Ubuntu Network Configuration

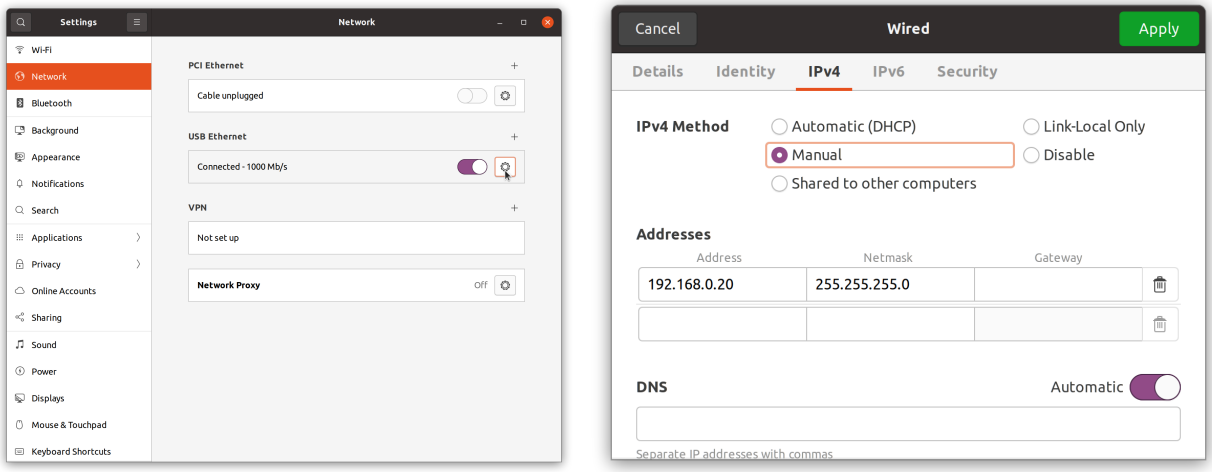

Figure 21. Ubuntu Network Settings

1. Navigate to the Settings pane
2. Select the ‘Network’ tab and configure your Ethernet adapter settings
3. Set ‘IPv4’ network settings to ‘Manual’ with an IP address of 192.168.0.20 and a subnet mask of 255.255.255.0
4. Select ‘Apply’
5. Check your network settings have been applied by using the `ifconfig` utility and ensure the settings displayed there match the settings you selected

Windows 11 Network Configuration

Figure 22. Windows Network Settings

Step 1: Access Network Settings

1. Click on the **Start** menu and select **Settings** (or press Win + I).
2. In the **Settings** window, navigate to **Network & Internet** from the left sidebar.
3. Under **Network & Internet**, click on **Ethernet**.

Step 2: Modify IP Configuration

4. Scroll down and locate the **IP assignment** section.
5. Click the **Edit** button next to **IP assignment**.
6. In the **Edit IP settings** window, select **Manual** from the drop-down menu.
7. Toggle **IPv4** to the "On" position.

Step 3: Enter IP Address Details

Copyright © 2025, Zadar Labs Inc. All rights reserved.

8. In the **IP address** field, enter: 192.168.0.20.
9. In the **Subnet mask** field, enter: 255.255.255.0

Step 4: Save and Apply Settings

10. Click **Save** to apply the static IP configuration.
11. Close the **Settings** window.

Step 5: Verify Configuration

12. Open **Command Prompt** (Win + R, type cmd, then press Enter).
13. Type the following command and press Enter:

```
None
ipconfig /all
```

14. Confirm that the Ethernet adapter now shows the manually assigned IP address (192.168.0.20) with the correct subnet mask and gateway.

Your Windows 11 device is now configured with a static IP address on the **192.168.0.0/24** network.

Copyright © 2025, Zadar Labs Inc. All rights reserved.

MacOS Network Configuration

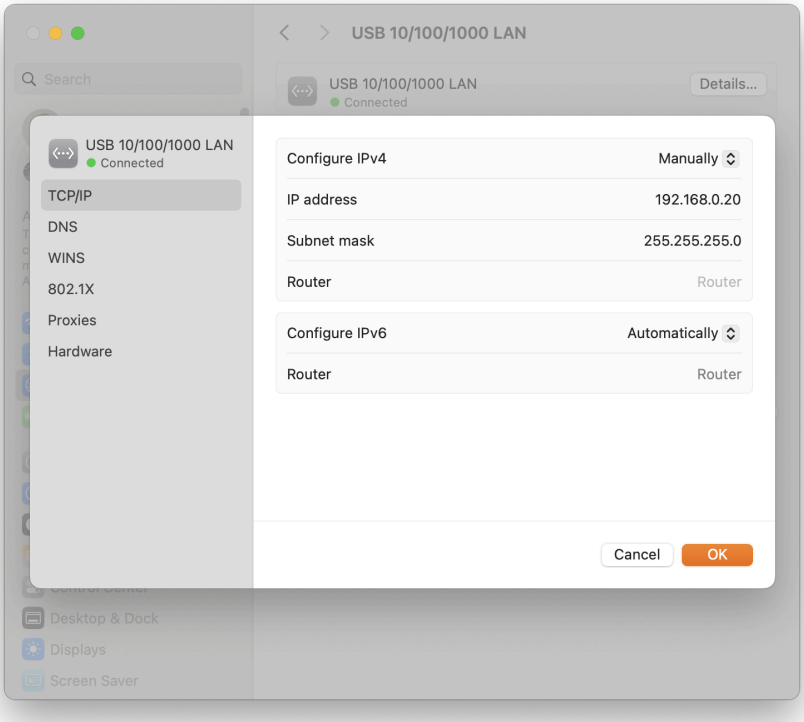

Figure 23. MacOS Network Settings

1. Plug in your ethernet adapter and the network cable.
2. Click the Apple Logo in the upper left corner > System Settings… → left-side panel Network.
3. In the main pane you'll see your interfaces. Select USB 10/100/1000 LAN (or whatever name your adapter shows).
4. Click the Details… button on the right.
5. In the pop-up sidebar, choose TCP/IP.
6. Next to Configure IPv4, open the dropdown and pick Manually.
7. Fill in the Fields:

Field	Value
IP Address	`192.168.0.20`
Subnet Mask	`255.255.255.0`
Router (Gateway)	*(leave blank or add if needed, e.g.* `192.168.0.1`*)*

8. Press OK → Apply.
9. Verify the new address appears under the interface name and the status dot is green.

Copyright © 2025, Zadar Labs Inc. All rights reserved.

4.4 Verifying Network Connection

Each sensor presents a unique local URL at `zPRIME-[radar serial number].local`.

> ☝ For example, a radar with serial number 4225CCW001 would be reachable at `zPRIME-4225CCW001.local`

A valid connection to the radar can be checked one of four ways:

1. Pinging the sensor directly at its IP address (by default, 192.168.0.10)
    a. `ping 192.168.0.10`
2. Pinging the sensor at its .local sensor URL
    a. `ping zPRIME-4225CCW001.local`
3. Visiting the configuration webpage in your web browser by its .local URL
4. Using the avahi-discover utility to discover the sensor using mDNS. This method can be used to recover the IP address of a sensor if it has been misconfigured or lost.

The IP address and DHCP configuration can be changed via the configuration webpage or via HTTP webAPI. Details on how to do this can be found later in this guide.

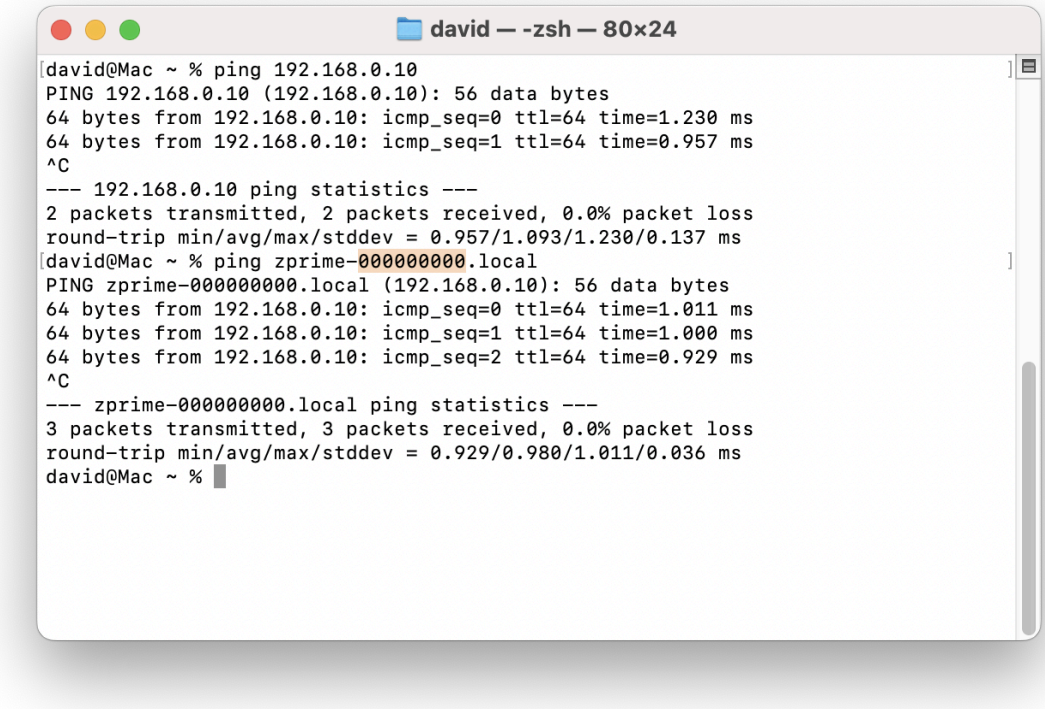

Figure 23. Pinging the Radar

Copyright © 2025, Zadar Labs Inc. All rights reserved.

4.4 Point Cloud Viewer and Management Webpage

Each zPRIME sensor hosts a local webpage that can be used to change radar settings and view the live radar point cloud, all through your internet browser.

This webpage can be reached one of two ways:

1. Browsing the radar's .local URL: `zPRIME-[radar serial number].local`
2. Browsing the radar's IP address, by default: `192.168.0.10`

If you have issues connecting or visiting the webpage:

- Make sure the radar is powered correctly and can be reached via pinging the network. Check the previous section to ensure the network is active and connected.
- The radar's .local URL should work even if the radar's IP address has been changed, as long as the radar's IP remains on the host computer's subnet.

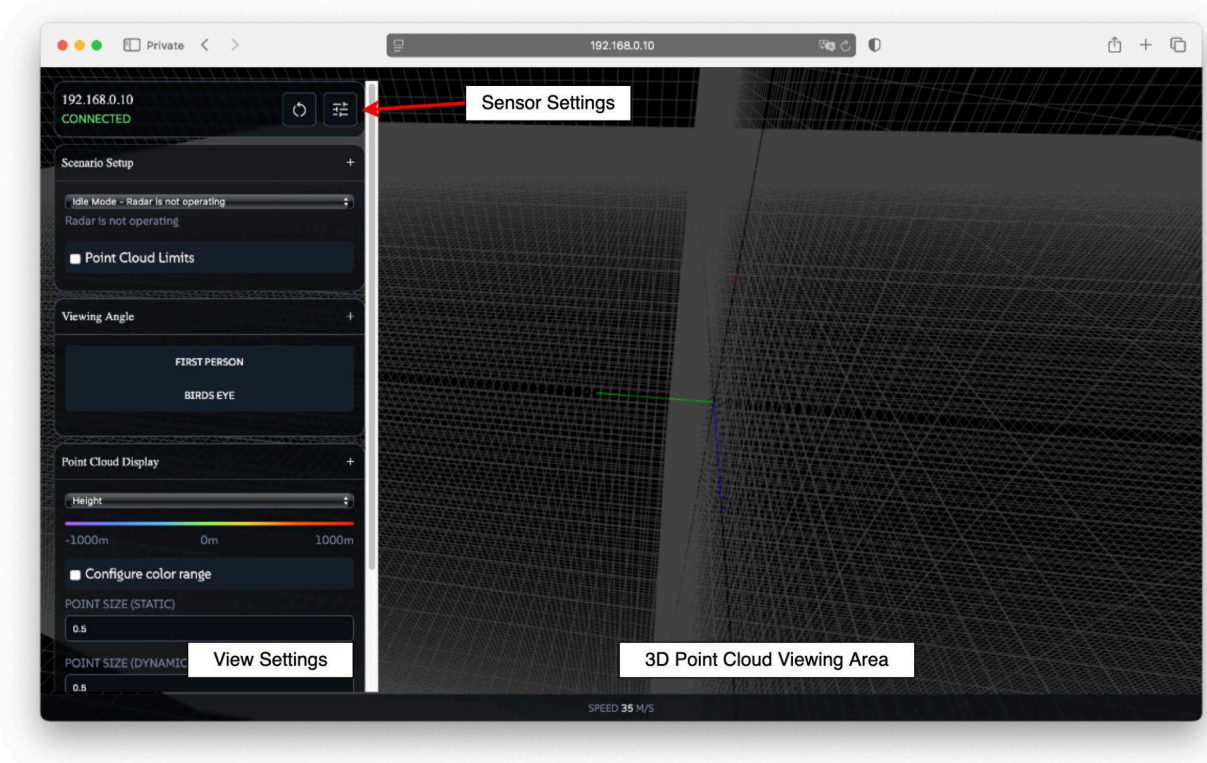

Figure 24. Radar Webpage

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Basic Operations

Action	Result
Left click - Drag	Rotate the 3D view
Right click - Drag	Translate/shift the 3D view
Left click an object	Brings up object information display
Scroll	Zoom in/out to the display center
S key	Snap to nearest flat view (upwards, sideways, or forward)
O key	Toggle between orthographic and perspective projections
X, Y, Z keys	Toggles YZ, XZ, and XY coordinate planes
Spacebar	Freezes data view to the current frame and stops updates to the display

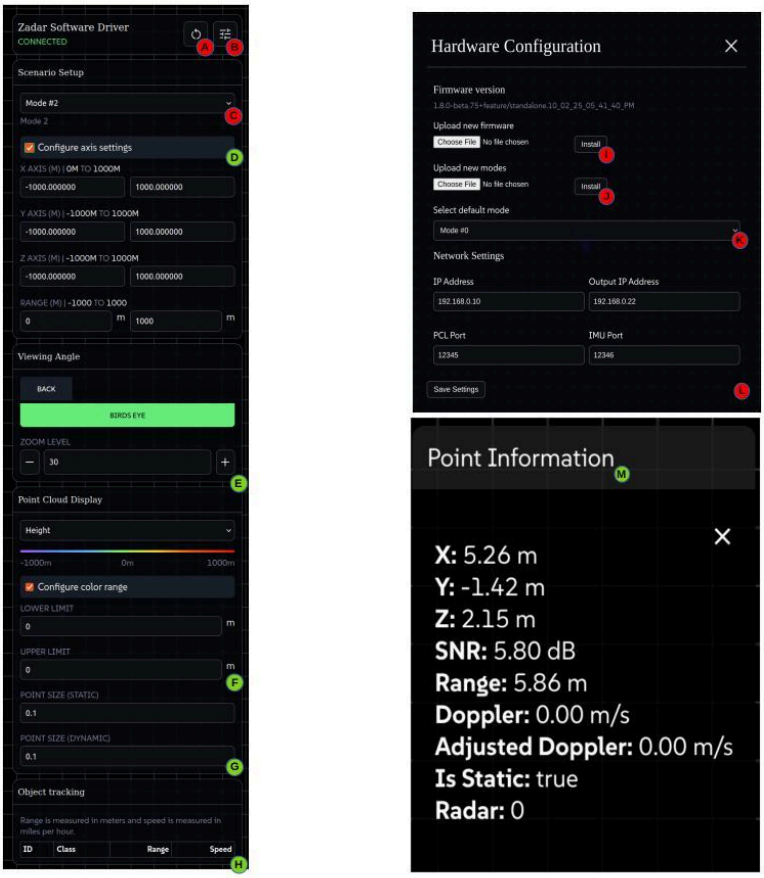

Figure 25. Radar Viewer Options

User Interface Functions

A. Reboot sensor
B. Open Hardware Configuration Menu
C. Switch between Running Modes
D. Filter Point Cloud based on X, Y, Z, range

Copyright © 2025, Zadar Labs Inc. All rights reserved.

E. Controls Viewer - Viewing Presets and Zoom Level
F. Coloring Ranges
G. Sizing of Points
H. Object Tracking Menu
I. Install new Firmware (this operation is slow and requires a reboot)
J. Install New Modes
K. Changes the Default Mode (mode which runs automatically at startup)
L. Change Networking Output Settings
M. Point Information

Radar Settings Configuration via Webpage

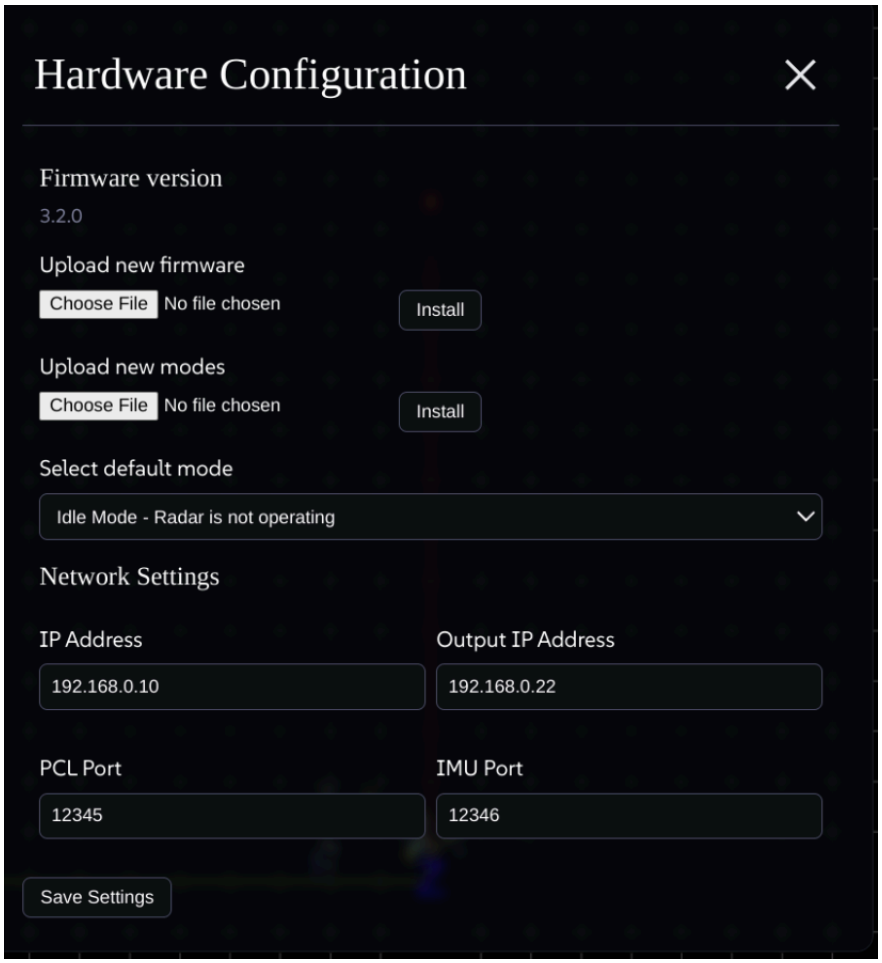

Figure 26. Radar Settings Pane

IP Address	Set the static IP address of the radar.
Output IP Address	Set the destination where UDP radar data is sent to. Typically, this is the host computer's IP address.

Copyright © 2025, Zadar Labs Inc. All rights reserved.

PCL port	Set the destination port which receives the radar data.
IMU port	Change the destination port which receives the onboard IMU data.

5 HTTP REST API Guide

This is an **optional** section, most radar settings can be configured manually via the web page.

The radar can be configured via our configuration REST API. This allows users to:

- Query radar status
- Start the radar
- Change operating modes
- Change data output destination
- Update firmware and operating modes
- And more

Specific details, exact specifications and example cURL commands for the REST API can be found in the API Documentation: zPRIME-Standalone-Configuration-Endpoints.pdf

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Rest API Endpoint Diagram

/api/v1

- device
  - modes [GET]
  - reboot [PUT]
  - clock [GET]
  - running_mode [GET, POST]
    - stop [POST]
  - network
    - ipv4
      - address [GET]
  - system
    - sensor_info [GET]
- system
  - network
    - ipv4
      - address [GET, PUT, DELETE]
  - output_destination
    - ipaddress [GET, PUT, DELETE]
    - pcl
      - port [GET, PUT, DELETE]
    - imu
      - port [GET, PUT, DELETE]
  - flash
    - image [POST]
    - modes [POST]
  - startup_mode [GET, PUT, DELETE]
  - system_config [DELETE]

Copyright © 2025, Zadar Labs Inc. All rights reserved.

API Endpoint Quick Guide

Refer to the API Documentation for specific cURL commands. Some common ones are here:

Setting	Endpoint Name (In API Documentation)
Radar IP Address	Device IP Address - PUT (Static IP)
None `curl -X PUT http://zPrime-${radarSN}.local:8080/api/v1/system/network/ipv4/address -H "Content-Type: application/json" -d '{"ip_address": "192.168.1.100"}'`	
Destination IP Address	Output Destination IP Address - PUT
None `curl -X PUT http://zPrime-${radarSN}.local:8080/api/v1/system/output_destination/ipaddress -H "Content-Type: application/json" -d '{"ip_address": "192.168.0.101"}'`	
Destination Point Cloud / DQE Port	Point Cloud (PCL) Port - PUT
None `curl -X PUT http://zPrime-${radarSN}.local:8080/api/v1/system/output_destination/pcl/port -H "Content-Type: application/json" -d '{"pcl_port": 12345}'`	
Destination IMU Data Port	IMU Port - PUT
None `curl -X PUT http://zPrime-${radarSN}.local:8080/api/v1/system/output_destination/imu/port -H "Content-Type: application/json" -d '{"imu_port": 12346}'`	
Radar Mode on Boot	Startup Mode - PUT

Copyright © 2025, Zadar Labs Inc. All rights reserved.

```
None
curl -X PUT http://zPrime-${radarSN}.local:8080/api/v1/system/startup_mode -H "Content-Type:
application/json" -d '{"mode": 1}'
```

5 Data Format and Interpretation

The radar will send point cloud data over UDP to the destination IP address and PCL port that you configured. You will need to create a UDP listener on these ports and IP addresses to receive the data. 4D Point Cloud data and optional SW outputs such as clusters, tracks, classified objects, and onboard radar odometry are also sent to the PCL port.

zPRIME comes with an onboard IMU. This IMU is not factory calibrated and the IMU data is provided without any accuracy guarantees. The configuration of the IMU cannot be changed, but the raw IMU data is accessible via UDP stream. The raw IMU data is sent on a different endpoint from the point cloud data since the IMU is another sensor inside of your zPRIME. Data from the IMU will be sent to the configured IMU port.

5.1 Receiving and Deserializing Radar (Point Cloud) Data

All radar data is sent over UDP to the destination IP address and port(s). At the destination address, a listener should be set up to receive the data. A parser must be written to reconstruct the incoming packets into a complete radar scan frame. Zadar provides sample code in Python and C++ to parse incoming radar data.

The radar scan can include:

- Point cloud (provided on all radars)
- Clusters
- Tracks
- Classification
- Radar-based odometry
- Custom outputs on a customer-by-customer basis

Copyright © 2025, Zadar Labs Inc. All rights reserved.

UDP Transfer Scheme

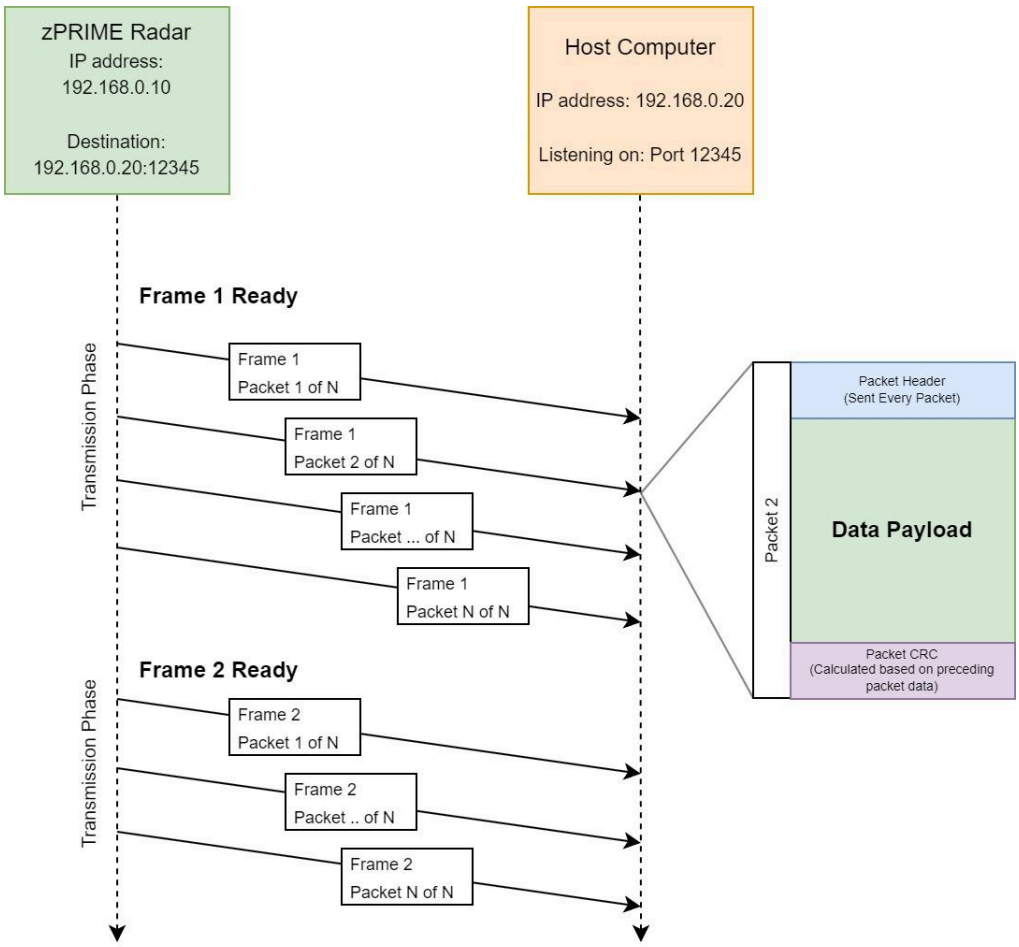

Figure 27. UDP Packet Transfer

Each frame is fragmented into packets, and each packet has a maximum size of 1500 bytes.
Each packet consists of:

1. Packet header
2. Data payload (protobuf binary data fragment)
3. Packet footer/CRC

The header identifies how to assemble the frame from these packets. When the full data payload is assembled, it can be decoded into the radar data stream using protobuf. Since UDP is used for transmission there is a possibility of data loss or reordering over the network. If data loss occurs the whole frame must be dropped, and data loss can be detected by ensuring:

1. All packets for a given frame have been received and are in order based on packet header data
2. The data in each packet received matches the CRC appended to each packet

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Packet Struct

Note: The values in the header are little-endian

Note: All timestamps come from the radar's internal clock. If a PTP grandmaster is active on the network, the radar's internal clock will be continuously synchronized to the PTP time.

```C/C++
struct __attribute__((__packed__)) UDPPointCloudHeader {
    uint16_t radar_id;                    // ID number for radar
    uint16_t data_format;         // 0 for pre-scaled values, 1 for unscaled values
    uint32_t points_in_frame;             // Number of points in the current frame
    struct timespec timestamp;            // Timestamp of current frame
    uint16_t frame_id;                    // Current frame number
    uint16_t packets_in_frame;            // Number of packets required for this frame
    uint16_t packet_index;                // Index of current packet, out of packets_in_frame
    uint16_t data_size;                   // Number of bytes in the data section of the packet
};

struct __attribute__((__packed__)) UDPPointCloudPacket {
    UDPPointCloudHeader header;
    char buffer[1400];
    uint64_t crc;
};
```

In the header, the *frame_id* tells you which frame this packet belongs to. *packets_in_frame* tells you how many packets this frame will have, and *packet_index* (0-indexed) tells you the order in which the packet belongs. *data size* tells you how many bytes of data are in the data segment.

To reconstruct a full data payload, concatenate data segments in order from index 0 to packets_in_frame - 1. The final 8 bytes of the packet are the crc, we use crc-64-xz as the algorithm to determine the crc of the packet. Please compare the packet crc with the computed crc of the entire packet excluding the crc section.

Once the last packet is received and the full data payload is assembled, pass the data payload to protobuf to deserialize the wire-format. Once deserialized, please refer to our .proto file and your protobuf library to handle and extract the data. Additional details about the meaning of our data outputs can be found in our software documentation.

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Sample Radar Data UDP Parser Code

Sample code can be found in the ZadarSoftwareDriver under api_templates, additionally linked below is the same sample code.

Standalone Python Examples

The above examples, covers parsing and deserializing the information from UDP to be usable by an application. The radar example shows the X, Y, Z, and doppler of 5 points from the scan along with the number of points in the point cloud.

The following examples will assume the existence of the deserialized ZadarFrame object in python named **radarframe**

Extracting 4D Point Cloud

The deserialized ZadarFrame object contains a radar_scan object that will contain the point cloud. Please refer to our .proto file to see which fields exist, and our software documentation contains information on the meaning of each field.

```python
radarframe.radar_scan
len(radarframe.radar_scan.points) #Number of points in the point cloud
radarframe.radar_scan.points[59].snr #SNR of the 60th point
```

Extracting Radar Odometry

Not all sensors have onboard radar odometry enabled.

The deserialized ZadarFrame object contains an odometry object that will contain the odometry data. Please refer to our .proto file to see which fields exist, and our software documentation contains information on the meaning of each field.

```python
radarframe.odometry
radarframe.odometry.vx #X-Axis component of velocity of the radar
```

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Extracting Clusters

Not all sensors have onboard radar clusters enabled.

The deserialized ZadarFrame object contains a clusters object that will contain cluster information. Please refer to our .proto file to see which fields exist, and our software documentation contains information on the meaning of each field.

```python
len(radarframe.clusters) #Number of clusters in radarframe
radarframe.clusters[5].doppler #Doppler of 6th cluster
```

Extracting Tracks

Not all sensors have onboard radar tracks enabled.

The deserialized ZadarFrame object contains a tracks object that will contain the tracking information. Please refer to our .proto file to see which fields exist, and our software documentation contains information on the meaning of each field.

```python
len(radarframe.tracks) #Number of tracks in radarframe
radarframe.tracks[2].track_id #Track id of the 3rd track
```

Copyright © 2025, Zadar Labs Inc. All rights reserved.

5.2 Receiving and Unpacking 6-Axis IMU Data

All IMU data is sent over UDP to the destination IP address and IMU receive port. At the destination, a listener should be set up to receive the data. A parser must be written to deserialize the incoming IMU frames. Zadar provides sample code in Python and C++ to parse incoming IMU data. The crc code is computed with crc-64-xz on the packet up to but excluding the crc.

IMU Data Struct

Note: The values in the header are little-endian

Note: All timestamps come from the radar's internal clock. If a PTP grandmaster is active on the network, the radar's internal clock will be continuously synchronized to the PTP time.

C/C++

```
typedef struct UDPHeader {
    uint64_t stream_id;           // stream type, e.g. ID_ZDEST
    uint32_t version;             // protocol version
    uint32_t radar_id;            // reserved for radar id
    uint32_t frame_id;            // frame #
    uint32_t data_size;
    uint32_t dopplers;
    uint32_t ranges;
    struct timespec timestamp;
} UDPHeader_t;

typedef struct imu_data
{
  uint64_t diagnostic_timestamp;
  uint64_t accelerometer_timestamp;
  uint64_t gyroscope_timestamp;
  float acceleration_x;
  float acceleration_y;
  float acceleration_z;
  float angular_rate_x;
  float angular_rate_y;
  float angular_rate_z;
} imu_data_t;

typedef struct IMUPacket {
    UDPHeader_t header;
    imu_data_t data;
   uint_64t crc;
} IMUPacket_t;
```

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Sample IMU Data UDP Parser Code

Sample code can be found in the ZadarSoftwareDriver under api_templates, additionally linked below is the same sample code.

[Standalone Python Examples]

The above examples cover parsing and deserializing the information from UDP to be usable by a user application. The IMU example shows the components of the gyroscope and accelerometer readings printed to the console.

5.3 Coordinate Systems

zPRIME follows ISO 8855 coordinate system standards:

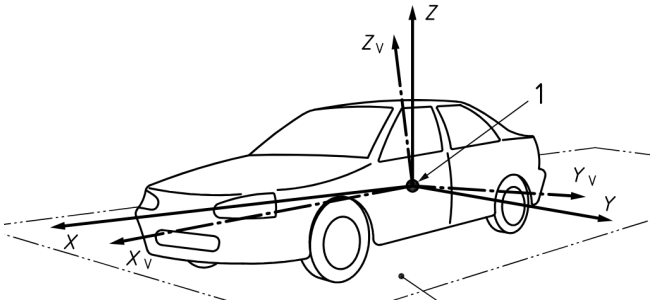

Figure 27. Coordinate System Conventions

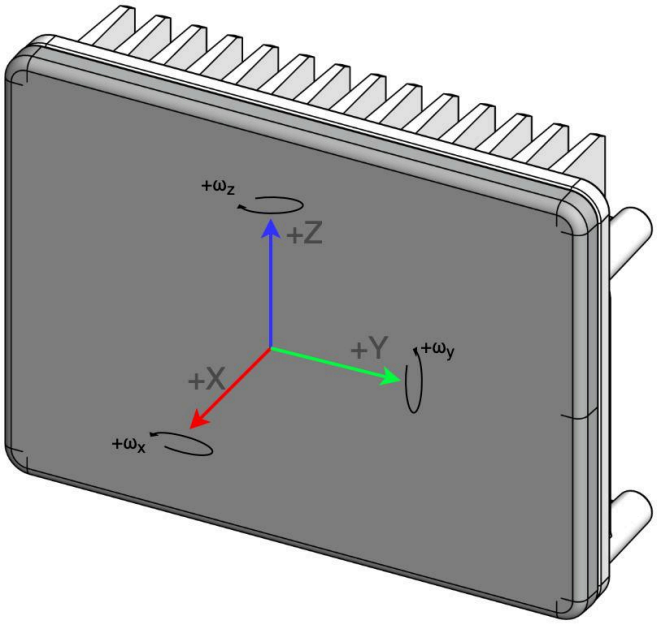

Figure 28. Radar and IMU Coordinate Systems
Omega (rotational rate) is clockwise positive

Copyright © 2025, Zadar Labs Inc. All rights reserved.

Figure 29. IMU Offsets

Copyright © 2025, Zadar Labs Inc. All rights reserved.

5.4 PTP Time Synchronization

zPRIME supports synchronized operation using either IEEE1588v2 PTP and 802.1AS gPTP profiles. When a grandmaster is present on the network, zPRIME will automatically synchronize its internal clock and timestamps according to the grandmaster.

PTP Network Connection Topologies

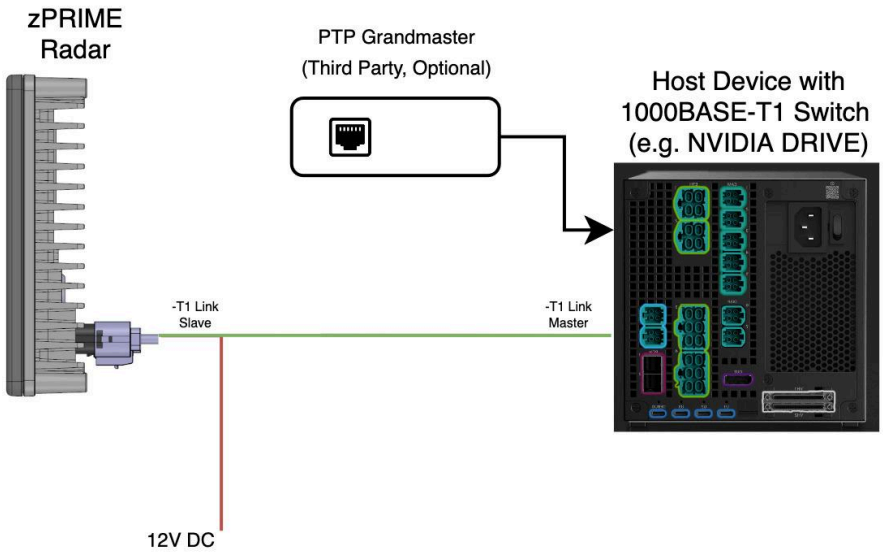

Figure 30. PTP Network Topology with 1000BASE-T1 Network

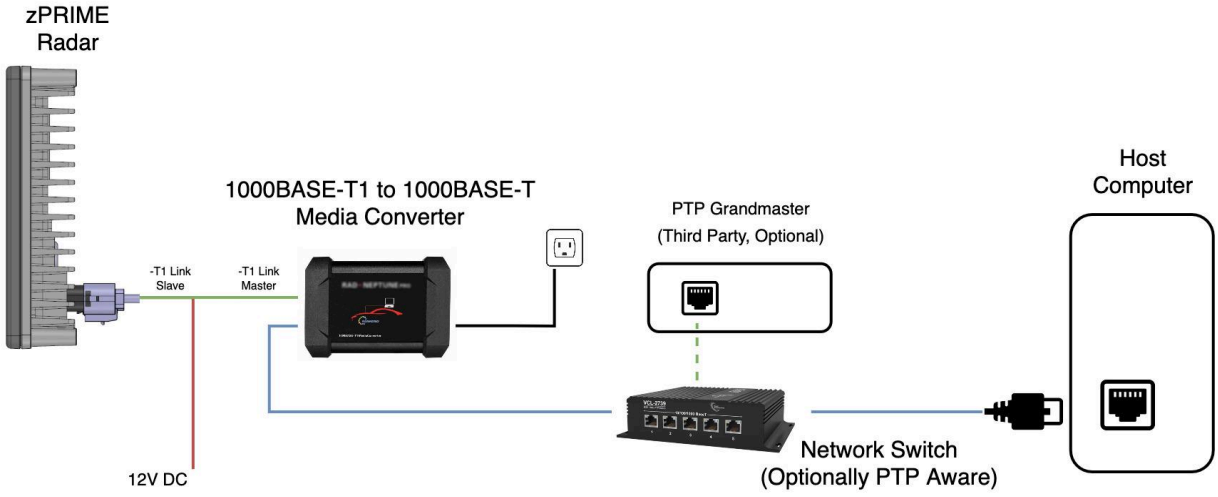

Figure 31. PTP Network Topology with RJ45/1000BASE-T Network

Copyright © 2025, Zadar Labs Inc. All rights reserved.

PTP Timestamping & Triggering

zPRIME supports PTP timestamping of frame start times and synchronized triggering of frames.

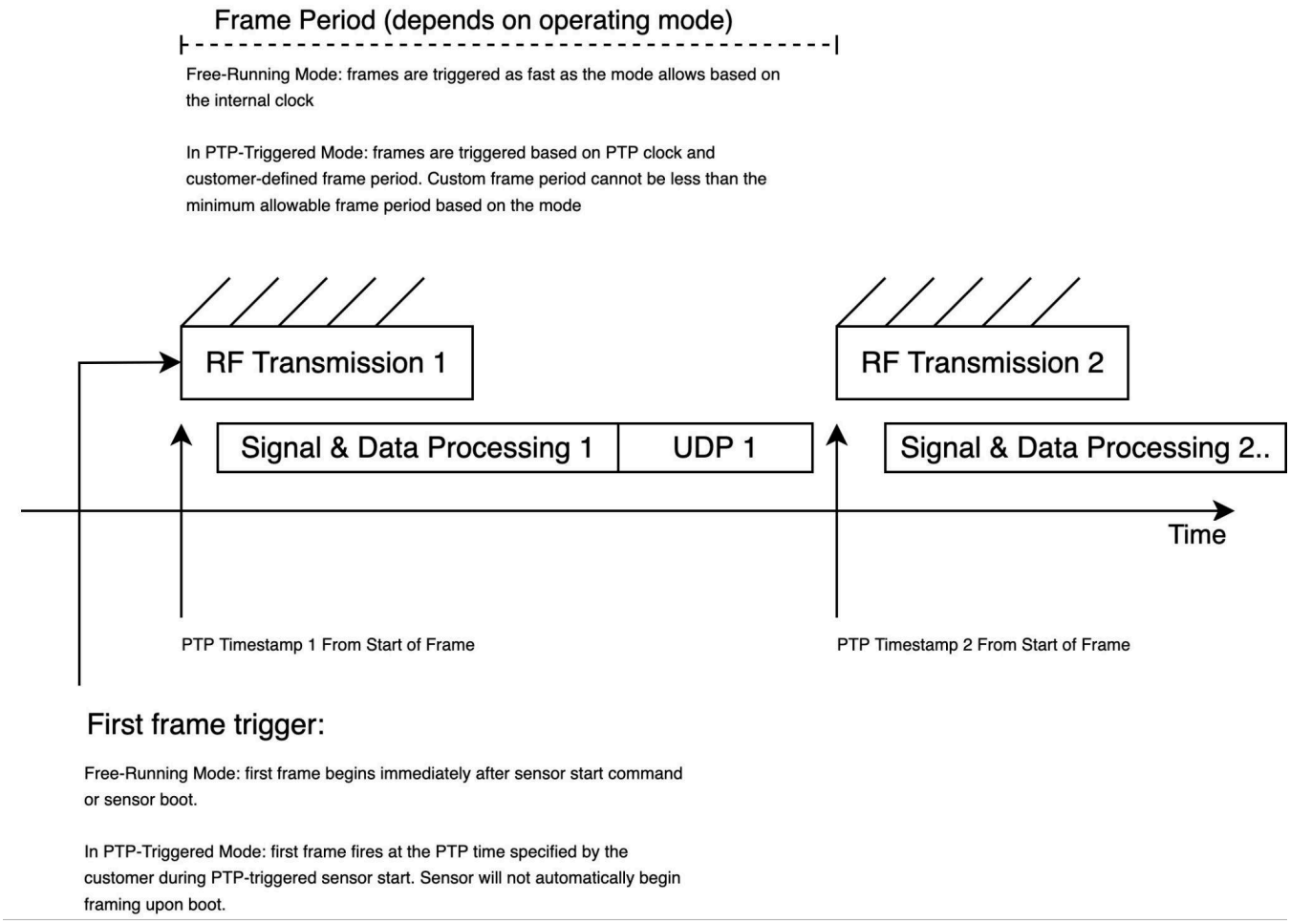

Figure 32. PTP Timing Scheme

Copyright © 2025, Zadar Labs Inc. All rights reserved.

6 Troubleshooting and Maintenance

6.1 Basic Diagnostics, Common Issues, and Fixes

I can't visit the webpage of the radar.

- Check your network configuration settings and see if you can ping the radar via the network. Make sure to wait 7-8 seconds after power on for the radar to boot before trying to ping or visit the webpage.
- Try visiting the bare IP address (default 192.168.0.10) and the local address of the radar (zPRIME-[serialnumber].local) in your browser.
- Make sure your computer's network is properly configured with the correct IPv4 static address. Use the network configuration section to double check this is done properly.

I've lost the IP address of zPRIME.

- If you still have contact to zPRIME, ping the sensor or visit the webpage via the local address of the radar (zPRIME-[serialnumber].local). The ping destination or the webpage settings pane will tell you the IP address of the sensor.
- If you do not have contact to zPRIME and you do not know if your local IP address is set correctly, you can use Avahi/mDNS discovery to find the IP address of the sensor. In Linux, this can be done via the avahi-browse utility.

I'm not sure if zPRIME is booting. How can I check?

- After applying power, zPRIME should immediately link up a 1000BASE-T1 link master. Even if you don't have a host connected, the link up should occur. Check the link status on your 1000BASE-T1 PHY or media converter- if it doesn't come up, check the 1000BASE-T1 cable and make sure the power supply is stable.

6.2 Firmware Updates

Firmware updates can be applied to zPRIME in two ways:

1. Via web browser: visit the web configurator for zPRIME. In the settings pane, there will be an option to upload a new firmware image. Select your downloaded firmware image here and wait for the upgrade process to complete. It may take a few minutes and the radar will automatically reconnect after the process is complete. DO NOT DISCONNECT POWER OR DATA TO THE RADAR DURING THIS PROCESS OR DAMAGE MAY OCCUR.
2. Via HTTP/REST API: there is an endpoint to upload new firmware images for automatic firmware upgrade.

Copyright © 2025, Zadar Labs Inc. All rights reserved.

7 Appendix

7.1 Technical Specifications

8 Revision History

Date	Revision	Notes
8/07/2025	1.0	Initial revision of unified user manual

Copyright © 2025, Zadar Labs Inc. All rights reserved.