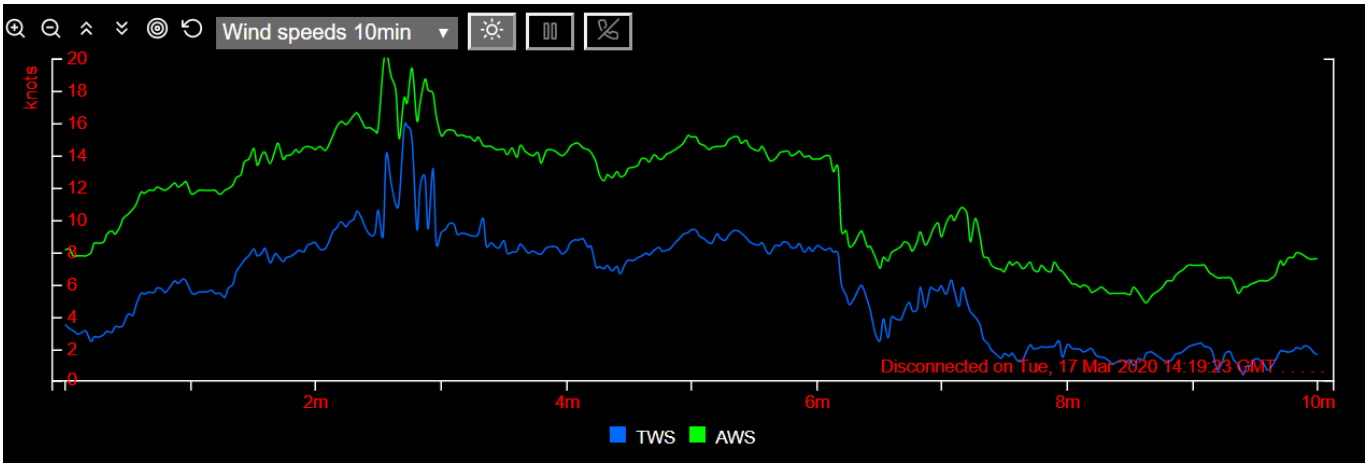
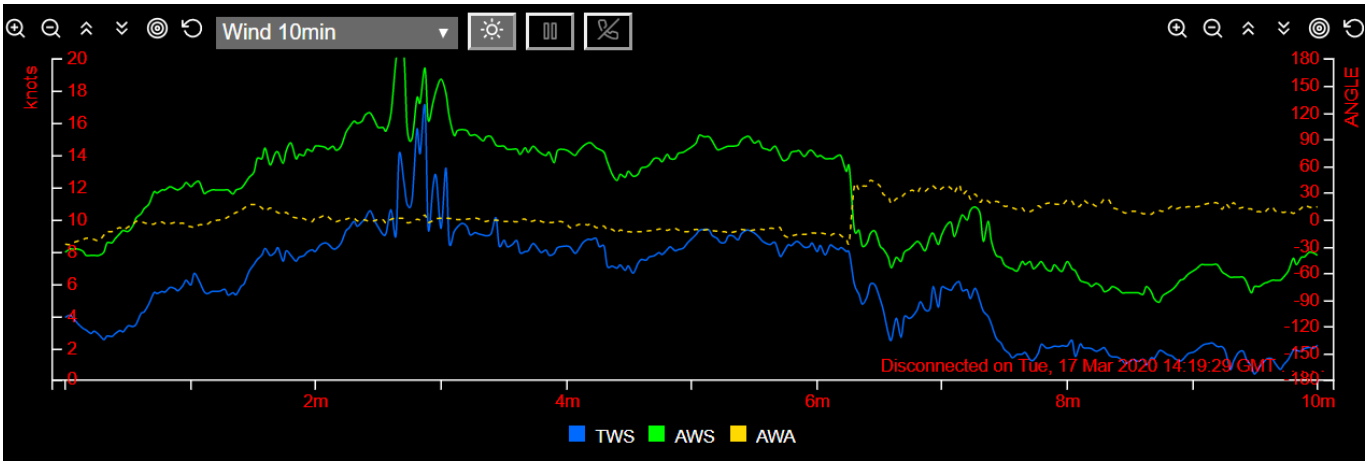
Signalk-stripcharts: generate strip charts from Signal K live boat data.

A stripchart displays the most recent boat data (from one or more Signal K paths) as a graph along a time axis (x-axis). Legends identify by abbreviations what paths are charted. Here is a chart with a 10 minutes time window plotting the true wind speed (TWS) and apparent wind speed (AWS):



The above chart has a y-axis on the left hand side. Optionally, as shown on the chart below (right hand side), a y2-axis may also be defined with its own unit. The dotted line corresponds to the plot according to the y2 axis, here the apparent wind angle (AWA).



The browser window can display two such charts on top of each other. For instance you may have wind speeds on the top chart and boat speeds on the bottom chart, or wind speeds with two different time windows, say 10 minutes and 24 hours. Those two charts can be selected with a dropdown list from a set of active charts.

Now it's time to install and start using the default charts specifications provided.

Installation

Signalk-stripcharts comes with all required dependencies (including c3 charting library and d3 visualization library).

Choose one of the following methods as applicable.

Available as a Signal K Webapp:

If Signal K node server (https://github.com/SignalK/signalk-server-node) is installed on your server:

- log in to the Signal K dashboard
- install Signalk-stripcharts from its Webapps Appstore
- restart the dashboard You can now start Signalk-stripcharts launcher from the Webapps Appstore Installed apps

Installation on a client device:

- Download the ZIP file from https://github.com/edefalque/signalk-stripcharts and unzip in a folder

Installation on a node server (typically the boat Signal K server):

- Use npm: [sudo] npm install signalk-stripcharts This will install Signalk-stripcharts and its dependencies c3 and d3 under the closest node_modules folder higher up in the hierarchy.

Basic usage

Use index.html as a launch menu.

Here you can choose the Signal K server and choose a set of charts, then push the "Launch Signalk-stripcharts" button.

When the charts are displayed:

- hover on a legend: the corresponding plot is highlighted, the others dimmed; the corresponding Signal K path is displayed above the legend
- hover on a plot line, a tooltip is diplayed with the values corresponding to the legends
- by clicking on a legend you can toggle the dim status of the legend and of the corresponding plot

You may wish to bookmark the launched page(s) for easier later launching. Modify the query parameters (server & specs) as needed, e.g. if you have defined your own charts psecifications (see later section "How it works").

y- and y2-axis buttons

These buttons affect the size and position of the plotted lines corresponding to the axis:

From left to right:

- zoom in
- zoom out
- raise the plot relative to the axis
- lower the plot relative to the axis
- try to center the plot at present time

- reset to the initial scale and position settings

Drop-down list

You can select any chart in the set for display. If you choose the same chart as the other one displayed, the chart will be displayed on the whole window.

Main buttons

From left to right:

- toggle day/night display
- toggle pause/play plotting
- disconnect from Signal K server (reload the page if you want to connect again)

They apply to the whole window.

How it works

The result is governed by some general options, the chart specifications and the units.

A chart is primarily specified by:

- a name
- the length of the time axis (timeWindow); typically 10 minutes, 2 hours or 24 hours
- the averaging period (further explained below)
- the y axis range and unit (unit conversion is taken care of)
- the same properties for the y2 axis, if present
- the list of Signal K paths to be charted

For each path the charted value can be the average, the maximum and/or the minimum of the values received from Signal K during the averaging period. The averaging period (avgInterval) should be set such that the ratio timeWindow/avgInterval is between 300 and 1000 (i.e. a reasonable number of plots along the timeWindow); avgInterval must be longer than the sampling period defined in the Signal K subscription period (subscribePolicy.period), which is typically 1 second. By default, the chart is refreshed every avgInterval (provided that some new data has come for that chart). When data stops coming for a particular path, it is "hotdecked" from the last data received until new data arrives; hotdecking stops after twenty seconds and thereafter the corresponding data will be 0 (in Signal K unit) indicating probably that the corresponding instrument was disconnected or switched off.

A chart specification is provided as a Javascript object. Here is the specification for the chart shown at the top of this document:

```
const Wind_speeds_10min =
    { stripChartName: "Wind_speeds_10min",   // stripChartName MUST BE THE SAME as
the containing object name
    // and hence follows javascript variables naming rules (see
```

```
https://javascript.info/variables#variable-naming)
        timeWindow: 600,            // 10min
        avgInterval: 2,             // 2 sec
        intervalsPerRefresh: 2,     // default 1
        x: { label: "min/sec ago", tickCount: 11 },   // a tick every min (10, + 1
for zero)
        y: { unit: "Knot" },
        paths:
        [
            { path: "environment.wind.speedTrue", AVG: "TWS" },  // average is
plotted, legend is "TWS"
            { path: "environment.wind.speedApparent", AVG: "AWS" }
        ]
    }
;
```

By default, the chart is refreshed every avgInterval (provided that some new data has come); refreshing can be made slower with the intervalsPerRefresh property in order to spare some processing.

Related chart specifications objects are grouped into a set.

When the application is started, the following parameters must be provided:

- the address and port of the Signal K server (defaults to the address & port the page is loaded from)
- the file name containing the chart specifications set (without .js extension).

The application subscribes to the Signal K server deltas for all the paths in the set. The values are then continuously collected and aggregated for all charts in a set. At any one time, two charts can be displayed as selected by the user from drop down lists. If the user selects the chart 'none' in one of the drop down lists, the remaining chart covers the whole window area.

Charts can be paused. When paused, data collection continues. So the charts display will catch up when "unpaused". When the chart window is minimized or not in view, the charts are paused likewise and will catch up when brought to view. This minimizes processing load.

There is no persistency: when the window is closed it is disconnected from Signal K and the buffers are "lost". In order to reconnect reload the page.

The charts specifications sets are each stored in a .js file in the ./specs folder. The following specifications files are provided at installation:

- sail.js
- environment.js
- engines.js (provided as a partially tested example) Each of them can be started in its own browser tab and run concurrently.

In a specification file, a chart specification can be derived from another chart and only the properties that differ need to be specified (e.g. a two-hours chart can be derived from a ten-minutes chart, with most of the properties inherited). Inheritance is provided at the first level of properties only.

Signalk-stripcharts buffers having no persistency, they cannot be used to store the history. Persistency and more powerful charting capabilities can be provided with InfluxDB and Grafana as explained here (https://github.com/tkurki/signalk-to-influxdb/blob/master/README.md) or could be provided with the help of the history capability of Signal K if available.

Customization

Currently, customization is easy if the package is installed on the client, but less if it is installed on a server as the specs files may be then less accessible. If installed on a Raspberry PI from Signal K webstore, they will probably be in /home/pi/.signalk/node_modules/signalk-stripcharts/specs/ .

Options

Options governing all charts are given in signalk-stripcharts/js/stripcharts_options.js. See comments in the file. Those comments explain how time is managed in signalk-stripcharts.

The following options can also be entered as query parameters after the url when launching stripcharts.html: timeTolSec and logTypes. They will then override the values in stripcharts_options.js.

Chart specifications

The specifications files are installed in signalk-stripcharts/specs/. Ample comments are provided for those features that were not explained above.

Some specs refer to paths that are unlikely to be provided by many current Signal K servers.

New specifications files can be easily derived from those provided at installation. Copy them under another name and edit them with a text editor, or better with a code editor such as Geany or Visual Studio Code.

The launch menu has a button that lists all paths and sources currently provided by the selected Signal K server. Switch on all your instruments and systems in order to obtain a full list of what you can chart.

Legends and lines colors

Colors can be specified per legend at the bottom of the chart specifications file in order to insure consistency accross multiple charts of a same set. If not provided colors will be assigned automatically by the c3 library. See sail.js for how to assign colors.

Units

signalk-stripcharts/specs/units provides the list of Signal K units and charting units, with the conversion factors and algorithms. It also provides the following default properties for the y and y2 axis as a function of the charting unit:

- label
- range (min and max)
- tick count and format

Those can be overridden in the chart specs.

A special unit "Percent" is provided. It allows to plot values of different units on a same "Percent" y or y2 axis by providing reference values in the Signal K unit for 0% and for 100%. See engines.js for an example with comments.

Browser compatibility

Signalk-stripcharts uses ECMAScript 2015 (ES6). It was mostly developed on Raspbian Chromium 72.0.3626.121 and Windows 10 Chrome 80.0.3987.132.

It seems to also work fine on:

- Windows 10 with Edge 18362 and with Firefox-ESR 68.6.0esr
- Raspbian with Firefox-ESR 52.9.0
- Androïd with Chrome 78.0.3904.108 (on mobile phone, it is recommended to display one chart only)

Recent non-ESR versions of Firefox show some svg rendition problems.

CPU and memory requirements

On a Raspberry Pi 3B+ with Chromium: a short burst of approximately 50% cpu consumption is observed when 2 charts corresponding to 6 paths are refreshed on the window (typically every 4 seconds for 10 minute timeWindow charts, or every 10 seconds for 2 hour timeWindow charts); this is only when the window tab is visible and "playing". Augment avgInterval and/or intervalsPerRefresh if you want to reduce the frequency of those bursts. Between refreshing bursts, managing the data collection and aggregation for 10 charts, each with approximately 3 paths, takes less than 1% CPU. For the same amount of charts and paths, memory footprint is about between 40K and 60K at all time.

Troubleshooting

Syntax errors in the charts specifications files will be caught by javascript. Logical errors in the specs and errors in the data returned by Signal K may also be reported. Inspect the console as needed.

Some tracing options are provided in ./js/stripcharts_options.js. Tracing occurs below the displayed charts.

Functional improvements

- ☐ Filter by sources: in a chart specs, at path level, specify sources wanted as an array
- ☐ Filter out from the specs the path/sources which are never provided by the signalk server and/or derive missing path values from other paths when possible
- ☐ Provide a better way to create/manage custom specs files and preserve them when installing new versions of the package

Technical improvements

Signalk-stripcharts is essentially an html/javascript front-end application. It was developed with no or very limited prior knowledge of Linux, Javascript, HTML, CSS, Signal K, Node.js and other js-related tools. Therefore much technical improvement is of course possible. In particular the following improvements could be considered without structural changes:

- ☐ Transpiling (Babel)

- ☐ Minification.
- ☐ Retrieval of boat data is inspired from Signal K websocket example. Possibly it could be replaced by the more robust https://www.npmjs.com/package/@signalk/client

Any hints are most welcome.

Credits

The following open source or free software contributions have been especially inspiring:

- NavMonPC stripcharts: http://www.navmonpc.com/
- Signal K: http://signalk.org/
- c3.js: https://c3js.org/

Thank you to all who have shown the way!