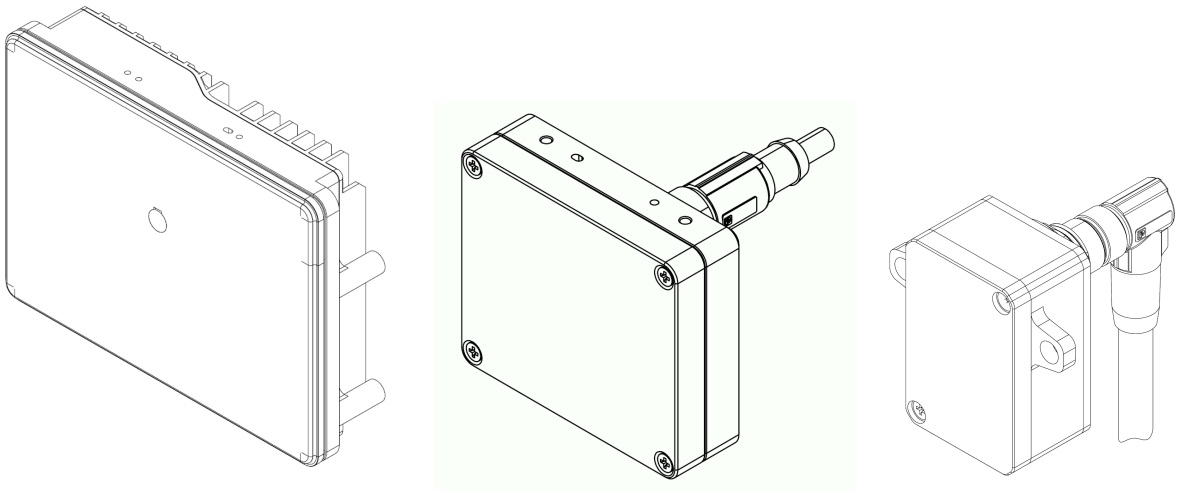
Zadar Labs, Inc.
1650 Coleman Ave
Santa Clara, CA 95050 USA



Software Integration Manual

Release v3.1.0

December 2024

Zadar Radar Platforms

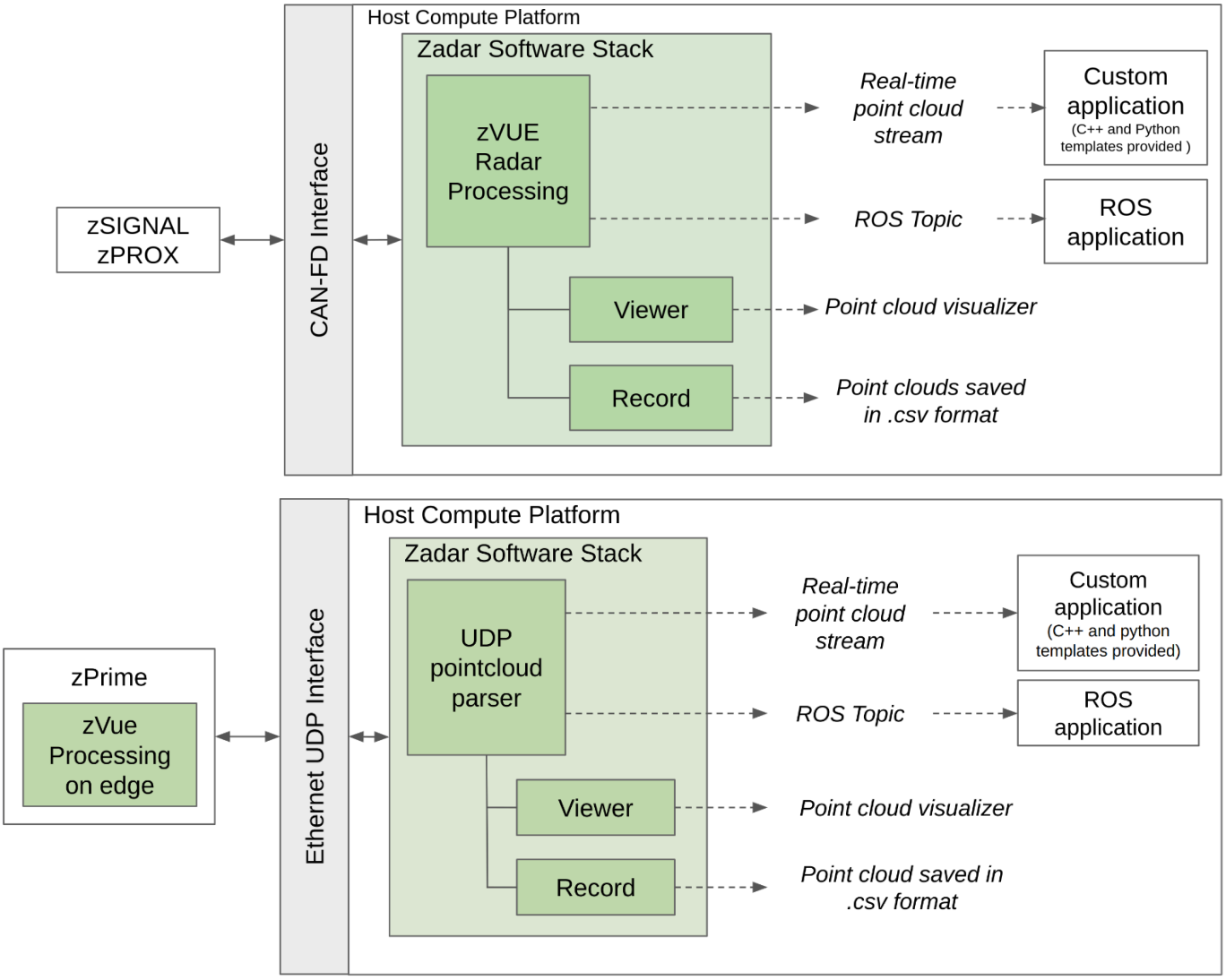
Table of Contents

1 Introduction

This guide discusses the installation and use of the Zadar Software driver for the Zadar sensors including: zSIGNAL, zPROX and zPRIME. The Zadar Software driver includes 3 main functionalities:

- **zVUE** implements the main data processing to receive the point cloud from the sensor. This is a key component required for operating zSIGNAL, zPRIME and zPROX. For zPRIME the main processing burden is implemented directly on the radar.
- **Viewer** allows users to visualize the point cloud in real-time.
- **Record** allows users to save the point cloud into CSV format.

Additionally, we provide template programs to receive the point cloud in order for you to start implementing your own custom application.



[figure 1] Sensor and Software overview for zSignal/zProx and zPrime case

2 System Requirements

2.1 System Requirements zSIGNAL / zPROX

Below are the requirements for the installation and use of the sensor with Zadar Software driver:

- Operating System
  - GNU/Linux - Tested on: x86 Ubuntu 20.04 and 22.04 LTS
    - Note: ARM Linux platforms are supported. Please contact support for non-x86 platforms.
    - Note: for other operating systems, please contact support.
    - Note: Ubuntu 22.04 LTS does not support the viewer
  - Kernel must support SocketCAN
- Hardware
  - 1 radar sensor (zSIGNAL or zPROX)
  - 1 CAN-FD receiver
    - This can be a CAN-FD to USB interface, or it can be a CAN peripheral on your embedded system that supports 8Mbps CAN-FD through SocketCAN.
- Software
  - ZadarSoftware/ folder containing the Zadar software driver

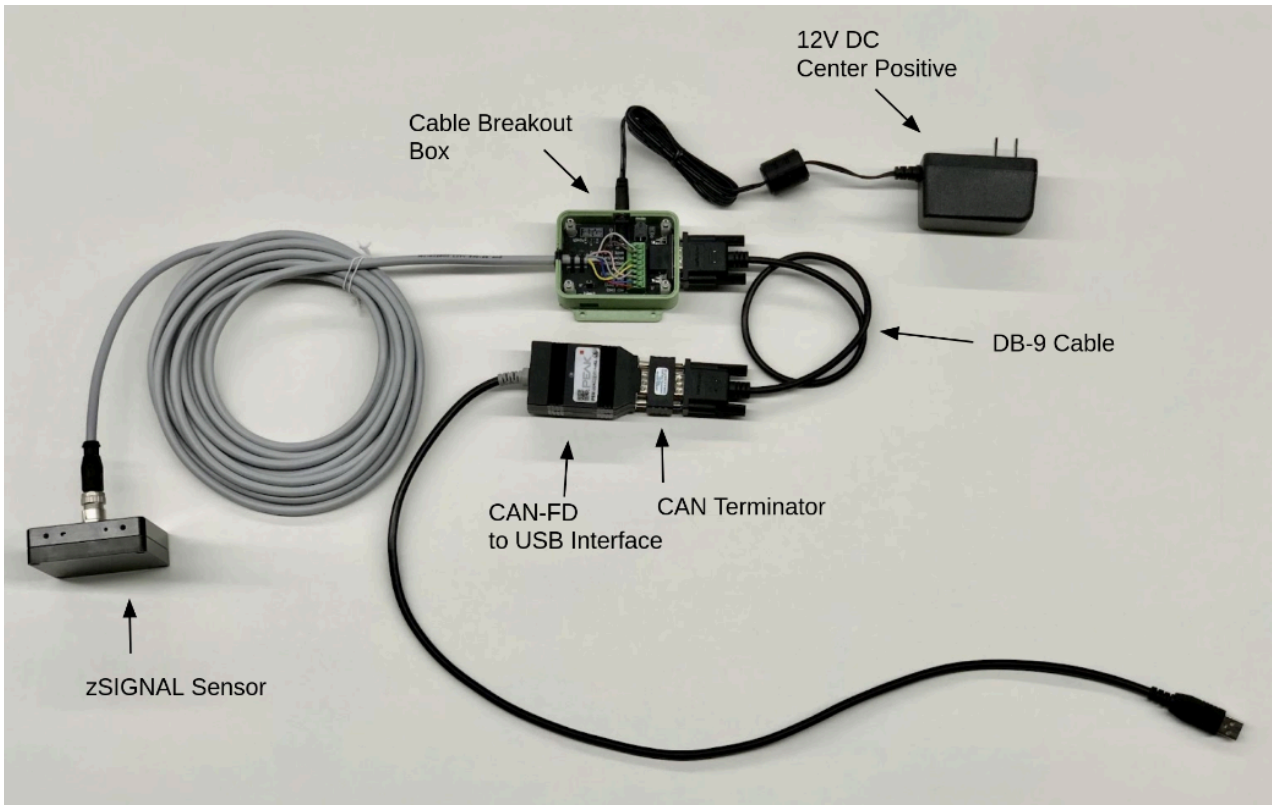
2.2 System Requirements zPRIME

Below are the requirements for the installation and use of zPrime with Zadar Software driver:

- Operating System
  - GNU/Linux - Tested on: x86 Ubuntu 20.04 and 22.04 LTS
    - Note: ARM Linux platforms are supported. Please contact support for non-x86 platforms.
    - Note: for other operating systems, please contact support.
    - Note: Ubuntu 22.04 LTS does not support the viewer
- Hardware
  - 1 radar sensor (zPrime)
  - Power over Ethernet adapter (POE++ Power rating) **OR** 1000BASE-T to 1000BASE-T1 converter with 12V adapter
- Software
  - ZadarSoftware/ folder containing the Zadar software driver

2.3 Setup zSIGNAL / zPROX

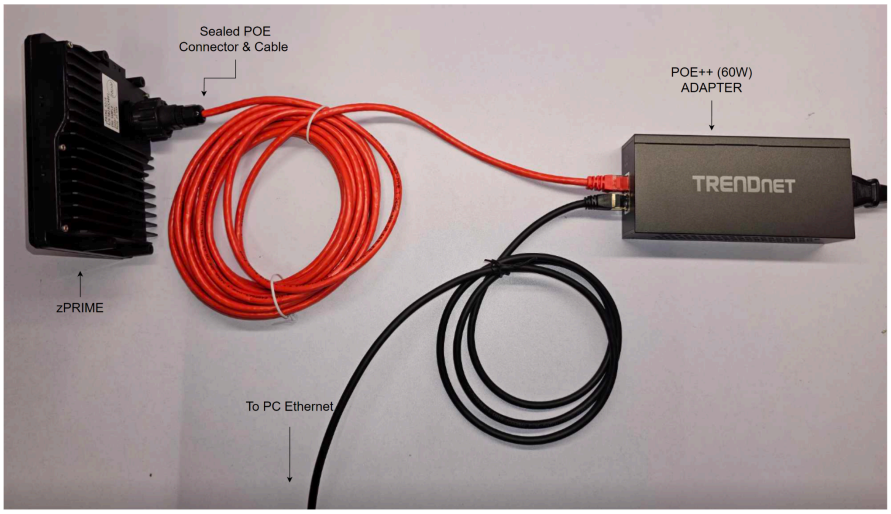
1. For your evaluation convenience, we provide a complete kit to test our sensor with any Linux computer. This includes the cable breakout, power supply, and CAN-FD interface.
2. For volume deployment in an embedded/ECU environment, zSIGNAL can integrate directly into a CAN bus with a custom harness.
3. Connect the sensor to the computer as pictured and power it up with all the required cables.
    - Sensor -> Breakout Cable -> DB-9 cable -> CAN DB9 terminator -> CAN-FD to USB interface
    - 12V 1A Power



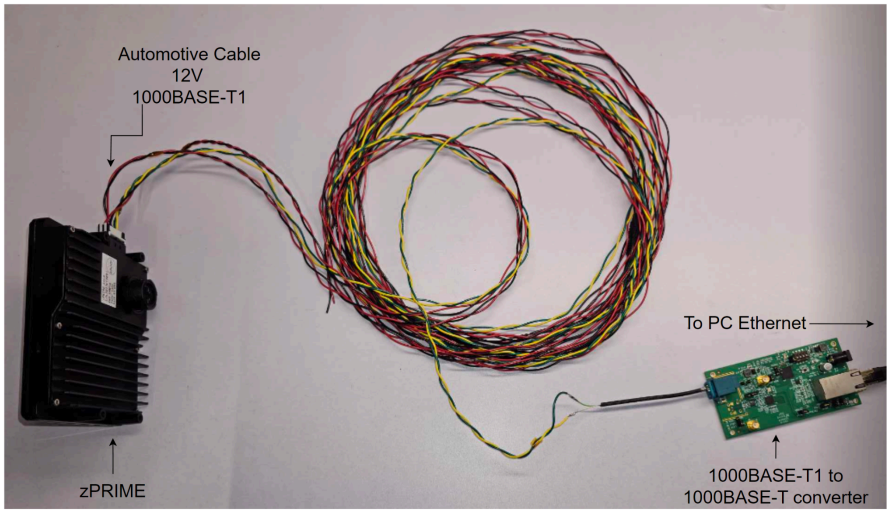
4. Extract the archive ZadarSoftware.zip, provided by our software team
5. Place yourself inside the unzipped ZadarSoftware/ folder, right-click the mouse and select “open in terminal” (you will now see a window terminal where you can run the commands that will follow).

2.4 Setup zPRIME

- For your evaluation convenience, we provide a complete kit to test our sensor with any Linux computer. This includes the necessary power supply and ethernet adapter.
- Connect the sensor to the computer as pictured.
    - Sensor -> POE++ Adapter -> PC (This for Automotive that already uses Ethernet)



    - Sensor -> 1000BASE-T1 to 1000BASE-T Converter -> 12V and PC *(For Automotive Requiring a 1000BASE-T1 to Ethernet Conversion)*



- Please statically assign the ip address of the ethernet port on your computer to ‘192.168.0.20’ with a subnet mask of ‘255.255.255.0’.
- Extract the archive ZadarSoftware.zip, provided by our software team

- Place yourself inside the unzipped ZadarSoftware/ folder, right-click the mouse and select “open in terminal” (you will now see a window terminal where you can run the commands that will follow).

4 Quick Start

4.1 Installing Prerequisites

Platform Prerequisites (for zSignal and zProx only)

Your system must support communication over CAN-FD. This can be achieved in two ways:

- Communication over a CAN-USB interface such as a PEAK-CAN device. This is typically used for PC/x86 hosts
- Communication over an integrated CAN-FD peripheral with 8Mbps data rate support on an embedded system. This is typically used for integrated systems such as Nvidia Jetson/Orin.

For evaluation and prototyping purposes we support testing our sensors on Linux x86 PC systems with a PEAK-CAN USB interface as pictured in the previous section. SocketCAN is used to allow our software driver to communicate with the CAN bus. To load the appropriate kernel modules:

```
$ sudo modprobe can
$ sudo modprobe peak_usb
```

For other types of interfaces (including embedded CAN peripherals), contact our support team for integration help.

Software Prerequisites

Install dependencies with the following command on the terminal:

```
$ sudo ./install_dependencies.sh
```

Wait for all dependencies to install before proceeding.

4.2 Running the Radar

With the radar powered up and connected, run the radar with the following command on the terminal. Ctrl + C will terminate the process. In case of errors, try restarting the system by power cycling the sensor and plugging/unplugging the USB-CAN interface or ethernet cable.

```
$ ./run_radar.sh <radar_num> <radar_types> <radar_sns> <mode_ids> [<viewer>]
[<log>] [<override_receive_time>]
```

For single-radar zsignal or zprox example:

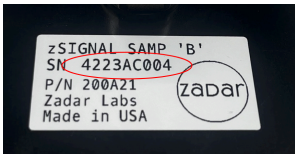
```
$ ./run_radar.sh 1 zsignal 3223AC002 1 on off
```

For single-radar zprime example:

```
$ ./run_radar.sh 1 zprime 3223AC001 1 on off
```

For multi-radar pass a list of values for every argument of every radar to run for example:

```
$ ./run_radar.sh 2 zsignal zsignal 3223AC001 3223AC002 1 1 on off
```

	Replace *<radar_num>* with the correct number of sensors to launch. If more than 1 the rest of the attributes *<radar_types>*, *<radar_sns>* and *<mode_ids>* will contain a list of values for every radar respectively.
	Replace *<radar_types>* with the correct sensor type (lowercase): *zsignal*, *zprox or zprime*.
⚠	Replace *<radar_sns>* with the correct sensor serial number you are using (label on the back of the sensor identified by 9 alphanumeric digits). 
	Replace *<mode_ids>* with the correct mode number defined in the table section 4.3 for your application. If you have a custom mode defined by the Zadar team, put the correct ID for the mode as provided to you.
	Replace [*<viewer>*] with *on* or *off* to open the viewer if needed. More details about the viewer in section 4.4
	Replace *[<log>]* with *on* or *off* to save into a .txt file the terminal outputs. The file will be created into a /logs/ folder in the zadar software driver main folder.
	Replace [<override_receive_time>] with either nothing or on/off. If on then this will override the timestamp, with when it was received by the host software

4.3 Radar Modes

Note: custom modes may override default modes based on custom specifications.

zPROX Modes

Mode ID	1	2	3	4
Maximum Range	20m	50m	20m	50m
Minimum Range	5cm	15cm	10cm	25cm
Field of View (H x V)	140° x 120°			
Angular Resolution (HxV)	5° x 5° (dynamic)			
Range Resolution	4.4cm	11cm	9.1cm	21cm
Vehicle Detection	20m	50m	20m	50m
Pedestrian Detection	20m	>25m	20m	>25m
Doppler Ambiguity *	27m/s	27m/s	52m/s	52m/s
Doppler Resolution	0.2m/s			
Doppler Accuracy	0.05m/s			
Frame Rate	40 Hz			

*The platform speed needs to be less than 1/4 of the provided range for Doppler Ambiguity, otherwise the reported velocity might be folded.

zSIGNAL Modes

Mode Number	1	2	3	4
Maximum Range	50m	100m	175m	175m
Minimum Range	7cm	0.5m	0.5m	1m
Field of View (H x V)	130° x 24°			
Angular Accuracy	3°			
Range Resolution	5.2cm	43cm	31.8cm	72cm
Vehicle Detection	50m	100m	175m	175m
Pedestrian Detection	50m	100m	>100m	>100m
Doppler Ambiguity *	42m/s	158m/s	85m/s	158m/s
Doppler Resolution	0.2m/s			
Doppler Accuracy	0.05m/s			
Frame Time	35ms	25ms		

*The platform speed needs to be less than 1/4 of the provided range for Doppler Ambiguity, otherwise the reported velocity might be folded.

zPRIME Modes For Dynamic Platforms

Mode ID for each zPRIME Variant											
	LR	NA		1	2	3	4	5	6	7*	8*
	W			1	2	3	4	5	6	NA	
	U	1	2	3	4	5	6	NA			
Max Range		30m	30m	85m	85m	250m	250m	400m	400m	800m	800m
Minimum Range		0.05m	0.05m	0.1m	0.2m	0.3m	0.75m	0.5m	1.0m	1.0m	2.0m
Range Resolution		3.3cm	3.3cm	8.9cm	17.8cm	26.2cm	52.3cm	41.9cm	82.2cm	83.4cm	167.4cm
Doppler Ambiguity		40.4m/s	74.4m/s	74.4m/s	143.4m/s	74.4m/s	143.4m/s	74.4m/s	143.4m/s	74.4m/s	143.4m/s
Doppler Resolution		0.079m/s	0.145m/s	0.145m/s	0.14m/s	0.145m/s	0.14m/s	0.145m/s	0.14m/s	0.145m/s	0.14m/s
Field of View (H x V)		130° x 24° (LR) / 130° x 48° (W) / 130° x 90° (U)									
Angular Res. (H x V)		0.35° x 0.35° (LR) / 0.35° x 0.90° (W) / 0.35° x 1.6° (U)									
Frame Time		50ms									

* Provided under request.

zPRIME Modes For Traffic Monitoring Applications (W Variant)

Mode Number	1*	2	3
Max Range	85m	250m	400m
Minimum Range	0.2m	0.75m	1m
Range Resolution	17.8cm	52.3cm	82.2cm
Doppler Ambiguity	143.4m/s	143.4m/s	143.4m/s
Doppler Resolution	0.14m/s	0.14m/s	0.14m/s
Field of View (H x V)	130° x 48°		
Angular Resolution (H x V)	0.35° x 0.9°		
Frame Time	50ms		

* Provided under request.

zPRIME Specification

Frequency	76-77 or 76-81GHz	
Interface	1000BASE-T1	1000BASE-T
Connections	6-pin automotive	Waterproof RJ45
Power Supply	+9-24V DC	802.3bt Type 3 (PoE++)
Power Consumption	22W avg, 34W peak based on transmit duty cycle	
Synchronization	IEEE 1588 PTP, gPTP	
Operating Temperature	-40°-85°C	
Ingress Protection	IP68 2-meter	
Dimensions	140 x 103 x 30mm	
Weight	490g	
Material	Polycarbonate black, 6061 aluminum black anodized	

4.4 Viewing the Point Cloud and Processed Data

4.4.1 Zadar Viewer

We provide a point cloud viewer for quick and easy visualization of the radar's output data. The viewer window allows you to visualize the point cloud and control a few features for visualization and filtering.

Basic commands:

- Left mouse button hold: tilt the view of the 3D point cloud.
- Right mouse button hold: translate or shift the view of the 3D point cloud.
- 'a': show axes
- '[space]': switches camera view from orthographic to perspective mode

Please refer to the figure on the next page for details about every section.

***Zadar Viewer* - Legend**

1. Main header, allows to switch between viewer types and save/upload the configuration file.

2. Radar Monitor provides an overview over the main status of the radar and some configurations. In case of multi-radar processing, you will visualize multiple rows, one for each sensor.

3. Display settings allow you to change the grid, font, zoom of the viewer and color.

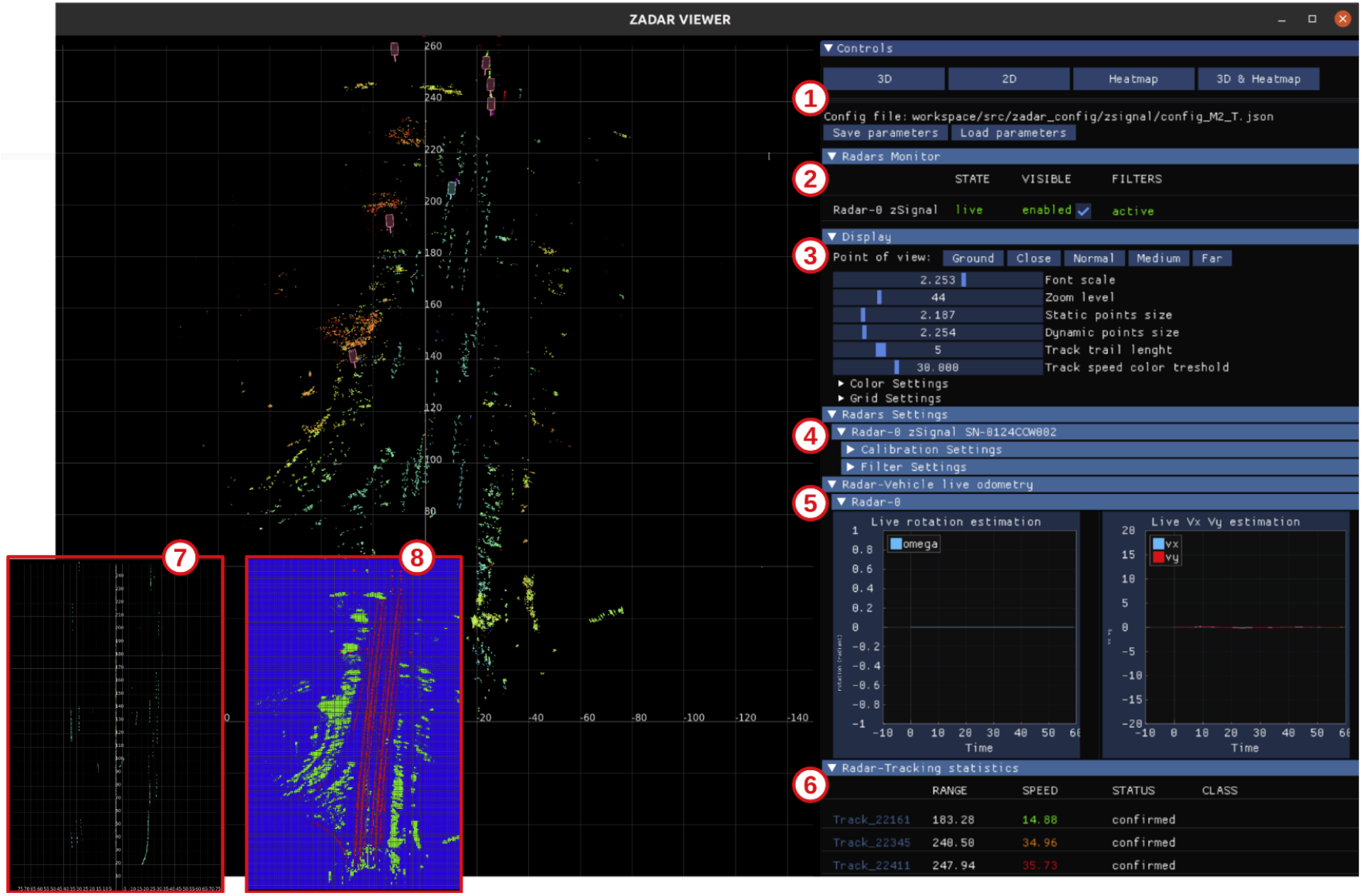
4. *[zSignal and zProx only]* For every radar connected you will be able to:
    a. Control the calibration settings to tune the position of the radar in respect to the vehicle. Also optional online automatic calibration of phi provided (Section 4.5 for more details about the automatic calibration procedure).
    b. Filter on the point cloud, applied to the x,y,z,range,doppler and snr. Filters follow a min-max logic, so only points in that range will be visible to the viewer.

5. *[Odometry deliverable only - provided under request] For every radar* allows to visualize the evolution of the estimation of linear velocity and rotational velocity of the moving vehicle where the radar is placed.

6. *[Tracking deliverable only - provided under request]* Allows to visualize the active tracks identified in the pointcloud live, also with some information about every track's speed, range, status (confirmed or confirmed_stopped), class identified, and fence id. You will also be able to see all the corresponding tracks directly on the point cloud with a square and a line indicating the direction of movement.

7. When the "2D" option is selected from the main header the default 3D point cloud will switch to 2D point cloud with on the x axis the doppler value and the y axis the range.

8. *[Tracking deliverable only - provided under request]* When the "Heatmap" option is selected from the main header, the default 3D point cloud will switch to an heatmap viewer where the green color intensity shows the presence of static points, red color for dynamic points and blue for no points.



4.4.2 Zadar Web Viewer

The webviewer is another way of viewing the point cloud using a website. Please start the radar software with ./run_radar as outlined above, once it is running navigate to the Zadar software package and under zadar_viewer_webapp double click viewer.html.

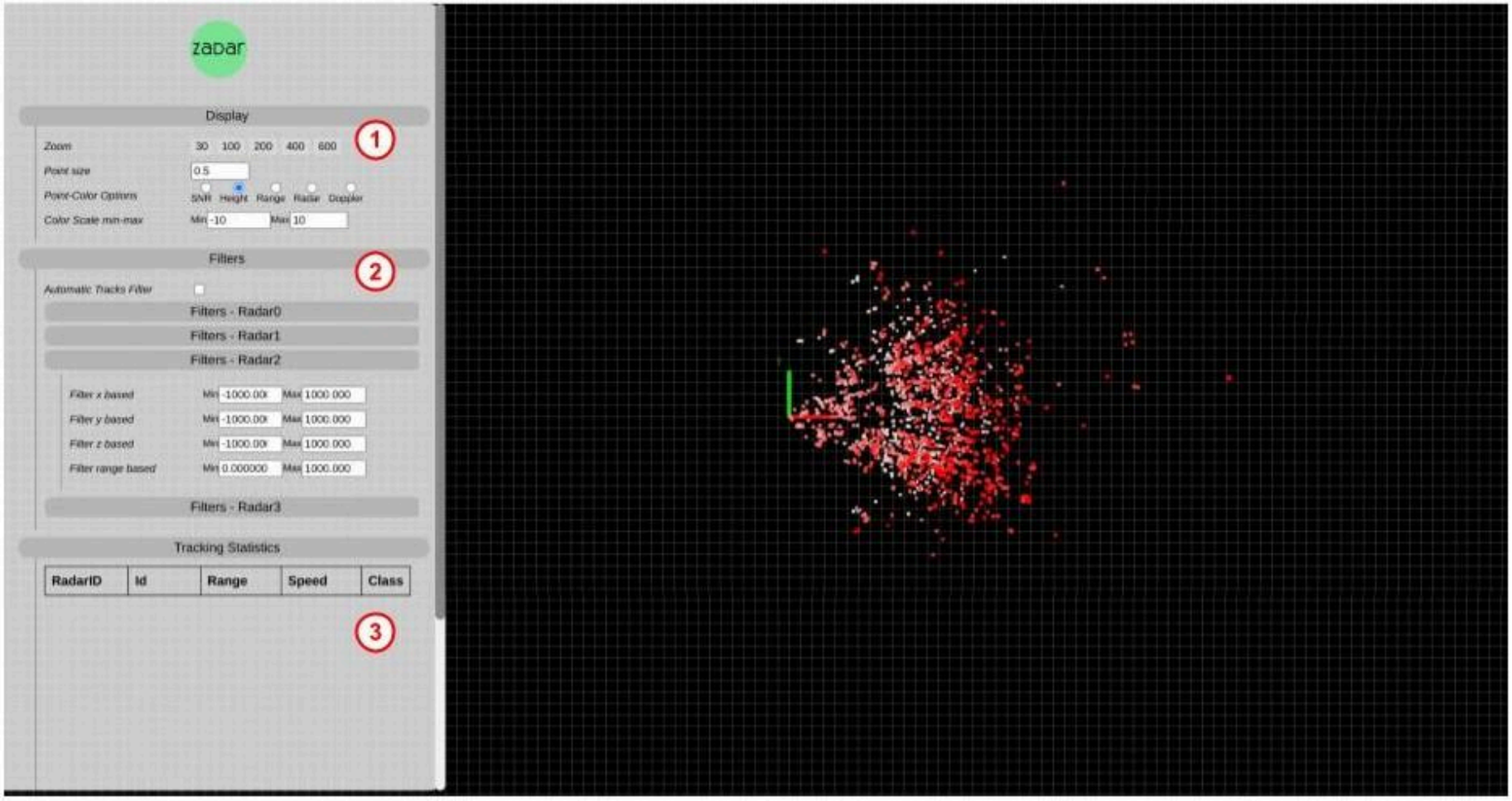
Basic Operations:

- Left mouse button hold: Rotate the view of the 3D point cloud
- Right mouse button hold: Translate or shift the view of the 3D point cloud.
- Scroll: Zoom into or out of the center of the display

Explanation:

***Zadar Web Viewer* - Legend**

1. Display Settings - Control basic display options such as the visualization mode and the point size
2. Filter Settings - Control filtering settings both global settings and per radar settings
3. Tracking Statistics - Displays tracking statistics and the radar they are from *[Tracking deliverable only - provided under request]*

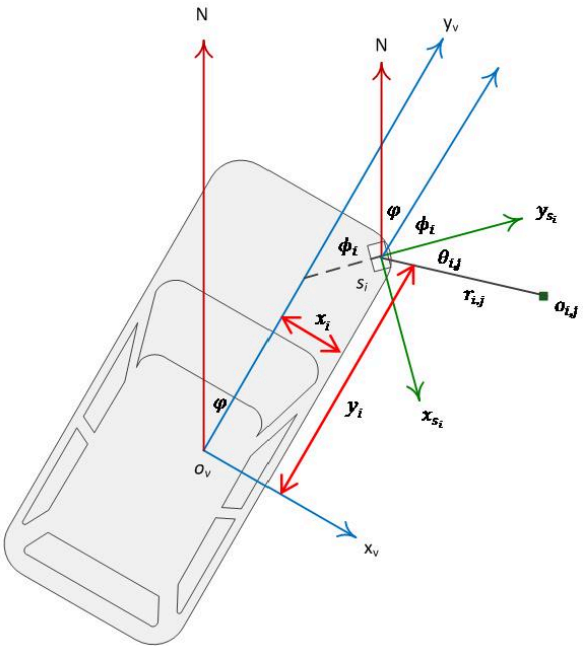


4.5 Radar Calibration

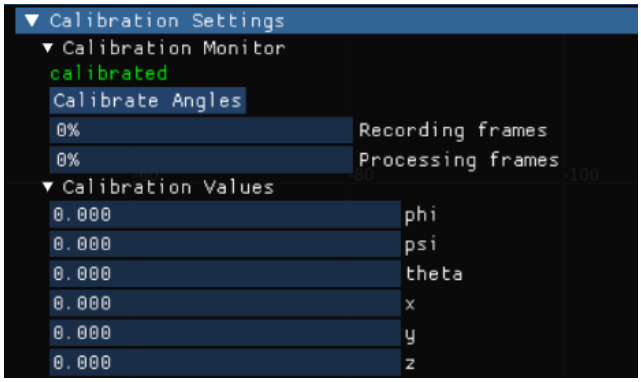
For moving radar applications only (mounted on a vehicle) with zProx and zSignal, we allow to fine tune the radar calibration in respect to the vehicle directly on the viewer. The angle displacement of the radar in respect of the center of mass for all 3 axes of the vehicle can either be measured precisely by the customer, or, Zadar's online calibration method can be utilized to accurately estimate φ_{rc} (azimuth phi).

The process for automatic online calibration will be as follows:

- Instruct the car driver or the ego platform (such as a robot) to proceed in a straight path without any wheel movement or rotation for at least 5 seconds.
- Press the calibration button on the viewer during this time (see section 4.4 for viewer calibration button)
- We recommend that the user align the theta and roll angles carefully using precise measurement devices to facilitate a smooth convergence of the algorithm.



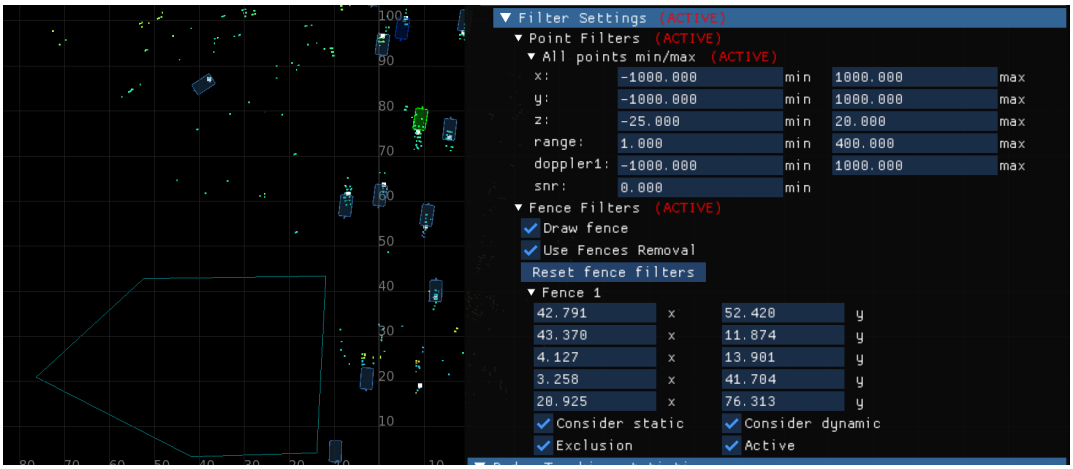
[Figure 3] Indicative car-radar odometry



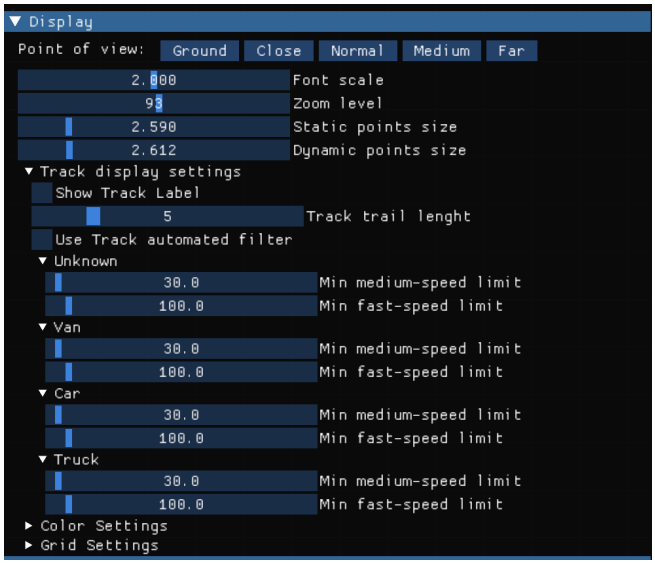
[Figure 4] Calibration settings on the viewer

4.6 Advanced configuration

- Fence filtering: when the viewer is in 3D or Heatmap mode by clicking ‘F’ on the keyboard you activate the interactive fence drawing that allows you with the left click of the mouse to directly place the fence vertices on the map on bird-eye view (means only x, y will be considered). Once placed on all the vertices by clicking again ‘F’ you will confirm and save the fence that will appear on the radar settings under the fence filter section. You can delete the created fences by un-toggle the “active” checkbox or using the “reset fence filters” (to delete all fences). You can edit the vertices by modifying the values x and y. You can create a max of 5 fences.



- Track display settings *[Tracking deliverable only]*: Under the “Display” section “Track settings” you have the option of activating/deactivating the track filtering that estimates the boundaries of the environment to properly filter out incorrect tracks, “Use Track automated filter” checkbox. Furthermore you can tune the speed limits for every class of tracks that will impact the coloring in the “Radar-Tracking statistic” table.



4.7 Capturing the Point Cloud and processed data

Record the data to CSV file

While the sensor is running you have the option to record the radar data into a .csv file. By running the following command into a new terminal tab, the script will create a new folder with point cloud, odometry, clusters and/or tracks data in csv format (1 frame per file). Ctrl+C to stop the recording.

```
$ sudo ./record_csv.sh <radar_num> <raw_can> <scan> <odometry> <clusters> <tracks>
```

For recording raw scan and not processed_scan/odometry/clusters/tracks example:

```
$ sudo ./record_csv.sh 1 on off off off off
```

> Output format
> In the “logs/” folder you will find a new folder containing the saved files in the following format with type=[scan,odometry,tracks]: */logs/*<timestamp>/*zadar_<type>0_frame<frame_id>.csv*

Receiving Real-Time data

these scripts should be ran while the radar is active with the run_radar.sh

In the “api_templates/” folder, we provide templates for Python scripts to allow receiving the point cloud, odometry, clusters and tracks frame by frame in real-time and connect to your own application. These templates are to be considered as a starting point to receive and forward the data to your own pipeline, and are openly available for you to modify and customize:

- ApiSocketScan.py
- ApiSocketClusters.py
- ApiSocketTracks.py
- ApiSocketOdometry.py
- ApiRos.py (*Requires a specific zadar software version for ROS, please ask support from the zadar software team*)

Recording data to a bagfile (ROS)

Requires a specific zadar software version for ROS, please ask support from the zadar software team

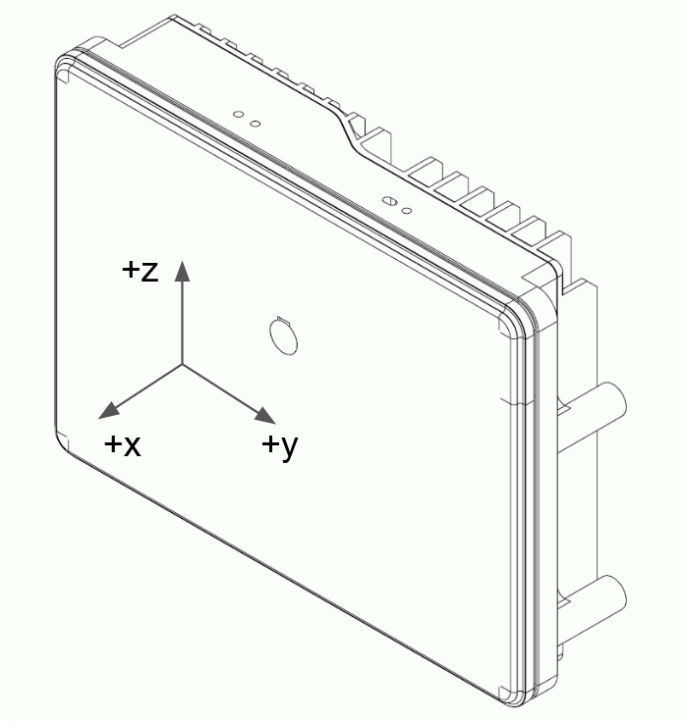
While the sensor is running, you also have the option to record the point cloud into .bag format (if you have ROS available on your system). By running the following command into a new terminal tab, the script will create a new folder containing the saved point cloud into .bag format (entire stream per file). Ctrl+C to stop the recording.

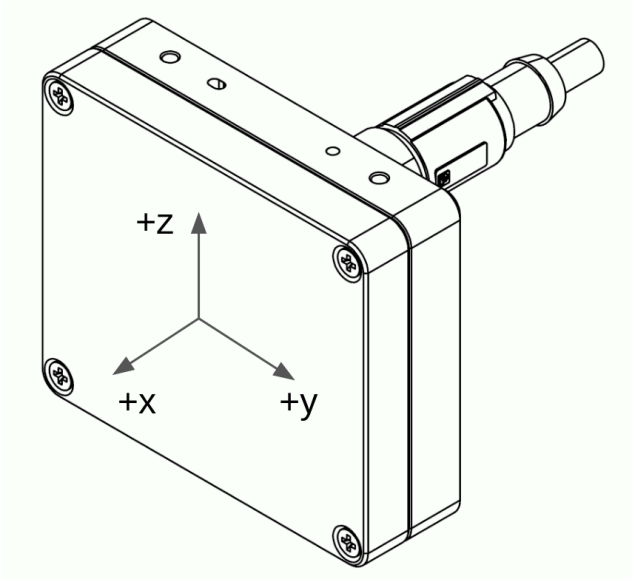
```
$ sudo ./record_ros.sh
```

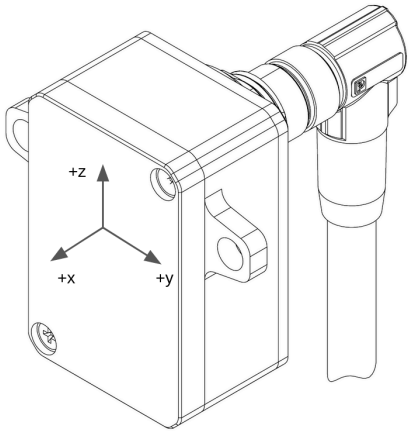
> Output format
> In the “logs/” folder you will find a new directory containing the saved files in the following format: /logs/<timestamp>.bag

5 Data Format

The output coordinate system per radar is as follows:







5.1. Point Cloud data-format

The point cloud data format of zSIGNAL, zPROX and ZPRIME is an array of points with following fields:

Header Per Frame

Field	Unit	Description
seq	-	Frame sequence number
timestamp	ns	Time elapsed since sensor boot
width	-	Number of points received in the frame

Data Per Point

Field	Unit	Description
x	meters	X coordinate of point in meters
y	meters	Y coordinate of point in meters
z	meters	Z coordinate of point in meters
snr	dB	Signal to noise ratio. A higher SNR means increased confidence in the received point
power	dB	Relative power level of the point normalized (Only available on zPrime)
range	meters	Radial range of the point in meters
noise	dB	Reserved variable
doppler	meters/second	Relative radial velocity of the point as measured via Doppler shift
adjusted_doppler	meters/second	Computed absolute radial velocity of the point in the ground reference frame
is_static	bool	Flag to indicate if a point has been detected as stationary (not moving) relative to the environment
frame_num	frame number	Frame number counter starting from 0

5.2. Odometry data-format

The odometry data format of zSIGNAL, zPROX and ZPRIME is a list of values with following fields:

Field	Unit	Description
vx	meters/second	Radar velocity over x axis smoothed
vy	meters/second	Radar velocity over y axis smoothed
omega	Radian/second	Radar rotational velocity smoothed
phi	degree	Radar yaw in respect to the vehicle
psi	degree	Radar roll in respect to the vehicle
theta	degree	Radar pitch in respect to the vehicle
raw_vx	meters/second	Radar velocity over x axis
raw_vy	meters/second	Radar velocity over y axis
raw_omega	Radian/second	Radar rotational velocity over z axis
frame_num	frame number	Frame number counter starting from 0

5.3. Clusters data-format

The clusters data format of zSIGNAL, zPROX and ZPRIME is a list of values with following fields:

Field	Unit	Description
cluster_id	integer	Cluster incremental id
x	meters	Cluster center over x axis
y	meters	Cluster center over y axis
z	meters	Cluster center over z axis
doppler	meters/second	Average cluster points doppler
snr	dB	Average Signal to noise ratio for the cluster points
noise	dB	Reserved variable
d_min	meters	Minimum doppler in the cluster points
d_max	meters	Maximum doppler in the cluster points
r_min	meters	Minimum range in the cluster points
r_max	meters	Maximum range in the cluster points
frame_num	frame number	Frame number counter starting from 0

5.4. Tracks data-format

The tracks data format of zSIGNAL, zPROX and ZPRIME is a list of values with following fields:

Field	Unit	Description
track_id	integer	Track incremental id
x	meters	Track coordinate over x axis
y	meters	Track coordinate over y axis
z	meters	Track coordinate over z axis
vx	meters/second	Track velocity over x axis
vy	meters/second	Track velocity over y axis
vz	meters/second	Track velocity over z axis
ax	meters/second^2	Track acceleration over x axis
ay	meters/second^2	Track acceleration over y axis
az	meters/second^2	Track acceleration over z axis
yaw	Radian	Track yaw in respect to radar origin and track position
speed	km/hour	Track resulting speed vector
state	integer	Track state confirmed or confirmed_stopped
class	integer	Track class [0=none 1=van 2=bike 3=motorbike 4=car 5=truck 6=train]
fence_id	integer	If fencing, the fence the track is in.
frame_num	frame number	Frame number counter starting from 0