

6축 힘/토크 센서

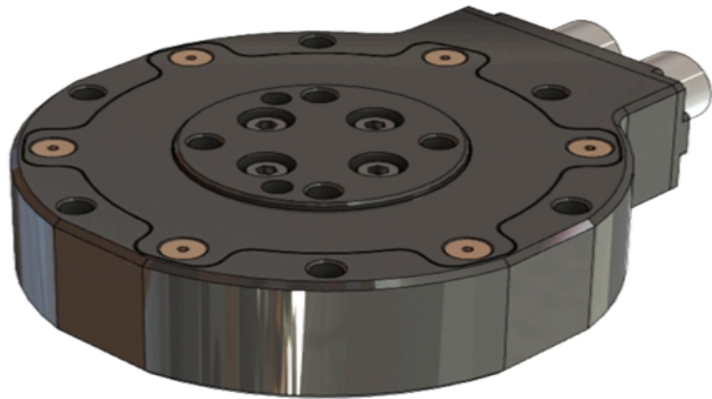
6축 힘/토크 센서 AFT200-D80-C (KOR)

Rev. 2026.09.15

AIDIN
ROBOTICS

Force/Torque sensors user manual

This document is a user manual for AIDIN ROBOTICS
Force/Torque sensors



매뉴얼 안내

에이딘로보틱스 힘/토크 센서를 구매해 주셔서 감사합니다.

이 매뉴얼에는 AIDIN ROBOTICS AFT200-D80 센서를 올바르게 사용하기 위한 필수 정보가 포함되어 있습니다.

소프트웨어 사용 시 반드시 본 매뉴얼을 자세히 확인해주시기 바랍니다.

로봇 시스템이 규정된 조건 밖에서 사용될 경우, 제품의 기본 성능이 발휘되지 않을 수 있음을 알려드립니다.

이 매뉴얼은 힘/토크 센서 사용 중 발생할 수 있는 위험과 그로 인한 결과에 대해 설명하고 있습니다.

안전하고 올바른 사용을 위해 반드시 본 매뉴얼에 명시된 안전 수칙을 준수하시기 바랍니다.

공지

본 매뉴얼은 AIDIN ROBOTICS의 허가 없이 복사, 재생산, 공유를 엄격히 금지합니다.

매뉴얼 또는 제공된 지침에서 오류를 발견하시면 저희에게 반드시 알려주시기 바랍니다.

(연락처: Sales@aidinrobotics.co.kr)

제조사

주식회사 에이딘로보틱스

안전 수칙

 이 기호는 관련 지침을 제대로 따르지 않을 경우 심각한 부상 또는 사망 위험이 있음을 나타냅니다.

 이 기호는 관련 지침을 제대로 따르지 않을 경우 인명 피해나 장비 및 시설의 물리적 손상 위험이 있음을 나타냅니다.

 AIDIN ROBOTICS의 AFT200-D80-C 센서는 CE 인증 "A 등급"을 획득하였습니다.

1. 개요

1.1 6축 힘/토크 센서 AFT200-D80



1.2 주요 특징

- 정전용량 방식의 스마트 6축 힘/토크 센서
- 올인원 센서 (추가 증폭기 불필요)
- 디지털 출력 통신 (CAN 등)
- 간편한 설치 및 데이터 수집
- 그리퍼, 로봇 손, 협동 로봇, 산업용 로봇에 적용 가능

1.3 사양

Index	Unit	Value
Operating voltage	VDC	5
Max. safe excitation voltage	VDC	12
Nominal force range	N	200

Index	Unit	Value
Nominal torque range	Nm	15
Limit force (Fxyz)	N	300
Limit torque	Nm	22.5
Force Resolution	N	0.15
Torque Resolution	Nm	0.015
Force Noise-free resolution (STD)	N	0.4
Torque Noise-free resolution (STD)	Nm	0.025
Maximum sample rate	Hz	1,000
Dimensions	mm	D80 x H21.5
IP rating		IP56
Operating temperature		10-50°C

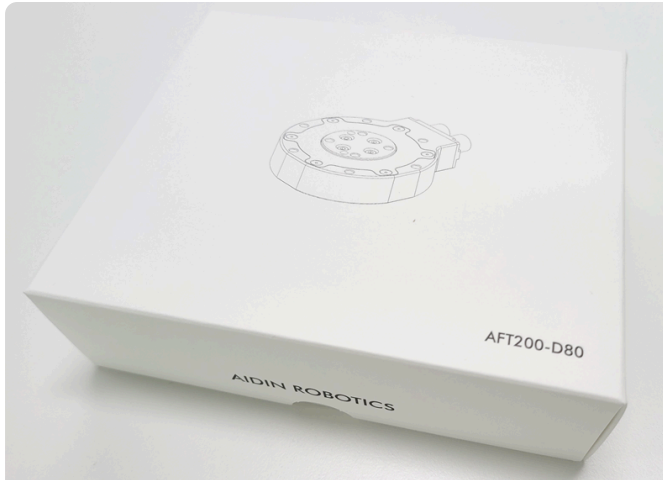
1.4 Flyer

-  [AIDIN ROBOTICS] Mini_AFT200 Leaflet(KO).pdf (파일 준비 중)

2. 설치 가이드

2.1 Basic Components

- AFT200-D80 x 1 EA
- 3m 케이블 x 1 EA



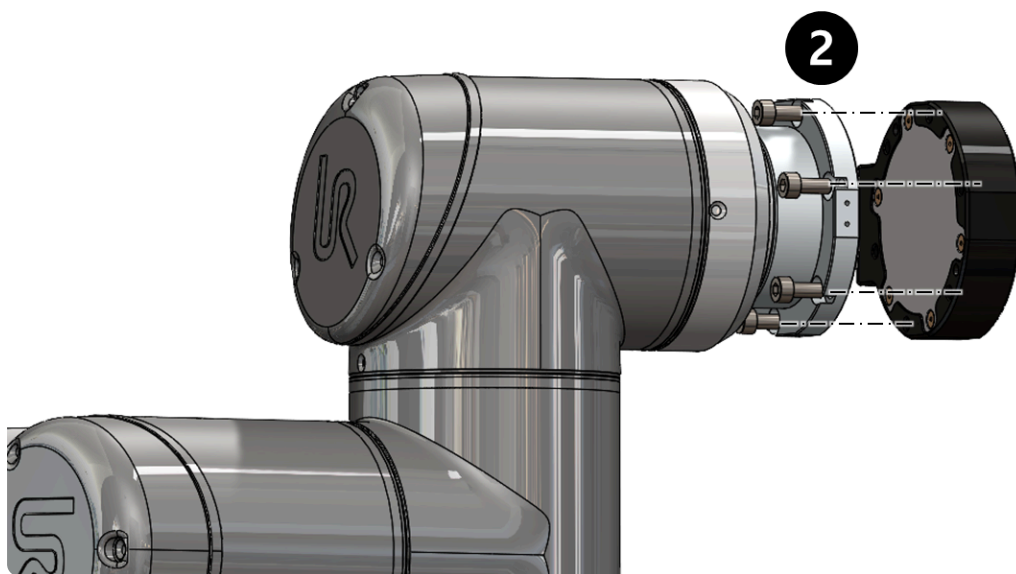
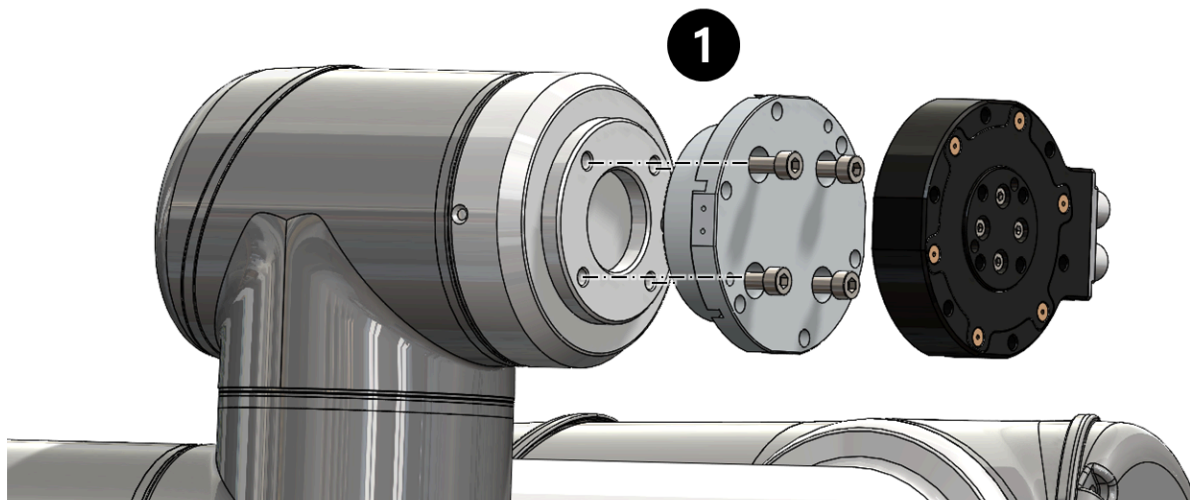
2.2 마운팅

- M5 볼트의 체결 토크는 5.2Nm입니다.
- 내부/외부 볼트를 분해할 경우 성능 보증이 불가능하며, A/S가 제공되지 않습니다.



*Transparent parts are user part examples

- 센서 체결순서



- **케이블 절단 및 과도한 당김에 대한 주의사항**

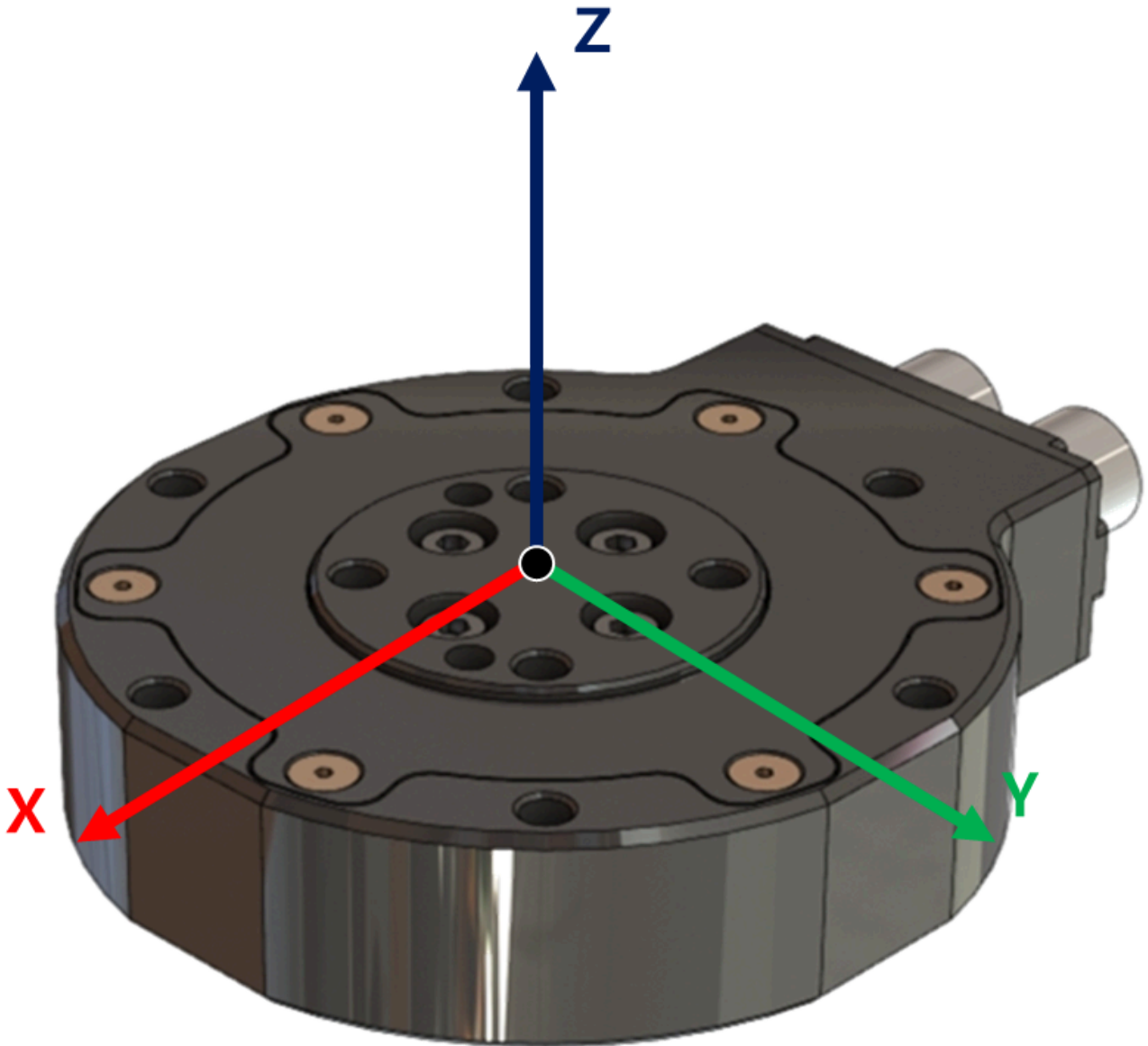
- 센서 라인이 로봇 움직임에 따라 당겨지지 않도록 로봇과 함께 안전하게 고정해 주세요.
 - 케이블 타이 등으로 로봇에 직접 고정하거나, 다른 선들과 함께 묶어서 케이블 타이로 고정하는 것은 금지합니다.
 - 로봇과 고정할 때는 벨크로(찍찍이)를 사용하는 것을 권장하며, 다른 선들과 묶을 경우에도 벨크로를 사용해 고정해 주세요.

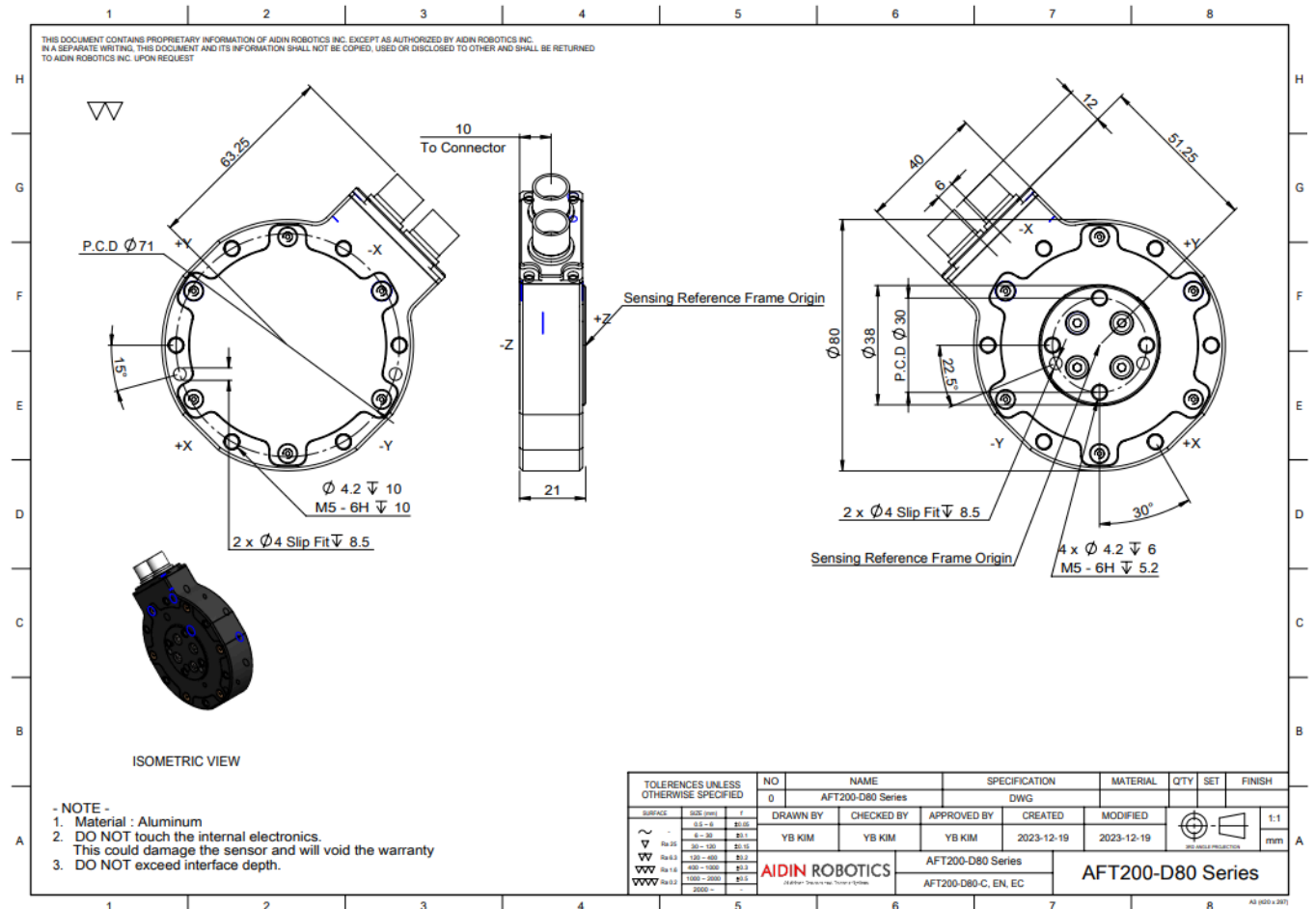
💡 체결 토크 가이드에 따르지 않을 경우 성능 보증이 불가능합니다.

💡 센서 라인이 로봇에 고정되지 않으면, 당기는 힘으로 인해 출력 신호에 노이즈가 발생할 수 있습니다.

💡 자세한 설치 가이드는 다음을 참고해 주세요: 설치 가이드

2.3 축 & 도면





Drawing File :

- [AFT200_D80.pdf](#)

STEP File:

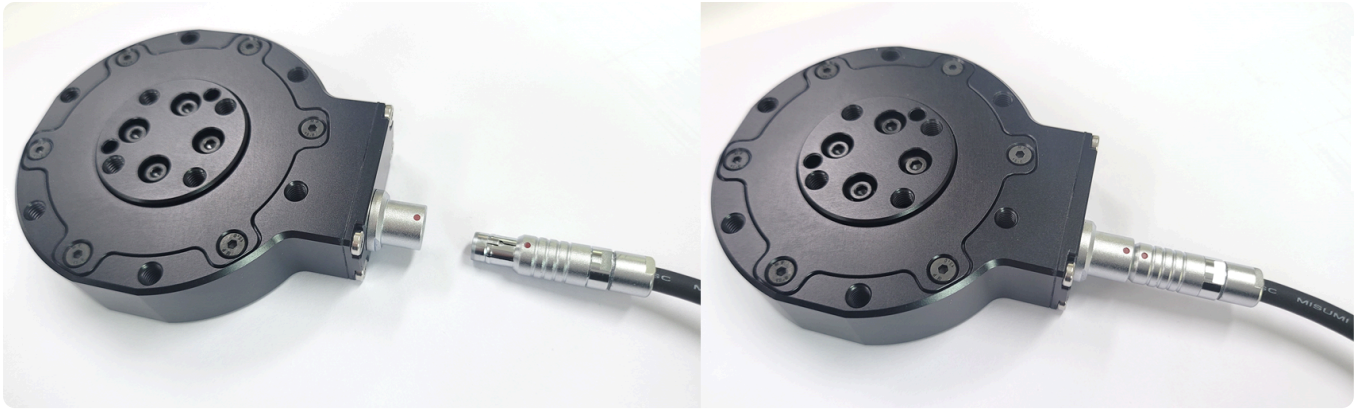
- [AFT200-D80-C_배포용.zip](#)

💡 Port Location Notice

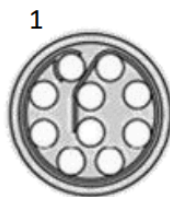
- **EtherCAT** Type model has two ports: **IN** and **OUT**.
- **Ethernet** Type model has one port, located on the **left side**.
- **CAN** Type model has one port, located on the **right side**.

2.4 와이어링

2.4.1 핀 홀



Pin out (CAN)



- 1 – NC
- 2 – 5V
- 3 – GND
- 4 – NC
- 5 – CAN_HIGH
- 6 – CAN_LOW
- 7 – NC
- 8 – NC
- 9 – NC
- 10 – NC

2.4.2 케이블

- 케이블 핀 배선
 - 길이: 3m



- 5V(VCC) – 오렌지색에 검은점
- GND – 회색에 검은점
- CANH – 노란색에 검은점

💡 다른 핀은 사용하지 않습니다.

- PCB 연결 설치 가이드
 - 아래 그림과 같이 선을 삽입한 후 터미널의 볼트를 조여 고정하세요.
 - 각 연결 지점은 PCB 상에 표시되어 있으며, 연결 순서는 다음과 같습니다:



1. 센서 커넥터 (센서와 연결되는 커넥터 부분)
2. 이더넷 커넥터 (PC와 연결되는 커넥터 부분)
3. 센서 커넥터

- a. 상단 2포트 터미널 블록: VCC, GND
- b. 하단 4포트 터미널 블록: VCC, GND, CAN High, CAN Low (선택 사항)

- VCC와 GND는 안정적으로 5V 전원을 공급하는 장치에 연결해야 합니다.
- 9핀 D-sub 커넥터는 eCAN 장치와 연결됩니다.

⊘ 승인을 받은 전원 공급장치만 사용하시는 것을 권장합니다.

2.5 기본 데이터 출력 속도

- 출력 속도는 100Hz에서 1000Hz까지 선택할 수 있습니다.
- 변경 방법은 아래의 "2.6 모드 세팅"을 참고해 주세요.

Table 1.3 - Default Value

Receiver ID	0x01
Data Output Rate	100Hz

2.6 모드 세팅

Table 1.4 - Object index(Hex)

Index	Data field			Description
	Data field [0]	Data field [1]	Data field [2]	
0x102	ID	0x00		Default
		0x01		CAN ID setting mode
		0x02	0x00	Bias set (w/o Temperature compensansation)
			0x01	Bias set (w Temperature compensansation)
		0x03	0x00	Transmitting mode (w/o Temperature compensansation)
			0x01	Transmitting mode (w Temperature compensansation)
		0x05		Sample rate setting mode
	0xFF	0x00		ID confirming mode
		0xFF		Factory Reset mode

💡 센서 신호를 안정적으로 유지하기 위해 약 30분간 작동하는 것을 권장합니다.

- 사용 전 **최소 10분 이상** 센서 데이터를 켜주시기 바랍니다.

💡 데이터를 켜 직후 10분간의 센서 출력 데이터는 다소 변동이 있을 수 있습니다.

2.6.1 모드 세팅

CAN ID 세팅 모드

Table 1.5 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	Current ID	0x01	Setting ID	xx	xx	xx	xx	xx

- 데이터 필드 [0] : 현재 설정 ID (공장 기본값 0x01)
- 데이터 필드 [2] : 전송을 위한 사용자 설정 ID

Table 1.6 - Transmit

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
ID	Setting ID	mode	xx	xx	xx	xx	xx	xx

바이어스 설정 모드

Table 1.7 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	ID	0x02	0x00 or 0x01	xx	xx	xx	xx	xx

- 데이터 필드 [2]: 0x00 — 온도 보상 없이 바이어스 설정 (이 명령은 온도 보상 없이 전송 모드 명령과 함께 사용해야 합니다)
- 데이터 필드 [2]: 0x01 — 온도 보상과 함께 바이어스 설정 (이 명령은 온도 보상과 함께 전송 모드 명령과 함께 사용해야 합니다)

연속 전송 모드

Table 1.8 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	ID	0x03	0x00 or 0x01	xx	xx	xx	xx	xx

- 데이터 필드 [2]: 0x00 — 온도 보상 없이 전송 모드 (이 명령은 온도 보상 없이 바이어스 설정 명령과 함께 사용해야 합니다)
- 데이터 필드 [2]: 0x01 — 온도 보상과 함께 전송 모드 (이 명령은 온도 보상과 함께 바이어스 설정 명령과 함께 사용해야 합니다)

Table 1.9 - Tansmit

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
ID	Force Output [0]	Force Output [1]	Force Output [2]	Force Output [3]	Force Output [4]	Force Output [5]	xx	xx

- Fx 출력 = Force Output[0] × 256 + Force Output[1]
- Fy 출력 = Force Output[2] × 256 + Force Output[3]
- Fz 출력 = Force Output[4] × 256 + Force Output[5]
- 힘 [N] = Force Output ÷ 100 – 300

Table 1.10 - Tansmit

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
ID+1	Torque Output [0]	Torque Output [1]	Torque Output [2]	Torque Output [3]	Torque Output [4]	Torque Output [5]	xx	xx

- Tx 출력 = Torque Output[0] × 256 + Torque Output[1]
- Ty 출력 = Torque Output[2] × 256 + Torque Output[3]
- Tz 출력 = Torque Output[4] × 256 + Torque Output[5]
- Torque[Nm] = Torque Output / 500 – 50

샘플 속도(Sample rate) 설정 모드

Table 1.11 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	ID	0x05	Sample rate parameter[0]	Sample rate parameter[1]	xx	xx	xx	xx

- Data field [2] : Sample rate parameter [0]
- Data field [3] : Sample rate parameter [1]
- Sample rate parameter = Sample rate parameter [0] * 256 + Sample rate parameter [1]
- Sample rate = (1,000,000 / Sample rate parameter)

ex) 1000Hz, Sample rate parameter [0] = 0x03, Sample rate parameter [1] = 0xE8

ex) 500Hz, Sample rate parameter [0] = 0x07, Sample rate parameter [1] = 0xD0

ex) 333Hz, Sample rate parameter [0] = 0x0B, Sample rate parameter [1] = 0xB8

ex) 200Hz, Sample rate parameter [0] = 0x13, Sample rate parameter [1] = 0x88

ex) 100Hz, Sample rate parameter [0] = 0x27, Sample rate parameter [1] = 0x10

ID 확인 모드

Table 1.12 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	0xFF	0x00	xx	xx	xx	xx	xx	xx

Table 1.13 - Tansmit

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
ID	xx	xx	xx	xx	xx	xx	xx	xx

팩토리 리셋 모드

Table 1.14 - Command setting

Index	Data field [0]	Data field [1]	Data field [2]	Data field [3]	Data field [4]	Data field [5]	Data field [6]	Data field [7]
0x102	0xFF	0xFF	xx	xx	xx	xx	xx	xx

2.6.2 디바이스

USB-to-CAN 디바이스

- 다른 CAN 통신 지원 보드를 사용할 수 있으나, 샘플 프로그램을 사용하려면 반드시 IXXAT의 컨버터를 사용해야 합니다.



Pin no. X1	Signal
1	-
2	CAN-Low
3	GND
4	-
5	-
6	-
7	CAN-High
8	-
9	-



- ixxat.com — **Version used in the manual:** 1.01.0281.11001
- Downloads and Documentation for Ixxat Products, **VCI V4** - Windows 11, 10, 8, 7 [Driver, canAnalyser-Mini, Manuals, LabView and other add-ons]
 - Ixxat VCI Setup 4.0.xxx.0.exe
 - 설치 중 **VCI V4**를 확인하고, 설치 완료 후 재부팅하십시오.

EtherNET to CAN 디바이스

- Systembase의 eCAN 모델 사용



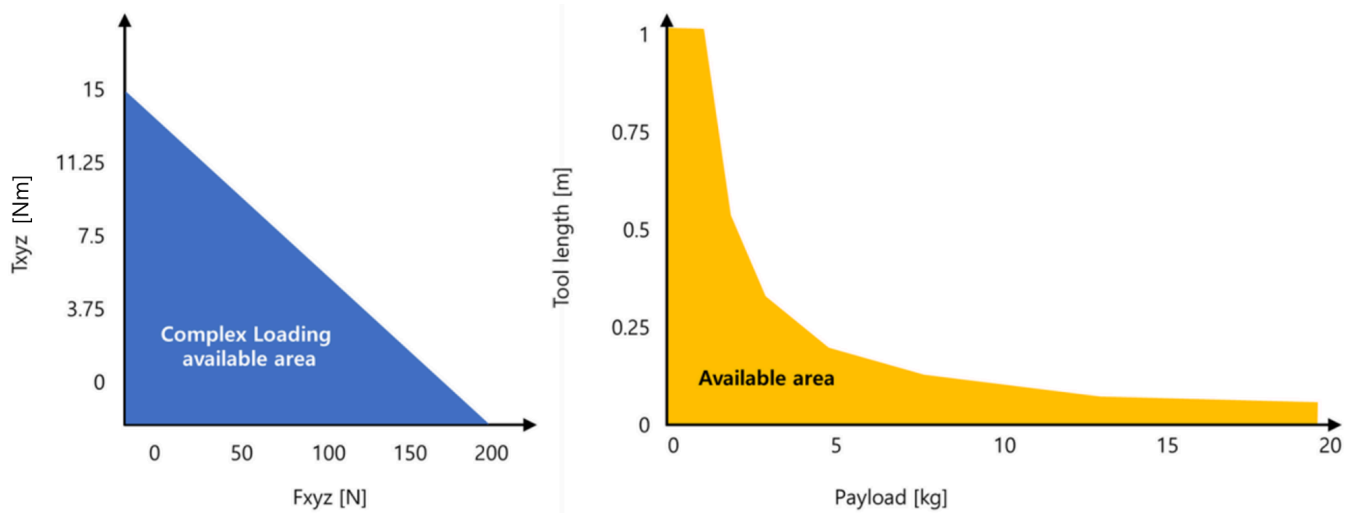
- 아래 사이트를 통해 매뉴얼과 ecANView 프로그램을 다운받아 주십시오.
 - <https://www.sysbas.com/en/downloads/ecan/>

2.7 힘/토크 범위 초과

센서는 단일 축 하중 조건에서 정격 용량까지 동작할 수 있습니다.

정격 용량을 초과하면 측정값이 부정확하고 유효하지 않습니다.

복합 하중의 정격 용량은 다음과 같습니다. 아래 그림은 복합 하중 시나리오를 나타낸 것입니다. 센서는 정상 동작 범위를 벗어난 조건에서는 작동할 수 없습니다.



아래 그래프는 **AFT200-D80-C** 센서 를 사용할 때, 중간~고난이도 수준의 정밀도가 요구되는 응용 분야에서 적용 가능한 **하중 범위와 공구 길이** 를 보여줍니다.

- **Fxy** 축 과 **Tz** 축 에서 사용되는 보정 범위의 총 비율이 **105%를 초과** 하는 경우. 해당 내용은 아래의 **Fxy, Tz** 계산식 을 참조하십시오.

$$\sqrt{F_x^2 + F_y^2} / F_{xy} Range + |T_z| / T_z Range > 105\%$$

- **Fz** 축 과 **Txy** 축 에서 사용되는 보정 범위의 총 비율이 **105%를 초과** 하는 경우. 해당 내용은 아래의 **Fz, Txy** 계산식 을 참조하십시오.

$$\sqrt{T_x^2 + T_y^2} / T_{xy} Range + |F_z| / F_z Range > 105\%$$

3. 소프트웨어

3.1.1 IXXAT의 USB-to-CAN

- 샘플 프로그램 (IXXAT 제공)
 - CanAnalyzer3 Mini

windows → IXXAT folder → IXXAT CanAnalyzer3 Mini

The screenshot displays the IXXAT canAnalyser3 Mini software interface. The main window is titled "(USB-to-CAN V2 compact CAN-1) IXXAT canAnalyser3 Mini". It features a toolbar with various icons for file operations, settings, and data viewing. The interface is divided into several sections:

- Controllers:** Shows the connected device "USB-to-CAN V2 compact HW643084" and "CAN-1".
- Statistics:** Displays "Receive Counter: 9081" and "Error Counter: 0".
- Hardware:** Lists details such as "Controller: Bosch C_CAN", "Clock Rate: 1 MHz", "Serial Number: HW643084", "Revision: 1.5", and "Driver Version: 4.1.212.0".
- Features:** Lists capabilities like "standard and extended remote-frames", "error-frames reception", "send-list", "passive mode", "delayed transmission", "single shot transmission", "high priority messages", "automatic bitrate detection", "FD Frames", "fast timing", "ISO conform frame (ISO 11898-1:2...)", and "64-bit time stamp counter".
- Receive Table:** A table showing incoming CAN messages with columns for No, Time (abs), State, ID (hex), DL, Data (hex), and ASCII. It lists 20 messages with IDs ranging from 0x047 to 0x087.
- Transmit Table:** A table showing outgoing CAN messages with columns for Tx, ID (hex), Description, Ext., RTR, Data (hex), Count, Cycle options (Time (ms), Inc Mode, Byte). It lists 5 messages with IDs 102, 102, 102, 102, and 102.

● 에이딘로보틱스 제공 샘플 프로그램

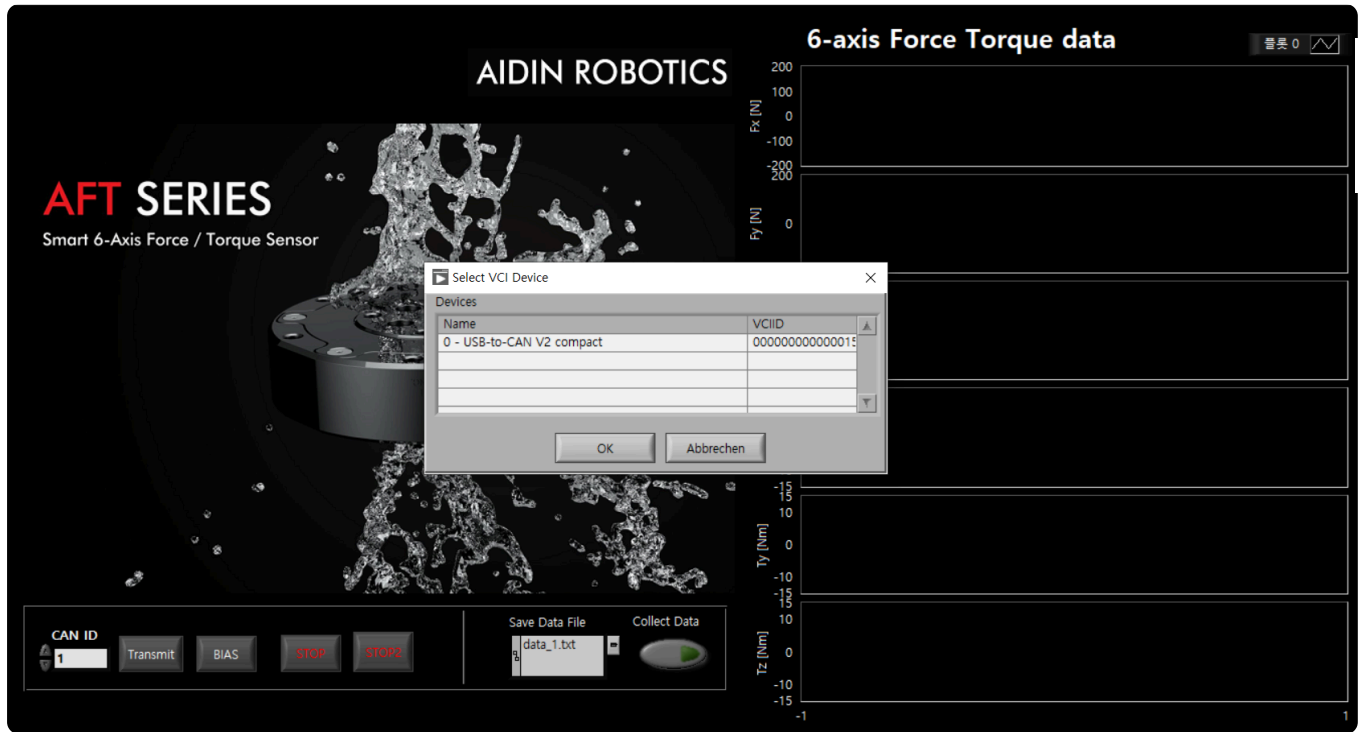
○ 아래 IXXAT CAN VCI V4와 인스톨러를 설치해주세요

○ 샘플 프로그램

- [Installer.zip \(대용량 파일 — 별도 제공\)](#)
- [AFT SAMPLE_v1.1.zip \(대용량 파일 — 별도 제공\)](#)
- [AFT SERIES SAMPLE_v1.2.zip](#)

○ 설치 후 AFT Series Sample을 실행합니다.

● 프로그램 설치시 나타나는 팝업창

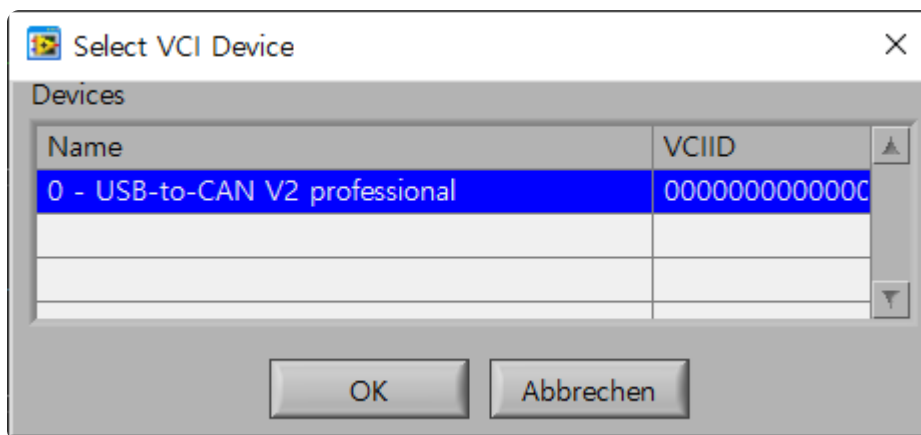


1. CAN 장치 선택

장치를 선택하십시오. 현재 PC에 연결된 장치들이 목록에 표시됩니다.

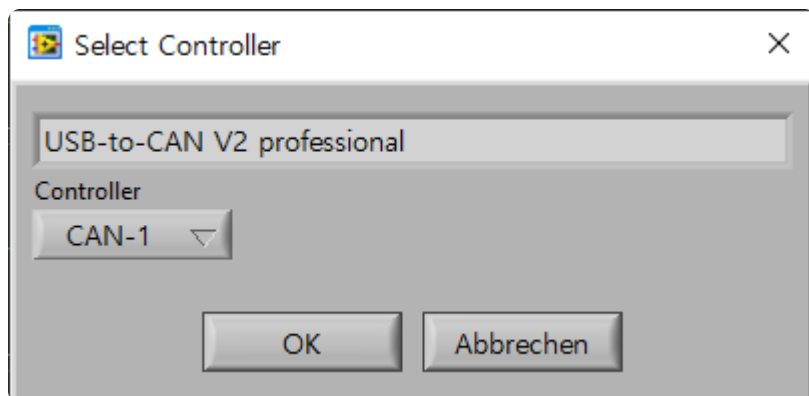
해당 장치를 클릭한 후 **OK** 버튼을 누르십시오.

※ 샘플 프로그램은 **IXXAT**의 **USB-to-CAN 컨버터**를 기반으로 합니다.



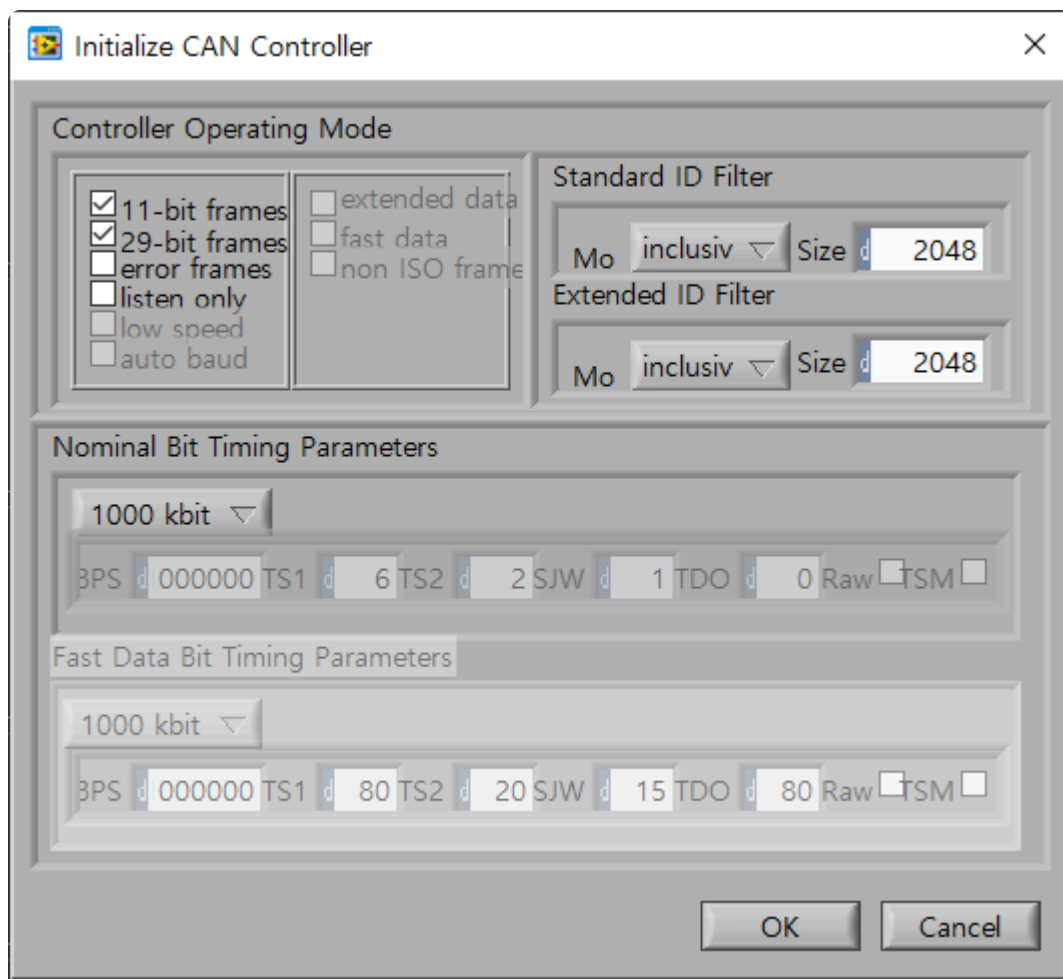
2. CAN 컨트롤러 선택

이전 단계에서 선택한 장치 이름이 표시됩니다. 아래에서 컨트롤러 채널을 선택한 후 **OK** 버튼을 누르십시오.

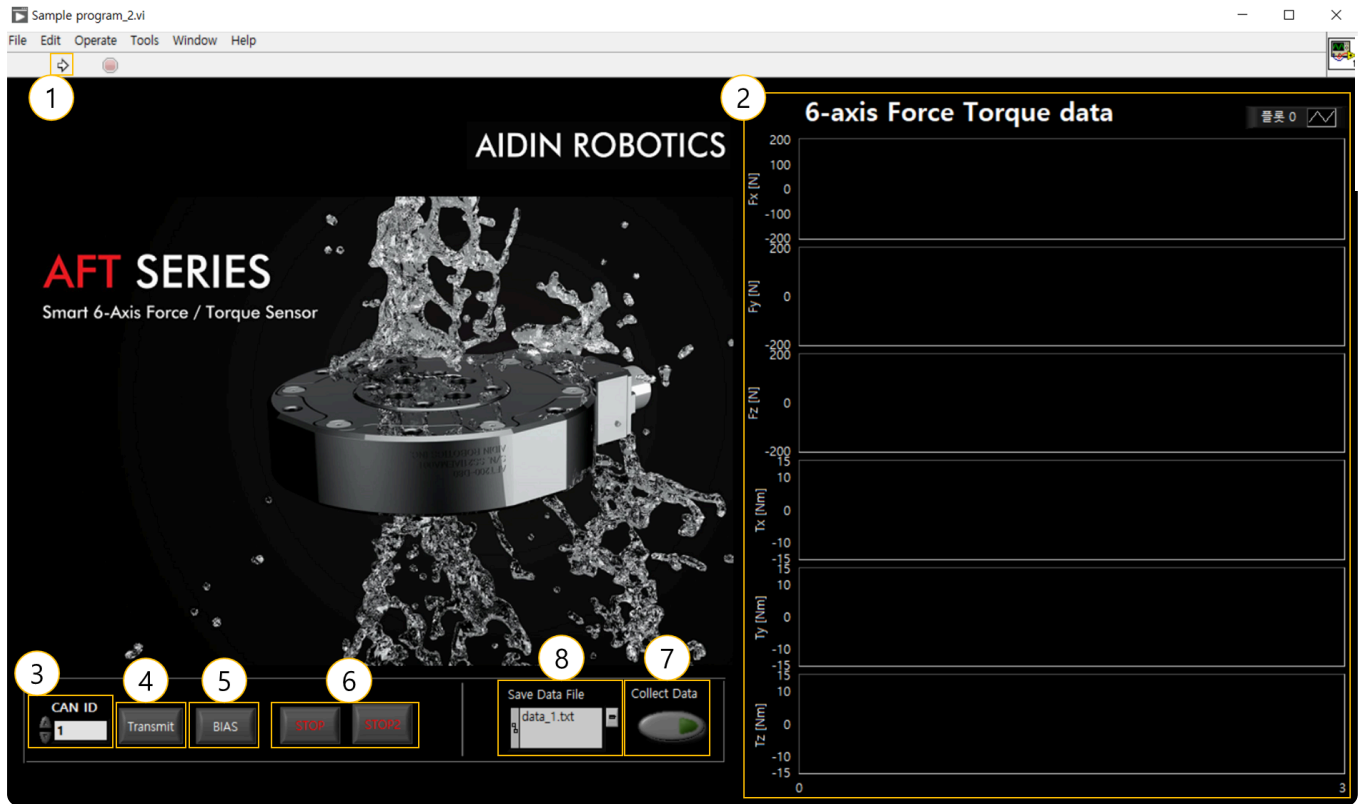


3. CAN 컨트롤러 초기 설정

이전 단계에서 선택한 컨트롤러를 초기화하는 단계입니다. 프레임과 전송 속도(baud rate)를 확인한 후 **OK** 버튼을 누르십시오.



- 샘플 프로그램 화면 예시



① 프로그램 실행 버튼

프로그램 실행 버튼 입니다.

② 그래프 플롯

실시간 6축 힘/토크에 대한 값을 보실 수 있습니다.

③ CAN ID

연결된 센서의 CAN ID를 설정합니다. 여러 개의 ID가 있을 경우 가장 낮은 ID를 입력하세요.

예) 0x01(1), 0x02(2)가 있다면 1을 입력합니다.

④ CAN 전송

6축 힘/토크 값을 출력하기 위해 CAN 메시지를 전송합니다.

예: CAN Msg [0x102, 0x01, 0x03, 0x01]

⑤ 바이어스

6축 힘/토크 데이터를 모두 초기화하여 0으로 설정합니다.

⑥ 정지

프로그램을 종료합니다. STOP 버튼을 두 번 누르면 3초 후 프로그램이 종료됩니다.

⑦ 데이터 수집

힘/토크 데이터 로그를 저장합니다. 데이터 저장 관련 내용은 아래 페이지 설명을 참고하세요.

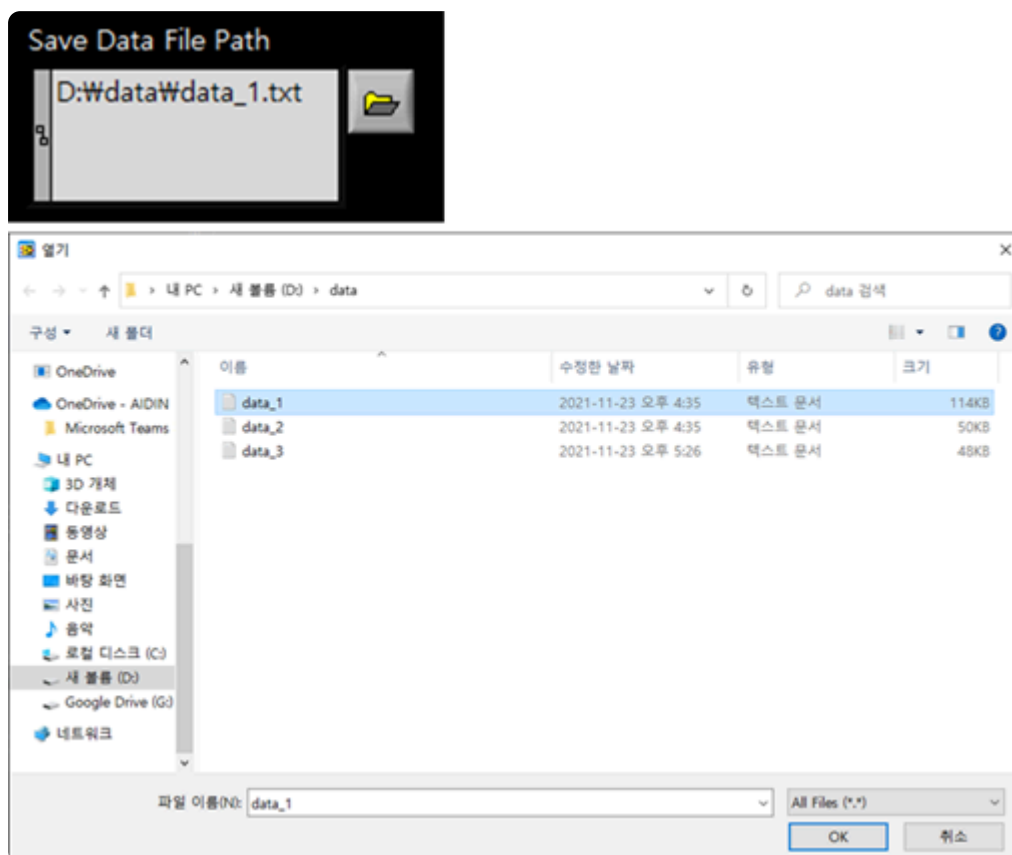
⑧ 데이터 파일 경로 저장

힘/토크 데이터 로그를 저장할 경로를 설정합니다. 데이터 저장 관련 내용은 아래 설명을 참고하세요.

3.1.2 데이터 로그 저장

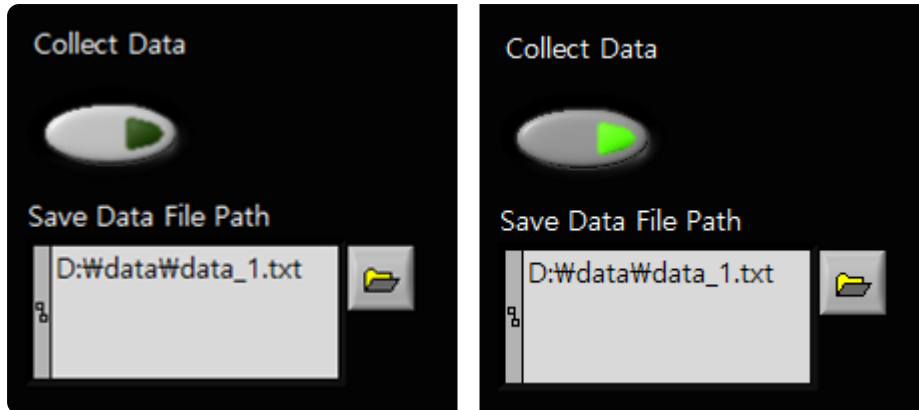
1. 저장 디렉토리 설정

폴더 버튼을 터치하면 팝업 창이 열립니다. 원하는 디렉토리 내의 파일(.txt)을 선택한 후 **OK** 버튼을 눌러 지정하세요. '경로(Path)'에 표시된 경로를 확인합니다. (지정한 경로에 파일이 없으면 자동으로 파일이 생성됩니다.)



2. 데이터 로그 저장하기

'데이터 수집(Collect Data)' 버튼을 눌러 저장을 시작합니다. 저장이 시작되면 버튼이 형광색으로 표시됩니다. 저장을 종료하려면 버튼을 다시 누르세요. 데이터는 파일에 누적되어 저장되므로, 저장할 때마다 새 파일을 지정하는 것을 권장합니다.



3. 데이터 로그 확인하기

지정한 디렉토리에서 파일을 열어 저장된 힘/토크 데이터를 확인할 수 있습니다.

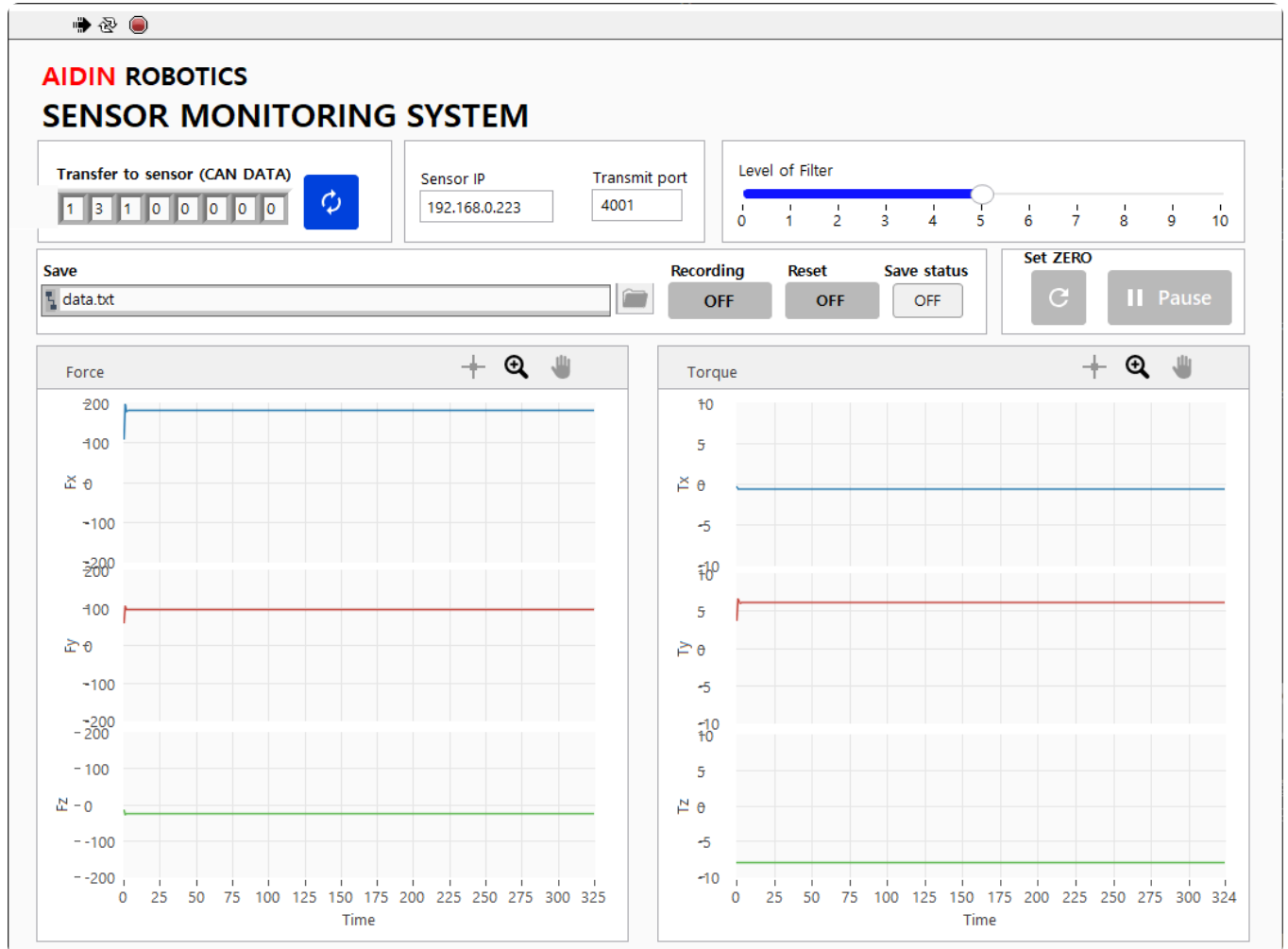
데이터는 Fx, Fy, Fz, Tx, Ty, Tz 순서로 각 열에 저장되어 있습니다.

data_1 - Windows 메모장					
파일(F)	편집(E)	서식(O)	보기(V)	도움말(H)	
-0.031	0.146	0.195	-0.000	0.001	0.000
-0.037	0.040	0.240	-0.001	0.001	0.001
-0.037	0.040	0.240	-0.001	0.001	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.011	0.093	0.116	0.000	0.000	0.001
-0.011	0.093	0.116	0.000	0.000	0.001

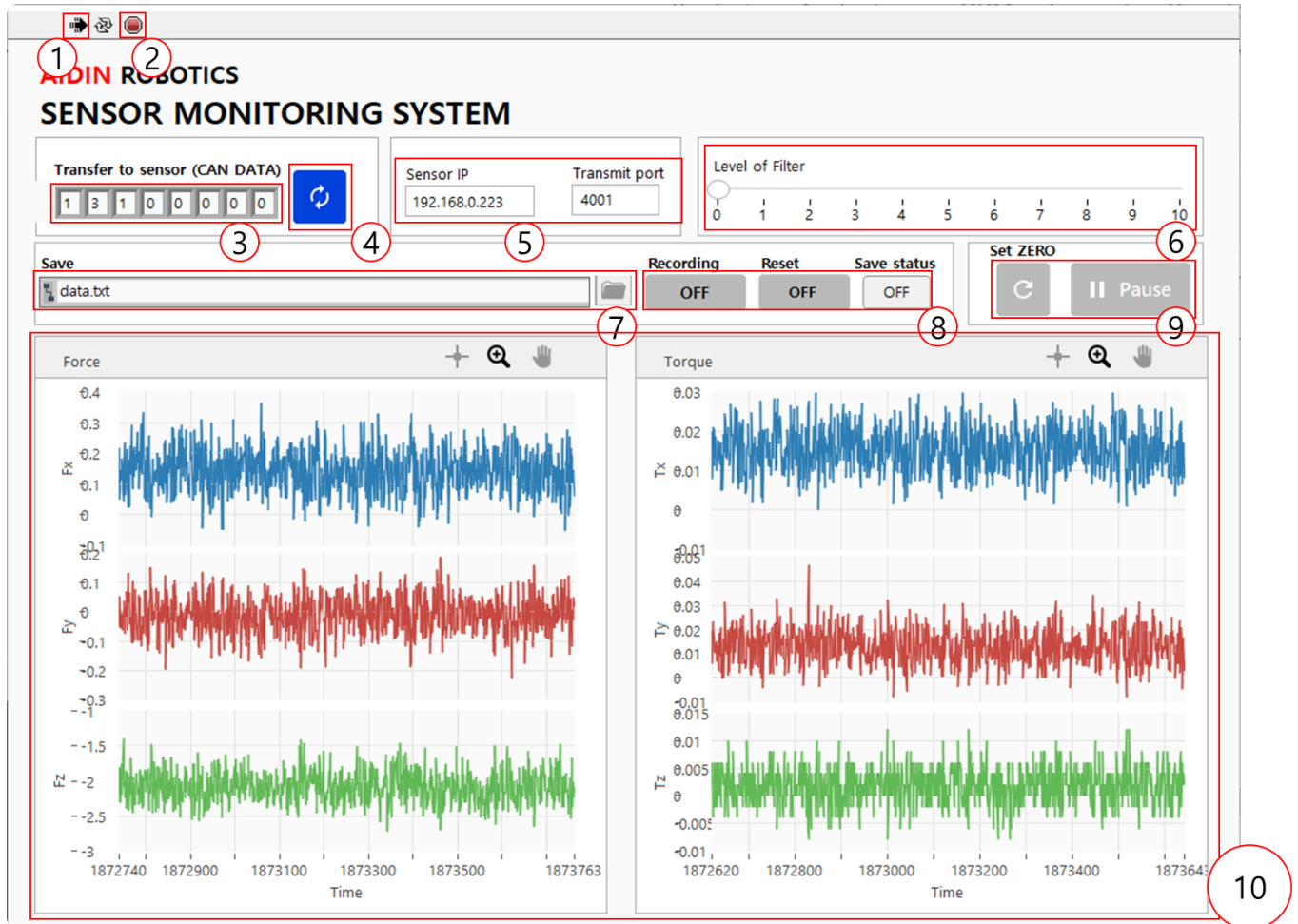
3.2.1 시스템베이스의 EtherNET to CAN

- 에이딘로보틱스 제공 샘플 프로그램
 - AIDIN_Sample_EtherNET.zip (대용량 파일 — 별도 제공)
 - AIDIN_Installer → Volume → setup , Install setup

- Set IP through Systembase's manual or the document below
- AFT200-D80 EtherNET Communication
- Run AIDIN_Sample_EN → AIDIN_Sample_EN.exe
- 샘플 프로그램 작동시 팝업 창 상태



- 샘플 프로그램 인터페이스



① 프로그램 실행

샘플 프로그램을 실행합니다.

② STOP

프로그램을 종료합니다. STOP 버튼을 두 번 누르면 3초 후 프로그램이 종료됩니다.

③ CAN 전송 메시지

CAN Msg [0x102, 0x01, 0x03, 0x01]

④ CAN 전송 버튼

6축 힘/토크 값을 출력하기 위해 CAN 메시지를 전송합니다.

⑤ IP 설정

위 설정에서 일치하는 센서의 IP를 입력합니다.

⑥ 필터링

기본적으로 LPF 필터가 설정되어 있으며, 0에서 9.9까지 설정 가능 (10은 불가).

⑦ 데이터 파일 경로 저장

힘/토크 데이터 로그를 저장할 경로를 설정합니다. 데이터 저장 관련 내용은 아래 페이지 설명을 참고하세요.

⑧ 데이터 수집

힘/토크 데이터 로그를 저장합니다. 데이터 저장 관련 내용은 아래 페이지 설명을 참고하세요.

⑨ BIAS, PAUSE

6축 힘/토크 데이터를 모두 초기화하여 0으로 설정합니다.

설치나 부착물의 영향을 제거하여 원하는 힘 값을 측정할 수 있습니다.

PAUSE 버튼을 클릭하면 센서 데이터 값을 일시 정지할 수 있습니다.

⑩ 그래프

6축 힘/토크 데이터(Fx, Fy, Fz, Tx, Ty, Tz)의 실시간 데이터를 그래프로 확인할 수 있습니다.

3.2.2 데이터로그 저장



1. 저장 디렉토리 설정

폴더 버튼을 터치하면 팝업 창이 열립니다. 원하는 디렉토리 내에서 파일(.txt)을 지정하세요.

'경로(Path)'에 표시된 경로를 확인합니다.

(지정한 경로에 파일이 없으면 자동으로 파일이 생성됩니다.)

2. 데이터 로그 저장하기

'데이터 수집(Collect Data)' 버튼 아래의 버튼을 눌러 저장을 시작합니다.

저장 중일 때 버튼은 형광색으로 표시됩니다.

- 저장을 종료하려면 버튼을 다시 누르세요. 데이터는 파일에 누적 저장되므로, 저장할 때마다 새 파일을 지정하는 것을 권장합니다.
- '리셋(Reset)' 버튼을 클릭하면 저장된 데이터가 초기화되고 다시 저장이 시작됩니다. 리셋 버튼 사용 시 주의해 주세요.



3. 데이터 로그 확인하기

지정한 디렉토리에서 파일을 열어 저장된 힘/토크 데이터를 확인할 수 있습니다.

데이터는 Fx, Fy, Fz, Tx, Ty, Tz 순서로 각 열에 저장되어 있습니다.

The screenshot shows a Windows Notepad window titled 'data_1 - Windows 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The text area contains a table of data with 6 columns and 12 rows. The status bar at the bottom indicates 'Ln 11, Col 38', '100%', 'Windows (CRLF)', and 'UTF-8'.

-0.031	0.146	0.195	-0.000	0.001	0.000
-0.037	0.040	0.240	-0.001	0.001	0.001
-0.037	0.040	0.240	-0.001	0.001	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.011	0.093	0.116	0.000	0.000	0.001
-0.011	0.093	0.116	0.000	0.000	0.001

소스코드

1. 파이썬 코드 – Client

```

#*****
#
# Author: AIDIN ROBOTICS <info@aidinrobotics.com>
# Date: 12-08-2023
#
# -----
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY;
#
#*****/
import struct
import socket

IP_ADDR = '192.168.0.223'
PORT    = 4001

CMD_TYPE_SENSOR_ID      = '01'
SENSOR_TRANSMIT_TYPE_MODE      = '03'
SENSOR_TRANSMIT_TYPE_SET_TEMP  = '01'

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

def main():
    global s
    s.settimeout(2.0)
    s.connect((IP_ADDR, PORT))

    sendData = '04' + '00' + '00' + '01' + '02' + '06' + CMD_TYPE_SENSOR_ID
+ SENSOR_TRANSMIT_TYPE_MODE+SENSOR_TRANSMIT_TYPE_SET_TEMP
    sendData = bytearray.fromhex(sendData)
    s.send(sendData)
    recvData = recvMsg()

    #Initializing
    Fx=0
    Fy=0
    Fz=0
    Tx=0
    Ty=0

```

```

Tz=0

for i in range(10000):
    recvData = recvMsg()
    if recvData[4]==0x01:
        Fx = ((recvData[6]*256+recvData[7])/100.)-300.
        Fy = ((recvData[8]*256+recvData[9])/100.)-300.
        Fz = ((recvData[10]*256+recvData[11])/100.)-300.
        # print("Fx : " + str(round(Fx,2)) + " " + "Fy : " + s
tr(round(Fy,2)) + " " + "Fz : " + str(round(Fz,2)) + " ")
    elif recvData[4]==0x02:
        Tx = ((recvData[6]*256+recvData[7])/500.)-50.
        Ty = ((recvData[8]*256+recvData[9])/500.)-50.
        Tz = ((recvData[10]*256+recvData[11])/500.)-50.
        # print("Tx : " + str(round(Tx,2)) + " " + "Ty : " + s
tr(round(Ty,2)) + " " + "Tz : " + str(round(Tz,2)) + " ")
        print("Fx : " + str(round(Fx,2)) + " " + "Fy : " + str(round(F
y,2)) + " " + "Fz : " + str(round(Fz,2)) + " " + "Tx : " + str(round(Tx,2)) + "
" + "Ty : " + str(round(Ty,2)) + " " + "Tz : " + str(round(Tz,2)) + " ")
        s.close()

def recvMsg():
    recvData = bytearray(s.recv(14))

    printMsg(recvData)

    return recvData

def printMsg(msg):

    dataStr = "DATA: "
    for i in range(6, 14):
        dataStr += str(msg[i]) + " "

    # print(dataStr)

if __name__ == "__main__":
    main()

```

2. 윈도우 코드 – 클라이언트

```

/*****
*
* Author: AIDIN ROBOTICS <info@aidinrobotics.com>
* Date: 12-08-2023
*
* -----
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY;
*
*****/

//Std
#pragma comment(lib, "ws2_32.lib")
#include <iostream>

//Windows Socket
#include <winsock2.h>
#include <ws2tcpip.h>

#define IP_ADDR "192.168.0.223"
#define PORT 4001

#define CMD_TYPE_SENSOR_ID 0x01
#define SENSOR_TRANSMIT_TYPE_MODE 0x03
#define SENSOR_TRANSMIT_TYPE_SET_TEMP 0x01

void printMsg(const char *msg);
void recvMsg(char *msg);
double bytesToDouble(const char *bytes);

SOCKET socket_desc;

int main()
{
    WSADATA wsadata;
    int error = WSAStartup(0x0202, &wsadata);
    if (wsadata.wVersion != 0x0202)
    {
        WSACleanup(); //Clean up Winsock
        return false;
    }
}

```

```

socket_desc = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (socket_desc < 0)
    std::cout << "error opening socket" << std::endl;
    struct sockaddr_in server;

    server.sin_addr.s_addr = inet_addr(IP_ADDR);
server.sin_family = AF_INET;
    server.sin_port = htons(PORT);

std::cout << "Connecting..." << std::endl;

if(error = connect(socket_desc , (SOCKADDR*)&server , sizeof(server))) {
    std::cout << "Error code: " << error << std::endl;
        return 1;
    }

    std::cout << "Connected to the AFT200-D80 Module" << std::endl;

    char recvData[50] = {0};
    char sendData[10] = {0};

    sendData[0] = 0x04;
    sendData[1] = 0x00;
    sendData[2] = 0x00;
    sendData[3] = 0x01;
    sendData[4] = 0x02;
    sendData[5] = 0x06;
    sendData[6] = CMD_TYPE_SENSOR_ID;
    sendData[7] = SENSOR_TRANSMIT_TYPE_MODE;
    sendData[8] = SENSOR_TRANSMIT_TYPE_SET_TEMP;

    send(socket_desc, sendData, 10, 0);
    recvMsg(recvData);

    double Fx, Fy, Fz, Tx, Ty, Tz;
    for(size_t i = 0; i < 10000; ++i) {
        recvMsg(recvData);
        if(recvData[4]==0x01){
            Fx = ((recvData[6]*256+recvData[7])/100.)-300.;
            Fy = ((recvData[8]*256+recvData[9])/100.)-300.;
            Fz = ((recvData[10]*256+recvData[11])/100.)-300.;
        }

        else if(recvData[4]==0x02){
            Tx = ((recvData[6]*256+recvData[7])/500.)-50.;
            Ty = ((recvData[8]*256+recvData[9])/500.)-50.;
            Tz = ((recvData[10]*256+recvData[11])/500.)-50.;
        }
    }

```

```

        std::cout << Fx << " " << Fy << " " << Fz << " " << Tx << " "
<< Ty << " " << Tz << std::endl;
    }

    closesocket(socket_desc);

    return 0;
}

void recvMsg(char *msg) {
    size_t received = recv(socket_desc, msg, 14, 0);

    printMsg(msg);
}

void printMsg(const char *msg)
{
    for(size_t i = 6; i < 14; ++i) {
        //std::cout << (int)msg[i] << " ";
    }

    //std::cout << std::endl << std::endl;
};

```

3. 리눅스 – 클라이언트

```

/*****
*
* Author: AIDIN ROBOTICS <info@aidinrobotics.com>
* Date: 12-08-2023
*
* -----
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY;
*
*****/

//Std
#include <iostream>

//Linux Socket
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

#define IP_ADDR "192.168.0.223"
#define PORT 4001

#define CMD_TYPE_SENSOR_ID 0x01
#define SENSOR_TRANSMIT_TYPE_MODE 0x03
#define SENSOR_TRANSMIT_TYPE_SET_TEMP 0x01

void printMsg(const char *msg);
void recvMsg(char *msg);
double bytesToDouble(const char *bytes);

int socket_desc;

int main()
{
    socket_desc = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in server;

    server.sin_addr.s_addr = inet_addr(IP_ADDR);
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);

```

```

        if(connect(socket_desc , (struct sockaddr *)&server , sizeof(server))
< 0) {
            return 1;
        }

        std::cout << "Connected to the CANToEth Module" << std::endl;

        char recvData[50] = {0};
        char sendData[10] = {0};

        sendData[0] = 0x04;
        sendData[1] = 0x00;
        sendData[2] = 0x00;
        sendData[3] = 0x01;
        sendData[4] = 0x02;
        sendData[5] = 0x06;
        sendData[6] = CMD_TYPE_SENSOR_ID;
        sendData[7] = SENSOR_TRANSMIT_TYPE_MODE;
        sendData[8] = SENSOR_TRANSMIT_TYPE_SET_TEMP;

        send(socket_desc, sendData, 10, 0);
        recvMsg(recvData);

        double Fx, Fy, Fz, Tx, Ty, Tz;
        for(size_t i = 0; i < 10000; ++i) {
            recvMsg(recvData);
            if(recvData[4]==0x01){
                Fx = ((recvData[6]*256+recvData[7])/100.)-300.;
                Fy = ((recvData[8]*256+recvData[9])/100.)-300.;
                Fz = ((recvData[10]*256+recvData[11])/100.)-300.;
            }

            else if(recvData[4]==0x02){
                Tx = ((recvData[6]*256+recvData[7])/500.)-50.;
                Ty = ((recvData[8]*256+recvData[9])/500.)-50.;
                Tz = ((recvData[10]*256+recvData[11])/500.)-50.;
            }

            std::cout << Fx << " " << Fy << " " << Fz << " " << Tx << " "
<< Ty << " " << Tz << std::endl;
        }

        close(socket_desc);

        return 0;
    }

    void recvMsg(char *msg) {
        size_t received = recv(socket_desc, msg, 14, 0);
    }

```

```

        printMsg(msg);
    }

    void printMsg(const char *msg)
    {
        for(size_t i = 6; i < 14; ++i) {
            //std::cout << (int)msg[i] << " ";
        }

        //std::cout << std::endl << std::endl;
    };

```

Revision History

날짜	내용
2026.09.15	초기 버전 (현재 버전)



찾으시는 내용이 없나요?

자주 묻는 질문을 먼저 확인해 보세요.

해결되지 않으면 문의하기를 작성해 주세요.

자주 묻는 질문

문의하기 →