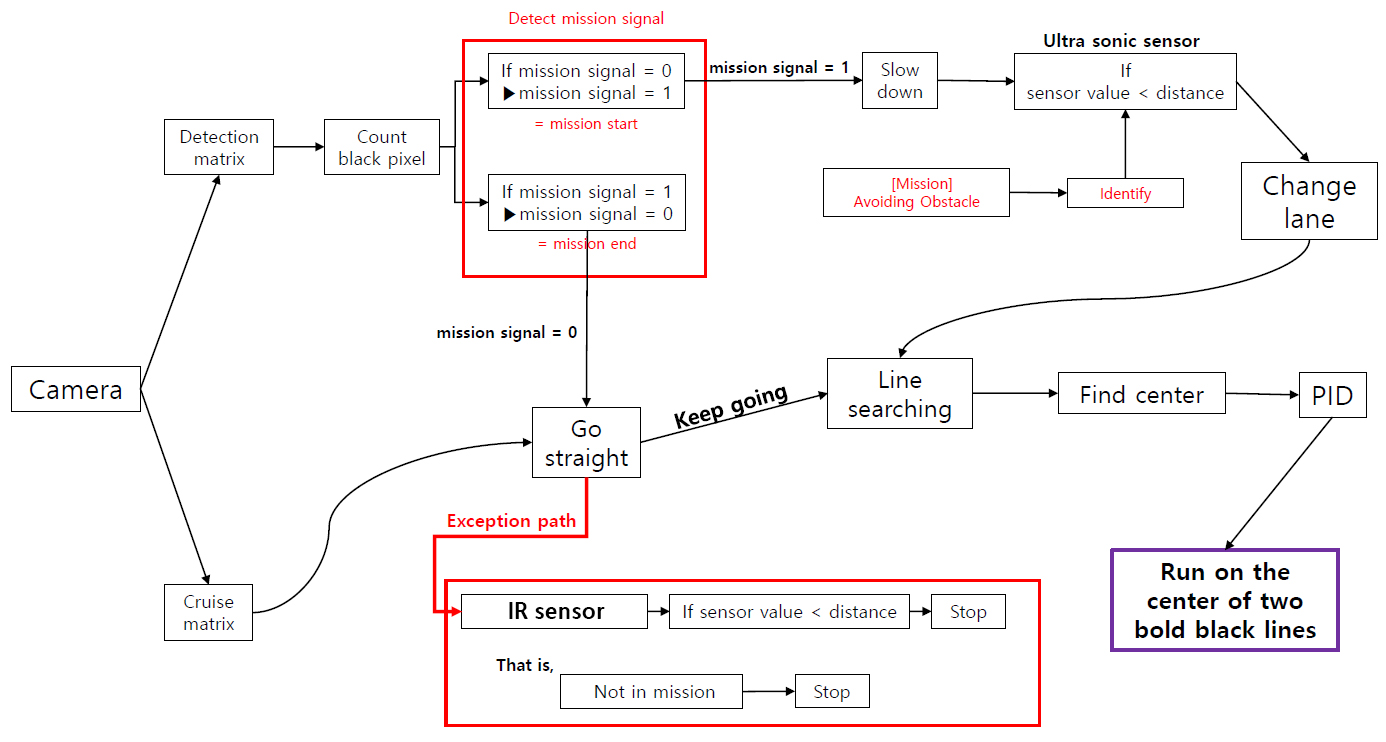


Figure 11. Logic Blocks

2.2.2 Algorithm

2.2.2.1 Camera signal processing

Lane detecting을 위해 TSL1401 모델을 이용하였다. 카메라에 CLK을 TOM을 이용해 50kHz 주기로 넣어주고 Timer를 통해 50000count를 센다. Count값이 15일 때 SI를 발생시키고 SI가 상승엣지일 때 다음 클럭부터 A0핀의 아날로그 데이터를 읽어 온다. 이때 A0핀에 128개의 아날로그 데이터가 들어오고 이를 ADC를 통해 디지털 값으로 변환한다. 이 값을 배열에 저장한 후, 일정한 Threshold값을 정하여 data를 검은색 0 흰색은 1로 변환한다.

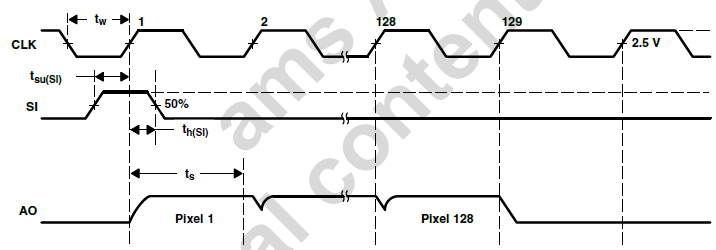


Figure 12. Clock Diagram

* + - 1. Lane detecting을 통한 안정적인 방향&속도 제어

자동차가 라인을 넘지 않은 채로 주행하기 위해서는 양 라인의 정 중앙을 끊임없이 추적하는 알고리즘이 필요하다. Lane detecting의 전체 과정을 짧게 요약하면, 1차선의 경우 화면의 중심이 양쪽 라인의 정 중앙을 향하도록 PID 제어를 통해 추적하는 것이고, 2차선의 경우 차선 위치를 확인 후 가장자리의 실선을 따라 움직이도록 하였다. 구체적인 과정은 다음과 같다.

1. Line detecting

카메라 화면의 각 데이터는 검은색이면 0, 흰색이면1로 이루어져 있다. 대회 규정상 주행 트랙이 바탕은 흰색, 차선은 검은색으로 되어 있으므로 검은색 픽셀들로 이루어진 선을 찾는 과정이 line detecting 이라고 할 수 있다. 따라서, 카메라 화면의 오른쪽부터 픽셀을 하나하나 읽어 가면서 1값을 가지는 픽셀을 찾는 방식을 통해 차선을 인식한다. 2차선의 경우 차의 위치가 오른쪽이면 오른쪽 실선만, 왼쪽이면 왼쪽 실선만 읽어 차선을 인식한다. 1차선은 두 차선을 모두 읽는다.

1. 1차선과 2차선 line detecting Method

2-1) 1차선

Figure. 11, 12를 통해, 양 라인의 중앙과 현재 카메라 화면의 중앙 픽셀이 차이가 나는 상황을 알 수 있다. 현재 라인의 중앙을 빨간 공으로 표현하고, 현재 카메라의 초점을 점선으로 표현하였다.



Figure 13. 카메라 초점이 라인의 정 중앙을 바라볼 때



Figure 14. 카메라 초점이 라인의 우측에 있을 때

Figure. 11을 보면 카메라의 초점(점선) 과 라인의 중심(빨간 공) 이 일치한다. 즉, 카메라가 라인의 정 중앙을 바라보고 있는 것 이다. 따라서 자동차의 방향을 바꿀 필요는 없다. 그러나 Figure. 12를 보면, 카메라의 초점(점선) 과 라인의 중심(빨간 공) 이 일치하지 않는다. 카메라의 초점이 라인의 중심 오른쪽에 잡혀있는 상황이다. 또한 이 둘의 pixel 차이는 Error 라는 변수 값으로 매 순간 확인할 수 있게 된다. Error 값이 양수이면, 카메라가 라인 중심의 오른쪽을 보고 있는 것이고(왼쪽으로 틀어야 함) Error 값이 음수이면 그 반대의 경우일 것이다.

2-2) 2차선

2차선의 경우 1차선과 달리 line detecting 과정에서 한쪽 차선이 실선이 아닌 점선이 되어 문제가 발생한다. 점선의 흰 부분만 들어오게 된다면 라인을 읽을 수 없어 오류가 발생한다.  
이 때문에 Figure. 13과 같은 상황을 만들어 보았다.

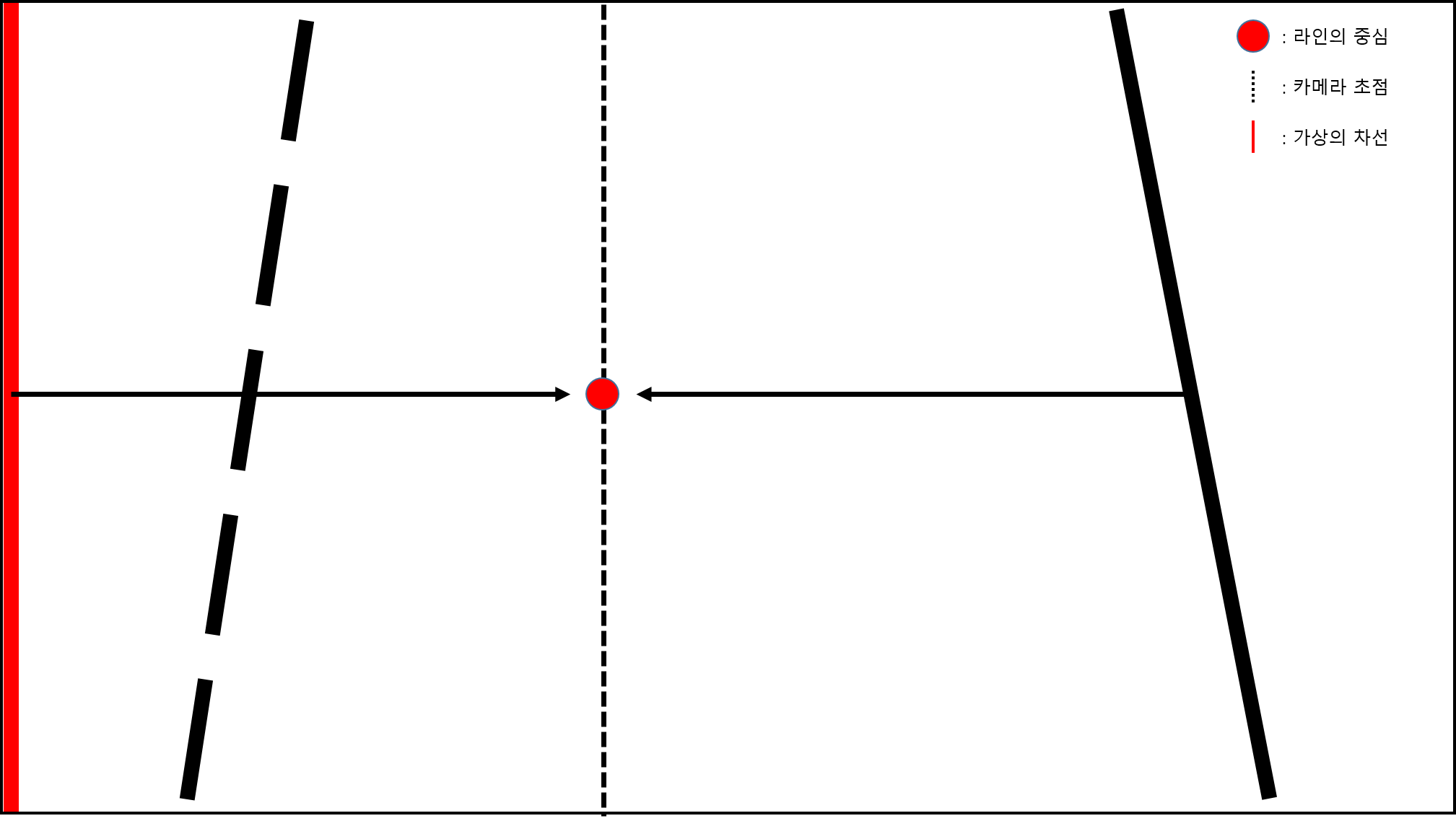


Figure 15. 차선에서 차가 오른쪽 차선에 있을 때

1차선과 같은 방식으로 오른쪽에서부터 카메라의 값을 읽어온다. 이때 실선인 한쪽 차선만 읽어오는데, 차선이 위 그림과 같이 차가 오른쪽 차선에 있다면 카메라로 받는 데이터 행렬 중 비교적 앞쪽에 검은 pixel이 인식이 될 것이고, 차선이 왼편에 있다면 비교적 데이터 행렬 뒤쪽에 검은 pixel이 인식될 것이다. 이제 다른 하나의 차선이 필요한데, 위 그림과 같이 차가 오른쪽 차선에 위치하면 오른쪽 실선만 읽기 때문에 왼쪽 실선이 필요하게 된다. 이 경우 데이터 행렬의 끝부분에 임의로 검은 pixel을 인식한 것처럼 값을 넣어 가상의 차선을 만들어 주게 되면, 이를 이용하여 1차선의 경우와 마찬가지로 차선을 따라 이동할 수 있게 된다. 차가 왼쪽 차선에 있는 경우 데이터 행렬의 가장 앞부분에 값을 넣어 만들 수 있다.

2.2.2.3 속도제한구간

1) 속도제한구간에서의 속력 조절

속도제한구간의 시작을 알리는 굵은 선이 나타나면 카메라가 보내는 모든 데이터가 검은색을 나타나기 때문에, 검은색 데이터 개수가 일정 값 이상으로 증가하면 속도제한구간으로 인식하고 속도를 낮춘다. 그리고 두 번째로 굵은 선을 감지하면 속도제한구간이 끝났음을 의미하기 때문에 속도를 다시 높인다.

2) 속도제한구간이 끝날 때의 주행로 결정

속도제한구간이 끝났을 때 차량이 병합되는 차선에 있다면 주행로를 바꾸어야 한다. 따라서 현재 차량이 어느 차선에 있는 지를 구분하기 위해 우측차선의 line\_value를 0으로, 좌측차선의 line\_value를 1로 설정하였다. 초기 line\_value를 0으로 설정하고 차선을 바꿀 때 마다 line\_value는 바뀐다.

두번째 굵은 선을 감지했을 때, line\_value가 0이면 그대로 주행하고, line\_value가 1이면 차선을 우측으로 옮겨서 주행한다.

2.2.2.4 장애물회피

1) 장애물의 회피와 비상 급제동의 구분

적외선 센서를 통해 장애물을 인지하면 다른 차선으로 장애물을 회피하여 주행하는 미션이다. 하지만, 자율비상제동도 역시 적외선 센서를 통해 장애물을 인지한다는 점에서 자율비상제동과 어떻게 구분할 지가 난관이었다. 이에 장애물 회피는 속도제한구간 내에서 이루어지는 미션이라는 것에 착안하여 state를 통해 속도제한구간 안에서 장애물을 인지하면 회피, state가 일반 주행일때 장애물을 인지하면 정지하도록 알고리즘을 구성하였다.

2) 장애물 회피 방향의 결정

우측차선에 있을 때 적외선 센서를 통해 장애물을 인지하면 좌측차선으로 이동하여 장애물을 회피하고, 좌측차선에 있을 때 장애물을 인지하면 우측차선으로 이동하여 장애물을 회피하는 미션이다. 따라서, 현재 차가 우측차선에 있는 지, 좌측차선에 있는 지를 구별해주고 그에 따라 회피 방향을 결정하는 식으로 알고리즘을 구성하였다.

우측 차선의 line\_value를 0으로, 좌측 차선의 line\_value를 1으로 초기설정을 한다. 그리고 장애물을 감지했을 때 line\_value가 0이면 좌측으로 이동하면서 line\_value를 1로 바꾸고, line\_value가 1이면 우측으로 이동하면서 line\_value를 0으로 바꾼다.

2.2.2.5 자율비상제동 AEB(Autonomous emergency braking)

자율 비상 제동은 적외선 센서를 통해 거리를 감지함으로써 수행한다. 대상 물체에 닿지 않은 체로 정지 하는 것이 목표이므로, 물체를 인지하자 마자 속도를 빠르게 줄이는 것이 관건이다. 알고리즘은 간단한데, 적외선 센서에 의해 일정거리 안에 장애물이 인지되면 모터 출력을 0으로 바꾸는 것이다.

3.1 PID Controller 이론 (Lane 의 중심을 향하도록 하는 알고리즘에 사용)

3.1.1 PID Block Diagram

위에서 언급하였듯이, 양쪽 lane의 중심을 향하도록 하는 조향 알고리즘에 PID 제어를 사용하였다. PID 제어는 쫓고자 하는 값과 현재 값 차이를 구한 뒤, 이것을 비례, 적분, 미분 한 각각의 값을 구하여 행동에 반영하는 것이다. 아래 그림은 PID 제어기의 대표적인 block diagram 이다.

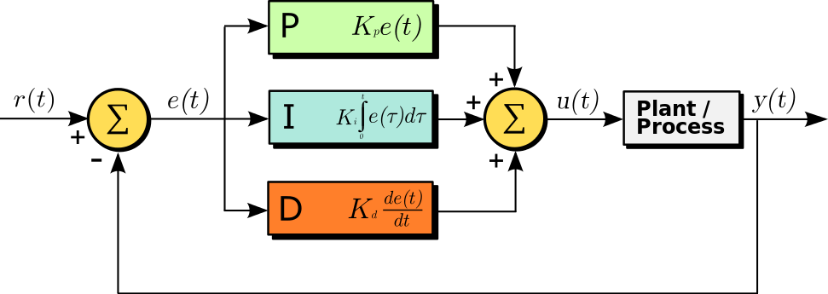


Figure 16. PID Block Diagram

*r(t)* 는 추종하고자 하는 신호이고, *y(t)*는 현재 상태를 의미한다. 제어부의 왼쪽부터 보면, 먼저 추종하고자 하는 값과 현재 상태의 차이인 *e(t)*를 구한다. 그 후, 이 값을 각각 비례(P), 적분(I), 미분(D) 한 값을 더하여 제어신호인. *u(t)* 를 만들어 낸다. 이때, P I D 제어에 각각 사용되는 상수 값인 는 PID 제어 각각에 해당하는 Weight를 나타낸다. 즉, 변화가 심한 환경일 경우에는 D제어의 Weight를 크게 하는 것이 맞고, 변화가 없는 잔잔한 환경일 경우에는 I제어의 Weight를 높여 축적되는 에러를 방지하는 것이 상수 값을 정하는 방법이다.

3.1.2 Transfer function & stability analysis

수학적으로, PID 제어기의 Transfer function과 Closed Loop Transfer function는 다음과 같다.

시간영역 PID 제어신호 알고리즘: 

PID 제어신호 Open loop 전달함수: 

Closed loop PID 제어기 전달함수: 

: Plant transfer function, : PID transfer function

만약 어떠한 s에 대하여 위에서 구한 Transfer function H(s) 가 발산하게 된다면, 위의 PID 제어기는 잘못 설계된 것이다. H(s)가 발산하는 상황은 의 frequency shift 가 180 일 때의 Gain 의 절대값이 1이 넘는 경우이다. 이때 Loop 특성에 의하여 신호가 무한히 증폭되며 결과적으로 *y(t)* 가 발산하게 된다. 하지만, 현실적으로는 이론적인 *G(s)*를 구하는 것이 사실상 불가능 하기 때문에, 실험적인 방법을 통해 안정적인 K(s)를 정해 나가야 한다.

4. 결론 및 토의

4.1 Infineon 보드에 관하여

모터를 제어하기 위한 PWM 신호를 발생시키고, 카메라와 적외선 센서 등 각종 센서의 신호(펄스)를 받아 처리하기 위해 GTM Module에 대해 집중적으로 공부하였다. 이를 통해 펄스의 발생, 측정 원리에 대해 배울 수 있었다. 카메라 데이터가 아날로그이므로 ADC를 통해 값을 받았고 또한 각 Module의 처리 결과를 CPU로 알리기 위해 Interrupt를 다수 사용하였으므로, Interrupt의 처리 과정에 대해서도 공부하였다. LCD를 활용하여 주행 테스트 도중에 데이터를 실시간을 확인할 수 있게 하였고, ASC와 DMA를 통해 카메라의 아날로그 데이터를 확인할 수 있었다.

**4.2 Camera TSL1401 사용에 관한 조사**

카메라 모듈로부터 데이터를 받기 위해 timing을 맞추고, 아날로그 데이터를 받아보는 것을 자세히 공부하였다. TSL1401은 CLK과 SI에 input clock을 넣어주고 A0에서 output clock을 내보낸다. SI는 데이터를 받아오겠다는 신호를 나타낸다. SI 클럭의 상승 엣지 이후로부터 다음 클럭부터 값을 받아오면 된다.

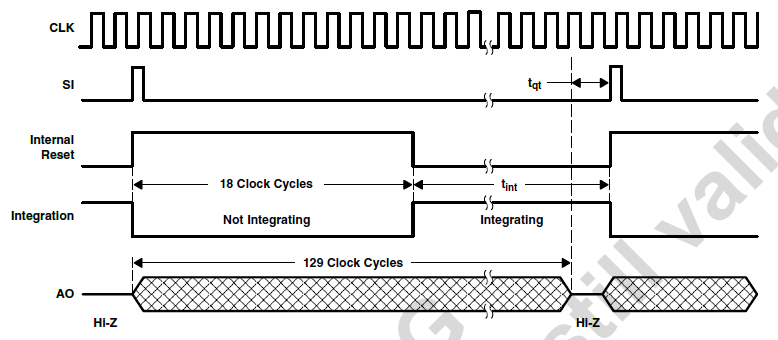


Figure 17. Clock Diagram

**4.3 Lane detecting algorithm제작**

완주를 위해서는 효율적이고 강인한 Lane detecting algorithm알고리즘을 사용해야 한다. 왜냐하면 속도가 빠르더라도 양쪽 Lane 을 넘는 순간 실격처리가 되기 때문이다. 만약 Lane detecting알고리즘이 굉장히 강건하지만, 시간이 오래 걸린다면, 빠른 속력에서 안정적인 성능을 보장하지 않을 것이다. 반대로 Lane detecting알고리즘 시간은 짧지만 강건성이 떨어진다면, 이 또한 안정적인 주행을 보장하지 못한다. 따라서 알고리즘을 제작하는 데에 적절한 속도와 적절한 robust함을 맞추기 위해 노력하였다. 주행을 안정적으로 마치기 위해 PID를 이용한 간단한 중앙 라인 추적알고리즘을 생각하게 되었고, 구체적인 코드를 작성할 수 있었다. 이후 Loop time을 확보하기위해 선택적 데이터 수신 아이디어를 생각하게 되었고, 이를 이용하여 효과적으로 loop time을 줄일 수 있었다.

[첨부양식 1]

**전자 부품 목록**

|  |  |  |  |
| --- | --- | --- | --- |
| **제조사** | **부품명** | **수량** | **사용목적** |
| Freescale | TSL1401CL | 1 | Lane detect camera |
|  | KIS3R33S | 1 | 보드 전원공급을 위한 stepdown converter |
| SHARP | GP2Y0A710K0F | 1 | 장애물 거리 측정 |
| Enrich Power | EP-TA2000 | 1 | 배터리 |
|  | Servo motor | 1 | 조향용 서보 모터 |
| Mabuchi motor | RS-540SH | 1 | 구동 모터 |
| Infineon | IFX007T | 1 | 모터 드라이버 |
| Infineon | TC275 | 1 | MCU |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

※ 보고서는 주어진 서식을 이용하되 15장 이내로 작성하여 주십시오.

※ 소스코드는 포함하지 마시고 따로 압축하여 제출하여 주십시오. (5MB 이내)

※ **보고서는 7월 5일(수요일)까지 제출**하십시오.

[첨부양식 2]

**차체 변경 목록**

**\*변경 내용 및 변경 사유**

|  |  |
| --- | --- |
| **변경내용(사진 포함)** | **변경사유** |
| **(사진)** | 1. **변경사유** 2. **변경 근거 조항** |
| **(사진)** | 1. **변경사유** 2. **변경사유 근거 조항** |
| **(사진)** | 1. **변경사유** 2. **변경 근거 조항** |
| **(사진)** | 1. **변경사유** 2. **변경 근거 조항** |

\*칸이 부족할 경우 칸을 추가하여 사용

\* 검차 시 구별가능한 사진 포함

※ 보고서는 주어진 서식을 이용하되 15장 이내로 작성하여 주십시오.

※ 소스코드는 포함하지 마시고 따로 압축하여 제출하여 주십시오. (5MB 이내)

※ **보고서는 7월 3일(수요일)까지 제출**하십시오.