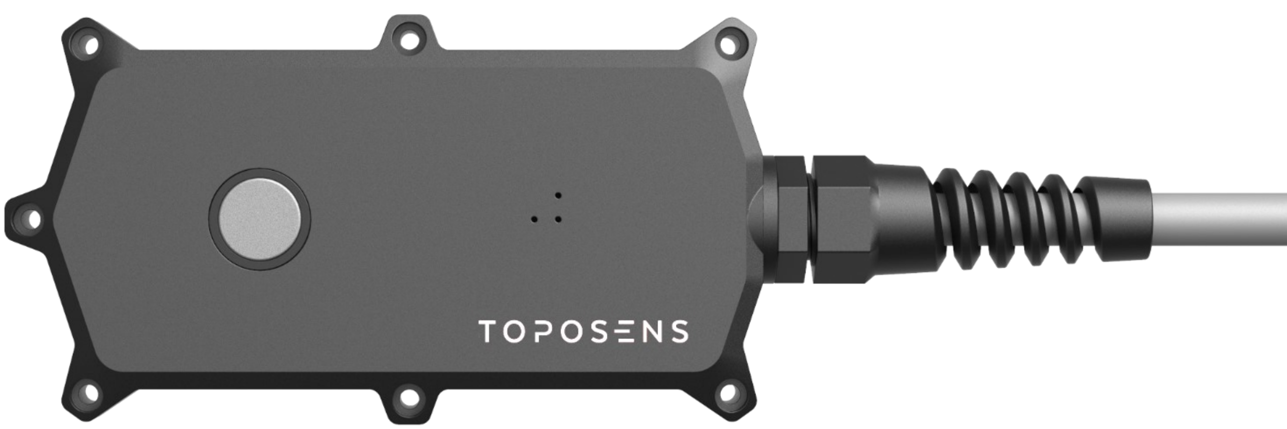
TOPOSENS

ECHO ONE Development Kit

3D Ultrasonic Echolocation and Ranging Sensor



Data Sheet 1.2 | 2021

The content of this document is subject to change without notice.
The most recent version of this document can be found at Toposens.com/Downloads

Content

1. Technical Data

Features	
Technology	3D Ultrasonic Echolocation and Ranging
Detection Range	Up to 3000 mm*
Field of View	up to ±80 ° Horizontal up to ±40 ° Vertical
Range Resolution	1 cm (without atmospheric disturbance) 5 cm nominal
Positional Resolution (Azimuth and Elevation)	±3 cm @ 100 cm Range Distance ±6 cm @ 200 cm Range Distance ±8 cm @ 300 cm Range Distance
Signal Source	40 kHz / 80 dB (@ 100 cm Distance)
Firmware Version	V 1.2.0

* Target: 75 mm pole centered in front of the sensor

Electrical Properties	
Supply Voltage (nominal / range)	12 VDC 7V - 28 VDC
Current Consumption (Average) @ 12V	170 mA
Peak Current Consumption @ 12V	500 mA

Performance	
Max. Target Number	max. 40 targets per frame
Response Time	< 100 ms
Startup Delay	< 5000 ms

Interface

CAN	ISO 11898-2:2016 / CAN 2.0A
Connection Type	DSub15 (F), Standard Density

Ambient Data

Ambient operating temperature	0°C to 55°C
Storage temperature	-20°C to 80°C
Electromagnetic compatibility	EN55032 / CISPR 22 Class A & B

General Notes

Note on use	Online Resources: https://toposens.com/members/

Dimensional Drawing

Outline Dimensions [LWH]	193mm x 69mm x 24mm
Weight	210 g (excl. Cable)
Enclosure Rating	IP 67 (Connector is not ingress protected)

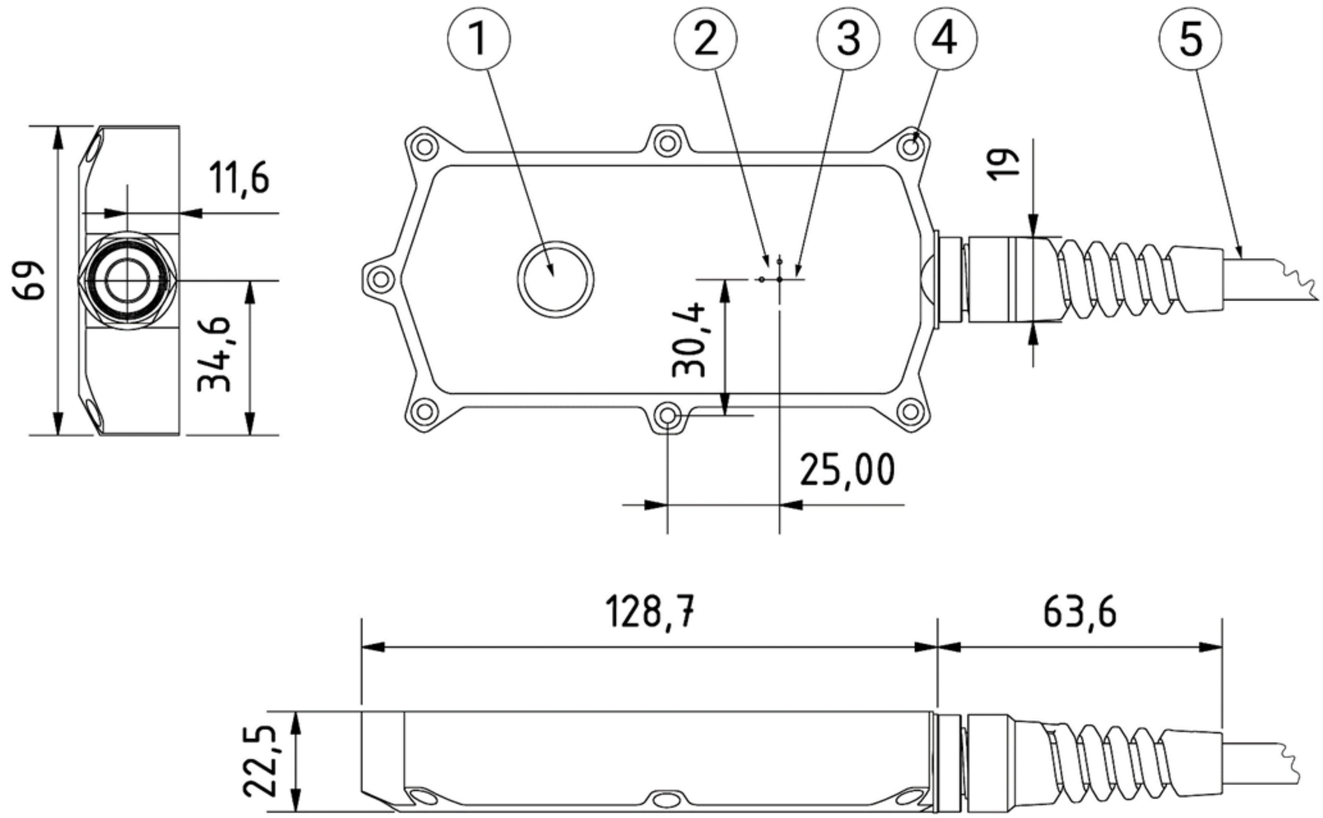


Figure 1 - Sensor Dimensions

1.	Transducer
2	Microphone receiver array
3	Acoustic Axis
4	3mm through holes, 13.5mm deep (7x), for mounting
5	Power connection / Data inputs and outputs / Cable (l= 800mm d=8mm) / PG9-Strain-Relief

Sensor Coordinate System

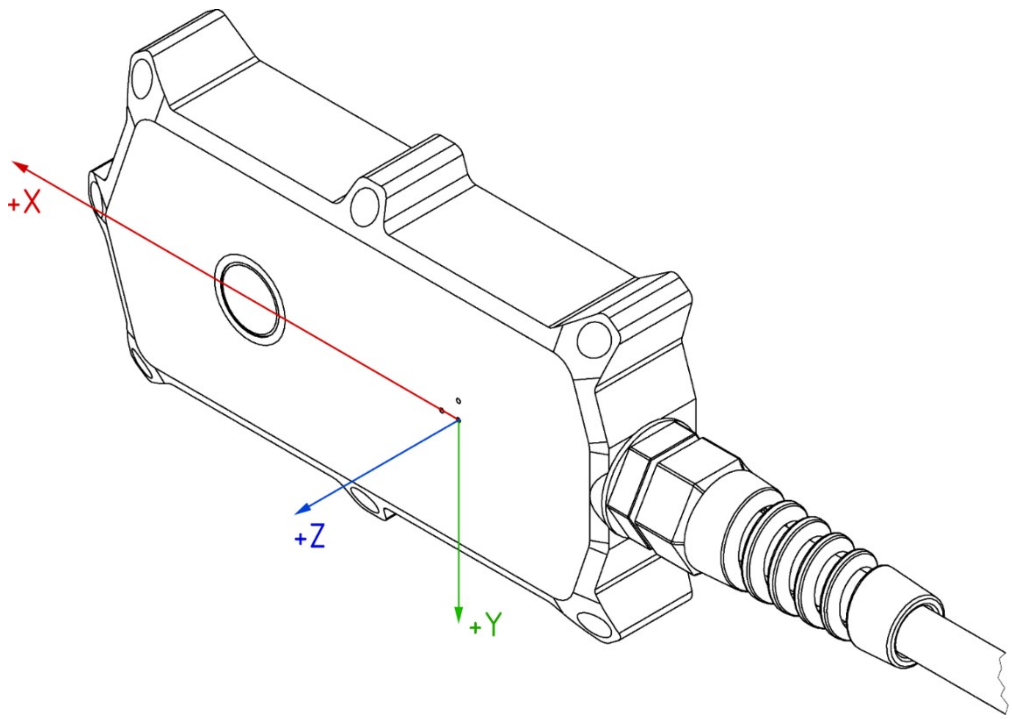


Figure 2- Sensor Coordinate System

Applications

Robotics

- Collision Avoidance
- Area Surveillance

Automotive

- Collision Avoidance
- Cocooning of Vehicle
- Power Side Door
- In-Cabin Detection

Other

- Autonomous Vehicles
- Presence Control

2. Operational Information

2.1 Overview

Toposens 3D Echolocation Technology works by combining the time of flight principle of conventional ultrasonic sensors with triangulation and advanced signal processing algorithms. A measurement cycle starts with the transmission of an ultrasonic pulse by the transducer element. This pulse travels through the air and is reflected by surrounding objects and surfaces. Several echoes are reflected back to the sensor which is equipped with a patented microphone array. Using the data gathered by the microphone array the 3D coordinates of the echo's origins are calculated and output at the end of the measurement cycle.

2.2 Theory of Operation

The figure below and corresponding descriptions show a basic example in 2D of the operating principle behind the sensor. [1] the transducer (red) sends out an ultrasonic pulse, [2] the wave is carried forward by the air molecules, [3] the wave is reflected by an object, [4] a portion of the echo is directed back to the sensor, [5] the echo is sequentially captured by the microphone array, arriving first at (a) the left microphone, and then at (b) the right microphone, [6] a 3D location of the echo's origin (light red) is determined from the signal's time-of-flight and the delay between microphones receiving the echo.

Figure 3 - Theory of Operation

2.3 Point Cloud Examples

Objects within the sensors field of view reflect the transmitted signal back towards the sensor. Due to the small wavelengths of ultrasound (below 1cm), a reflecting surface has to approximately face the sensor to be detectable. The surface area of the reflecting surface additionally defines the signal strength of the detected target.

With this prior knowledge, the following examples explain the expected targets for a complex object (e.g. a person) and for a less complex object (e.g. a pole).

Point cloud examples for a less complex scene

Less complex objects (such as walls and poles) are composed out a limited number of surfaces. This results in less points per object, as shown in the graphic below.

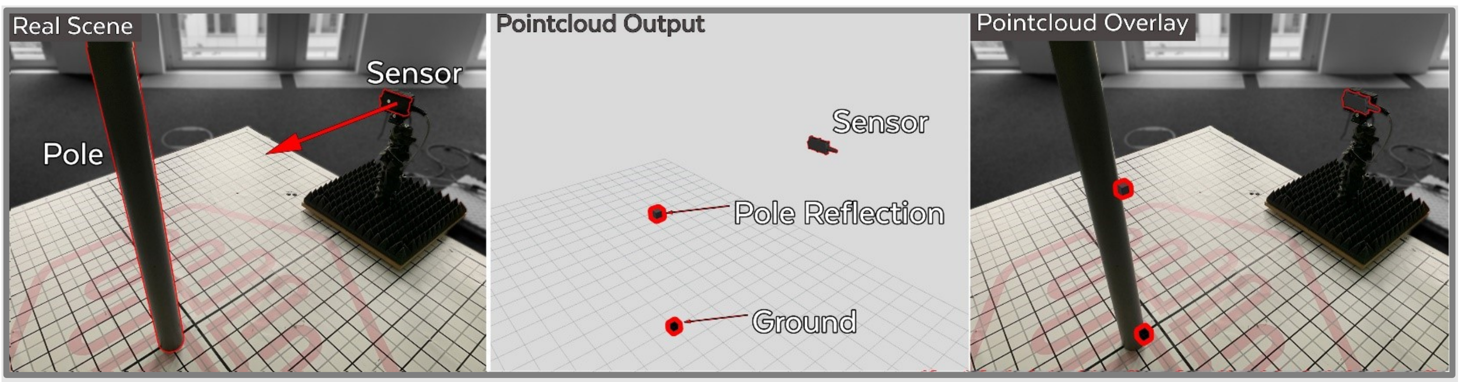


Figure 4 - Sensor Data Example 1

Objects which are positioned on the ground in front of the sensor can be detected reliably due to the formation of a retro reflector. This results in a reflection being detected at the position the pole touches the ground.

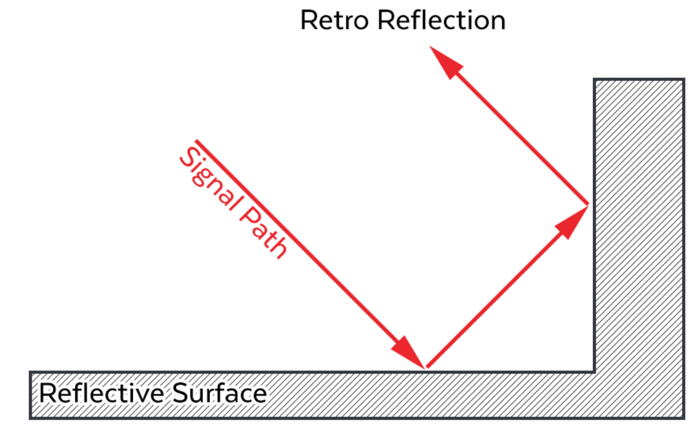


Figure 5 - Retro reflection of sound

Point cloud example for a complex scene

A complex object (such as a person) is a composition of multiple surfaces, forming the shape. The sensor perceives all surfaces facing the sensor, which are of a large enough area to reflect enough acoustic energy. This results in a target cloud as shown in the graphic below.

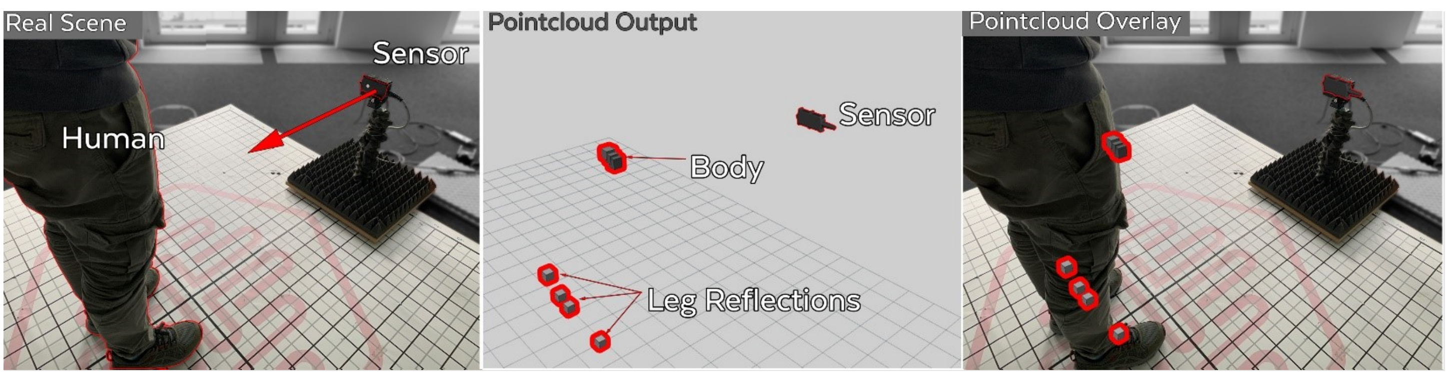


Figure 6 - Sensor Data Example 2

2.4 Field of View: Measurements

The field-of-view measurements are performed in a laboratory environment. The sensor is placed at a height of 50 cm above the ground plane. The sensor is mounted to a rotary platform which automatically rotates the sensor horizontally from -90° to +90°. The target object is placed at different distances from the sensor along the 0° Z axis position of the sensor. Each position is held for 100 frames. The expected spatial volume of the target position is monitored. Positions which have a detection rate >95% are plotted. To measure the vertical field of view, the sensor is rotated 90° about its z axis and the measurement is repeated.

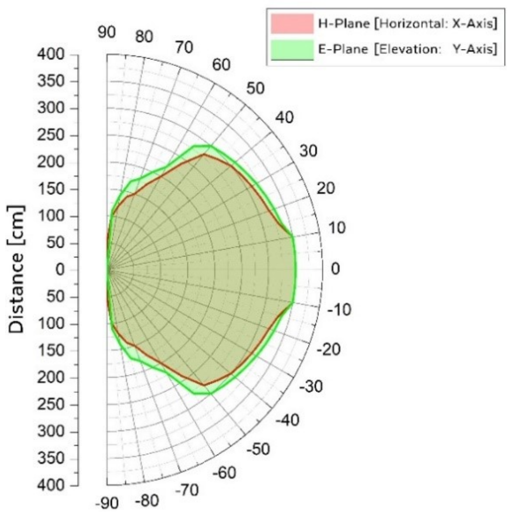


Figure 7 Plate-Target FOV Results

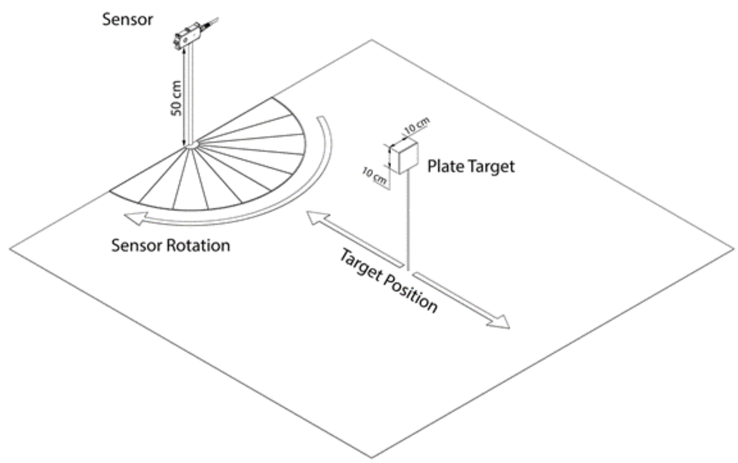


Figure 8 - Plate-Target FOV Measurement Setup

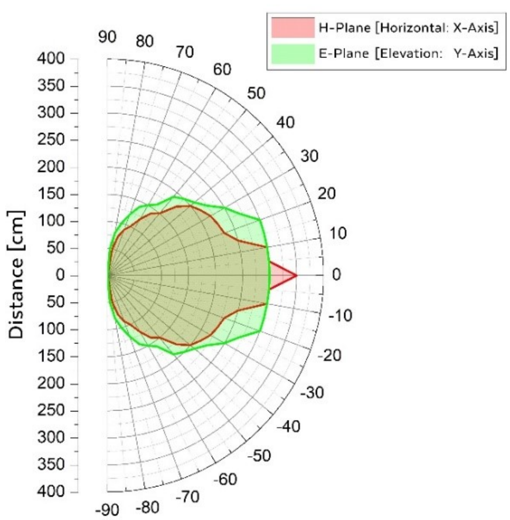


Figure 9 - Pole-Target FOV Results

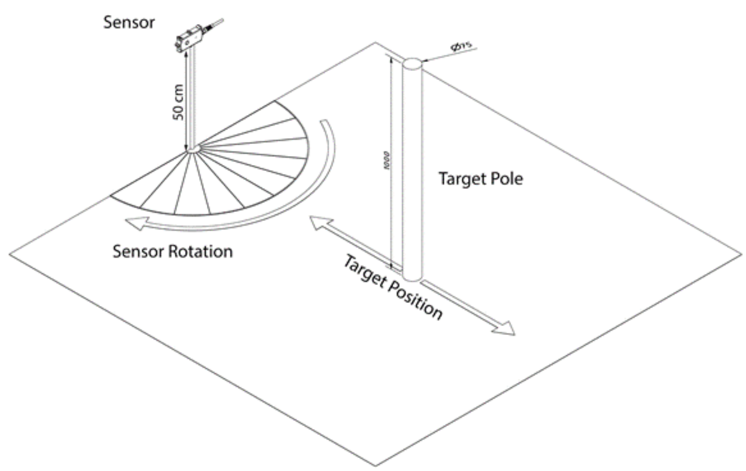


Figure 10 - Pole-Target FOV Measurement Setup

2.5 Field of View: Limitations

Surfaces not facing the sensor are visible to a certain degree. Depending on the signal strength of the reflection, a diffuse reflection can be detected and located. High acoustic frequencies, such as ultrasound, have a narrow diffuse reflection compared to lower frequency sound. The following measurement shows the reflection of the 10 cm x 10 cm plane. The target is moved parallel along the horizontal respectively along the vertical axis of the sensor.

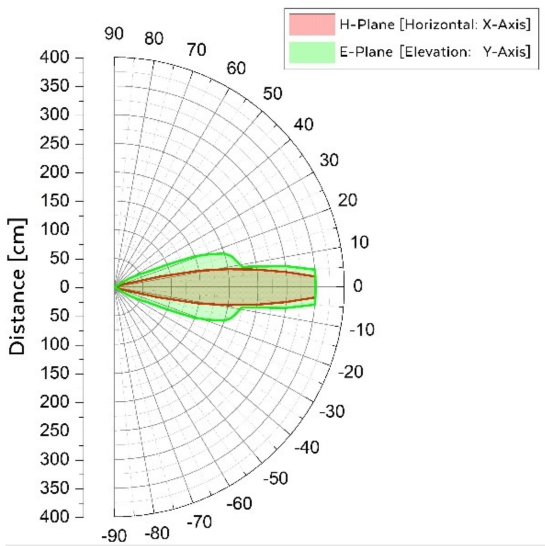


Figure 11 - Plate-Target Parallel FOV Results

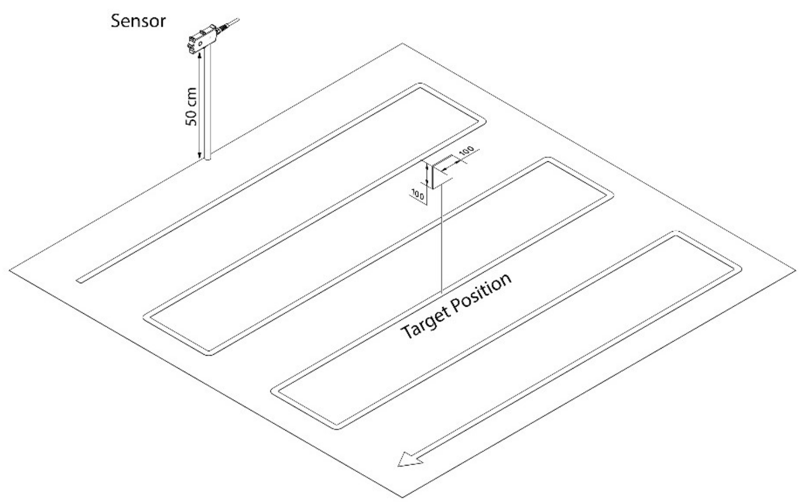


Figure 12 - Plate-Target Parallel FOV Measurement Setup

For high-frequency acoustic signals, the angle of incidence equals the angle of reflection. The reflection peak intensity is located on the middle axis (=angle of reflection) of the echo cone, defined by the diffuse reflection angle.

3. Connector Pinout

3.1 Sensor Connector

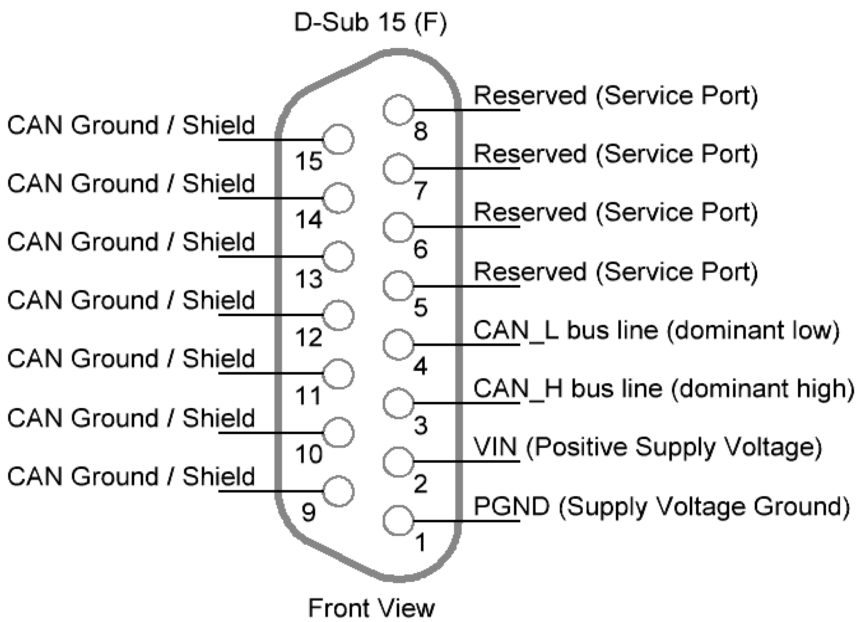


Figure 13 -D-Sub 15 Pin Allocation

⚠ **Do not remove the connector from the cable**. Removing the connector from the cable can result in electromagnetic shielding issues. It is recommended to use the included breakout box. The removal of the connector voids any warranty!

3.2 CAN Communication Cable Connectors

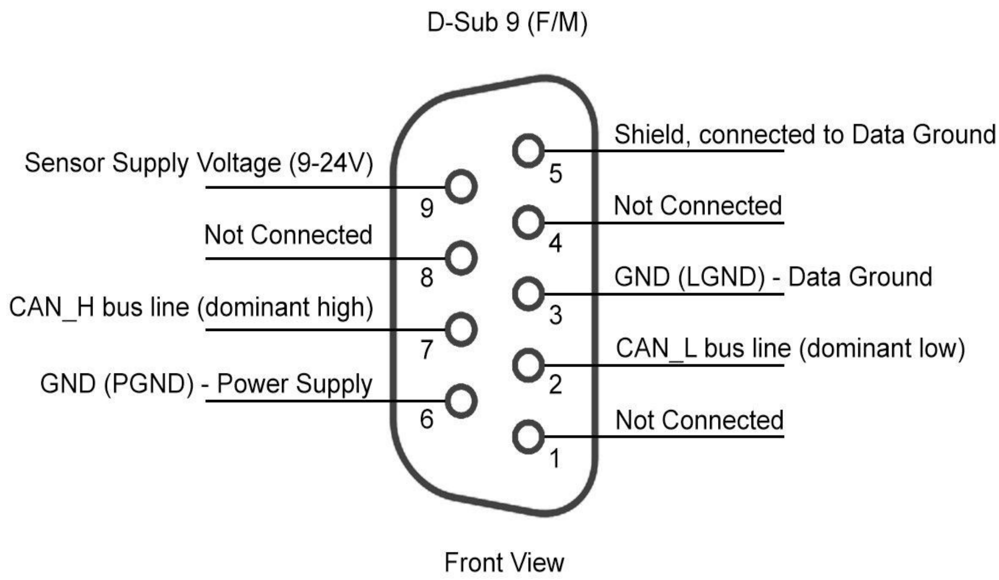


Figure 14 - D-Sub 9 Pin Allocation

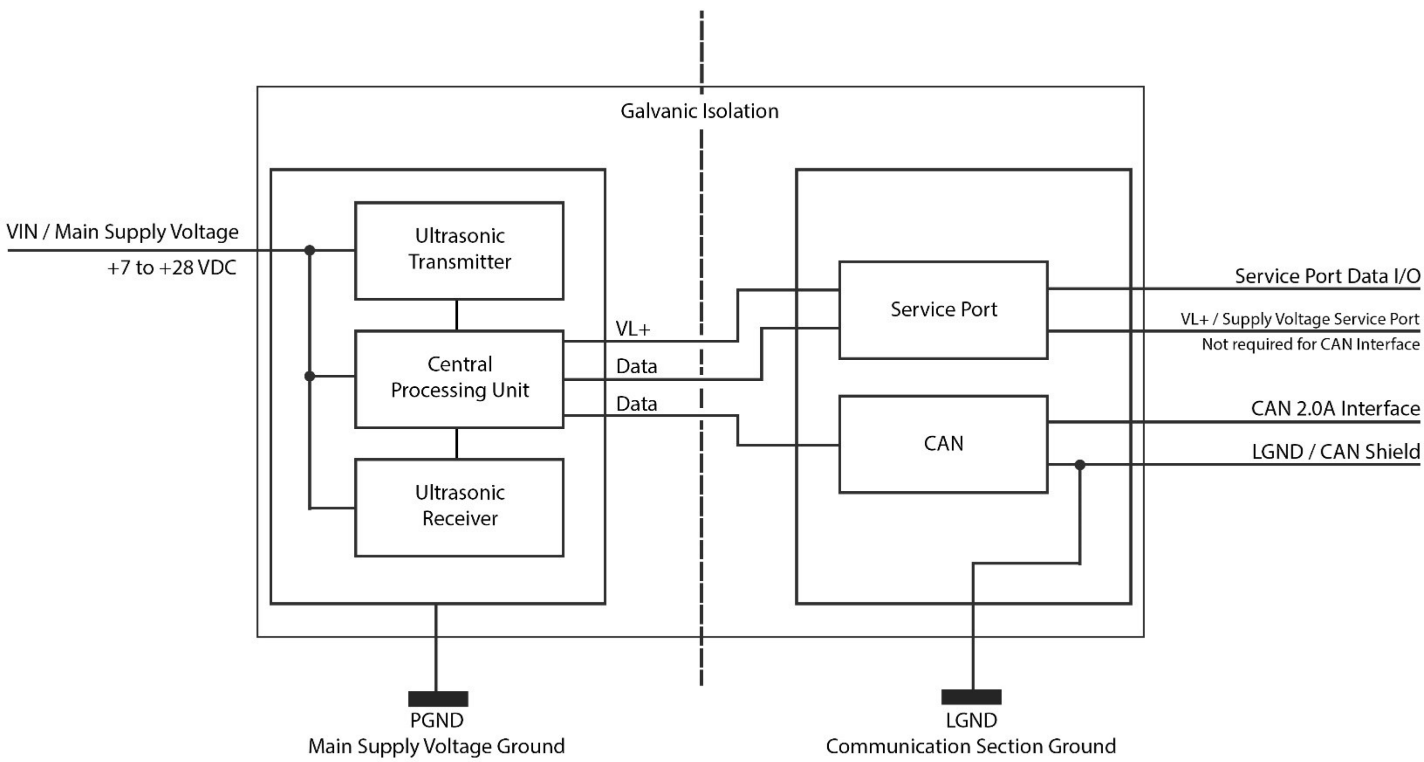


Figure 15: Block diagram of galvanic isolation

General Male Connector Pin Out (to Terminator)

Connector	Pin. No.	Name	Pin Description
D-Sub 9 (M)	1	n.c	Not Connected
	2	CAN_L	CAN Bus (dominant low)
	3	GND (LGND) – Data Ground	Cable Shield and CAN-Ground
	4	n.c.	Not Connected
	5	Shield	Cable Shield and CAN-Ground
	6	GND (PGND) – Power Supply GND	Power Ground
	7	CAN_H	CAN Bus (dominant high)
	8	n.c.	Not Connected
	9	Sensor Supply Voltage (9-24V)	Positive Supply Voltage (7 to 24 V)

General Female Connector Pin Out (to CAN-Master)

Connector	Pin. No.	Name	Pin Description
D-Sub 9 (M)	1	n.c	Not Connected
	2	CAN_L	CAN Bus (dominant low)
	3	GND (LGND) - Data Ground	Cable Shield and CAN-Ground
	4	n.c.	Not Connected
	5	Shield	Cale Shield and CAN-Ground
	6	n.c.	Not Connected
	7	CAN_H	CAN Bus (dominant high)
	8	n.c.	Not Connected
	9	n.c.	Not Connected

3.3 Power Connector

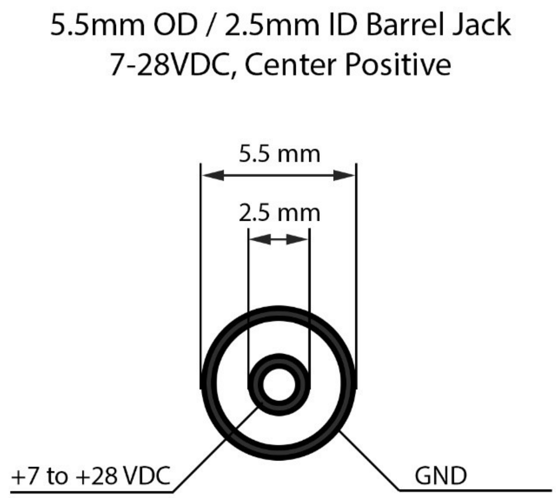


Figure 16: Power Jack

4. Installation

4.1 Installation Schematics

The sensor is equipped with 7 mounting holes, compatible with the M3 screw size. The sensor was designed to be mounted with DIN EN ISO 4762 or similar cylinder head M3 screws. The frontal surface area must be kept clear from obstructions. Installing the sensor in a recessed position requires a protrusion of 4.7 mm between the frontal surface of the sensor and the integration plane.

⚠ D-SUB15 Connector is not rated for wet or dusty environments

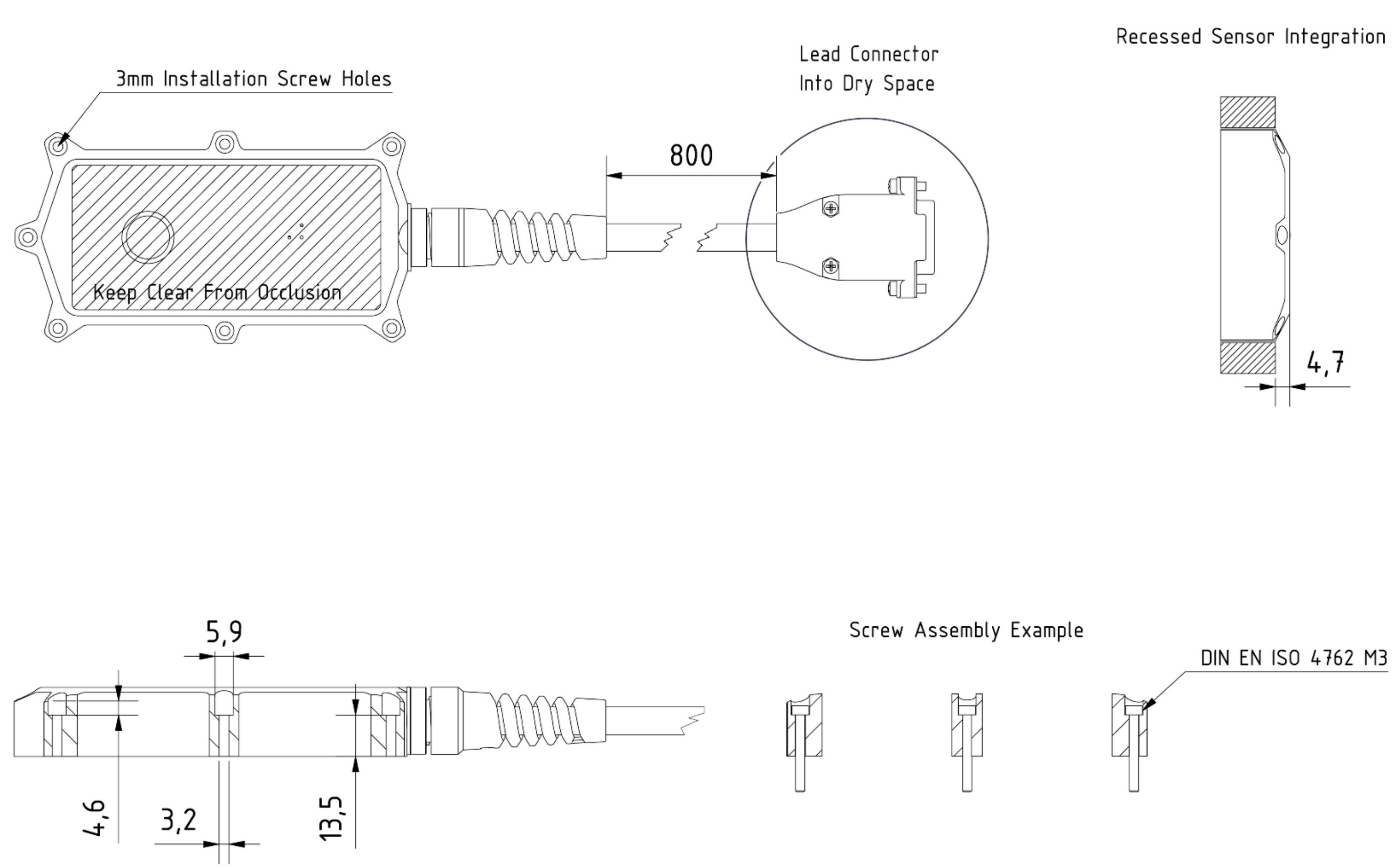


Figure 17 - Installation Schematics

4.2 Installation Guideline

The measurements of the drawing are centered around the coordinate origin of the sensor. When installing the sensor, the rugged cable strain relief must be considered. Please note the coordinate origin is positioned 22.5 mm above the installation plane, on the frontal surface of the sensor.

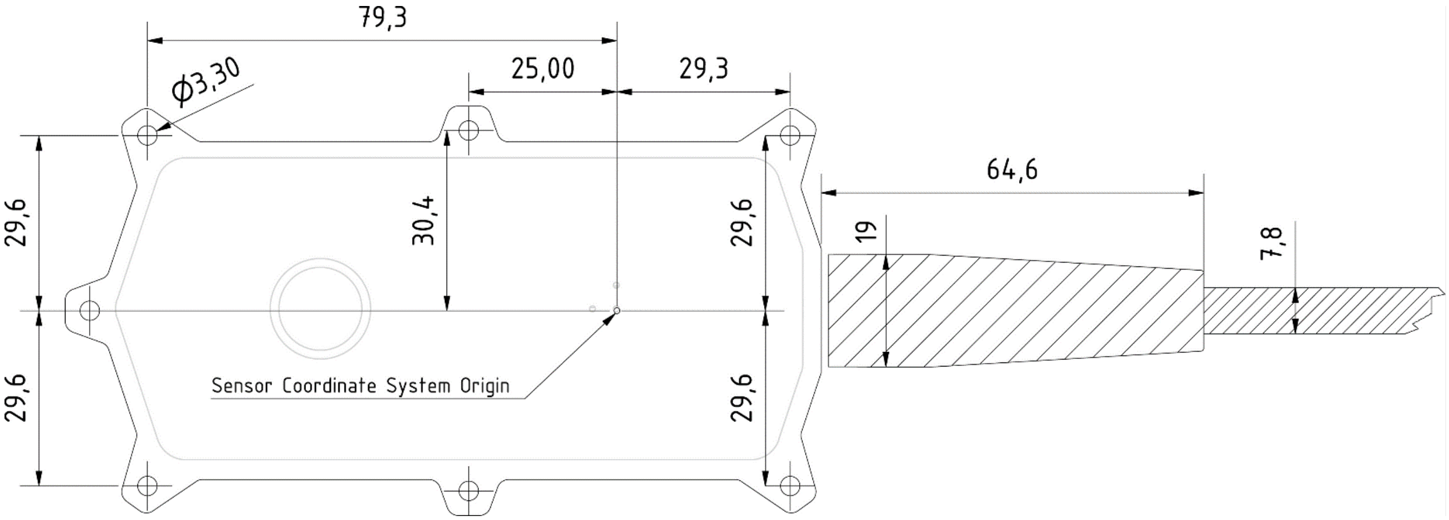


Figure 18 - Mounting Hole Positions

5. Application Information

No parts other than those provided by Toposens should be used. If third party parts (e.g. cables) are used, the function of the device cannot be guaranteed.
The sensor system is recommended to be connected to an exclusive CAN-Bus. If the sensor is connected to a non-exclusive CAN-Bus, interference free operation of the sensor and third-party devices cannot be guaranteed.

5.1 Sensor Connection Diagram

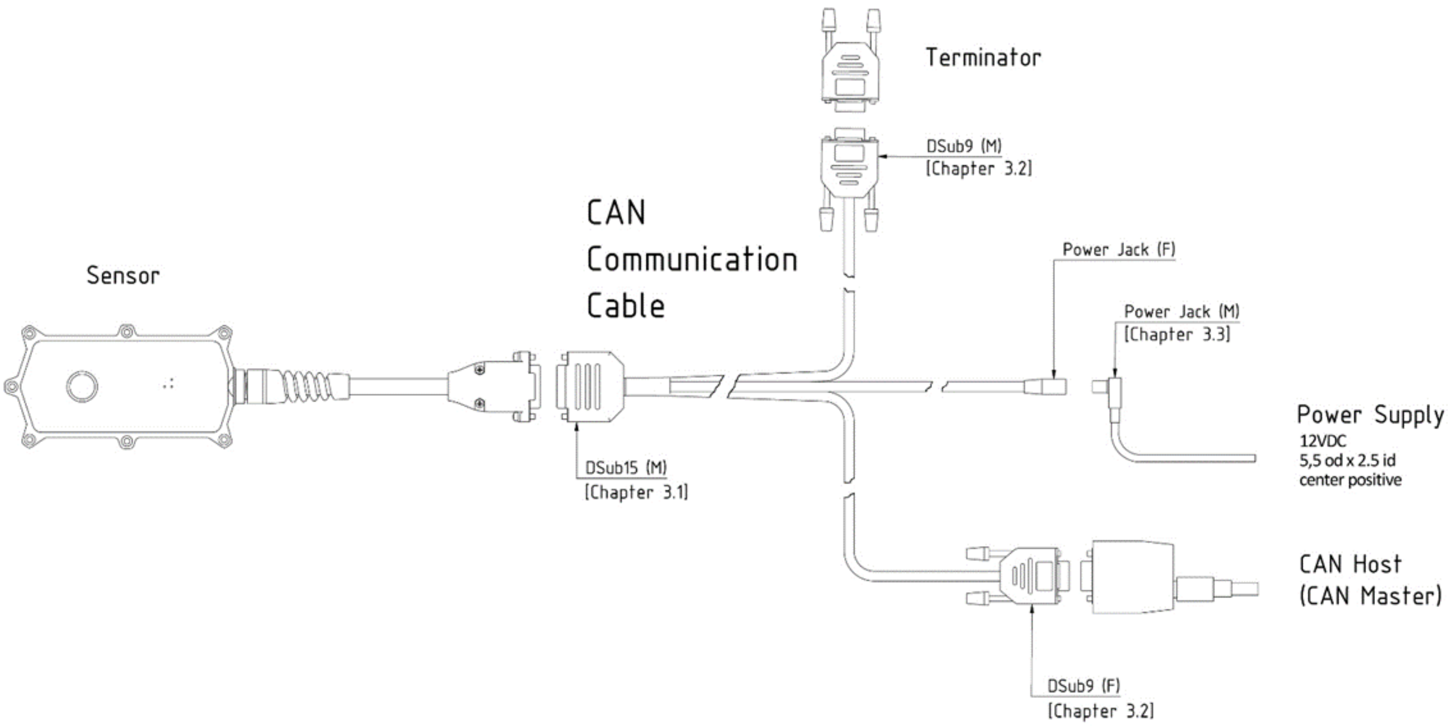


Figure 19 – CAN Communication Cable Connection Schematic

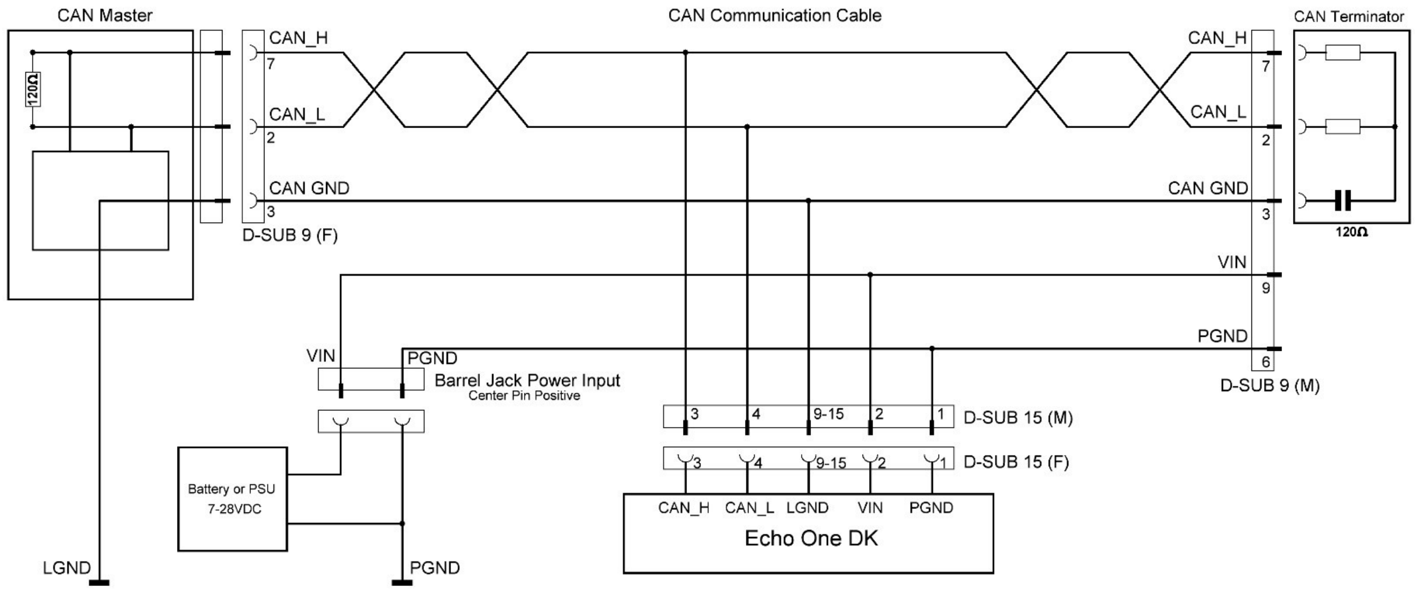


Figure 20: CAN Topology

5.2 CAN Cable Information

The CAN Data Rate used by the Sensor is set to 1 Mbps (default value). In order to preserve data integrity, we recommend the use of one of the following cable types, limiting the maximum length to below 25 meters and keeping the length of stubs below 0.3m. The Bus needs to be terminated at both ends with a 120 Ohms 0.5W Resistor or with the included CAN-Terminator.

<u>Recommended Cable Types:</u>

- Lapp UNITRONIC® BUS CAN 1X2X0,5

- Lapp UNITRONIC® BUS CAN FD P 2X2X0,25

- igus chainflex® CFBUS-001 1X2X0,25

- HELUKABEL 81286 1x2x0,22

- SAB S CB 625 1x2x0,20

5.2 Available Software

- **Toposens Sensor Library (see section 5.3)**

  Available via GitLab. Enables implementation of the Sensor into customer projects. Based on Linux-Socket-CAN. Open-source C library.

- **Firmware Update Tool (see section 5.4)**

  Enables updating of the sensor's firmware via Interface Adapter.

- **Toposens Visualizer → see resources section 8 for online ROS documentation**

  PC software for Sensor 3D raw-data visualization and capturing via Interface Adapter.

- **ROS Package → see resources section 8 for online ROS documentation**

  Robot-Operating-System-Packages for integration with ROS.

  Dockerfile hosted on Dockerhub for quick setup of our demo environment on Linux.

5.3 Firmware

The firmware of the sensor can be updated by downloading the newest firmware package and the Firmware-Update-Tool from https://toposens.com/members/. To update the Firmware, connect the sensor to the interface adapter. Connect the adapter to the included power supply and the micro-USB cable to the interface adapter and to the PC.

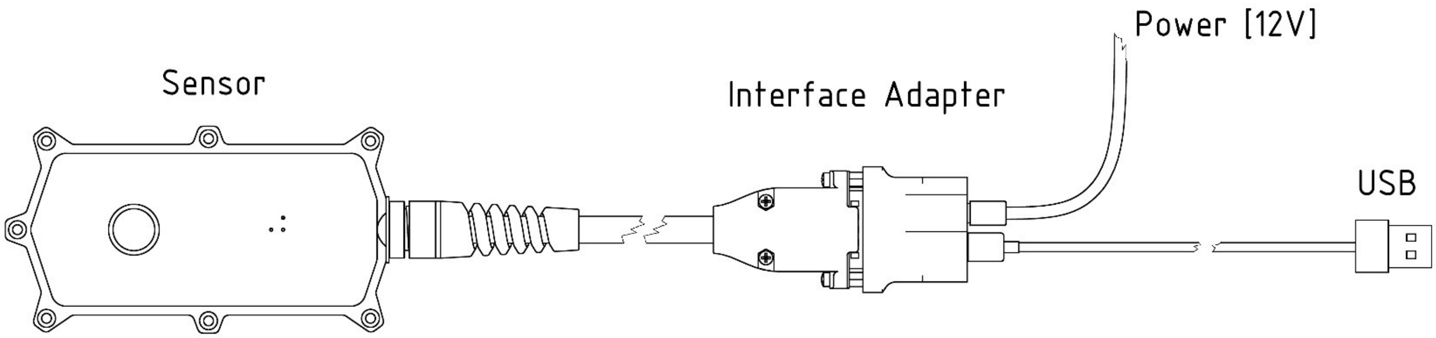


Figure 21 - Interface Adapter Connection Schematic

<u>Windows 10</u>

To run firmware-uploader, extract the downloaded archive and execute firmware-uploader.exe

<u>Linux</u>

Check if your current user is in the dialout/uucp group: "groups $USER"
If NOT add user to dialout/uucp group by using the following commands:

<u>Debian based:</u>
```
>sudo adduser $USER dialout
```

<u>Redhat based:</u>
```
>su -
>usermod -a -G dialout $USER
```

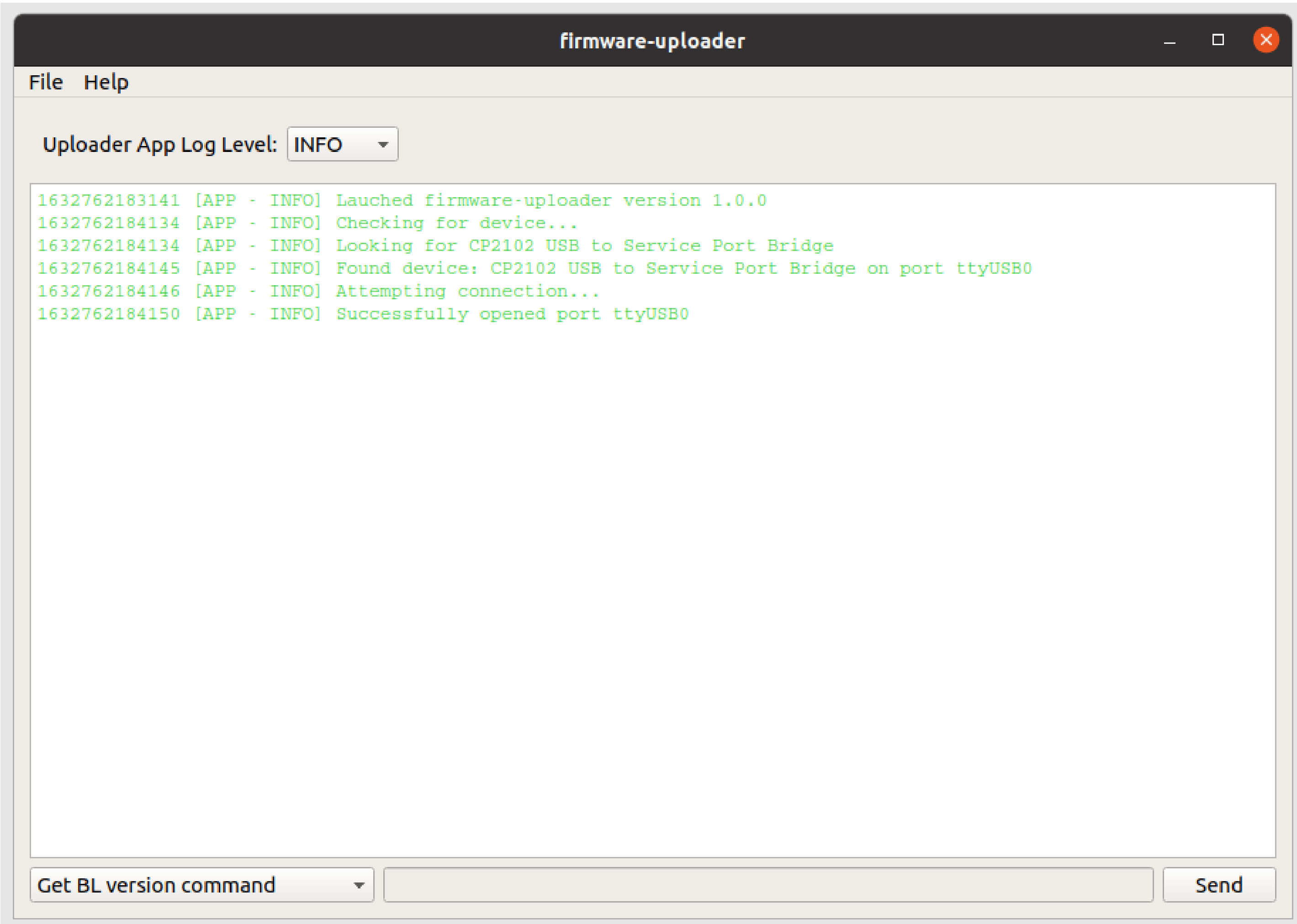
<u>Arch based:</u>
```
>sudo usermod -a -G uucp $USER
```

Re-login or reboot the system to make sure the user is in group dial out.

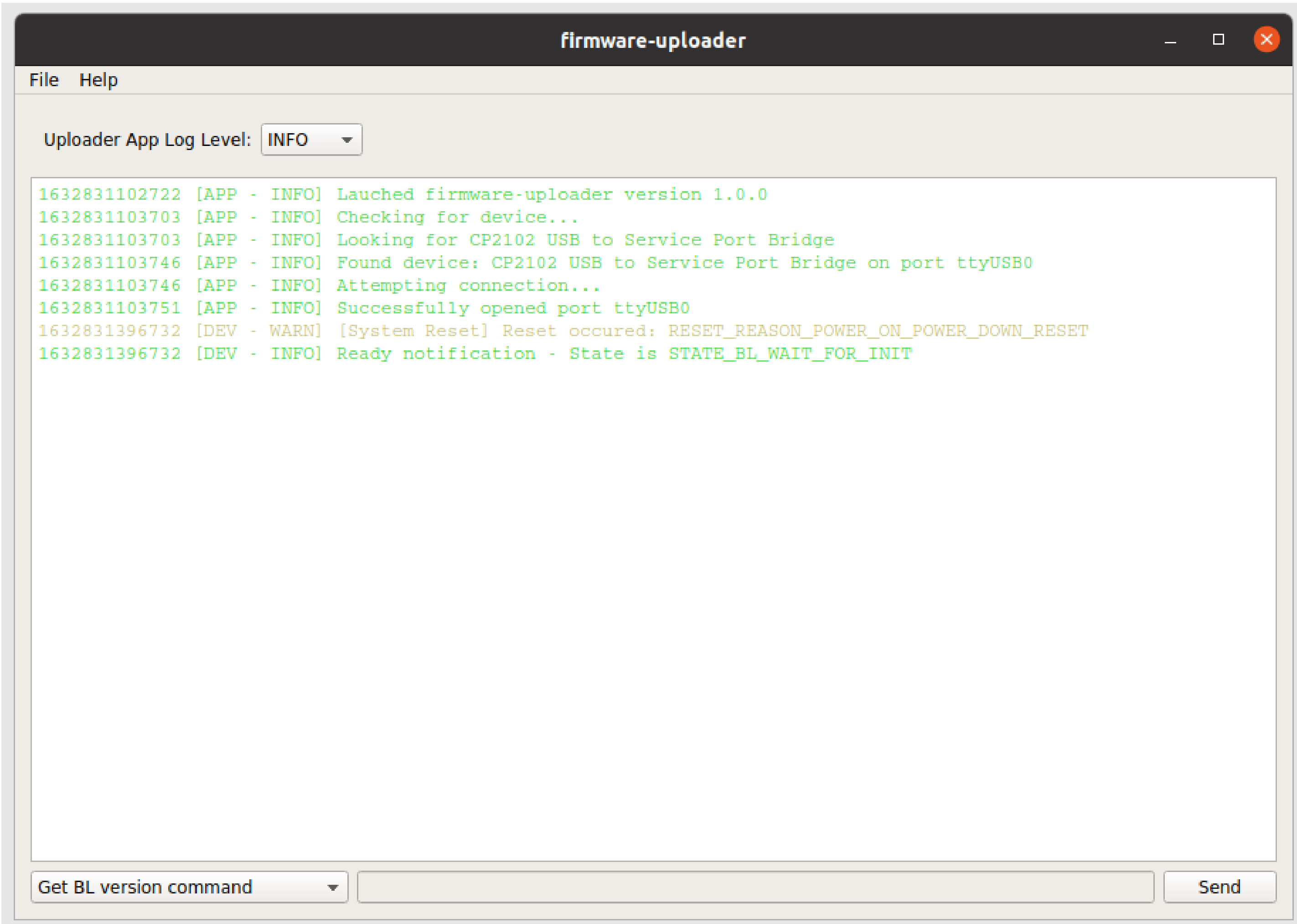
<u>Run firmware-uploader:</u>
Extract the archive
(Note: if the archive is extracted as a folder try another tool to extract the archive e.g. file-roller)
`>./Toposens_Firmware_Uploader-v1.0.1_lin64.AppImage` or double click the executable.

5.4 Firmware Update Software

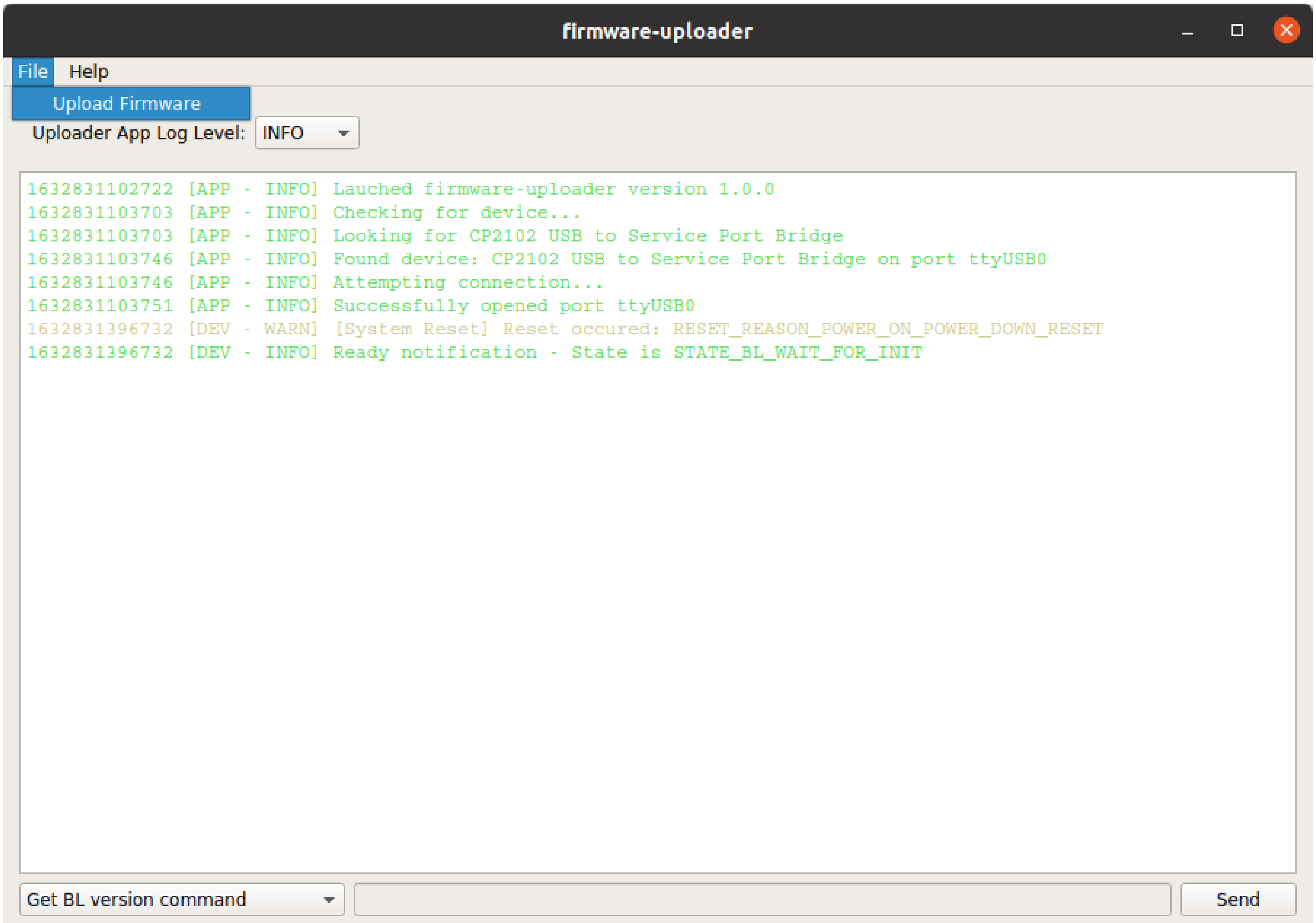
After starting the Firmware Uploader application, you should see a window similar to the one shown here:



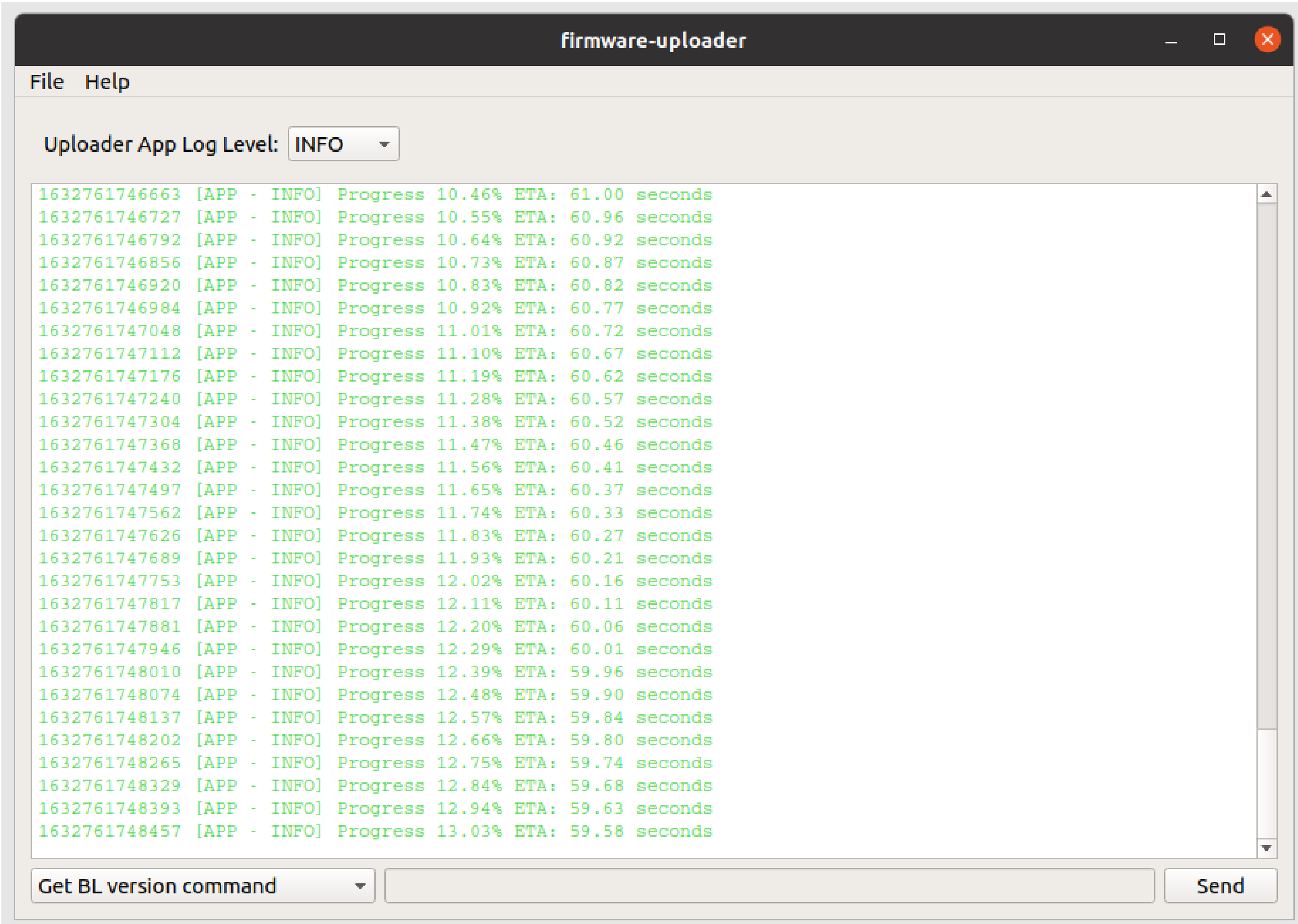
After powering on the device, you should see an output similar to the one shown:



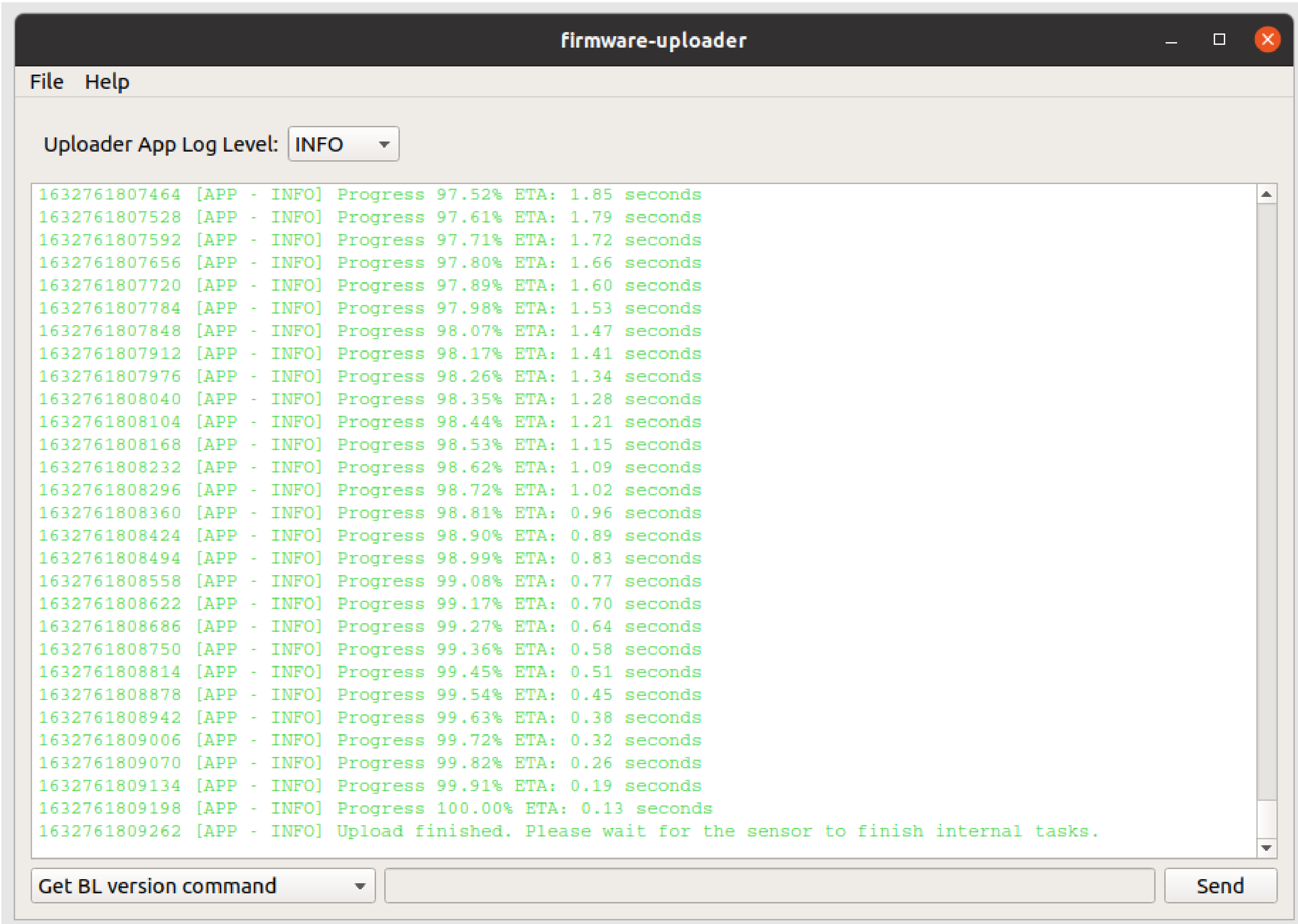
Select the new firmware file by clicking on the Menu → File → Upload Firmware:



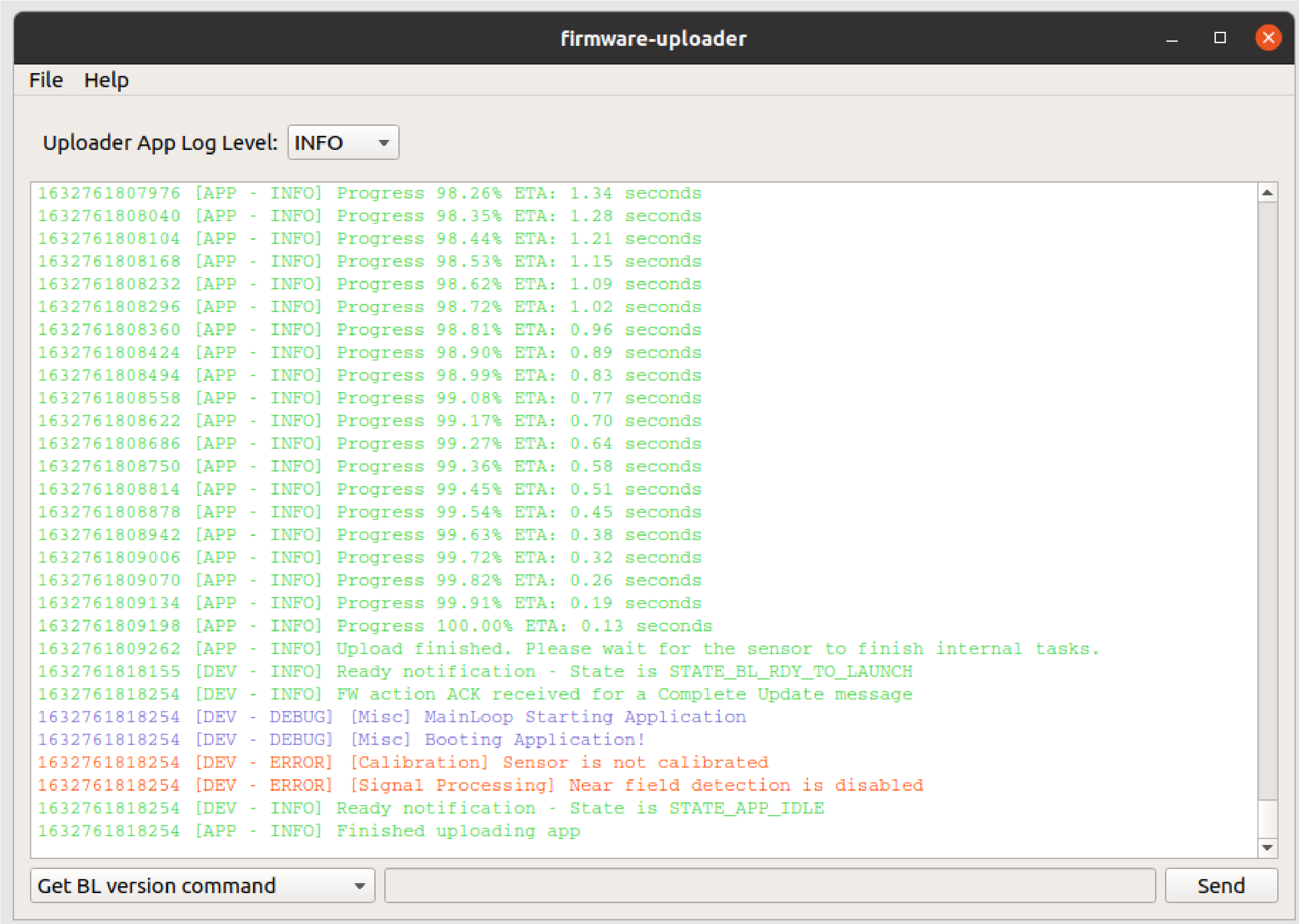
The upload process should start immediately (the transfer can take up to several minutes):



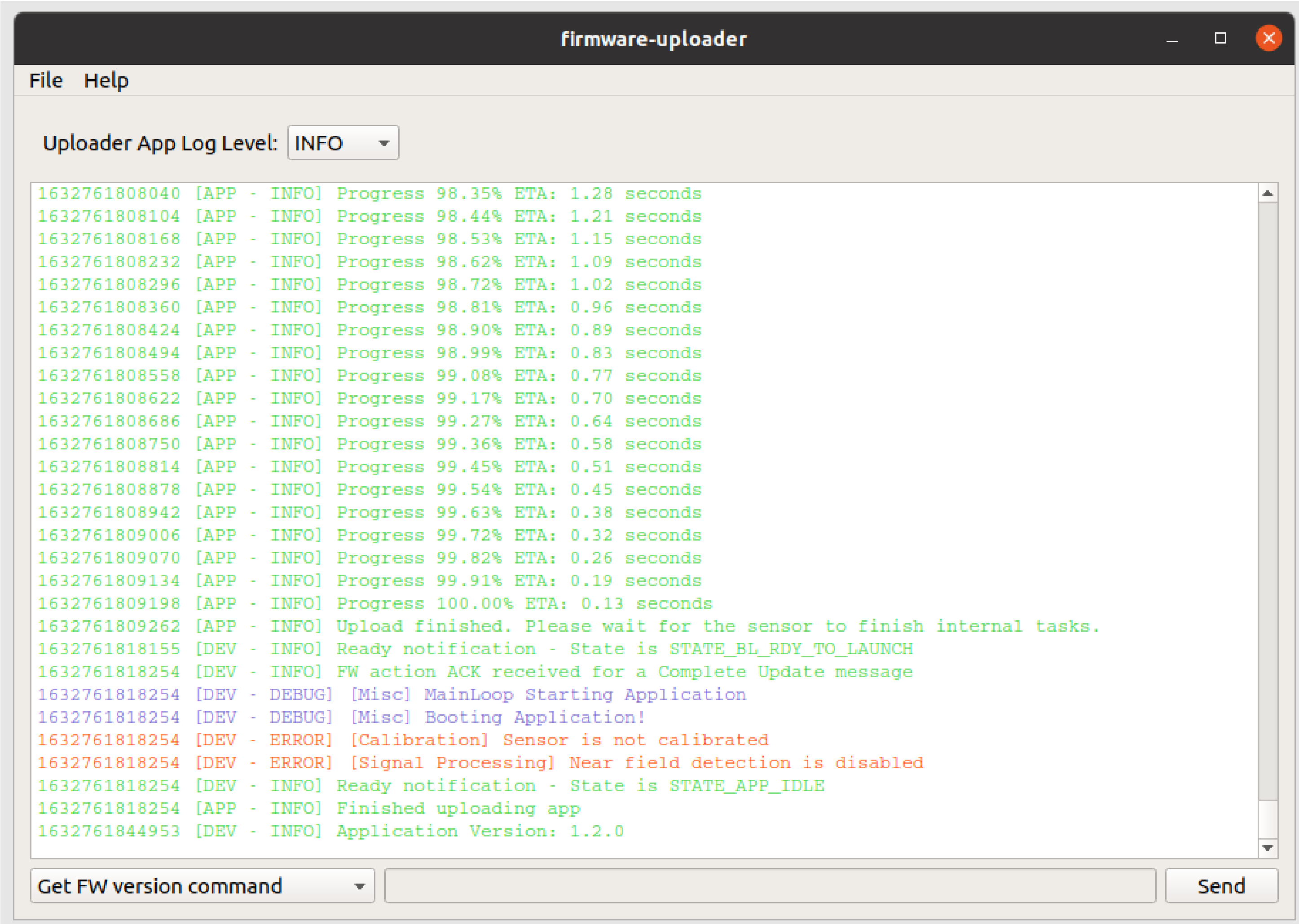
After the transfer is complete (Sending record 1) the device handles the new update internally, please be patient:



Once the upload is finished you get the message "Finished uploading app":

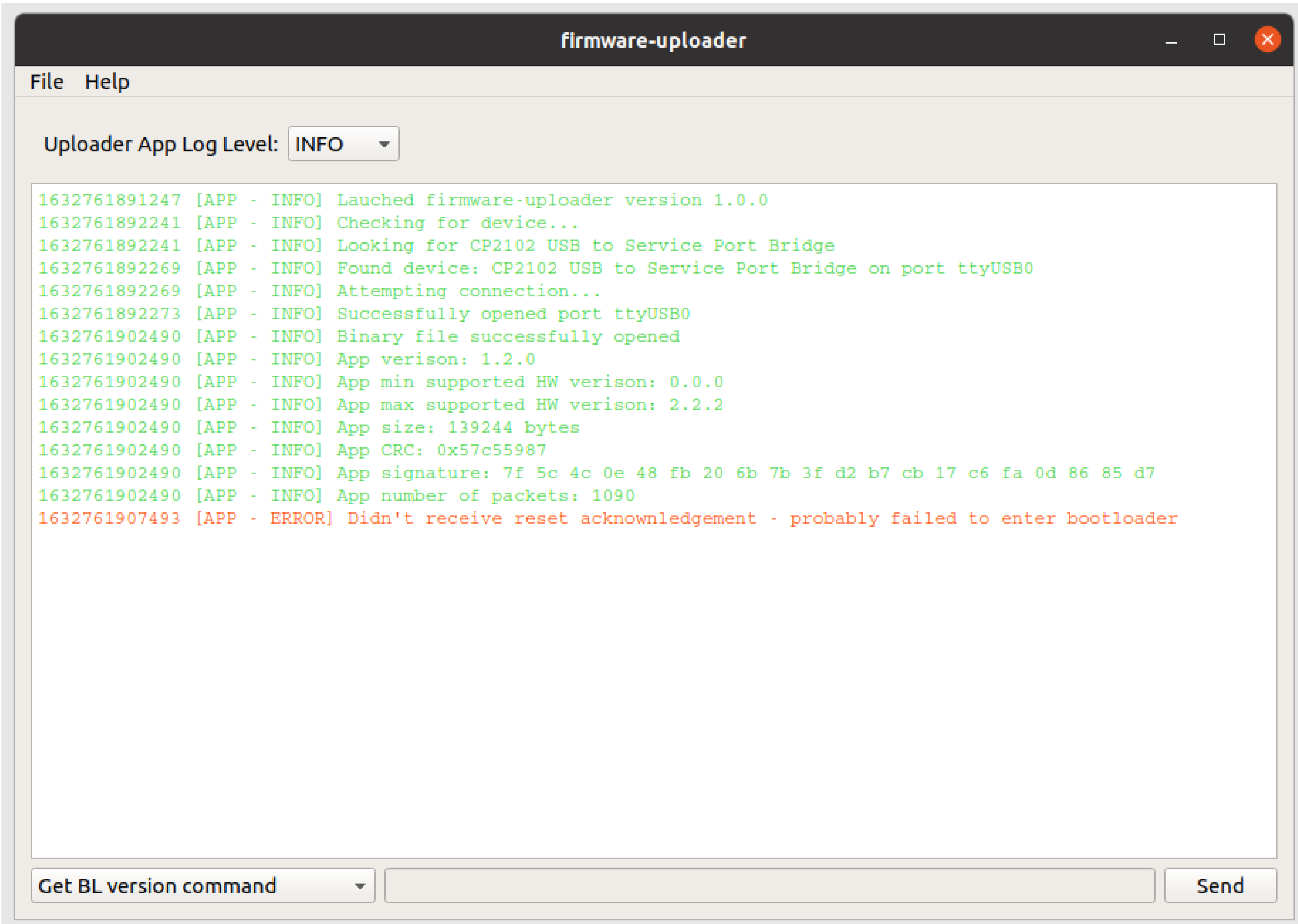


Verify your new version by selecting “Get FW version command“ in the dropdown-menu on the bottom left corner and clicking “Send“:



Update successful!

In case the update failed the uploader app will notify you by shown an error related to the upload process (in the shown example the sensor was not powered on thus the device could not enter the bootloader):



6. Instruction Set Description

6.1 CAN-Protocol

6.1.1 Transport Layer

The Toposens ECHO ONE DK communicates via high-speed CAN-Bus with the CAN 2.0A Standard. The CAN transceiver component operates under compliance of the ISO 11898-2:2016. Commands to control the sensor, as well as data output messages of the sensor are sent via the same CAN-Bus.

Variable	Configuration
Identifier Length	11 bit (Standard Frame)
Bit-Rate	1 Mbit/s (fixed rate)
Termination	Sensors are not terminated See connection diagram for more information

6.1.2 Presentation Layer

Each command or request exchanged between CAN-Host and Sensor is acknowledged with an ACK signal. For example, if a parameter is SET, the sensor will answer with an ACK message. It is possible to read out configured parameters without changing them, by sending a GET command. In this case, the sensor answers with a message containing the current parameter. After a measurement is taken, the sensor will request to start a point session. When this “start point session“-request is acknowledged by the CAN-Host, the sensor will send the point data which does not need to be acknowledged. When all of the point data has been sent, the sensor will send an “end of session“-request which needs to be acknowledged by the CAN-Host. When a sensor on the bus sends a CAN frame, it will always use its node ID as the CAN frame ID.

A sensor on the bus will only respond to messages which have the same frame ID as its node ID or have a frame ID of 0 (broadcast address). All other frame IDs will be ignored by the sensor.

In order to determine which sensors are on the bus, the host/master device could send any command with the broadcast address and listen for the replies that come back. For example, the reboot command.

By default, each sensor's node ID is a XORed version of the bytes that make up its UID. It is highly unlikely that two or more sensors will have the same node ID.

On the sensor side, all the commands will work if the sensor is addressed using the broadcast address (host/master uses a CAN frame ID of 0). When the library is using the multi-cast ID and sends a get request such as "Get Firmware Version" the API will block until the sensor on the bus replies but will only return the Firmware Version of the sensor that was first to respond.

The sensor will periodically resend messages if they do not receive a reply within a certain time period. For example, when running the single_shot_example, and the program is terminated mid-session, the sensor will periodically send a start/end of session request until it gets a reply.

6.1.3 Data Packet Format

Control Byte	Sub Control Byte	Parameter Byte [1-6]	
Usual Signals			
Command Byte	Sub-Command Byte	Signal Payload	
Defines the message type GET/SET/Trigger/End of Session/etc.	Defines the type of command or message.	The signal payload is of varying length and datatype, defined by the command.	
(Not) Acknowledged Signal - (N)ACK			
0x01(ACK) / 0x02 (NACK)		Control Byte	Sub Control Byte
If a received signal can (not) be processed the (N)ACK control byte is sent in front of an echo of the received signal. Note: Some signal types (e.g. GET) have their own (N)ACK control bytes.		Echo of the received signals control byte.	Echo of the received signals sub control byte. The parameter bytes are added as needed.
End of Session			
0x00			
A one-byte payload with 0x00 as command byte indicates the end of the session. The end of the session as acknowledged by the receiver.			

Each command and message have a defined number of parameter bytes and datatype.

6.1.4 Example Commands

The following example describes a "SET" command using the "Set Number of Pulses" command as an example.

	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
Host to Sensor	0x60	0x01	0x01	0x05
	Set Command	Category Transducer	Parameter "Number of Pulses"	Configuration Value. In this example, the sensor's transmitted signal is set to a length of 5 pulses.

6.1.5 Example Acknowledged Response

The sensor will answer with a "SET-ACK" control byte, the sub-control and parameter byte 1 used by the set command, and the acknowledgment status.

	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
Sensor to Host	0x61	0x01	0x01	0x00
	Set Command + Acknowledged-Nibble	Repeated: Category Transducer	Repeated: Parameter "Number of Pulses"	Indicating that the "Number of Pulses" was successfully set to the desired value.

6.1.6 Example Not Acknowledged Response

If the sensor is not able to answer or follow the command the response will be a NACK signal. For example, if it was tried to set the "Number of Pulses" parameter to 21 (which is out of range for this parameter), the response would be as follows:

If the sensor is not able to answer or follow the command the response will be a NACK signal. For example, if it was tried to set the "Number of Pulses" parameter to 21 (which is out of range for this parameter), the response would be as follows:

	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
Sensor to Host	0x61	0x01	0x01	0x01
	Set Command + Acknowledged-Nibble	Repeated: Category Transducer	Repeated: Category Transducer	Indicating that the "Number of Pulses" was not successfully updated because the chosen value for this parameter was out of range. In this case, the sensor retains the value it had for the "Number of Pulses" before this attempt was made.

6.1.7 Example Point session

Every point session is started with a Request-For-Session (RFS) signal. Once this request is ACKed by the host the point data is transmitted. This data needs no ACKs. The session is ended with an EOS (End-of-Session) signal. This EOS also needs to be ACKed.
Note: The ACKs for RFS and EOS are not required for all continuous Sensor-Modes (e.g. SENSOR_MODE_CONTINUOUS_TRANSMIT_LISTEN).

	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
Host to Sensor	0x30	0x00		
	Action Byte	Trigger measurement		
Sensor to Host	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
	0x31	0x00	0x00	
	ACK Action	Trigger measurement	Success	
Sensor to Host	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
	0x10	0x00	0x05	
	RFS	Not used	Number of points in this session	
Host to Sensor	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
	0x01	0x10	0x00	0x05
	ACK	RFS	Not used	Number of points in this session

Sensor to Host	Control Byte	Point-Data		
	0x11,0x12,0x13, 0x14	1-7 bytes according to the point type		
	Point-Data	See point data specification		**Note: these signals are not ACKed!**
Sensor to Host	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
	0x00			
	EOS			
Host to Sensor	Control Byte	Sub Control Byte	Parameter Byte 1	Parameter Byte 2
	0x01	0x00		
	ACK	EOS		

- Link to public library:
  https://gitlab.com/toposens/public/toposens-library
- Link to protocol documentation:
  https://gitlab.com/toposens/public/toposens-library/-/blob/master/communication_protocol/protocol_documentation.md

6.2 Library Command Overview

Description	Command	Page
Reboot sensor	`RequestReboot`	39
Reset sensor to factory settings	`RequestFactoryReset`	39
Store current settings to flash	`RequestStoreSettings`	40
Request a new measurement	`RequestMeasurement`	40
Configure transmission signal volume	`SetParameterTransducerVolume` `GetParameterTransducerVolume_u8`	42
Configure transmission signal length	`SetParameterTransducerNumOfPulses` `GetParameterTransducerNumOfPulses_u8`	43
Configure signal noise threshold	`SetParameterSignalProcessingNoiseLevelThresholdFactor` `GetParameterSignalProcessingNoiseLevelThresholdFactor_f`	44
Configure signal noise ratio	`SetParameterSignalProcessingNoiseRatioThreshold` `GetParameterSignalProcessingNoiseRatioThreshold_u8`	45
Enable/Disable multipath filtering	`SetParameterSignalProcessingEnableMultipathFilter` `GetParameterSignalProcessingEnableMultipathFilter_b`	46
Configure custom node ID	`SetParameterSystemNodeID` `GetParameterSystemNodeID_u16`	47
Get sensor state	`GetParameterSystemSensorState_t`	48
Get reset reason	`GetParameterSystemResetReason_t`	49
Get internal temperature	`GetParameterSystemMCUTemperature_f`	50
Get sensor versions	`RequestVersion_t`	50

6.3 Action Commands

The following section provides an overview of the APIs included in the Toposens Library.

Reboot

Function	Description	Return Value
`RequestReboot();`	Reboot sensor. Sensor will load stored settings. Unsaved settings will be lost. The blocking function `WaitForReady()` should be called if the `RequestReboot()` command was successfully carried out. The `WaitForReady()` function will block the calling program until the sensor sends a ready message indicating that it has fully rebooted and is ready for other commands. Commands sent to the sensor before this ready message is received will likely be ignored.	Boolean [`bool`] True = Reboot Command ACK False = No reboot possible

Factory Reset

Function	Description	Return Value
`RequestFactoryReset();`	Restore all sensor settings back to default values. This includes the Node ID, which could lead to a connection loss when the command is executed. Note: These settings are not stored to the sensor by default.	Boolean [`bool`] True = Factory Settings Restored False = Command execution not possible

Store Settings

Function	Description	Return Value
`RequestStoreSettings();`	Stores all current settings of the sensor. Parameters have to be set prior to executing the command.	Boolean [`bool`] True = Parameters stored False = Parameters could not be stored

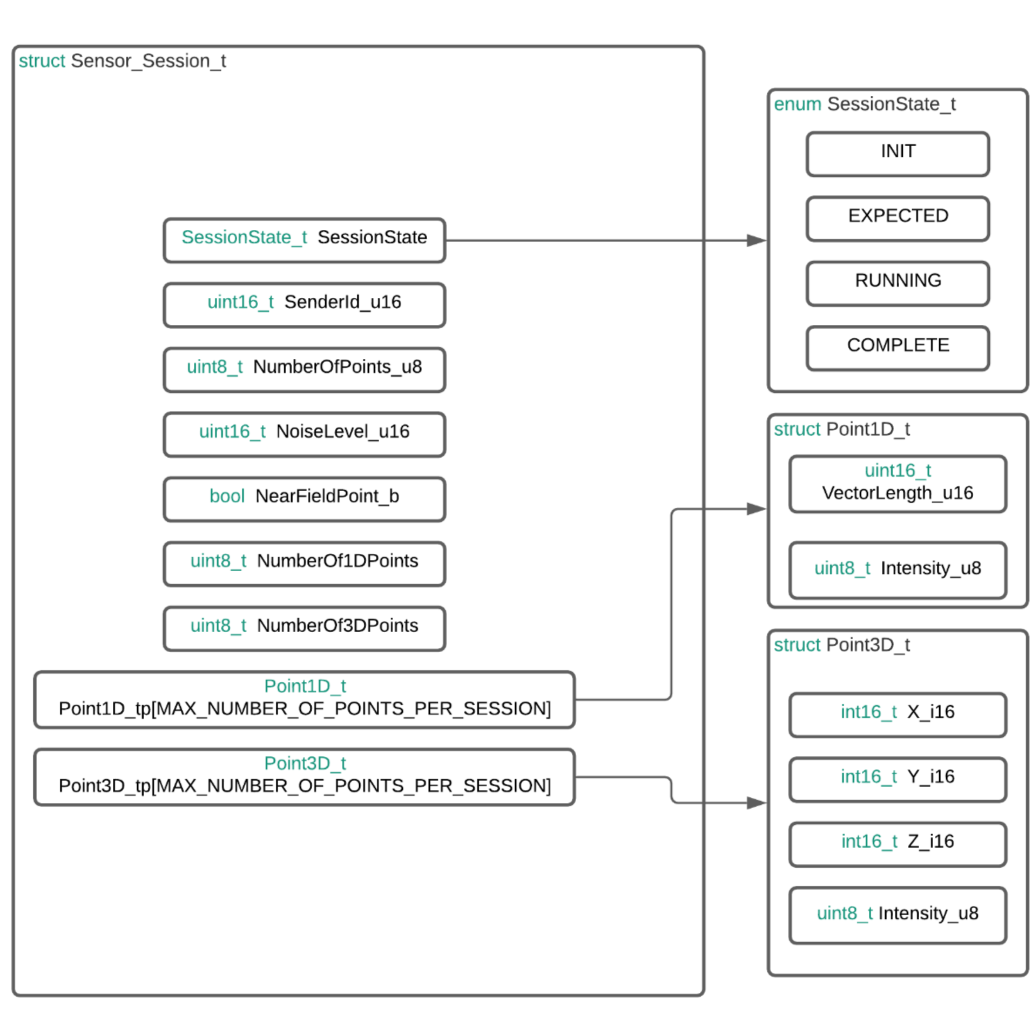
Request Measurement

Function	Description	Return Value
`RequestMeasurement();`	Triggers a single measurement. Sensor answers with acknowledge and starts the measuring process. A point session will be initialized by the sensor when the point cloud data is available. Note: Make sure to set the according mode before calling this function.	Boolean [`bool`] True = Measurement trigger successful False = Request failed

Sensor Data Availability

Function	Description	Input Value	Output Value
`Sensor_Session_t *SessionData = GetSessionData(SenderId_u16);`	Once the point session has finished the point data can be accessed through the `GetSessionData()` function which returns a pointer to a `Sensor_Session_t` data structure.	Unsigned 16 Bit [`uint16_t`] Sensor Node ID	Sensor_Session [`Sensor_Session_t`] Sensor Session Data

Sensor Session Data Structure



The session data struct holds all measurement data received from the sensor. This depends on the configuration of the sensor, e.g. disabled near field detection will result in no relevant data in the `NearFieldPoint_b` variable.

The number of detected points is saved in `NumberOf3DPoints` and has to be used to access the point cloud information in the `Point3D_tp` array.

6.3 Performance Settings

The sensor can be configured by the user to adjust the performance to the requirements of the application.

Causes for not acknowledged configuration commands.

- Parameter is out of range
- Parameter format is wrong
- Sensor is transmitting data and cannot respond to request
- Sensor is rebooting

Volume

Default Value	Variable Range	Description
100%	0-100	Value (percentage) which defines the transmission pulse amplitude emitted by the sensor during measurement. This value influences the maximum range and signal quality of the sensor and should be left at "100" for most applications.

Library Command

Function	Description	Input Value	Return Value
`SetParameterTransducerVolume(uint8_t Volume_u8);`	Set volume parameter	Unsigned 8 Bit [`uint8_t`] Volume Value	Boolean [`bool`] True = Configuration acknowledged False = Command not acknowledged
`GetParameterTransducerVolume_u8();`	Get volume parameter	None	Unsigned 8 Bit [`uint8_t`] Volume Value

Number of Pulses

Default Value	Variable Range	Description
5	0-10	Value (number) of consecutive ultrasonic pulses emitted by the sensor. This value determines the duration of the transmitted ultrasonic pulse and will influence the sensor's ability to detect small objects, framerate and separation of objects located close to each other. The default value of "5" is a good compromise between object separation and sensitivity.

Library Command

Function	Description	Input Value	Return Value
`SetParameterTransducerNumOfPulses(uint8_t NumOfPulses_u8);`	Set volume parameter	Unsigned 8 Bit [`uint8_t`] Number of Pulses	Boolean [`bool`] True = Configuration acknowledged False = Command not acknowledged
`GetParameterTransducerNumOfPulses_u8();`	Get volume parameter	None	Unsigned 8 Bit [`uint8_t`] Number of Pulses

Noise Level Threshold Factor

Default Value	Variable Range	Description
1.0	0 - 10.0	Value (Multiplier) which influences the minimum dynamically determined minimum Signal-to-Noise ratio. A Higher value causes more weak reflections to be rejected, an thus a lower quantity of available data. A lower Value increases the quantity of detected reflections but also increases the amount of detections with low positional accuracy and precision.

Library Command

Function	Description	Input Value	Return Value
`SetParameterSignalProcessingNoiseLevelThresholdFactor(float Factor_f);`	Set noise level parameter	Floating Point Number [`float`] Rejection Factor	Boolean [bool] True = Configuration acknowledged False = Command not acknowledged
`GetParameterSignalProcessingNoiseLevelThresholdFactor_f()`	Get noise level parameter	None	Floating Point Number [`float`] Rejection Factor

Noise Ratio Threshold

Default Value	Variable Range	Description
50	0-100	Value (percentage) which defines the minimum acceptable signal envelope variation. This value influences which signals are rejected if received in short succession and/or are contaminated by interference. A lower value leads to more precise data output, but at a reduced quantity. The default value of "50" represents a balanced configuration for most situations.

Library Command

Function	Description	Input Value	Return Value
`SetParameterSignalProcessingNoiseRatioThreshold(uint8_t Threshold_u8);`	Set	Unsigned 8 Bit [`uint8_t`] Noise ratio threshold	Boolean [`bool`] True = Configuration acknowledged False = Command not acknowledged
`GetParameterSignalProcessingNoiseRatioThreshold_u8();`	Get	None	Unsigned 8 Bit [`uint8_t`] Noise ratio threshold

Multipath Filtering

Default Value	Variable Range	Description
1	0 [Off] / 1 [On]	Enables the filtering of multi-path echo detections by creating a virtual shadow around detected objects, deleting detections directly behind the first detection.

Library Command

Function	Description	Input Value	Return Value
`SetParameterSignalProcessingEnableMultipathFilter(bool Enable_b);`	Set	Boolean [bool] True = Multipath filtering enabled False = Multipath filtering disabled	Boolean [bool] True = Configuration acknowledged False = Command not acknowledged
`GetParameterSignalProcessingEnableMultipathFilter_b();`	Get	None	Boolean [bool] True = Multipath filtering enabled False = Multipath filtering disabled

6.4 General Commands

This section describes all support functions, general settings, and other configurations available to the user.

Sensor Configuration Parameters

Node ID

Default Value	Variable Range	Description
11 Bit random number based on the unique chip ID of the sensor.	1 – 2047 Note that the address 0 is reserved for the broadcast address.	Change Node ID. This function should not be necessary due to the low probability of two sensors having the same Node ID. Full Multi sensor support will be released in future firmware and library versions.

Library Command

Function	Description	Input Value	Return Value
`SetParameterSystemNodeID(uint16_t NodeID_u16);`	Set Node ID	Unsigned 16 Bit [`uint16_t`] Node ID	Boolean [`bool`] True = Configuration acknowledged False = Command not acknowledged
`GetParameterSystemNodeID_u16();`	Get Node ID	None	Unsigned 16 Bit [`uint16_t`] Node ID

Get Sensor State

Default Value	Variable Range	Description
Sensor State 6 = Idle **STATE_APP_IDLE**	6 STATE_APP_IDLE 7 STATE_APP_NOISE_SAMPLE 8 STATE_APP_WAIT_FOR_NOISE_SAMPLING 9 STATE_APP_TRANSDUCE_AND_SAMPLE 10 STATE_APP_TRANSMIT 11 STATE_APP_SAMPLE 12 STATE_APP_WAIT_FOR_SAMPLING 13 STATE_APP_SIG_PRO_CALIBRATION 14 STATE_APP_CALCULATE_POINTS 15 STATE_APP_OUTPUT_POINTS 16 STATE_APP_WAITING_FOR_POINT_SESSION_END 17 STATE_APP_OUTPUT_ADC_SIGNALS 18 STATE_APP_WAITING_FOR_ADC_DUMP_SESSION_END	Output current sensor state.

Library Command

Function	Description	Input Value	Return Value
`GetParameterSystemSensorState_t();`	Get Sensor State	None	Enumeration [`enum SensorState_t`] See variable range for definition.

Get Reset Reason

Default Value	Variable Range	Description

None	`0 RESET_REASON_UNKNOWN`
	`1 RESET_REASON_LOW_POWER_RESET`
	`2 RESET_REASON_WINDOW_WATCHDOG_RESET`
	`3 RESET_REASON_INDEPENDENT_WATCHDOG_RESET`
	`4 RESET_REASON_SOFTWARE_RESET`
	`5 RESET_REASON_POWER_ON_POWER_DOWN_RESET`
	`6 RESET_REASON_EXTERNAL_RESET_PIN_RESET`
	`7 RESET_REASON_BROWNOUT_RESET`
		Output reset reason if sensor reset occurred.

Library Command

Function	Description	Input Value	Return Value
`GetParameterSystemResetReason_t();`	Get Reset Reason	None	Enumeration [`enum ResetReason_t`] See variable range for definition.

Get Internal Temperature

Default Value	Variable Range	Description
None	-10 to 100	Output of the sensors internal sensor temperature.

Library Command

Function	Description	Input Value	Return Value
`GetParameterSystemMCUTemperature_f();`	Get MCU Temperature	None	Floating Point Number [`float`] MCU Temperature in °Celsius

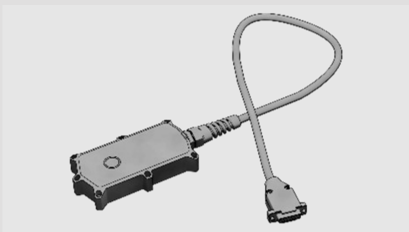
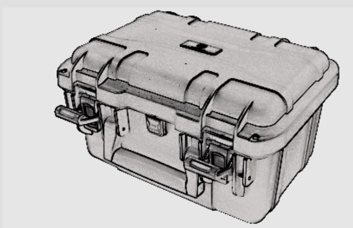
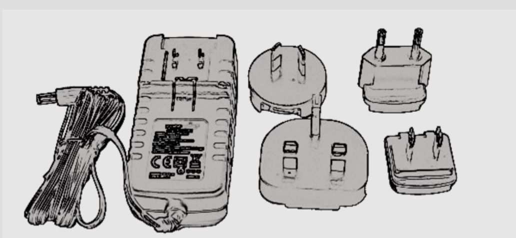
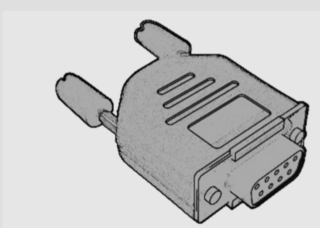
Get Version

Library Command

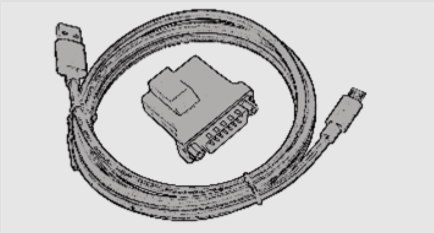
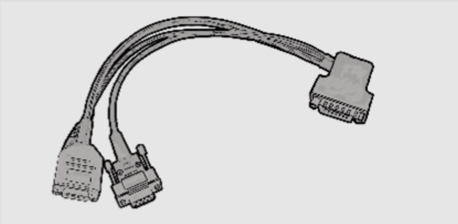
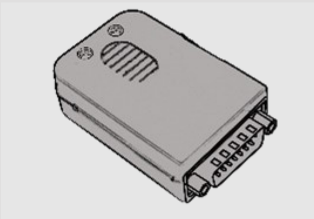
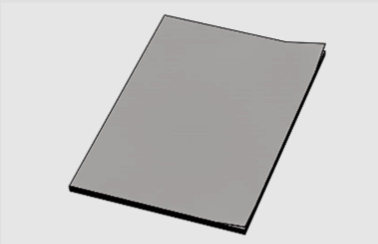
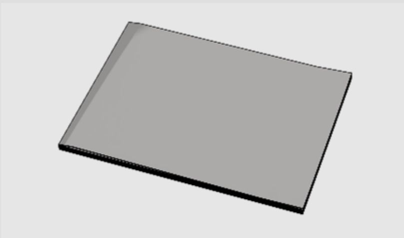
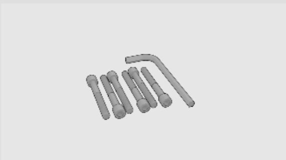
Function	Description	Input Value	Return Value
`RequestVersion_t(VersionByte_t);`	Get Version	Enumeration [`enum VersionByte_t`] VERSION_BYTE_BOOTLOADER = 0x00 VERSION_BYTE_APP = 0x01 VERSION_BYTE_HW = 0x02 VERSION_BYTE_SIG_PRO_LIB = 0x03 VERSION_BYTE_COMMS_LIB = 0x04	Struct [`struct Version_t`]

7. Order Information

Contents of ECHO ONE DK Set - 1/2

Hardware	Image	Description
ECHO ONE DK		3D Ultrasonic Development Sensor
Protective Case		Water-tight transport case for ECHO ONE DK Set Components
Power Supply		12V 3A Power Supply Socket Types: UK, US/JP, AUS, EU / 100-240V
CAN Terminator		D-Sub 9 CAN-Terminator Split Termination 120Ω
Sensor Mount		Mount for attaching the ECHO ONE DK to a tripod or table

Contents of ECHO ONE DK Set – 2/2

Hardware	Image	Description
Interface Adapter		USB Adapter + Micro USB Cable to connect the ECHO ONE DK to a PC (Usage of Visualizer and Firmware Upload Tool only possible via Interface Adapter)
CAN Communication Cable		Cable to connect the sensor to a CAN Network
Breakout Box		Provides convenient access to all sensor pins
Instruction Manual		Containing all necessary information and instructions for the commissioning, installation, safe use and maintenance of the ECHO ONE DK
Quick-Start-Guide		General information about the ECHO ONE DK and first-steps setup guide
Other		1x pack of screws 1x adhesive strip 1x hex key

Additional Hardware available upon request

- **Additional ECHO ONE DK 3D Sensor**

  E.g. for higher area coverage around the autonomous vehicle, using the multi-sensor functionality of the system, connecting several ECHO ONE DK in a daisy chain network.

- **Sensor Network Extension Cable**

  For connecting several sensors with each other via a daisy chain mechanism

- **Development ECU (only for selected development partners)**

  Provides embedded ROS-based data postprocessing for multiple communication interfaces and IO-Ports.
  Enables Features:
  - 3D collision avoidance functionality with adjustable collision zones.
  - Advanced adjustable filter algorithms enabling a condensed and noise filtered point cloud output.
  - Output signal generation for digital I/Os for e.g. connection to industrial PLC to enable simple "Go, slow down, stop" commands to an AGV or other autonomous system based on the read out of the 3D collision avoidance functionality.

These components are only intended for use in conjunction with the ultrasonic sensor system Echo-One-DK.

8. Resources

Documentation (www.toposens.com/downloads).

- ECHO ONE DK Data Sheet
- ECHO ONE DK Instruction Manual
- Quick-Start-Guide (Getting started with Toposens Visualizer)
- Toposens 3D Visualizer Manual
- Toposens ROS Manual

Software (www.toposens.com/members).

- **Toposens Sensor Library (see section 5.3)**
  Available via GitLab: https://gitlab.com/toposens/public/toposens-library
- **Firmware Update Tool (see section 5.4)**
  Downloadable via: www.toposens.com/members
- **Toposens 3D Visualizer**
  Downloadable via: www.toposens.com/members
- **ROS Implementation Packages of Toposens**
  http://wiki.ros.org/toposens.

Toposens GmbH

Lyonel-Feininger-Straße 28
80807 Munich
www.toposens.com

+49 089 23751540

Document Version: 1.2
Release Date: October 2021