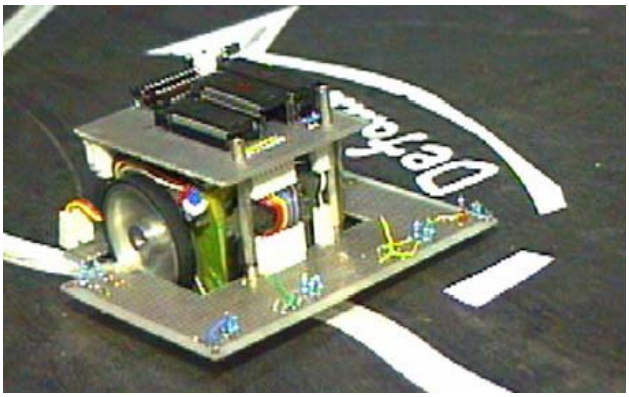
라인트레이서 강좌
4. 프로그래밍

2005년 8월 1일
김민석 (mjkms@hanmail.net)

1. 라인트레이서란?

라인트레이서는 정해진 주행선을 따라 움직이는 자율이동로봇이다. 현재 공장자동화 부분에서 이용되고 있는 무인 반송차가 라인트레이서이다.

라인트레이서의 기본적인 원리는 주어진 주행선을 센서로 검출하여 이것에 따라 목적위치 까지 이동하는 것이다.



라인트레이서는 크게 3부분 - 컨트롤러 부, 센서부, 모터구동부로 구성 된다.

우선 센서부의 센서를 잘 배치해야 길을 잘 인식하고 턴마크 인식할 수 있다.

모터구동부분은 잡음이 많이 생기기 때문에 잘 만들지 않으면 잡음으로 인해 다른 부분이 정상적으로 동작하지 못하는 경우가 생길 수 있다.

컨트롤러부는 기본적인 라인트레이서를 제작할 때는 상대적으로 비중이 낮으나 지능적으로 만들려면 많은 작업이 필요로 한다.

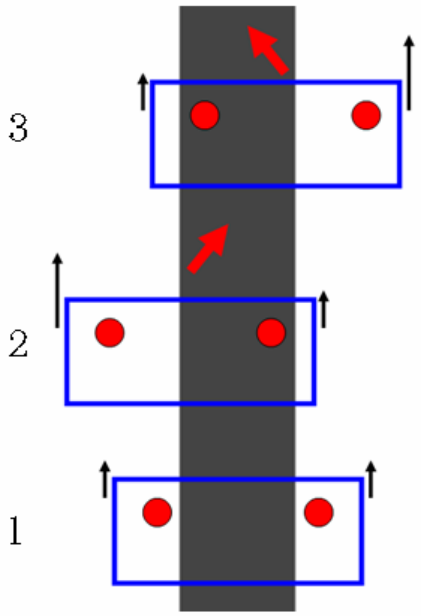
본 강좌에서는 라인트레이서를 공부하여 만들고자 하는 초보자를 위한 강좌이므로 라인트레이서로서의 동작에 목표를 두고 최대한 간단하게 동작시킬 수 있도록 한다.

2. 라인트레이서의 기본 동작 알고리즘

라인트레이서 기본 동작

센서 2쌍을 라인의 두께보다 약간 크게 위치하여 하드웨어를 제작할 경우 센서가 라인 외곽을 따라 주행 하도록 한다.

더 복잡하게 할 수도 있지만 여기서는 위와 같이 가장 간단한 알고리즘으로 알아보겠다.



1 번 상황에서 센서는 라인을 감지하지 못하였으므로 직진한다.

2 번 상황에서 오른쪽 센서가 인식 되었으므로 왼쪽 바퀴의 속도를 증가시켜 라인트레이서를 오른쪽으로 향하게 한다.

3 번 상황에서 왼쪽 센서가 인식 되었으므로 오른쪽 바퀴의 속도를 증가시켜 라인트레이서를 왼쪽으로 향하게 한다.

위의 동작을 반복하면 라인트레이서는 라인을 따라 진행하게 된다.

3. 프로그래밍 하기

기본적인 C 언어 프로그래밍과 인터럽트에 관한 공부를 먼저 해야 한다. C언어와 인터럽트에 관련해서는 관련 서적에서 자세히 나와 있으므로 이 강좌에서는 다루지 않는다.

① 타이머 인터럽트를 이용 하여 스테핑 모터 구동하기

타이머 인터럽트를 학습하여 타이머 인터럽트를 이용하여 스테핑 모터 구동해본다. 라인트레이서는 스테핑 모터를 2개 구동해야 하기 때문에 타이머 인터럽트를 2개 사용하여 타이머 인터럽트 각각 바퀴 1개씩을 구동 하도록 한다.

② 센서입력 받기

센서의 입력을 받아 흰색과 검정색을 구분 할 수 있도록 한다. 시리얼 통신을 이용하여 값을 확인 하면 쉽게 할 수 있다.

③ 메인함수와 인터럽트 루틴간의 데이터 공유

메인함수에서는 속도를 변경하면 타이머 인터럽트에서 변경된 속도값을 타이머에 적용하여 스테핑 모터의 속도를 변경할 수 있도록 한다. 그렇게 하기 위해서는 속도값을 전역변수로 선언하여 메인함수와 인터럽트 루틴이 공유할 수 있도록 해야한다.

④ 센서 입력에 따른 스테핑 모터 속도 변경

왼쪽 센서 입력이 있을 경우 왼쪽 스테핑 모터의 속도를 감소시키고 오른쪽 센서 입력이 있을 경우 오른쪽 스테핑 모터의 속도를 감소시키도록 프로그램을 작성한다.

4. 예제 프로그램

① 타이머 인터럽트를 이용 하여 스테핑 모터 구동하기

아래 소스는 타이머 2 인터럽트의 초기화 루틴이다.

```
void timer2_init(void)
{
        TCCR2 = 0x00; //stop
        TCNT2 = 0xF1; //setup
        OCR2   = 0x0F;
        TCCR2 = 0x04; //start
}
```

아래 소스는 타이머 2 인터럽트의 인터럽트 서비스 루틴이다.

```
void timer2_ovf_isr(void)
{
        unsigned char temp;

        CLI();              //disable all interrupts
        temp = PORTD & 0x0f;

        TCNT2 = l_speed;  //reload counter value
        if(l_speed == 0){
                temp = temp;
        }else{
                temp = motorOneClock(temp, 1);
        }
        PORTD = temp | (PORTD & 0xf0);
        SEI();              //re-enable interrupts
}
```

② **센서입력 받기**

아래 소스는 AD Convert 의 구동 소스이다.

startConvertion 함수는 해당 채널의 컨버팅을 시작하라는 함수이고 readConvertData 함수는 컨버팅한 채널의 데이터를 읽어오라는 함수이다.

```
void startConvertion(unsigned char ch)
{
        ADMUX = 0x20 | (ch & 0x0f);
        ADCSRA = 0xc7;
}

unsigned char readConvertData(void)
{
        volatile unsigned char temp;
        while((ADCSRA & 0x10)==0);
        ADCSRA = ADCSRA | 0x10;
        temp = ADCH;
        temp = ADCL;
        temp = ADCH;
        return temp;
}
```

③ **메인함수 루틴**

다음 소스는 메인 함수에서 수행되는 무한루프로 센서가 연결되어 있는 6번과 7번 센서를 컨버팅하여 변수에 저장시키고 이것을 임계값과 비교하여 왼쪽바퀴와 오른쪽 바퀴의 속도를 결정하도록 하였다.

```
void main(void)
{
        volatile unsigned char temp, l_sensor, r_sensor;
        volatile unsigned char step=0;

        ……

        while(1){

                l_sensor = 0;
                r_sensor = 0;
```

```
                startConvertion(6);             // Left sensor Converting
                l_sensor = readConvertData();
                startConvertion(7);             // Right sensor Converting
                r_sensor = readConvertData();

                PORTF = 0x03;

                if(l_sensor > THRESHOLD && r_sensor < THRESHOLD){
                        l_speed = SLOW_SPEED;
                        r_speed = FAST_SPEED;
                        PORTF = 0x01;
                }else if(l_sensor < THRESHOLD && r_sensor > THRESHOLD){
                        l_speed = FAST_SPEED;
                        r_speed = SLOW_SPEED;
                        PORTF = 0x02;
                }else{
                        l_speed = FAST_SPEED;
                        r_speed = FAST_SPEED;
                }

                printf("ch6 :%d     ch7 : %d\n\r", l_sensor, r_sensor);
        }
}
```

5. 소스 코드

```
// LineTracer Application
// Target : M128
// Crystal: 11.059Mhz
// Programed by Minsuk Kim

#include <iom128v.h>
#include <macros.h>

#define   FAST_SPEED     0x80     // 센서가 감지되었을때 빠른 바퀴의 센서값
#define   SLOW_SPEED     0x10     // 센서가 감지되었을때 느린 바퀴의 센서값
```

```
#define THRESHOLD          17          // 센서의 라인 감지 임계값

unsigned char l_speed=0, r_speed=0;   // 모터를 구동 위한 타이머 값

void port_init(void)
{
        PORTA = 0x00;
        DDRA  = 0x00;
        PORTB = 0x00;
        DDRB  = 0x00;
        PORTC = 0x00;
        DDRC  = 0x00;
        PORTD = 0x00;
        DDRD  = 0xff;               // Stepping Motor Port Left, Right
        PORTE = 0x00;
        DDRE  = 0x00;
        PORTF = 0x00;
        DDRF  = 0x03;               // LED Port PF1, PF2
        PORTG = 0x00;
        DDRG  = 0x00;
}

//UART0 initialize
// desired baud rate: 9600
// actual: baud rate:9600 (0.0%)
// char size: 8 bit
// parity: Disabled
void uart0_init(void)
{
        UCSR0B = 0x00; //disable while setting baud rate
        UCSR0A = 0x00;
        UCSR0C = 0x06;
        UBRR0L = 0x47; //set baud rate lo
        UBRR0H = 0x00; //set baud rate hi
        UCSR0B = 0x08;
}
```

```
//ADC initialize
// Conversion time: 75uS
void adc_init(void)
{
	ADCSRA = 0x00; //disable adc
	ADMUX = 0x00; //select adc input 0
	ACSR  = 0x80;
	ADCSRA = 0xC6;
}

//TIMER2 initialize - prescale:64
// WGM: Normal
// desired value: 1KHz
// actual value:  1.008KHz (0.8%)
void timer2_init(void)
{
	TCCR2 = 0x00; //stop
	TCNT2 = 0xF1; //setup
	OCR2  = 0x0F;
	TCCR2 = 0x04; //start
}


#pragma interrupt_handler timer2_ovf_isr:11
void timer2_ovf_isr(void)
{
	unsigned char temp;

	CLI(); //disable all interrupts
	temp = PORTD & 0x0f;

	TCNT2 = l_speed;  //reload counter value
	if(l_speed == 0){
		temp = temp;
	}else{
```

```
                temp = motorOneClock(temp, 1);
        }
        PORTD = temp | (PORTD & 0xf0);
        SEI();                          //re-enable interrupts
}

//TIMER3 initialize - prescale:64
// WGM: 0) Normal, TOP=0xFFFF
// desired value: 1KHz
// actual value:    1.008KHz (0.8%)
void timer3_init(void)
{
        TCCR3B = 0x00; //stop
        TCNT3H = 0xFF; //setup
        TCNT3L = 0xF1;
        OCR3AH = 0x00;
        OCR3AL = 0x0F;
        OCR3BH = 0x00;
        OCR3BL = 0x0F;
        OCR3CH = 0x00;
        OCR3CL = 0x0F;
        ICR3H    = 0x00;
        ICR3L    = 0x0F;
        TCCR3A = 0x00;
        TCCR3B = 0x04; //start Timer
}

#pragma interrupt_handler timer3_ovf_isr:30
void timer3_ovf_isr(void)
{
        unsigned char temp;

        CLI();      //disable all interrupts
                    //TIMER3 has overflowed
        TCNT3H = 0xFF; //reload counter high value
```

```
	temp = (PORTD & 0xf0)>>4;

	TCNT3L = r_speed; //reload counter low value
	if(r_speed == 0){
		temp = temp;
	}else{
		temp = motorOneClock(temp, 0);
	}
	PORTD = (temp<<4) | (PORTD & 0x0f);
	SEI(); //re-enable interrupts
}

//call this routine to initialize all peripherals
void init_devices(void)
{
	//stop errant interrupts until set up
	CLI(); //disable all interrupts
	XDIV    = 0x00; //xtal divider
	XMCRA = 0x00; //external memory
	port_init();
	uart0_init();
	adc_init();
	timer2_init();
	timer3_init();

	MCUCR = 0x00;
	EICRA = 0x00; //extended ext ints
	EICRB = 0x00; //extended ext ints
	EIMSK = 0x00;
	TIMSK = 0x40; //timer interrupt sources
	ETIMSK = 0x04; //extended timer interrupt sources
	SEI(); //re-enable interrupts
	//all peripherals are now initialized
}
```

```
/* Analog Converter--------------------------------*/
void startConvertion(unsigned char ch)
{
        ADMUX = 0x20 | (ch & 0x0f);
        ADCSRA = 0xc7;
}

unsigned char readConvertData(void)
{
        volatile unsigned char temp;
        while((ADCSRA & 0x10)==0);
        ADCSRA = ADCSRA | 0x10;
        temp = ADCH;
        temp = ADCL;
        temp = ADCH;
        return temp;
}
/*-------------------------------------------------*/

void delay(int n)
{
        volatile int i,j;
        for(i=1;i<n;i++)
        {
        for(j=1;j<600;j++);
        }
}

/* Stepping Motor derive----------------------------*/
unsigned char    motorOneClock(unsigned char step, char dir)
{
        unsigned char test;

        step = step & 0x0f;
        if(dir){
                switch(step){
```

```
                        case 0x03 : step=0x06; break;
                        case 0x06 : step=0x0c; break;
                        case 0x0c : step=0x09; break;
                        case 0x09 : step=0x03; break;
                        default   : step=0x03; break;
                }
        }else{
                switch(step){
                        case 0x03 : step=0x09; break;
                        case 0x06 : step=0x03; break;
                        case 0x0c : step=0x06; break;
                        case 0x09 : step=0x0c; break;
                        default   : step=0x03; break;
                }
        }
        return step;

}


/* main function-----------------------------------*/
void main(void)
{
        volatile unsigned char temp, l_sensor, r_sensor;
        volatile unsigned char step=0;

        init_devices();
        //insert your functional code here...
        printf(" \n\n\rtest program...\n\r");

        PORTF = 0x03;
        delay(1000);
        PORTF = 0x00;
        delay(1000);
        PORTF = 0x03;
```

```
        while(1){

                l_sensor = 0;
                r_sensor = 0;
                startConvertion(6); // Left sensor Converting
                l_sensor = readConvertData();
                startConvertion(7); // Right sensor Converting
                r_sensor = readConvertData();

                PORTF = 0x03;

                if(l_sensor > THRESHOLD && r_sensor < THRESHOLD){
                        l_speed = SLOW_SPEED;
                        r_speed = FAST_SPEED;
                        PORTF = 0x01;
                }else if(l_sensor < THRESHOLD && r_sensor > THRESHOLD){
                        l_speed = FAST_SPEED;
                        r_speed = SLOW_SPEED;
                        PORTF = 0x02;
                }else{
                        l_speed = FAST_SPEED;
                        r_speed = FAST_SPEED;
                }

                printf("ch6 :%d      ch7 : %d\n\r", l_sensor, r_sensor);
        }
}
```

6. Epilogue

하드웨어를 공부하는데 있어서 펌웨어 프로그래밍은 매우 중요한 부분이다.
본인 스스로 프로그램을 작성하여 멋진 라인트레이서를 만들어 보기 바란다.