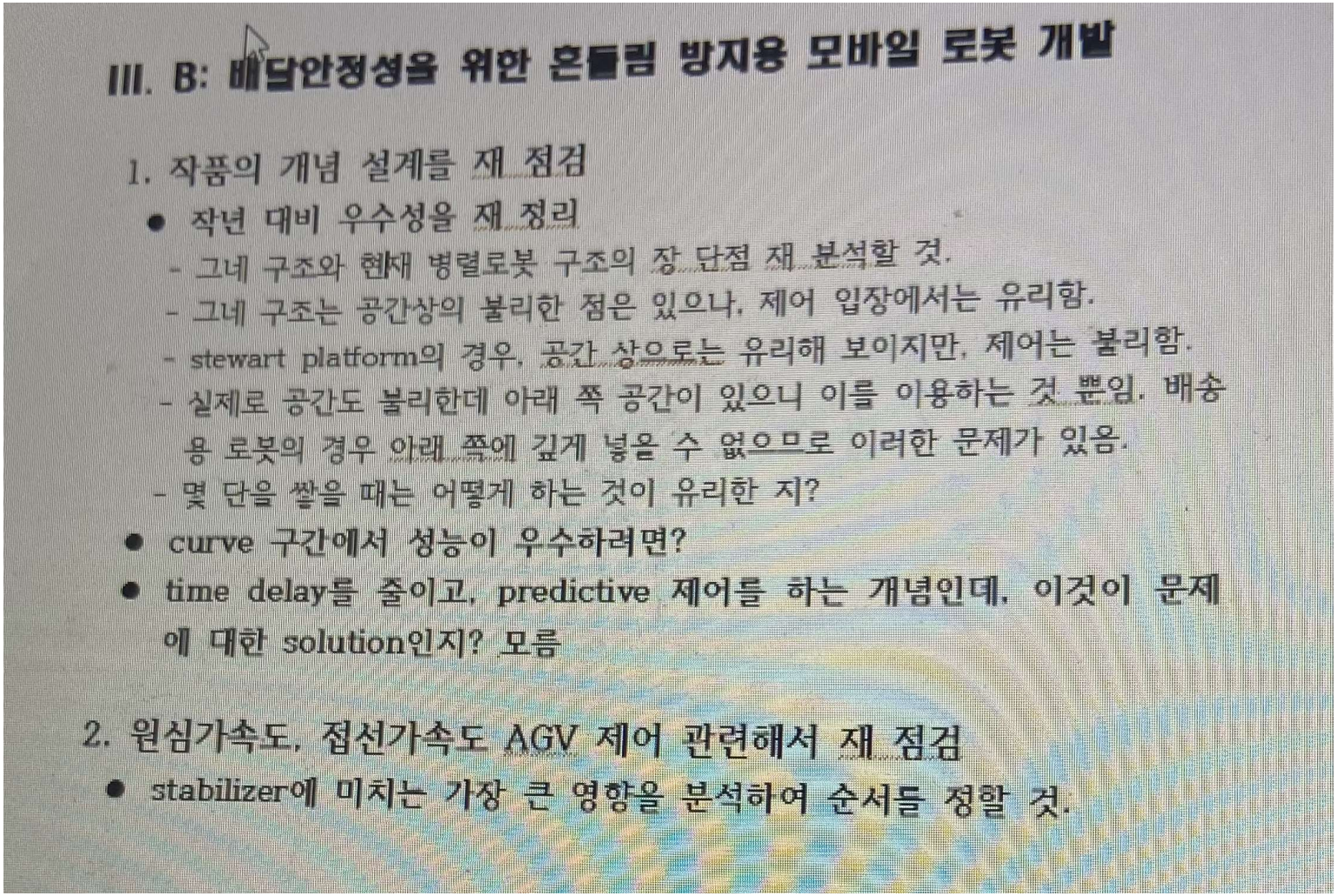
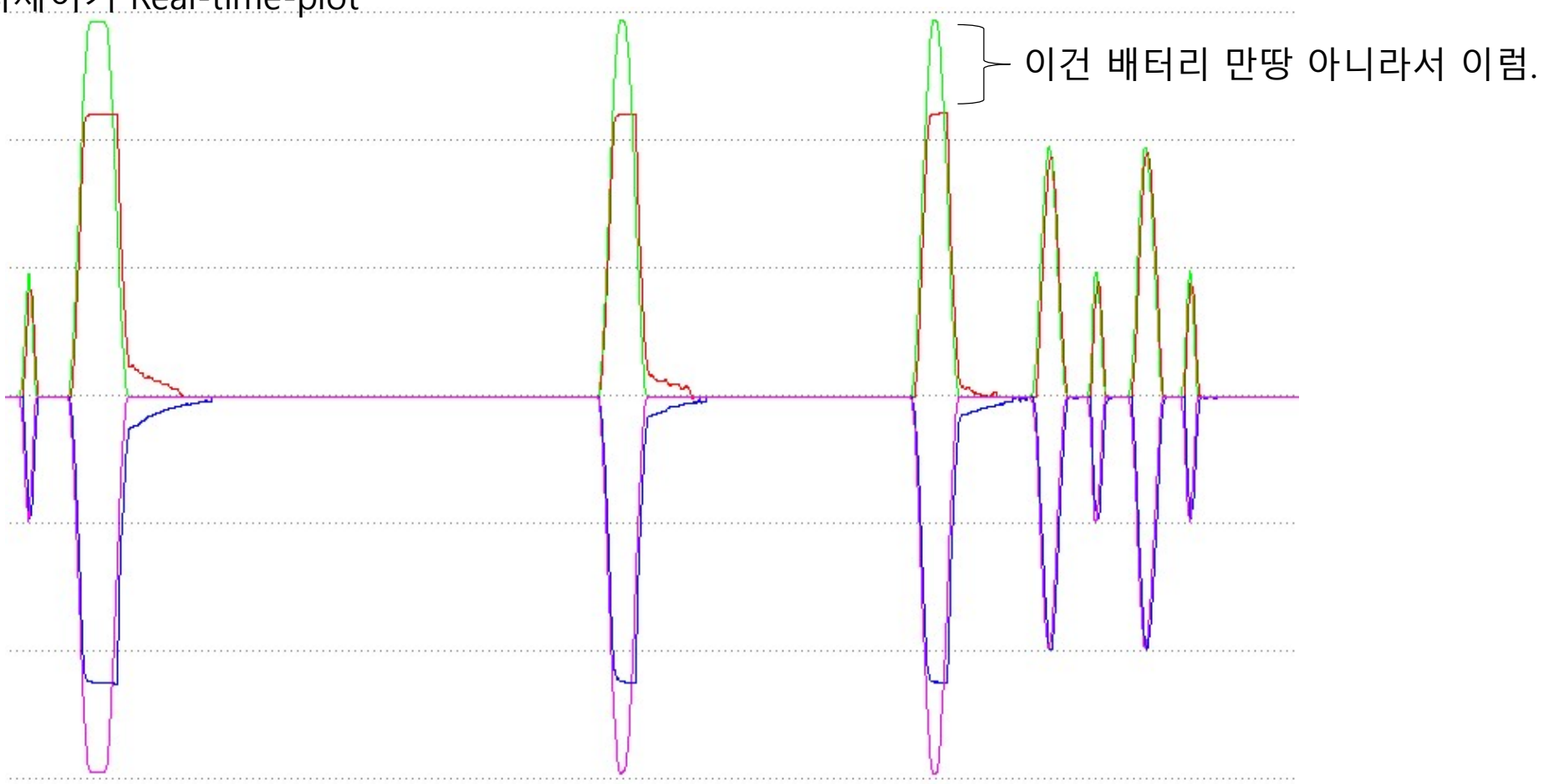
4월 8일 미팅(4월 2주차)

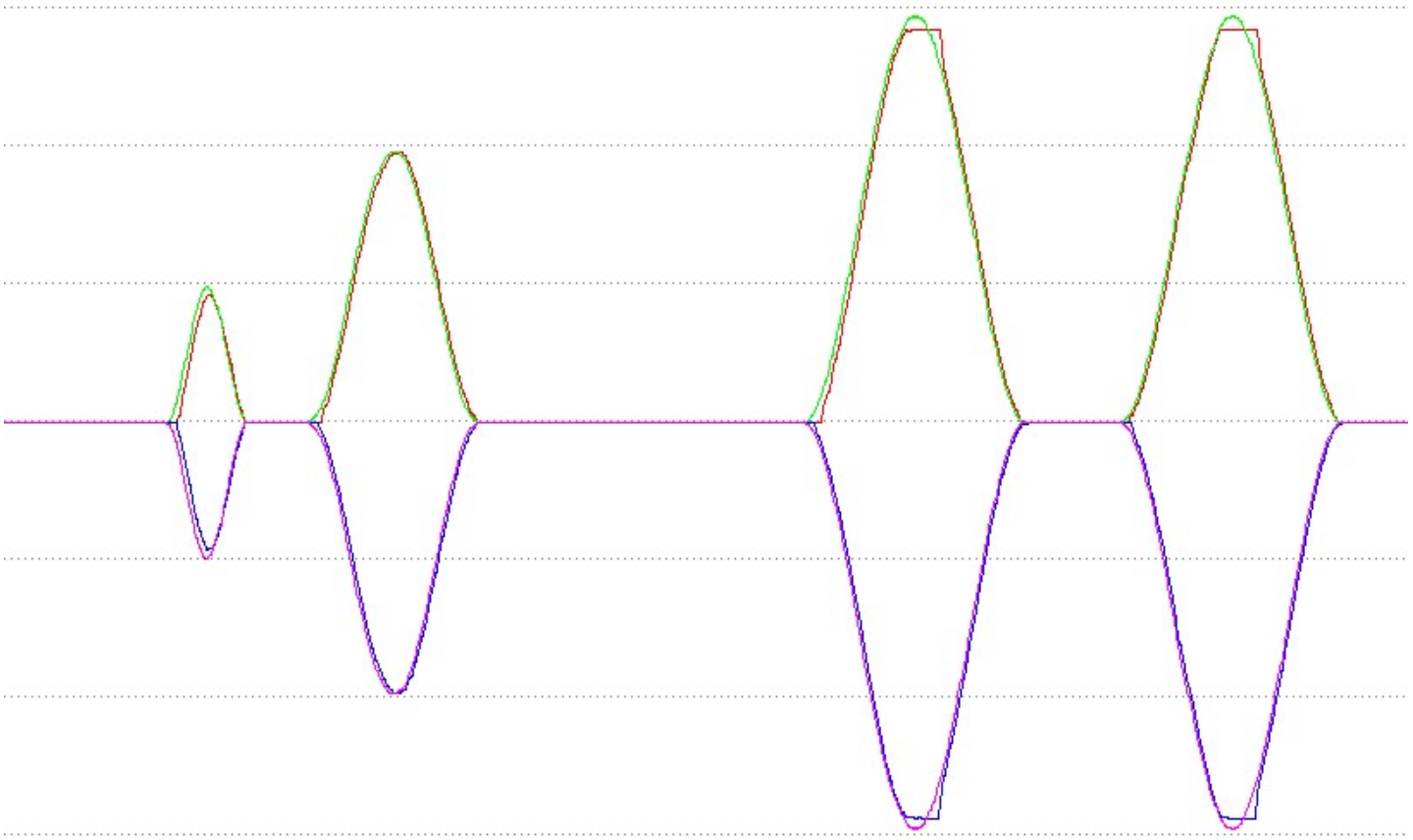


모터제어기 Real-time-plot



3ms까지 가속했다가 감속하면 감속할 때 끝부분에서 잔류속도가 생긴다.
이유는?
생각 1. 무게를 주지 않고 공중에 띄어서 구동해 봤을 때의 결과이다. 따라서 무게를 쫌 주면 괜찮지 않을까?
생각 2.

모터제어기 Real-time-plot



배터리 문제였다.
배터리를 충전한후 다시 해보니 되었다.

❖주행시간 조정

Scale과 delay
로 시간 맞추기

<millis()함수로 시간재봄.>
-millis(): ms 단위로 출력.

◆ 3ms 속도(목표:가속2초+감속2초=4초)

[Scale=명령간격, delay_time(한번명령유지시간)]

Scale=0.01; delay_time=15;
-> 가속+감속=6665ms

Scale=0.02; delay_time=15;
-> 가속+감속=3357ms

Scale=0.02; delay_time=20;
-> 가속+감속=4373ms

Scale=0.02; delay_time=18;
-> 가속+감속=3967ms(4초근접)

◆ 1ms 속도(목표:가속0.5초+감속0.5초=1초)

[Scale=명령간격, delay_time(한번명령유지시간)]

Scale=0.01; delay_time=10;
-> 가속+감속=1185ms

Scale=0.01; delay_time=9;
-> 가속+감속=1083ms(1초근접)

◆ 2ms 속도(목표:가속1초+감속1초=2초)

[Scale=명령간격, delay_time(한번명령유지시간)]

Scale=0.02; delay_time=10;
-> 가속+감속=1199ms

Scale=0.01; delay_time=10;
-> 가속+감속=2350ms

Scale=0.01; delay_time=8;
-> 가속+감속=1943ms(2초근접)

➡ Scale과 delay둘중에 어느것을 조절해야 더 좋은가?

→그래프의 모양을 보고 판단 해보자? Scale은 쪼갤수록 여러 간격으로 모터에 명령을 전달. Delay는 길게 할수록 한 속도를 오래 유지.
어떤 것이 모터에 좀더 안 좋은 영향을 끼칠까?

❖4월 마지막주

❖4월 마지막주

1. ‘AGV+짐벌’ 합친 코드 딜레이 분석
2. AGV경로 알고리즘 구성

❖AGV모터제어기

2.2 전원 연결

이 절은 제어기와 전원을 보호하기 위한 회로 설계와 배선 방법 그리고 안전하게 전원을 ON/OFF 하는 방법에 대해 설명합니다.

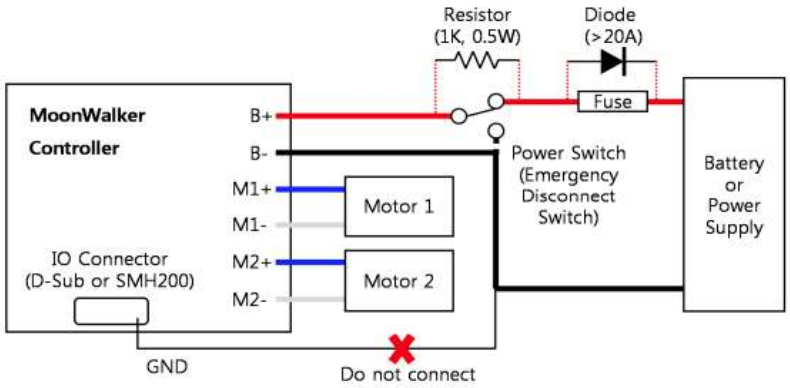


그림 2-3 제어기 및 전원 보호 회로 구성

그림 2-3 제어기 및 전원 보호 회로 구성에서 실선으로 표시된 연결은 필수사항이고 점선으로 표시된 연결은 권장사항입니다. I/O 커넥터는 제어기 모델에 따라 D-Sub과 SMH200 커넥터로 구분됩니다. 제어기의 데이터시트를 참조하기 바랍니다.

※주의※ 그림 2-3은 제어기를 사용하기 위한 기본적인 전원 연결에 대한 회로도 입니다. 이는 단순한 참조 예시일 뿐이며, 사용자는 제품을 사용할 환경에 맞게 보호 회로를 설계하여야 합니다. 다시 한번 강조하자면, 그림 2-2와 같은 보호 회로 구성이 모든 환경에 대해 정상적으로 대응 또는 동작하는 것은 아닙니다. 사용자의 환경에 맞게 회로를 구성하기 바랍니다. 사용자의 잘못된 연결로 인한 제어기의 손상 및 파손은 제품 보증이 적용되지 않습니다.

2.2.2 퓨즈와 다이오드 삽입

퓨즈는 주로 과전압과 과전류로 인한 전원 배선 및 배터리의 손상을 보호합니다. 손상을 방지하기 위해 안전 조치로 ATO 또는 MAXI 시리즈보다 높은 전류의 퓨즈(Fuse) 사용을 권장합니다. 퓨즈를 선정할 때는 안전을 위해 제어기에 연결된 모터의 용량 합에서 80~90% 사이의 용량을 선택하는 것을 권장합니다.

참고로 40A 이상의 퓨즈는 일반적으로 끊어지는 속도가 느리기 때문에 제어기를 보호하는데 한계가 있습니다. 퓨즈가 끊어진 경우에도 배터리로의 반환 경로를 보장하기 위해서 고전류 다이오드를 퓨즈와 병렬로 연결해야 합니다.

※주의※ 일반적으로 퓨즈를 전원에 연결하면 제품을 안전하게 보호할 수 있다고 생각합니다. 하지만 이는 잘못된 생각입니다. 퓨즈는 일반적으로 탈 때까지 약간의 시간이 걸리기 때문에 일시적으로 퓨즈를 통해 높은 전류가 흐를 수 있습니다. 이러한 경우, 모터(구동 제품)의 가속이나 동작 시 일시적으로 높은 전류를 끌어다 쓸 수 있는 장점도 있지만, 반대로 퓨즈가 제어기를 제대로 보호하지 못하기도 합니다.

5.1.1 최대 전류와 최대 전압

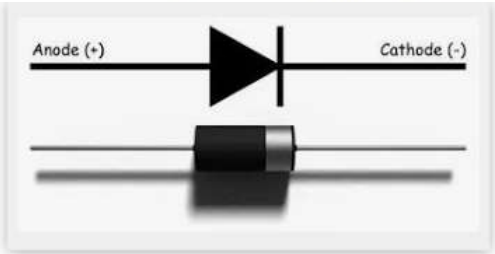
'Max Current'는 보통 정격 부하가 있을 때 허용되는 전류입니다. 모터제어기의 페루프 전류제어기는 최대 허용 전류를 넘어가지 않도록 모터에 흐르는 전류를 제어합니다.

DC 모터에 흐르는 전류는 발생하는 토크와 비례 관계에 있습니다. 그리고 모터의 토크는 보통 회전 속도에 반비례합니다. 모터 기동 시 많은 전류가 흐르면서 큰 토크가 가해집니다. 모터가 최고 속도로 회전하면 모터에 가해지는 전압과 역기전력이 비슷해지고 모터에 흐르는 전류는 낮아집니다. 이때 모터가 낼 수 있는 토크는 작아집니다.

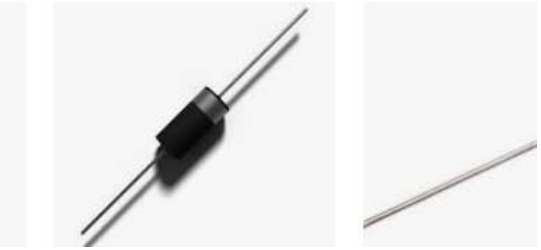
'Max Voltage'는 모터에 가장 높게 가해질 수 있는 정격 전압입니다. 부하가 일정할 때, 모터의 회전속도는 모터에 가해지는 전압에 비례합니다. 만일 모터에 정격 전압 이상을 가하게 되면 모터의 속도가 정격 속도 이상으로 증가합니다. 하지만 모터에서 열이 발생하고 수명을 단축하게 됩니다. 극단적으로는 모터가 탈 수도 있습니다.

모터의 최대 허용 전류와 전압은 Motor Control UI 유틸리티를 통해 제어기에 연결된 각각의 모터에 설정할 수 있습니다. 모터의 손상을 방지하기 위해, 모터의 데이터시트를 참고하여 올바른 값을 설정해야 합니다.

다이오드(**diode**)는 저마늄(영어: germanium 또는 게르마늄(독일어: germanium), Ge)이나 규소(Si)로 만들어지고, 주로 한쪽 방향으로 전류가 흐르도록 제어하는 반도체 소자를 말한다. 정류, 발광 등의 특성을 지니는 반도체 소자이다.



Diode: Definition, Symbol, and Types of Diodes | Electrical4U
electrical4u.com



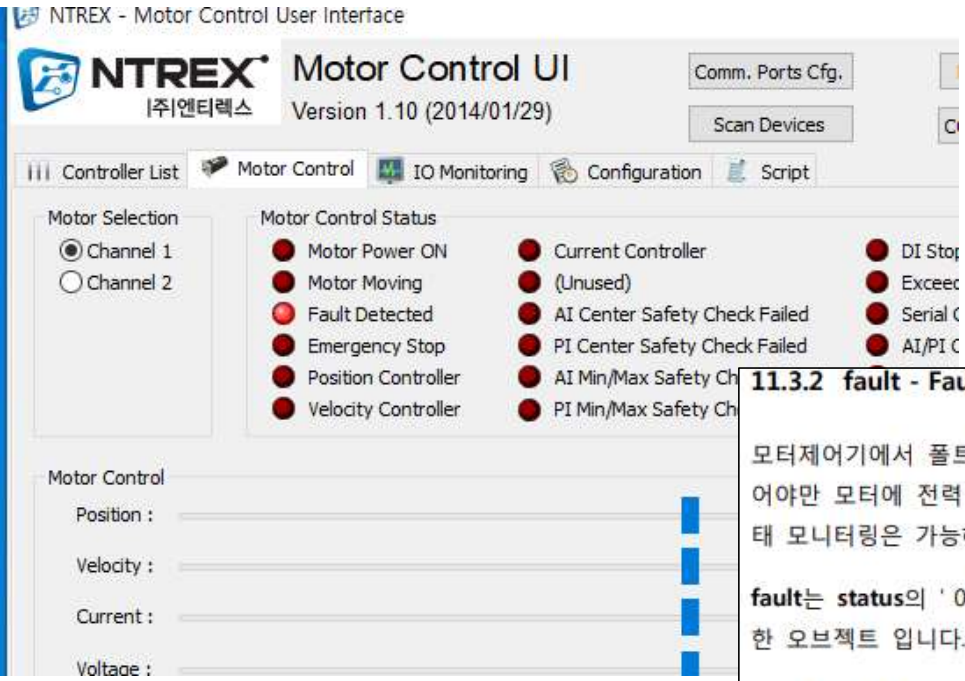
2.2.2퓨즈와 다이오드 삽입

퓨즈는 주로 과전압과 과전류로 인한 전원 배선 및 배터리의 손상을 보호합니다. 손상을 방지하기 위해 안전 조치로 ATO 또는 MAXI 시리즈보다 높은 전류의 퓨즈(Fuse) 사용을 권장합니다. 퓨즈를 선정할 때는 안전을 위해 제어기에 연결된 모터의 용량 합에서 80~90% 사이의 용량을 선택하는 것을 권장합니다.

참고로 40A 이상의 퓨즈는 일반적으로 끊어지는 속도가 느리기 때문에 제어기를 보호하는데 한계가 있습니다. 퓨즈가 끊어진 경우에도 배터리로의 반환 경로를 보장하기 위해서 고전류 다이오드를 퓨즈와 병렬로 연결해야 합니다.

※주의※
일반적으로 퓨즈를 전원에 연결하면 제품을 안전하게 보호할 수 있다고 생각합니다. 하지만 이는 잘못된 생각입니다. 퓨즈는 일반적으로 탈 때까지 약간의 시간이 걸리기 때문에 일시적으로 퓨즈를 통해 높은 전류가 흐를 수 있습니다. 이러한 경우, 모터(구동 제품)의 가속이나 동작 시 일시적으로 높은 전류를 끌어다 쓸 수 있는 장점도 있지만, 반대로 퓨즈가 제어기를 제대로 보호하지 못하기도 합니다.

문워커 모터제어기 Fault



Fault 발생으로 모터에 전원공급이 차단됨.

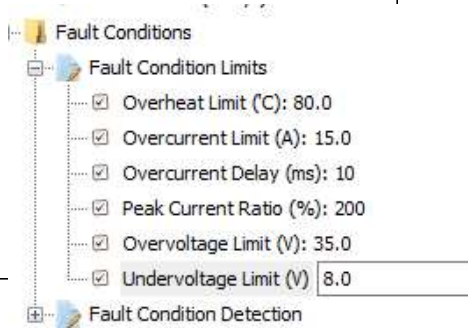
-배터리 전원 부족으로 인한 현상.

11.3.2 fault - Fault

모터제어기에서 폴트(Fault)가 발생하면 모터에 공급되는 전원은 차단됩니다. 폴트 상황이 해제되어야만 모터에 전력 공급이 가능합니다. 폴트 상태에서 모터제어기의 구성 파라미터 설정이나 상태 모니터링은 가능하지만 구동 명령은 실행되지 않습니다.

fault는 **status**의 ' 0x0004 - Fault Detected' 플래그가 켜졌을 때 폴트의 발생 원인을 파악하기 위한 오브젝트 입니다. 다음 목록을 참고하십시오:

- 0x0001 - Overcurrent
- 0x0002 - Overvoltage
- 0x0004 - Undervoltage
- 0x0008 - Overheat
- 0x0010 - Short Circuit
- 0x0020 - Stall Detection
- 0x0040 - Velocity Error Detection
- 0x0080 - Position Error Detection



14.2 C언어와의 차이

Mini-C 스크립트 언어는 C언어의 문법을 참조하여 설계되었습니다. 스크립트 언어는 C언어에서 다음과 같은 것들을 제공하지 않는 서브셋 언어 입니다:

- `#define, #ifdef, #if` 등의 전처리 문
- 함수의 정의와 선언 (`main` 함수 포함)
- `return` 키워드
- `int, float, long` 등 자료형 키워드
- 문자열 상수 (Ex: `"abc", "ABC"`)와 문자 상수 (Ex: `'a', 'A'`)
- 배열 연산자([]), 포인터 연산자(*), 주소 참조 연산자(&), 멤버 연산자(., ->)
- `sizeof` 연산자
- `? :` 연산자
- 캐스트 연산자 (Ex: `(int), (float)`)
- `typedef, struct, union, enum` 등 자료구조 관련 키워드
- `switch, case` 분기문 키워드

변수의 선언에 자료형은 사용되지 않으며, 변수는 정수형이나 실수형으로 초기화 하면서 선언됩니다. 다음 변수 선언 예를 참조하기 바랍니다:

- `a=123` - 정수형 변수 a를 선언하면서 값 할당
- `b=4.567` - 실수형 변수 b를 선언하면서 값 할당

C언어에서 사용되는 다음 분기문과 반복문을 사용 가능합니다:

- `if, else` - 조건에 따라 정해진 영역의 프로그램을 실행
- `goto, label` - 특정 프로그램 코드 위치로 무조건 점프
- `for` - 특정 조건이 완료 될 때까지 특정 영역을 반복 실행
- `while` - 특정 조건이 만족할 동안 특정 영역을 반복 실행
- `do, while` - 먼저 특정 영역을 실행하고 조건이 맞으면 이를 반복
- `break` - 현재 실행하고 있는 반복문 블록을 빠져 나옴
- `continue` - 현재 실행하고 있는 반복문 블록의 초기로 이동

C언어에서 사용되는 다음 수학 연산자는 사용 가능합니다:

- 산술 연산자: `+, -, *, /, %, ++, --`
- 관계 연산자: `==, !=, >, <, >=, <=`
- 논리 연산자: `!, &&, ||`
- 비트 연산자: `~, &, |, ^, <<, >>`
- 대입 연산자: `=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=`

Mini-C 스크립트 언어에서 사용하는 키워드는 다음과 같습니다:

- `if, else, do, for, while, goto, break, continue`

Mini-C 스크립트 언어에서는 다음 상수가 정의되어 있습니다:

- `_E, _PI, _EULER, _status, _s, _fault, _f`

l4.9.4 getv

Declaration: getv (index, sub_index)

인수 index와 sub_index로 지정된 제어기의 오브젝트 값을 읽어옵니다. `getv()` 함수로 제어기의 모든 오브젝트 값을 읽을 수 있습니다.

ndex는 제어기 오브젝트명의 참조 값이고 sub_index는 I/O나 모터의 채널 번호입니다. index는 JSB/RS-232 통신에서 사용하는 Short Name과 Long Name으로 대체할 수 있으며, 스크립트에서 사용하기 위해서는 Short Name과 Long Name 앞에 언더스코어 문자('_')를 붙여야 합니다.

Remarks: 유효하지 않은 index와 sub_index에 대해 읽기를 시도하는 경우, `getv()` 함수는 아무런 경고 없이 0을 돌려줍니다. 읽은 값이 0일 때, 실제 오브젝트의 값이 0인지 혹은 유효하지 않은 오브젝트를 엑세스 하였는지 알 수 없기 때문에 사용자는 `getv()` 함수로 넘기는 index와 sub_index가 유효한지 미리 판단하여야 합니다.

사용 가능한 index와 Short Name, Long Name에 대해서는 "**14.5.3 미리 정의된 상수**"를 참조하기 바랍니다.

Examples:

```
// 모터 채널 1의 전압 명령 값을 읽어옴, 다음 3개의 함수는 동일한 기능을 수행함
a = getv (114, 1);
b = getv (_vtc, 1);
c = getv (_voltage_command, 1);
```

7월_테스트베드
코드실험

```
/*
3ms속도=-((sec*sec*sec*3)/4)+((sec*sec*9)/4);
2ms속도=-((sec*sec*sec)*4)+((sec*sec*6));
1ms속도=-(16*sec*sec*sec)+(12*sec*sec);
-1ms속도=-(-(16*sec*sec*sec)+(12*sec*sec));
*/


max_time_3ms=2;
max_time_2ms=1;
max_time_1ms=0.5;
scale_3ms=0.01;
scale_2ms=0.02;
scale_1ms=0.01;

delay_3ms=19.3;
delay_2ms=10.2;
delay_1ms=9;

r = getv (_wheel_radius, 0);                    // 바퀴 반지름
// b = getv (_axle_length, 0);
g = getv (_gear_ratio, 0);                      // 감속비

enc = getv (_encoder_ppr, 2);              // 엔코더 해상도
enc2_first = getv (_position, 2); //초기 엔코더 값

pulse2met = 2*_PI*r*100/enc;                    // 엔코더 펄스를 meter로 바꾸는 scale factor
sec=0;
met2rpm=60/(_PI*2*r);
while (1) {

//1ms속도-----------------------------------------------
    sleep(1000);
    start_t1=clock();
    VEL=1111;
    while (sec<max_time_1ms){
```

목표값	거리	시간	최대rpm	속도
1M/s	0.5m	0.5+0.5=1s	335	1
2M/s	2m	1+1=2s	670	2

▶가+감속거리(cm단위): enc_meter
▶가+감속시간(ms) : end_t

1m/s

#	variable	Value
13	enc2_first	7432
14	pulse2met	0.0535919
15	sec	-0.00999998
16	met2rpm	329.286
17	start_t1	843259
18	VEL	1111
19	cycle_start...	844419
20	v2	2.66454e-015
21	Rpm2	8.77394e-013
22	enc2	7432
23	cycle_end_...	844421
24	cycle_time	2
25	target_delay	8
26	end_t1	1172
27	de2	1108
28	enc_meter	59.3798
29	start_t2	0
30	end_t2	0

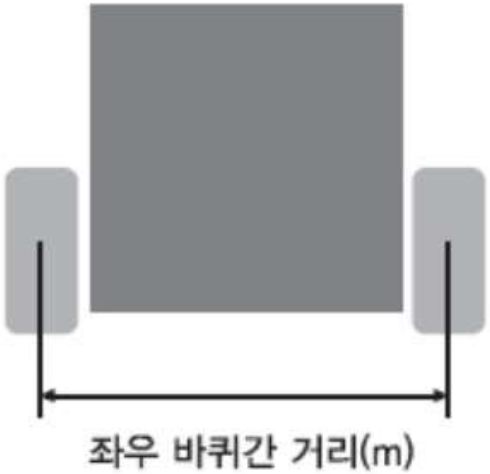
2m/s

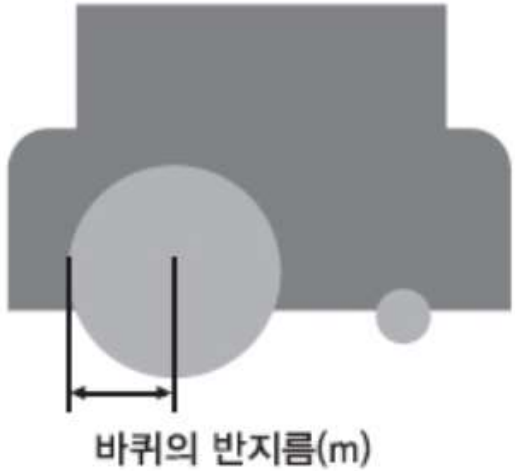
#	variable	Value
13	enc2_first	12655
14	pulse2met	0.0535919
15	sec	-0.00999998
16	met2rpm	329.286
17	start_t1	1.00465e+...
18	VEL	2222
19	cycle_start...	1.00901e+...
20	v2	0.000596003
21	Rpm2	0.196255
22	enc2	12655
23	cycle_end_...	1.00902e+...
24	cycle_time	2
25	target_delay	18
26	end_t1	1180
27	de2	4115
28	enc_meter	220.531
29	start_t2	1.00683e+...
30	end_t2	2202

엔코더모터 이용한 정밀한 위치 및 속도제어

2축 구동형 모바일 로봇에 특화된 명령 및 환경 지원

- 고급형 모바일 플랫폼의 주 제어기로 바로 사용 가능
- 2축 구동형 모바일 로봇의 주행 명령 탑재
- 2축 구동형 모바일 로봇을 Joystick과 RC 조종기를 이용한 직접 구동 지원
- Joystick과 RC 조종기를 사용해서 전류 / 속도 제어 설정 가능





2축 구동형 모바일 로봇에 특화된 명령 및 환경 지원

- Digital in/out port 및 Analog / Pulse input port 지원
- 내부 변수 및 구동 명령과 외부 IO를 사용할 수 있는 스크립트 (script) 지원
- 사용자 스크립트 사용 시 내부 변수의 모니터링 가능
- 제품의 설정과 구동을 편하게 할 수 있는 환경 설정 및 구동용 PC 프로그램 제공
- EEPROM을 이용한 사용자 세팅 값 저장 및 재부팅 시 자동 불러오기
- CAN 통신에서 멀티드롭(Multidrop) 연결을 위한 Device ID 최대 255개까지 설정 가능
- 배터리 전압 측정 및 제어기의 과전압 / 저전압 보호 기능
- 모터의 전류 측정 및 과전류 보호 기능
- 통신 연결 중단 시 모터 정지를 위한 Watchdog Timer 기능
- 내부 FET 방열판의 온도 측정을 통해 과열 보호 기능 탑재
- RGB LED를 사용한 에러, 동작, 통신상태 표시 기능

MW-MDC24D200S/200D 특징

- Unipolar/Bipolar PWM 스위칭 방법 설정 및 18kHz에서 40kHz까지 PWM 주파수 설정 가능
- CAN, RS-232 연결 지원 (CAN 통신 속도: 10K ~ 1M bps, RS-232 통신 속도: 9600 ~ 921600bps)
- CAN, RS-422, RS-485에서 멀티드롭 연결을 위한 Device ID(1~255) 설정 가능
- 시리얼 통신(CAN, RS-232) 연결 중단 시 모터 정지를 위한 Watchdog timer 기능 지원
- 시리얼 통신과 Analog Input, Pulse Input 명령어 동시 사용 가능
- Script 작성과 컴파일, 제어기로 다운로드 및 실행, PC에서 시뮬레이션 실행
- Anti-windup이 적용된 PID 위치제어기
- 위치 제어 시 가속도와 감속도가 고려된 사다리꼴 형태의 속도 프로파일 생성
- Anti-windup이 적용된 PI 속도제어기 / 전류제어기
- Incremental Encoder 피드백으로 정밀한 위치제어 및 속도제어
- Analog/Pulse input에 연결된 속도센서(Tachometer) 피드백으로 폐회로(Closed loop) 속도 제어
- Analog/Pulse input에 연결된 위치센서(Potentiometer) 피드백으로 폐회로 위치제어
- PWM ratio의 직접 출력으로 개회로(Open loop) 속도 설정
- 사다리꼴 프로파일을 적용한 모터의 속도제어 및 PWM ratio 출력 설정 (프로파일의 가속도와 감속도를 각각 지정)
- 베터리 전압 측정으로 제어기의 과전압, 저전압 보호기능
- 모터의 전류 측정으로 모터의 과전류 보호기능
- FET 방열판의 온도 측정으로 제어기 과열 보호기능
- 모터 특성 설정에 따른 출력 제한 (정격 전압 제한, 최고 전류 제한, 최고 속도 제한)
- Min/Max 위치 범위 설정과 소프트웨어 리미트 기능

PC조종을 이용한 MoonWalker 테스트 http://www.ntrexgo.com/archives/15714

5.1 엔코더 커넥터 연결

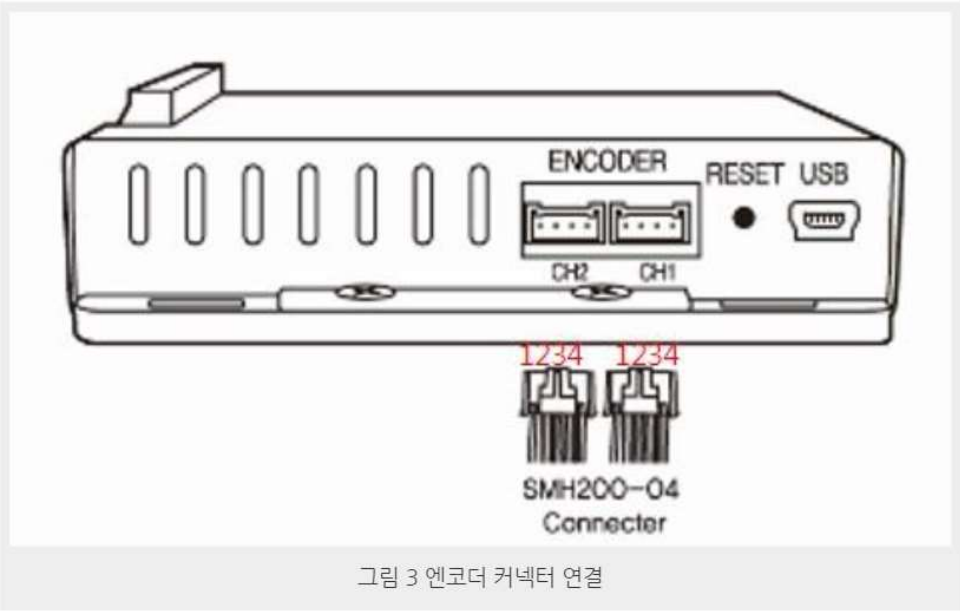
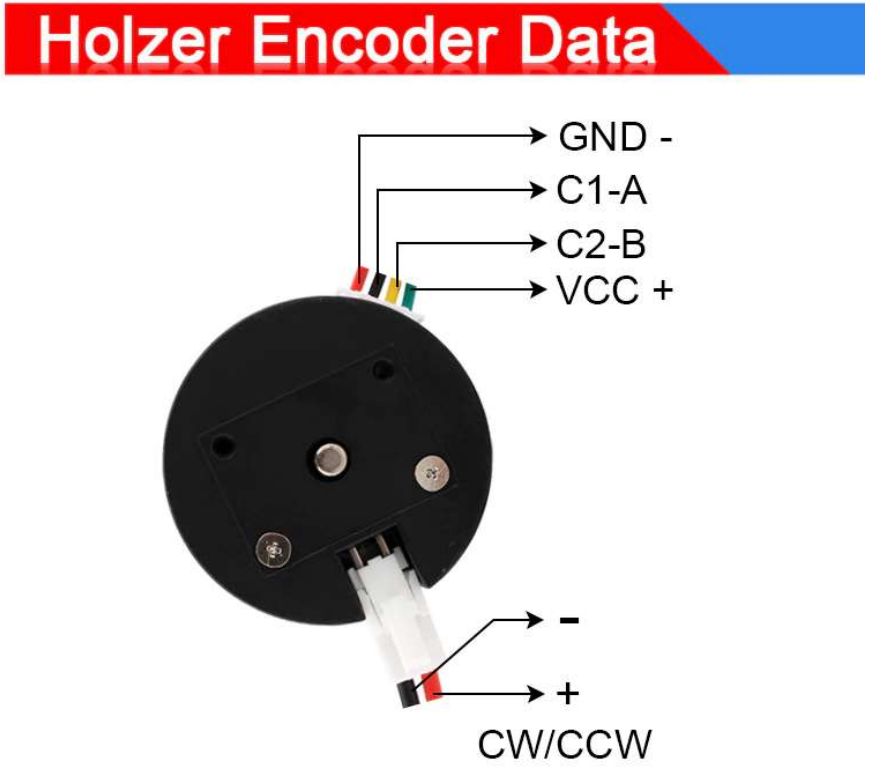


그림 3 엔코더 커넥터 연결

	Motor 2 Encoder	Motor 1 Encoder
1	Motor 2 VCC	Motor 1 VCC
2	Motor 2 Encoder A	Motor 1 Encoder A
3	Motor 2 Encoder B	Motor 1 Encoder B
4	Motor 2 GND	Motor 1 GND



Fault Detected

실험할 때 AGV주행하다 턱에 걸리거나 어딘가에 충격을 받았을시 Fault Detected 불 들어오고 정지함.

16.8.10 Motors – Fault Conditions

Fault Condition은 사용자 그리고 제품의 안전을 위해 Fault 감지 조건을 설정할 수 있습니다. Fault Condition Limits는 과열온도, 과전류, 과전류 흐르는 시간, 제어기의 FET가 흘릴 수 있는 순간 최대 전류범위, 과전압, 저전압 한계값을 설정할 수 있고, Fault Condition Detection은 모터의 Fault 감지 조건을 설정할 수 있습니다.

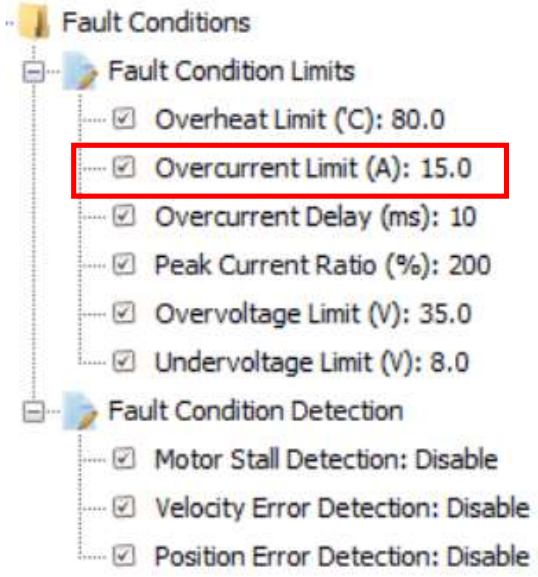


그림 16-20 Fault Condition

급 충돌 시 과전류가 되어 그랬던 거임. 원래 4A로 돼있었는데 15A로 바꾸니 괜찮아 짐.