2015**지능형 모형차 경진대회 보고서**

|  |  |
| --- | --- |
| **학 교** | 경희대학교 |
| **팀 명** | Wing |
| **유 형** | 본경기 |
| **팀 장** | 신동주 (컴퓨터공학과) |
| **팀 원** | 김만수 (컴퓨터공학과)  김병우 (컴퓨터공학과)  안재성 (컴퓨터공학과) |

1. 개요

1.1 설계 배경

우리가 평소에 타고 다니는 자동차들은 모두 사람이 직접 운전하는 기계적인 방식으로 작동하고 또한 이동성에만 중점을 두어 효율적으로 이동하는데 초점을 두고 있다. 하지만 국내의 ‘현대자동차’나 해외의 ‘Google’에서 발표하는 자동차들의 기술들을 살펴보면 사람이 직접 운전하지 않고 자동차가 스스로 차선을 파악하고 위험상황을 인지하여 판단하고 움직이는 무인자동차 기술들이 대다수를 차지하고 있다. 이러한 기술들은 사람들이 단순히 이동성에 중점을 두는 것이 아니라 자동차를 사람이 생활하는 공간으로 인식하고 편하게 자동차를 이용할 수 있는 것에 중점을 두고 있다.

여러 회사들에서 발표하는 자동화되어 있는 기술들을 우리가 직접적으로 다룰 수는 없지만 해당 기술들을 실제로 모형차를 이용하여 지금까지 대학교에서 배운 내용들을 바탕으로 직접 구현해 봄으로 이에 관련한 알고리즘을 생각해보고 이를 어떻게 적용시킬 수 있는지에 대하여 고민해 봄으로써 간접적으로 체험해보고 싶어 이번 모형차를 설계하게 되었다.

1.2 설계 목표

대회 측에서 각 팀에게 제공한 개발보드(MPC5606B, FRDM-TFC)들을 이용하여 모형차를 제작하는데 목표를 둔다.

제작된 모형차에 line scan camera를 이용하여 차선을 검출하도록 하고 이에 대한 정보들을 받아와서 필요한 부분에 적절한 값을 줌으로써 자동차가 차선을 따라 이동하도록 설계한다. 그리고 적외선 센서나 Encoder 등을 이용하여 부수적인 세부조정을 한다.

이런 종합적인 데이터를 바탕으로 라인을 이탈하지 않고 라인을 따라서 주행하며 위험을 감지하여이에 대한 올바른 대처를 하도록 만드는 것을 주된 목표로 한다.

2. 설계 내용

2.1 하드웨어 구성

2.1.1 센서부

2.1.1.1 거리감지센서부(적외선 센서)

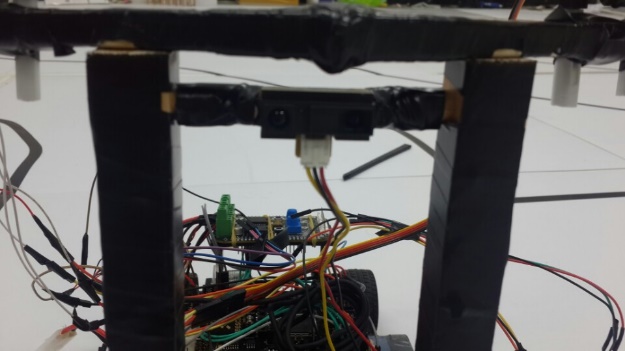
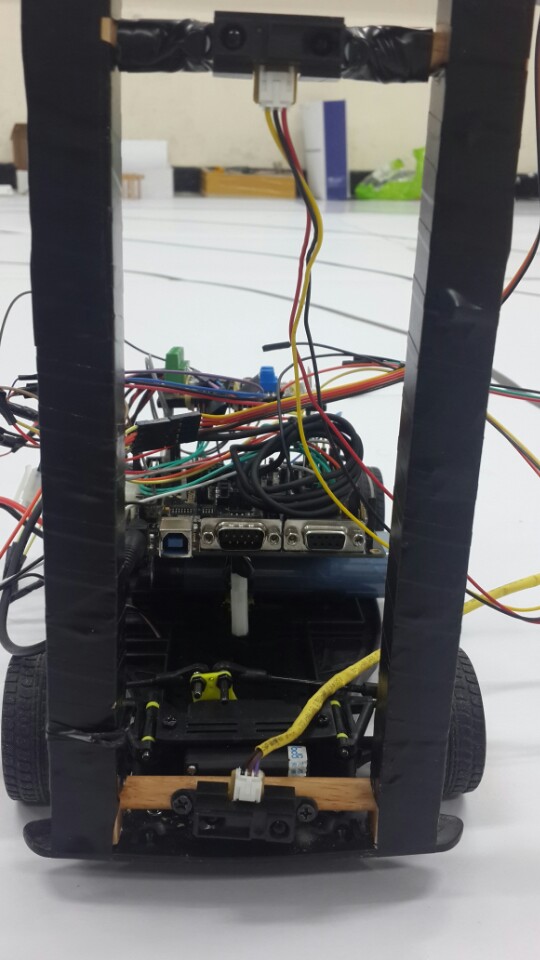


Figure 2. 상단 거리감지센서



Figure 3. 하단 거리감지센서

Figure 1. 거리감지센서부 전면

장애물 감지를 위하여 거리감지센서 2개를 사용하였다. 거리감지 센서로 사용한 센서는 Sharp사에서 나온 2YOA21F43과 0A41SKF32라는 센서이다. 해당 센서는 5V를 입력하면 거리에 따라서 0~5V를 반환하기 때문에 ADC를 이용하여 값을 받아와서 사용하는 센서이다. 거리에 따라서 다른 값을 반환하기 때문에 이번 대회의 트랙에서 장애물에 대한 처리가 가능할 것으로 예상되어 사용하였다. 이 센서는 최대거리 80cm까지 파악이 가능하기 때문에 미리 파악하고 장애물에 대한 대처를 할 수 있다.

2.1.1.2 Line Scan Camera

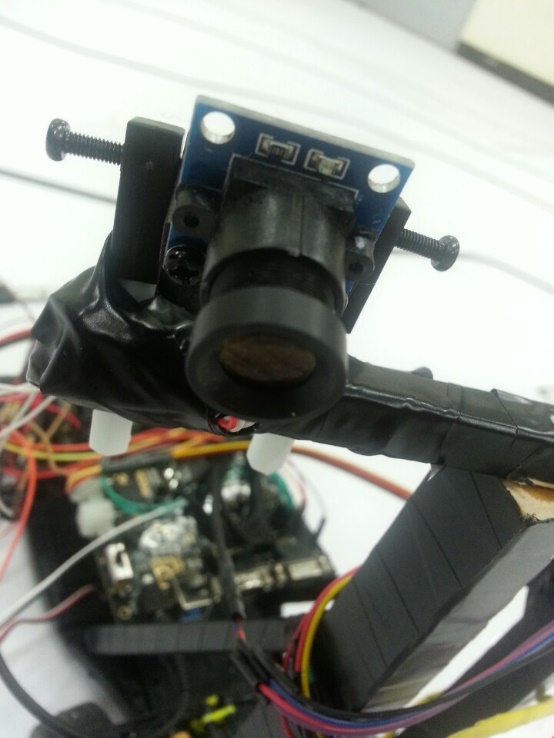
 

Figure 4. 우측 카메라 Figure 5. 좌측 카메라

장애물 감지를 위하여 Line Scan Camera를 사용하였다. 각각의 카메라는 Freescale에서 기본적으로 제공한 Line Scan Camera와 TSL-1401이라는 모델 두 가지를 사용하였다. 각각의 카메라는 명도에 따라서 값을 0부터 1024까지 반환하며 이를 이용하여 현재 카메라가 보고 있는 부분이 라인을 보고 있는지, 보지 않고 있는지에 대한 정보를 받아서 자동차를 라인에 맞춰 주행할 수 있도록 도와준다.

2.1.2 구동부

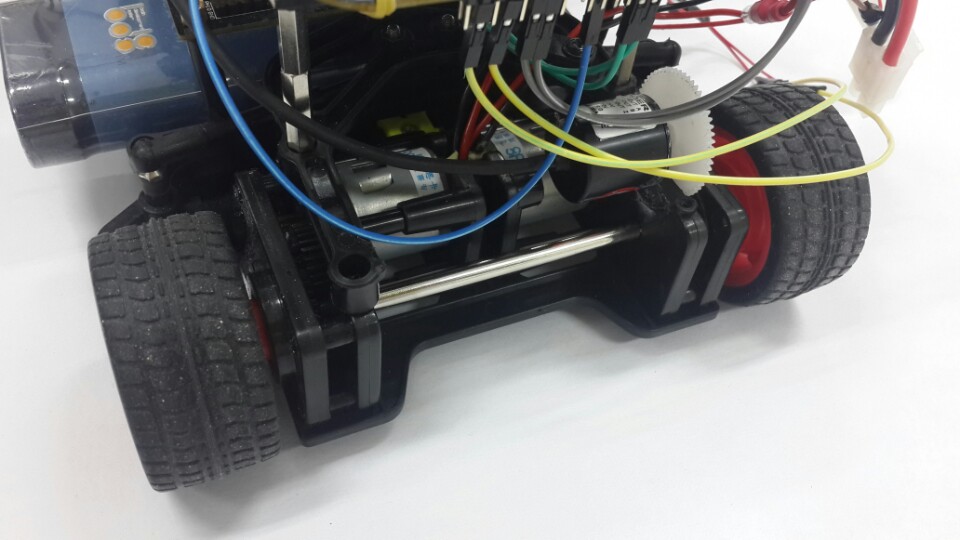


Figure 6. 구동부 전면

모터보드는 대회에서 지급한 FRDM-TFC보드를 사용하여 DC모터에 전원을 공급하여 사용할 수 있도록 하였다. 각각의 DC모터에 7.2V씩 나누어서 공급하기 때문에 양쪽에 원활하게 전원 공급이 가능한 보드이다.

Encoder는 Autonics사의 E20S2-200-3-N-5-R이라는 모델을 사용하였다. 이 모델은 5V를 입력하면 동작하도록 설계되어 있으며 바퀴의 회전 수를 받아와서 속도를 측정할 수 있도록 한다. 이 때 바퀴가 회전하고 있는 회전 수를 받기 위하여 모형차 뒤쪽의 차체를 개조하여 Encoder를 부착하였고, 톱니바퀴를 이용하여 Encoder가 바퀴의 회전에 따라서 동시에 회전할 수 있도록 설계하였다.

2.1.3 조향부



Figure 7. 조향부 전면

조향부에 서보모터는 Futaba사에서 제조한 S3010을 사용하였다. 바퀴방향 정면을 기준으로 좌우방향으로 45도씩의 조향을 9등분하여 servo에 주는 PWM duty cycle을 조절함으로 각 방향을 제어하였다.

. 2.1.4 전체외관

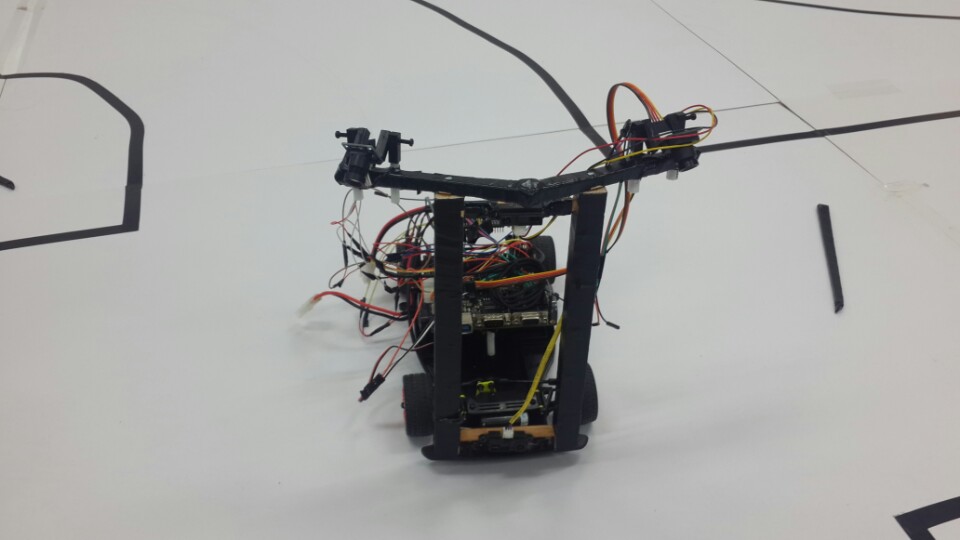


Figure 8. 전체외관 전면

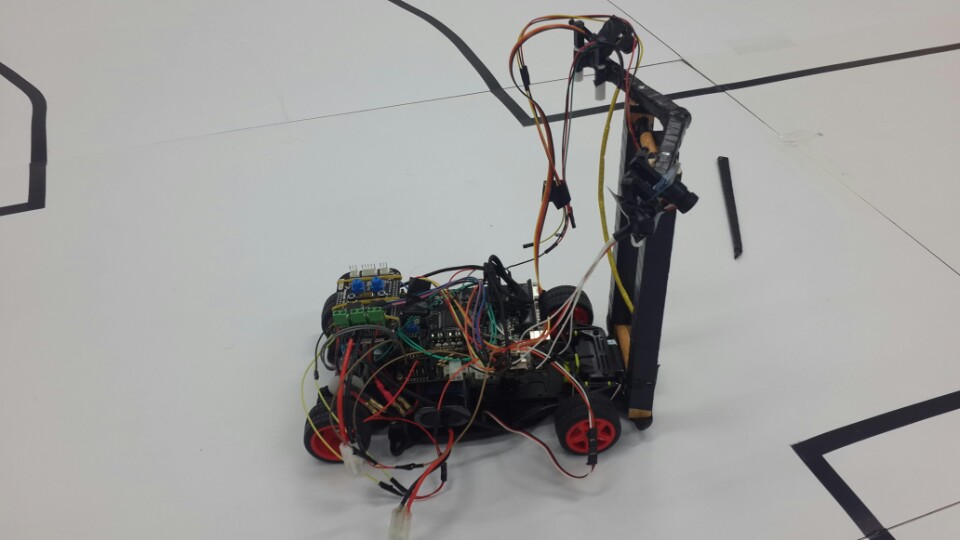


Figure 8. 전체외관 우측

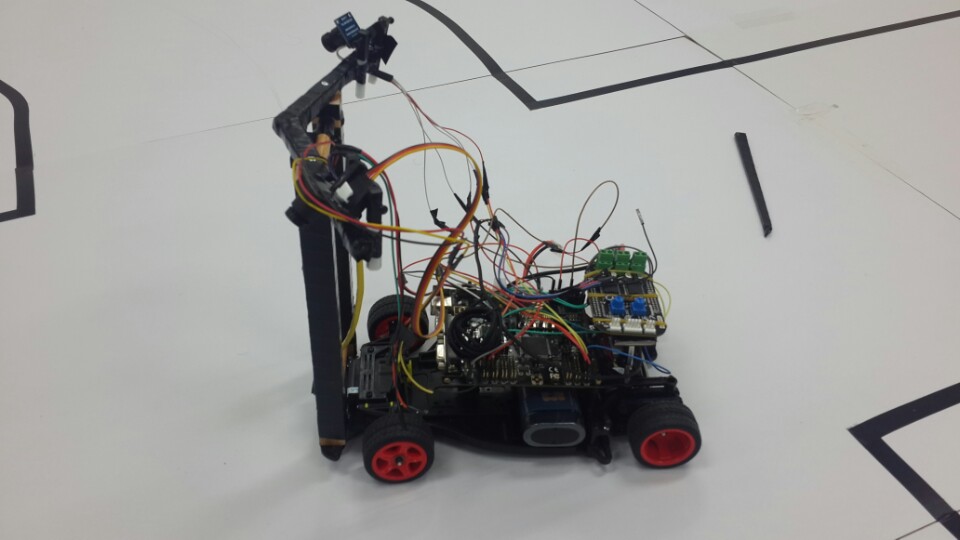


Figure 9. 전체외관 좌측

2.2 소프트웨어 구성

2.2.1 Camera Signal Processing

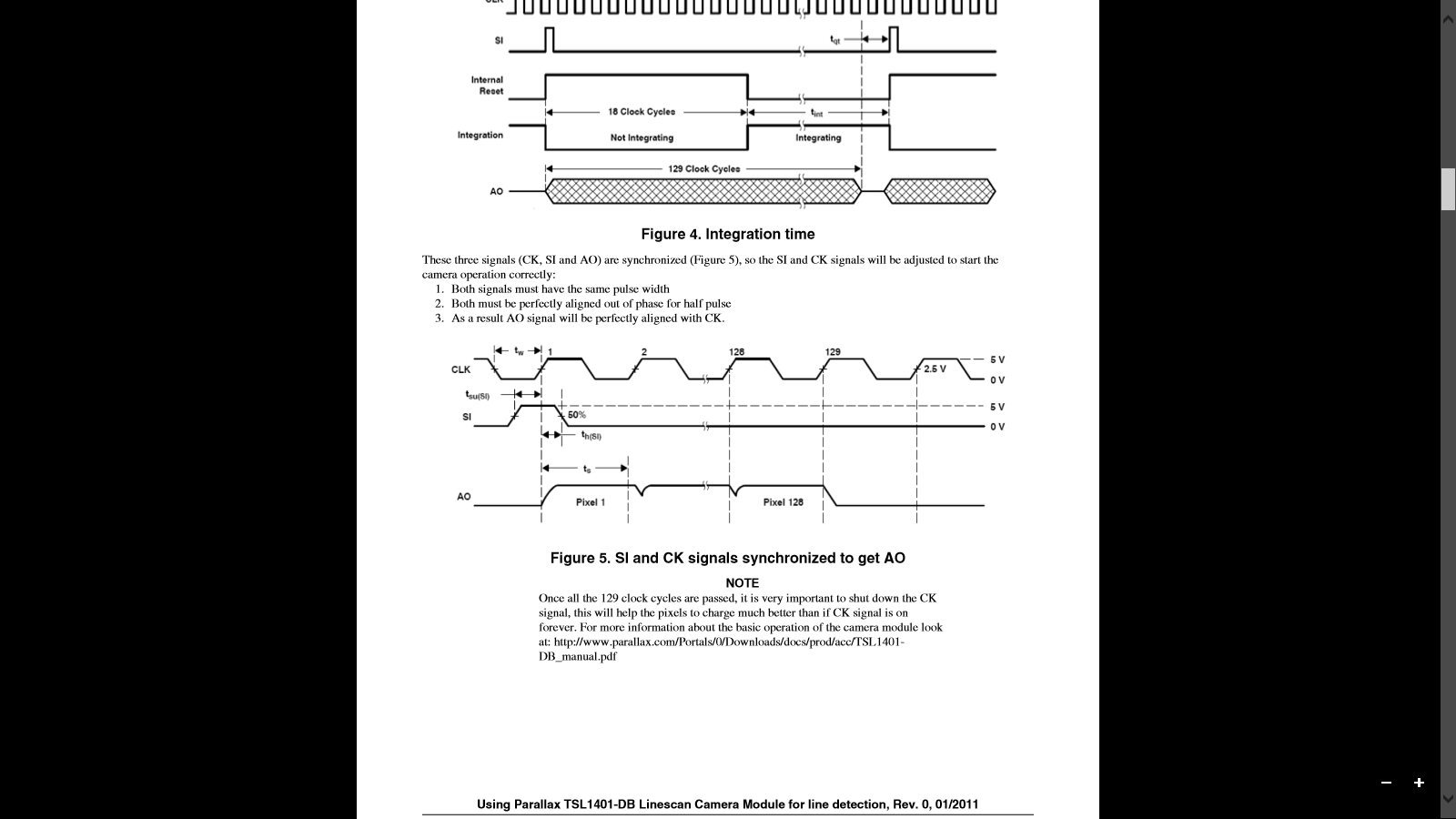
2.2.1.1 이미지 검출

Figure 10. Line Scan Camera Datasheet

Line Scan Camera는 Figure 10. 을 참고하면 SI신호로 한번의 pulse를 발생시키면 128클락이 진행되는 동안 순차적으로 아날로그 값을 출력한다. 이를 사용하기 위하여 MPC-5606B보드에서 GPIO핀을 이용하여 카메라에 SI신호와 클락 신호를 주어 발생되는 아날로그 값을 ADC핀을 통하여 디지털 값으로 변환하여 1픽셀씩 순차적으로 입력 받아온다.

2.2.1.2 ROI(Region Of Interest)처리

Edge검출을 위하여 사용하는 값은 128pixel이 모두 필요하지 않다. 그리고 line scan camera를 이용하여 값을 받아오는 경우에 양 끝 지점의 값을 살펴보면 노이즈가 많고 edge검출을 위해서 필요한 부분이 아니므로 계산에서 제외한다. 이 처리를 하게 되면 계산 횟수와 노이즈를 줄일 수 있다.

2.2.1.3 Mean Filter

Line scan camera를 통하여 받아온 값에 노이즈가 섞여 있는 경우에 오작동할 가능성이 커지게 된다. 따라서 이를 방지하기 위해 사용한 필터링 알고리즘이 mean filter이다. 자동차대회에서의 배경부분은 값의 변화량이 크지 않다. 따라서 잡음의 형태가 임펄스 잡음 형태로 발견된다. 이를 처리하기 위하여 mean filter를 사용한다. 먼저 5개 픽셀 크기의 마스크를 생성한다. 그리고 각 픽셀들에 마스크를 씌워서 해당 마스크에 있는 값의 평균을 구한다. 그리고 그 구한 평균값으로 해당 마스크를 채운다. 이렇게 하면 픽셀에 생긴 노이즈를 주변 값을 이용하여 부드럽게 변화시킬 수 있다.

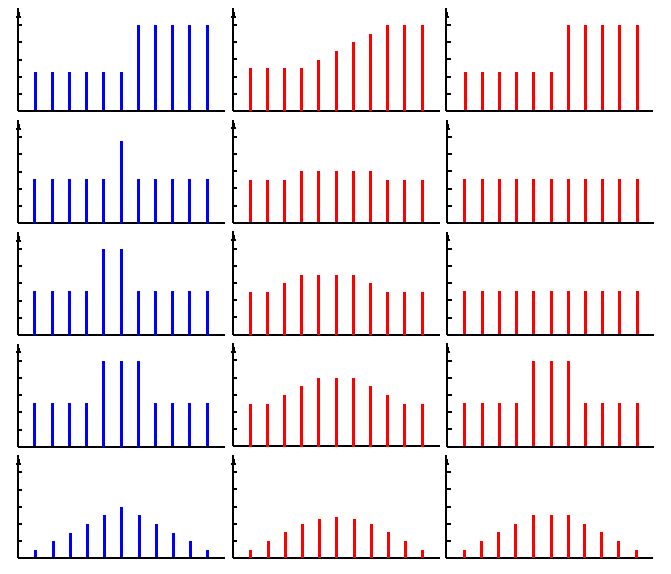


Figure 11. mean filter

2.2.2 Lane Detecting

카메라 값들이 들어와서 한번 평균값 필터를 거쳐서 노이즈가 제거된 이후에 왼쪽 카메라에서 받아온 값에서는 차체의 중앙에 가까운 127번째 index부터 0번째 index방향으로 라인을 탐색하여 값들을 미분하여 기울기를 비교하면서 기울기가 감소한 뒤에 증가하는 부분을 찾는다. 그래서 이 가운데 지점을 라인의 중심이라고 판단하고 이 값을 기준점으로 삼는다. 그리고 오른쪽 카메라에서 받아온 값에서는 차체의 중앙이 0번째 index가 되므로 0번째 index에서 127번째 index방향으로 값을 탐색하여 왼쪽 카메라와 똑같이 미분하여 기울기를 비교하여 기울기가 감소한 뒤에 증가하는 부분을 찾는다. 그렇게 해서 기준점을 잡고 해당 부분을 명확히 하기 위하여 라인이라고 인식한 부분을 1로 바꾸고 나머지 유효한 카메라의 값들을 150으로 변환한다.

이러한 방식으로 받아온 카메라 값들을 처리하여 라인이라고 인식하고 이에 맞추어 모형차가 작동하도록 설계하였다.

2.2.3 Drive Algorithm

차량은 main.c속에서 while(1)내의 함수들로 실행된다. 이 while(1)의 1사이클 마다 savedServoAngle, savedMotorSpeed변수로 이전 상태를 계속 저장해 나가며 이후 차량의 행동에 영향을 준다. 1사이클의 행동의 주역 함수를 순서대로 분석해보면, 사이클이 시작되면 GPIO\_SetState(16,1)로 뒷바퀴 모터를 키고, getCamera()함수로 좌/우 카메라의 line정보를 가져온다. 그리고 CameraArrRight[128] 와 CameraArrLeft[128]로 라인정보를 저장한다. 회전중일 때 코스를 벗어날 위험이 있기 때문에 이 경우에는 좌/우측 카메라의 정보를 받아들이는 우선순위가 회전 방향에 따라 다르다. 좌회전을 할 때는 우측 카메라의 정보를 먼저 찍어 놓는다.

이후 leftCameraFilter()함수와 rightCameraFilter()함수를 실행해서 받아온 정보에 평균값 필터를 씌워 error값들을 제거한다. 해당 필터함수에서 카메라정보 좌/우측 끝의 판단 불가능한 부분은 0으로 바꿔주고, 나머지 부분만 5 index범위의 평균값을 대입한다.

이후 detectLineFromLeftCam() 함수에서 0~127범위의 좌측 카메라 정보에서 차체의 중앙에 가까운 127index의 정보부터 0index까지 line을 검사해서 반환해준다. 검사내용은 min값과 max값의 index를 찾아내는 것이며, max-50 보다 작고, min보다 작은 값을 min에 해당하는 index를 저장해나가며, 검사도중 min+50 보다 큰 값이 나타나면 min주위가 검은색의 라인임을 알아내어 min index를 반환한다.

이후 위에서 알아낸 min index를 파라미터로 verySimpleFilterL(min\_index)를 실행하여 CameraArrLeft[128]에 line이라 인식한 정보만 1로 바꾸고 나머지 유효정보는 150으로 초기화한다.

이후 우측 카메라를 위해 detectLineFromRightCam() 과 verySimpleFilterR(min\_index)를 실행한다. 단, 좌측 카메라와는 다르게 차체의 중심에 가까운 0index에서 127index로 검사를 진행한다.

이후 changeServoAngle()함수를 실행해 Servo각도와 Motor속도를 바꾼다. Servo각도는 총 9가지의 기본 셋팅으로 나눠놓고 자유 각도조정 구간을 제외하고는 정해진 각도로 주행한다. verySimpleFilterR(min\_index)과 verySimpleFilterL(min\_index)을 통해 알아낸 line index를 이용해 좌/우측 카메라의 line정보를 이용한 36가지 경우의 수에 따른 주행을 한다. 경우의 수에 따른 행동 패턴은 아래의 표이다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 좌 우 | 15<= <35 | | 35<= <49 | | 49<= <79 | | 79<= <94 | | 94<= <114 | | Not Exist | |
| 94<= <114 | **Same** | S2 | **Same** | S2 | **R2** | S2 | **R1** | S1 | **R1** | S1 | **R4** | S2 |
| 79<= <94 | **Same** | S2 | **Same** | S2 | **R2** | M | **R1** | F1 | **R1** | M | **R3** | S2 |
| 49<= <79 | **L2** | S2 | **L2** | M | **M** | F2 | **R1** | F2 | **R1** | F1 | **R2** | M |
| 35<= <49 | **L1** | S1 | **L1** | F1 | **L1** | F2 | **M** | F2 | **M** | F2 | **R1** | M |
| 15<= <35 | **L1** | S1 | **L1** | M | **L1** | F1 | **M** | F2 | **M** | F2 | **R1** | F2 |
| Not Exist | **L4** | S2 | **L3** | S2 | **L2** | M | **L1** | F1 | **L1** | F2 | **CASE** | |

Table 1. 행동 Table

이 표에서 L1, L2~~~ R1, R2는 회전각도 왼쪽/오른쪽을 뜻하고, S1, S2, F1, F2는 속도가 Slow, Fast를 의미하고 M은 각도가 중앙, 속도가 중간을 의미한다. CASE는 안 보이는 상황에 대한 다양한 예외행동을 한다.

기본적으로 이 Table을 바탕으로 행동을 하되, 회전도중 코너와 너무 붙어서 카메라가 어떤 것도 찾아내지 못한 경우, 코너를 빠져나갈 위기에 Table과 다른 행동 CASE를 한다. 코너를 도는데 코너가 안 보이게 되는 상황은 이전 Servo각도가 L2~L4 , R2~R4의 회전각이며, 카메라의 index들이 없는 상황이다. 따라서 이때 회전을 더 강하게 시키고 속도도 줄여간다. 또 다른 CASE로 교차로에 진입했을 때는 이전 Servo각도가 L1~M~R1의 회전각인 상태에서 카메라 index들이 전부 없을 때이다. 이 때는 CASE조건에서 Servo: M, Speed: F2로 주행한다. 이 외에 index들이 없을 경우에는 이전Servo, 이전Speed로 주행한다.

마지막으로 changeServoAngle()함수 속에서 차에 Speed를 입력하기 위해 BalanceMotor(Servo\_Value, Motor\_Speed)가 실행된다. 이 함수는 입력한 Servo값과 위의 표와 특정 CASE에 따라 주어진 Speed값에 따라 뒷바퀴의 회전 속도를 차에 입력한다. 예를들어 급격한 커브에는 바깥쪽 바퀴의 속도는 파라미터로 넘어온 Speed를 넣어주고, 안쪽 바퀴는 각도에 따른 일정 값을 변화한 Speed를 넣어준다. 이렇게 사이클 별로 동작하면서 자동차가 동작하게 된다.

2.2.4 AEB(Autonomous Emergency Breaking)

2개의 거리감지 센서를 사용하여 해당 알고리즘을 구현하였다. 2개의 거리감지센서를 높이를 다르게 하여 차에 부착한 뒤 두 개의 센서의 값을 받아온다. 센서의 값이 없는 경우에는 정상적으로 주행한다. 센서가 물체를 인식하게 되는 경우에는 이에 따른 플래그를 설정하여 앞에 물체를 두 센서가 다 감지한 경우에는 차량이 주행을 멈추게 된다. 높이를 다르게 한 이유는 아래에 적외선 센서만 물체를 감지한 경우에는 이것이 언덕임을 알 수 있도록 하였고, 정확도 또한 두 개의 적외선 센서일 때 상승하기 때문에 두 개의 센서를 이용하여 장애물을 판단하도록 하였다.

2.2.5 School Zone Detect Algorithm

카메라 값을 받아온 배열을 확인하여 배열을 처음부터 끝까지 검색하여 어느 한쪽이라도 기준 값을 찾아낸 값이 일정 수를 넘어가는 순간 스쿨존을 판단한다. 스쿨존을 판단한 이후부터는 엔코더에서 값을 받아서 모형차와 스쿨존 사이의 거리만큼 이동 후 속도를 줄이기 시작한다. 속도를 줄인 상태로 이동한 뒤, 스쿨존이 끝나는 선을 다시 만나면 엔코더 값을 받아 자동차와 스쿨존 사이의 거리만큼 이동한 후 속도를 원래대로 복구한다.

3. 주요 장치 이론 및 적용 방법

3.1 주요 알고리즘 이론

3.1.1 Action Table

카메라로 받아온 값이 필터를 거치고 나면 라인을 찾아서 값을 가지고 있게 된다. 그리고 Servo의 현재 상태를 저장하여 라인 값과 비교하여 바퀴의 최종적인 방향을 정하게 된다. 이를 테이블로 만들어 프로그램에 입력한 뒤 이에 따라서 올바른 Servo의 방향을 제어할 수 있도록 하였다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 좌 우 | 15<= <35 | | 35<= <49 | | 49<= <79 | | 79<= <94 | | 94<= <114 | | Not Exist | |
| 94<= <114 | **Same** | S2 | **Same** | S2 | **R2** | S2 | **R1** | S1 | **R1** | S1 | **R4** | S2 |
| 79<= <94 | **Same** | S2 | **Same** | S2 | **R2** | M | **R1** | F1 | **R1** | M | **R3** | S2 |
| 49<= <79 | **L2** | S2 | **L2** | M | **M** | F2 | **R1** | F2 | **R1** | F1 | **R2** | M |
| 35<= <49 | **L1** | S1 | **L1** | F1 | **L1** | F2 | **M** | F2 | **M** | F2 | **R1** | M |
| 15<= <35 | **L1** | S1 | **L1** | M | **L1** | F1 | **M** | F2 | **M** | F2 | **R1** | F2 |
| Not Exist | **L4** | S2 | **L3** | S2 | **L2** | M | **L1** | F1 | **L1** | F2 | **CASE** | |

Table 1. 행동 Table

4. 결론 및 토의

4.1 결론

4.1.1 Freescale 보드들의 사용

MPC5606B보드와 FRDM-TFC보드를 사용하여 각 센서들을 제어하였다. 이를 제어하는 데에는 MPC5606B보드에서 제공되는 VDD와 GND, GPIO, Emios, ADC 등이 이용되었다. VDD와 GND를 이용하여 각 센서들에 전원을 공급하였고, GPIO를 이용하여 각 센서들의 값을 받아올 수 있었으며, Emios를 이용하여 Servo와 DC Motor에 PWM을 입력할 수 있었으며 ADC를 이용하여 line scan camera와 적외선 센서의 값을 받아와서 처리할 수 있었다. 그래서 모든 알고리즘을 만들 수 있는 기초적인 데이터를 받아오는데 많은 역할을 하였다. 그리고 FRDM-TFC보드는 쉽게 DC Motor를 제어할 수 있도록 하는 역할을 하였다. 전원을 공급하고 이를 DC Motor에 쉽게 공급할 수 있었으며, PWM을 받아와서 DC Motor의 속도를 쉽게 제어할 수 있도록 하였다.

4.1.2 Line Scam Camera 사용

Line Scan Camera는 맨 처음 사용하기 전에는 한번에 한 라인단위로 값을 받아올 것 같았는데 Datasheet를 보니 예상과는 다르게 동작하여 처음에 약간 헤맸었는데 Datasheet를 확인하고 Clock과 SI Pulse에 따라서 동작함을 알고 이에 따른 값을 GPIO를 이용하여 MPC5606B 보드를 통해 카메라에 공급하여 값을 받아올 수 있었다. 그래서 한 cycle의 clock pulse마다 한 픽셀씩 값을 받아와서 프로그램에서 배열에 저장한 뒤에 라인을 파악하고 자동차의 주된 알고리즘을 작성하였다.

4.1.3 Encoder 사용

Encoder는 Autonics사의 E20S2-200-3-N-5-R이라는 모델을 사용하였다. 이 Encoder는 5V의 전원을 연결하면 각각 A, B, Z상을 통해서 값을 검출하여 사용할 수 있었다. Encoder를 사용함으로써 바퀴의 회전 수를 구할 수 있었고 구한 값을 바탕으로 속력을 구할 수 있었다. 구한 속력을 이용하여 커브나 직선구간에서 각각 현재 속력을 구할 수 있었다.

4.1.4 Servo 사용

자동차에 사용된 Servo는 MPC5606B 보드를 통해서 바퀴방향 정면으로부터 PWM값을 주어 좌우방향으로 조절할 수 있도록 하였고, 이는 PWM Duty cycle을 조절함으로써 제어할 수 있었다. 조절방법은 카메라를 이용하여 라인을 받아오고, 받아온 데이터에 따라서 각 구간마다 방향을 제어할 수 있도록 구성하였다.

4.1.5 거리감지센서 사용

거리감지센서는 언덕과 장애물 등을 판단하기 위하여 설치하였다. 거리감지센서의 값은 메인보드에서 제공하는 ADC를 이용하여 값을 변환하여 사용하였다. 들어온 값은 거리가 가까워질수록 커졌고, 이를 이용하여 앞에 어떠한 물체가 있는지를 알 수 있었다. 그리고 거리감지센서를 2개 장착하여 언덕도 확인하고, 장애물 판단에 대한 정확성 또한 향상시켰다.

4.1.6 DC Motor 사용

DC Motor의 경우 MPC5606B보드에서 PWM을 주고 FRDM-TFC보드에서 전원을 주어서 사용하였는데, PWM Duty Cycle이 짧아질수록 모터의 속도가 줄어들었으며 이를 이용하여 각 바퀴의 속도를 다르게 하여 커브를 돌 때 원활하게 돌 수 있도록 하였다.

4.2 배운점 및 토의

보드들을 사용하면서 처음에는 익숙하지 않은 보드들을 사용하였기 때문에 실수들이 많이 발생하였다. 핀을 설정하는 것부터 프로그램을 작성하고 그 프로그램을 보드에 입력하여 실제로 자동차를 동작시키는데까지 많은 과정들이 있었고 어려움들도 있었다. 하지만 보드의 Datasheet를 확인해가면서 각 핀의 역할을 설정하고 연결하여 동작했을 때는 뿌듯함을 느낄 수 있었다. 그리고 간단한 보드들을 이용하여 여러가지 기능들을 구현할 수 있음도 신기하였고, 보드에 대한 거부감을 줄일 수 있었던 기회가 된 것 같다.

보드 이외에도 여러가지 센서들을 다루면서 센서에 대한 Datasheet를 찾아보고 그에 맞게 설정을 하고 사용을 함으로써 센서들의 활용법을 익힐 수 있게 되었고 새로운 장치의 사용에 대한 두려움도 없앨 수 있는 기회였다. 그리고 각 상황에 맞는 센서를 직접 찾아가면서 그에 맞는 장치를 사용할 수 있게 되어서 앞으로도 이와 비슷한 상황이 있으면 해당 상황에 맞는 센서들을 찾아서 사용할 수 있는 판단력을 기를 수 있었다.

마지막으로 학교에서 이론적으로만 배운 알고리즘(영상처리 등)들을 실제로 적용해보면서 알고리즘의 문제점과 그에 대한 보완점등을 파악하고 해당 알고리즘에 대하여 팀원들끼리 소통하면서 팀 프로젝트로써 협력하는 방법에 대하여 직접적으로 경험할 수 있는 기회가 된 것 같다.

Appendix – 부품 목록

|  |  |  |  |
| --- | --- | --- | --- |
| **제조사** | **부품명** | **수량** | **사용목적** |
| Autonics | E20S2-200-3-N-5-R | 1 | Encoder |
| ACE battery | 7.2V 3000mA | 1 | Battery |
| SHARP | 2YOA21F43 | 1 | Distance Measuring Sensor |
| SHARP | 0A41SKF32 | 1 | Distance Measuring Sensor |
| Freescale | FRDM-TFC | 1 | MCU |
| Freescale | TRK-MPC5606B | 1 | Motor Board |
| Freescale | AN4244 | 1 | Line Scan Camera |
| AMS-TAOS | TSL-1401 | 1 | Line Scan Camera |
| Standard Motor | RN260-CN-18130 | 2 | DC Motor |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |