

Datasheet for conditioning sensor

Model: A2E-TRI

Updated 29 September 2026

Index

- [Datasheet for conditioning sensor](#)
- [Model: A2E-TRI](#)
 - [Updated 29 September 2026](#)
- [Index](#)
- [Description](#)
 - [A2E-TRI](#)
 - [A2E-TRI Flanged](#)
 - [Open Source Software](#)
- [Specifications Overview](#)
 - [Electrical Specifications](#)
 - [Mechanical Specifications](#)
- [System Overview](#)
 - [Physical Connection](#)
 - [Data Flow](#)
 - [Communication Architecture](#)
 - [Data Format \(Protobuf\)](#)
 - [Benefits](#)
 - [Data Packet Structure](#)
 - [Decoding the Data](#)
 - [Supported Environments](#)
- [Sensor Configurations](#)
 - [Decimation setting](#)
 - [Low-pass or High-pass filter options](#)
- [On-board FFT](#)
 - [Independent from the raw stream](#)
 - [Configuration](#)
 - [FFT output rate](#)
 - [Payload size and multi-chunk assembly](#)
 - [Coexistence with the raw stream](#)
- [Security Features](#)
 - [Secured by HMAC-MD5](#)
 - [Encrypted secure boot](#)

- Mechanical Drawing
 - A2E-TRI
 - A2E-TRI-F
- Connector & Pinout
- Performance Characteristics
- Data Throughput
- Startup & Configuration
 - Boot Sequence
 - Network Initialization
 - IP Assignment Flow
 - Link State Changes
 - Timing
 - SNTP Time Synchronization
 - Boot Modes
 - Configuration Persistence
- API / Software
 - Device Discovery
 - UDP Control Protocol
 - Request Frame
 - Response Frame
 - Available Commands
 - Sensor Configuration Parameters
 - Full Scale Range
 - Axis Selection
 - Output Data Rate Divider
 - Digital Filter
 - Filter Cutoff Frequency
 - DC Removal
 - FFT Size
 - FFT Precision
 - TCP Data Stream
 - Framing
 - Message Structure
 - Payload: FrameStream (Acceleration Data)
 - Payload: FFTStream (Frequency Domain Data)
 - UDP Data Transport
 - Application-layer fragmentation header
 - Receiver reassembly

- When multi-chunk fires
 - Throughput considerations
- Response Status Codes
- Environmental & Compliance
 - Applicable Standards
- LED Indicators
 - Status LED
- Update firmware by TFTP
- Ordering Information
- Revision History

Description

The A2E-TRI is a PoE powered accelerometer for 10/100Mbit Ethernet networks.

It features a high performance, low noise sensor with very high sampling rate of 26.7kHz and a wide bandwidth.

The sensor is enclosed in an IP67 CNC machined aluminum enclosure, fully potted.

A2E-TRI



A2E-TRI Flanged



Control communication is done securely over UDP (HMAC-MD5 authenticated). Data transmission runs over either TCP or UDP — each of the two data streams (raw acceleration and FFT) can be switched independently. The default is TCP, streaming a nominal 256 samples per packet. The protocol used for both configuration and data is Google's protobuf.

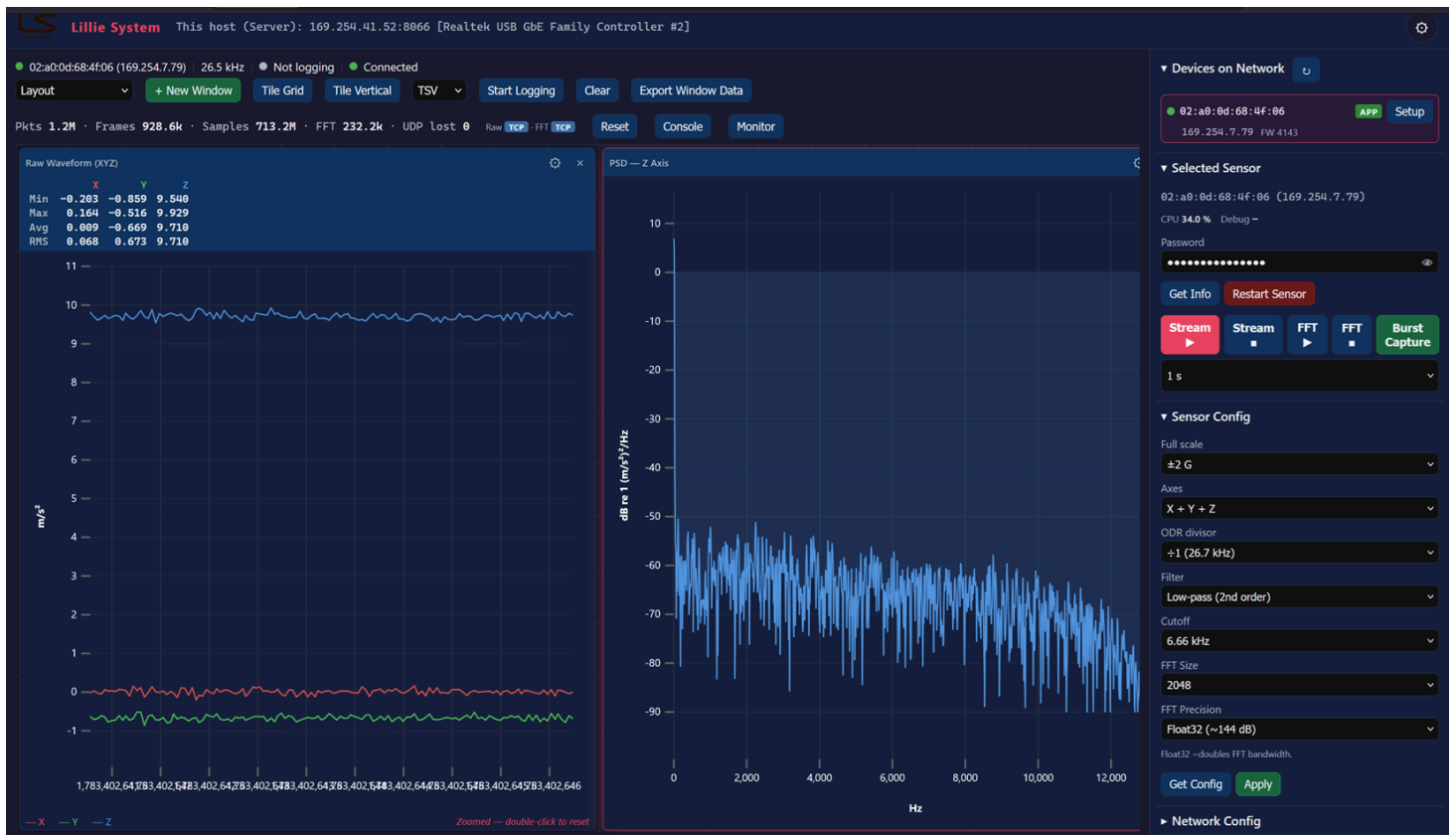
The A2E-TRI can simultaneously stream **raw 3-axis acceleration at the full 26.7 kHz sample rate** and **on-device FFTs at up to 2048-sample resolution** on the same Ethernet interface. Both streams

are independent, can be enabled together or in isolation, and each chooses its own transport (TCP or UDP).

Open Source Software

A fully open source app to connect, configure and log data is provided. It can be downloaded from:

[Lillie Systems Github - browser-viewer](#)



Specifications Overview

Electrical Specifications

Parameter	Value
Supply Voltage	48 V (PoE) IEEE 802.3af/at compatible
Power Consumption	< 2 W Typical operation
Interface	10/100 Ethernet - Auto-negotiation
Protocols	TCP (data), UDP (config) - Protobuf-based
Sampling Rate	Configurable 104 Hz to 26.7 kHz
Output Data Format	Signed 16-bit integers or floats (selectable FFT resolution)
Axes	Configurable single / dual or 3-axis simultaneously
Full Scale Range	±2g / ±4g / ±8g / ±16g
Bandwidth	Up to ~6.3 kHz, configurable
Packet Size	256 samples / packet
Time Synchronization	NTP supported

Mechanical Specifications

Parameter	Value
Enclosure	CNC machined aluminum, Fully potted
Protection Rating	IP67
Mounting	Bolt-on (flat surface)

Parameter	Value
Connector	M12 D-coded Industrial Ethernet
Dimensions	67 × 42 × 19.6 mm
Weight	~112 g
Operating Temperature	-40°C to +75°C

System Overview

Physical Connection

The A2E-TRI sensor connects using a single **M12 D-coded Ethernet cable**, which provides both:

- Power (PoE)
- Data communication

No additional wiring is required.

Data Flow

The sensor operates as a streaming device on a standard Ethernet network:

1. Sensor powers up via PoE
2. Network connection is established (DHCP or static IP)
3. A client connects via TCP or UDP
4. The sensor streams vibration data continuously

Communication Architecture

Function	Protocol	Description
Configuration	UDP	HMAC-MD5 authenticated commands
Raw Sample Streaming	TCP or UDP	Selectable per stream via <code>NetworkConfigV4.sample_transport</code>
FFT Streaming	TCP or UDP	Selectable per stream via <code>NetworkConfigV4.fft_transport</code>
Discovery	mDNS	Zero-configuration device detection

Data Format (Protobuf)

The A2E-TRI uses **Google Protocol Buffers (protobuf)** for all communication.

Benefits

- Compact binary format
- Fast parsing on embedded systems
- Open and cross-platform
- Easy integration in Python, C, C++, and other environments

Data Packet Structure

Each data packet is a two-part protobuf Message :

1. **Header** — timestamp, device identity, stream identifier, sequence number, per-axis metadata (data type, unit, scaling), and a stream-specific sub-message (`FrameStream` for raw acceleration, `FFTStream` for FFT).
2. **Payload** — a separate `bm_Payload` message carrying the raw sample or FFT bytes, formatted per the metadata block (typically int16 LE for raw samples, int16 Q15 or float32 for FFT magnitudes).

See [Message Structure](#) and the `FrameStream` / `FFTStream` field tables below for the concrete fields. Sensor payloads are variable-sized; do not assume 256 samples per packet.

Decoding the Data

To decode the data stream:

1. Open a TCP or UDP socket to the sensor (per the transport configured for the stream)
2. Receive protobuf binary data
3. Deserialize using a protobuf library

The `.proto` schema and pre-compiled bindings are published at:

<https://github.com/jacoblilsys/lillie-protobuf>

Clone the repo and use the schema definitions directly, or use the generated Python / C / C++ bindings.

Supported Environments

- Python (recommended)
- C / C++
- Embedded systems

Sensor Configurations

Different sampling frequencies and low/high pass filters are possible.

Decimation setting

Decimation reduces the effective sampling rate by averaging multiple consecutive samples.

The packet rate below is the number of protobuf `Message` payloads emitted per second (256 samples per packet). It is the same for TCP and UDP transports — the payload contents are identical, only the framing differs.

Setting	Description	Effective Output Rate	Packet rate
1	No decimation (raw 26.7 kHz)	26.7 kHz	105 / second
2	Average every 2 samples	13.35 kHz	52 / second
4	Average every 4 samples	6.675 kHz	26 / second
8	Average every 8 samples	3.333 kHz	13 / second
16	Average every 16 samples	1.667 kHz	6.5 / second
32	Average every 32 samples	0.833 kHz	3.2 / second
64	Average every 64 samples	0.416 kHz	1.6 / second
128	Average every 128 samples	0.208 kHz	0.8 / second
256	Average every 256 samples	0.104 kHz	0.4 / second

Low-pass or High-pass filter options

An additional high or low pass filter can be enabled **in the accelerometer**. These two are mutually exclusive: the part has a single filter output path and a single shared cutoff register, so enabling the high-pass necessarily disables the

low-pass. Because the low-pass also serves as the anti-alias filter ahead of the decimation stage above, the low-pass is usually the one you want.

To remove DC or gravity *while keeping the low-pass*, use the separate [DC Removal](#) setting, which is implemented in firmware and is independent of the filter selected here.

Setting	Bandwidth
ODR DIV 1	6.3 kHz
ODR DIV 4	1.56kHz
ODR DIV 10	0.63kHz
ODR DIV 20	0.315kHz
ODR DIV 45	0.140kHz
ODR DIV 100	63Hz
ODR DIV 200	32Hz
ODR DIV 400	16Hz
ODR DIV 800	8.9Hz

On-board FFT

The A2E-TRI computes real-input FFTs on the device itself. Only the resulting spectrum is shipped over the network — the host does not need to run its own transform.

Independent from the raw stream

The FFT stream is a **separate data channel** from the raw acceleration stream. Either can be enabled, disabled, or reconfigured without affecting the other:

Config field (NetworkConfigV4)	Purpose
data_stream	Enable / disable the raw acceleration stream
fft_stream	Enable / disable the on-board FFT stream
sample_transport	TCP or UDP for the raw stream
fft_transport	TCP or UDP for the FFT stream

A common configuration is to run **raw at TCP** (for lossless capture) and **FFT at UDP** (for low-latency spectral monitoring) — or vice versa — on the same device at the same time.

Configuration

Parameter	Enum	Values
FFT size	fft_size	256 / 512 / 1024 / 2048 samples
Precision	fft_precision	Q15 (int16) or Float32
Window	fft_window	Hann. <i>Not configurable</i> — fixed in firmware and reported in the FFT stream header.

Parameter	Enum	Values
Result format	fft_result	Magnitude. <i>Not configurable</i> — fixed in firmware and reported in the FFT stream header.
Axis selection	axes	Per-axis modes emit one spectrum per axis; vector modes emit one combined-magnitude spectrum

Number of bins per axis is `fft_size / 2`. Hop is `fft_size / 2` (50 % overlap), so the effective FFT frame rate is `2 × sample_rate / fft_size`.

FFT output rate

At the native 26.7 kHz sample rate (no decimation), the on-board FFT emits at these rates:

fft_size	Bins	FFT frames / second	Notes
256	128	~208	Highest time resolution, coarsest frequency bins
512	256	~104	
1024	512	~52	
2048	1024	~26	Highest frequency resolution, lowest frame rate

Rates scale linearly with the effective sample rate — halve the ODR, halve the FFT rate.

Payload size and multi-chunk assembly

The FFT payload size depends on precision, `fft_size`, and the number of active axes (per-axis or single-vector). Payloads larger than the ~1388-byte per-datagram limit are split into multiple UDP chunks:

Precision	fft_size	Approx. payload	UDP chunks
Q15	256 – 2048	≤ 1.4 kB	1
Float32	256	~1.7 kB	2
Float32	512	~3.3 kB	3

Precision	fft_size	Approx. payload	UDP chunks
Float32	1024	~6.4 kB	5
Float32	2048	~12.5 kB	10

Over TCP: each FFT payload is delivered as a single length-prefixed `Message` — the TCP layer handles fragmentation and reassembly. The host just reads the 4-byte length and then that many bytes.

Over UDP: each chunk carries the 12-byte `udp_chunk_hdr_t` (see [UDP Data Transport](#)). The host reassembles by grouping chunks with the same `packet_id`, ordering them by `chunk_idx`, and concatenating their payloads once all `chunk_count` chunks have arrived. Partial payloads should be dropped after a short timeout (100 ms typical) — UDP does not retransmit.

Coexistence with the raw stream

Running raw and FFT simultaneously at their highest settings (raw 3-axis at 26.7 kHz + FFT Float32 at 2048 samples over UDP) is supported. The firmware paces multi-chunk FFT sends with a 1ms yield between chunks so raw sample datagrams keep flowing while a large FFT is being transmitted.

Security Features

Secured by HMAC-MD5

Authentication for UDP configuration messages.

Encrypted secure boot

Ensures only trusted firmware is executed.

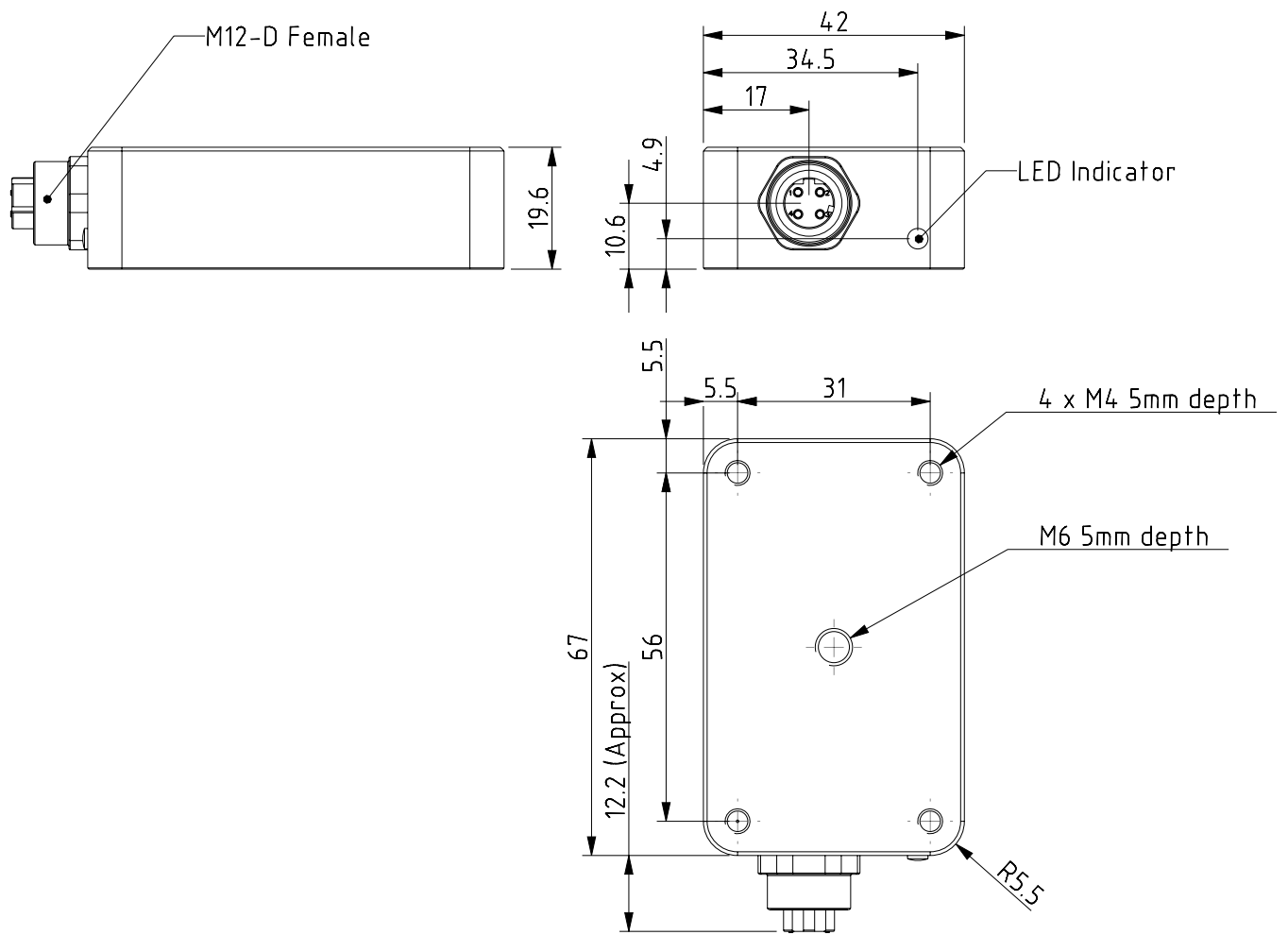
Mechanical Drawing

Feature	Detail
Mounting	4 × M4 threaded holes, 5 mm depth
Stud Mount	M6 thread
Connector	M12-D Female
Drawing Reference	ASSY_COMPLETE_REV4, Rev 1

All dimensions are in millimeters. Refer to the drawing above for full dimensional detail.

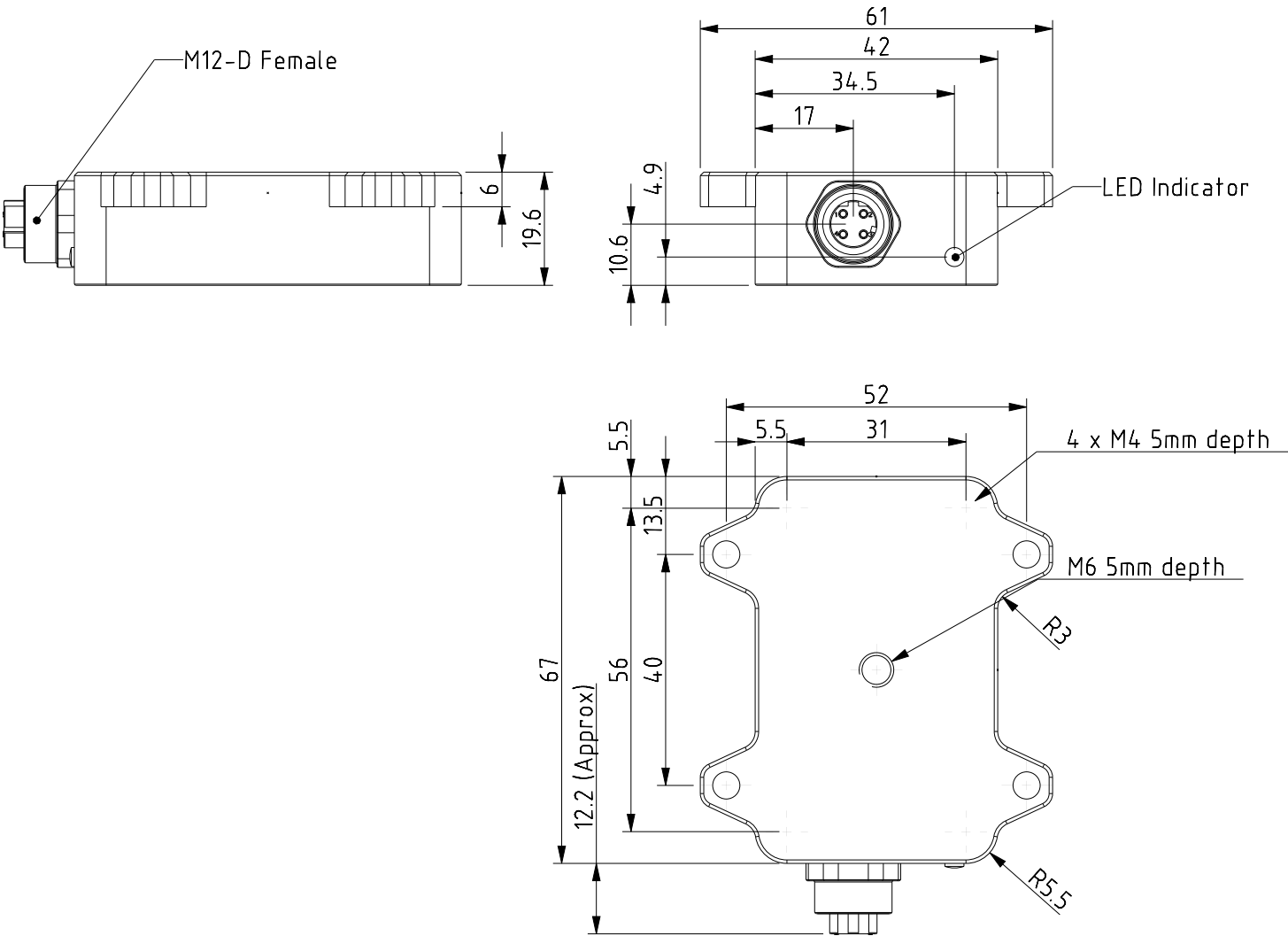
A2E-TRI

Normally mounted from the bottom with 4xM4 screws or a single M6.



A2E-TRI-F

Normally mounted from the top with 4xM4



Connector & Pinout

The A2E-TRI uses an **M12 D-coded Female** connector per IEC 61076-2-101, the standard for Industrial Ethernet.

Pin	Signal	Description
1	TD+	Transmit Data +
2	RD+	Receive Data +
3	TD-	Transmit Data -
4	RD-	Receive Data -

Power over Ethernet (IEEE 802.3af/at) is delivered over the same four data pins. No separate power wiring is required.

Performance Characteristics

Parameter	Value
Full Scale Range	$\pm 2g$ / $\pm 4g$ / $\pm 8g$ / $\pm 16g$ (selectable)
Native Sampling Rate	26.7 kHz
Output Resolution	16-bit signed integer
Sensitivity ($\pm 2g$)	~ 0.061 mg/LSB
Noise Density ($\pm 2g$)	~ 75 $\mu g/\sqrt{Hz}$ typical / ~ 60 $\mu g/\sqrt{Hz}$ single axis mode
Operating Bandwidth	Up to 6.3 kHz (configurable via filter settings)
Axes	3-axis (X, Y, Z), individually or jointly selectable
Internal Temperature Sensors	2 $\times$ board sensors + 1 $\times$ MCU core

Sensitivity scales with full-scale range: higher ranges reduce sensitivity proportionally. Noise density is specified at the $\pm 2g$ setting; it increases at higher full-scale ranges.

Data Throughput

Each TCP packet carries 256 samples per active axis as 16-bit integers (2 bytes/sample).

ODR Divider	Effective Rate	Data Rate (3-axis)	Data Rate (1-axis)	Packets/sec
1	26.7 kHz	~160 KB/s	~53 KB/s	~104
2	13.35 kHz	~80 KB/s	~27 KB/s	~52
4	6.675 kHz	~40 KB/s	~13 KB/s	~26
8	3.333 kHz	~20 KB/s	~6.7 KB/s	~13
16	1.667 kHz	~10 KB/s	~3.3 KB/s	~6.5
32	833 Hz	~5 KB/s	~1.7 KB/s	~3.3
64	417 Hz	~2.5 KB/s	~0.8 KB/s	~1.6
128	208 Hz	~1.2 KB/s	~0.4 KB/s	~0.8
256	104 Hz	~0.6 KB/s	~0.2 KB/s	~0.4

Calculation basis: Data rate = effective_rate × num_axes × 2 bytes/sample. Packets/sec = effective_rate / 256.

Each TCP packet includes additional framing overhead: a 4-byte big-endian length prefix and a protobuf header (~40–60 bytes depending on metadata).

Startup & Configuration

Boot Sequence

The device always starts in **bootloader mode** first. This ensures that even if the main application firmware is corrupted, the device remains recoverable via firmware update. The bootloader waits for a TFTP firmware update for a configurable timeout (typically **12 seconds**), during which it acquires an IP address. If no update is received, the main application starts automatically.

1. **Power on** via PoE (48 V, IEEE 802.3af/at)
2. **RED LED** illuminates briefly (power-on indication)
3. **1 second delay** — EEPROM initialization, all LEDs off
4. **Bootloader** — acquires IP, waits for firmware update (~12 s timeout)
5. **Main application starts** — network and sensor initialization begins
6. **Orange LED solid** — waiting for Ethernet link / DHCP
7. **Orange LED blinking (2 Hz)** — link established but NTP or TCP server not yet reachable
8. **Green LED solid** — fully connected (NTP synced, TCP connected), ready to stream
9. **Green LED blinking (2 Hz)** — actively streaming sensor data

Network Initialization

On startup, the device builds a hostname from the last two bytes of its MAC address (e.g. `vs15105.local`) and starts the **mDNS** responder. The mDNS service announces a `_nw-config._udp.local.` service with TXT data including:

```
Host: vs15105.local    Mac: 02:a0:12:63:3b:01  IP: 192.168.0.112    Port: 56671
Sensor: A2E_TRI  Versions: HW=3  FW App=0x102F  FW Boot=0x17, Lillie Systems Ltd
```

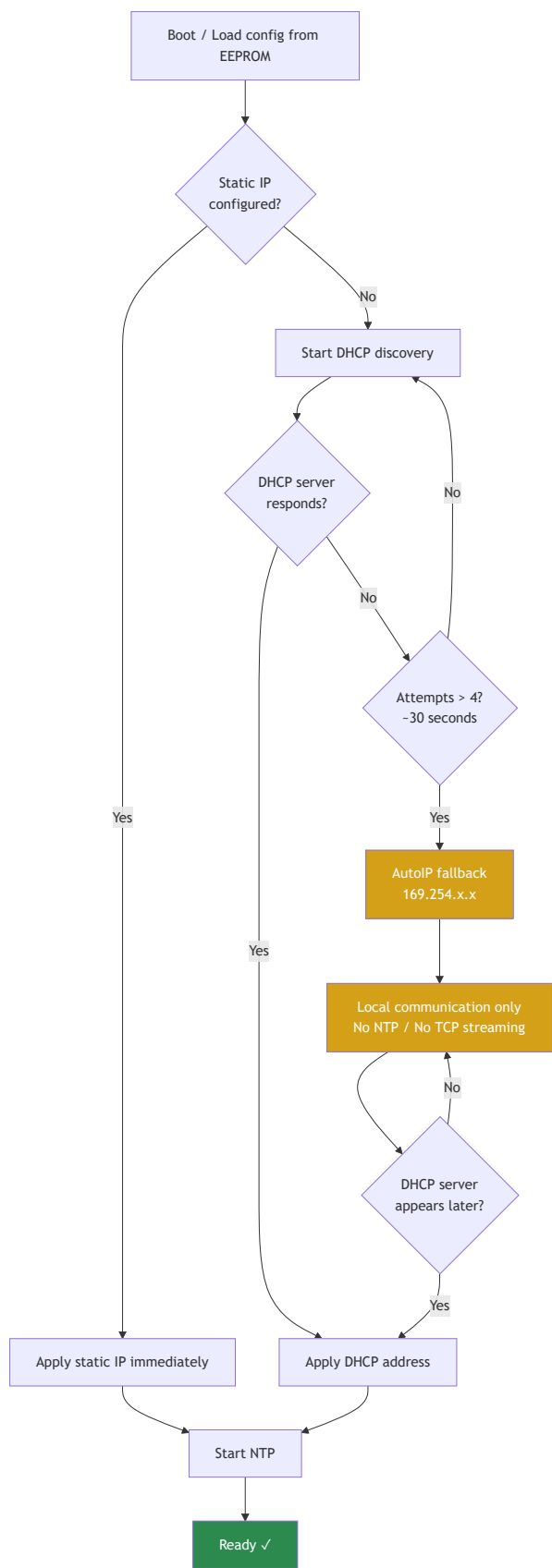
These fields allow discovery tools to identify units, firmware state, and differentiate between bootloader and application mode.

On boot, the firmware reads the network configuration from EEPROM. If a static IP address is configured, it is applied immediately and the device is ready. If no static IP is set (or DHCP is explicitly enabled), the device starts a DHCP discovery process with exponential backoff over approximately 30

seconds. If no DHCP server responds within 4 attempts, LwIP's AutoIP assigns a link-local address in the 169.254.x.x range (RFC 3927), allowing local communication without a DHCP server. **Both TCP and UDP data streams work on link-local addresses** — a receiver on the same L2 segment (also with a 169.254.x.x address) can connect and stream normally. In NTP_MODE_POLL (the default) SNTP is skipped on link-local, since no external time source is reachable; in NTP_MODE_LISTEN SNTP keeps running, since a broadcast time source may still be present on the segment (see [SNTP Time Synchronization](#)). If a DHCP server appears later, the device automatically transitions from link-local to the assigned address.

The bootloader (SBSFU) follows a similar DHCP flow but falls back to a hardcoded static IP rather than link-local.

IP Assignment Flow



Link State Changes

When the Ethernet link goes down and comes back up, the DHCP process restarts automatically. The LED shows solid orange while no IP is assigned.

Timing

Event	Timing
Static IP assignment	Immediate
DHCP attempt 1	~2 s
DHCP attempt 2	~6 s
DHCP attempt 3	~14 s
DHCP attempt 4	~30 s
AutoIP fallback	~30 s (after DHCP timeout)
First NTP poll	15 s after IP assigned
NTP poll interval	1 hour (configurable)

SNTP Time Synchronization

The sensor supports three SNTP operating modes, selected by the `ntp_mode` field in `NetworkConfigV4` :

ntp_mode	Behaviour
NTP_MODE_POLL (default)	Unicast SNTP client. The sensor sends outbound requests to <code>ntp_server_ip</code> at the configured polling interval.
NTP_MODE_LISTEN	Broadcast/multicast listener. The sensor never initiates outbound SNTP traffic — it only accepts SNTP broadcasts arriving on UDP port 123 from a time source on the same L2 segment. Useful on isolated LANs, on link-local (169.254.x.x) segments, and in installations where outbound NTP is blocked.

ntp_mode	Behaviour
NTP_MODE_DISABLED	SNTP is stopped entirely. No outbound requests, no broadcast listening. The device runs without time synchronisation; the <code>no_time_sync</code> flag stays set on outbound data messages.

Absent field (`NTP_MODE_UNDEFINED = 0`) is interpreted as `POLL` for backward compatibility with older host configurations.

Poll mode. Once the interface has a valid routable IP (static or DHCP), the SNTP client synchronizes the device clock to network time using the configured NTP server IP at the configured polling interval.

Listen mode. The sensor keeps SNTP running on **any** valid IP address including link-local — an isolated LL segment can still have a broadcast time source. The device's clock does not update until the first broadcast arrives. LED state reports `MISCFG_NO_NTP_OR_TCP` until then.

When a valid SNTP response is received (poll) or a valid SNTP broadcast is received (listen), the device converts seconds and fractional seconds into a UNIX timestamp with nanosecond precision and compares it to the local high-resolution hardware timer. If the difference exceeds ~1 ms, the local timer base is updated to maintain timestamp accuracy. The hardware real-time clock (RTC) is also updated to UTC.

On the first successful sync in poll mode, the SNTP polling interval changes to 1 hour (default) or to the user-configured interval.

Boot Modes

Mode	Description
Bootloader	Entered on every power-on. Acquires IP, waits for TFTP firmware update (~12 s). Also re-entered via <code>boot_now</code> command.
Application	Normal operating mode — sensor streams data. Starts automatically after bootloader timeout.

Configuration Persistence

All sensor and network settings are stored in non-volatile memory (EEPROM) and persist across power cycles. A `factory_reset` command restores all settings to defaults.

Settings are held as a single fixed-size, CRC-protected block with no version field. If a firmware update changes the layout of that block, the stored CRC no longer matches and every setting reverts to its factory default on the next boot — including the network configuration. Firmware 0x1032 does **not** change the layout, so upgrading to it preserves all existing settings.

API / Software

Device Discovery

Devices are discoverable on the local network via **mDNS** using the service type:

```
_nw-config._udp.local.
```

This enables zero-configuration discovery without prior knowledge of the sensor's IP address. The mDNS TXT record includes hostname, MAC address, IP, port, sensor type, hardware and firmware versions, and manufacturer.

UDP Control Protocol

All configuration and command messages are sent over **UDP port 56671** using multicast address **224.0.0.251**.

Request Frame

Field	Size	Description
header_size	4 bytes	Fixed: 30 (total header length)
MAC address	6 bytes	Target device MAC (big-endian)
request_id	4 bytes	Unique request identifier
HMAC-MD5	16 bytes	Authentication digest
Protobuf payload	Variable	Serialized Request message

HMAC calculation: HMAC-MD5(key=MD5(password), message=payload)

Response Frame

Field	Size	Description
header_size	4 bytes	Fixed: 8 (total header length)
request_id	4 bytes	Matching request identifier
Protobuf payload	Variable	Serialized Response message

Available Commands

Command	Description
get_sensor_info	Returns device type, HW/FW/bootloader versions, temperatures (board × 2, MCU core), UTC time, error bits, CPU usage, and a short diagnostic string
get_sensor_config	Returns current sensor configuration
set_sensor_config	Sets full_scale , axes , odr_div , filter type, filter cutoff, dc_removal , fft_size , fft_precision
get_network_config	Returns IP, netmask, gateway, server IP/port, NTP config, DHCP state, data_stream / fft_stream enables, sample_transport / fft_transport selections, ntp_mode
set_network_config	Configures network parameters, DHCP, NTP (including ntp_mode : poll / listen / disabled), per-stream enables, and per-stream transport (TCP or UDP)
command: stream_fft	Enable or disable FFT data streaming (immediate)
command: reset_device	Reboot the sensor
command: factory_reset	Restore all settings to factory defaults
command: boot_now	Bootloader mode only — the running application ignores this. To enter bootloader mode from the app, use reset_device and let the SBSFU pick up TFTP for the bootloader window.
new_app_password	Change the application HMAC password

Command	Description
new_factory_password	Change the factory HMAC password
calibration	Multi-step accelerometer calibration. Sub-commands: CALIBRATION_START_XY_ZERO_Z_NEG , CALIBRATION_START_X_NEG , CALIBRATION_START_Y_NEG , CALIBRATION_START_Z_NEG , CALIBRATION_SAVE .
get_factory_info	Returns serial number, calibration offsets, manufacturing date

Note on raw sample streaming: the raw acceleration stream is enabled via `set_network_config → data_stream` (not a top-level command). The `stream_data` protobuf field in `Command` is reserved and not consumed by the current application image.

Sensor Configuration Parameters

Full Scale Range

Value	Enum
±2g	ACCEL_FS_2G
±4g	ACCEL_FS_4G
±8g	ACCEL_FS_8G
±16g	ACCEL_FS_16G

Axis Selection

Two families are supported: **per-axis** modes emit one data channel per selected axis; **vector-magnitude** modes emit a single channel per packet holding the Euclidean magnitude of the selected axes ($\sqrt{x^2+y^2}$ etc.). Vector modes are computed on-device before transport, so bandwidth is 1 channel regardless of how many axes contribute.

Selection	Enum	Channels emitted
X only	AXIS_X	1 (x)
Y only	AXIS_Y	1 (y)
Z only	AXIS_Z	1 (z)
X + Y	AXIS_XY	2 (x, y)
X + Z	AXIS_XZ	2 (x, z)
Y + Z	AXIS_YZ	2 (y, z)
X + Y + Z	AXIS_XYZ	3 (x, y, z)
X, Y vector	AXIS_XY_VECTOR	1 ($\sqrt{x^2+y^2}$)
X, Z vector	AXIS_XZ_VECTOR	1 ($\sqrt{x^2+z^2}$)
Y, Z vector	AXIS_YZ_VECTOR	1 ($\sqrt{y^2+z^2}$)
X, Y, Z vector	AXIS_XYZ_VECTOR	1 ($\sqrt{x^2+y^2+z^2}$)

Output Data Rate Divider

Divider	Effective Rate	Enum
1	26.7 kHz	ODR_DIV_1
2	13.35 kHz	ODR_DIV_2
4	6.675 kHz	ODR_DIV_4
8	3.333 kHz	ODR_DIV_8
16	1.667 kHz	ODR_DIV_16
32	833 Hz	ODR_DIV_32
64	417 Hz	ODR_DIV_64
128	208 Hz	ODR_DIV_128
256	104 Hz	ODR_DIV_256

Digital Filter

Selects the accelerometer’s internal filter. Only one may be active — see [Low-pass or High-pass filter options](#).

For DC/gravity removal that coexists with the low-pass, see [DC Removal](#).

Filter Type	Enum
None	FILTER_NONE
Low-pass 2nd order	FILTER_LOW_PASS2
High-pass	FILTER_HIGH_PASS
Slope filter	FILTER_SLOPE_FILTER

Filter Cutoff Frequency

Cutoff	Enum
6.66 kHz	CUTOFF_6p66_KHZ
2.66 kHz	CUTOFF_2p66_KHZ
1.33 kHz	CUTOFF_1p33_KHZ
590 Hz	CUTOFF_0p59_KHZ
260 Hz	CUTOFF_0p26_KHZ
130 Hz	CUTOFF_0p13_KHZ
60 Hz	CUTOFF_0p06_KHZ
30 Hz	CUTOFF_0p03_KHZ
None	CUTOFF_NONE

Note that in `FILTER_HIGH_PASS` mode the two highest settings behave identically: the accelerometer has no ODR/4 high-pass stage, so both `CUTOFF_6p66_KHZ` and `CUTOFF_2p66_KHZ` select the ODR/10 corner.

DC Removal

A **software** high-pass filter that removes the DC / gravity component from the samples. It is independent of the `Digital Filter` and `Filter Cutoff` settings above and may be enabled alongside them.

This exists because the accelerometer cannot do both. Its low-pass 2 and high-pass stages share a single output path and a single cutoff register, so selecting the hardware high-pass also disables the low-pass — and since the low-pass doubles as the anti-alias filter ahead of the decimator, giving it up is rarely acceptable. Note also that the hardware "high-pass" is not a DC blocker in any setting: its corners run from 2.67 kHz down to only 33 Hz.

Enum	Cutoff (−3 dB)	Settling to <1 %
DC_REMOVAL_OFF	— (default)	—
DC_REMOVAL_0p1_HZ	0.1 Hz	7.3 s
DC_REMOVAL_0p5_HZ	0.5 Hz	1.5 s
DC_REMOVAL_1_HZ	1 Hz	730 ms
DC_REMOVAL_2_HZ	2 Hz	370 ms
DC_REMOVAL_5_HZ	5 Hz	150 ms
DC_REMOVAL_10_HZ	10 Hz	75 ms

Behaviour worth knowing:

- **Cutoffs are absolute Hz and do not move with the decimation setting.**
This differs from the hardware filter, whose cutoffs are expressed as ODR/N and therefore shift whenever the output rate changes.
- **Single pole, −6 dB/octave.** Choose a corner well below the lowest frequency of interest: at `DC_REMOVAL_1_HZ` a 2 Hz component is still attenuated about 3 dB. For a slow-turning shaft, `0p1_HZ` or `0p5_HZ` is usually the right choice.
- **Applies to both the raw sample stream and the FFT stream**, which are fed from the same filtered samples. FFT bin 0 collapses to near zero when DC removal is enabled; it reports `|DC|` when disabled.
- **Default is OFF, and a firmware update does not change it.** DC removal must be enabled explicitly, so upgrading a deployed sensor never alters the

data it produces.

- **Automatically bypassed during a calibration run**, because offset calibration measures the DC mean it would otherwise remove.
- **Forced off when the hardware high-pass or slope filter is selected**, where it would have nothing to do but add a settling transient. The value is still stored and reported; it takes effect again when the hardware filter returns to `FILTER_NONE` or `FILTER_LOW_PASS2`.
- **In vector-magnitude axis modes** the filter is applied per axis before the magnitude is formed. The magnitude of a zero-mean signal is still non-negative, so it retains a non-zero mean of its own — and for single-axis vibration the magnitude is rectified, which doubles the apparent frequency. Vector modes combined with the FFT stream are not recommended when DC removal is enabled.
- **Changing the setting restarts sampling**, as every sensor setting does.
- The setting is **not signalled in the data stream header**, so record it from `get_sensor_config` alongside any capture you intend to analyse later.
- If the sensor is close to full scale on the gravity axis, removing the DC can make the *filtered* output clip where the raw input did not — the output is centred on zero, so it has less negative headroom. If this is a concern, widen the full-scale range.

FFT Size

The value is the FFT input block size in **samples**. The number of magnitude bins returned per axis is `fft_size / 2`. Hop size is `fft_size / 2` (50 % overlap), so the effective FFT rate is $2 \times \text{sample_rate} / \text{fft_size}$.

Samples	Bins	Enum
256	128	FFT_SIZE_256
512	256	FFT_SIZE_512
1024	512	FFT_SIZE_1024
2048	1024	FFT_SIZE_2048

FFT Precision

Selects the numeric format the FFT engine uses internally and on the wire.

Enum	Wire format per bin	Dynamic range	Notes
FFT_PRECISION_Q15	int16 (Q15 fixed-point)	~90 dB	Default. Half the payload size.
FFT_PRECISION_FLOAT32	32-bit IEEE-754 float	~144 dB	For low-level signals and detailed spectral analysis.

Float32 payloads are larger and, at `fft_size ≥ 512` , span multiple UDP chunks (see [UDP Data Transport](#)). Q15 payloads fit in a single chunk up to `fft_size = 2048` .

TCP Data Stream

The sensor streams data over TCP to the configured `server_ip:server_port` when the corresponding `*_transport` config is set to TCP (the default). For UDP transport, see [UDP Data Transport](#) — the protobuf `Message` payloads below are identical; only the framing changes.

Framing

Each TCP message consists of:

1. **4-byte big-endian length prefix** — size of the protobuf payload
2. **Protobuf `Message`** — serialized binary payload

Message Structure

Field	Description
msg_version	Protocol version
device_timestamp	Timestamp of first sample (google.protobuf.Timestamp)
device_id	MAC address as 6-byte identifier
device_sn	Device serial number
stream_uid	Unique stream identifier

Field	Description
sequence_number	Packet sequence for loss detection
flags	FLAG_STREAM_END, FLAG_NO_TIME_SYNC, FLAG_TEST_RUN

Payload: FrameStream (Acceleration Data)

Field	Description
frame_number	Cumulative frame count since stream start
frame_count	Number of sample tuples in this packet (nominal 256; may vary)
target_frequency_hz	Requested sampling rate
actual_frequency_hz	Achieved sampling rate
meta_data	Per-axis metadata: data type, axis, unit, numpy format, scaling
channel_id	Logical data channel identifier

The raw payload bytes follow the metadata layout: each axis is sent as a contiguous block of `frame_count` samples in the format specified by `meta_data` (typically 16-bit signed integer, little-endian).

Payload: FFTStream (Frequency Domain Data)

Field	Description
frame_number	Cumulative FFT frame count
frame_count	Number of FFT frames in this packet (currently always 1)
duration_ns	Time window duration in nanoseconds
fft_bins	Number of FFT output bins (= <code>fft_size</code> / 2)
hop_size	FFT hop size in samples (= <code>fft_size</code> / 2 , 50 % overlap)
fft_window	Windowing function. Only <code>WINDOW_HANN</code> is implemented; <code>HAMMING</code> and <code>BLACKMAN</code> are reserved.
fft_result	Result format. Only <code>FFT_MAGNITUDE</code> is implemented; <code>FFT_COMPLEX</code> is reserved.

Field	Description
actual_frame_rate_hz	Input sample rate feeding the FFT
fft_frame_rate_hz	Effective FFT output rate = $2 \times \text{actual_frame_rate_hz} / \text{fft_size}$ (one FFT per hop of $\text{fft_size}/2$)
channel_id	Logical data channel identifier
start_bin	Index of the first bin in the payload (currently 0)
meta_data	Per-axis metadata (same format as FrameStream)

UDP Data Transport

The raw sample stream and the FFT stream can each be routed over UDP instead of TCP. Selection is per-stream and persists in EEPROM:

Config field (NetworkConfigV4)	Values
sample_transport	FEATURE_DISABLED = TCP (default), FEATURE_ENABLED = UDP
fft_transport	FEATURE_DISABLED = TCP (default), FEATURE_ENABLED = UDP

Both streams share a single UDP socket to the configured `server_ip:server_port` . UDP is connectionless — no handshake, no per-stream acknowledgement, and no retransmission. The sensor will keep sending as long as the network interface is up, even if the receiver isn't listening.

Application-layer fragmentation header

Every UDP datagram is prefixed with a **12-byte packed header** so a receiver can reassemble payloads larger than the MTU and detect per-chunk loss:

Offset	Field	Size	Encoding	Description
0	packet_id	4	u32 LE	Monotonic per-payload counter (shared across raw and FFT streams). Wraps at 2^{32} .

Offset	Field	Size	Encoding	Description
4	chunk_idx	2	u16 LE	Zero-based index of this chunk within the payload.
6	chunk_count	2	u16 LE	Total chunks that make up the payload. Same on every chunk.
8	chunk_size	2	u16 LE	Bytes of protobuf payload in this datagram (excludes the 12-byte header).
10	total_size	2	u16 LE	Bytes of the complete protobuf payload (all chunks summed). Saturates at 0xFFFF.

The maximum protobuf payload carried in a single datagram is **1388 bytes** (1400-byte cap minus the 12-byte header), chosen to fit in one Ethernet frame without IP fragmentation.

Receiver reassembly

1. Read one datagram. Parse the 12-byte header.
2. If `chunk_count == 1`, forward `chunk_size` bytes of payload directly to the protobuf decoder.
3. Otherwise, buffer by `packet_id`. Accumulate `chunk_count` chunks by `chunk_idx`; when all have arrived, concatenate their payloads (in order) and decode. Discard any partial payload after a small timeout (100 ms is typical) — datagrams may arrive out of order or be dropped by the network.
4. Missing `chunk_idx` values within a completed `packet_id` indicate in-flight loss; report per-chunk loss statistics.

When multi-chunk fires

Raw acceleration payloads at 256 samples/packet fit in a single UDP chunk regardless of axis selection.

FFT payloads span multiple chunks based on precision and size:

Precision	fft_size	Approx. chunks
Q15	256 – 2048	1
Float32	256	1

Precision	fft_size	Approx. chunks
Float32	512	3
Float32	1024	5
Float32	2048	10

Throughput considerations

UDP has no back-pressure. Multi-chunk sends yield briefly (~1 tick, 1 ms) between chunks to let LwIP's pbuf pool drain — without this, back-to-back sendto's exhaust the pool at Release-build clock speeds and interior chunks are dropped. Effective per-datagram wire time on 100 Mbit is ~120 μ s, so the inter-chunk yield dominates and caps large-FFT throughput at approximately the natural FFT rate.

Response Status Codes

Code	Meaning
RESPONSE_STATUS_OK	Command executed successfully
RESPONSE_STATUS_INVALID_PARAM	Invalid parameter in request
RESPONSE_STATUS_INVALID_HMAC	Authentication failed (wrong password)
RESPONSE_STATUS_INVALID_FULL_SCALE_RANGE	Full scale value out of range
RESPONSE_STATUS_INVALID_ODR	ODR divider value out of range
RESPONSE_STATUS_INVALID_AXIS_MASK	Invalid axis selection
RESPONSE_STATUS_INVALID_FILTER_SETTING	Invalid hardware filter configuration, or invalid <code>dc_remove1</code> value
RESPONSE_STATUS_INVALID_NETMASK	Invalid subnet mask
RESPONSE_STATUS_INVALID_GATEWAY	Invalid gateway address
RESPONSE_STATUS_INVALID_NTP_INTERVAL	Invalid NTP sync interval

Code	Meaning
RESPONSE_STATUS_INVALID_DHCP_STATE	Invalid DHCP toggle value
RESPONSE_STATUS_INVALID_DATA_STREAM_STATE	Invalid data stream toggle value
RESPONSE_STATUS_INVALID_NTP_MIN_UPDATE_MS	NTP <code>min_ms_error_to_update</code> out of range
RESPONSE_STATUS_UNKNOWN_ERROR	Unspecified internal error

Environmental & Compliance

Parameter	Value
Operating Temperature	−40°C to +75°C
Protection Rating	IP67
Enclosure	CNC machined aluminum, fully potted
CE Marking	Yes

Applicable Standards

Standard	Description
IEC 61000-4-2	Electrostatic discharge immunity
IEC 61000-4-3	Radiated electromagnetic field immunity
IEC 61000-4-4	Electrical fast transient / burst immunity
IEC 61000-4-5	Surge immunity
IEC 61000-4-6	Conducted disturbances immunity
IEC 61000-4-8	Power frequency magnetic field immunity
IEC 61000-6-4	Emission standard for industrial environments (Class A & B)

LED Indicators

Status LED

LED State	Color	Pattern	Condition
No network	Orange	Solid	Ethernet link DOWN (no IP / waiting for DHCP)
Partial connectivity	Orange	2 Hz blink	Link UP but NTP unreachable or TCP not connected
Ready	Green	Solid	Link UP, NTP synced, TCP connected, streaming disabled
Streaming	Green	2 Hz blink	All services connected, sensor data streaming active
Firmware update	Green	2 Hz blink	Receiving OTA firmware via TFTP
Error	Red	Solid	Critical error — system halted

Blink timing: 2 Hz (500 ms period), 50% duty cycle (250 ms ON / 250 ms OFF).

Update firmware by TFTP

The firmware is updated via TFTP.

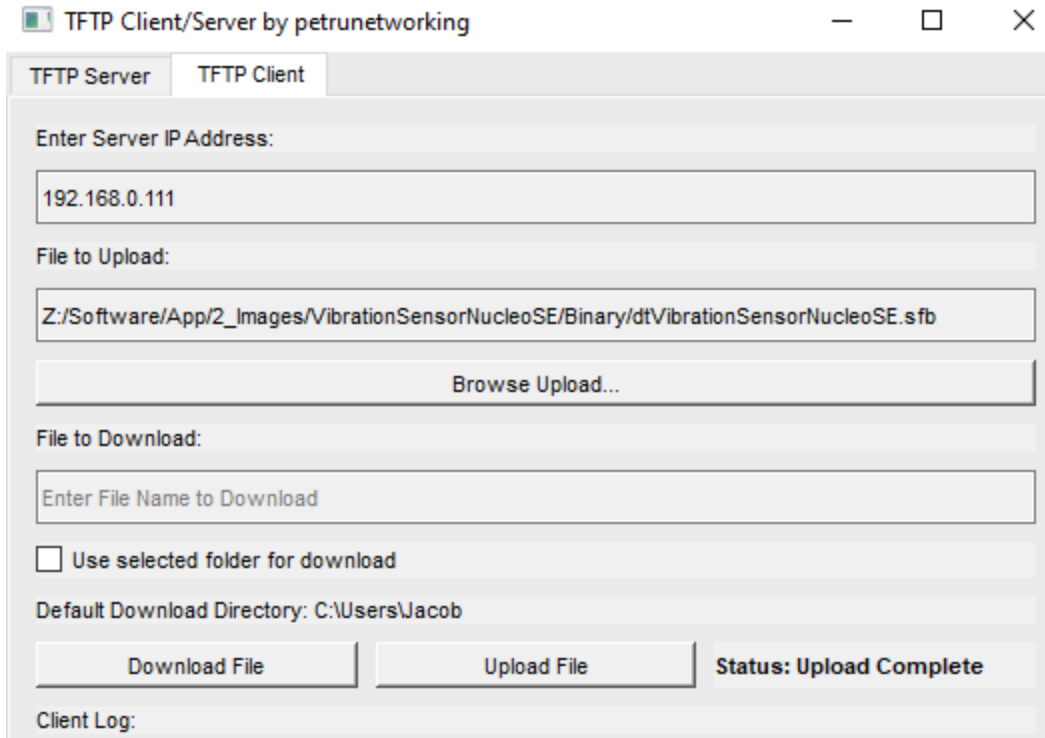
Python dependencies:

```
>pip install psutil pyqt5
```

To start the GUI run:

```
>python TFTP\_GUI\_Server.py
```

Select the TFTP Client tap:



TFTP Client/Server by petrnetworking

TFTP Server TFTP Client

Enter Server IP Address:

192.168.0.111

File to Upload:

Z:/Software/App/2_Images/VibrationSensorNucleoSE/Binary/dtVibrationSensorNucleoSE.sfb

Browse Upload...

File to Download:

Enter File Name to Download

☐ Use selected folder for download

Default Download Directory: C:\Users\Jacob

Download File Upload File Status: Upload Complete

Client Log:

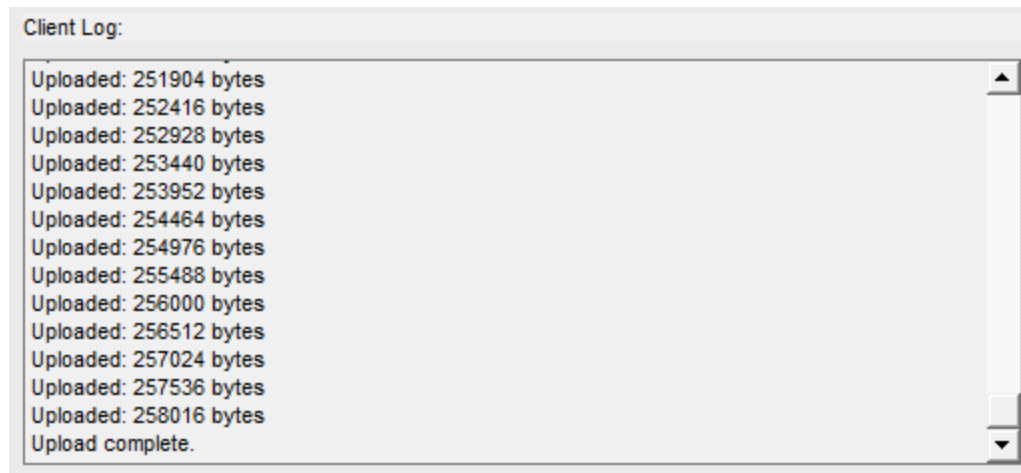
Enter the device IP address.

Click Browse Upload and find the provided file xxx.sfb in

Issue the reset device command using the console tool or via the browser-viewer app

```
>py console.py config-reset-device 192.168.0.111 test
```

Then click the Upload File button. The log should show the status:



After upload is complete it will take about 60s before the sensor is back online.

Ordering Information

Part Number	Description	Material
A2E-TRI-M12D-A	A2E-TRI accelerometer sensor unit	Aluminum
A2E-TRI-M12D-F-A	A2E-TRI accelerometer sensor unit with flanges	Aluminum

Revision History

Rev	Date	Author	Description
1.0	October 2025	Jacob Lillie	Initial release
2.0	27 March 2026	Jacob Lillie	Complete datasheet: mechanical drawing, connector pinout, API documentation, LED indicators, data throughput, startup sequence, performance characteristics, environmental & compliance
3.0	7 July 2026	Jacob Lillie	UDP data transport (per-stream TCP/UDP selection, 12-byte fragmentation header). Float32 FFT precision. Vector-magnitude axis modes. Link-local (169.254.x.x) streaming supported. <code>SensorInfo</code> extended (CPU usage, debug string). New response code for NTP min-update. FFT Size table corrected (samples vs bins). FFT window/result options scoped to what's implemented (Hann + Magnitude). Command table clarifies <code>stream_data</code> (via <code>set_network_config</code>) and <code>boot_now</code> (bootloader-only). Calibration expanded with sub-command enum. mDNS example updated to current firmware 0x102F.
3.1	21 July 2026	Jacob Lillie	Broadcast-based SNTP synchronisation. New <code>NtpMode</code> enum on <code>NetworkConfigV4</code> selects <code>POLL</code> (current), <code>LISTEN</code> (broadcast-only, no outbound), or <code>DISABLED</code> . Link-local behaviour updated: SNTP stays running on 169.254.x.x when <code>ntp_mode = LISTEN</code> . Firmware bumped to 0x1030.
3.2	4 August 2026	Jacob Lillie	Fixes on top of Rev 3.1, shipped as firmware 0x1031. <code>LISTEN</code> mode now actually accepts broadcast frames (the SNTP originate-timestamp check is <code>POLL</code> -only). <code>reset_device</code> no longer deadlocks the network stack instead of resetting — a regression present in 0x102F and 0x1030. <code>factory_reset</code> now reboots the device after restoring defaults, as always documented.

Rev	Date	Author	Description
3.3	5 August 2026	Jacob Lillie	New <code>dc_removal</code> setting on <code>SensorConfig</code> (tag 7, <code>DcRemoval</code> enum), shipped as firmware 0x1032. A software high-pass filter that removes DC/gravity while the accelerometer's low-pass 2 stays enabled — the two hardware filters share one output path and cutoff register, so they were previously an either/or choice. Cutoffs are absolute Hz and do not shift with the decimation setting. Defaults to <code>OFF</code> , so upgrading a deployed sensor does not change its data. Also fixes offset calibration, which derived every <code>x/y/z_ofs_usr</code> value from a mean that was too small by a factor of <code>odr_div</code> whenever decimation was enabled.
3.4	29 September 2026	Jacob Lillie	Added A2E-TRI-F version information