

---

## ROSboard

ROS node that runs a web server on your robot. Run the node, point your web browser at <http://your-robot-ip:8888/> and you get nice visualizations.

**ROS1/ROS2 compatible.** This package will work in either ROS version.

**Mobile friendly.** Designed so you can walk around next to your robot with a phone while viewing ROS topics.

**Light weight.** Doesn't depending on much. Consumes extremely little resources when it's not actually being used.

**Easily extensible.** Easily code a visualization for a custom type by only adding only one .js file here and referncing it inside the top of index.js.

You can run it on your desktop too and play a ROS bag.

Also be sure to check out my terminal visualization tool, ROSshow.



---

## Running it the easy way (without installing it into a workspace)

```
1 source /opt/ros/YOUR_ROS1_OR_ROS2_DISTRO/setup.bash
2 ./run
```

Point your web browser at `http://localhost:8888` (or replace localhost with your robot's IP) and you're good to go.

## Installing it as a ROS package

This ROS package should work in either ROS1 or ROS2. Simply drop it into your `catkin_ws/src/` or `colcon_ws/src/` and it should just work.

For ROS 1, run it with `roslaunch rosboard rosboard_node` or put it in your launch file.

For ROS 2, run it with `ros2 run rosboard rosboard_node` or put it in your launch file.

## FAQ

### How do I write a visualizer for a custom type?

Just add a new viewer class that inherits from `Viewer`, following the examples of the default viewers. Then add it to the imports at the top of `index.js` and you're done.

### How does this work in both ROS1 and ROS2?

I make use of `rospy2`, a shim library I wrote that behaves like ROS1's `rospy` but speaks ROS2 to the system, communicating with `rclpy` in the background. This allows using the same ros node code for both ROS1 and ROS2, and only needs slight differences in the package metadata files (`package.xml` and `CMakeLists.txt`, hence the configure scripts). It does mean that everything is written in ROS1 style, but it ensures compatibility with both ROS1 and ROS2 without having to maintain multiple branches or repos.

### Why don't you use `rosbridge-suite` or `Robot Web Tools`?

They are a great project, I initially used it, but moved away from it in favor of a custom Tornado-based websocket bridge, for a few reasons:

- It's less easy to be ROS1 and ROS2 compatible when depending on these libraries. The advantage of doing my own websocket implementation in Tornado is that the custom websocket API speaks exactly the same language regardless of whether the back-end is ROS1 or ROS2.

- 
- Custom implementation allows me to use lossy compression on large messages (e.g. Image or PointCloud2) before sending it over the websocket, robot-side timestamps on all messages, and possibly throttling in the future.
  - I don't want the browser to have "superuser" access to the ROS system, only have the functionality necessary for this to work.
  - I also want to add a basic username/password authorization at some point in the future.
  - Many times in the past, the robot web tools are not available immediately on apt-get when ROS distros are released, and one has to wait months. This depends on only some standard Python libraries like [tornado](#) and optionally [PIL](#) and does not depend on any distro-specific ROS packages, so it should theoretically work immediately when new ROS distros are released.

## Credits

This project makes use of a number of open-source libraries which the author is extremely grateful of.

- Tornado: Used as a web server and web socket server. Copyright (C) The Tornado Authors Apache 2.0 License
- simplejpeg: Used for encoding and decoding JPEG format. Copyright (C) Joachim Folz MIT License
- psutil - Used for monitoring system resource utilization. Copyright (C) Giampaolo Rodola BSD 3-clause license
- Masonry: Used for laying out cards on the page. Copyright (C) David DeSandro MIT License
- Leaflet.js: Used for rendering sensor\_msgs/NavSatFix messages. Copyright (C) Vladimir Agafonkin CloudMade, BSD 2-clause license
- Material Design Lite - Used for general UI theming and widgets of the web-based client. Copyright (C) Google, Inc. Apache 2.0 License
- jQuery - Used for client-side DOM manipulation. Copyright (C) OpenJS Foundation MIT License
- litegl.js - Used as a wrapper for the WebGL API for 3D visualizations. Copyright (C) Evan Wallace, Javi Agenjo MIT License
- glMatrix - For Matrix manipulations for 3D visualizations. Copyright (C) Brandon Jones, Colin MacKenzie IV. MIT License
- rosbag.js - Used for reading ROS 1 .bag files. Copyright (C) Cruise Automation MIT License

- 
- uPlot - Used for all time-series plots. Copyright (C) Leon Sorokin MIT License
  - JSON5 - Used for encoding/decoding JSON5 messages. Copyright (C) Aseem Kishore, and others. MIT License
  - JetBrains Mono - Used for all fixed-width text in the web UI. Copyright (C) The JetBrains Mono Project Authors SIL Open Font License 1.1
  - Titillium Web - Used for all variable-width text in the web UI. Copyright (C) Accademia di Belle Arti di Urbino and students of MA course of Visual design SIL Open Font License 1.1