G.INS

MW-AHRS

User's Manual

매뉴얼 버전 V1.01

머리말

G.INS의 G는 관성항법장치에서 중요한 요소인 Gravity를 의미하기도 하며, 사용자에게 좀 더 나은 결과값을 제시하기 위해 Genius한 제품을 만들기 위한 저희의 의지를 의미하기도 합니다. 또한 G.INS의 INS는 관성항법장치(Inertial Navigation System)를 의미하는데, 관성항법장치는 엔코더와 같이 고정점이 있어야만 상대각도를 측정할 수 있는 센서를 사용할 수 없는 경우 자신의 자세를 측정하기 위한 필수 장비입니다.

이번에 저희 ㈜엔티렉스는 G.INS 제품군의 두번째 제품으로 MW-AHRS를 출시하게 되었습니다. MW-AHRS는 3축 가속도 센서와 3축 자이로 센서, 3축 자기 센서, 32bit ARM Cortex-M3 마이크로프로세서를 탑재하여 센서의 회전된 자세각을 측정하는 AHRS 모듈입니다.

AHRS(Attitude Heading Reference System)는 관성항법장치들 중에서 진행방향(x)과 측면방향(y), 수직방향(z) 축을 중심으로 회전하는 각도(roll, pitch, yaw)를 측정하는 시스템을 의미합니다. 저희 MW-AHRS는 3축 자이로 센서와 3축 가속도 센서, 3축 자기 센서 데이터를 칼만필터로 융합하여 **roll과 pitch, yaw 각도를 측정**하고 사용자에게 RS-232나 CAN통신으로 Degree 단위로 출력해 주도록 설계되어 있습니다.

MW-AHRS는 내부의 가속도 센서나 자이로 센서의 특성에 따른 오차를 줄이기 위해 출하전 모든 캘리브레이션을 합니다. 또한 캘리브레이션 명령을 내장하고 있기 때문에 사용자가 필요에 따라 캘리브레이션을 할 수 있습니다.

MW-AHRS는 다음과 같은 곳에 활용될 수 있습니다:

- 소형 항공기 및 차량의 자세측정
- 로봇의 몸체나 핸드의 절대각도 측정
- 세그웨이와 같은 밸런싱로봇의 기울어짐 측정

목차

1 G.INS MW-AHRS

1.1 측정 개요

AHRS(Attitude Heading Reference System)는 진행방향(x), 측면방향(y), 수직방향(z) 축을 중심으로 회전하는 각도(roll, pitch, yaw)를 측정하는 관성항법장치 시스템 입니다. 저희 NTREX의 MW-AHRS 센서는 3축 자이로 센서, 3축 가속도 센서, 그리고 3축 자기 센서로부터의 출력 데이터를 칼만필터로 융합하여 **roll과 pitch, yaw 각도를 측정**합니다. 칼만필터는 자이로센서에서 측정한 각속도를 적분하여 각도를 **추정**하고 가속도 센서에서 **측정**한 중력가속도의 방향으로 roll, pitch 각도를 **보정**합니다. 또한 자기 센서에서 측정한 지자기의 방향으로 yaw 각도를 보정합니다.

MW-AHRS는 다음 그림 1-1에서와 같이 **오른손좌표계(Right-handed Cartesian Coordiantes)를 사용**합니다. 오른손좌표계는 xy평면에서 +z축의 방향이 위쪽(하늘, 천장)을 향하고 있습니다. 오른손좌표계는 로봇이나 차량, 핸드 등의 몸체 좌표계를 정의하기 위해 주로 사용되며, 좌표계의 각 축의 방향은 다음과 같습니다: 몸체의 전진 방향을 +x축으로 보고 몸체의 왼쪽 방향을 +y축으로 봅니다. 또한 +z축은 몸체의 위쪽 방향이 됩니다. MW-AHRS를 로봇이나 차량 등에 설치할 때도 좌표계의 방향을 일치시켜 설치하는 것이 위치 및 각도의 연산에 용이합니다.

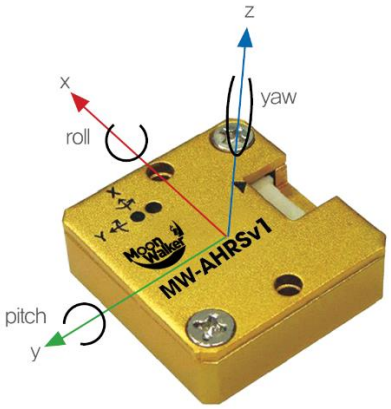


그림 1-1 MW-AHRS 센서의 좌표계와 오일러각

Roll-pitch-yaw는 주로 항공분야에서 비행기의 회전을 기술할 때 많이 사용됩니다. MW-AHRS에 사용된 roll-pitch-yaw는 상기 그림 1-1과 같습니다. 몸체의 로컬 좌표계를 기반으로 **x축으로의 회전을 roll, y축으로의 회전을 pitch, z축으로의 회전을 yaw로 정의**하고 회전 방향은 각 축에 대하여 반시계 방향입니다(각 축의 +에서 내려다 볼 때).

오일러각(Euler Angle) 회전변환은 x, y, z축 중 하나의 축을 선택하여 회전 하는 과정을 3번 반복하는데, 회전 축의 조합이 다양하게 만들어질 수 있습니다. 주로 x-y-z축, z-y-x축 혹은 z-y-z축 순서로 회전하는 것이 일반적입니다. MW-AHRS는 오일러각으로 회전변환 할 때, **z-y-x축 순서로 회전**합니다. 오일러각 회전변환은 회전 순서에 따라 결과가 달라지기때문에 주의해서 사용해야 합니다.

1.2 구성품

그림 1-2 MW-AHRS 구성품

구성품	수량
MW-AHRS	1 EA
케이블(30Cm)	1 EA

MW-AHRS에 사용된 커넥터와 터미널은 12507HS-06L 이며, 터미널은 12507TS 입니다.

- 케이블 판매페이지: http://www.devicemart.co.kr/29865

구매하신 재품에는 MW-AHRS 센서의 구동 소프트웨어나 예제 프로그램, 사용자 메뉴얼이 동봉되어 있지 않습니다. 센서의 구동 및 통신에 필요한 유틸리티 및 예제 코드는 다음 AHRS v2 홈페이지에서 다운로드 가능합니다.

- MW-AHRSv1 판매페이지: www.devicemart.co.kr/1310790

1.3 특징 및 사양

MW-AHRS는 32비트 ARM Cortex-M3 마이크로프로세서를 탑재하고 있으며, 3축 가속도센서와 3축 자이로센서, 3축 자기센서, 온도센서의 데이터를 사용하여 6개의 위치와 자세정보(x, y, z, roll, pitch, yaw) 중 **roll과 pitch, yaw 각을 구하는 AHRS(Attitude Heading Reference System) 모듈**입니다. 이 센서를 사용하여 로봇 몸체나 핸드의 자세를 측정가능합니다.

MW-AHRS는 RS-232 통신과 CAN 통신이 지원됩니다. 범용으로 쓰이는 RS-232 통신을 사용하여 측정 데이터를 읽어올 수 있고, CAN(Control Area Network) 버스를 사용함으로 여러 개의 센서로부터 측정 데이터를 읽어올 수 있습니다.

각 센서마다 고유한 오차를 줄이기 위해 생산 단계에서 캘리브레이션을 하여 출고합니다. 따라서, 처음 전원 인가 시 별도의 캘리브레이션 없이 바로 사용가능합니다. 하지만 사용자가 AHRS UI를 사용하여 직접 캘리브레이션 가능합니다. 캘리브레이션에 관한 자세한 서술은 아래의 5.5장을 참조하시면 됩니다.

센서의 외형은 두께 1.6mm의 알루미늄 케이스 타입으로 견고하고 소형화하여 제작하여, 로봇이나 무인반송차, 각종 산업용 장비 등에 쉽게 적용하고 사용가능합니다.

1.3.1 General Specifications

- 두께 1.6mm의 알루미늄 케이스
- 칼만필터를 이용한 가속도와 자이로 센서 융합
- 0° 유지 에러: < 0.2°
- Dynamic Error: < 2°
- Euler angle's Resolution: 0.01°
- Response Time: < 1ms
- 크기(L, W, H): 29.6mm, 31.4mm, 10mm
- 무게: 약 20g

1.3.2 Electric Specifications

Parameter	Symbol	Max Range	Recommend	Unit
Supply Voltage	V_{DD}	4.5 ~ 10	5 ~ 8	V
Supply Current	I_{DD}	50		mA
Power		330		mW
Operating Temperature	T	-10 ~ +80	0 ~ 45	℃
Gyroscope Range		±2000		°/s
Accelerometer Range	g_{FS}	±16		g

Magnetometer Range		±4800	μT
Booting Time		50	ms
Initializing Time	T_B	500	ms

MW-AHRS의 허용전압범위는 4.5~10V 이지만, 내부 레귤레이터의 발열을 감안하여 8V이내 사용을 권장합니다. 전압을 인가하고나서 명령어 수행이 가능한 상태까지의 부팅 시간은 50ms 이내입니다. 또한, 안정된 데이터는 부팅후 대략 500ms 이후부터 수신 가능합니다. 이는 자이로센서, 가속도센서 등 각종 IC 들에 워밍업 시간이 필요하기 때문입니다.

1.3.3 Interfaces

- RS-232 Interface:
  - 비트/초: 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bps (Default: 115200)
  - 데이터 비트: 8bits (Fixed)
  - 패리티: None (Fixed)
  - 정지 비트: 1bit (Fixed)
  - 흐름제어: None (Fixed)
- CAN (CAN2.0B – Extended CAN) Interface:
  - 비트/초: 1000K, 800K, 500K, 250K, 125K, 50K, 25K, 10K bps (Default: 1000K)
  - Device ID: 0 ~ 255 (Default: 1)
- 가속도, 각속도, 각도, 자기, 온도 데이터 출력
- 최대 1kHz로 동기화 데이터(가속도, 각속도, 각도, 자기) 출력
- RS-232 기반의 모니터링 및 설정 유틸리티(AHRS UI) 제공

1.3.4 Software functions

- RS-232 Baudrate 설정
- CAN Bit rate 설정
- 장치 ID 설정
- 캘리브레이션 명령을 통해 0° 캘리브레이션
- 공장 출하시 초기값 읽기
- 소프트웨어 리셋

1.4 구조

1.4.1 알고리즘 구조

MW-AHRS의 소프트웨어는 자이로 센서, 가속도 센서와 자기 센서로부터 1kHz 주기로 각속도, 가속도, 그리고 자기 값을 읽어옵니다. 이 데이터로부터 센서의 온도 변화에 따른 바이어스를 보정하고 다시 센서의 측정 바이어스를 보정하고 스케일을 조정합니다. 또한 Dynamic Drift Rection 알고리즘, Dynamic Scale Factor Adjustment 알고리즘이 동작하면서 각속도와 가속도 값에 동적으로 발생하는 드리프트와 스케일 펙터를 조정합니다.

칼만 필터는 1kHz로 동작하면서 가속도와 각속도, 자기 값을 융합하여 회전행렬 (Rotation Matrix)을 지속적으로 업데이트 합니다. 이후 8Hz 주기로 회전행렬의 직교성을 유지하는 알고리즘이 수행됩니다. 그리고 CAN이나 RS-232 포트를 통해 데이터가 요청될 때마다 회전행렬은 오일러각으로 변환되어 전송됩니다.

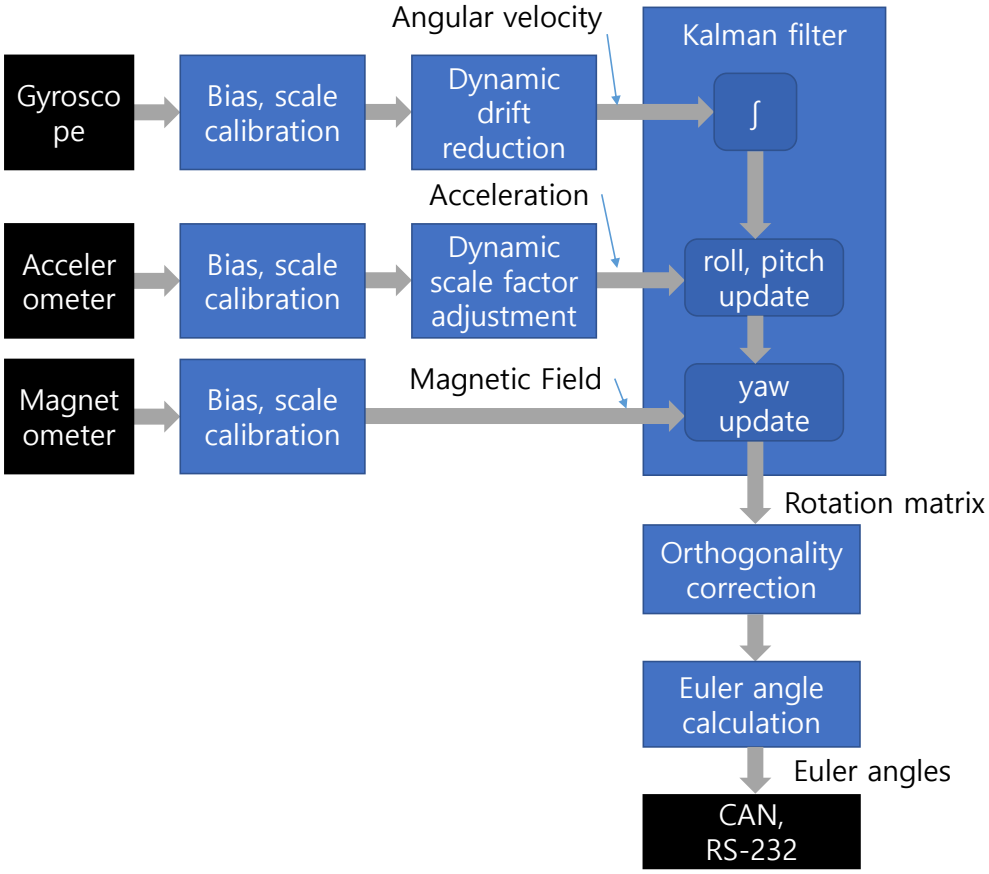


그림 1-3 MW-AHRS의 소프트웨어 블록도

1.4.2 하드웨어 구성

MW-AHRS는 32비트 ARM Cortex-M3 마이크로프로세서를 탑재하고 있으며, InvenSense사의 MPU-9255 센서로부터 3축 가속도 데이터, 3축 자이로 데이터, 3축 자기 데이터와 온도 데이터를 받아옵니다. 그리고 PC나 마이크로컨트롤러와 연결하기 위한 RS-232와 CAN 통신 트랜시버를 내장하고 있습니다.

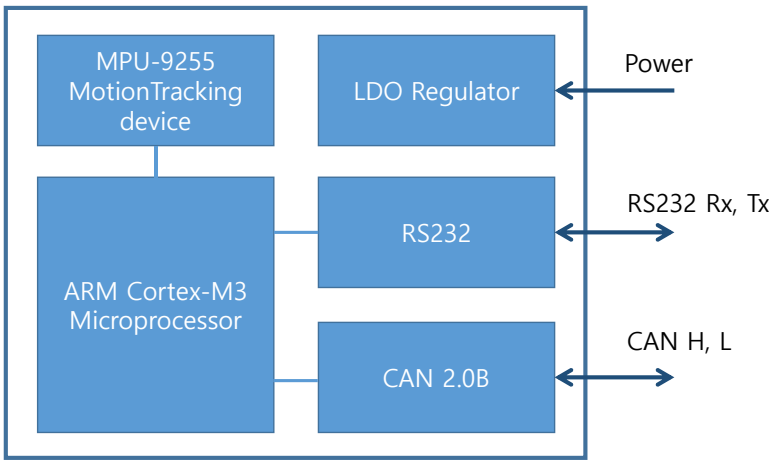
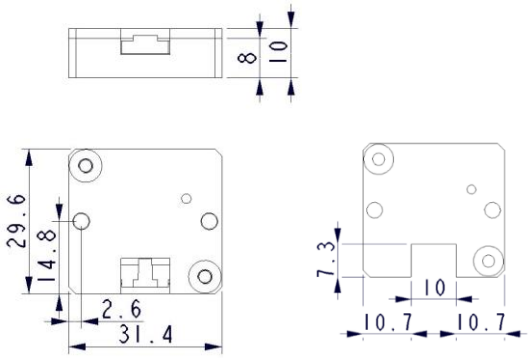


그림 1-4 MW-AHRS의 하드웨어 기능 블록도

1.5 기구부

MW-AHRS는 1.6T의 알루미늄 케이스 타입으로 소형화하여 제작하였기 때문에 가볍고(무게: 20g) 크기도 작으며(L, W, H: 29.6, 31.4, 10mm) 충격에 강합니다.

1.5.1 도면 (단위: mm)



1.5.2 설치방법

MW-AHRS의 케이스에 뚫려 있는 ϕ3.2mm의 홀을 이용하여 다른 기구에 부착 할 수 있습니다. 측정대상체의 회전중심이나 무게중심에 설치하길 권장합니다. 만약 회전중심에 설치하지 않았을 경우, 구심가속도의 영향을 받아 각도가 한쪽으로 편향되어 추출되는 경우가 발생할 수 있습니다.

또한, 진동이 심한 환경에서는 방진패드 등을 사용하여 센서로 전달되는 진동을 최소화 하여야 합니다. 그렇지 않은 경우, 센서의 측정데이터의 신뢰성이 진동으로 인한 오류로 인해 낮아집니다.

※주의※ MW-AHRS는 각속도를 적분하여 각도를 계산하고 중력가속도의 방향으로 roll, pitch 각을 보정하기 때문에, 장착하실 때 반드시 평평한면 그리고 가능하면 몸체의 중심부에 장착해 주시기 바랍니다. 또한 지자기의 방향으로 yaw 각을 보정하기 때문에, 모터나 자석 가까이 설치하면 안됩니다.

1.6 주의사항

- 전원이 인가된 상태에서 배선 또는 점검을 하지 마십시오. 고장의 주요 원인이 됩니다.
- 제품을 분해 또는 개조 하지 마십시오. 파손의 위험이 있습니다.
- 전원을 연결하기 전에 반드시 커넥터의 +, - 극성을 확인하기 바랍니다.
- 가속도가 16g 이상의 환경에서는 정상작동이 어렵습니다.
- 각속도가 2000°/s 이상의 환경에서는 정상작동이 어렵습니다.
- 모터나 자석, 철 구조물 등에 의해 지구 자기장이 왜곡되는 경우 정상작동이 어렵습니다.

2 인터페이스

2.1 LED

MW-AHRS 세서는 한 개의 붉은 색 LED가 표시되니다. LED는 센서의 동작 상태를 알리기 위해 사용됩니다. 센서가 부팅되면서 가속도와 자이로 센서가 초기화되고 안정될 때까지는 LED는 약 500ms정도간 켜진 상태로 유지됩니다.

센서 초기화가 완료되고 정상적으로 동작 중이라면, LED는 1초 주기로 깜박이게 됩니다. 만일 LED가 계속 켜진 상태나 꺼진 상태로 있다면 센서가 올바르게 동작하지 않는것입니다. 이런 경우에는 본사에 A/S를 요청부탁드립니다.

사용자가 캘리브레이션 명령을 내리면, 캘리브레이션이 진행중임을 알리기 위해 LED는 빠른 주기(200ms)로 깜박입니다. 그 후 2초간 켜진 상태를 유지합니다:

- 200ms 주기로 깜박임: 자기 센서와 가속도 센서의 스케일과 바이어스를 보정 중일 때
- 2초간 켜진 상태 유지: 자이로 센서의 스케일을 보정하였을 때

가속도 센서와 자이로 센서의 캘리브레이션이 한 번 완료되는데는 2초 정도가 소요되며, 성공하였을 때는 LED가 2초간 켜진 상태를 유지합니다. 만약 실패하였다면 2초간 켜진 상태없이 바로 동작 모드로 복귀합니다.

자기 센서 캘리브레이션은 60초 정도가 소요되며, 성공하였을 때는 2초간 켜진 상태를 유지하고 실패하였을 때는 바로 동작 모드로 복귀합니다.

2.2 커넥터

MW-AHRS는 12507HS-06L (6핀 커넥터)를 사용하고 있습니다. 각 핀에 대한 설명은 아래 그림 2-1을 참조 하기 바랍니다. (케이스 상단에 1번핀 표시가 되어있습니다.)

커넥터에는 전원과 CAN, RS-232 통신 핀을 제공하고 있습니다. 여기서 GND 핀은 전원과 CAN, RS-232 통신에서 공통으로 사용됩니다.

Pin #	Description
1	VCC (5~8V)
2	TX
3	RX
4	CAN_H
5	CAN_L
6	GND

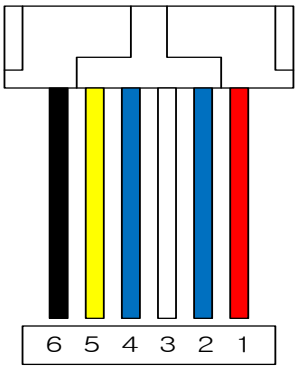


그림 2-1 커넥터 핀맵

2.3 RS-232 연결

RS-232 연결은 통신이 지원되는 PC 혹은 다른 마이크로 컨트롤와 MW-AHRS를 연결할 때 사용됩니다. MW-AHRS는 자체에 RS-232 통신을 할 수 있도록 SP3232 칩이 내장되어 있습니다. 그러므로 별도의 드라이버 없이 RS-232 통신이 가능합니다. 만약 마이크로컨트롤러를 통해 MW-AHRS의 데이터를 처리하고자 한다면, 반드시 MAX232나, SP3232와 같은 기능을 하는 칩을 거쳐서 통신 하기 바랍니다.

2.3.1 연결 구성

RS-232 연결 케이블은 DB9 암 커넥터와 3선 케이블을 사용하여 직접 제작하여 사용하실 수 있습니다.

PC의 RS-232 포트는 보통 DB9 수(male) 커넥터로 구성됩니다. 그렇기때문에 센서로부터 연결되는 신호 선들은 DB9 암(female) 커넥터에 결선 되어야합니다. 다음 그림을 참조하여 센서의 RX, TX, GND 핀을 DB9 암 커넥터의 핀으로 연결합니다.

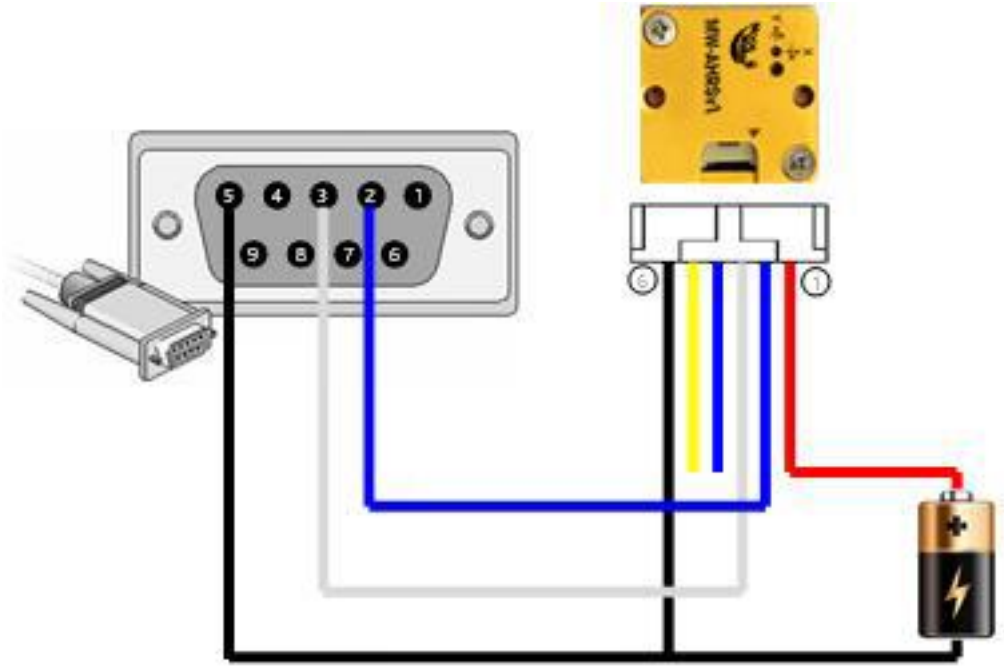


그림 2-2 센서와 DB9 Female 커넥터 연결

여기서 커넥터의 핀 번호를 혼동하지 않아야 합니다. 상기 그림의 핀 번호는 전면에서 커넥터를 보는 것을 기준으로 합니다. 대부분의 커넥터 브랜드는 플라스틱 성형 부분에 핀 번호가 적혀 있습니다.

2.3.2 연결 설정

MW-AHRS는 하이퍼터미널과 같은 프로그램을 통해 사용할 수 있습니다. 통신 속도는 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bps 중 하나를 선택할 수 있지만, 데이터 비트, 패리티, 정지 비트, 흐름제어는 다음 표에서와 같이 고정되어 있습니다.

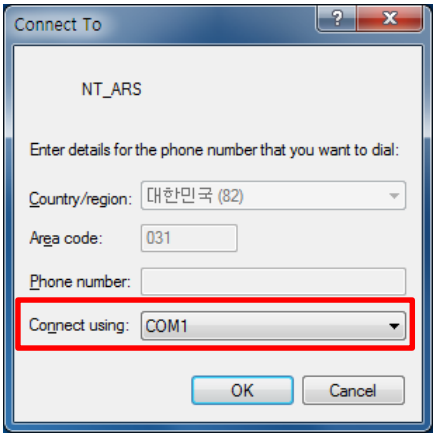
설정 항목	설정 값
Baudrate	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bps
Data Bits	8 bits
Parity	None
Stop Bit	1bit
Flux Control	None

※ 제품 출하 시 기본 통신속도는 115200bps로 설정되어 있습니다.

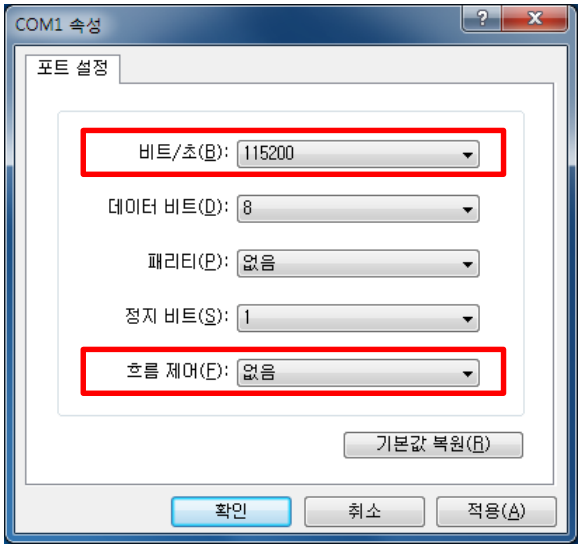
2.3.3 PC 에서 연결 예

다음은 하이퍼터미널을 사용하여 MW-AHRS에 연결하는 과정입니다.

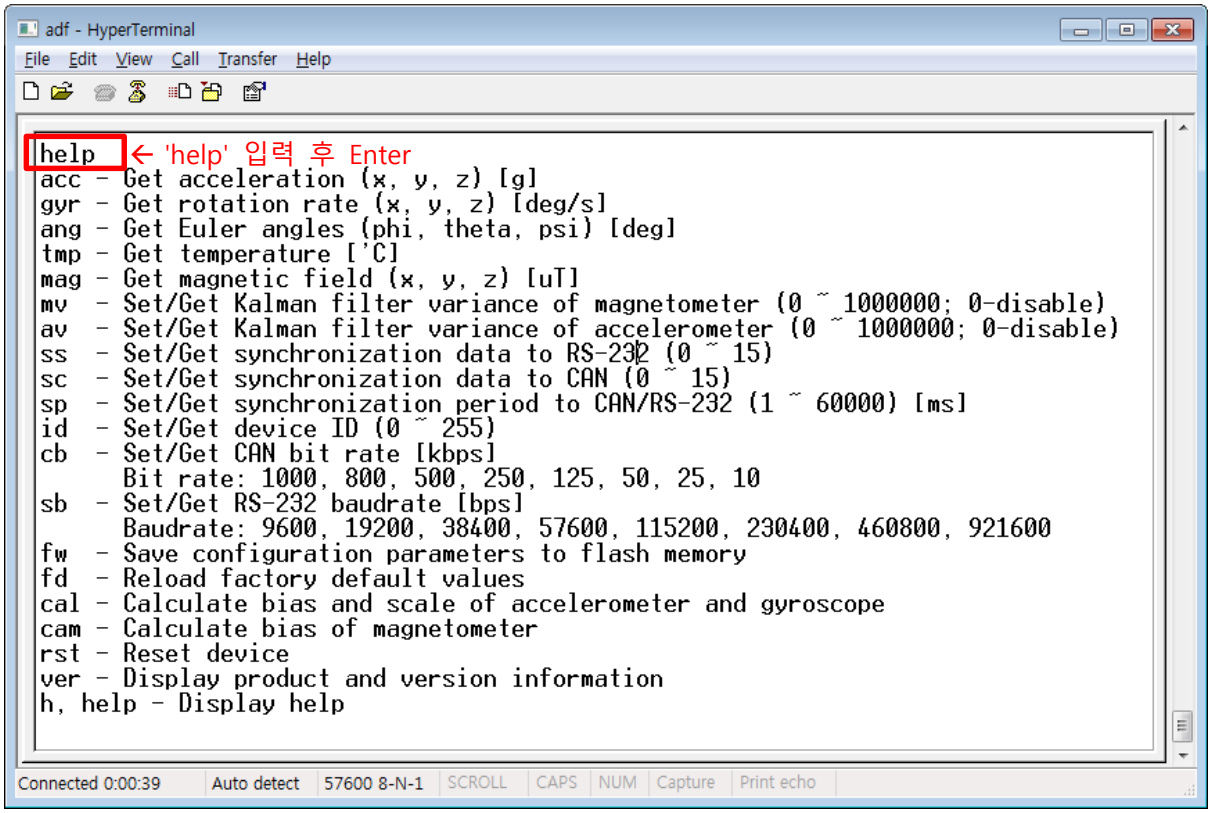
하이퍼터미널을 실행하고 상단 툴바의 전화기 모양 아이콘(Call)을 클릭합니다. 그러면 New Connection 대화상자가 실행되고, 여기서 Name 항목에 'NT_AHRS'를 입력합니다. 그리고 [OK] 버튼을 클릭하면 다음 그림과 같이 Connect To 대화상자가 실행됩니다. 여기서 MW-AHRS가 연결된 통신 포트를 선택합니다.



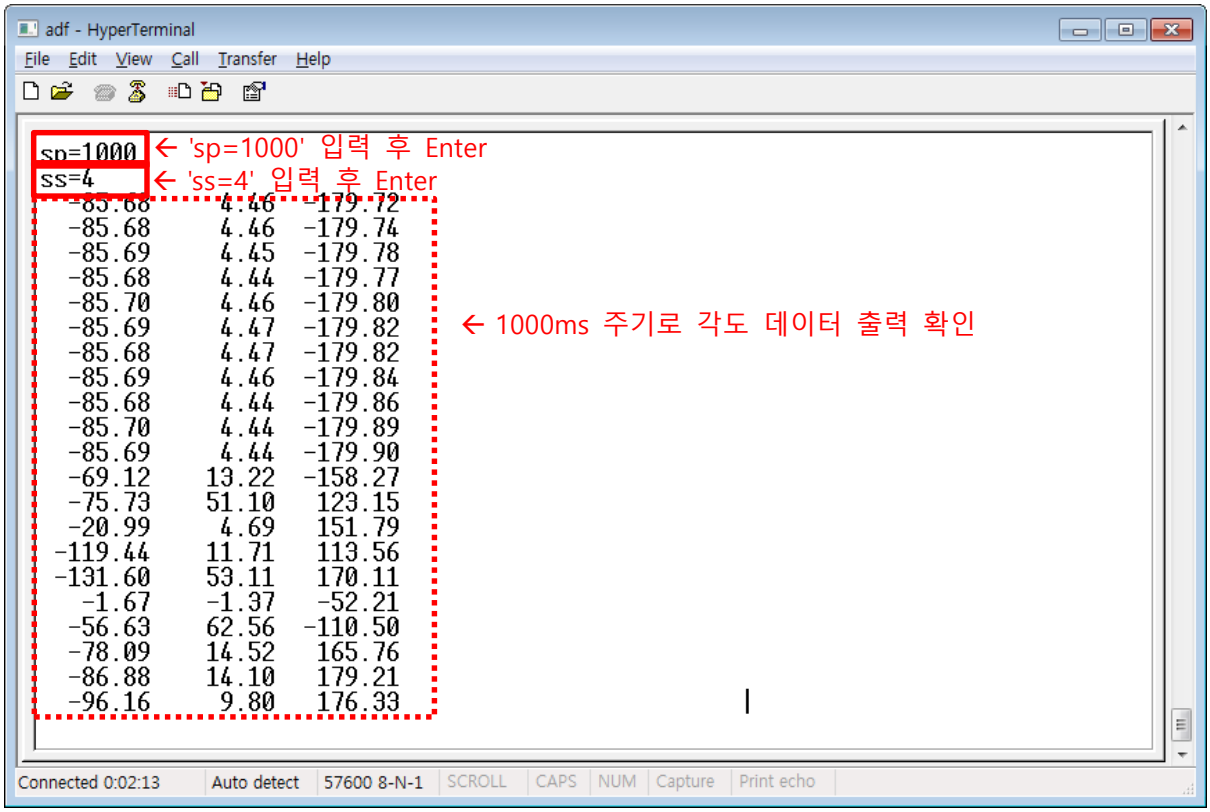
[OK] 버튼을 클릭하면 COM 포트에 대한 속성 설정 대화상자가 실행됩니다. 여기서 '비트/초' 항목을 '115200'(통신 속도는 사용자가 설정한 값에 따라 달라질 수 있음)로 설정합니다. 그리고 '흐름제어' 항목을 '없음'으로 선택합니다. 그리고 [확인] 버튼을 클릭합니다.



속성 설정 창이 닫힌 후, 하이퍼터미널은 MW-AHRS에 연결됩니다. 연결을 확인하기 위해 MW-AHRS의 전원을 껏다 켜고 'help'를 입력한 후 키보드의 Enter 키를 누릅니다. 그러면 다음과 같이 도움말이 하이퍼터미널 창에 표시됩니다.



RS-232 포트로 주기적인 데이터 전송을 위해 다음과 같이 입력합니다. 이후 1초 주기로 각도(roll, pitch, yaw) 데이터를 반복적으로 출력합니다.



2.4 CAN 연결

CAN(Control Area Network) 통신은 차량용 네트워크 시스템으로 개발된 통신입니다. 주로 마이크로컨트롤러들 간의 통신을 위해 설계된 시리얼 네티워크 통신방식으로, 국제표준 규격으로 제정되어 있습니다. CAN 통신은 1:1 통신뿐 아니라 멀티 마스터 통신이 가능하고, 특히 장거리 통신이 가능하기 때문에, 하나의 컨트롤러를 통해 여러 개의 디바이스를 제어 할 수 있습니다.

CAN 통신의 장점은 다음과 같습니다:

- 통신속도: 최대 1Mbps
- Multi-Master 구조: 통신 신호 충돌 대책이 있음(CSMA/CA)
- 메시지 ID간 우선순위가 있음
- 데이터 송신 충돌 정지 시, 선로가 비어 있을 때 자동 재 전송 기능
- 통신 데이터 수: 8byte
- 통신 프로토콜/에러 처리를 Hardware 적으로 처리
- 멀티 통신 가능(ID구분- Standard: 11bit, Extended: 29bit)
- Noise 환경에 강함

MW-AHRS는 CAN2.0B 규격(Extended CAN)의 프로토콜을 따르고 있으며, 장치 ID는 1에서 255까지 설정 가능합니다.

다음 표는 CAN Bus의 길이에 따른 통신속도(Bitrate)의 설정을 보여주고 있습니다. 사용자는 다음 표를 참조하여 CAN Bus의 길이에 따른 올바른 통신 속도를 사용하기 바랍니다.

BUS Length	Bitrate	Bit time
25M	1000Kbps	1us
100M	500Kbps	2us
250M	250Kbps	8us

2.4.1 연결 구성

PC에서 MW-AHRS으로부터 CAN통신을 통해 데이터를 주고받기 위해서는 USB2CAN과 같은 어댑터가 필요합니다. USB2CAN 어댑터는 DeviceMart에서 구매 가능합니다.

또한 마이크로컨트롤러에서 CAN통신을 사용하기 위해서는 CAN bus에 직접적으로 연결되는 CAN transceiver가 필요합니다. 흔히 사용되는 CAN transceiver는 ATA6660이나, MCP2551, PCA6558 등이 있습니다. MW-AHRS에는 CAN transceiver가 내장되어 있기 때문에 별도로 달아줄 필요는 없습니다.

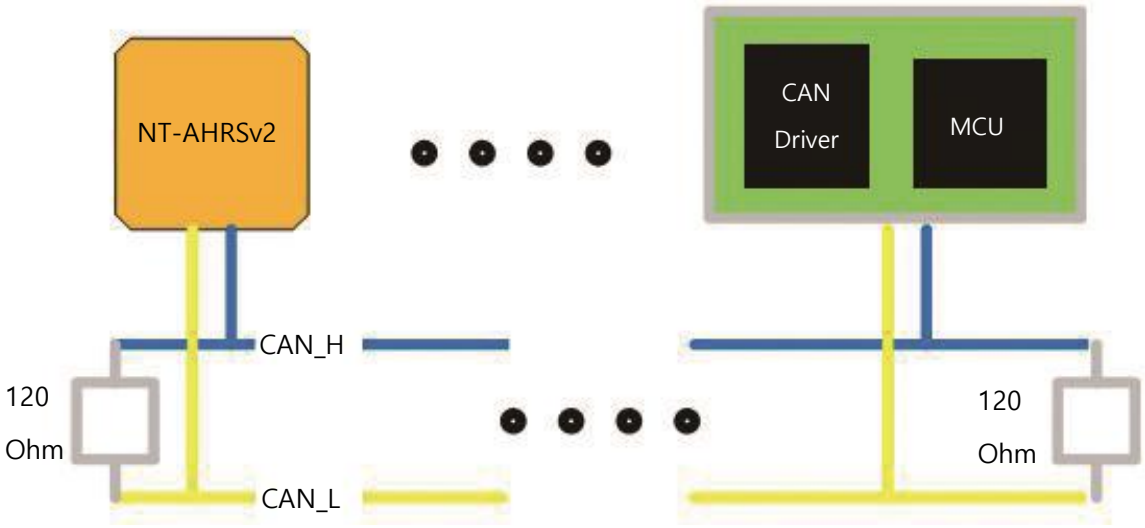


그림 2-3 CAN 연결 구성도

CAN 통신선은 2가닥의 CAN_H, CAN_L 권선으로 이루어진 bus 형태 입니다. 이 bus에 각종 디바이스가 붙게 되며, bus를 통해 통신을 하게 됩니다. 그리고 bus의 양 단에는 임피던스 매칭용 종단 저항(120~130 Ohm)을 달아주어야 합니다.

※주의※ MW-AHRS에는 별도의 종단저항이 달려있지 않습니다. 만약 종단저항을 올바르게 연결하지 않은 경우 CAN 버스를 통해 데이터를 주고 받을 수 없는 상황이 발생합니다.

2.4.2 연결 설정

MW-AHRS의 CAN 통신 속도는 1000K, 800K, 500K, 250K, 125K, 50K, 25K, 10K bps 중 하나를 선택할 수 있습니다. 그리고 CAN의 장치 ID는 1에서 255까지 설정할 수 있습니다.

설정 항목	설정 값
Bitrate	1000K, 800K, 500K, 250K, 125K, 50K, 25K, 10K bps
Device ID	1 ~ 255

※ 제품 출하 시 기본 통신속도는 1000K bps 로 설정되어 있습니다. 그리고 장치 ID는 1로 설정되어 있습니다.

2.4.3 PC 에서 연결예

PC에서는 CAN 인터페이스를 지원하지 않기 때문에 CAN bus에 접속하기 위해서는 USB2CAN과 같은 어댑터가 필요합니다. 다음 그림 2-4는 NTREX USB2CAN 어댑터를 사용하여 PC에서 MW-AHRS 센서로 CAN 통신을 사용하여 연결한 예입니다.

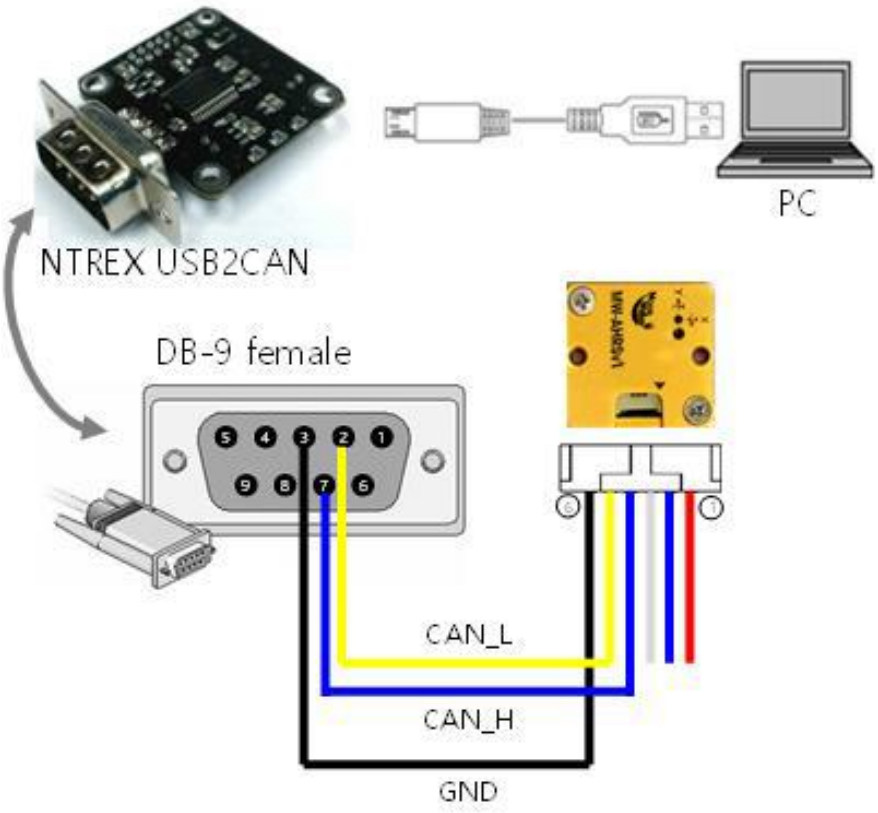


그림 2-4 USB2CAN을 사용한 MW-AHRS 센서와 PC간 연결

USB3CAN의 CAN 포트는 DB9 수(male) 커넥터로 구성되어있습니다. 그렇기때문에 센서로부터 연결되는 신호 선들은 DB9 암(female) 커넥터에 결선 되어야합니다. 다음 그림을 참조하여 센서의 CAN_L, CAN_H, GND 핀을 DB9 암 커넥터의 해당 핀으로 연결합니다.

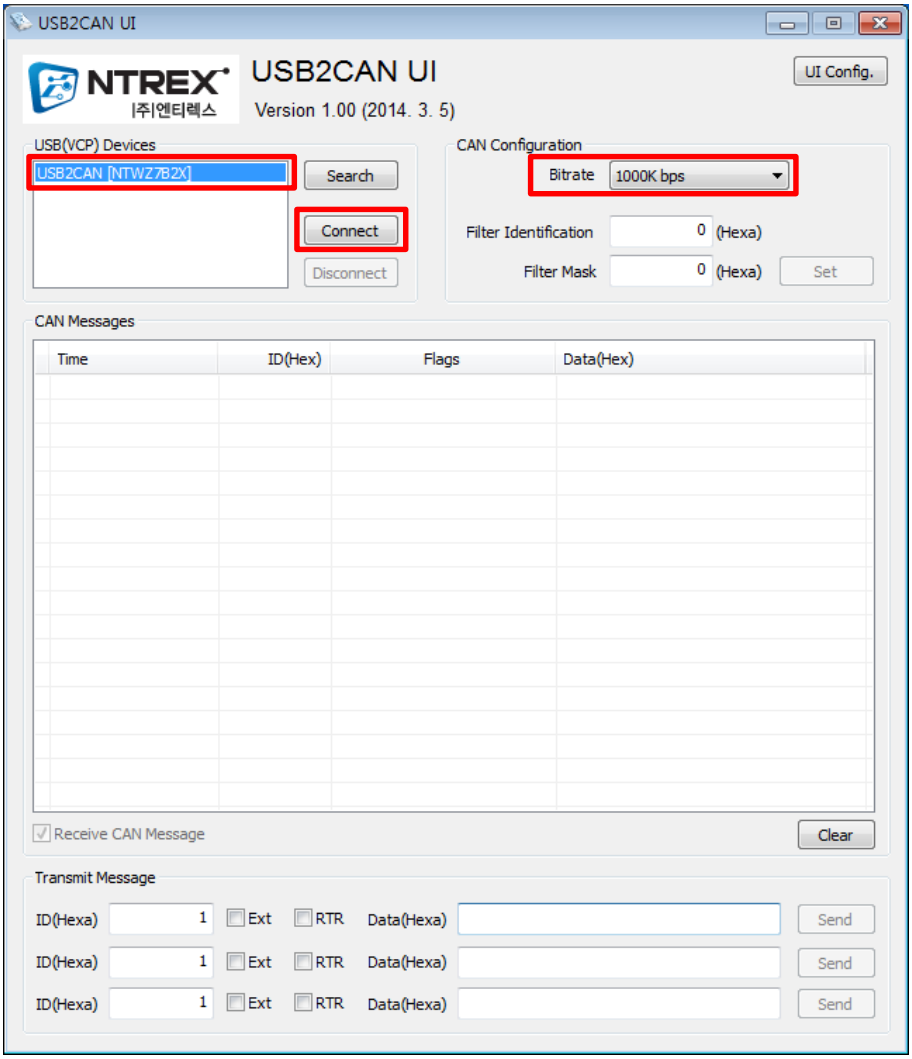
Pin #	Description
2	CAN_L
3	GND
6	CAN_H

그림 2-5 DB-9 CAN 커넥터 핀맵

MW-AHRS 센서를 USB2CAN에 직접 연결하였다면 센서의 CAN_H와 CAN_L 단자간 종단저항 120 Ω을 연결하여야 합니다. 그리고 USB2CAN에서는 CON4의 점퍼를 연결합니다. 이 점퍼는 USB2CAN의 종단저항을 연결하는 점퍼입니다.

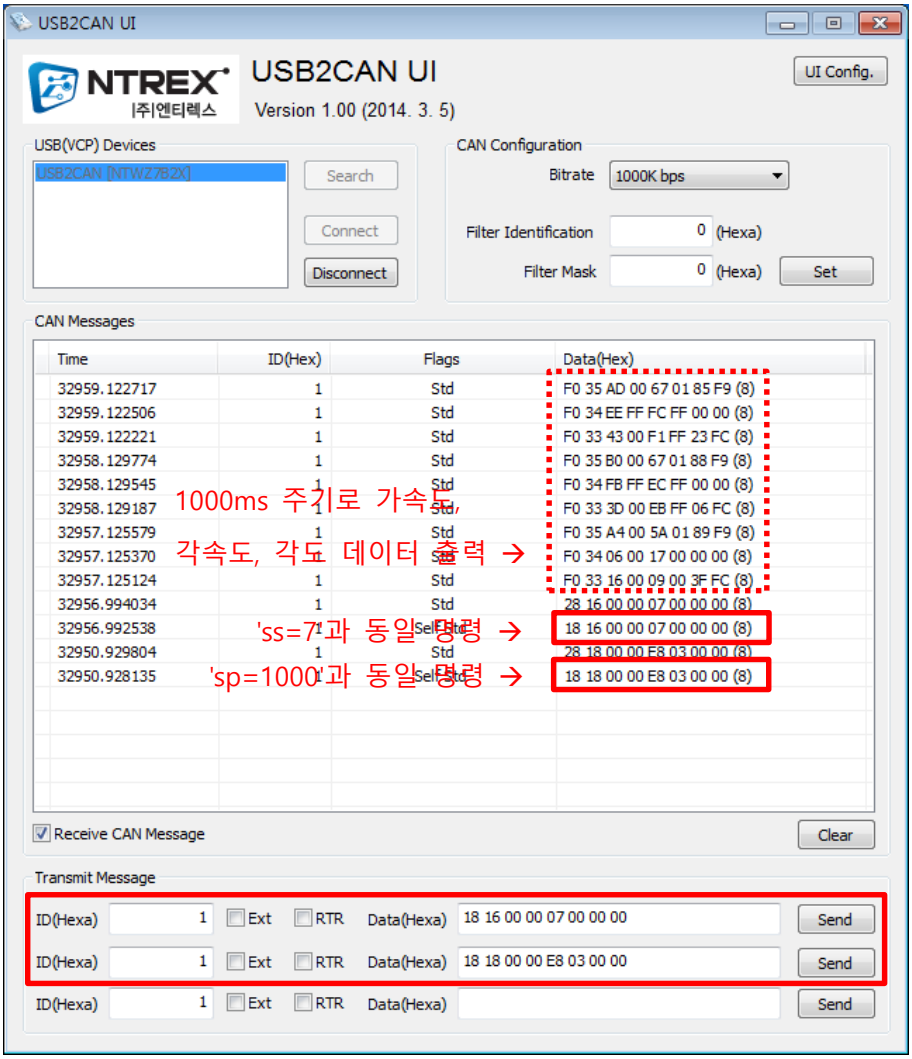
만일 CAN 통신이 올바르게 되지 않는다면, 테스터를 사용하여 CAN_L과 CAN_H 단자간 저항이 60Ω 근처 값으로 측정되는지 확인하기 바랍니다.

이제 USB2CAN UI 프로그램을 실행하고 [Search] 버튼을 누릅니다. 그러면 다음과 같이 연결된 USB2CAN 어댑터를 검색합니다. 검색된 어댑터 중 하나를 선택하고 [Connect] 버튼을 누릅니다. 그리고 왼편 CAN configuration 그룹에서 CAN 통신 속도(Bitrate)가 1000Kbps 로 설정되어 있는지 확인합니다. 만일 MW-AHRS 에서 CAN 통신속도를 변경하였다면 변경한 속도에 맞게 설정합니다. 그리고 [Set] 버튼을 누릅니다.



이제 MW-AHRS 센서로 CAN 패킷을 전송해 보겠습니다. 먼저 하단의 데이터 패킷을 구성하는 창에서 ID를 1로 설정합니다. 그리고 4.2절을 참고하여 데이터 전송주기(sp=1000)를 1000ms로 설정하는 패킷과 CAN 데이터 전송(ss=7)으로 가속도, 각속도, 각도를 설정하는 패킷을 구성합니다.

[Send] 버튼을 누르면 1000ms 주기로 가속도, 각속도, 각도 데이터가 수신되는 것을 확인할 수 있습니다.


USB2CAN UI
NTREX
㈜엔티렉스
USB2CAN UI
Version 1.00 (2014. 3. 5)
UI Config.
USB(VCP) Devices
USB2CAN [NTWZ7B2X]
Search
Connect
Disconnect
CAN Configuration
Bitrate
1000K bps
Filter Identification
0
(Hexa)
Filter Mask
0
(Hexa)
Set
CAN Messages
Time
ID(Hex)
Flags
Data(Hex)
32959.122717 1 Std F0 35 AD 00 67 01 85 F9 (8)
32959.122506 1 Std F0 34 EE FF FC FF 00 00 (8)
32959.122221 1 Std F0 33 43 00 F1 FF 23 FC (8)
32958.129774 1 Std F0 35 B0 00 67 01 88 F9 (8)
32958.129545 1 Std F0 34 FB FF EC FF 00 00 (8)
32958.129187 1 Std F0 33 3D 00 EB FF 06 FC (8)
32957.125579 1 Std F0 35 A4 00 5A 01 89 F9 (8)
32957.125370 1 Std F0 34 06 00 17 00 00 00 (8)
32957.125124 1 Std F0 33 16 00 09 00 3F FC (8)
32956.994034 1 Std 28 16 00 00 07 00 00 00 (8)
32956.992538 1 Std 18 16 00 00 07 00 00 00 (8)
32950.929804 1 Std 28 18 00 00 E8 03 00 00 (8)
32950.928135 1 Std 18 18 00 00 E8 03 00 00 (8)
1000ms 주기로 가속도,
각속도, 각도 데이터 출력 →
'ss=7'과 동일명령 →
'sp=1000'과 동일명령 →
Receive CAN Message
Clear
Transmit Message
ID(Hexa) 1 Ext RTR Data(Hexa) 18 16 00 00 07 00 00 00 Send
ID(Hexa) 1 Ext RTR Data(Hexa) 18 18 00 00 E8 03 00 00 Send
ID(Hexa) 1 Ext RTR Data(Hexa) Send

3 오브젝트

MW-AHRS 센서에 명령을 내리거나 센서의 구성 파라미터를 설정하는 것은 센서 내부의 오브젝트(object)에 값을 쓰는 것을 말하며, 센서의 상수나 상태를 모니터링하는 것은 센서 내부의 오브젝트에서 값을 읽는 것을 말합니다. 이처럼 센서가 가진 각종 오브젝트는 센서와 통신으로 연결된 다른 장치와 데이터를 교환하는 기본 단위가 됩니다.

3.1 오브젝트의 종류

센서 내부의 오브젝트들(objects)은 다음과 같이 4가지 속성을 가지는 그룹으로 나뉘어 집니다:

- 상수(Constant) 오브젝트: 읽기 전용, 센서 실행 도중 변하지 않음
- 명령(Command) 오브젝트: 쓰기 전용, 센서가 수행할 명령 전달
- 상태(Status) 오브젝트: 읽기 전용, 센서의 상태나 측정된 데이터
- 구성 파라미터(Configuration Parameter) 오브젝트: 읽고 쓰기 가능, 플래시 메모리 저장

상수 오브젝트는 센서에서 고정된 값으로 변하지 않는 오브젝트입니다. 센서의 소프트웨어/하드웨어 버전, 모델 번호와 같은 것들이 여기에 해당합니다.

명령 오브젝트는 센서가 지정된 동작을 실행하도록 하는 명령 코드 값을 쓰는 오브젝트입니다. 명령 코드를 쓰는 것은, 명령 오브젝트가 이미 개별 코드에 대해 '1-Flash Write, 2-Load Factory Default Values, 3-Calibrate'과 같이 기능이 부여된 상태에서 한 가지 기능을 선택하는 것과 같습니다.

상태 오브젝트는 센서의 측정값을 저장하고 있는 오브젝트로 마스터 PC는 읽기만 가능합니다. 이 값은 센서가 실행되면서 계속 바뀌게 됩니다. 보통 마스터 PC가 센서의 측정값을 알기 위해 상태 오브젝트를 주기적으로 읽어오게 됩니다.

구성 파라미터 오브젝트는 센서의 구동과 관련된 여러 파라미터를 설정하는데 사용됩니다. 센서가 가진 대부분의 오브젝트가 구성 파라미터이며, 이 오브젝트를 어떻게 설정하는지에 따라 센서를 다양한 방식으로 운용할 수 있게 됩니다.

3.2 오브젝트 요약

다음 표는 MW-AHRS 센서가 가진 오브젝트를 요약해 놓은 표입니다. 각각의 오브젝트에는 Name이 할당되어 있으며, 각 이름과 매칭되는 Index가 있습니다. 그리고 오브젝트가 한 개 이상의 값을 가질 때는 각각의 값은 Sub-index(표에서는 Sub-i로 표기)로 구분됩니다.

또한 표에서 각 오브젝트의 엑세스 속성은 RO(Read Only), WO(Write Only), RW(Read Write)로 표기되어 있으며, 오브젝트의 크기 속성은 INT32(부호있는 32bit 정수형)와 FLOAT(부호있는 32bit 실수형)로 표기되어 있습니다.

표 3-1 MW-AHRS 센서의 오브젝트 요약

Name	Index	Sub-i	Access, Size	Description	Default
ver	1	0	RO, INT32	공급자 ID, 0으로 고정	0
ver	2	0	RO, INT32	제품(AHRS 센서) ID	5001
ver	3	0	RO, INT32	장치 펌웨어 버전	100
ver	4	0	RO, INT32	장치 하드웨어 버전	200
cmd	7	0	WO, INT32	장치에 내려지는 명령 (RS-232 Text 모드에서는 다음 명령 사용 가능: fw, fd, cal, cam, zro, rcd, rst, ver, h, help)	
id	11	0	RW, INT32	장치 ID	1
cb	12	0	RW, INT32	CAN 통신 속도 [Kbps]	1000
sb	13	0	RW, INT32	RS-232 통신 속도 [bps]	115200
gs	15	0	RW, INT32	자이로 센서의 측정 스케일 설정	0 ~ 3
as	16	0	RW, INT32	가속도 센서의 측정 스케일 설정	0 ~ 3
mv	19	0	RW, INT32	자기 센서의 측정값에 대한 분산 설정	0
av	20	0	RW, INT32	가속도 센서의 측정값에 대한 분산 설정	1000
ss	21	0	RW, INT32	RS-232로 동기화 데이터 전송 설정	0
sc	22	0	RW, INT32	CAN으로 동기화 데이터 전송 설정	0
sp	24	0	RW, INT32	동기화 데이터 전송 주기 설정 [ms]	100
st	25	0	RW, INT32	장치에 전원이 투입될 때, RS-232 데이터 전송 타입 결정 (0-Binary, 1-Text)	1
acc	51	1~3	RO, FLOAT	가속도 데이터 전송 (x, y, z) [g]	
gyr	52	1~3	RO, FLOAT	각속도 데이터 전송 ($\omega_x, \omega_y, \omega_z$) [°/s]	
ang	53	1~3	RO, FLOAT	오일러 각도 전송 (ϕ, θ, ψ) [°]	
mag	54	1~3	RO, FLOAT	자기 데이터 전송 (x, y, z) [μT]	
tmp	57	1	RO, FLOAT	온도 전송 [°C]	

※ gs, as 오브젝트는 펌웨어 버전 2.0 이상에서 사용 가능합니다.

3.3 제품 ID 및 버전 정보

제품 ID 및 소프트웨어/하드웨어 버전은 상수 오브젝트로 읽기만 가능합니다. 이 상수들은 제품 생산 시 결정되며 사용자가 변경할 수 없습니다.

```
Ex) ver↲              // 제품 ID 및 버전 정보 읽기 요청
NTREX NT_AHRS         // 센서가 출력한 내용
H/W Ver 1.00
S/W Ver 1.10
```

3.3.1 공급자 ID

공급자 ID는 0으로 고정되어 있습니다.

3.3.2 제품 ID

현재 센서에 적용된 제품 ID는 다음과 같습니다:

- 5001 - NT-ARSv2
- 5011 – MW-AHRS

새로운 제품이 출시됨에 따라 제품 ID의 종류는 늘어날 수 있습니다.

3.3.3 소프트웨어(펌웨어) 버전

버전은 다음과 같이 소수점을 기준으로 상위 1~2자리와 하위 2자리로 구분됩니다.

버전: XX.YY

XX는 메이저 버전이고 YY는 마이너 버전입니다. 메이저 버전이 변경된 경우에는 MW-AHRS 센서와 AHRS UI 유틸리티 및 기타 응용 소프트웨어들간에 호환되지 않습니다. 센서와 관련 소프트웨어들이 서로 호환되는 범위 내에서 마이너 버전은 변경될 수 있습니다.

소프트웨어 버전은 센서의 생산 일자에 따라 달라질 수 있으며 센서에 최신 펌웨어를 업데이트 하는 경우에도 달라질 수 있습니다.

3.3.4 하드웨어 버전

버전은 다음과 같이 소수점을 기준으로 상위 1~2자리와 하위 2자리로 구분됩니다.

버전: XX.YY

XX는 메이저 버전이고 YY는 마이너 버전입니다. 메이저 버전이 변경된 경우에는 MW-AHRS 센서와 AHRS UI 유틸리티 및 기타 응용 소프트웨어들간에 호환되지 않습니다.

하드웨어 버전은 센서의 생산 일자에 따라 달라질 수 있습니다.

3.4 명령

3.4.1 센서 명령 (cmd)

cmd에 명령 코드를 쓰는 것으로 센서의 특정 기능을 실행할 수 있습니다. 다음은 명령 코드 목록입니다:

- 1 - 센서의 구성 파라미터를 플래시 메모리에 저장 ('cmd=1' 또는 'fw')
- 2 - 센서의 구성 파라미터를 공장 출하시 설정값으로 초기화 ('cmd=2' 또는 'fd')
- 3 - 가속도와 각속도 센서의 바이어스와 스케일 보정 실시 ('cmd=3' 또는 'cal')
- 4 - 자기 센서의 바이어스와 스케일 보정 실시 ('cmd=4' 또는 'cam')
- 5 - Euler 각도를 0으로 리셋 ('cmd=5' 또는 'zro')
- 9 - 캘리브레이션 데이터만 리셋 ('cmd=9' 또는 'rcd')
- 99 - 센서를 소프트웨어 리셋 함 ('cmd=99' 또는 'rst')

```
Ex) cmd=3↲     // 가속도 센서와 자이로 센서의 바이어스와 스케일 보정 명령을 내림
```

※ 'rcd' 명령은 펌웨어 버전 2.0 이상에서 사용 가능합니다.

3.4.2 플래시 메모리 저장 (cmd=1, fw)

센서의 구성 파라미터 값을 변경하면, 변경된 값들은 RAM에 기록되어 센서가 켜져 있는 동안만 유지됩니다. 만일 센서가 재시작 되면 이 값들은 소실되고 플래시 메모리에 저장된 값을 RAM으로 다시 읽어 들이게 됩니다. 그렇기 때문에 구성 파라미터 값을 변경하고 계속 유지되기를 바란다면, 'cmd=1' 명령을 사용하여 변경한 구성 파라미터 값을 플래시 메모리에 저장해야 합니다.

```
Ex) cmd=1↲ 또는 fw↲  // 센서 설정들을 플래시 메모리에 저장 명령을 내림
```

3.4.3 공장 출하시 설정값으로 초기화 (cmd=2, fd)

이 명령 코드는 현재 센서에 설정된 모든 구성 파라미터 값들을 무시하고 제품 초기 설정 값(Factory Default Value)으로 복구합니다. 이 명령을 실행하면, RAM 및 플래시 메모리에 있는 모든 값들이 제품 초기 설정 값으로 복구됩니다.

```
Ex) cmd=2↲ 또는 fd↲  // 구성 파라미터 설정들을 공장 출하시 설정값으로 초기화
```

3.4.4 가속도와 자이로 센서의 스케일/바이어스 보정 (cmd=3, cal)

가속도와 자이로 센서의 스케일/바이어스 보정(scale/bias calibration)은 MW-AHRS에 내장되어 있는 가속도, 자이로 센서의 스케일과 바이어스 값을 다시 설정해주는 기능입니다. 스케일/바이어스 보정을 한 번 수행하는데에는 2초 정도의 시간이 소요되며, 여러 번의 반복적인 보정 과정이 수

행되어야 합니다. 보정에 대한 자세한 내용은 '5.5 센서 측정값 보정' 절을 참고하시기 바랍니다. 이 명령을 수행하는 동안에는 다른 어떤 명령어도 실행하지 않습니다. 스케일/바이어스 보정 기능이 실행 중일때는 LED가 빠른 속도로 깜박이거나 켜진 상태를 2초간 유지합니다.

```
Ex) cmd=3↲ 또는 cal↲   // 가속도와 자이로 센서의 스케일/바이어스 보정 명령을 내림
```

※ 스케일/바이어스 보정 명령은 MW-AHRS를 기본설정 후 MW-AHRS을 재부팅하고 설정하기 바랍니다. 또한 MW-AHRS는 제품출하시 적절한 스케일과 바이어스 값을 적용하여 출하 되므로 반드시 재설정이 필요할 때만 사용하기 바랍니다.

※주의※ 스케일/바이어스 보정 명령을 수행하기 전에 MW-AHRS 센서를 정반과 같이 수평이 보장되는 곳에 올려두고, 스케일/ 바이어스 보정 중일 때는 센서를 움직이거나 진동이 가해지면 안됩니다.

3.4.5 자기 센서의 스케일/바이어스 보정 (cmd=4, cam)

자기 센서의 스케일/바이어스 보정(scale/bias calibration)은 MW-AHRS에 내장되어 있는 자기 센서의 스케일과 바이어스 값을 다시 설정해주는 기능입니다. 스케일/바이어스 보정에는 60초 정도의 시간이 소요됩니다. 이 명령을 수행하는 동안에는 다른 어떤 명령어도 실행하지 않습니다. 스케일/바이어스 보정 기능이 실행 중일때는 LED가 빠른 속도로 깜박이다가 보정 과정이 성공적으로 완료되면 켜진 상태를 2초간 유지한 후 정상 동작 모드로 복귀합니다.

```
Ex) cmd=4↲ 또는 cam↲   // 자기 센서의 스케일/바이어스 보정 명령을 내림
```

3.4.6 Euler 각도를 0 으로 리셋 (‘cmd=5’ 또는 ‘zro’)

센서의 각속도 데이터를 적분한 Euler 각도를 모두 0으로 리셋 합니다. 센서는 주변 환경에 영향을 받기 때문에(특히 온도) 센서를 올바르게 캘리브레이션 했더라도 센서의 전원을 켜면 초기에는 약간의 바이어스로 인한 yaw 각도에서 드리프트가 발생합니다. 센서는 내부적으로 바이어스 보정 알고리즘이 동작하여 수 분 후에는 안정화 됩니다. 이 때 zro 명령으로 Euler 각도를 모두 리셋하여 원점으로 사용하면 됩니다.

```
Ex) cmd=5↲ 또는 zro↲   // Euler 각도를 0으로 리셋
```

※ Version 2.0 이상에서 사용 가능한 명령입니다.

3.4.7 캘리브레이션 데이터 리셋 (cmd=9, rcd)

각속도 센서, 가속도 센서, 자기 센서의 스케일/바이어스 등의 센서 캘리브레이션 값이 모두 초기값으로 리셋 됩니다. 캘리브레이션과 관련 없는 센서 파라미터는 초기값으로 리셋 되지 않습니다.

만일 센서의 캘리브레이션 데이터를 리셋 하였다면, 본 메뉴얼의 캘리브레이션 절차를 참고하여 캘리브레이션을 진행한 후 사용하시기 바랍니다.

```
Ex) cmd=9↲ 또는 rcd↲   // 캘리브레이션 데이터만 리셋
```

※ Version 2.0 이상에서 사용 가능한 명령입니다.

3.4.8 장치 리셋 (cmd=99, rst)

이 명령 코드는 센서를 소프트웨어적으로 리셋 시킵니다. 센서의 전원을 껐다 켜는 것과 같습니다.

```
Ex) cmd=99↲ 또는 rst↲   // 센서 리셋 명령을 내림
```

3.5 통신 설정

이 절에서는 PC 및 마이크로컨트롤러와 센서를 연결하는 RS-232와 CAN 포트의 통신 설정에 대해 다룹니다.

3.5.1 장치 ID (id)

CAN 또는 RS-422, RS-485 버스에는 여러 장치가 동시에 연결되어 통신 선로를 서로 공유합니다. 이때, 장치(device) ID는 각각의 장치를 구분하기 위한 ID 값입니다. 그러므로 하나의 통신 선로에 연결된 장치의 ID는 모두 달라야 합니다.

제품 초기 설정 값(Factory Default Value)으로 센서의 장치 ID는 1로 설정되어 있습니다. 이 값은 0과 255사이의 값으로 변경될 수 있습니다.

PC나 마이크로컨트롤러에 CAN이나 RS-232 포트를 통해 하나의 센서만 연결하는 경우, 이 값을 변경하지 않고 기본 설정 값을 사용하면 됩니다.

```
Ex) id=10↲     // 장치 ID를 10으로 설정
```

3.5.2 CAN 통신 속도 (cb)

만일 센서를 CAN 버스에 연결한다면, CAN 버스에 연결되는 모든 장치들은 서로 통신 속도가 일치해야 합니다.

CAN 통신 속도 파라미터의 값으로 다음 중 하나를 설정할 수 있습니다:

- 10 – 10Kbps

- 25 – 25Kbps
- 50 – 50Kbps
- 125 – 125Kbps
- 250 – 250Kbps
- 500 – 500Kbps
- 800 – 800Kbps
- 1000 – 1Mbps (기본값)

```
Ex) cb=800↲     // CAN 통신 속도를 800Kbsp로 설정
```

※ CAN 통신 속도를 변경한 경우에는 센서 전원을 껏다 켜거나 센서를 소프트웨어적으로 리셋 ('cmd=99' 또는 'rst') 하여야 변경된 통신 속도가 적용됩니다. 전원을 끄기 전 'fw' 명령으로 변경된 파라미터를 플래시 메모리에 저장하십시오.

3.5.3 시리얼(RS-232) 통신 속도 (sb)

PC나 마이크로컨트롤러의 RS-232 포트에 센서를 연결하는 경우, 시리얼(RS-232) 통신 속도를 설정해야 합니다.

시리얼 통신 속도 파라미터의 값으로 다음 중 하나를 설정할 수 있습니다:

- 9600 – 9600bps
- 19200 – 19200bps
- 38400 – 38400bps
- 57600 – 57600bps
- 115200 – 115200bps (기본값)
- 230400 – 230400bps
- 460800 – 460800bps
- 921600 – 921600bps

```
Ex) sb=460800↲     // 시리얼 통신 속도를 460800bsp로 설정
```

※ 시리얼 통신 속도를 변경한 경우에는 센서 전원을 껏다 켜거나 센서를 소프트웨어적으로 리셋 ('cmd=99' 또는 'rst') 하여야 변경된 통신 속도가 적용됩니다. 전원을 끄기 전 'fw' 명령으로 변경된 파라미터를 플래시 메모리에 저장하십시오.

3.6 센서 스케일 설정

3.6.1 자이로 센서의 입력 스케일 (gs)

자이로 센서의 최대 측정 스케일을 설정합니다. 센서가 빠르게 회전하는 경우 입력 스케일은 큰 값으로 설정해야 합니다. 하지만 측정값의 정밀도는 낮아지게 됩니다. 반대로 센서가 천천히 회전

하는 경우 입력 스케일은 작은 값으로 설정하고 측정값의 정밀도를 높일 수 있습니다.

자이로 센서의 입력 스케일 값으로 다음 중 하나를 설정할 수 있습니다:

- 0 – 250dps
- 1 - 500dps
- 2 - 1000dps
- 3 - 2000dps

```
Ex) gs=3↲      // 자이로 센서의 입력 스케일을 2000dps로 설정
```

※ 자이로 센서의 입력 스케일을 변경한 경우에는 센서 전원을 껏다 켜거나 센서를 소프트웨어적으로 리셋('cmd=99' 또는 'rst') 하여야 변경된 통신 속도가 적용됩니다. 전원을 끄기 전 'fw' 명령으로 변경된 파라미터를 플래시 메모리에 저장하십시오.

3.6.2 가속도 센서의 입력 스케일 (as)

가속도 센서의 최대 측정 스케일을 설정합니다. 센서가 빠르게 움직이는 경우 입력 스케일은 큰 값으로 설정해야 합니다. 하지만 측정값의 정밀도는 낮아지게 됩니다. 반대로 센서가 천천히 움직이는 경우 입력 스케일은 작은 값으로 설정하고 측정값의 정밀도를 높일 수 있습니다.

가속도 센서의 입력 스케일 값으로 다음 중 하나를 설정할 수 있습니다:

- 0 – 2g
- 1 - 4g
- 2 - 8g
- 3 - 16g

```
Ex) as=3↲      // 가속도 센서의 입력 스케일을 16g로 설정
```

※ 가속도 센서의 입력 스케일을 변경한 경우에는 센서 전원을 껏다 켜거나 센서를 소프트웨어적으로 리셋('cmd=99' 또는 'rst') 하여야 변경된 통신 속도가 적용됩니다. 전원을 끄기 전 'fw' 명령으로 변경된 파라미터를 플래시 메모리에 저장하십시오.

3.7 칼만 필터 융합

칼만 필터에서 각속도를 적분하여 얻은 각도를 중력과 지구 자기장으로 보정하는데 관련되는 파라미터에 대해 설명합니다.

3.7.1 가속도 센서의 분산 (av)

칼만필터에서 사용하는 자이로 센서와 가속도 센서의 측정 데이터에 대한 정밀도를 상대적으로 설정합니다. 자이로 센서의 분산은 1로 고정되어 있습니다. 그리고 가속도 센서의 분산을 1 ~ 1,000,000 사이의 값으로 조정함으로 상대적인 측정 정밀도를 설정하게 됩니다. 기본 값으로 1,000이 설정되어 있습니다. 이 값을 0으로 설정한 경우, 칼만필터는 중력의 방향으로 roll과 pitch 각도 보정 과정을 수행하지 않습니다.

분산이 커지게 되면 그만큼 측정데이터에 노이즈를 많이 포함하고 있다는 의미가 됩니다. 가속도 센서의 분산을 크게하면 중력가속도에 의해 보정되는 roll과 pitch 각도의 비율이 작아집니다. 반대로 작게하면 비율이 커지게 됩니다.

```
Ex) av=10000↲     // 가속도 센서의 분산을 10,000으로 설정
```

3.7.2 자기 센서의 분산 (mv)

칼만필터에서 사용하는 자이로 센서와 자기 센서의 측정 데이터에 대한 정밀도를 상대적으로 설정합니다. 자이로 센서의 분산은 1로 고정되어 있습니다. 그리고 자기 센서의 분산을 1 ~ 1,000,000 사이의 값으로 조정함으로 상대적인 측정 정밀도를 설정하게 됩니다. 기본 값으로 0이 설정되어 있습니다. 이 값을 0으로 설정한 경우, 칼만필터는 지구 자기장의 방향으로 yaw 각도 보정 과정을 수행하지 않습니다.

분산이 커지게 되면 그만큼 측정데이터에 노이즈를 많이 포함하고 있다는 의미가 됩니다. 자기 센서의 분산을 크게하면 지구 자기장에 의해 보정되는 yaw 각도의 비율이 작아집니다. 반대로 작게하면 비율이 커지게 됩니다.

```
Ex) mv=10000↲     // 자기 센서의 분산을 10,000으로 설정
```

3.8 측정 및 동기화(주기적 데이터 전송)

동기화와 관련된 파라미터들은 CAN이나 RS-232 포트로 측정 데이터를 주기적으로 전송하는데 필요한 설정을 제공합니다.

3.8.1 시리얼(RS-232) 데이터 전송 (ss)

RS-232 포트를 통해 주기적으로 전송되는 데이터 종류를 선택합니다. 시리얼 데이터 전송 파라미터에 설정되는 값에 따라 표 3-2에서와 같은 데이터가 전송됩니다.

표 3-2 데이터 전송 설정에 따른 전송 데이터 종류

값	데이터 종류
0	데이터 전송 중단
1	가속도 데이터
2	각속도 데이터
3	가속도, 각속도 데이터
4	각도 데이터
5	가속도, 각도 데이터
6	각속도, 각도 데이터
7	가속도, 각속도, 각도 데이터
8	자기 데이터
9	가속도, 자기 데이터
10	각속도, 자기 데이터
11	가속도, 각속도, 자기 데이터
12	각도, 자기 데이터
13	가속도, 각도, 자기 데이터
14	각속도, 각도, 자기 데이터
15	가속도, 각속도, 각도, 자기 데이터

```
Ex) ss=4↲     // RS-232 포트를 통해 각도 데이터 전송
```

시리얼 텍스트 통신에서 전송되는 포멧은 4.3.5절을 참고하십시오.

시리얼 바이너리 통신에서 전송되는 포멧은 4.4.6절을 참고하십시오.

3.8.2 CAN 데이터 전송 (sc)

CAN 포트를 통해 주기적으로 전송되는 데이터 종류를 선택합니다. CAN 데이터 전송 파라미터에 설정되는 값에 따라 표 3-2에서와 같은 데이터가 전송됩니다.

```
Ex) sc=7↲     // CAN 포트를 통해 가속도, 각속도, 각도 데이터 전송
```

CAN 통신에서 전송되는 포멧은 4.2.6절을 참고하십시오.

3.8.3 데이터 전송 주기 (sp)

CAN이나 RS-232 포트를 통해 전송되는 데이터의 전송 주기를 설정합니다. 데이터 전송 주기는

1ms부터 60,000ms까지 설정 가능합니다. 만일 CAN이나 RS-232 통신으로 데이터를 전송하기 위한 충분한 대역폭이 확보되지 않았다면, 그만큼 전송속도는 느려지게 됩니다.

```
Ex) sp=100↲     // 데이터 전송 주기를 100ms로 설정
```

시리얼 통신 속도가 460800bps일 때, 데이터 전송 주기를 1로 설정(sp = 1)하고 시리얼 데이터 전송에서 데이터 종류를 4로 설정(ss = 4)한 경우, 실험적으로 측정한 초당 전송 데이터 수는 약 500개 정도가 됩니다.

시리얼 통신 속도가 460800bps인 경우 초당 약 46Kbyte의 문자를 전송할 수 있습니다. 그리고 ss가 4이고 sp가 1인 경우, 초당 약 30Kbyte의 문자가 생성되어 전송되어야 합니다. 이는 시리얼의 대역폭이 데이터를 전송가능한 수준이지만, MCU가 AHRS 알고리즘의 수행에도 많은 시간을 소비 하기 때문에 데이터 전송 주기를 느려지게 합니다.

※ 데이터 전송 주기는 보내야 할 데이터 량과 시리얼이나 CAN의 통신속도를 고려하여 여유있게 설정하기 바랍니다.

3.9 측정 데이터

MW-AHRS 센서는 가속도 센서와 자이로 센서, 자기 센서로부터 가속도와 각속도, 자기 값을 측정합니다. 그리고 칼만필터로 이 세 데이터를 융합하여 오일러각도를 계산합니다. 다음은 센서로부터 읽어올 수 있는 측정 데이터에 대해 설명합니다.

3.9.1 가속도 (acc)

가속도 센서에서 측정한 가속도 값에서 바이어스와 스케일을 보정한 값입니다. 가속도의 단위는 [g]입니다. 1g는 9.8m/s^2 입니다.

시리얼 텍스트 통신에서는 x축, y축, z축 가속도를 동시에 읽어옵니다. 회신 문자열에서 각 축의 값은 8byte의 ASCII 문자로 소수점 셋째 자리까지 표시됩니다. 각 축의 값을 구분하기 위해 공백(0x20)이 삽입됩니다. 표시되는 수가 8byte가 되지 않을 때는 오른쪽 정렬되고 앞에는 공백이 삽입됩니다. 다음 예제를 참고하십시오.

```
Ex) acc↲                                  // 가속도 요청
    acc=  -0.487   -0.101   -0.855↲        // 가속도 요청에 대한 응답
```

하지만 시리얼 바이너리나 CAN 통신에서는 각 축별 가속도를 Sub-indx를 지정하여 각각 읽어옵니다.

Index	Sub-index	데이터 종류
51	1	x축의 가속도 [g]
51	2	y축의 가속도 [g]
51	3	z축의 가속도 [g]

3.9.2 각속도 (gyr)

자이로 센서에서 측정한 각속도 값에서 바이어스와 스케일을 보정한 값입니다. 각속도의 단위는 [°/s]입니다. 여기서 각도(°)는 degree로 표시됩니다.

시리얼 텍스트 통신에서는 x축, y축, z축 각속도를 동시에 읽어옵니다. 회신 문자열에서 각 축의 값은 8byte의 ASCII 문자로 소수점 둘째 자리까지 표시됩니다. 각 축의 값을 구분하기 위해 공백(0x20)이 삽입됩니다. 표시되는 수가 8byte가 되지 않을 때는 오른쪽 정렬되고 앞에는 공백이 삽입됩니다. 다음 예제를 참고하십시오.

```
Ex) gyr↲                                    // 각속도 요청
    gyr=   0.80    0.20    0.00↲            // 각속도 요청에 대한 응답
```

하지만 시리얼 바이너리나 CAN 통신에서는 각 축별 각속도를 Sub-indx를 지정하여 각각 읽어옵니다.

Index	Sub-index	데이터 종류
52	1	x축의 각속도 [°/s]
52	2	y축의 각속도 [°/s]
52	3	z축의 각속도 [°/s]

3.9.3 각도 (ang)

각속도와 가속도 데이터를 칼만필터로 융합한 오일러각을 읽어옵니다. **오일러각은 z-y-x축 순서로 회전**하였으며, x축으로 회전한 각을 phi, y축으로 회전한 각을 theta, z축으로 회전한 각을 psi로 정의합니다.

각도의 단위는 [°]입니다. 여기서 각도는 degree로 표시됩니다.

시리얼 텍스트 통신에서는 phi, theta, psi 각도를 동시에 읽어옵니다. 회신 문자열에서 각 축의 값은 8byte의 ASCII 문자로 소수점 둘째 자리까지 표시됩니다. 각 축의 값을 구분하기 위해 공백(0x20)이 삽입됩니다. 표시되는 수가 8byte가 되지 않을 때는 오른쪽 정렬되고 앞에는 공백이 삽입됩니다. 다음 예제를 참고하십시오.

```
Ex) ang↲                                    // 각도 요청
    ang=    8.43   -29.86   117.37↲        // 각도 요청에 대한 응답
```

하지만 시리얼 바이너리나 CAN 통신에서는 각 축별 각속도를 Sub-indx를 지정하여 각각 읽어옵니다.

Index	Sub-index	데이터 종류
53	1	x축의 회전각, phi [°]
53	2	y축의 회전각, theta [°]
53	3	z축의 회전각, psi [°]

3.9.4 자기 (mag)

자기 센서에서 측정한 자기 값에서 바이어스와 스케일을 보정한 값입니다. 자기의 단위는 [μT]입니다. 우리나라(대한민국)에서 읽은 자기 값은 대략 50 μT 정도가 됩니다.

시리얼 텍스트 통신에서는 x축, y축, z축 각속도를 동시에 읽어옵니다. 회신 문자열에서 각 축의 값은 8byte의 ASCII 문자로 소수점 둘째 자리까지 표시됩니다. 각 축의 값을 구분하기 위해 공백(0x20)이 삽입됩니다. 표시되는 수가 8byte가 되지 않을 때는 오른쪽 정렬되고 앞에는 공백이 삽입됩니다. 다음 예제를 참고하십시오.

```
Ex) mag↲                                    // 자기 요청
    mag=  -44.03    1.26    20.15↲          // 자기 요청에 대한 응답
```

하지만 시리얼 바이너리나 CAN 통신에서는 각 축별 자기를 Sub-indx를 지정하여 각각 읽어옵니다.

Index	Sub-index	데이터 종류
54	1	x축의 자기 [μT]
54	2	y축의 자기 [μT]
54	3	z축의 자기 [μT]

3.9.5 온도 (tmp)

센서 내부의 온도 센서에 의해 측정된 값입니다.

온도의 단위는 [℃]입니다.

시리얼 텍스트 통신으로 회신되는 문자열은 8byte의 ASCII 문자로 소수점 둘째 자리까지 표시됩니다. 표시되는 수가 8byte가 되지 않을 때는 오른쪽 정렬되고 앞에는 공백이 삽입됩니다. 다음 예제를 참고하십시오.

```
Ex) tmp↲                    // 온도 요청
    tmp=   36.71↲           // 온도 요청에 대한 응답
```

시리얼 바이너리나 CAN 통신에서는 Sub-indx를 지정하여 읽어옵니다.

Index	Sub-index	데이터 종류
57	0 or 1	센서의 내부온도 [℃]

4 통신 프로토콜

이 장에서는 PC나 마이크로컨트롤러에서 센서의 오브젝트 값을 읽고 쓰기 위한 통신 프로토콜에 대해 설명합니다. 센서에 동작을 명령하거나 센서의 구성 파라미터를 설정하는 것은 해당 오브젝트에 특정 값을 쓰는 것을 의미하며, 센서의 상태를 읽거나 센서의 구성 파라미터를 읽는 것은 해당 오브젝트에서 특정 값을 읽는 것을 의미합니다.

통신 프로토콜은 크게 CAN 포트에서 사용되는 프로토콜과 시리얼(RS-232) 포트에서 사용되는 프로토콜로 구분됩니다. 시리얼 포트 프로토콜은 다시 패킷의 구조에 따라 **바이너리(binary)** 형태와 **텍스트(text)** 형태로 구분됩니다.

4.1 용어의 정의

센서의 객체를 통신 프로토콜로 읽고 쓰는데 사용되는 용어를 표 4-1에서 정의합니다.

표 4-1 통신 프로토콜에 사용되는 용어

Term	Description
Index	오브젝트의 참조 색인
Sub-index	오브젝트의 참조 부-색인
Name	Index와 호환되는 오브젝트의 짧은 이름

상기 표에서 사용되는 용어 중 Index와 Name은 오브젝트를 표현하는 방식의 차이입니다. Index는 센서 내부의 오브젝트를 참조하는 색인으로, CAN 및 시리얼 바이너리 패킷에서 사용됩니다.

Sub-index는 오브젝트를 참조하는 부-색인으로, 측정 데이터의 좌표 축을 의미합니다. x축의 측정값이라면 Sub-index로 1을, y축의 측정값에는 2를 사용하면 됩니다.

Name은 Index와 호환되는 오브젝트의 이름으로, 시리얼 텍스트 패킷에서 사용됩니다. 이 이름은 센서 내부에서 Index 값으로 변환되어 Index와 동일하게 사용됩니다.

표 4-2에서는 오브젝트의 크기를 보여줍니다. 여기서는 INT32와 FLOAT 두 종류를 사용합니다.

표 4-2 오브젝트의 크기

Object Type	Size	Description
INT32	4bytes	부호를 가지는 32 bit 정수형
FLOAT	4bytes	부호를 가지는 32 bit 실수형 (IEEE 754 표준)

또한, 표 4-3에서는 오브젝트의 엑세스 속성을 나타냅니다. RO로 표시된 오브젝트는 읽기 전용으로 상수(Constant)와 상태(Status) 오브젝트가 여기에 해당하며, 값을 쓰려고 할 때 에러를 리턴할 것입니다. WO로 표시된 오브젝트는 쓰기 전용으로 명령(Command) 오브젝트가 여기에 해당하며, 값을 읽으려고 할 때 에러를 리턴할 것입니다. RW로 표시된 오브젝트는 구성 파라미터(Configuration Parameter) 오브젝트가 여기에 해당합니다.

표 4-3 오브젝트의 엑세스 속성

Access	Object	Description
RO	Constant, Status	읽기 전용 (Read Only)
WO	Command	쓰기 전용 (Write Only)
RW	Configuration Parameter, Variable	읽고 쓰기 가능 (Read Writeable)

4.2 CAN 통신

센서의 구성 파라미터(Configuration Parameter)를 설정하고 명령(Command)을 내리거나 상태(Status)를 읽기 위해서 센서에 존재하는 여러 오브젝트들을 CAN 통신으로 엑세스 할 수 있습니다. 이 절은 센서에 존재하는 오브젝트들을 엑세스하기 위한 CAN 통신의 메시지 구조에 대해 기술합니다.

※ 마스터 PC와 센서가 CAN 버스에 연결되어 통신하기 위해서는, 네트워크에 연결된 모든 노드의 통신 속도가 일치해야 하고 각각의 장치 ID는 모두 달라야 합니다.

4.2.1 CAN 패킷의 기본 구조

마스터 PC와 센서 간에 CAN 통신으로 주고받는 패킷의 주요 구성 요소는 Node ID와 메시지의

길이(Length) 그리고 전송되는 메시지(Message)가 됩니다(아래 그림 참조).

Node ID	Length	Message
11bit	0~8	0 ~ 8byte

Node ID로는 최대 11bit를 설정할 수 있는데, 센서의 장치 ID가 여기에 사용됩니다. 마스터에서 장치(센서)로 CAN 메시지를 보내거나 장치가 마스터로 회신 할 때, Node ID에는 장치 ID가 동일하게 지정되어야 합니다. Length는 메시지(Message)의 길이를 나타내는데, 메시지의 최대 크기가 8byte 이므로 0부터 8까지의 값이 됩니다. 메시지에는 전송되는 데이터를 담습니다.

다음 그림은 메시지(Message)의 기본 구조를 나타냅니다. 메시지는 최대 8byte 크기를 가질 수 있으며, 1byte의 Command와 2byte의 Index, 1byte의 Sub-index와 나머지는 상황에 맞는 오브젝트의 값(Value)으로 구성되어 있습니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value (INT32형 혹은 FLOAT형 숫자)			

여기서 Command는 표 4-4과 같은 액세스 형식(Access Code)과 오브젝트 형식(Object Type)의 조합으로 1byte를 구성하게 됩니다.

표 4-4 액세스 형식(Access Code)과 오브젝트 형식(Object Type)

구분	코드	내용
Access Code	0x10	오브젝트 쓰기 요청
	0x20	오브젝트 쓰기 요청에 대한 응답
	0x30	오브젝트 읽기 요청
	0x40	오브젝트 읽기 요청에 대한 응답
	0x80	오브젝트 읽기/쓰기 요청에 대한 에러 응답
	0xF0	동기화 오브젝트 전송 (센서 → PC)
Object Type	0x00	INT8 - 8 bit signed integer
	0x04	INT16 - 16 bit signed integer
	0x08	INT32 - 32 bit signed integer
	0x0C	FLOAT - 32 bit IEEE Standard 754 floating-point

Index나 Value와 같이 한 바이트 이상의 데이터가 메시지에 저장될 때는 데이터의 하위 바이트부터 왼쪽에 저장되는 리틀 인디안(Little-Endian)방식을 따릅니다.

4.2.2 오브젝트 읽기 요청

마스터 PC가 센서의 오브젝트를 읽기 위해, 마스터 PC가 센서에 보내는 쿼리 패킷을 구성합니다.

오브젝트의 값을 읽을 때는 표 4-4의 Access Code 0x30과 읽고자 하는 Object Type을 조합해서

Command를 먼저 구성해야 합니다. 읽고자 하는 오브젝트의 형이 INT32 이라면 Command에는 0x38을 사용합니다. Value 영역(5th ~ 8th byte)은 사용하지 않기 때문에 NUL문자(0x00)을 채웁니다. 그리고 Message의 길이는 8bytes가 됩니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	NUL(0x00) x 4			

4.2.3 오브젝트 쓰기 요청

마스터 PC가 센서의 오브젝트에 값을 쓰기 위해, 마스터 PC가 센서에 보내는 쿼리를 구성합니다.

오브젝트에 값을 쓸 때는, 4th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단, Command는 표 4-4의 Access Code에 의해 0x10과 Object Type에 맞는 조합으로 작성되어야 하며, 그 형식에 맞춰 오브젝트의 값(Value)을 작성합니다. Message의 길이는 8bytes가 됩니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value(INT32)			
				Value(FLOAT)			

4.2.4 오브젝트 읽기/쓰기 요청에 대한 성공 응답

센서가 마스터 PC의 오브젝트 읽기/쓰기 요청에 응답하기 위해, 센서가 마스터 PC에 보내는 응답 패킷을 구성합니다.

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 패킷의 구성은 오브젝트 쓰기 요청에서와 같습니다. 단, Command는 표 4-4의 Access Code에 제시된 것처럼 0x20이나 0x40중 하나와 Object Type에 맞는 조합으로 작성됩니다. Message의 길이는 8bytes가 됩니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value(INT32)			
				Value(FLOAT)			

※ 센서는 사용자가 쓴 값이 적용되지 않을 수 있기 때문에 사용자의 오브젝트를 읽는 요청뿐만 아니라 오브젝트에 쓰기 요청에 대해서도 해당 오브젝트의 값을 응답합니다.

4.2.5 오브젝트 읽기/쓰기 요청에 대한 실패 응답

센서가 마스터 PC의 오브젝트 읽기/쓰기 요청에 실패를 알리기 위해, 센서가 마스터 PC에 보내는

실패 응답 패킷을 구성합니다.

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 센서에 요청했을 때, 오류가 발생하는 경우는 다음 그림과 같은 형식의 오류 패킷이 돌아옵니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Error Code	NUL(0x00) x 6					

오류 메시지는 표 4-5에 나타나 있습니다. CAN 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

표 4-5 오브젝트의 읽기/쓰기 과정에서 발생하는 오류

Error Code	Error Message	내용
1	Undefined name	Index나 Name으로 지정한 오브젝트가 존재하지 않음
2	Packet format error	패킷의 구성이 잘못 되었음
3	Object access error	읽기 전용 오브젝트에 쓰거나, 쓰기 전용 오브젝트의 읽기를 시도함
4	Wrong value assignment	쓰기 오브젝트에 범위를 벗어나는 값을 쓰기 시도함, Ex) id=5000

4.2.6 동기화 데이터 전송

동기화 데이터는 CAN 데이터 전송(sc)에 의해 설정된 값에 따라 주기적으로 전송되는 데이터입니다. 동기화 데이터는 오브젝트 읽기와 쓰기 요청에 무관하게 전송됨으로 동기화 데이터 전송을 사용할 때는 통신 프로토콜의 구현에 주의를 기울여야 합니다.

다음은 동기화 데이터를 전송하기 위한 패킷 구조이며, Command는 동기화 데이터임을 나타내는 0xF0를 사용합니다. 그리고 데이터의 종류를 구분하는 1byte 크기의 Index를 가집니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command (0xF0)	Index	Value1 (INT16)		Value2 (INT16)		Value3 (INT16)	

다음 표 4-6은 Index 값에 따른 데이터의 종류를 나타냅니다.

표 4-6 동기화 데이터 종류

Index	Data Type
51(0x33)	x축, y축, z축 가속도 데이터
52(0x34)	x축, y축, z축 각속도 데이터
53(0x35)	오일러각(phi, theta, psi)
54(0x36)	x축, y축, z축 자기 데이터

Index에 설정된 동기화 데이터의 종류에 따라 Value1, Value2, Value3은 각각 x축, y축, z축 가속도나 각속도, 각도 데이터가 됩니다. 가속도나 각속도, 각도, 자기 값은 내부적으로 32bit 실수형(FLOAT) 오브젝트지만 동기화 데이터로 전송될 때는 각각 1000, 10, 100, 10을 곱해 16bit 정수(INT16)로 변환됩니다.

Value1, Value2, Value3를 가속도와 각속도, 각도로 복원할 때는 다음 변환식을 사용하기 바랍니다:

- x축, y축, z축 가속도 = Value1,2,3/1000
- x축, y축, z축 각속도 = Value1,2,3/10
- 오일러각(roll, pitch, yaw) = Value1,2,3/100
- x축, y축, z축 자기 = Value1,2,3/10

4.3 시리얼(RS-232) 텍스트 통신

시리얼(RS-232) 통신에서는 텍스트(text) 기반으로 센서의 오브젝트를 읽고 쓸 수 있도록 합니다. 이는 사용자가 통신을 위한 전용 프로그램을 사용하지 않더라도 Hyperterminal과 같은 유틸리티를 사용하여 센서의 오브젝트들을 쉽게 엑세스 할 수 있도록 합니다.

시리얼 텍스트 기반 패킷에서는 Index를 직접적으로 사용하지 않고 표 3-1의 Name을 사용합니다.

4.3.1 오브젝트 읽기 요청

오브젝트 값을 읽을 때는 텍스트 패킷을 다음과 같이 구성합니다.

Name	CR/LF

Name은 오브젝트의 Index에 해당하는 이름입니다.

CR/LF는 Carriage Return/Line Feed의 약자로 ASCII 코드상 각각 0x0D, 0x0A입니다. 일반적으로는 키보드의 Enter 키에 해당하며 하이퍼터미널과 같은 유틸리티로 PC에서 연결했다면 명령 입력 후 Enter 키를 입력하는 것입니다. (이후 CR/LF와 동일한 의미로 ↲ 기호 사용)

만약 장치 ID의 설정 값을 알고자 할 때는 아래와 같이 입력합니다.

```
Ex) id↲
```

4.3.2 오브젝트 쓰기 요청

오브젝트 값을 쓸 때는 텍스트 패킷을 다음과 같이 구성합니다.

Name	=	Value	CR/LF

Name은 오브젝트의 Index에 해당하는 이름입니다.

만약 장치 ID를 10으로 바꾸고자 할 때 아래와 같이 입력합니다.

```
Ex) id=10↲
```

4.3.3 오브젝트 읽기/쓰기 요청에 대한 성공 응답

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 센서가 응답하는 형식은 다음과 같습니다.

Name	=	Value	CR/LF

응답하는 문장의 끝에는 CR 문자와 LF 문자가 함께 붙습니다.

두 개 이상의 sub-index를 가지는 오브젝트는 다음과 같이 응답합니다.

Name	=	Value1	SPACE	Value2	SPACE	Value3	CR/LF

여기서 Value1, Value2, Value3는 8byte 크기를 가지며, 값을 구성하는 문자열이 8bytes가 안될 때는 앞에 공백(0x20)이 삽입되어 8bytes를 채우게 됩니다. 데이터와 데이터 간에는 공백(SPACE, 0x20)이 추가되어 데이터를 구분합니다.

```
Ex) id↲                              // 장치 ID 요청
    id=1                             // 장치 ID 응답
    ang↲                             // 오일러각 요청
    ang=    7.49   -30.13    54.54 // 오일러각 응답
```

4.3.4 오브젝트 읽기/쓰기 요청에 대한 실패 응답

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 센서에 요청했을 때, 오류가 발생하는 경우는 다음 그림과 같은 형식의 오류 패킷이 돌아옵니다.

"!ERR("	Error Code	"): "	Error Message	CR/LF

오류 메시지는 표 4-5를 참조하기 바랍니다.

```
Ex) aa↲
    !ERR(1): Undefined Command (hit 'h' or 'help' for help)
```

※ 오류 메시지는 패킷의 구성에 관련된 오류로 센서의 오류 상황이나 오작동에 대한 부분이 아

닙니다.

4.3.5 동기화 데이터 전송

시리얼 텍스트 모드에서 동기화에 의해 전송되는 데이터는 다음과 같은 포멧을 가집니다.

i) 3개의 데이터 전송: ('ss=1', 'ss=2', 'ss=4'인 경우)

8 bytes	1 byte	8 bytes	1 byte	8 bytes	1 byte	1 byte
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터	CR (0x0D)	LF (0x0A)

ii) 6개의 데이터 전송: ('ss=3', 'ss=5', 'ss=6'인 경우)

8 bytes	1 byte	8 bytes	1 byte	8 bytes
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터

8 bytes	1 byte	8 bytes	1 byte	8 bytes	1 byte	1 byte
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터	CR (0x0D)	LF (0x0A)

iii) 9개의 데이터 전송: ('ss=7'인 경우)

8 bytes	1 byte	8 bytes	1 byte	8 bytes
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터

8 bytes	1 byte	8 bytes	1 byte	8 bytes
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터

8 bytes	1 byte	8 bytes	1 byte	8 bytes	1 byte	1 byte
x축 데이터	공백 (0x20)	y축 데이터	공백 (0x20)	z축 데이터	CR (0x0D)	LF (0x0A)

데이터를 표시하는 문자열은 소수점 셋째 자리까지 표시되며, 문자열이 8bytes가 안될 때는 앞에 공백(0x20)이 삽입되어 8bytes를 채우게 됩니다. 데이터와 데이터 간에는 공백이 추가되어 데이터를 구분합니다. 그리고 전송 문자열의 끝에는 CR(0x0D)과 LF(0x0A)이 옵니다.

4.4 시리얼(RS-232) 바이너리 통신

센서의 오브젝트들은 시리얼(RS-232) 통신으로도 읽고 쓰기가 가능합니다. 시리얼 통신은 텍스트 기반의 패킷과 함께 바이너리 기반의 패킷 통신을 함께 지원합니다.

이 절은 센서에 존재하는 오브젝트들을 엑세스하기 위한 시리얼 통신의 바이너리 패킷 구조에 관

해 설명합니다.

4.4.1 바이너리 패킷의 기본 구조

바이너리 패킷의 메시지 구조(4th ~ 11th byte)는 CAN 패킷에서 사용되는 메시지 구조와 동일합니다. 패킷 전체 길이는 13byte로 고정됩니다.

1st byte	2nd byte	3rd byte	4th ~ 11th byte	12th byte	13th byte
STX (0x02)	Length (=13)	Device ID	Message	Checksum	ETX (0x03)

상기 그림에서 패킷의 처음과 끝의 STX, ETX는 패킷의 시작과 끝을 의미하는 문자입니다. 그리고 Length는 13으로 고정되어 있습니다.

Checksum은 3rd ~ 11th byte 지점 까지를 바이트 단위로 더한 후 결과에서 1byte만 취한 값입니다. 다음은 C언어로 작성된 Checksum 계산 예제 코드입니다:

```
char Checksum (char *msg, int len)
{
	char cs = 0;
	for (int i=0; i<len; ++i)
		cs += msg[i];
	return cs;
}
```

4th ~ 11th byte 영역의 Message는 CAN 통신에서의 메시지 구조와 동일합니다.

시리얼 바이너리 패킷에서는 CAN 패킷에서와 같이 리틀 인디안(Little-Endian)방식으로 패킷을 구성합니다.

4.4.2 오브젝트 읽기 요청

오브젝트의 값을 읽을 때는, 패킷의 4th ~ 11th byte 지점에 위치하는 Message를 아래와 같이 구성하면 됩니다. Command는 표 4-4의 Access Code 0x30과 읽고자 하는 Object Type을 조합해서 Command를 먼저 구성해야 합니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	NUL(0x00) x 4			

사용되지 않는 8th ~ 11th byte 영역에는 NUL문자(0x00)을 채웁니다.

4.4.3 오브젝트 쓰기 요청

오브젝트에 값을 쓸 때는, 7th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단

Command는 표 4-4의 Access Code에 의해 0x10과 오브젝트 형에 맞는 조합으로 작성되어야 하며, 그 형식에 맞춰 오브젝트의 값(Value)을 작성합니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	Value(INT32)			
				Value(FLOAT)			

Value 영역에는 오브젝트의 타입에 따라 적절한 값을 구성합니다.

4.4.4 오브젝트 읽기/쓰기 요청에 대한 성공 응답

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 패킷의 구성은 오브젝트 쓰기 요청에서와 같습니다. 단, Command는 표 4-4의 Access Code에 제시된 것처럼 0x20이나 0x40중 하나와 Object Type에 맞는 조합으로 작성됩니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	Value(INT32)			
				Value(FLOAT)			

센서는 사용자가 쓴 값이 적용되지 않을 수 있기 때문에 사용자의 오브젝트를 읽는 요청뿐만 아니라 오브젝트에 쓰기 요청에 대해서도 해당 오브젝트의 값을 응답합니다.

4.4.5 오브젝트 읽기/쓰기 요청에 대한 실패 응답

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 센서에 요청했을 때, 오류가 발생하는 경우는 다음 그림과 같은 형식의 오류 패킷이 돌아옵니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Error Code	NUL(0x00) x 6					

오류 메시지는 표 4-5에 나타나 있습니다. 시리얼 바이너리 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

4.4.6 동기화 데이터 전송

다음은 동기화 데이터를 전송하기 위한 패킷 구조이며, Command는 동기화 데이터임을 나타내는 0xF0를 사용합니다. 그리고 데이터의 종류를 구분하는 1byte 크기의 Index를 가집니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command (0xF0)	Index	Value1 (INT16)		Value2 (INT16)		Value3 (INT16)	

Index 값에 따른 데이터의 종류는 표 4-6을 참조하시기 바랍니다.

Index에 설정된 동기화 데이터의 종류에 따라 Value1, Value2, Value3은 각각 x축, y축, z축 가속도나 각속도, 각도, 자기 데이터가 됩니다. 가속도나 각속도, 각도, 자기 값은 내부적으로 32bit 실수형(FLOAT) 오브젝트지만 동기화 데이터로 전송될 때는 각각 1000, 10, 100, 10을 곱해 16bit 정수(INT16)로 변환됩니다.

Value1, Value2, Value3를 가속도와 각속도, 각도로 복원할 때는 다음 변환식을 사용하기 바랍니다:

- x축, y축, z축 가속도 = Value1,2,3/1000
- x축, y축, z축 각속도 = Value1,2,3/10
- 오일러각(roll, pitch,yaw) = Value1,2,3/100
- x축, y축, z축 자기 = Value1,2,3/10

5 AHRS UI 유틸리티

PC 기반의 AHRS UI(User Interface) 유틸리티는 무료로 다운로드 하여 사용 가능합니다. 이 프로그램은 직관적인 GUI를 제공하며, 그래프 및 3D 그래픽으로 MW-AHRS 데이터를 모니터링 하고 운영합니다.

5.1 소프트웨어 다운로드 및 실행

5.1.1 시스템 요구사항

이 유틸리티를 실행하기 위해서는 다음과 같은 PC 환경이 필요합니다:

- Window XP/7/8/10 32bit/64bit OS
- 10MByte의 HDD 여유공간
- 1GByte 이상의 RAM
- 시리얼 COM(RS-232) 포트

AHRS UI 유틸리티는 제품에 동봉되어 있지 않습니다. 이 유틸리티를 다운로드하기 위해서 PC는 인터넷에 연결되어 있어야 합니다.

※ AHRS UI 유틸리티는 RS-232 포트를 통한 연결만 지원하며 CAN을 통한 연결은 사용할 수 없습니다.

5.1.2 다운로드

UI 유틸리티 프로그램은 MW-AHRS 판매페이지에서 다운로드 받을 수 있습니다:

- AHRS v1 판매페이지: http://www.devicemart.co.kr/1310790#review

5.1.3 실행

AHRS UI 유틸리티는 설치 과정이 필요 없습니다. 압축을 푼 폴더 안에 있는 AHRS_UI.exe 파일을 실행하면 됩니다. 그림 5-1에서 이 유틸리티의 첫 실행 화면을 보여주고 있습니다.

이 유틸리티를 올바르게 사용하려면 MW-AHRS가 PC에 COM 포트(RS-232)를 통해 연결되어 있어야 합니다. 그렇지 않으면 유틸리티의 모든 기능이 활성화 되지 않습니다.

※ AHRS UI 유틸리티는 수시로 업데이트될 수 있습니다. 따라서 사용자는 최신 버전을 확인하고 사용하기 바랍니다.

5.2 메인 화면

AHRS UI 프로그램의 메인 화면은 가속도와 각속도, 자기 센서 값을 그래프로 보여주는 부분과 센서의 회전각(roll, pitch, yaw)을 3D 그래픽으로 보여주는 부분으로 구성됩니다.

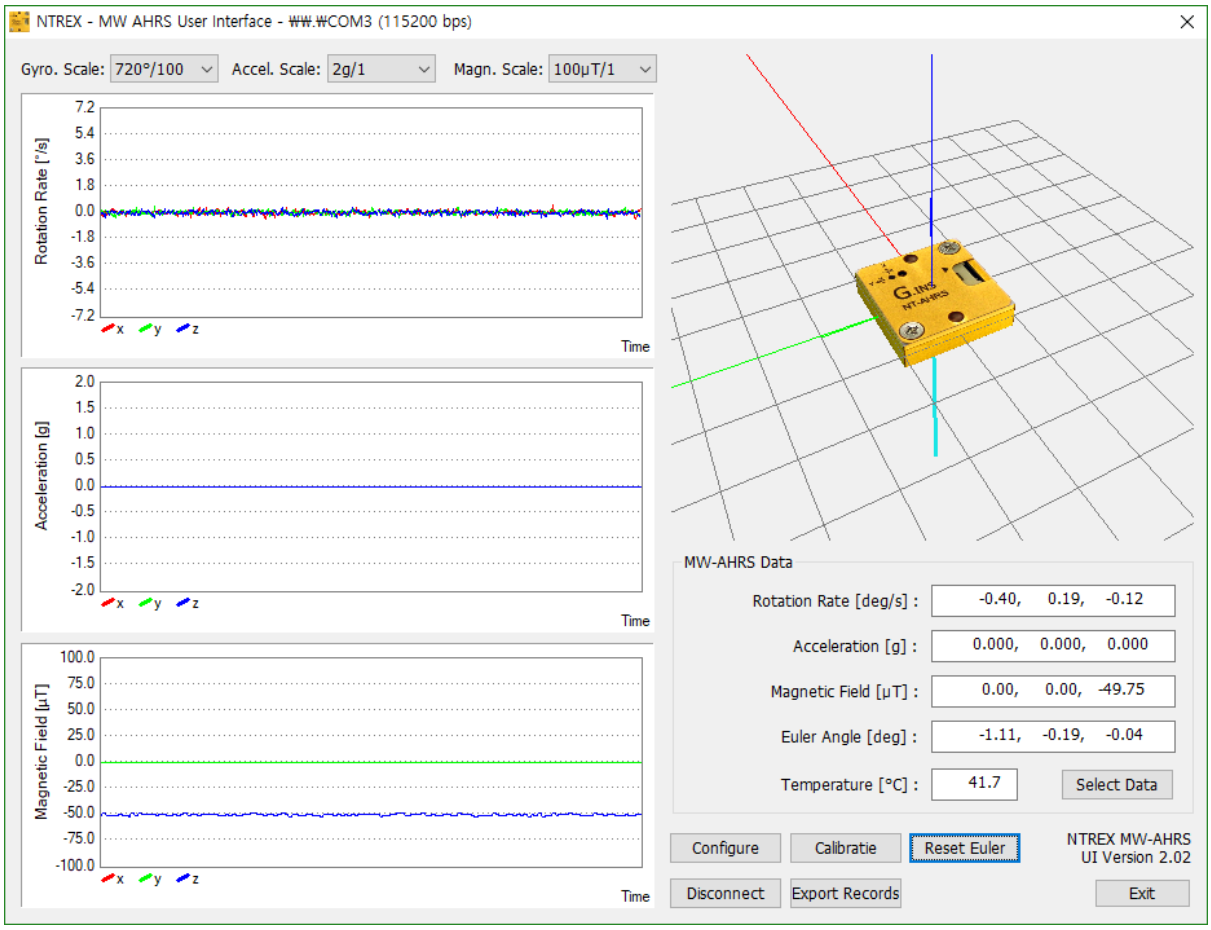
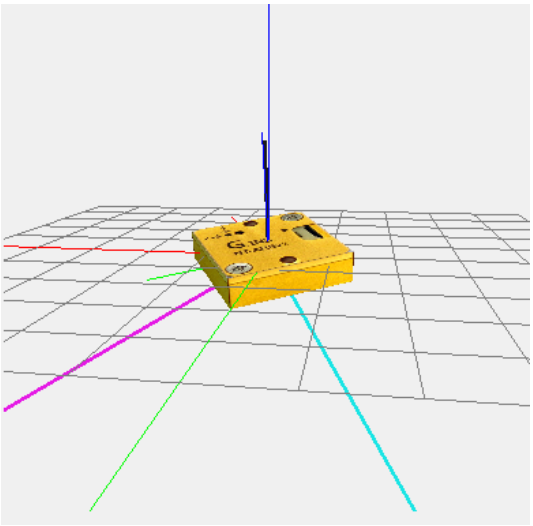


그림 5-1 AHRS-UI의 실행 화면

그리고 몇 개의 버튼으로 구성됩니다. 다음은 각 버튼의 기능에 대한 설명입니다:

- [Connect / Disconnect] – MW-AHRS에 연결하거나 연결을 종료합니다. 버튼을 누를 때마다 이 두 상태는 토글됩니다. 자세한 내용은 '5.3 연결' 절을 참조하기 바랍니다.
- [Configure] – MW-AHRS의 각종 구성 파라미터를 설정하고 명령을 내립니다. 자세한 내용은 '5.4 설정' 절을 참조하기 바랍니다.
- [Calibrate] – 가속도, 자이로, 자기 센서의 캘리브레이션 파라미터를 조정합니다. 자세한 내용은 '5.5 센서 측정값 보정' 절을 참조하기 바랍니다.
- [Reset Euler] – Euler 각도를 0으로 리셋 합니다.
- [Export Records / Stop Recording] – 수신되는 데이터를 파일로 기록하거나 기록을 종료합니다. 버튼을 누를 때마다 이 두 상태는 토글됩니다.
- [Exit] – 프로그램을 종료합니다.



우측 상단의 그래픽 창에서 보여지는 센서에서 검정색, 자주색, 하늘색 굵은 막대를 표시하는데, 각각의 막대는 다음과 같습니다:

- 검정색 굵은 막대 – 가속도의 크기와 방향을 표시합니다. 센서가 정지해 있는 경우 중력에 의해 z축으로 향하게 됩니다.
- 자주색 굵은 막대 – 각속도의 크기와 방향을 표시합니다. 센서가 회전하는 경우 각속도의 크기에 따라 막대의 길이가 결정됩니다.
- 하늘색 굵은 막대 – 지구 자기장의 크기와 방향을 표시합니다.

5.3 연결

메인 화면에서 [Connect] 버튼을 누르면 MW-AHRS과 연결하기 전에 연결 정보를 물어보는 Connection 대화상자가 실행됩니다.

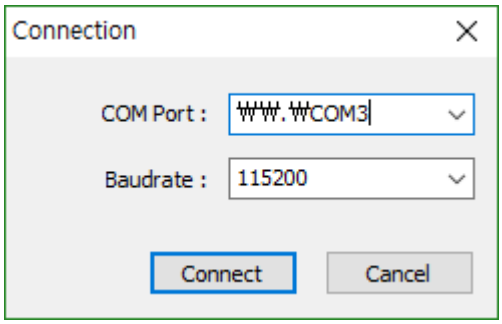


그림 5-2 Connection 대화상자

여기서 COM 포트와 통신속도, 장치 ID를 설정합니다. 그리고 [Connect] 버튼을 누르면 대화상자가 닫히고 MW-AHRS과 연결을 시도합니다. 만일 UI 프로그램이 성공적으로 연결되었다면 [Connect] 버튼은 [Disconnect] 버튼으로 바뀌고 메인 화면에서 수신되는 가속도, 각속도, 각도 데이터를 실시간으로 표시합니다.

※ 제품 출하 시 기본 통신속도는 115200bps 로 설정되어 있습니다.

5.4 설정

메인 화면에서 [Configure] 버튼을 누르면 MW-AHRS의 정보 표시와 파라미터 설정, 명령 전송이 가능한 Configuration 대화상자가 실행됩니다.

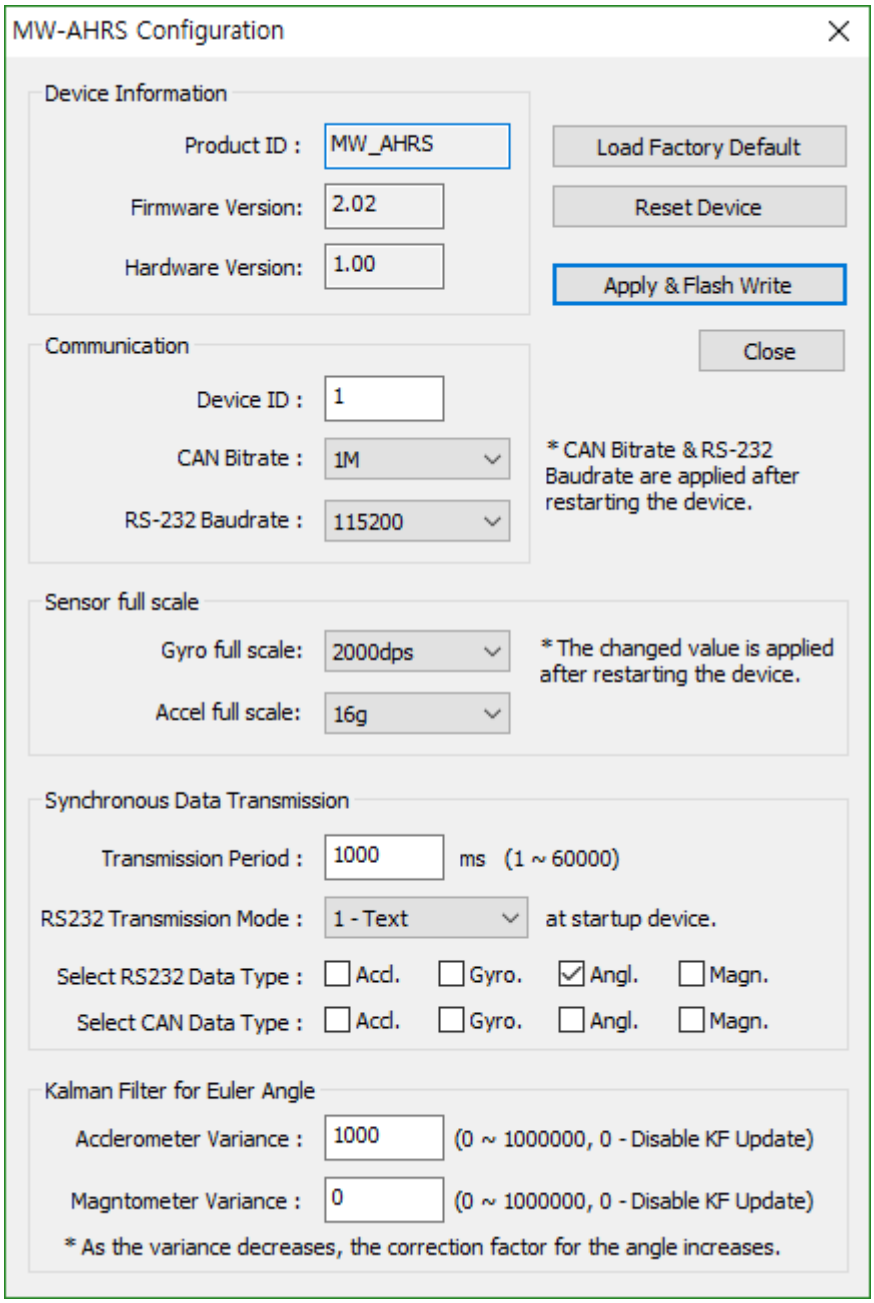


그림 5-3 AHRS Configuration 대화상자

5.4.1 명령 버튼

다음은 MW-AHRS에 명령을 보내기 위한 버튼에 대해 설명합니다:

- [Load Factory Default] – MW-AHRS의 모든 파라미터들을 공장출하시 초기 설정 값으로 되돌립니다.
- [Reset Device] – MW-AHRS를 소프트웨어적으로 재시작 합니다.
- [Apply] – 변경된 파라미터들을 적용하고 Flash Memory에 저장합니다.
- [Close] – Configuration 대화상자를 닫습니다.

Factory Default 값들은 표 3-1의 Default 열을 참조하기 바랍니다.

5.4.2 Device Information 그룹

Device Information 그룹에서는 다음과 같은 정보를 표시합니다:

- Product ID – 제품의 ID를 표시합니다. 여기서는 NT_AHRS이 표시됩니다.
- Firmware Version – 장치의 펌웨어 버전을 표시합니다.
- Hardware Version – 장치의 하드웨어 버전을 표시합니다.

5.4.3 Communication 그룹

Communication 그룹에서는 MW-AHRS 센서의 통신과 관련되는 파라미터를 설정합니다:

- Device ID – 장치 ID를 설정, 이 값은 0부터 255까지 설정가능
- CAN Bitrate – CAN 통신 속도를 설정
- RS-232 Baudrate – 시리얼(RS-232) 통신 속도를 설정

Device ID는 CAN 버스에 연결된 장치를 구분하는데 사용됩니다. 또한 RS-232 바이너리 통신에서도 장치 ID를 모르면 센서와 통신할 수 없게됩니다. 그리고 하나의 통신 선로에 연결된 센서의 Device ID는 모두 달라야 합니다.

제품 초기 설정 값(Factory Default Value)으로 Device ID 파라미터는 1로 설정되어 있습니다. 이 값은 1과 255사이의 값으로 변경될 수 있습니다. 보통 마스터가 되는 PC나 마이크로컨트롤러의 장치 ID로 0을 설정하기 때문에, 센서에서는 0을 사용할 수 없습니다. PC나 마이크로컨트롤러에서 CAN 이나 RS-232 포트를 통해 하나의 센서만 연결하는 경우, 이 값을 변경하지 않고 기본 설정 값을 사용하여도 상관없습니다.

※ CAN Bitrate와 RS-232 Baudrate 설정을 변경한경우, 변경된 내용을 Flash Memory에 저장하고 센서를 재시작 하여야 변경된 값이 적용됩니다.

5.4.4 Sensor Full Scale 그룹

자이로 및 가속도 센서의 측정 범위를 설정합니다. 각 센서에 해당하는 측정 범위를 높게 설정하면 더 큰 값의 데이터를 측정할 수 있지만 데이터의 해상도는 떨어집니다. 반대로 측정 범위를 낮게 설정할수록 데이터의 해상도는 높아지지만 측정 범위를 벗어나는 데이터는 잘려나가게 됩니다.

- Gyro Full Scale – 자이로 센서의 측정 범위를 설정합니다.
- Accel Full Scale – 가속도 센서의 측정 범위를 설정합니다.

5.4.5 Synchronous Data Transmission 그룹

Synchronous Data Transmission 그룹에서는 MW-AHRS 센서가 주기적으로 전송하는 데이터의 종류 및 전송주기를 설정합니다.

- Transmission Period – 가속도, 각속도, 각도, 자기 데이터의 동기 전송 주기를 설정
- RS232 Transmission Mode - 처음 기기를 시작할 때의 RS232 연결 모드 설정
- Select RS232 Data Type - RS232 연결로 전송되는 동기 데이터 종류를 설정
- Select CAN Data Type - CAN 연결로 전송되는 동기 데이터 종류를 설정

동기화 데이터의 Transmission Period는 CAN이나 RS-232 포트로 동기화 데이터(요청이 없어도 주기적으로 전송되는 데이터)를 전송하는 주기를 설정합니다. 이 항목의 설정값은 AHRS-UI에서 데이터를 읽어오는 주기에 영향을 미치지 않습니다. UI에서 데이터를 읽어오고 파일로 저장하는 주기는 33ms로 설정되어 있으며, 변경할수 없습니다. 이 주기는 통신 포트의 속도에 따라 더 느려질 수 있습니다.

5.4.6 Kalman Filter for Euler Angle 그룹

Kalman Filter for Euler Angle 그룹에서 칼만필터의 데이터 융합과 관련된 설정을 합니다.

- Accelerometer Variance – 가속도 센서의 분산 설정
- Magnetometer Variance – 자기 센서의 분산 설정

Accelerometer Variance는 칼만필터에서 사용하는 자이로 센서와 가속도 센서의 측정 데이터에 대한 정밀도를 상대적으로 설정합니다. 자이로 센서의 분산은 1로 고정되어 있습니다. 그리고 가속도 센서의 분산을 1 ~ 1,000,000 사이의 값으로 조정함으로 상대적인 측정 정밀도를 설정하게 됩니다. 여기서 분산이 커지게 되면 그만큼 측정데이터에 노이즈를 많이 포함하고 있다는 의미가 됩니다. 가속도 센서의 분산을 크게하면 중력가속도에 의해 보정되는 각도의 비율이 작아집니다.

반대로 작게하면 비율이 커지게 됩니다.

MW-AHRS센서의 회전은 적은 반면 가속도가 크게 작용하는 곳에 사용하는 경우에는 Acceleration Variance의 값을 크게하여 가속도에 의해 각도가 틀어지는 현상을 줄이거나 0으로 설정하여 각도의 보정을 막아야 합니다. 반면 센서의 회전이 크게 발생하는데 가속도는 크지 않은 경우 Acceleration Variance의 값을 작게 하여 중력가속도에 의해 각도가 더 높은 비율로 업데이트 되도록 하여야 합니다.

5.5 센서 측정값 보정

가속도와 자이로, 자기 센서의 출력 값을 읽으면 센서의 고유 특성과 주변 온도 변화 등에 영향을 받아 편향된 값이 측정됩니다. Kalman filter가 계산하는 각도의 정밀도를 높이기 위해서는 가속도와 각속도 값으로부터 이러한 오차들이 보정되어야 합니다.

MW-AHRS 센서에서 사용하는 보정 파라미터는 다음과 같습니다:

1) 가속도와 각속도, 자기 데이터의 temperature coefficient
2) 가속도와 각속도, 자기 데이터의 bias와 scale
3) 가속도 센서의 중력에 대한 gravity rotation matrix

이 중, 1 항목의 temperature coefficient는 사용자가 보정할 수 없습니다. 이 값은 제품 생산시 계산되어 고정된 값으로 출하됩니다. 하지만 2와 3은 보정(calibration) 명령을 실행하여 사용자가 직접 보정 가능합니다.

여기서는 AHRS-UI를 사용한 센서의 보정 방법과 절차에 대해서 상세하게 설명합니다. 다음 그림 5-4와 그림 5-5, 그림 5-6은 각속도 센서와 가속도 센서, 자기 센서의 출력 값을 보정하는 과정을 보여줍니다.

자이로 센서, 가속도 센서, 자기 센서로부터 읽은 값에는 센서의 특성과 주변온도, 마운트된 방향에 따른 에러를 포함하고 있습니다. 제일 먼저 센서 출력값에서 온도에 의한 바이어스를 보정하게 됩니다. 그리고 센서의 특성에 따른 바이어스와 스케일을 보정합니다.

자이로 센서로부터의 측정 값에서 bias와 scale을 보정하더라도 시간이 지남에 따라 미소하게 드리프트가 발생할 수 있습니다. 이는 HDR 알고리즘으로 보상하게 됩니다.

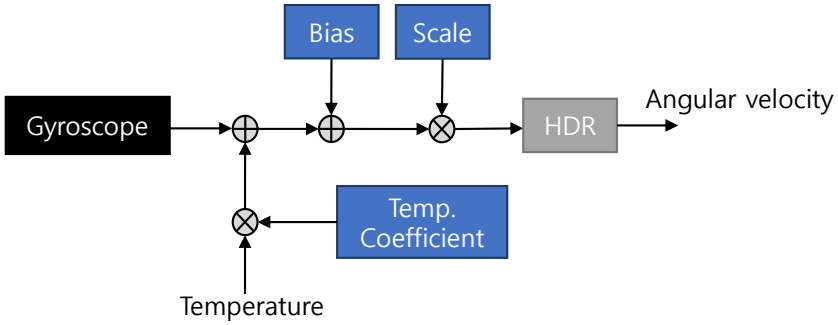


그림 5-4 각속도의 bias와 scale 보정 과정

가속도 센서가 PCB에 기울어진 상태로 마운트 되거나 PCB가 금속 프레임에 기울어진 상태로 마운트 되는 경우, MW-AHRS 센서가 수평을 유지하더라도 Kalman filter가 계산한 오일러 각은 0°가 되지 않을 수 있습니다. 이는 중력가속도의 방향이 약간 기울어진 상태에서 각도를 보정하기 때문입니다. 이러한 문제를 방지하기 위해서는 MW-AHRS 센서가 수평이 되었을 때 가속도 벡터가 (0, 0, 1)이 되도록 보정하여야 합니다. 이는 그림 5-5에서 Gravity Vector Rotation Matrix를 가속도 벡터에 곱하여 보정하게 됩니다.

가속도 센서도 자이로 센서와 마찬가지로 시간이 지남에 따라 미소하게 스케일이 변할 수 있습니다. 이는 HSR 알고리즘으로 보상하게 됩니다.

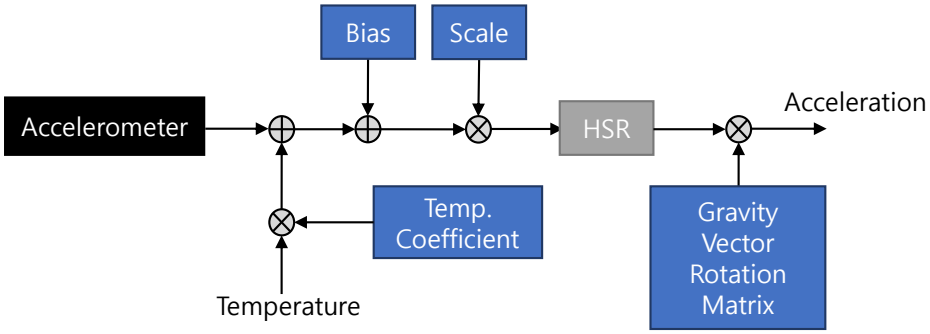


그림 5-5 가속도의 bias와 scale 보정 과정

다음 그림 5-6에서와 같이 자기센서는 단순히 bias와 scale 보정 과정만 거치게 됩니다.

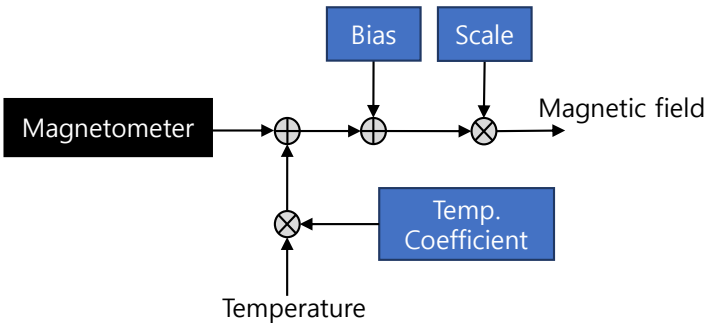


그림 5-6 자기의 bias와 scale 보정 과정

5.5.1 캘리브레이션 대화상자

메인 화면에서 [Calibrate] 버튼을 누르면 Calibration 대화상자가 실행됩니다.

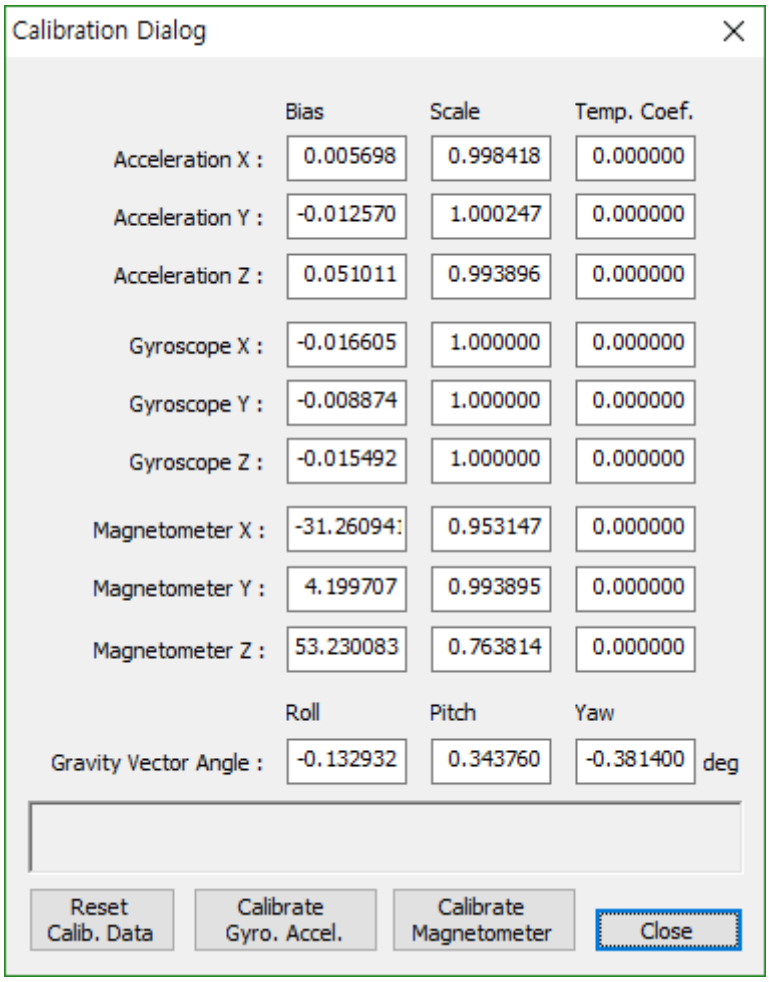


그림 5-7 Calibration 대화상자

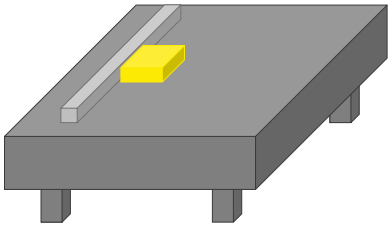
다음은 캘리브레이션 과정을 진행하기 위한 버튼에 대해 설명합니다:

- [Reset Calibration Data] – 센서의 캘리브레이션 데이터를 모두 초기값으로 리셋합니다.
- [Calibrate Gyro. Accel.] – 자이로와 가속도 센서의 바이어스와 스케일을 보정합니다.
- [Calibrate Magnetometer] – 자기 센서의 바이어스와 스케일을 보정합니다.
- [Close] – Calibration 대화상자를 닫습니다.

※ MW-AHRS 센서는 자이로와 가속도 센서의 스케일과 바이어스가 보정되어 출하됩니다. 그러므로 구입한 센서는 보정과정 없이 바로 사용가능합니다. 만일 새로 보정(calibration)해야할 경우에 보정에 대한 자세한 방법과 절차 등을 숙지하신 상태에서 수행하시기 바랍니다.

5.5.2 각속도의 bias

각속도의 바이어스를 계산하기 위해서 먼저 MW-AHRS 센서를 정반과 같이 평평한 곳에 수평을 유지하도록 설치하고 보정이 진행되는 동안 움직이지 않도록 고정합니다. 그리고 Calibration 대화상자의 [Calibrate Gyro. Accel.] 버튼을 누릅니다.



바이어스를 계산하는 과정은 약 2초정도 소요되며, LED가 빠른 주기로 깜박여서 보정 파라미터를 계산하고 있음을 표시합니다. MW-AHRS 센서는 2초동안 천 개의 각속도 센서 출력 데이터를 수집하고 평균을 내어 각속도의 바이어스를 계산합니다. 이 때 센서를 움직이거나 회전하면 절대 안됩니다.

각속도의 바이어스 보정 과정이 완료되면 Calibration 대화상자에서 다음 그림에서와 같이 보정된 값이 붉은색 글씨로 표시됩니다.

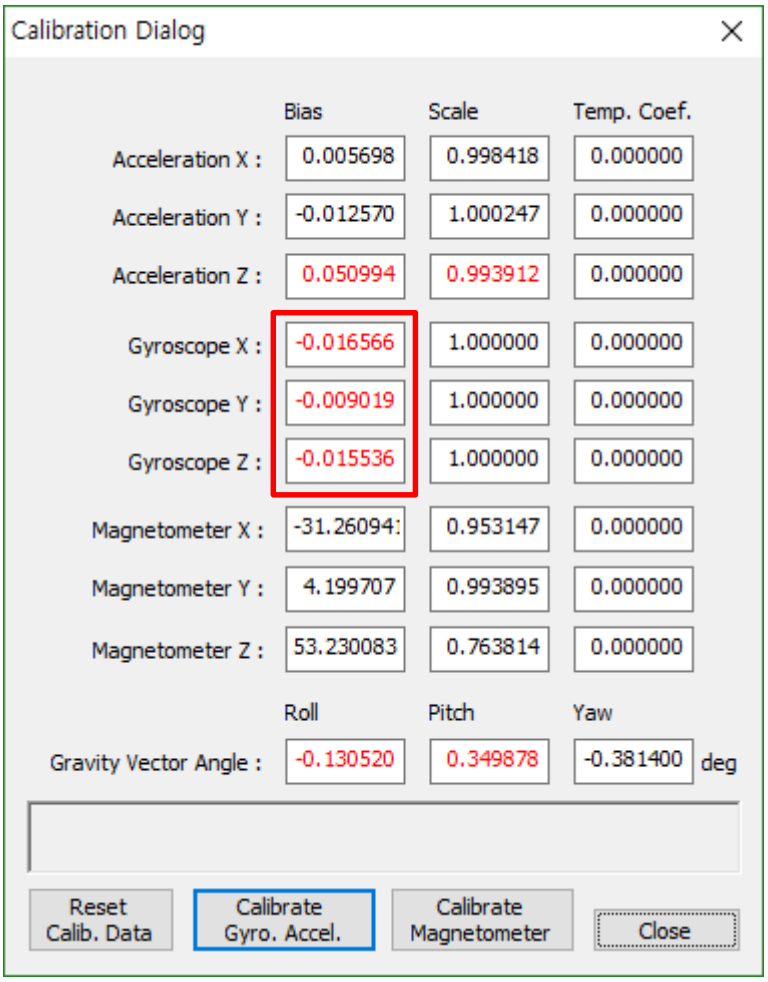
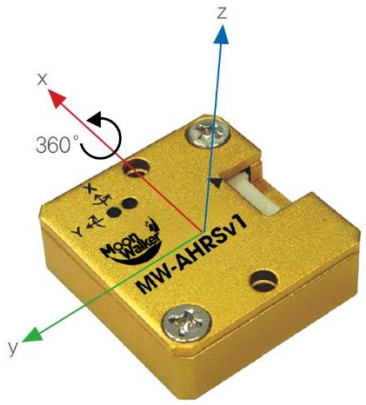


그림 5-8 각속도의 bias 보정 결과 표시

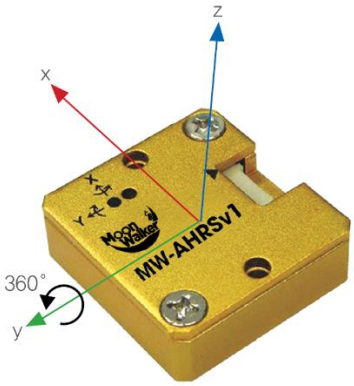
※주의※ 바이어스 보정 명령을 수행하기 전에 MW-AHRS 센서를 정반과 같이 수평이 보장되는 곳에 올려두어야 하고, 바이어스 보정 중일 때는 센서를 움직이거나 센서에 진동이 가해지면 절대 안됩니다.

5.5.3 각속도의 scale

각속도의 스케일은 MW-AHRS 센서를 x, y, z축으로 각각 360° 회전하였을 때 측정되는 각속도의 적분값을 사용하여 계산합니다. 먼저 x축 각속도 스케일을 보정하기 위해 센서를 아래 그림과 같이 z축이 위를 향하도록 정반과 같은 평평한 곳에 둡니다. 그리고 Calibration 대화상자의 [Calibrate Gyro. Accel.] 버튼을 눌러 보정작업을 시작합니다. 이때 센서는 z축으로의 각도 변위에 대한 초기값을 설정합니다.



그 후 30초 이내에 x축을 기준으로 360°만큼 천천히 회전합니다(roll 회전). 회전할 때 회전 축이 틀어지지 않도록 주의해야 하며 회전의 방향은 시계 방향이던 반시계 방향이던 무관합니다. 회전이 끝나고 센서를 다시 원위치에 평평하게 고정한 후 [Calibrate Gyro. Accel.] 버튼을 누릅니다. 그러면 센서는 초기값과 현재값을 비교해 x축(roll 회전)의 스케일을 계산합니다. 이 과정은 30초 이내에 수행되어야 하며, 각속도의 스케일이 성공적으로 계산된 경우에는 LED가 2초동안 켜진상태를 유지하다가 다시 깜박이게 됩니다. 이렇게 x, y, z축 중 x축의 스케일이 보정됩니다.



같은 방법으로 y축의 스케일을 보정합니다. 정반과 같은 평평한 곳에 센서를 두고 [Calibrate Gyro. Accel.] 버튼을 눌러 보정을 시작합니다. 그 후 y축을 기준으로 360° 회전합니다(pitch 회전). 그리고 다시 [Calibrate Gyro. Accel.] 버튼을 누릅니다. 그러면 센서는 초기값과 현재값 간의 차로부터 pitch 회전의 스케일을 계산하게 됩니다. 각속도의 스케일이 성공적으로 계산된 경우에는 LED가 2초동안 켜진상태를 유지하다가 다시 깜박이게 됩니다.

아래의 그림은 각속도의 y축 스케일 보정 과정이 완료되고 나서의 Calibration 대화상자을 보여줍니다. 다음 그림에서와 같이 보정된 값은 붉은색 글씨로 표시됩니다.

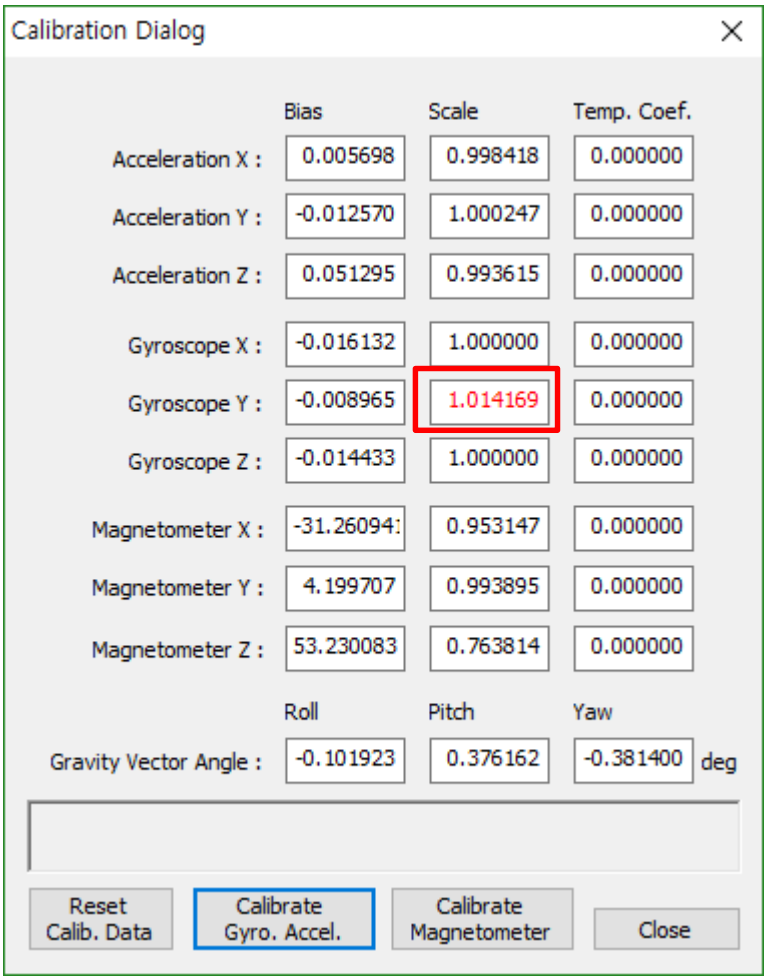
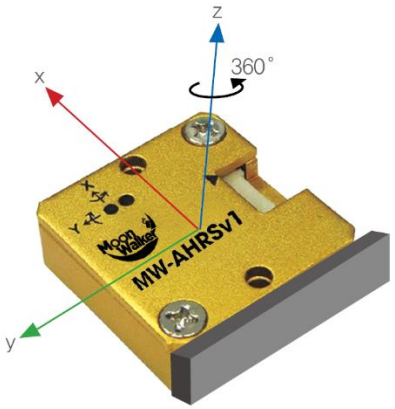


그림 5-9 각속도의 y축 scale 보정 결과 표시



z축을 기준으로 회전할 때는(yaw 회전), 상기 그림에서와 같이 센서를 정렬할 수 있는 고정 막대를 설치하고 진행해야 합니다. 그래야만 센서를 정확히 360° 회전할 수 있게 됩니다. z축에 대한 각속도 센서의 스케일 역시 x, y축에서 수행했던 과정과 동일하게 z축을 기준으로 수행하면 됩니다.

5.5.4 가속도의 bias 와 scale

가속도의 바이어스와 스케일은 지구 표면의 중력이 1g라는 사실을 활용하여 계산됩니다. 먼저 센서를 +z축이 위를 향하도록 정반과 같은 평평한 곳에 고정합니다. 그리고 Calibration 대화상자의 [Calibrate Gyro. Accel.] 버튼을 누릅니다. 센서의 LED는 2초동안 빠른 속도로 깜박입니다. 다시 센서의 -z축이 위를 향하도록 방향을 180°도 회전 하여 고정합니다. 그리고 [Calibrate Gyro. Accel.] 버튼을 누릅니다. 이때 센서는 z축으로 작용하는 중력가속도의 최대값과 최소값을 측정합니다. 그리고 그 측정된 값들로부터 z축에 대한 가속도 센서의 바이어스와 스케일을 계산합니다.

상기와 같은 과정을 +y축과 -y축, 그리고 +x축과 -x축에 대해서도 실시합니다. 이와 같이 가속도의 바이어스와 스케일의 계산을 위해선 총 6번 센서의 방향을 바꿔주며 [Calibrate Gyro. Accel.] 보정작업이 필요합니다.

다음 그림은 가속도의 z축에 대한 bias와 scale 보정 과정이 완료되었을 때의 Calibration 대화상자에 표시되는 내용입니다. 만일 x나 y축에 대하여 수행하였다면 x나 y축에 대한 bias와 scale 값이 붉은 색으로 표시되는 것을 볼 수 있습니다.

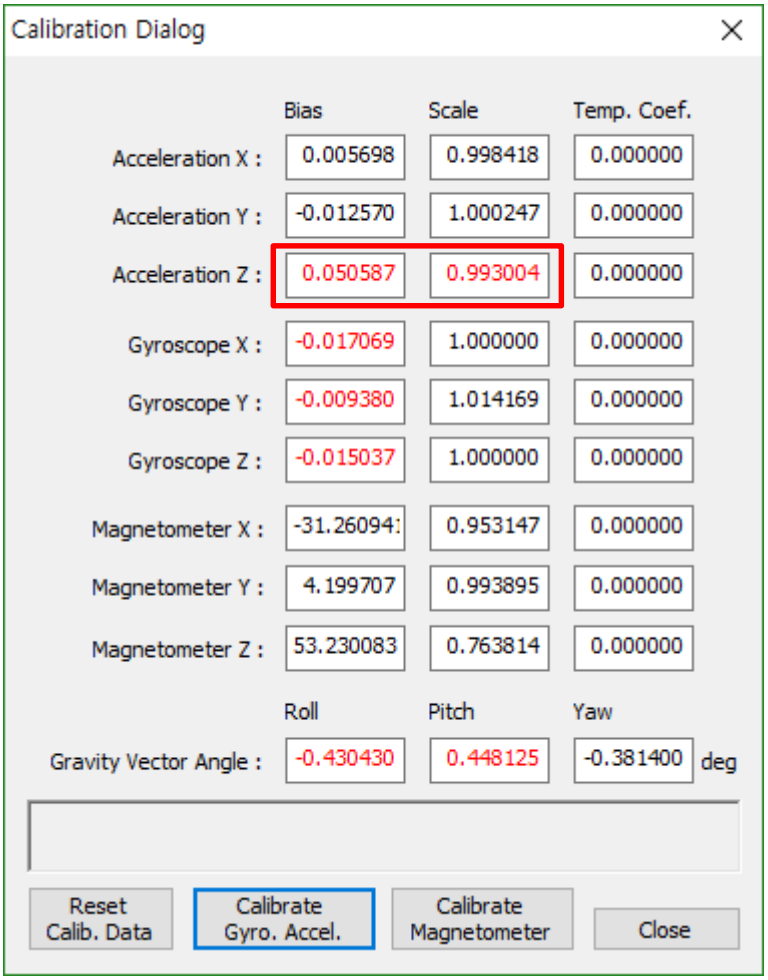


그림 5-10 가속도의 z축 bias, scale 보정 결과 표시

5.5.5 가속도 센서 칩의 기울어짐

가속도 센서 칩의 기울어진 각도는 가속도의 scale 계산과 동시에 수행됩니다. 하지만 가속도의 바이어스와 스케일이 먼저 보정되어야 가속도 센서 칩의 기울어짐을 올바르게 계산할 수 있기 때문에, 5.5.4절의 전 과정이 먼저 한번 수행되어야 합니다. 그리고 5.5.4절의 절차와 동일한 절차를 한 번 더 수행하면 가속도 센서 칩의 기울어진 각도를 계산하여 올바르게 보정하게 됩니다.

다음 그림은 가속도 센서 칩의 기울어짐을 보정하기 위한 roll, pitch, yaw 각의 계산이 완료되었을 때의 Calibration 대화상자에 표시되는 내용입니다.

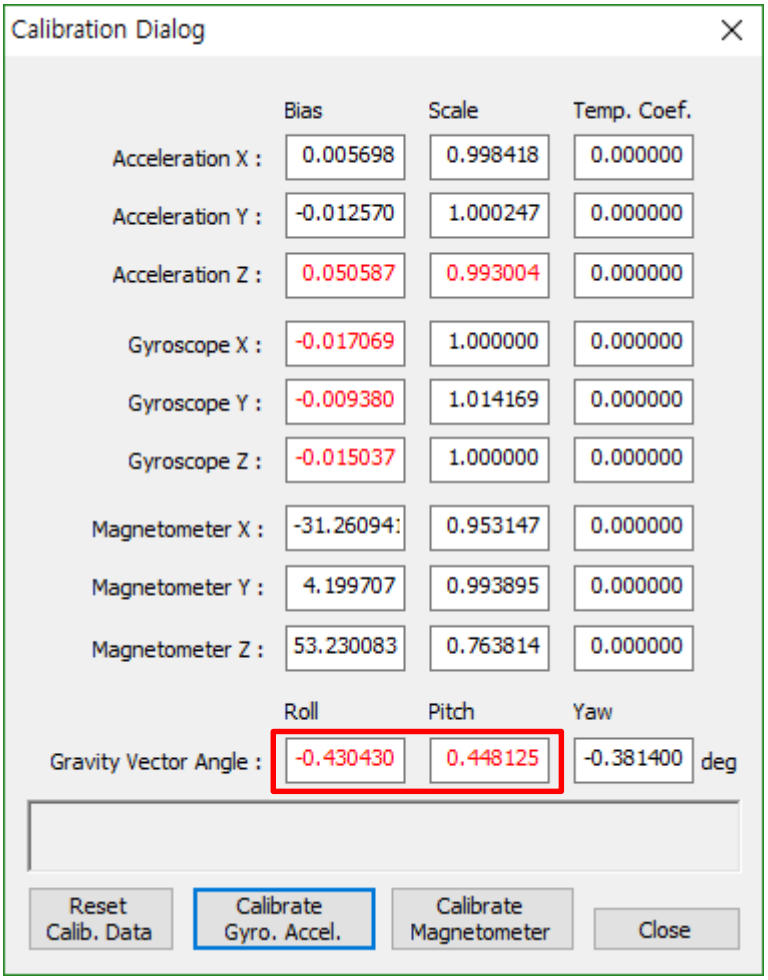
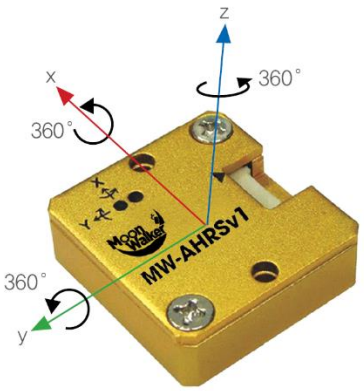


그림 5-11 가속도 센서 칩의 기울어진 각도 계산 결과 표시

5.5.6 자기 센서의 bias 와 scale

전 세계 지구 자기장의 크기는 20 μT에서 80 μT 정도이며 한국의 경우 50 μT입니다. 자기 센서의 보정은 자기 센서의 출력 값이 지구 자기장 크기인 50 μT가 되도록 조정합니다. 자기 센서의 바이어스와 스케일은 지구 자기장의 크기와 방향으로 보정하기 때문에 모터나 자석, 철제 프레임이 있는 장소에서는 올바르게 보정이 되지 않습니다. 되도록 센서를 지구 자기장의 오염이 없는 빈 공간에서 수행하여야 합니다.



센서를 빈 공간에서 손으로 들고 Calibration 대화상자의 [Calibrate Magnetometer] 버튼을 누릅니다. 센서의 LED는 60초동안 빠른 속도로 깜박이면서 지구자기 데이터 수집 중임을 표시합니다. 이때, 모든 방향의 자기장을 수집하여야 함으로 손으로 잡고 있는 센서를 여러 방향으로 향하도록 이리저리 돌려 줍니다. 60초 후 수집된 데이터로부터 각 축의 바이어스와 스케일을 계산합니다.

보정 과정이 성공적으로 끝난 경우 LED는 2초 동안 켜진 상태를 유지하다가 다시 깜박이게 됩니다. 만일 실패하였다면 LED는 2초 동안 꺼진 상태를 유지하다가 다시 깜박이게 됩니다. 자기 센서의 bias와 scale 보정 과정이 완료되면 Calibration 대화상자에서 다음 그림에서와 같이 보정된 값이 붉은색 글씨로 표시됩니다.

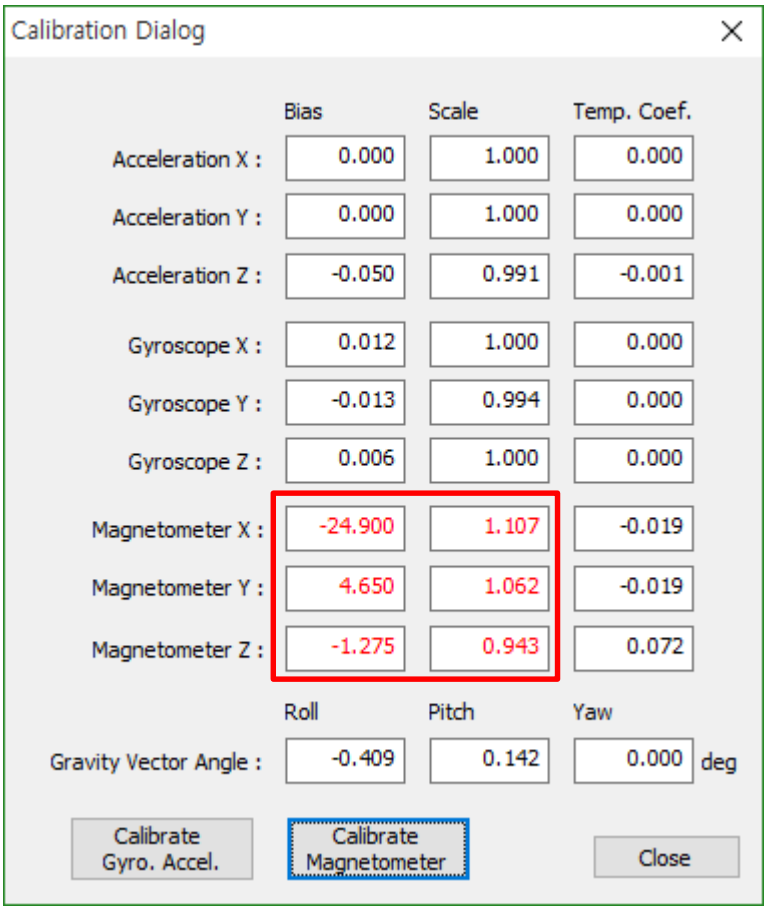


그림 5-12 자기 센서의 bias와 scale 보정 결과 표시

6 회전 변환

3차원 공간에서 물체의 위치와 자세를 표현하기위한 좌표계를 정의하는 방법에는 두 가지가 있습니다. 또한 물체의 자세를 표현하는 방법도 회전 순서에 따라 다양한 조합들이 존재합니다. 이러한 다양한 좌표계/회전 방법들은 여러 공학 분야와 응용분야에서 혼재되어 사용되고 있습니다.

여기서는 MW-AHRS에서 사용하는 좌표계와 회전에 대하여 정의하고 센서에서 측정한 값들을 다른 응용분야의 좌표계로 올바르게 변환할 수 있도록 합니다.

6.1 좌표계와 회전

6.1.1 좌표계

3차원 공간상에서 사용되는 xyz 좌표계는 xy 평면에 대한 z축의 방향에 따라 왼손좌표계(Left-hand Cartesian Coordinates)와 오른손좌표계(Right-handed Cartesian Coordiantes)가 정의됩니다. MW-AHRS는 오른손좌표계를 사용합니다.

Left-hand Cartesian Coordinates Right-hand Cartesian Coordinates

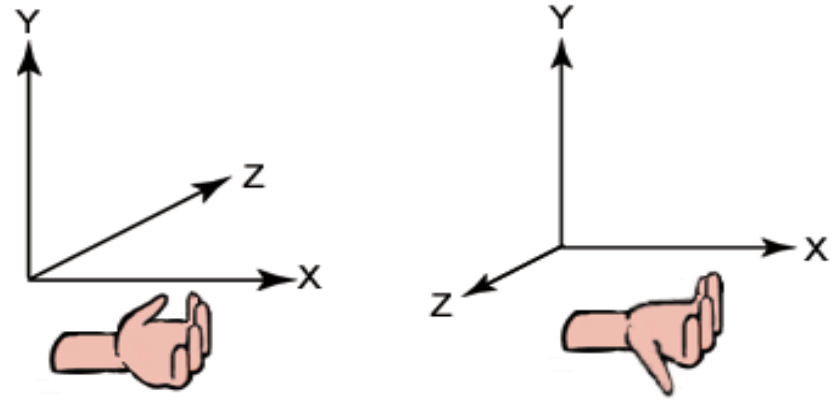


그림 6-1 오른손 좌표계와 왼손 좌표계

로봇이나 차량, 핸드 등의 body를 기반으로 정의되는 로컬 좌표계의 각 축의 방향은 다음과 같습니다: body의 전진 방향을 +x축으로 봅니다. 그리고 body의 오른쪽 방향을 +y축으로 봅니다. MW-AHRS는 오른손좌표계를 사용하기때문에 +z축은 body의 위쪽(하늘, 천정) 방향이 됩니다.

6.1.2 Roll-pitch-yaw 회전

Roll-pitch-yaw는 주로 항공분야에서 비행기의 회전을 기술할 때 사용됩니다. MW-AHRS에서 사용된 roll-pitch-yaw는 다음 그림과 같습니다.

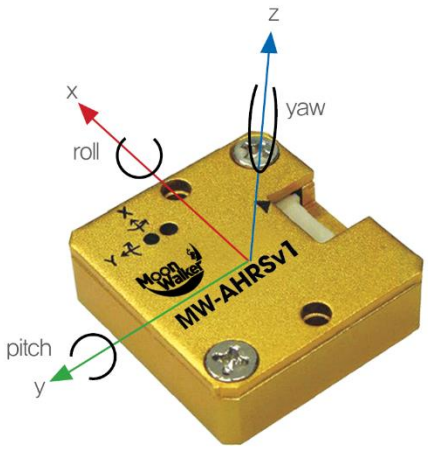


그림 6-2 MW-AHRS의 오른손좌표계와 roll, pitch, yaw 회전

Body의 로컬 좌표계를 기반으로 x축으로의 회전을 roll, y축으로의 회전을 pitch, z축으로의 회전을 yaw로 정의하고, 회전 방향은 각 축의 +에서 내려다 볼 때 반시계 방향입니다.

6.2 오일러각(Euler Angle)과 회전행렬

오일러각 회전변환은 xyz축 중 하나의 축을 선택하여 회전 하는 과정을 3번 반복하는데, 회전 축의 조합이 다양하게 만들어질 수 있습니다. 주로 x-y-z축 혹은 z-y-x축, z-y-z축 순서로 회전하는 것이 일반적입니다. MW-AHRS는 오일러각으로 회전변환 할 때, z-y-x축 순서로 회전합니다.

오일러 각은 ϕ(phi), θ(theta), ψ(psi)로 부르는 세 각의 조합으로 표시하는데, 일반적으로 회전하는 축에 상관없이 첫 번째 회전한 각을 phi, 두 번째 회전한 각을 theta, 세 번째 회전한 각을 psi로 정의합니다. 하지만, 여기서는 x축으로 회전한 각을 phi, y축으로 회전한 각을 theta, z축으로 회전한 각을 psi로 정의합니다.

- 오일러각 회전변환 순서: z축(ψ)→y축(θ)→x축(ϕ) 변환

※주의※ 오일러각 회전변환은 회전 순서에 따라 결과가 모두 다르기때문에 주의해서 사용해야 합니다.

6.2.1 오일러각(z-y-x 회전)을 회전행렬로 변환

관성좌표계에 대한 MW-AHRS 센서좌표계의 회전 결과는 다음과 같이 계산됩니다:

1. 관성좌표계를 x-축(roll)을 중심으로 ϕ만큼 회전한다: $\mathbf{R}_x(\phi)$
2. 관성좌표계를 y-축(pitch)을 중심으로 θ만큼 회전한다: $\mathbf{R}_y(\theta)$
3. 관성좌표계를 z-축(yaw)을 중심으로 ψ만큼 회전한다: $\mathbf{R}_z(\psi)$

MW-AHRS로부터 읽은 센서의 회전 각도는 오일러각(ϕ, θ, ψ)으로 표시되며, 오일러각을 회전행렬(Rotation Matrix)로 변환할 때는 z-y-x축 순서로 회전해야 합니다. 회전행렬은 방향코사인행렬(Direction Cosine Matrix)이라 부르기도 합니다. 다음은 회전행렬을 계산하는 식입니다.

$$\mathbf{R}_{zyx} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$$
$$= \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad \mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.2.2 회전행렬을 오일러각으로 변환

회전행렬 $\mathbf{R}$로부터 오일러각을 계산할 수 있습니다. 다음 식에서 사용되는 r_{ij}는 행렬 $\mathbf{R}$의 i 행과 j 열의 원소입니다.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

다음 오일러각을 계산하는 식들은 z-y-x 회전에 대해 유도된 식들입니다.

i) θ가 $(-\pi/2,\ \pi/2)$일 때:

$$\psi = \operatorname{atan2}\left(r_{21}, r_{11}\right),$$
$$\theta = \operatorname{asin}\left(-r_{31}\right),$$
$$\phi = \operatorname{atan2}\left(r_{32}, r_{33}\right).$$

ii) θ가 $(\pi/2,\ 3\pi/2)$일 때:

$$\psi = \operatorname{atan2}\left(-r_{21}, -r_{11}\right),$$
$$\theta = \operatorname{asin}\left(-r_{31}\right),$$
$$\phi = \operatorname{atan2}\left(-r_{32}, -r_{33}\right).$$

6.3 쿼터니언

쿼터니언(Quaternion; 사원수)은 물체의 회전이나 방향 설정에서 뛰어난 성능을 발휘합니다. 특히, 오일러 각의 연산에서 발생하는 짐벌락(Gimbal Lock)과 같은 각종 문제점들을 극복하기 위해 쿼터니언을 사용합니다. 그리고 9개의 원소를 사용하는 회전행렬에 비해 4개의 원소로 간결하게 표현할 수 있습니다.

6.3.1 기본 성질

쿼터니언은 3개의 벡터 요소와 하나의 스칼라 요소로 구성됩니다.

$$q = \{\eta,\ \boldsymbol{\varepsilon}\} = \{\eta,\ \varepsilon_x,\ \varepsilon_y,\ \varepsilon_z\}$$

여기서 η는 쿼터니언의 스칼라 성분이고, $\boldsymbol{\varepsilon} = (\varepsilon_x,\ \varepsilon_y,\ \varepsilon_z)$ 는 쿼터니언의 벡터 성분입니다.

Unit Quaternion:

쿼터니언의 크기가 1일 때 단위쿼터니언(Unit Quaternion)으로 부릅니다. 단위 쿼터니언은 쿼터니언의 놈(norm)을 나누어 계산합니다.

$$q_U = \frac{q}{\|q\|}$$

Product:

쿼터니언의 곱은 다음과 같이 계산합니다.

$$q_1 q_2 = \{\eta_1\eta_2 - \boldsymbol{\varepsilon}_1^T\boldsymbol{\varepsilon}_2,\ \eta_1\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_1 \times \boldsymbol{\varepsilon}_2\}$$

쿼터니언의 곱은 두 회전행렬 $\mathbf{R}_1,\ \mathbf{R}_2$의 곱 $\mathbf{R}_1\mathbf{R}_2$과 같은 의미입니다. 그리고 쿼터니언의 곱에 대한 역은 성립하지 않습니다. ($q_1 q_2 \neq q_2 q_1$)

Conjugate:

쿼터니언의 conjugate q^*는 다음과 같이 정의됩니다.

$$q^* = \{\eta,\ -\boldsymbol{\varepsilon}\}$$

즉, 쿼터니언의 벡터 부분 $\boldsymbol{\varepsilon}$의 부호를 바꿈으로 구합니다. 또한 쿼터니언의 곱의 conjugate는 다음 특성을 만족합니다.

$(q^*)^* = q$, $(pq)^* = q^* p^*$

Norm:

쿼터니언의 놈(norm)은 다음과 같이 정의됩니다.

$$\|q\| = \sqrt{\eta^2 + \varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2}$$

Inverse:

쿼터니언의 역(inverse) q^{-1}은 다음과 같이 계산합니다.

$$q^{-1} = \frac{q^*}{\|q\|^2}$$

6.3.2 오일러 각을 쿼터니언으로 변환

오일러각 (ϕ, θ, ψ)으로부터 쿼터니언을 다음과 같이 계산합니다. (Z-Y-X 회전)

$$\begin{aligned} q &= \mathrm{Q}_z(\psi)\,\mathrm{Q}_y(\theta)\,\mathrm{Q}_x(\phi) \\ &= \begin{bmatrix} \eta \\ \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} = \begin{bmatrix} \cos(\psi/2)\cos(\theta/2)\cos(\phi/2) + \sin(\psi/2)\sin(\theta/2)\sin(\phi/2) \\ \cos(\psi/2)\cos(\theta/2)\sin(\phi/2) - \sin(\psi/2)\sin(\theta/2)\cos(\phi/2) \\ \cos(\psi/2)\sin(\theta/2)\cos(\phi/2) + \sin(\psi/2)\cos(\theta/2)\sin(\phi/2) \\ \sin(\psi/2)\cos(\theta/2)\cos(\phi/2) - \cos(\psi/2)\sin(\theta/2)\sin(\phi/2) \end{bmatrix} \end{aligned}$$

여기서 각 축별 쿼터니언 변환 함수는 다음과 같습니다.

$$\begin{aligned} &\mathrm{Q}_z(\psi) = \{\cos(\psi/2), 0, 0, \sin(\psi/2)\}, \\ &\mathrm{Q}_y(\theta) = \{\cos(\theta/2), 0, \sin(\theta/2), 0\}, \\ &\mathrm{Q}_x(\phi) = \{\cos(\phi/2), \sin(\phi/2), 0, 0\}. \end{aligned}$$

6.3.3 쿼터니언을 회전행렬로 변환

쿼터니언을 회전행렬로 변환할 때는 다음과 같이 계산합니다.

$$\mathbf{R} = \begin{bmatrix} \eta^2 + \varepsilon_x^2 - \varepsilon_y^2 - \varepsilon_z^2 & 2(\varepsilon_x\varepsilon_y - \eta\varepsilon_z) & 2(\varepsilon_x\varepsilon_z + \eta\varepsilon_y) \\ 2(\varepsilon_x\varepsilon_y + \eta\varepsilon_z) & \eta^2 - \boldsymbol{\varepsilon}_1^2 + \boldsymbol{\varepsilon}_2^2 - \boldsymbol{\varepsilon}_3^2 & 2(\varepsilon_y\varepsilon_z - \eta\varepsilon_x) \\ 2(\varepsilon_x\varepsilon_z - \eta\varepsilon_y) & 2(\varepsilon_y\varepsilon_z + \eta\varepsilon_x) & \eta^2 - \boldsymbol{\varepsilon}_1^2 - \boldsymbol{\varepsilon}_2^2 + \boldsymbol{\varepsilon}_3^2 \end{bmatrix}$$

6.3.4 쿼터니언을 오일러각으로 변환

쿼터니언을 오일러각으로 변환하기 위해서는 회전행렬 $\mathbf{R}$ 로부터 오일러각을 계산하는 방법을 이용하면 됩니다. 다음 식에서 사용되는 r_{ij} 는 행렬 $\mathbf{R}$ 의 i 행과 j 열의 원소입니다. (z-y-x 회전)

$$\mathbf{R}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}=\begin{bmatrix} \eta^2+\varepsilon_x^2-\varepsilon_y^2-\varepsilon_z^2 & 2(\varepsilon_x\varepsilon_y-\eta\varepsilon_z) & 2(\varepsilon_x\varepsilon_z+\eta\varepsilon_y) \\ 2(\varepsilon_x\varepsilon_y+\eta\varepsilon_z) & \eta^2-\boldsymbol{\varepsilon}_1^2+\boldsymbol{\varepsilon}_2^2-\boldsymbol{\varepsilon}_3^2 & 2(\varepsilon_y\varepsilon_z-\eta\varepsilon_x) \\ 2(\varepsilon_x\varepsilon_z-\eta\varepsilon_y) & 2(\varepsilon_y\varepsilon_z+\eta\varepsilon_x) & \eta^2-\boldsymbol{\varepsilon}_1^2-\boldsymbol{\varepsilon}_2^2+\boldsymbol{\varepsilon}_3^2 \end{bmatrix}$$

i) θ가 $(-\pi/2,\ \pi/2)$일 때:

$$\psi=\operatorname{atan2}\left(r_{21},r_{11}\right)=\operatorname{atan2}\left(2(\varepsilon_x\varepsilon_y+\eta\varepsilon_z),\eta^2+\varepsilon_x^2-\varepsilon_y^2-\varepsilon_z^2\right),$$
$$\theta=\operatorname{asin}\left(-r_{31}\right)=\operatorname{asin}\left(-2(\varepsilon_x\varepsilon_z-\eta\varepsilon_y)\right),$$
$$\phi=\operatorname{atan2}\left(r_{32},r_{33}\right)=\operatorname{atan2}\left(2(\varepsilon_y\varepsilon_z+\eta\varepsilon_x),\eta^2-\boldsymbol{\varepsilon}_1^2-\boldsymbol{\varepsilon}_2^2+\boldsymbol{\varepsilon}_3^2\right).$$

ii) θ가 $(\pi/2,\ 3\pi/2)$일 때:

$$\psi=\operatorname{atan2}\left(r_{21},r_{11}\right)=\operatorname{atan2}\left(-2(\varepsilon_x\varepsilon_y+\eta\varepsilon_z),-(\eta^2+\varepsilon_x^2-\varepsilon_y^2-\varepsilon_z^2)\right),$$
$$\theta=\operatorname{asin}\left(-r_{31}\right)=\operatorname{asin}\left(-2(\varepsilon_x\varepsilon_z-\eta\varepsilon_y)\right),$$
$$\phi=\operatorname{atan2}\left(r_{32},r_{33}\right)=\operatorname{atan2}\left(-2(\varepsilon_y\varepsilon_z+\eta\varepsilon_x),-(\eta^2-\boldsymbol{\varepsilon}_1^2-\boldsymbol{\varepsilon}_2^2+\boldsymbol{\varepsilon}_3^2)\right).$$

6.3.5 회전행렬을 쿼터니언으로 변환

회전행렬로부터 쿼터니언은 다음과 같이 계산합니다.

$$q=\{\eta,\ \boldsymbol{\varepsilon}\}$$

$$\eta=\frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}$$

$$\boldsymbol{\varepsilon}=\frac{1}{4\eta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}$$

7 오작동시 체크 사항

상황 1) LED가 켜지지 않습니다.

- 센서와 전원 소스간에 배선이 올바르게 연결되어있는지 확인합니다.
- 센서에 공급되는 전압은 적정 범위인지 확인합니다. (4.5 ~ 10V)

상황 2) LED가 1초 주기로 깜박이지만 RS-232 통신이 안됩니다.

- RS-232 신호선(Rx, Tx, GND)이 올바르게 연결되었는지 확인합니다.
- RS-232 통신 속도(Baudrate)가 올바르게 설정되었는지 확인합니다.
- MCU와 통신하고 있다면 MAX232와 같은 라인드라이버 칩을 사용했는지 확인합니다.

상황 3) LED가 1초 주기로 깜박이지만 CAN 통신이 안됩니다.

- CAN 신호선(Rx, Tx, GND)이 올바르게 연결되었는지 확인합니다.
- CAN 통신 속도(Bitrate)가 올바르게 설정되었는지 확인합니다. CAN bus에 연결된 모든 장치의 통신 속도가 동일하게 설정되어야 합니다.
- CAN ID(Device ID)가 CAN bus에 연결된 다른 장치와 충돌하지 않는지 확인합니다.
- CAN Bus의 종단 저항을 확인합니다. CAN bus에 연결된 모든 장치의 전원을 끄고 CAN_H와 CAN_L 단자의 저항값을 측정하였을 때 60Ω이 되어야합니다.
- MCU와 통신하고 있다면 CAN 드라이버 칩을 사용했는지 확인합니다.

만약 위의 상황이 아닐 경우 A/S 요청 혹은 문의를 주시기 바랍니다.

※주의※ 전원부에 의한 손상일 경우, 무상 A/S가 안될 수도 있습니다.

메모

메뉴얼 버전 V1.01 수정 : 8페이지 내용변경

0° 유지 에러: < 0.1° => 0° 유지 에러: < 0.2°

Euler angle's Resolution: 0.001° => Euler angle's Resolution: 0.01°