초음파 센서를 이용한 자율주행 프로젝트

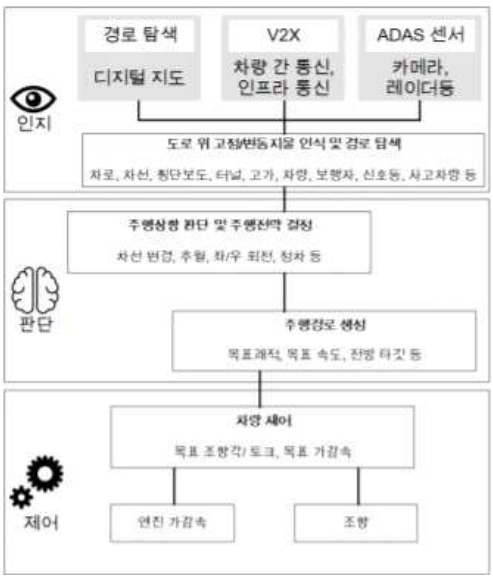
8조
17100179 이진우
17100215 한성민
17100222 홍태주

INDEX

1. Introduction

(1) Topic Selection

21세기에 들어와서 과학은 급속도로 발전하였고, 이러한 과학의 발전은 기술뿐만 아니라 우리의 삶에 밀접하고 큰 영향을 끼치며 삶의 질을 높여주고 있다. 그 중 특히 전 세계 4차 산업 혁명 열풍이 뜨겁다. 모빌리티 혁명은 우리의 삶을 크게 바꿔놓을 것이라고 기대하는데, 그 중 자율주행은 모빌리티 혁명에 핵심적인 주제이다. 테슬라와 구글 웨이모는 자율주행 자동차의 대표적인 주자들로 많은 국가, 기업들을 자율주행 시장으로 뛰어들게 하고 있다. 우리 8조는 이번 프로젝트를 통해 자율주행에 대해서 어떻게 알고리즘이 구성되어 있는지 알고, 전체적으로 실행되는 과정을 알아보고자 해당주제를 선정하였다.



[그림1] 자율주행 자동차 알고리즘

(2) Analysis

자율주행차란 운전자가 직접 조작하지 않아도 자동차가 주행 환경을 인식해 위험을 판단하고 주행 경로를 계획해 스스로 운전하는 자동차이다. 이를 위해 GPS, 카메라, 레이더 센서, 감시 시스템, 중앙제어 장치, 액츄에이터 등으로 구성되어있다.

자율주행이 이루어지는 단계는 크게 3가지 단계로 나뉜다.
1. 첫 번째는 인지단계로 인지 단계에서는 카메라, 센서, GPS 등을 이용해 교통상황이나 운행 환경 등 주변 환경을 파악하는 단계이다.

2. 두 번째는 판단 단계로 인지단계에서 얻은 정보를 바탕으로 주행계획을 세운다. 인지단계에서 얻은 정보가 정확하고 세부적일수록 정교한 주행계획을 세울 가능성이 높아진다.

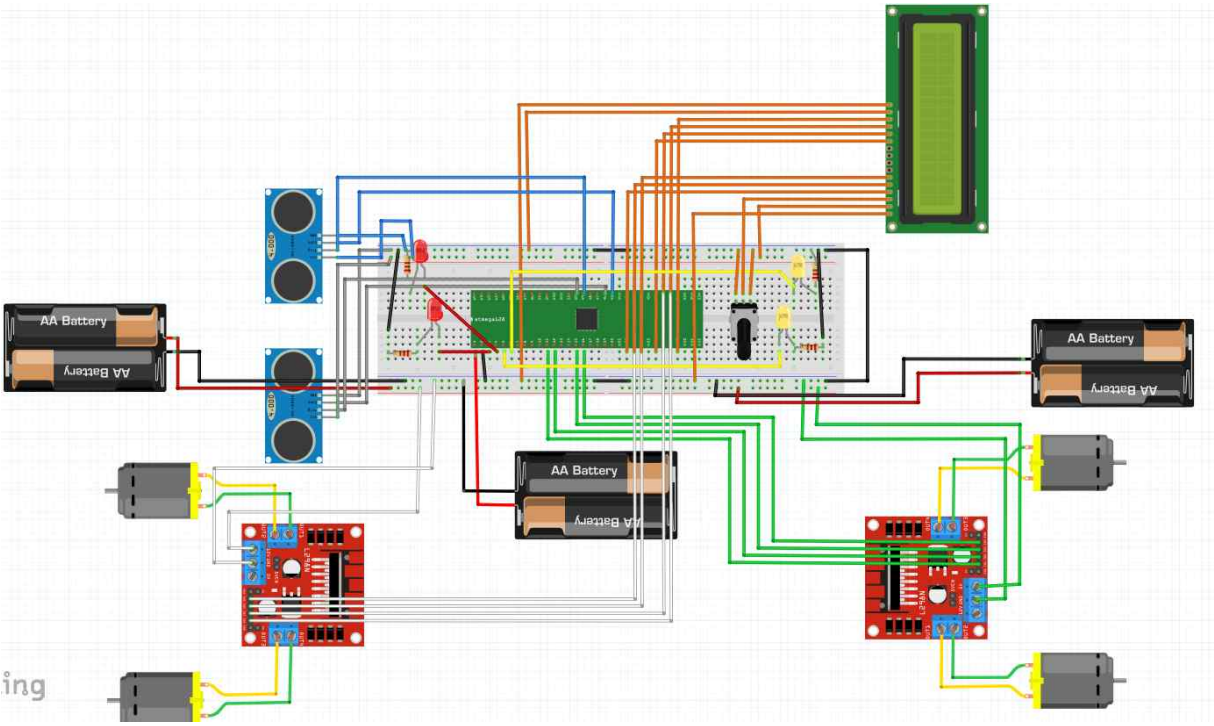
3. 마지막 제어단계에서는 앞선 인지, 판단 단계에서 얻은 정보를 종합하여 자동차 스스로가 엔진 구동이나 주행 방향, 속도 등을 결정하여 수행한다.

이 단계를 우리 프로젝트에 적용해보자면
1단계 이번 프로젝트에서 우리는 초음파센서를 사용하여 주변 환경과 장애물들을 인식
2단계 차량과 장애물간의 사이의 거리를 측정하면서 주행하는 알고리즘을 구상
3단계 그에 따른 모터제어를 통해 제어단계를 구상

으로 구분할 수 있을 것이다.
더 나아가 우리8조는 3단계에서 적절한 상황에 따른 모터제어에 있어 현재 위치를 아는 것이 매우 중요함을 알았다. 따라서 처음 현재 위치를 계산하여 차량의 실시간 위치를 LCD를 통해 보여주는 방향을 구상하였다.

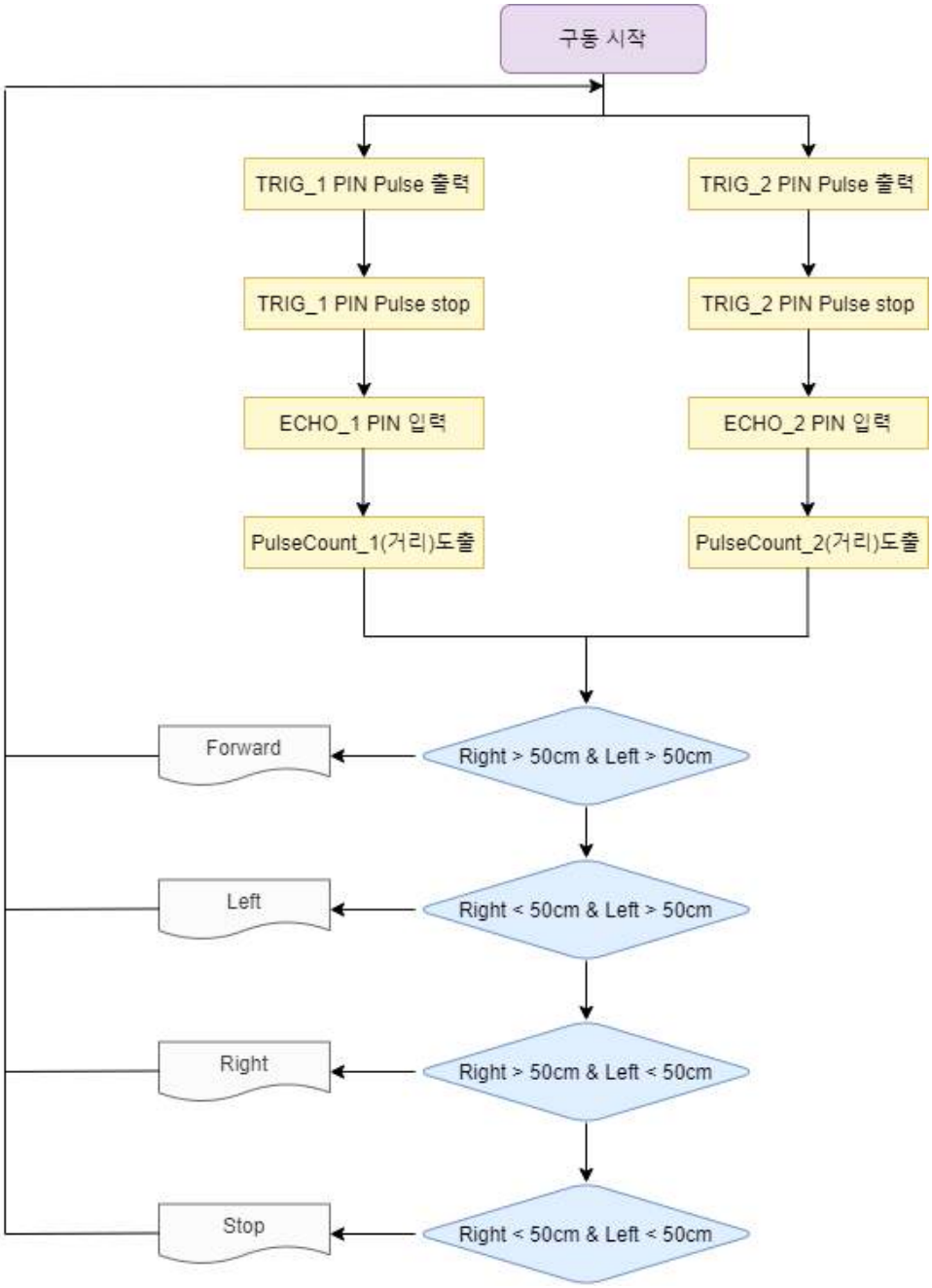
2. Algorithm

(1) Circuit diagram



[그림2] 회로도

(2) Block Diagram



[그림3] Block Diagram

(3) Source Code

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <math.h>

#include "LCD.h"

#define USControl_PORT PORTE          // 초음파센서 제어
#define USControl_DDR DDRE

#define DELAY_TIME 50 // 딜레이 타임
#define LENGTH 50              // 기준 거리 (cm)

volatile unsigned int pulse_count, pulse_count2 = 0;              // 58us 마다 증가되는
카운터변수
volatile unsigned char togle, togle2 = 0;


#define T 0.7 // 한바퀴 도는데 걸리는 시간
#define v 300 // 자동차 속도
#define pi 3.141592

volatile float L=0; // index L 초기화
volatile float R=0; // index R 초기화


//___이동 함수___
void Motor_enable(void);
void forward(void);
void backward(void);
void handlingRight(void);
void handlingLeft(void);
void stop(void);

void US1_pulse(void)              // 초음파 센서1 Trig핀에 10us input trigger pulse 출력
{
        USControl_PORT = 0x01;
        _delay_us(10);
        USControl_PORT = 0x00;
```

```
}

void US2_pulse(void)              // 초음파 센서2 Trig핀에 10us input trigger pulse 출력
{
        USControl_PORT = 0x02;
        _delay_us(10);
        USControl_PORT = 0x00;
}

//___타이머___
ISR(TIMER0_OVF_vect)              // Timer 0 Overflow Interrupt
{
        pulse_count++;            // 58us 마다 증가
        TCNT0 = 140;
}

ISR(TIMER2_OVF_vect)              // Timer 2 Overflow Interrupt
{
        pulse_count2++;           // 58us 마다 증가
        TCNT2 = 140;
}

//___에코핀 인터럽트___
ISR(INT4_vect)            // INT4핀에 Rising edge or Falling edge가 감지되면 인터럽트 호출
{
        if(togle == 0)            // Rising edge일때
        {
                pulse_count = 0;          // pulse_count 초기화

                TCCR0 = 0x02;             // Normal mode, Clock 8ck
                TCNT0 = 140;

                EICRB |= (1<<ISC41);              // Falling edge 감지
                EICRB &= 0xFE;

                TIMSK |= (1<<TOIE0);
// Timer/Counter 0 Overflow Interrupt Enable

                togle = 1;
        }

        else              // Falling edge일때
        {
```

```
                EICRB |= (1<< ISC41)|(1<<ISC40);          // Rising edge 일 때
INT4_vect 인터럽트 호출

                TIMSK &= (1<< TOIE0);          // Timer/Counter 0 Overflow
Interrupt Disable
                TCCR0 = 0x00;          // Timer/Counter 0 정지

                togle = 0;
        }
}

ISR(INT5_vect)          // INT5핀에 Rising edge or Falling edge가 감지되면 인터럽트 호출
{
        if(togle2 == 0)          // Rising edge일때
        {
                pulse_count2 = 0;                // pulse_count 초기화

                TCCR2 = 0x02;          // Normal mode, Clock 8ck
                TCNT2 = 140;

                EICRB |= (1<<ISC51);                // Falling edge 감지
                EICRB &= 0xFB;

                TIMSK |= (1<<TOIE2);                // Timer/Counter 2 이용

                togle2 = 1;
        }

        else            // Falling edge일때
        {
                EICRB |= (1<< ISC51)|(1<<ISC50);          // Rising edge 감지

                TIMSK &=0xBF;          // Timer/Counter 2 Overflow Interrupt Disable
                TCCR2 = 0x00;      // Timer/Counter 2 정지

                togle2 = 0;
        }
}


//___main___
int main(void)
{
```

```
//////속도 계산 인덱스들
float theta = 0;
float distanceX = 0; // x축으로 간 거리
float distanceY = 0; // y축으로 간 거리

char firstLine[] = "000000" ;
char secondline[] = "000000";
/////LCD끝////////
LCDOUT();
USControl_DDR = 0x0F;            // 초음파 센서 설정
Motor_enable();            // 모터 레지스터 활성화
DDRA = 0xFF;               // led 활성화
//___Rising edge 일 때 INT4_vect, INT5_vect 인터럽트 호출___
EIMSK |= (1<<INT4)|(1<<INT5);    // INT4,5 활성화
EICRB |= (1<<ISC40)|(1<<ISC41)|(1<<ISC50)|(1<<ISC51);// INT4,5 Rising edge 활성화

sei();               // 전체 인터럽트 Enable

while(1)  //반복문 시작
{
        US1_pulse();               // 초음파센서 1 동작
        _delay_ms(DELAY_TIME);

        US2_pulse();               // 초음파센서 2 동작
        _delay_ms(DELAY_TIME);


        Command(CLEAR);

        sprintf(firstLine, "x : %4.2f", distanceX);
        LCD_String(firstLine);
        sprintf(secondline, "y : %4.2f", distanceY);
        Command(LINE2);
        LCD_String(secondline);
        _delay_ms(100);


        if((pulse_count>LENGTH)&&(pulse_count2>LENGTH))
// 초음파 두개 모두 LENGTH 이상
        {
                theta = 90. + ((2.*180.) / T *0.05* (L+R));
//T는 차체가 한바퀴 도는데 걸리는 시간
                PORTA = 0x03;
```

```
                        forward();
                        _delay_ms(50);

                        distanceX += 0.05 * v * cos(pi/180*theta);
                        distanceY += 0.05 * v * sin(pi/180*theta);
                }
                else if((pulse_count>LENGTH)&&(pulse_count2<LENGTH))
// 오른쪽 감지 좌회전
                {
                        PORTA = 0x05;
                        handlingLeft();
                }
                else if((pulse_count<LENGTH)&&(pulse_count2>LENGTH))
// 왼쪽 감지 우회전
                {
                        PORTA = 0x0A;
                        handlingRight();
                }

                else                // 양쪽 감지 멈춤
                {
                        PORTA = 0x0F;
                        stop();
                }
        }
}

void Motor_enable(void) // 모터 레지스터 설정
{
        DDRF = 0x33;
        DDRB = 0x33;
}

void forward(void)// 전진
{
        PORTF=0x22;
        PORTB=0x21;
}

void backward(void)// 후진
{
        PORTF=0x21;
        PORTB=0x22;
```

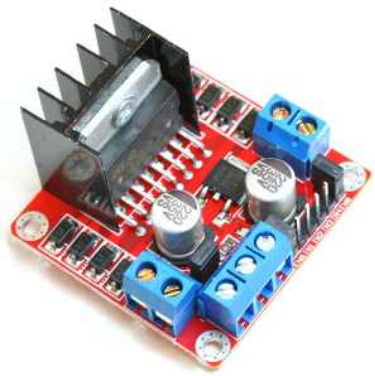
```
}

void handlingRight(void)// 우회전
{
        PORTF=0x20;
        PORTB=0x01;
        _delay_ms(50);
        R--; //우회전 시 index R 감소
}

void handlingLeft(void)// 좌회전
{
        PORTF=0x02;
        PORTB=0x20;
        _delay_ms(50);
        L++; //좌회전 시 index L 증가
}

void stop(void)// 정지
{
        PORTF=0x00;// 신호선 2개 비트 같으면 0x00 또는 0x11
        PORTB=0x00;
}
```

3. HardWare

(1) Used Materials

부품명	수량	비고
Atmega128	1EA	SoftWare
HC-SR04	2EA	초음파 센서
L298N	2EA	Motor Driver
LED	4EA	지시등
LCD	1EA	위치출력
9V Battery	2EA	Motor 외부전원
5V Battery	1EA	Atmega128 외부전원
DC MOTOR	4EA	바퀴 구동

(2) L298N 모터 드라이브



[그림4] L298N

1. 사양
1)모터 전압 : DC 5~35V
2)허용 전류 : 모터당 2A

2. 사용 설명
모터를 조절할 수 있는 2개의 채널을 가진 모터 드라이버 모듈.
IN1/IN2, IN3/IN4핀에 디지털 신호를 가하여 양쪽 신호가 같으면 정지, 다르면 정회전 또는 역회전을 줄 수 있다. DC모터 4개를 제어하기 위해 2개의 L298N 모터 드라이버를 사용하였으며, 각각의 모터 드라이버는 PORTF(front), PORTB(back)에서 출력으로 앞바퀴 2개, 뒷바퀴 2개를 제어한다.

다음은 각 바퀴의 디지털 신호에 대한 회전 방향을 표시한 것이다.

	정회전	역회전	PORT
FRONT1	0x02	0x01	F
FRONT2	0x10	0x20	F
BACK3	0x01	0x02	B
BACK4	0x10	0x20	B

팀의 목표는 출발점을 기준으로 주행하는 도중 RC카의 위치를 x,y축 2차원 평면좌표를 구하는 것이었으며, 이를 위해서는 차량의 속도가 필요했다. 교내 바닥에서 직접 측정한 차량의 속도는 인가전압 9v와 대조해봤을 때 상대적으로 작은 값이었다. 예상되는 가장 큰 원인으로는 바퀴 4개를 굴림과 동시에 노면에서 일어나는 마찰에 의한 감속과, 실제로 9v 건전지의 전압을 측정했을 때 약 7.8 v가 측정되어 속력이 작게 나옴을 예상할 수 있었다.

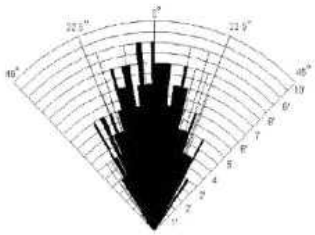
(3) HC-SR04 초음파 센서 모듈

우리 8조는 자율주행 자동차 동작에서 거리를 측정하는 센서로 초음파를 선택하였다. 초음파는 센서는 비교적 가격이 싸고 이용하기 간단하다. 그 중 HC-SR04를 사용했는데, 이 센서는 약 4m까지의 거리를 측정할 수 있고, 3mm의 측정해상도, 15도의 측정각을 가지고 있다. 아래는 HC-SR04의 상세정보이다.

[그림5] HC-SR04

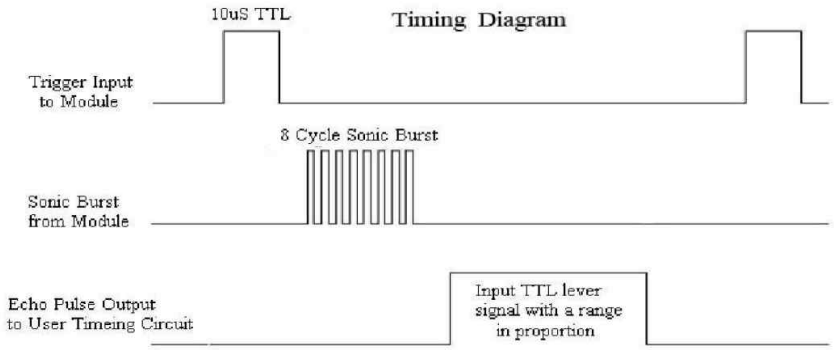
Features:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm



Practical test of performance,
Best in 30 degree angle

초음파 센서의 동작 원리는 다음과 같다. 먼저 Atmega128에서 High를 HC-SR04의 Trig핀으로 10us동안 보내둔다. 신호를 받은 HC-SR04은 40kHz의 초음파를 8사이클 발사하고 Echo핀에 High 신호를 출력한다. 초음파는 물체에 반사되어 다시 HC-SR04로 돌아오고, Echo핀의 출력을 Low로 바꾼다. 여기서 Echo핀의 High였던 길이는 발사된 초음파가 장애물에 반사되어 다시 수신될 때까지의 왕복시간을 나타낸다. 초음파가 출발하고 되돌아온 시간에 음속을 곱해주면 초음파가 측정한 거리값을 알 수 있다.



[그림6] 초음파 모듈 작동원리

코드를 살펴보면 TIMER0를 호출하는 주기를 58us로 놓았다. 즉, Echo핀이 High일 때 58us 마다 한번씩 pulse_count1값이 증가시킨다. 이와 같이 설정한 이유는 다음과 같다.

음속은 340m/s = 34000cm/s이고, 1cm 나아가는데 약 0.29us가 걸린다.

$$1/34000 = 0.000029 \fallingdotseq 29us$$

예를 들어 3cm를 측정하는데 음파는 0.29x3 = 0.87us가 걸렸을 것이다. 음파가 왕복하는 시간은 2배가 된다. 우리는 음파가 1cm 왕복했을 때의 시간을 고려하기 때문에 주기를 58us로 두었다.

$$29us \times 2 = 58us$$

58us마다 Timer/Counter의 Overflow Interrupt를 호출하기 위해서는 TCNT값 설정해야 한다. 8비트 Timer/Counter0를 사용하였다. 분주비 N = 8으로 두었을 때 출력 주파수가 16MHz인 Atmega128에서의 클럭 주기는 0.5us, ISR 실행주기는 128us가 된다.

$$(8/16MHz) \times 256 = 128us$$

ISR 실행주기가 58us가 되려면

$$(8/16MHz) \times k = 128us$$
$$k = 116$$

Timer/Counter 0을 116단계로 카운팅 해야 한다. 8bit Counter(0 ~ 255, 총256단계)이므로 TCNT0 = 140 이어야 한다.

$$256 - 116 = 140$$

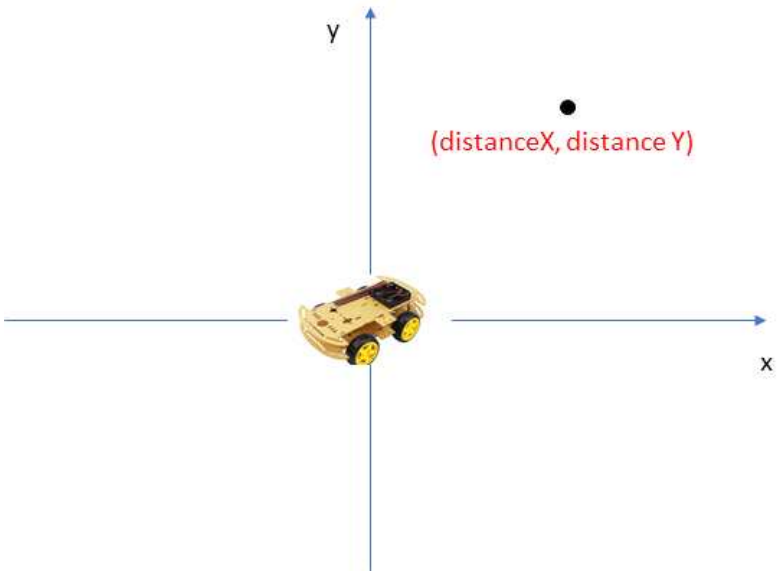
이러한 과정을 거쳐 HC-SR04가 음파를 출력하고, pulse_count1에서 거리 값을 cm단위로 계산한다. 우리는 2개의 초음파를 사용할 것이기 때문에 두 번째 초음파는 timer2를 이용하였고 그 값은 pulse_count2에 저장된다.

4. Software

(1) Location Aware

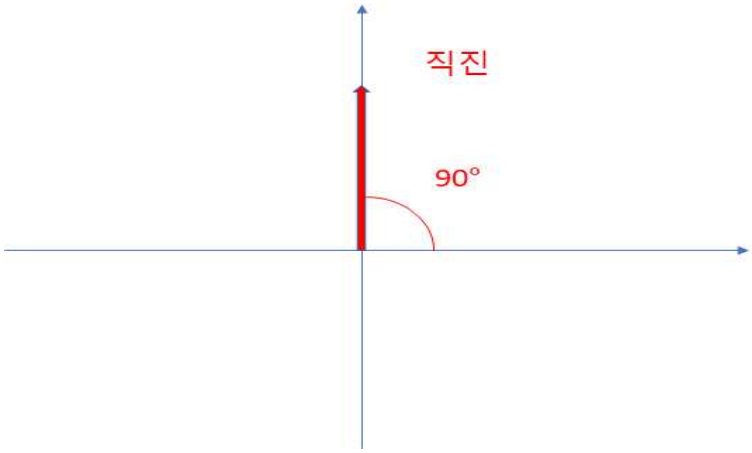
우리 조는 자율 주행 자동차의 핵심 요소 중 하나가 현재 위치에 대한 정보라고 생각하였다. 다만 IMU 또는 GPS를 사용하지 않고도 위치정보를 얻어낼 수 있는 방법에 대해 고려했으며, 자동차의 역학적인 성질을 이용하여 현 위치 정보를 얻어내는 알고리즘을 구상하였다.
원리는 다음과 같다.

먼저, 자동차가 주행하고 있는 노면의 높낮이는 고려하지 않고 모든 노면은 2차원 평면이라고 가정하였다. 이때, 임의의 위치에 대한 자동차의 좌표 값은 그림과 같이 (distanceX, distanceY)로 표시된다.

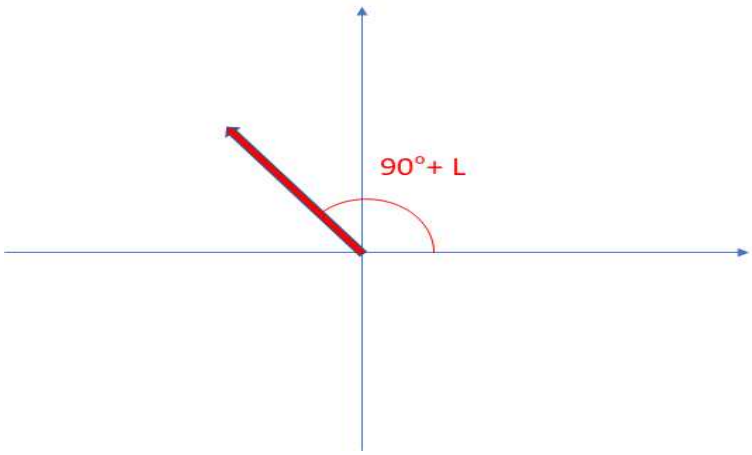


[그림7] 좌표축

자동차의 위치를 알기 위해 고려되어야 할 인자는 자동차의 주행 속도, 좌회전 또는 우회전 시 자동차의 회전 각도가 있다. 자동차의 주행 속도는 DC모터의 제원과 노면의 마찰력, 자동차의 무게 등에 의해서 결정 될 수 있다. 하지만, 자동차가 회전했을 시 얼마나 회전하는지 알기 위해선 회전량을 표현해줄 새로운 인자가 필요했고, 우리는 좌회전 시 증가하는 index 'L', 우회전 시 감소하는 index 'R'을 새로 선언하였다.



새로운 index를 적용하기 위해선 자동차의 주행 각도에 대한 설명이 필요하다. 2차원 평면상에서 표현되는 자동차의 위치는 그 주행 각도에 따라서 x, y축으로 증가 또는 감소하는 비율이 다르다. 이때 초기에 직진하는 자동차의 주행 각도를 90°라고 설정한다. 이 기울기에 따라 계속 앞으로만 가게 된다면 자동차의 위치는 (0,y)가 될 것이다.



이제 앞서 나온 index L과 R에 의해서 주행각도는 변하게 된다. 그림을 참고하면 왜 L이 증가하는지 알 수 있다. 좌회전을 하게 되면 주행각도는 반시계방향으로 이동해야 하고, 이는 각도가 증가하는 방향이기 때문에 L은 증가하게 되며, 우회전시 주행각도는 시계방향으로 이동해야 하므로 각도가 줄어들도록 R을 감소하게 설정한 이유가 설명된다.

최종적으로 현재 주행중인 자동차의 주행 각도는 코드 상에서 'theta'로 표시되며,
이때 자동차의 1회전 시 걸린 시간 'T'가 고려되는데, 그 이유는 아래 코드와 함께 설명된다.

```
void handlingRight(void)    // 우회전
{
  PORTF=0x20;
  PORTB=0x01;
  _delay_ms(50);
  R--;
}

void handlingLeft(void)   // 좌회전
{
  PORTF=0x02;
  PORTB=0x20;
  _delay_ms(50);
  L++;
}
```

[그림8] 좌/우회전 코드

다음은 자동차의 좌/우회전 코드이다. 한번 회전 신호가 들어오게 되면 delay인 50ms 만큼 회전을 하도록 설정되어 있고, 50ms에 한번씩 index L또는 R이 증감하게 되어있다. 이때 자동차가 1회전 하는데 걸리는 시간은 T라 했을 때 0,05초(50ms)에 자동차는 1rev*(0.05/T)만큼 회전하게 될 것이다. 따라서 T가 고려된 theta는 다음과 같이 수식으로 정리할 수 있다.

theta = 90.+((2.*180)/T*0.05*(L+R))

따라서 이 theta에 의한 자동차의 미소 이동거리는 x,y 축에 대하여 cos(theta), sin(theta)가 될 것이다.

그렇다면 이제 theta에 대한 위치 좌표 distanceX와 distanceY의 계산을 할 수 있다.

```
if((pulse_count>LENGTH)&&(pulse_count2>LENGTH))   // 초음파 두개 모두 5cm 이상
{
  theta = 90. + ((2.*180.) / T *0.05* (L+R)); //T는 차체가 한바퀴 도는데 걸리는 시간
  PORTA = 0x03;
  forward();
  _delay_ms(50);

  distanceX += 0.05 * v * cos(pi/180*theta);
  distanceY += 0.05 * v * sin(pi/180*theta);
}
```

[그림9] pulse_countn, 즉 초음파 센서에 의해 장애물이 없을 때 직진하는 코드

우리 조가 제작한 자동차의 특성상 좌/우회전시 자동차의 변위는 변하지 않고 제자리에서 회전을 하기에 변위는 전진을 할 때에만 변하게 설정되어 있다. 앞서 설명한 theta가 함수의 delay인 50ms마다 계산되었듯이 전진에 대해서도 같은 delay가 적용되었기 때문에 자동차의 미소 이동거리는 자동차의 속력 v를 고려했을 때

x축 미소 이동거리 : 0.05*v*cos(pi/180*theta) (MCU는 삼각함수 계산시 rad단위 계산)
y축 미소 이동거리 : 0.05*v*sin(pi/180*theta)
가 될 것이고, 계속 전진함에 따라 미소 이동거리가 더해져야 하므로 +=연산자가 사용되었다.

따라서 초기 위치 (0,0)에서 출발한 자율주행 자동차의 현 위치 좌표는 (distanceX, distanceY)가 됨을 알 수 있다.

5. Discussion

(1) Consideration & Improvements

우리는 앞서 말한 인지, 판단, 제어라는 3가지 단계를 초음파를 통해 주변 사물을 인식하는 인지 단계, 주변 사물의 위치를 파악하여 주행의 계획을 세우는 판단 단계, 마지막으로 앞서 인지, 판단에서 얻은 데이터를 바탕으로 자율주행을 구현하였다. 초음파를 사용하여 가까운 거리에 있는 장애물을 좌, 우로 피해 특정 상황에서 좀 더 유연하게 장애물에 충돌하지 않고 주행할 수 있게끔 변수에 다양한 값과 이론적으로 계산한 값을 적절히 넣어 결과 값을 도출할 수 있었다.

동시에 차량의 현재 위치를 센서를 사용하지 않고 Atmega128 내에서 계산하여 출력할 수 있을까에 대해서 고민하고 그 결과 각도의 변수 값 조정과 실제 차량의 속도를 측정하고 많은 실험과 계산, 오차를 통해 위치 값을 계산할 수 있는 함수를 만드는 데 성공하였다.

계산한 값과 실험값을 통해 적절한 변수를 설정하는데 성공했지만 이 과정에서 우리는 몇 가지 문제점을 발견하였다.

외부 전원의 불안정한 공급으로 전력이 점점 떨어져 기대했던 차량의 속도만큼 나오지 않을 뿐 아니라 실험값도 계속 달라져 속도 측정 및 거리 계산에 많은 어려움을 겪었다. 또한 초음파를 2개만 사용하여 장애물이 두 개의 초음파 모듈 사이에 있을 경우 이를 감지하지 못하고 충돌해 버리는 경우도 생겨났다. 이 경우 8bit TIMER 1개당 초음파 센서를 1개씩 사용하여서 이 당시 초음파의 제어 개수를 늘릴 방법을 찾지 못하였다.

전자의 문제들 해결하기 위해선 용량이 큰 외부 전원을 사용하는 방법이 있겠다. 지나시게 전력을 소모하는 부분들을 PID같은 control제어를 통해 최대한 줄여보는 것도 좋은 방법이다. TIMER에 관한 해결 방법으로는 코드로써 해결해야 한다. 초음파 모듈을 순차적으로 타이머를 사용해 하나의 타이머로도 여러 초음파 모듈을 사용하게끔 할 수 있을 것이다. 이때는 여러 초음파 모듈에서 얻은 거리값이 혼동되지 않게 주의해야 한다. 다른 방법으로는 Atmega128

과 초음파 센서의 개수를 늘려 장애물에 대한 사각지대가 없도록 보완하는 방법을 고려할 수 있다.

지금의 위치를 계산하는 방식은 오직 평면에서만 계산이 가능한데 추후 자이로 센서를 사용하여 이를 3차원 좌표공간으로 확장할 수 있다. 레이더와 라이더가 가장 정확한 센서이겠지만 비용과 난이도 문제가 존재한다. 평면상의 x,y 좌표값이 주어진 속도의 변화에 따라 오르막인지나 내리막인지를 감지하여 위치 값을 3차원으로 표현을 할 수 있는 방안을 고려해볼 수 있다. 다만 속도는 마찰에 의해서도 달라질 것이니 주의해야 한다. 더 나아가 원하는 위치를 지정한 후 주행경로상에 있는 장애물을 자율주행을 통해 피하면서 목적지로 주행하는 알고리즘 또한 충분히 구현할 수 있다고 생각한다.

참고자료

<자율 주행 자동차 보안 위협 및 기술 동향>

https://www.dbpia.co.kr/pdf/pdfView.do?nodeId=NODE09331210&mark=0&useDate=&bookmarkCnt=5&ipRange=N&accessgl=Y&language=ko_KR