

6-axis Force/Torque Sensors

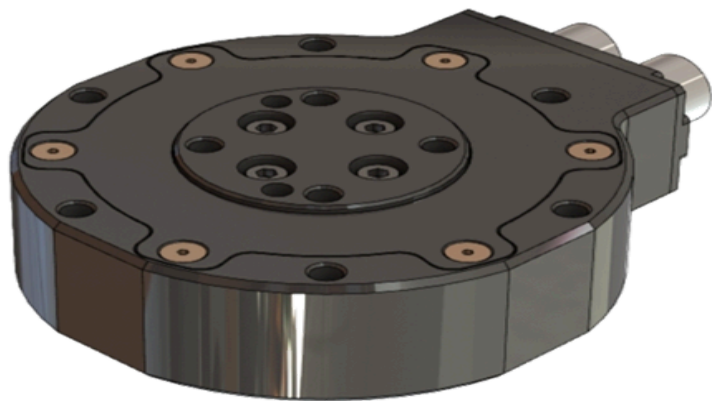
Smart 6-axis Force Torque Sensor, AFT200-D80-C

Rev. 2026.10.20



Force/Torque sensors user manual

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



FOREWORD

Thank you for purchasing our Force Torque sensor. This manual contains the information necessary for the correct use of the AIDIN ROBOTICS AFT200-D80 sensor. Please carefully read this manual when using this software. Please note that the basic performance of the product will not be

exhibited when the robot system is used out of conditions. This manual describes possible dangers and consequences using Force Torque sensor. Be sure to comply with safety precautions written in this manual to use safety and correctly.

NOTICE

This manual Do **NOT** allow copy, reproduction or share without authorization of AIDIN ROBOTICS. Please notify us any errors in this manual or the provided instructions.

MANUFACTURER

AIDIN ROBOTICS

SAFETY PRECAUTIONS

Installation of Force Torque sensor **MUST** be performed by qualified personnel in accordance with national and local codes. Please carefully read this manual when using this software.

WARNING & CAUTION

 This symbol indicates that a danger of possible serious injury or death exists if the associated instructions are not followed properly.

 This symbol indicates that a danger of possible harm to people or physical damage to equipment and facilities exists if the associated instructions are not followed properly.

 AFT200-D80-C sensor from AIDIN ROBOTICS meets CE-certified "A Class"

1. Introduction

1.1 Smart 6-axis Force Torque Sensor, AFT200-D80



1.2 Key features

- Smart 6-axis force/torque sensor
- All-in-one sensor (No additional amplifier)
- Digital output communications (CAN)
- Easy installation and data collection
- Grippers, robot hands, collaborative robot, industrial robot

1.3 Specs.

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

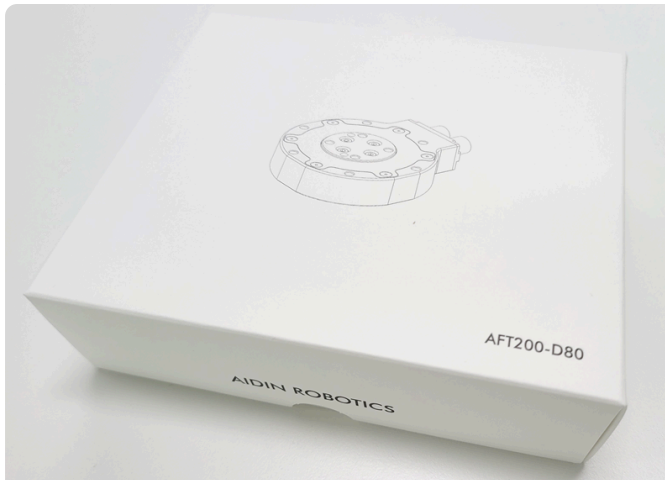
1.4 Flyer

-  [AIDIN ROBOTICS] AFT MINI+AFT200 leaflet(EN, 25.10).pdf

2. Installation Guide

2.1 Basic Components

- AFT200-D80 x 1 EA
- 3m Cable x 1 EA



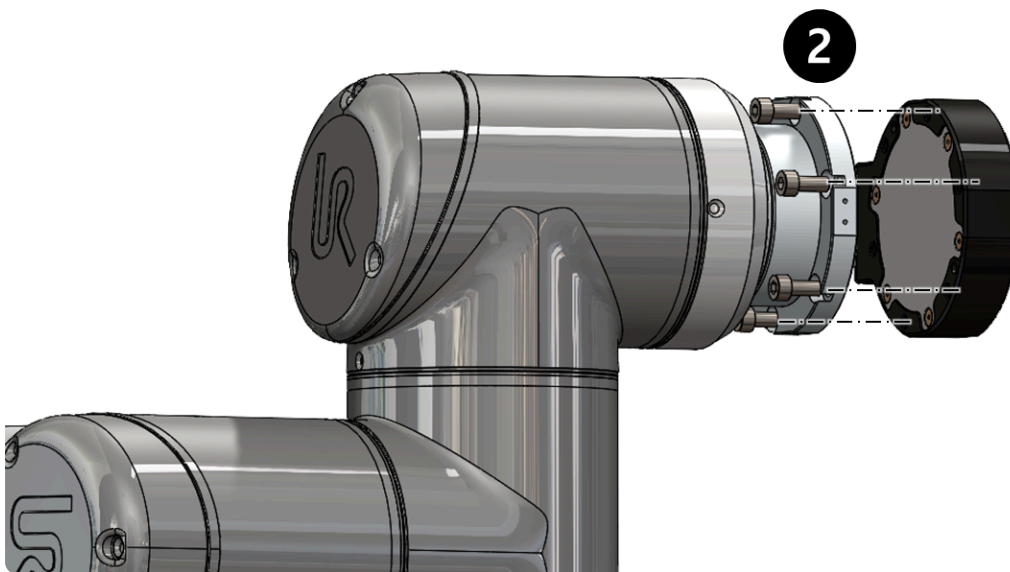
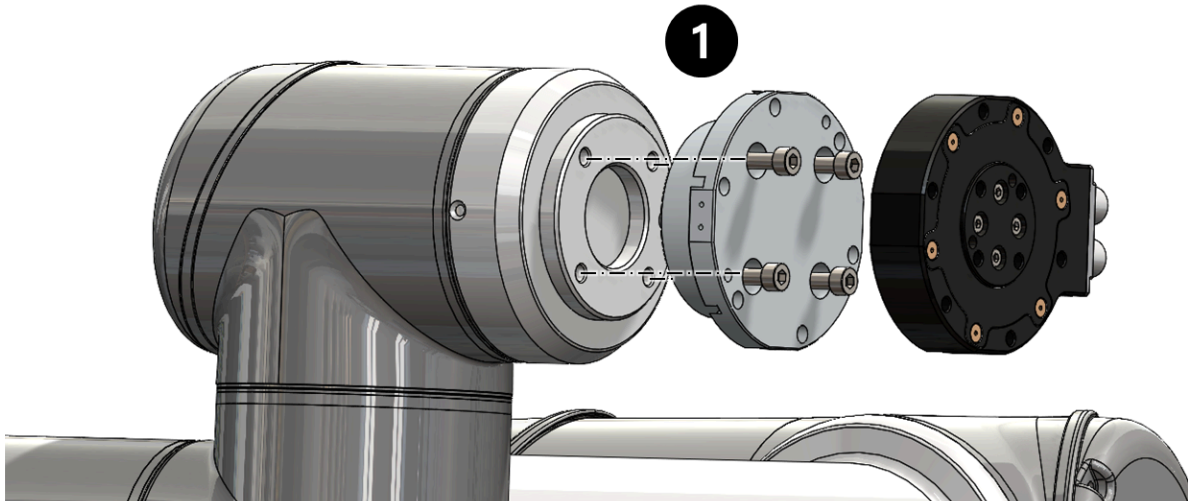
2.2 Mounting

- M5 bolt 5.2Nm required for tightening
- Performance cannot be guaranteed and A/S not possible when disassembling internal/external bolts



*Transparent parts are user part examples

- Sensor tightening order



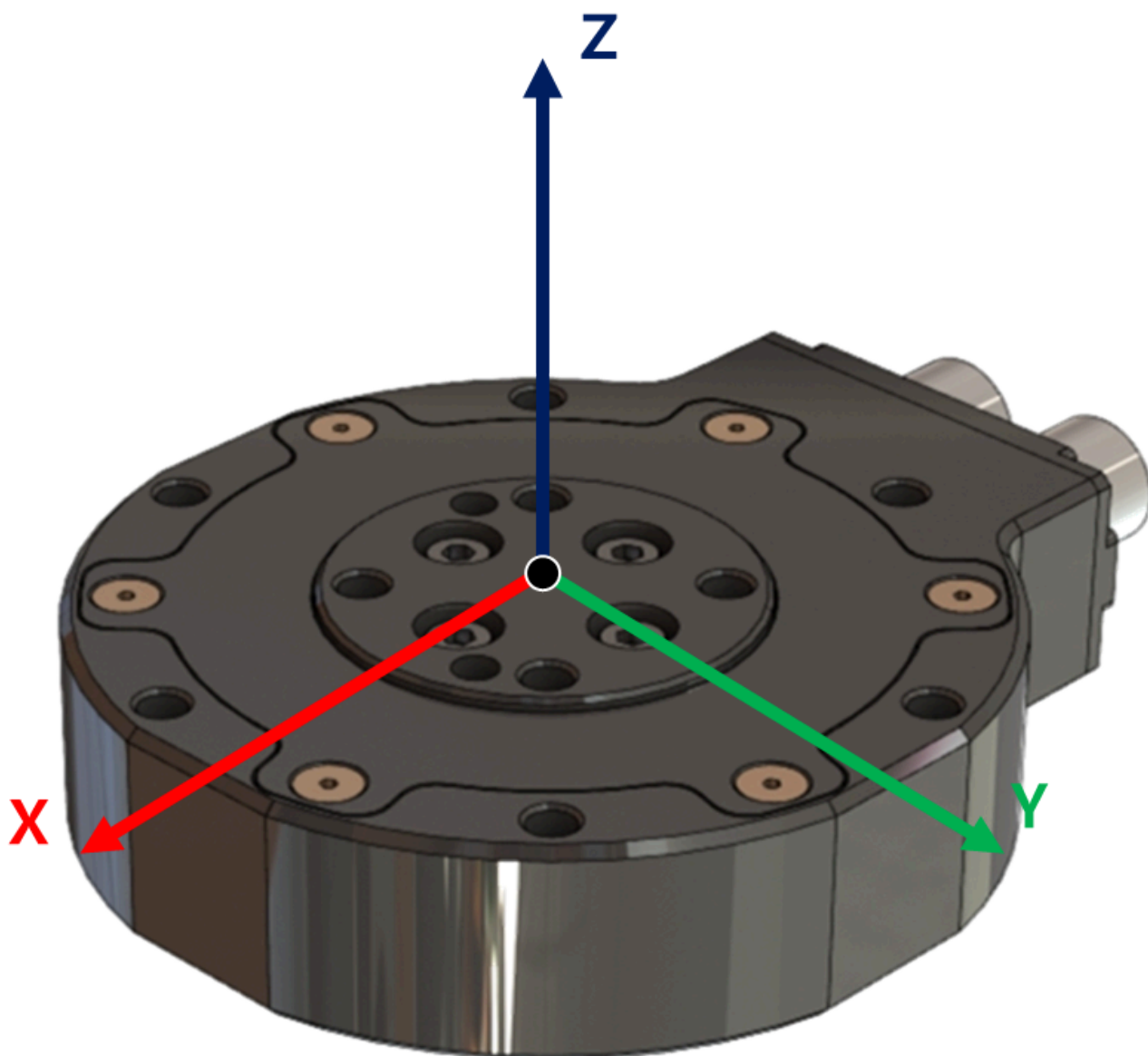
- CAUTION OF CABLE CUTTING AND EXCESS PULLING
- Secure the sensor line with the robot so that it does not pull as it moves
 - Do not fix it to the robot using cable ties, etc., or use cable ties to fix it in a bundle shape like other wires
 - We recommend that you use Velcro to fix it with the robot, and please use Velcro to fix it when fixing it in a bundle form with other lines

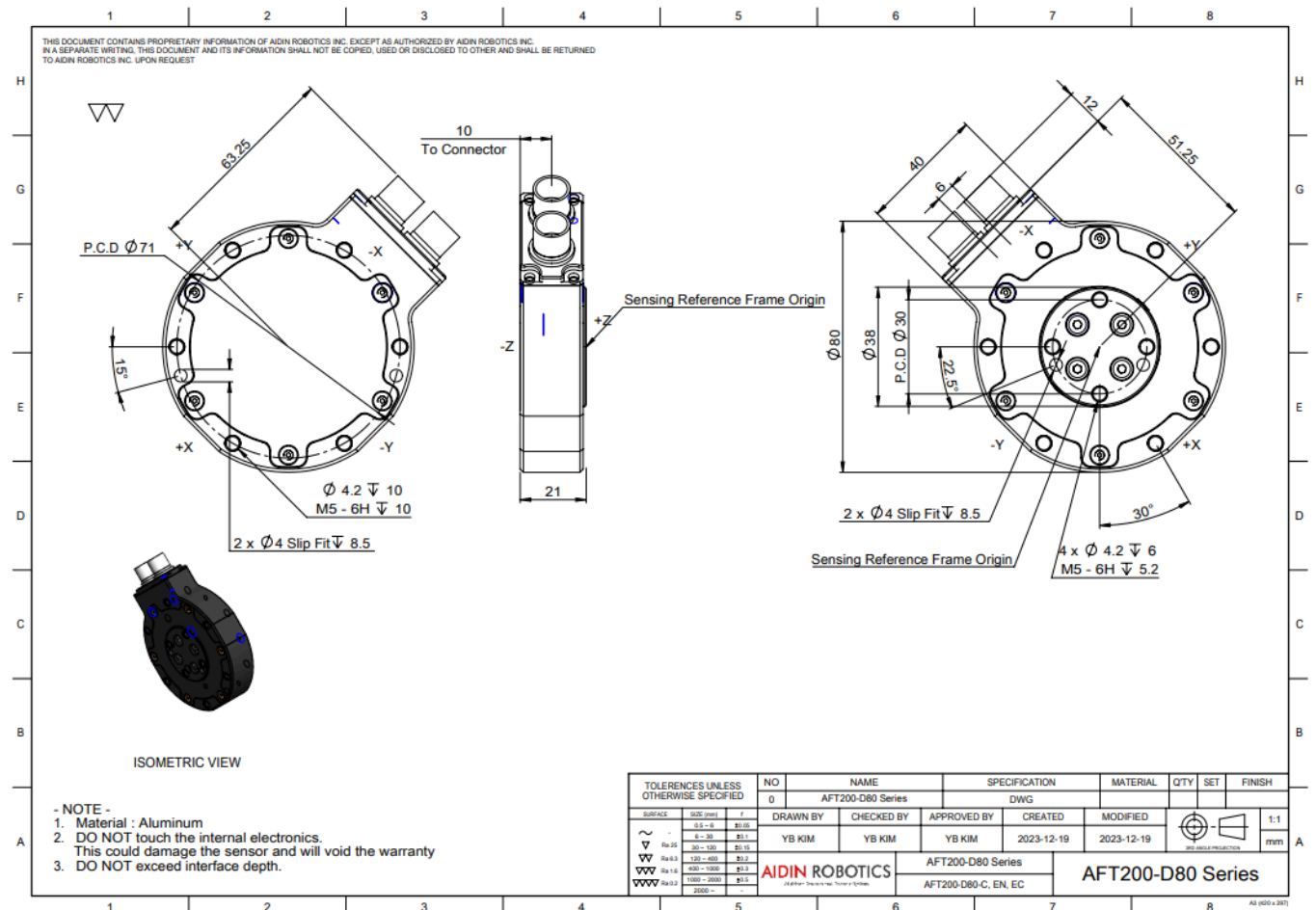
💡 Performance CANNOT be guaranteed if not followed to the tightening torque guide.

💡 If the sensor line is not fixed to the robot, the output signal may have a noise due to the pulled by robot

💡 Please check More detail setup guide as follow:

2.3 AXIS & Drawings





Drawing File :

- AFT200_D80.pdf



Port Location Notice

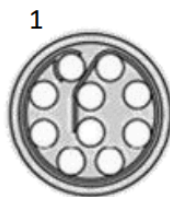
- **EtherCAT** Type model has two ports: **IN** and **OUT**.
- **Ethernet** Type model has one port, located on the **left side**.
- **CAN** Type model has one port, located on the **right side**. STEP File:
- AFT200-D80-C_배포용.zip

2.4 Wiring

2.4.1 Pin out



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 Cable open-end

- Cable pin out
 - Length: 3m



- 5V(VCC) - Orange, Black dot
- GND - Gray, Black dot
- CANH- Yellow, Black dot
- CANL - Pink, Black dot

💡 Other pins are not using

- Connection PCB Installation Guide
 - Insert the lines as illustrated below and fasten the bolts on the terminals.
 - Each connection point is indicated on the PCB connection point, and the sequence is as follows:



1. Sensor Connector (Connector section connecting to the sensor)
2. EtherNET Connector (Connector section connecting to the PC)
3. Sensor Connector
 1. Upper 2-port terminal block: VCC, GND
 2. Lower 4-port terminal block: VCC, GND, CAN High, CAN Low ****(Optional)****
 - VCC and GND should be connected to a power supply device providing stable 5 V.
 - The 9-pin D-sub is connected to the eCAN device.

⊘ We recommend using an approved power supply device.

2.4 Default Data Output Rate

- The output rate can be changed from 100 Hz to 1000 Hz
- Please refer to the “2.5 Mode setting” below for how to change it

Table 1.3 - Default Value

Receiver ID	0x01
Data Output Rate	100Hz

2.5 Mode setting

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 compensasion)
			0x01	Bias set (w Temperature compensasion)
		0x03	0x00	Transmitting mode (w/o Temperature compensasion)
			0x01	Transmitting mode (w Temperature compensasion)
		0x05		Sample rate setting mode
	0xFF	0x00		ID confirming mode
		0xFF		Factory Reset mode

 To stabilize the sensor signal, it is recommended to have a running time of about **30 minutes.

* P**lease leave the sensor data on for at least 10 minutes before using it

 The first 10 minutes of sensor output data can cause data to flow.

2.5.1 MODE SETTINGS

CAN ID Setting mode

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

- Data field [0] : Current Set ID (Factory Setting 0x01)
- Data field [2] : User setting ID for transmit

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

Bias Setting mode

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

- Data field [2] : 0x00 : Bias setting without temperature compensation (This command should be used with transmitting mode(without temperature compensation) command)
- Data field [2] : 0x01: Bias setting with temperature compensation (This command should be used with transmitting mode(with temperature compensation) command)

Continuous transmitting mode

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

- Data field [2] : 0x00 : Transmitting mode without temperature compensation (This command should be used with Bias setting(without temperature compensation) command)
- Data field [2] : 0x01: Transmitting mode with temperature compensation (This command should be used with Bias setting(with temperature compensation) command)

Table 1.9 - 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	Force Output [0]	Force Output [1]	Force Output [2]	Force Output [3]	Force Output [4]	Force Output [5]	xx	xx

- $F_x \text{ Output} = \text{Force Output [0]} * 256 + \text{Force Output [1]}$
- $F_y \text{ Output} = \text{Force Output [2]} * 256 + \text{Force Output [3]}$
- $F_z \text{ Output} = \text{Force Output [4]} * 256 + \text{Force Output [5]}$
- $\text{Force [N]} = \text{Force Output} / 100 - 300$

Table 1.10 - 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+1	Torque Output [0]	Torque Output [1]	Torque Output [2]	Torque Output [3]	Torque Output [4]	Torque Output [5]	xx	xx

- $T_x \text{ Output} = \text{Torque Output [0]} * 256 + \text{Torque Output [1]}$
- $T_y \text{ Output} = \text{Torque Output [2]} * 256 + \text{Torque Output [3]}$
- $T_z \text{ Output} = \text{Torque Output [4]} * 256 + \text{Torque Output [5]}$
- $\text{Torque [Nm]} = \text{Force Output} / 500 - 50$

Sample rate setting mode

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]
- $\text{Sample rate parameter} = \text{Sample rate parameter [0]} * 256 + \text{Sample rate parameter [1]}$
- $\text{Sample rate} = (1,000,000 / \text{Sample rate parameter})$ 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 Confirming mode

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

Factory Reset mode

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

2.5.2 DEVICES

USB-to-CAN Device

- You may use a different CAN-enabled board, but use IXXAT's converter to use the sample program



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



USB-to-CAN V2 (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]*

- lxxat VCI Setup 4.0.xxx.0.exe
- **VCI V4 check during installation and reboot after installation**

EtherNET-to-CAN Device

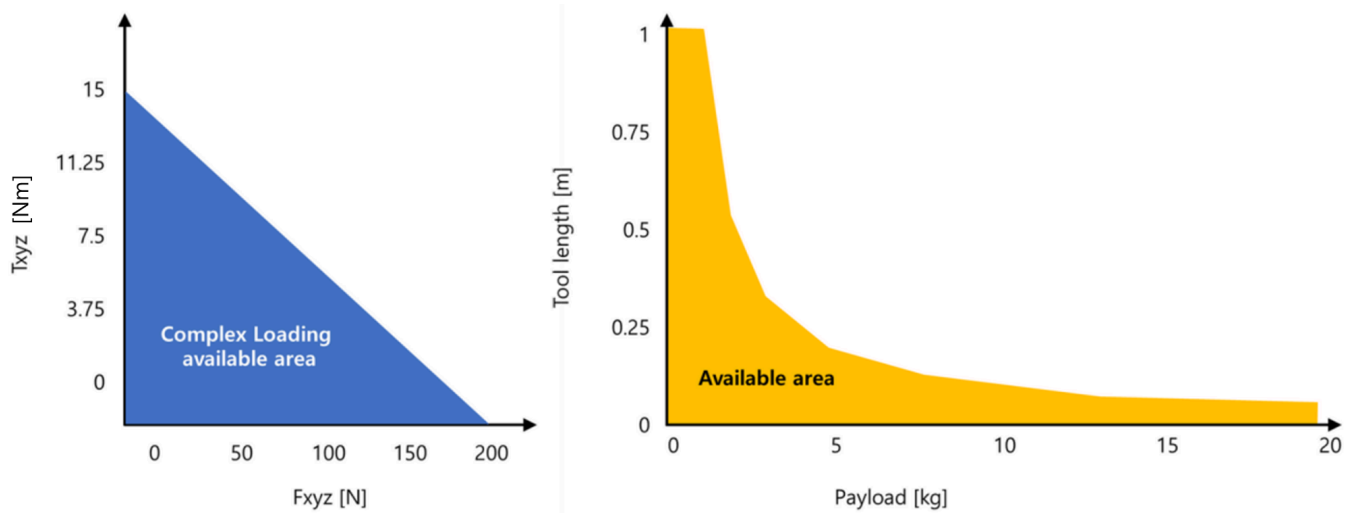
- **Using Systembase's eCAN model**



- **Download Manual and eCANView from the following site:**
- <https://www.sysbas.com/en/downloads/ecan/>

2.6 Force/Torque Out of Range

The sensor can operate up to its nominal capacity, while a single axis load. On Nominal capacity reading is incorrect and invalid. The nominal capacity of composite loading is The following diagram illustrates a complex loading scenario. The sensor cannot operate outside the normal operating area



The following graph shows the range of payloads and tool lengths that can be used with AFT200-D80-C sensor for applications requiring high or moderate precision. • The total percentage of the calibrated range used by Fxy and Tz axes is greater than 105%. Refer to the following Fxy, Tz equation.

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

• The total percentage of the calibrated range used by Fz and Txy axes is greater than 105%. Refer to the following Fz, Txy equation.

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

3. Software

3.1.1 IXXAT's USB-to-CAN

- Sample Program (Provided by IXXAT)
 - CanAnalyzer3 Mini windows → IXXAT folder → IXXAT CanAnalyzer3 Mini

USB-to-CAN V2 compact CAN-1

Statistics

Receive Counter	9081
Error Counter	0

Hardware

Controller	Bosch C_CAN
Clock Rate	1 MHz
Serial Number	HW643084
Revision	1.5
Driver Version	4.1.212.0

Features

- 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 11898-12...)
- 64-bit time stamp counter

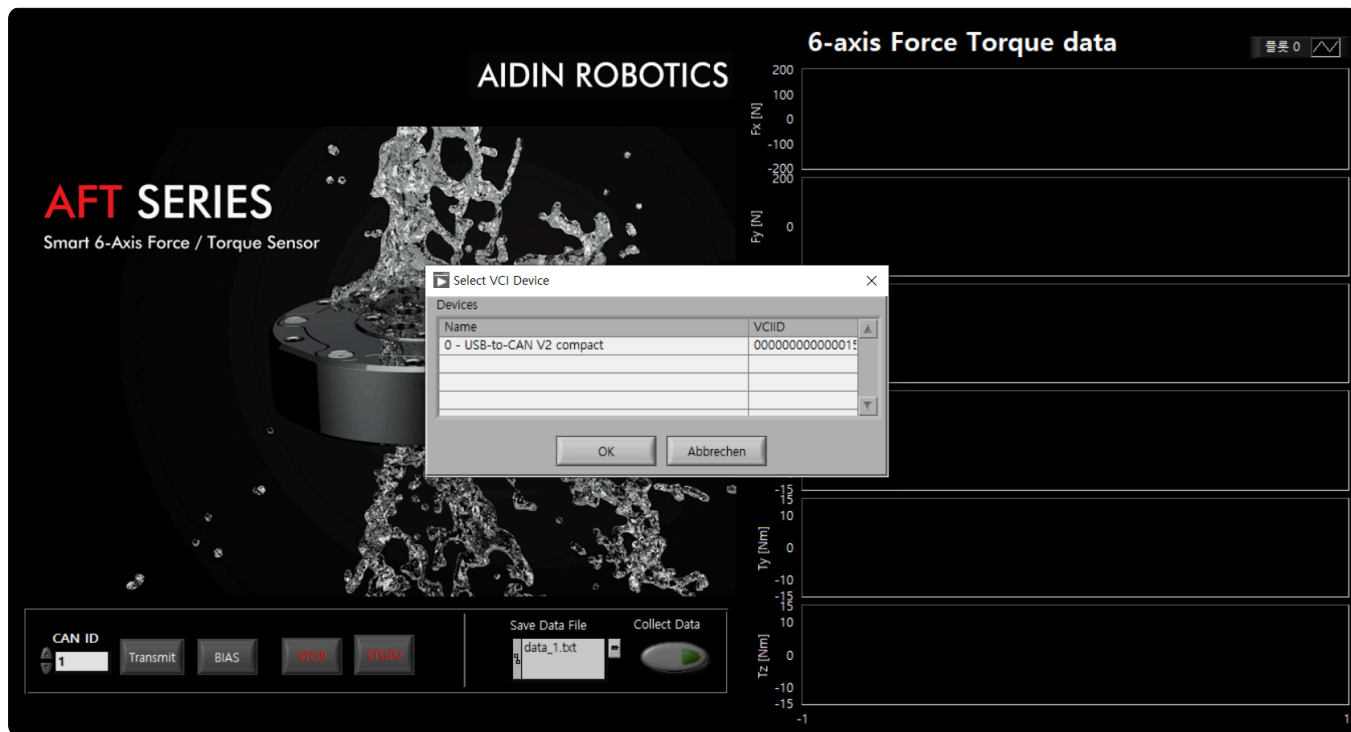
No	Time (abs)	State	ID (hex)	DL...	Data (hex)	ASCII
9,047	00:47:23.169		2	8	63 EA 5D A8 64 E7 4A DD	c].d.J.
9,048	00:47:23.189		1	8	67 18 78 F9 B7 60 00 00	g[.~.
9,049	00:47:23.189		2	8	63 F1 5D A3 64 EE 4A DD	c].d.J.
9,050	00:47:23.209		1	8	67 32 7C 00 B7 60 00 00	g2[.~.
9,051	00:47:23.209		2	8	63 E9 5D A8 64 E4 4A DD	c].d.J.
9,052	00:47:23.229		1	8	67 33 78 EA B7 63 00 00	g3[.c.
9,053	00:47:23.229		2	8	63 E9 5D A7 64 E8 4A DD	c].d.J.
9,054	00:47:23.249		1	8	67 27 7C 00 B7 66 00 00	g[.f.
9,055	00:47:23.249		2	8	63 EE 5D A5 64 E7 4A DE	c].d.J.
9,056	00:47:23.269		1	8	67 1F 7C 00 B7 57 00 00	g[.W.
9,057	00:47:23.269		2	8	63 EF 5D A5 64 ED 4A DE	c].d.J.
9,058	00:47:23.289		1	8	67 34 78 EF B7 60 00 00	g4[.~.
9,059	00:47:23.289		2	8	63 E9 5D A7 64 E8 4A DF	c].d.J.
9,060	00:47:23.308		1	8	67 28 78 F4 B7 68 00 00	g[.i.k.
9,061	00:47:23.309		2	8	63 ED 5D A6 64 E8 4A DF	c].d.J.
9,062	00:47:23.329		1	8	67 28 78 F5 B7 56 00 00	g[.i.V.
9,063	00:47:23.329		2	8	63 E8 5D A8 64 ED 4A DF	c].d.J.
9,064	00:47:23.349		1	8	67 31 78 F8 B7 57 00 00	g1[.W.
9,065	00:47:23.349		2	8	63 EC 5D A6 64 E4 4A DE	c].d.J.
9,066	00:47:23.369		1	8	67 33 78 F5 B7 4D 00 00	g3[.M.
9,067	00:47:23.369		2	8	63 ED 5D A9 64 E4 4A DE	c].d.J.
9,068	00:47:23.389		1	8	67 18 78 EE B7 5E 00 00	g[.~.^.
9,069	00:47:23.389		2	8	63 F2 5D A8 64 ED 4A DD	c].d.J.
9,070	00:47:23.409		1	8	67 31 78 F8 B7 66 00 00	g1[.f.
9,071	00:47:23.409		2	8	63 EA 5D A5 64 E8 4A DE	c].d.J.
9,072	00:47:23.429		1	8	67 32 78 F1 B7 58 00 00	g2[.~.
9,073	00:47:23.429		2	8	63 E9 5D AA 64 E9 4A DE	c].d.J.
9,074	00:47:23.448		1	8	67 34 78 F2 B7 4E 00 00	g4[.N.
9,075	00:47:23.449		2	8	63 E9 5D A8 64 E9 4A DF	c].d.J.
9,076	00:47:23.469		1	8	67 2E 78 F1 B7 58 00 00	g[.X.
9,077	00:47:23.469		2	8	63 ED 5D A5 64 E8 4A E0	c].d.J.
9,078	00:47:23.489		1	8	67 36 78 E0 B7 5E 00 00	g6[.~.
9,079	00:47:23.489		2	8	63 E8 5D A4 64 E8 4A E0	c].d.J.
9,080	00:47:23.508		1	8	67 2E 78 EC B7 66 00 00	g[.f.
9,081	00:47:23.509		2	8	63 E3 5D A5 64 E7 4A E1	c].d.J.
9,082	00:47:23.529		1	8	67 24 78 F5 B7 56 00 00	g5[.V.
9,083	00:47:23.529		2	8	63 E8 5D A5 64 E7 4A E1	c].d.J.
9,084	00:47:23.549		1	8	67 25 78 FE B7 63 00 00	g%[.c.
9,085	00:47:23.549		2	8	63 E9 5D A5 64 E8 4A E2	c].d.J.
9,086	00:47:23.569		1	8	67 1D 78 F3 B7 65 00 00	g[.e.
9,087	00:47:23.569		2	8	63 E9 5D A7 64 E8 4A E1	c].d.J.

Tx	ID (hex)	Description	Ext.	RTR	Data (hex)	Count	Time (ms)	Inc Mode	Byte
	102				00 01 01	0	10.00	None	
	102				FF FF	0	10.00	None	
	102				01 05 4E 20	0	10.00	None	
	102				01 02 01	0	10.00	None	
	102				01 03 01	0	10.00	None	

• A sample program provided by AIDIN ROBOTICS

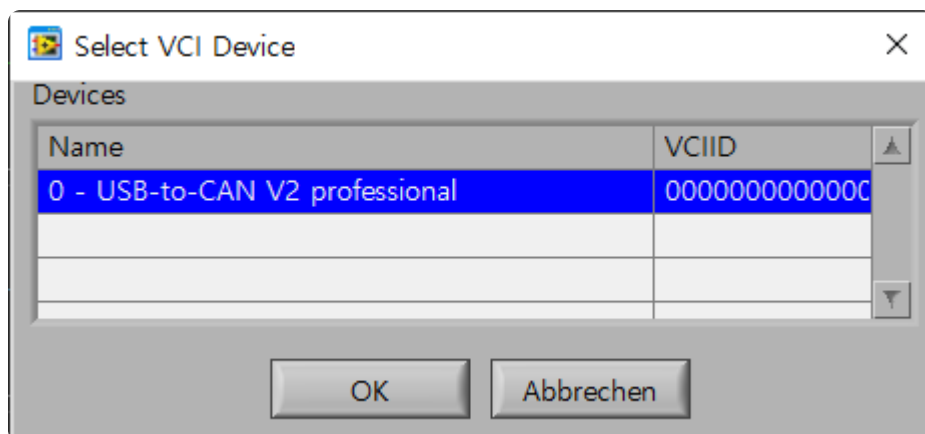
- Install IXXAT CAN VCI V4 & installer below
- Sample program

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

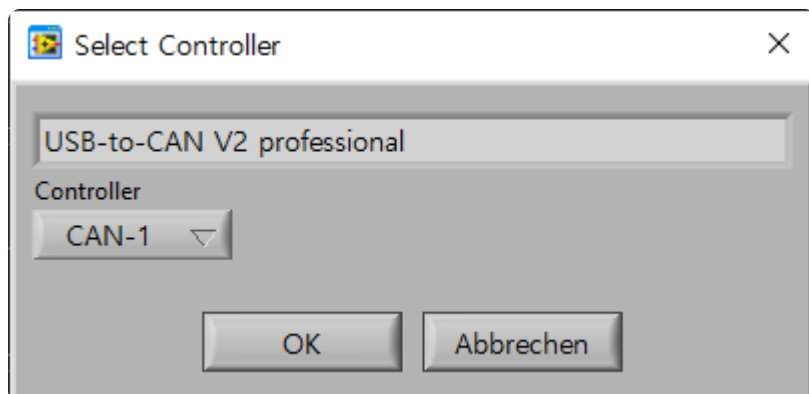


- Pop-up screen when running a sample program

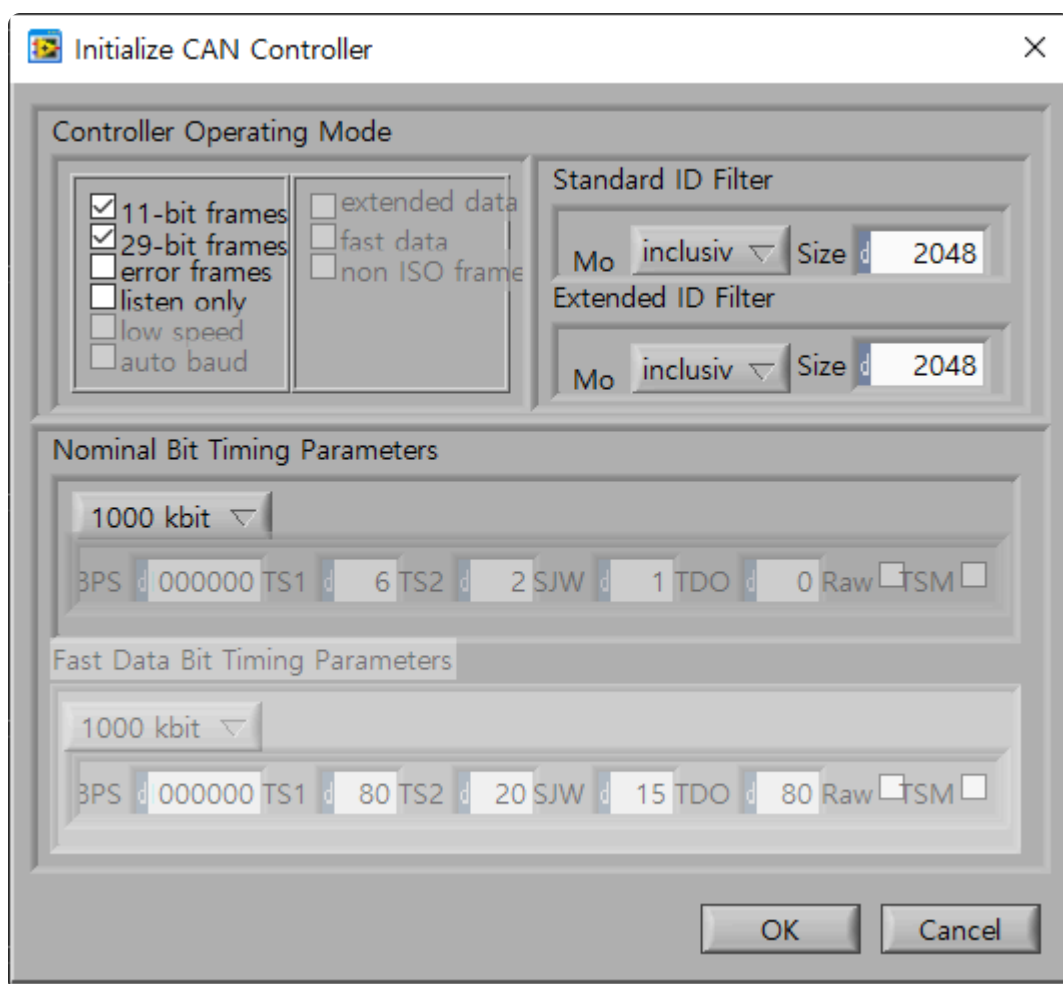
1. **CAN Device Selection** Select a device. Devices currently connected to your PC appear as a list. Click the appropriate device and press the OK button. ※ The sample program is based on IXXAT's USB-to-CAN converter.



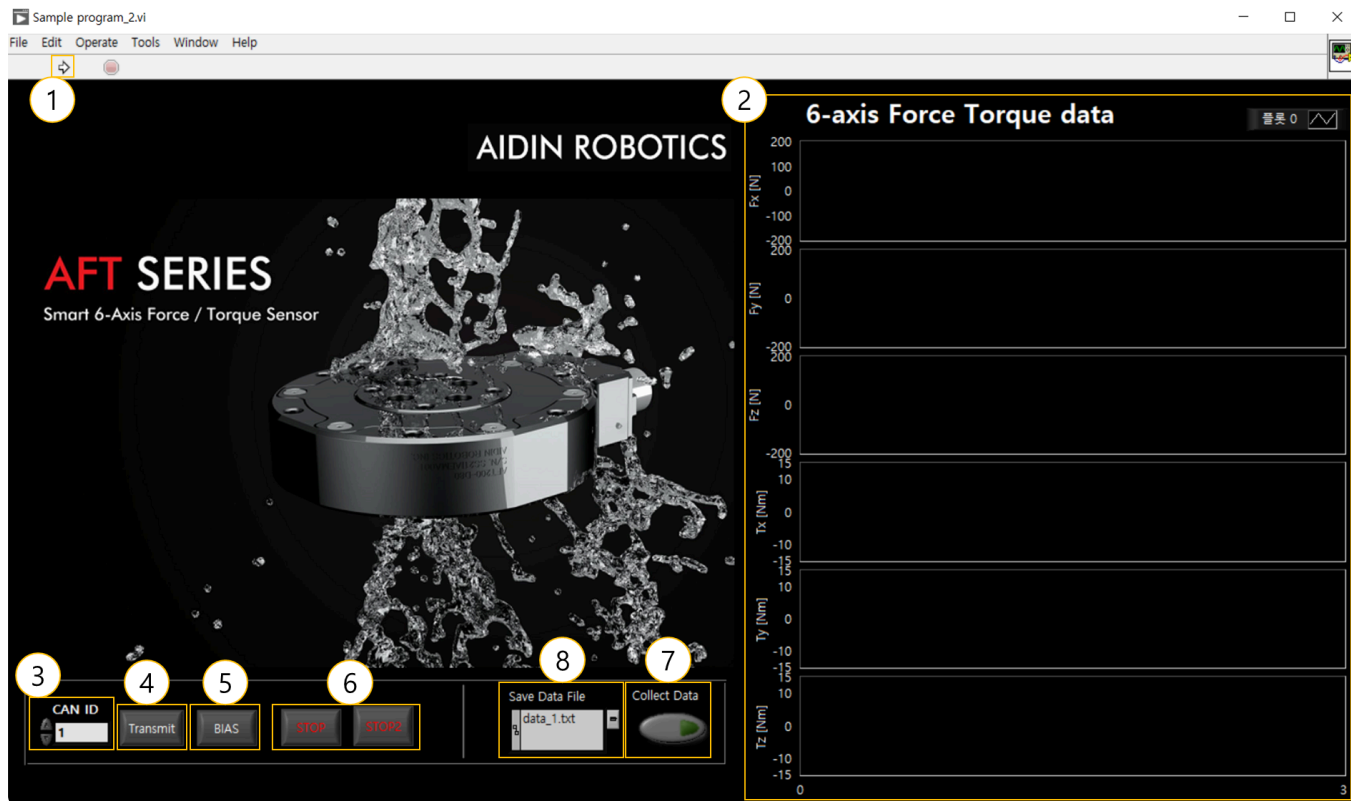
1. **CAN CONTROLLER SELECTION** Displays the device name selected in the previous step. Select the controller channel below and press the OK button.



1. **CAN Controller Initial Setup Steps** to initialize the controller selected in the previous step. Check the frame and baud rate and press the OK button.



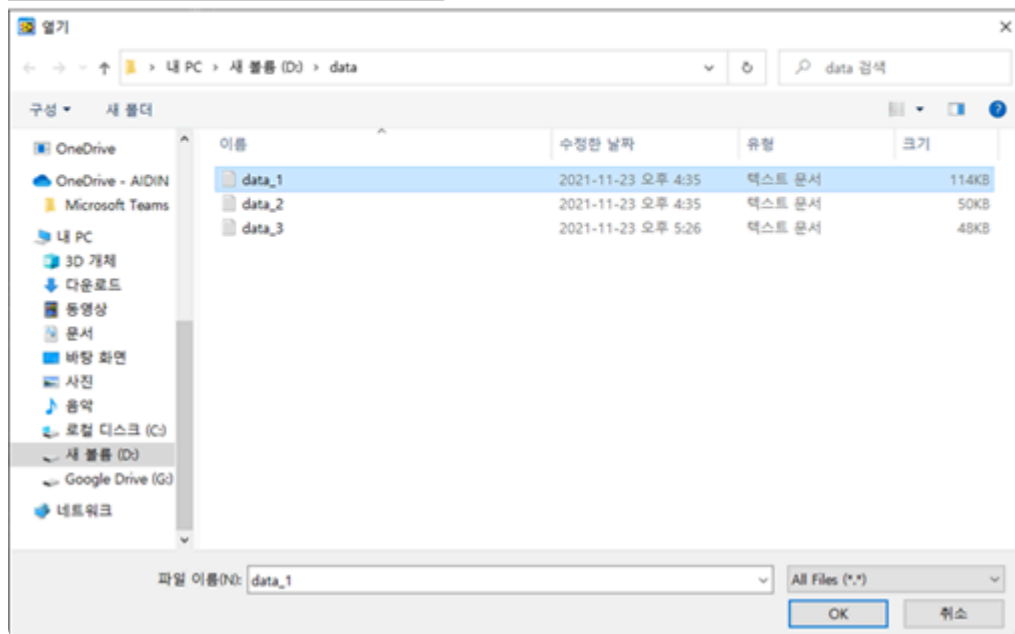
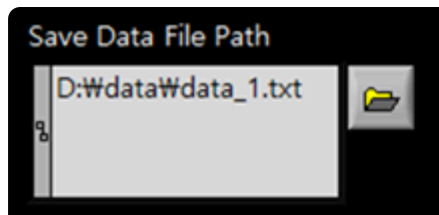
- **Sample Program Interface Description**



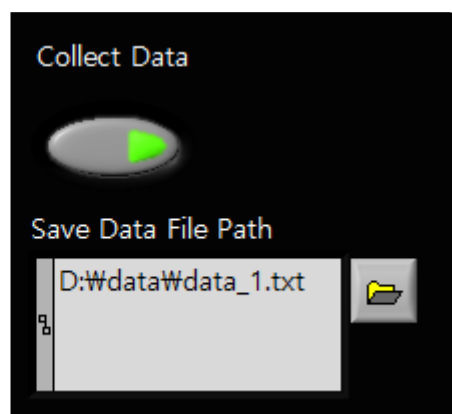
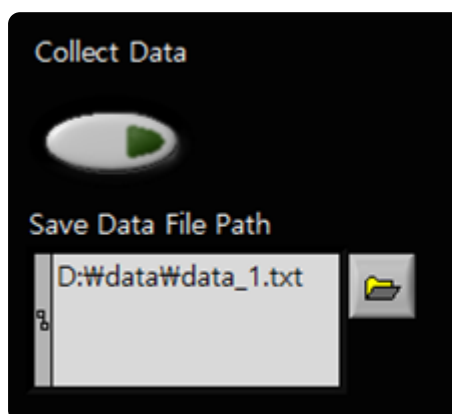
① Program run Run the sample program. ② Graph plot Real-time data of 6-axis force/torque data (Fx, Fy, Fz, Tx, Ty, Tz) can be viewed in a graph. ③ CAN ID Sets the CAN ID of the connected sensor. If you have more than one ID, enter the lowest ID. ex) For 0x01(1), 0x02(2), enter 1 ④ CAN transmit Send CAN message to output 6-axis force/torque value. CAN Msg [0x102, 0x01, 0x03, 0x01] ⑤ BIAS Initializes all 6-axis force/torque data to zero. ⑥ STOP Exit the program. Press both STOP buttons and the program will exit after 3 seconds. ⑦ Collect Data Save the force/torque data log. Please refer to the description of data storage on the page below. ⑧ Save Data File Path Sets the path for storing force/torque data logs. Please refer to the description of data storage on the page below.

3.1.2 Save data log

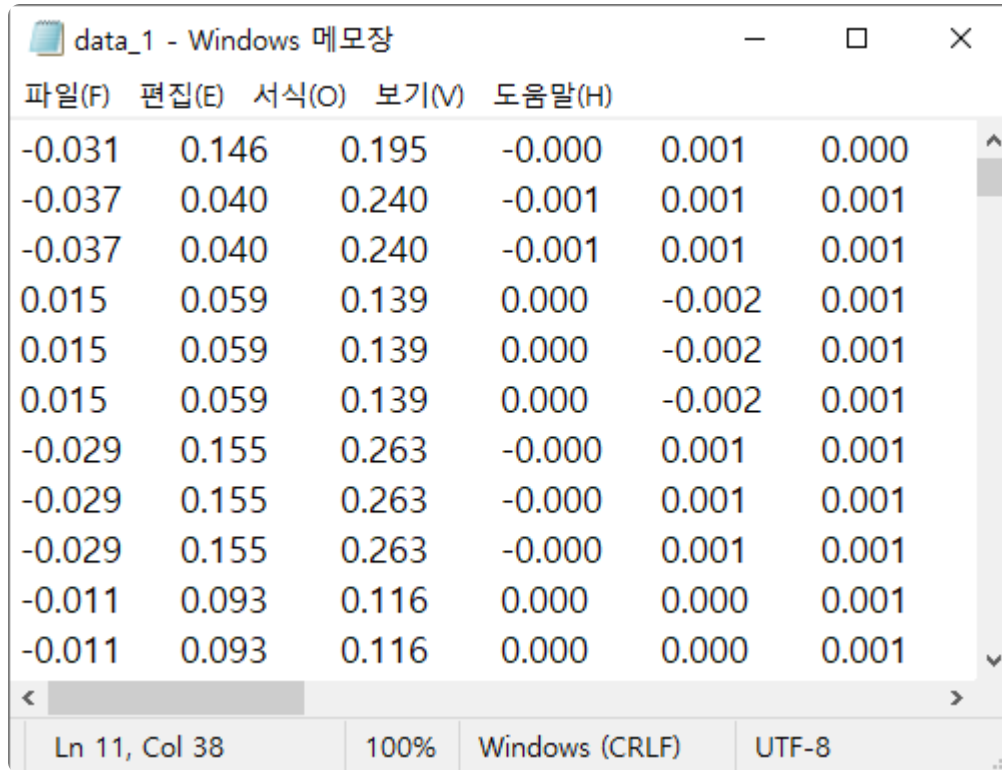
1. Save Directory Settings Touch the folder button to open a pop-up window. Select a file (.txt) in the desired directory and press the OK button to specify it. Check the path shown in Path. (If the specified file does not exist in that path, create the file automatically)



1. **Saving the Data Log** Press the button under Collect Data to start saving. When saving, the button displays a fluorescent color. Press the button to exit the save. Data is stored cumulatively in a file, so it is recommended that you specify a new file for each save.



1. **Checking the Data Log** You can open a file in that directory to view the stored force/torque data. The data are stored in each column in the order Fx, Fy, Fz, Tx, Ty, and Tz.

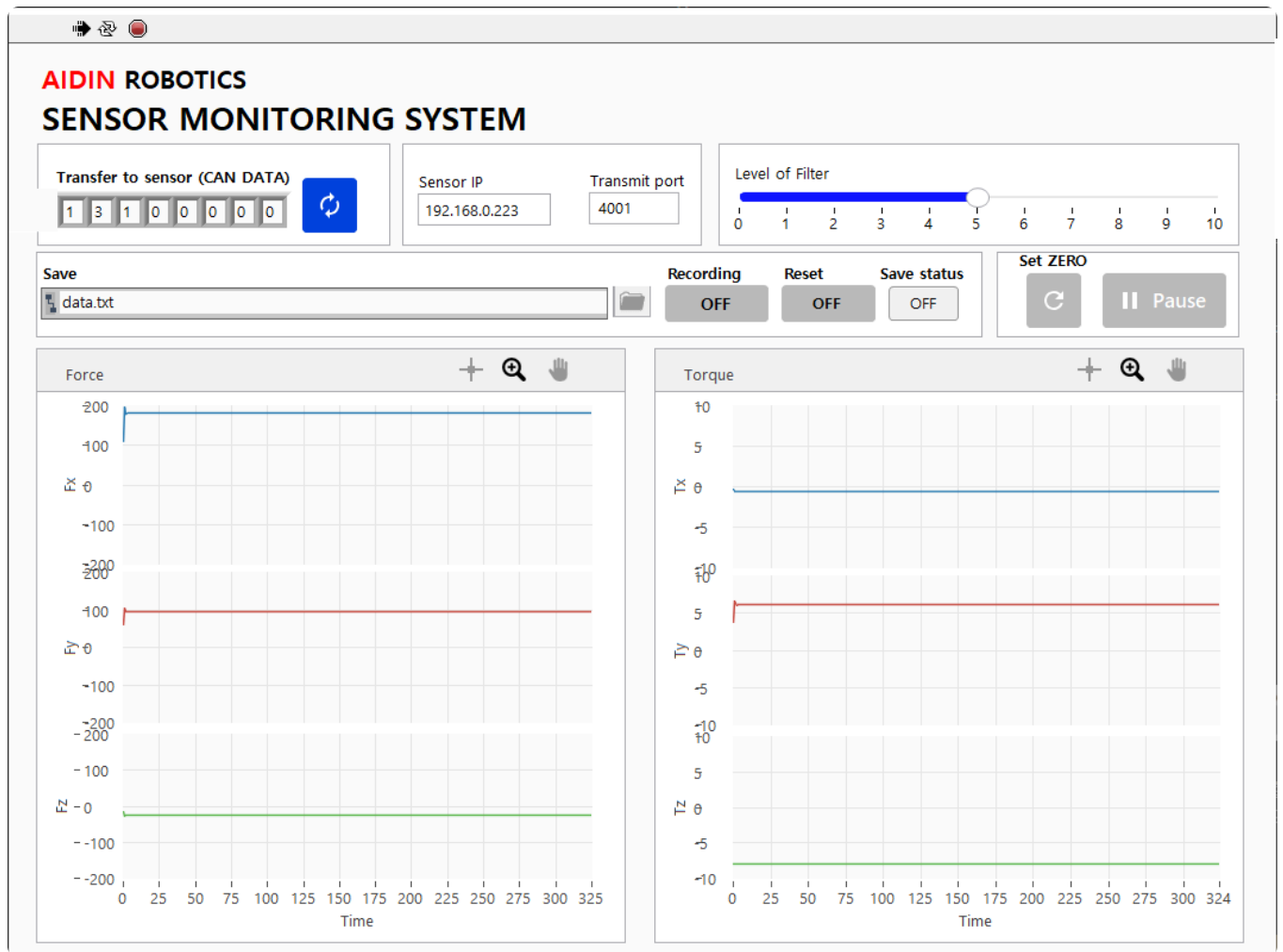


파일(F)	편집(E)	서식(O)	보기(V)	도움말(H)	
-0.031	0.146	0.195	-0.000	0.001	0.000
-0.037	0.040	0.240	-0.001	0.001	0.001
-0.037	0.040	0.240	-0.001	0.001	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
0.015	0.059	0.139	0.000	-0.002	0.001
-0.029	0.155	0.263	-0.000	0.001	0.001
-0.029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

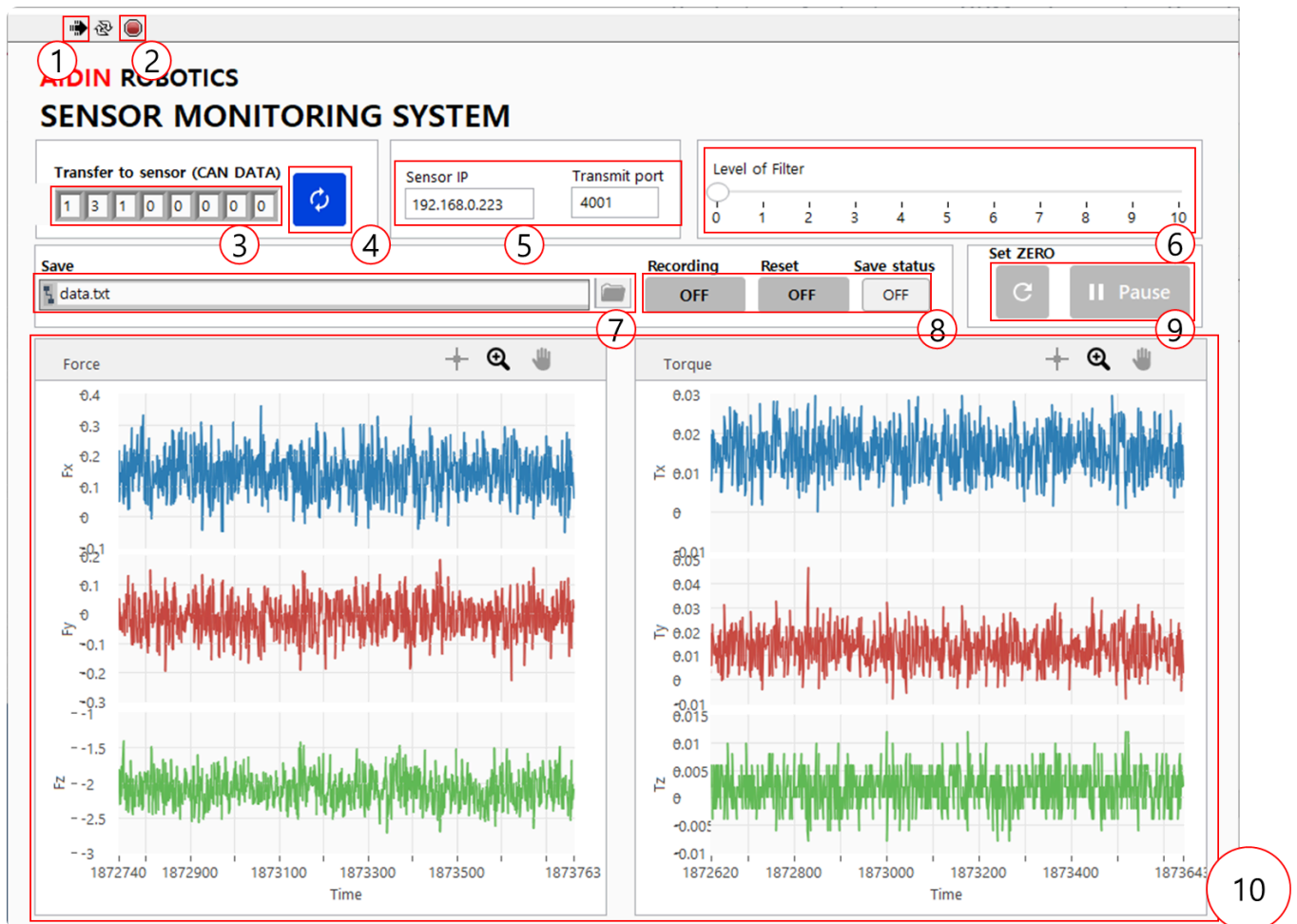
Ln 11, Col 38 100% Windows (CRLF) UTF-8

3.2.1 SystemBase's EtherNET to CAN

- A sample program provided by AIDIN ROBOTICS
-  AIDIN_Sample_EtherNET.zip (대용량 파일 — 별도 제공)
 - AIDIN_Installer→Volume→setup , Install setup
 - **Set IP through Systembase's manual or the document below**
 -
 - Run AIDIN_Sample_EN→AIDIN_Sample_EN.exe
- Pop-up screen when running a sample program



- **Sample Program Interface Description**



① Program run **Run the sample program.** ② STOP **Exit the program. Press both STOP buttons and the program will exit after 3 seconds.**

③ CAN transmit message **CAN Msg [0x102, 0x01, 0x03, 0x01]** ④ CAN transmit button **Send CAN message to output 6-axis force/torque value.** ⑤ IP Setting **Enter the IP of the sensor matched in the above setting** ⑥ Filtering **Default LPF filter configured, available from 0 to 9.9 (10 is not available)** ⑦ Save Data File Path **Sets the path for storing force/torque data logs. Please refer to the description of data storage on the page below.** ⑧ Collect Data **Save the force/torque data log. Please refer to the description of data storage on the page below.** ⑨ BIAS, PAUSE **Initializes all 6-axis force/torque data to zero. You can measure the desired force value by removing elements from the installation and attachment. When you click the PAUSE button, you can pause the sensor data value.** ⑩ Graph **Real-time data of 6-axis force/torque data (Fx, Fy, Fz, Tx, Ty, Tz) can be viewed in a graph.**

3.2.2 Save Datalog



- 1. Save Directory Settings** Touch the folder button to open a pop-up window. Specify the file (.txt) in the desired directory. Check the path shown in Path. (If the specified file does not exist in that path, create the file automatically)
- 2. Saving the Data Log** Press the button below Collect Data to start saving. When saving, the button displays a fluorescent color. Press the button to exit the save. Data is stored cumulatively in a file, so it is recommended that you specify a new file for each save. - Initialize the data saved when you click the Reset button and start saving again - Please be careful with the Reset button



- 3. Checking the Data Log** You can open a file in that directory to view the stored force/torque data. The data are stored in each column in the order Fx, Fy, Fz, Tx, Ty, and Tz.

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

Source codes

4. Python code -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()

```

5. Window code - 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;
*
*****/

//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;
};
```

6. Linux -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;
 *
 *****/

//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

Date	Description
2026.10.20	Q & A section removed (Current version)
2026.09.15	Initial release



Can't find what you're looking for?

Try our FAQs first. Still need help? Contact us and our team will get back to you.

[Browse FAQs](#)

[Contact Sales →](#)