chapter 6. 마이크로 마우스 보정

1. 서론

-이번 실험에선 마이크로 마우스 보정에대해 실험하였다.

2. 본론

1. 센서 정밀도 향상

(1) 마이크로 마우스 센서 부위의 빛을 물리적으로 차단

수축튜브 등을 이용하여 센서에 끼워주면 주변의 물리적인 빛을 어느 정도 차단할 수 있다. 이것은

후에 주행중 예상치 못한 빛의 영향을 줄여주게 되어 보다 안정적인 전진 주행을 하도록 돕는다.

(2) 마우스 수광 센서의 디지털 필터

디지털 필터는 크게 IIR 필터와 FIR 필터로 나눌 수 있다. 우리는 IIR 필터의 일종인 센서데이터의

평균화 작업을 통하여 데이터를 흔들리지 않게 할 수 있다. 예를 들면 2의 T배승에 해당하는

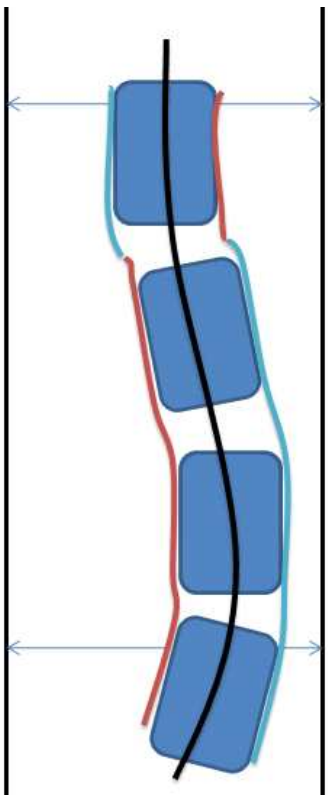
데이터를 모두 다 더해주고 이것을 다시 한 개의 데이터처럼 보기 위하여 T만큼 왼쪽으로 SHIFT해

주면 데이터는 보다 안정감 있는 흔들리지 않는 데이터를 출력할 것이다.

2. 보정

(1) 일반적으로 마이크로 마우스가 중간값을 가지는 세팅을 정하여 준다.

(2) 이 중간값을 이용하여 마우스가 중간으로 가도록 보정을 할 수 있다.



좌측 그림을 보자. 빨간 라인의 바퀴는 속도가 감속되는 것이고 파란 라인의 바퀴는 속도가 증가되는 것이다. 마우스는 현재의 센서를 확인하여 왼쪽 센서에서 벽과의 거리가 멀다면 왼쪽 모터의 속도를 낮춰주고 오른쪽 모터의 속도를 높여 주도록 한다. 만약 오른쪽 센서에서 벽과의 거리가 멀다면 오른쪽 모터의 속도를 낮춰주고 오른쪽 모터의 속도를 높여 주도록 한다.

이러한 보정 작업이 반복되면 마우스는 원래 처음 세팅한 중간값의 위치로 마우스가 위치하게 될 것이다.

3. 실험결과

```
int adjustmouse(void)
{
	int adjLeftSensor,adjRightSensor;
	int adjflagcnt = 0;

	adjLeftSensor = readSensor(LEFT_SENSOR);
	adjRightSensor = readSensor(RIGHT_SENSOR);

	if((adjRightSensor < StandardSensor[2])		// 오른쪽 벽이 존재하지 않을 경우
	|| (adjLeftSensor < StandardSensor[1]))			// 왼쪽 벽이 존재하지 않을 경우
	{
		vel_counter_high_L = vel_counter_high;  // 속도를 같게하고 리턴
		vel_counter_high_R = vel_counter_high;
		return 0;
	}

	if(adjRightSensor < CenterStandardSensor[2])	// 오른쪽 벽이 멀 경우
	{
		vel_counter_high_L+=1;
		vel_counter_high_R-=1;
		if(vel_counter_high_L > vel_counter_high+20)
		{
			vel_counter_high_L = vel_counter_high+20;
		}

		if(vel_counter_high_R < (vel_counter_high - 20))
		{
			vel_counter_high_R = (vel_counter_high - 20);
		}
	}
	else
	adjflagcnt++;

	if(adjLeftSensor < CenterStandardSensor[1])	// 왼쪽 벽이 멀 경우
	{
		vel_counter_high_L-=1;
		vel_counter_high_R+=1;
		if(vel_counter_high_R > vel_counter_high+20)
		{
			vel_counter_high_R = vel_counter_high+20;
		}
		if(vel_counter_high_L < (vel_counter_high - 20))
		{
			vel_counter_high_L = (vel_counter_high - 20);
		}
	}
	else
	adjflagcnt++;

	if(adjflagcnt == 2)				// 오른쪽 벽과 왼쪽 벽이 둘다 멀지 않을 경우
	{						// 속도 동일하게
		vel_counter_high_L = vel_counter_high;
		vel_counter_high_R = vel_counter_high;
		return 0;
	}
		return 1;
}
```

연습문제

1. 모드 프로그램을 작성하여 센싱값을 받고 마이크로 마우스의 보정 프로그램을 작성하여라. (5블럭 전진)

```
//위의 adjustmouse함수를 그대로 motor.c에 추가해주고 직진코드인 Direction함수에서
//모드 HALF부분을 adjustmouse로 수정해준다.
//case HALF:
while(LStepCount<(Information.nStep4perBlock>>1)                                    ||
RStepCount<(Information.nStep4perBlock>>1))
{
if(Flag.LmotorRun||Flag.RmotorRun)
{
adjustmouse();
VelocityLeftmotorTCNT1=vel_counter_high_L;
VelocityRightmotorTCNT3=vel_counter_high_R;
}
if(Flag.LmotorRun)
{
LStepCount++;
Flag.LmotorRun = FALSE;
adjustmouse();
VelocityLeftmotorTCNT1=vel_counter_high_L;
VelocityRightmotorTCNT3=vel_counter_high_R;
}
if(Flag.RmotorRun)
{
RStepCount++;
Flag.RmotorRun = FALSE;
adjustmouse();
VelocityLeftmotorTCNT1=vel_counter_high_L;
VelocityRightmotorTCNT3=vel_counter_high_R;
}
}
break;
//이렇게 수정해주면 이제 좌우에 벽이 얼만큼 가까운지 즉 있는지없는지를 판단해서
//좌우바퀴의 속도를 따로따로 조절해주어 직진하게 바꿀 수 있다. 이렇게 설정한후
//메인문의 모드3에서 Direction(HALF);를 10번 쳐주면 5블럭을 보정된 직진으로
//수행하게 된다.
```

3. 보정관련 속도 보정변수 조절과정 및 결과

```
int adjustmouse(void)
{
        int adjLeftSensor,adjRightSensor;
```

```
	int adjflagcnt = 0;

	adjLeftSensor = readSensor(LEFT_SENSOR);
	adjRightSensor = readSensor(RIGHT_SENSOR);

	if((adjRightSensor < StandardSensor[2])		// 오른쪽 벽이 존재하지 않을
경우
	|| (adjLeftSensor < StandardSensor[1]))			// 왼쪽 벽이 존재하지
않을 경우
	{
		vel_counter_high_L = vel_counter_high;  // 속도를 같게하고 리턴
		vel_counter_high_R = vel_counter_high;
		return 0;
	}

	if(adjRightSensor < (CenterStandardSensor[2])) // 오른쪽 벽이 멀 경우
	{				//
		vel_counter_high_L+=1;
		vel_counter_high_R-=1;
		if(vel_counter_high_L > vel_counter_high+5)
		{
			vel_counter_high_L = vel_counter_high+5;
		}

		if(vel_counter_high_R < (vel_counter_high - 5))
		{
			vel_counter_high_R = (vel_counter_high - 5);
		}
	}
	else
	adjflagcnt++;

	if(adjLeftSensor < (CenterStandardSensor[1]))  // 왼쪽 벽이 멀 경우
	{

		vel_counter_high_L-=1;
		vel_counter_high_R+=1;
		if(vel_counter_high_R > vel_counter_high+5)
		{
			vel_counter_high_R = vel_counter_high+5;
		}
```

```
                if(vel_counter_high_L < (vel_counter_high - 5))
                {
                        vel_counter_high_L = (vel_counter_high - 5);
                }
        }
        else
        adjflagcnt++;

        if(adjflagcnt == 2)                                    // 오른쪽 벽과 왼쪽 벽이 둘다
멀지 않을 경우
        {                                                              // 속도 동일하게
                vel_counter_high_L = vel_counter_high;
                vel_counter_high_R = vel_counter_high;
                return 0;
        }
                return 1;
}
```

- 본인의 보정함수 코딩으로 본인마우스는 돌아가지않앗으나 남의 보정마우스는
- 제대로 작동됨을 확인하였다. +-20은 너무커서 섹터를 2개로 분배한후 너무멀땐 10 가까울땐 5로 증감되게 설정하였다.

4.현재 하드웨어 제작상태

- PCB동작가능,만능기판 불가능 PCB는 마우스바디가좀이상하나 무리없이동작하고
-만능기판은 다시 납땜 하려고합니다.

4. 고찰

이번실험에서 보정관련을 충분히 이해하였다. 얼른 손봐서 제대로 굴리고싶다.

5. 참고 문헌