〈마이크로 프로세서 및 실습〉

– 라인 트레이서 결과 보고서 –

지도 교수	임영훈 교수님
학번	21725118
이름	류 예 준

<목차>

〈# 실험 코드〉

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

volatile int i = 0;
/* PORT E 모터 출력 PORT B 센서 입력*/
/* OCR3A 좌측 바퀴 OCR3B 우측 바퀴 */
void INIT_TIMER(void)
{
    TCCR3A |= (1<<WGM30) | (1<<COM3A1) | (1<<COM3B1); // 타이머카운터 설정. 8bit Fast PWM모드, 256분주율.
    TCCR3B |= (1<WGM32) | (1<<CS32);
    OCR3A = 0;
    OCR3B = 0;
}
void forward(void)
{
    if(i == 0){
        for(i = 60; i < 124; i += 1) // 전진 속도가 서서히 증가.
        {
            OCR3A = i;
            OCR3B = i;
            if((PINB & 0x02) == 0x02) break;
            if((PINB & 0x01) == 0x01) break;
            if((PINB & 0x03) == 0x03) break;
            _delay_ms(40);
        }
    }
}
void right(void)
{
    OCR3A = 80;
    OCR3B = 0;
    i = 0;
}
void left(void)
{
    OCR3A = 0;
    OCR3B = 80;
    i = 0;
}
void stop(void)
{
    OCR3A = 0;
    OCR3B = 0;
    i = 0;
}

void PORT_INIT(void)
{
    DDRE = 0xff;
    PORTE = 0xff;
    DDRB = 0x00;
    PORTB = 0x00;
}

int main(void)
{
    PORT_INIT();
    INIT_TIMER();
    _delay_ms(2000); // 스위치를 누르면 2초후에 출발하도록 딜레이 설정.

    while(1){

        if((PINB & 0x03) == 0x03)
        {
            stop();
        }
        else if((PINB & 0x02) == 0x02)
        {
            right();
        }
        else if((PINB & 0x01) == 0x01)
        {
            left();
        }
        else if((PINB & 0x00) == 0x00)
        {
            forward();
        }

    }
    return 0;
}
```

우회전 시 좌측바퀴만 구동,
우측바퀴는 정지하는
방법을 사용하였음.
좌회전시는 그 반대.

라인트레이서가 트랙을 벗어나면
정지할 수 있도록 하였음.

센서의 입력값을 PINB로 받아들임.
검은색 표면에서 센서값 0,
그 외의 색에서 센서값 1인 것을 활용.

PORTB: 0번핀에 우측 센서,
1번핀에 좌측 센서 연결함.
0x03 = 0000 0011 (정지)
0x02 = 0000 0010 (우회전)
0x01 = 0000 0001 (좌회전)
0x00 = 전진

〈# 결과〉

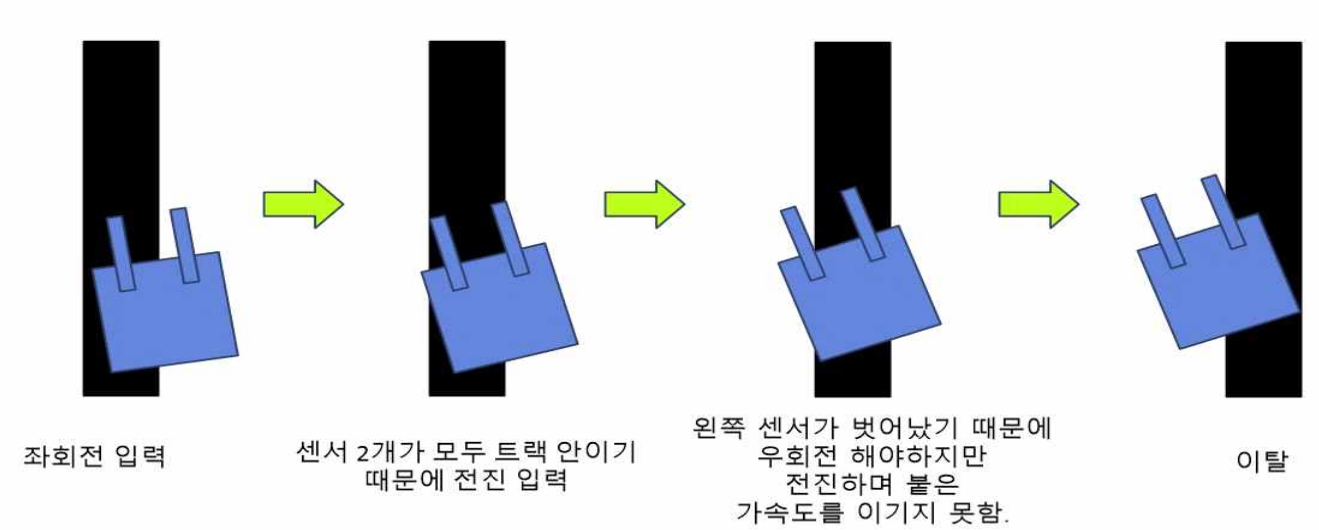
- 실험 결과 영상 : https://www.youtube.com/watch?v=a-i77yx5xLs

실험 내용에 대한 고찰

1번째 실험: 부저음이 들리며 모터가 작동하지 않음.
- 해결: 분주율 설정이 잘못되어 있었다. 64->256으로 분주율 수정하여 문제 해결.

2번째 실험: 모터의 이상작동(한 쪽만 작동, 급발진) 및 전원 꺼짐.
- 해결: ATmega128 보드의 LED가 평소보다 희미하게 점등되어 있는걸 확인했고 배터리 잔량 문제라 판단. 9V 배터리를 구입 후 교체하니 해결되었다. 1번째 실험의 해결방법을 찾는 과정에서 배터리를 너무 많이 소모한 것 같았다.

3번째 실험: 커브 구간에서의 트랙 이탈 현상
- 해결: 커브 구간에서 트랙 이탈이 잦게 일어나는 것을 보고 원인을 분석해본 결과, 다음 사진과 같은 원인 때문이라는 결론을 내렸다.



첫 번째로 전진하는 속도를 줄이는 방법을 적용해보았다. 그러자 트랙 이탈 문제는 해결이 되었지만 라인트레이서의 속도가 너무 느렸다. 그래서 두 번째로 전진 속도가 시간이 지남에 따라 증가되는 방법을 적용하였다. 그러자 긴 직선구간에서는 빠른 속도를 유지하고 커브 구간에서 전진이 약하게 일어나며 트랙 이탈없이 정상적으로 주행하는 결과를 얻을 수 있었다.

```
void forward(void)
{
    OCR3A = 124;
    OCR3B = 124;
}
```

```
void forward(void)
{
    if(i == 0){
        for(i = 60; i < 124; i += 1) // 전진 속도가 서서히 증가.
        {
            OCR3A = i;
            OCR3B = i;
            if((PINB & 0x02) == 0x02) break;
            if((PINB & 0x01) == 0x01) break;
            if((PINB & 0x03) == 0x03) break;
            _delay_ms(40);
        }
    }
}
```

- **전진방법 구현 : 전 / 후**

총평

이번 실험을 진행하며 인터넷 상에서 라인트레이서 경주 대회
영상을 참고하여 최대한 비슷하게 구현하려 노력을 많이 했지만
아무리 코드를 수정해봐도 동영상에 나오는 라인트레이서는 카트라이더 같았고
나의 라인트레이서는 운전 면허 시험장에서 기능시험을 보는 차와 같았다.
길게 고민한 결과,
원인은 IR센서가 2개이기 때문에 좌/우회전의 각도를 1개씩 밖에
설정하지 못해서였던 것 같다. 만약 센서를 3개, 5개로 늘린다면
트랙에서 벗어난 정도를 더 정밀하게 알 수 있을 것이고
따라서 트랙에서 벗어난 정도에 따라 필요한 만큼만의 회전각을 설정하여
더 좋은 라인트레이서를 만들 수 있을 것이라는 생각이 들었다.