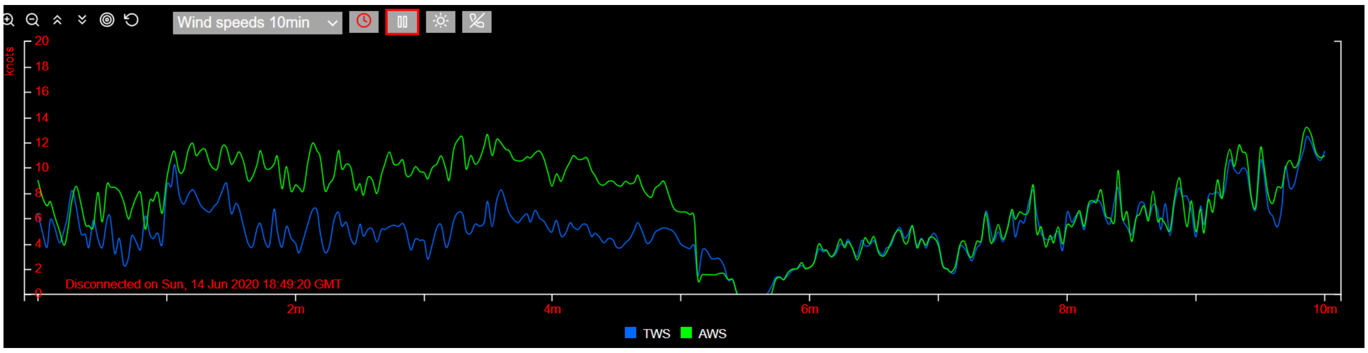
Signalk-stripcharts - Generate strip charts from Signal K live boat data and from InfluxDB past boat data

> New in version 0.2.x: Charting past data from InfluxDB made easier (you may need to adapt your options)

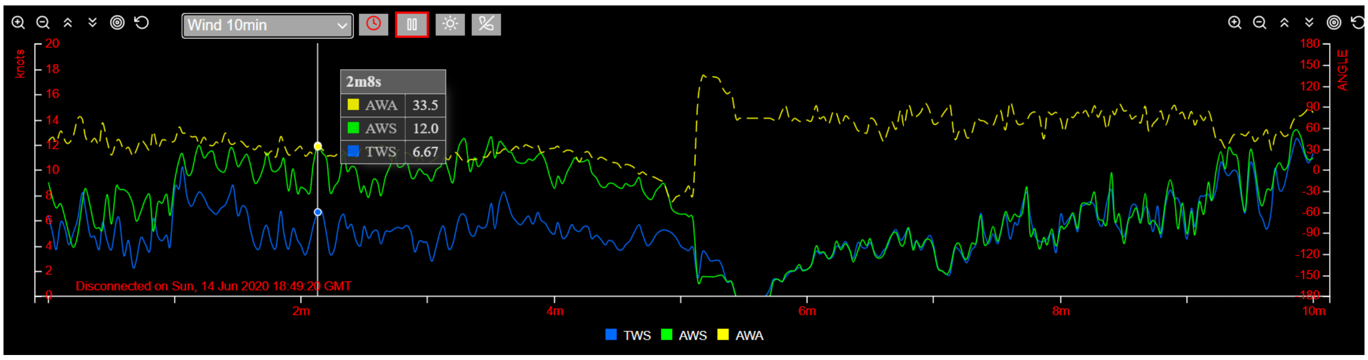
Jump to TOC

Introduction

A stripchart displays the most recent live boat data from one or more Signal K paths as a graph along a time axis (x-axis). The graph flows from left (most recent) to right: this direction allows the most recent data to be observed near the left hand side y-axis and its scale. Legends under the x-axis 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):



All charts have a y-axis with its scale and unit 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 scale and unit. The dotted line corresponds to the plot according to the y2 axis, here the apparent wind angle (AWA). When the mouse cursor browses the chart, the values associated to the nearest data points are displayed as "tooltips".



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.

This documentation first explains how to chart live data from Signal K server. Then it will explained how past boat data from InfluxDB, a time series database, can be charted as well.

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

Table of contents

Table of contents generated with markdown-toc

Installation

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

Choose one of the following 3 methods as applicable or desired:

Choice 1 - Install as Webapp from Signal K Dashboard

This is the recommended set up.

If Signal K node server is installed on your server:

- login to the Signal K dashboard
- open the left hand side menu
- select `Appstore`, then `Available` and find `Signalk-Stripcharts` (if it is not listed, it is probably already installed)
- click on the install icon next to the version number
- when installation is finished, click on the dashboard `Restart` button

You can now start signalk-stripcharts launcher from the dashboard `Webapps` menu.

Choice 2 - Install on a node server - typically the boat Signal K server

- Use npm: `[sudo] npm install signalk-stripcharts`

This will install signalk-stripcharts under the closest node_modules folder higher up in the hierarchy.

Choice 3 - Install on a client device

- Download the ZIP file and unzip it in a folder.

(by default, it will be installed in a folder named `signalk-stripcharts-master`; rename folder `signalk-stripcharts`)

↥ back to TOC

Basic usage

Start the launch menu:

- **If installed as a Signal K Webapps**: from the browser on your client or on the Signal K server, access the Signal K dashboard (e.g. `mysignalkserveripaddr:3000/admin/#/dashboard` or `localhost:3000/admin/#/dashboard`), then start Signalk-Stripcharts from the dashboard `Webapps` menu. Or start directly with `mysignalkserveripaddr:3000/signalk-stripcharts` or `localhost:3000/signalk-stripcharts`.

You may also:

- **If installed on a (Signal K) node server**: from your client browser enter url `mynodeipaddr:port/signalk-stripcharts`
- **If installed on a client**: double click on the file `...path.../signalk-stripcharts/index.html`

On the launch menu you can choose the Signal K server (same or distinct from the node server where signalk-stripcharts is installed) and choose a set of charts, then push the `Launch signalk-stripcharts` button. If you are connected to the Internet try first with the default selections of the launch menu.

When the charts are displayed:

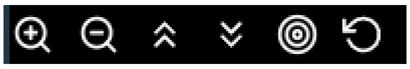
- hover on a legend: the corresponding plot is highlighted, the others dimmed; the corresponding Signal K path is displayed above the legend
- hover on a plot line, a tooltip is diplayed with the values corresponding to the legends
- by clicking on a legend you can toggle the dim status of the legend 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 specifications (see later section "How it works").

The following sections explain signalk-stripcharts accessing and charting only live data provided by a Signal K server: this is the default configuration at installation. A later chapter will explain how to set up an InfluxDB database to store historical boat data and chart the past data with signalk-stripcharts.

y- and y2-axis buttons

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



From left to right:

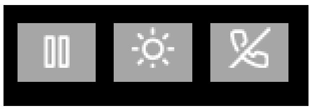
- zoom in
- zoom out
- raise the plot lines (y-axis range is shifted)
- lower the plot lines
- try to center the plot lines at present time (x = 0)
- reset to the initial scale and position settings

These buttons appear also on top of the right hand side y2 axis if used.

Drop-down list

You can select any chart in the set for display. If you choose `none` the other chart will be displayed on the whole window (similar effect if the same chart is selected twice).

Main buttons



From left to right:

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

They apply to the whole window.

↥ back to TOC

How it works

A chart contents and rendition is governed by the **chart specifications**, **unit conversion** and some **general options**.

In Signal K, data types are identified as paths: e.g. the true wind speed is identified by the path `environment.wind.speedTrue`.

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 (`avgInterval` explained below)
- the y axis range and unit (unit conversion is taken care of)
- the same properties for the optional y2 axis
- the list of Signal K paths to be charted and corresponding abbreviated legends

For each path the charted value can be the average (`AVG`), the maximum (`MAX`) and/or the minimum (`MIN`) 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` option), 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, the corresponding plot line is interrupted 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 =
    // for JavaScript variables naming rules see https://javascript.info/variables#variable-naming
    { stripChartName: "Wind_speeds_10min",   // use preferably the same as the containing object name
        timeWindow: 600,             // 10min in seconds
        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); however refreshing can be made slower with the `intervalsPerRefresh` property in order to spare some processing.

Related chart specification objects are grouped into a set.

When the application is started as an url, the following "query" parameters must be provided:

- `server` with the address and port of the Signal K server (defaults to the address & port the page is loaded from)
- `specs` with 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 (using the streaming WebSocket API).

At any one time, two charts can be displayed as selected by the user from the drop down lists. If the user selects `none` in one of the drop down lists, the remaining chart is extended to the whole window area.

Charts can be paused by clicking ⏸. When paused, data collection continues "behind the scene". And the charts display will catch up when ▷ is clicked. 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.

When the window is closed it is disconnected from Signal K and the buffers are "lost". In order to reconnect reload the page. (Persistency is only provided with the InfluxDB option explained further down)

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`

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; this means in particular that to add, modify or delete a path property, you must provide a new `paths` (plural) property with all its `path` (singular) property.

↥ back to TOC

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 then be less easily accessible. If installed on a Raspberry PI from Signal K Appstore, they will probably be in `$HOME/.signalk/node_modules/signalk-stripcharts/specs/`.

Chart specifications

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

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.

Derived data

Some path values are provided to Signal K by the instruments; some other path values may need to be derived. E.g. true wind speed, if not provided by the instruments, can be derived from apparent wind speed, apparent wind angle and speed over ground.

Use the following Signal K dashboard menus:

- `Server/Plugin Config/Derived Data` in order to configure and start path derivation from existing paths; see https://github.com/SignalK/signalk-derived-data (you may have to instal the plugin first from Signal K `Appstore`)
- `Data Browser` in order to visualize the active and derived paths with their current values

Filtering by sources

Different sources may provide data for the same path; as needed, the `path` property may be extended to filter a specific source, e.g. `navigation.speedOverGround[gps.1]`. The sources corresponding to a path on your Signal K server can be obtained from the launch menu provided that the corresponding instruments are on and connected. A certain path may be listed several times in the chart specs, e.g.:

```
{ path: "navigation.speedOverGround[gps.1]", AVG: "SOG1" },
{ path: "navigation.speedOverGround[gps.2]", AVG: "SOG2" },
{ path: "navigation.speedOverGround", AVG: "SOGx" },
```

The first two "path lines" will collect data each respectively from `gps.1` and from `gps.2`. The third line will collect SOG data from all sources (including the sources identified in line 1 and 2).

An example is provided in sources_filtering_example.js (read comments in the file before running: it runs on a test log file provided with Signal K node server).

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. Specification file sail.js shows how to assign colors.

If not provided, colors will be assigned automatically by the c3 library.

Unit conversion

units.js provides the list of Signal K units and charting units (not yet a complete set), 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 `tick.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.

General options

Options governing all charts are given in stripcharts_options.js. They are fully documented in the code. Some comments also explain how time is managed in signalk-stripcharts.

A few options deal with the usage with InfluxDB: ignore those for the moment.

The options can be overidden by setting them in the chart specs files, e.g. `options.pointRadius = 2`.

Any of the following options can also be entered as query parameters after the url when launching stripcharts.html: `timeTolSec`, `logTypes`, `policy`, `period` and `minPeriod`. They will then override the corresponding values in stripcharts_options.js and in the charts specs file.

Invalid option values will be replaced by defaults.

↥ back to TOC

Charting past data from InfluxDB

General description

InfluxDB is an open source time series database designed to handle high write and query loads and provides a SQL-like query language called InfluxQL. We will be using InfluxDB 1.x.

When the InfluxDB interface is not activated, signalk-stripcharts operates in the "**live mode**": the x-axis time origin is always the current time (possibly temporarily "paused"). With the InfluxDB interface, an additional mode is provided: the "**past mode**" which allows to choose another x-axis time origin in the past.

With the InfluxDB interface, the following behaviours are provided with two additional buttons:

- in **live mode**: while live data is being plotted and does not fill the complete time window yet, you can fill the right part of the chart with past data from an InfluxDB database by clicking on a "fill-right" button
- in **past mode**: after clicking on the "clock" button , a pop-up calendar can be used to define a new origin for the x-axis and the chart will be filled with historical data from the InfluxDB database starting at and after that new time origin.

In order to use those features, you first need to:

- install InfluxDB and start it,
- create an InfluxDB database and configure it,
- provide the database name in stripcharts_options.js,
- install and run the Signal K to InfluxDB plugin (this will write Signal K data to the database),

Detailed steps are provided in the following sections. You may also refer to InfluxDB get-started documentation.

Install InfluxDB and create a database

In order to install on a Raspberry PI refer to this startup guide.

Start the influx command line tool in a terminal:

```
influx
```

Then create the database "boatdata":

```
CREATE DATABASE boatdata
```

Signalk to InfluxDB Plugin

Signal K to InfluxDB plugin can write all or selected simple numeric Signal K values to InfluxDB with a chosen sampling period (parameter `Resolution`).

From Signal K Dashboard select Server/Plugin Config > InfluxDB writer. Fill the plugin form, e.g.:

- check `Active`
- change Host and Port as might be needed
- Database: `boatdata`
- Batch writes interval: `5` (or any other sensible value)
- Resolution (ms): `1000`
- If your are simulating from log files, consider checking `Override time with current timestamp`
- Define type of list and paths included/excluded
- Submit

Provide the influxDB database in the options

The relevant lines in stripcharts_options.js are:

```
var options = {
  ...
  useInfluxDB: true,   // <===== YOU MUST SET TO TRUE as false is the default
  influxDB: {
    dbName: "boatdata",   // if missing 'boatdata' is the default
```

```
      SK2InfluxResolution_ms: 1000   // optional, default 1000
    }
    ...
  }
```

Alternatively you could leave stripcharts_options.js unchanged and add the line `options.useInfluxDB = true;` at the top of the charts specs files for which you want InfluxDB access.

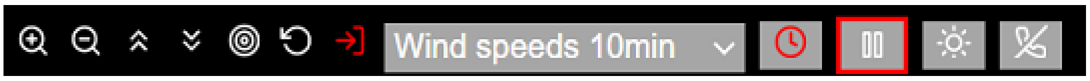
Note: the InfluxDB plugin `resolution` parameter value should be provided as SK2InfluxResolution_ms if not equal to 1000 msec.

Now start a chart as explained above in Basic usage and access past data as explained below.

Note: If you have been using InfluxDB before with several Retention Policies and Continuous Queries, you may need to further refine the options: read Stripcharts options for InfluxDB

Usage in live mode and past mode

When `options.useInfluxDB` is `true` two buttons with **red icons** are added to the user interface. The window top line now looks like this:



A "fill-right" button is added to the right of the y-axis buttons and a "clock button" is shown next to the top charts selection drop-down list.

When the charts are first displayed, the pause/play button has a red border: this indicates the "**live mode**" is active: the charts start being filled with live data. If you click the fill-right button the corresponding chart will be completely filled up to the right with past data from InfluxDB (if such data exist in the database). This action can be performed individually for each chart. Once a chart is filled up the chart fill-right button is hidden.

Let's now explore and chart older data.

Click the clock-button . A date/time picker window pops up (with the current GMT time). The clock-button is now red-bordered indicating that the "**past mode**" is active (while in that mode and unless Signal K is disconnected, Signal K live data continue to be received and buffered "behind the scene").

Pick a date/time a few minutes, hours or days in the past and click `OK` : the origins of both charts in view are set to the picked time and the charts are filled with historical data from InfluxDB provided that InfluxDB was active at that time and contains historical data.

You can further explore the past by selecting another chart in the drop-down list or by clicking the clock-button and selecting another date/time.

When the play button is clicked, the **live mode** is again activated, the charts x-axis are set to the current time and, unless SignalK was disconnected, become live-updated again (playing state).

Optimize with retention periods and constant queries

You can use the database created as is, however there is a way to configure it in order to optimize disk space and query resources consumption while providing access to a long history of boat data.

A natural solution is to downsample the data, i.e. keep the high precision raw data for a limited time, and store the lower precision, summarized data longer, as explained here: downsampling and data retention. It relies on the concepts of Retention Policy (RP) and Continuous Query (CQ).

The following target configuration will be configured, chosen to be compatible with the sampling periods (avgInterval) of the charts specs provided with signalk-stripcharts:

RPs & CQs	**"1day"**	**---"CQ1"-->**	**"1week"**	**---"CQ2"-->**	**"1month"**
CQ clause		GROUP BY time(10s)		GROUP BY time(120s)	
RP sampling period	1 sec		10 sec		120 sec
avgInterval ...	2 sec		10 sec		120 sec
... of chart types	"10min"		"2h"		"24h"

RP "1day" will be fed by the Signal K to InfluxDB plugin whose sampling period will be defined as 1 second. RP "1week" is produced by the Continuous Query "CQ1" periodically running on RP "1day" (first level down sampling with the clause "GROUP BY time(10s)"). RP "1month" is produced by the Continuous Query "CQ2" periodically running on RP "1week" (second level down sampling with the clause "GROUP BY time(2m)").

The sampling periods chosen here are less or equal to the `avgInterval` of the charts which have respectively a 10 min `timeWindow` ("10min" charts), a 2 hour `timeWindow` ("2h" charts) or a 24 hour `timeWindow` ("24h" charts). When fetching data from InfluxDB, Signalk-stripcharts will target preferably an RP according to `avgInterval`:

- RP "1day" for the "10min" charts,
- RP "1week" for the "2h" charts,
- RP "1month" for the "24h" charts.

Note: the retention policies numbers, names and lengths chosen here are arbitrary; for example, you could choose 4 RPs with names "A", "B", "C" and "D" of durations 1 day, 1 week, 2 months and 1 year if this meets your taste and requirements better.

Here is how to create the RPs using the influx command-line interface:

```
CREATE RETENTION POLICY "1day" ON "boatdata" DURATION 1d REPLICATION 1 DEFAULT
CREATE RETENTION POLICY "1week" ON "boatdata" DURATION 1w REPLICATION 1
CREATE RETENTION POLICY "1month" ON "boatdata" DURATION 31d REPLICATION 1
```

Check RPs created:

```
SHOW RETENTION POLICIES
```

"1day" is defined as the "default" RP, meaning it will receive the data from SignalK InfluxDB plugin.

Note: "1day" RP replaces the default "autogen" RP as default RP. You should now drop the "autogen" RP as follows (its data if any will be lost!):

```
DROP RETENTION POLICY autogen ON boatdata
```

Now CQ1 and CQ2 can be created:

```
CREATE CONTINUOUS QUERY CQ1 ON boatdata BEGIN SELECT mean(value), max(value),
min(value) INTO boatdata."1week".:MEASUREMENT FROM boatdata."1day"./.*/ GROUP BY
time(10s), * END
CREATE CONTINUOUS QUERY CQ2 ON boatdata BEGIN SELECT mean(value), max(value),
min(value) INTO boatdata."1month".:MEASUREMENT FROM boatdata."1week"./.*/ GROUP BY
time(2m), * END
```

Check them:

```
SHOW CONTINUOUS QUERIES
```

The sampling periods of the resulting RPs will be respectively 10 seconds and 2 minutes.

By default, the default RP and all RPs fed by an existing CQ are considered when charting past data. The RP most suitable for the requested past time window will be automatically selected. If the data available in InfluxDB is from an RP with a sampling period longer than the chart `avgInterval`, the plot lines will be dotted (sparse).

You may choose a very different configuration in order to meet your own needs. A good coherency between charts specs and RPs/CQs definitions as explained above is important for obtaining pleasant results.

Stripcharts options for InfluxDB

The former influxDB option `retentionPolicies: [...]` of version 0.1.1 is deprecated and ignored as the Retention Policies and Constant Queries details are now obtained automatically from InfluxDB.

By default, Signalk-stripcharts uses the default RP and all RPs fed by existing CQs for charting past data. However you may have created some CQs and RPs for other purposes than charting with Signalk-stripcharts; therefore those RPs should not be used by Signalk-stripcharts. In such a case you must provide in stripcharts_options.js the short list of RPs to be exclusively considered as chart data sources, e.g.:

```
var options = {
  ...
useInfluxDB: true,                // default is false
influxDB: {                       // optional
  dbName: "boatdata",             // optional, default "boatdata"
```

```
    SK2InfluxResolution_ms: 1000,   // optional, default 1000
    shortlistedRPs: ["1day", "1week", "1month"]  // optional,  <<<<<<<<<<<<<
    // default: default RP and all RPs fed by CQs
  },
    ...
  }
```

Note: The default RP is always considered when querying InfluxDB even if not included in `shortlistedRPs`. Non-default RPs included in `shortlistedRPs` are ignored if they are not fed by an existing CQ.

↥ back to TOC

Another way to chart InfluxDB data - GRAPHANA

Instead of using SignalK Stripcharts, you might consider using Graphana as explained here: https://grafana.com/docs/grafana/latest/datasources/influxdb/

↥ back to TOC

Browser compatibility

Signalk-stripcharts uses a subset of ECMAScript 2015 (ES6).

Version 0.0.8 was mostly developed on Raspbian Chromium 72.0.3626.121 and Windows 10 Chrome 80.0.3987.132.

It seems to work fine also 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, preferably use the landscape orientation and select `none` for one of the charts).

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

Version 0.1.0 was mostly developed on Raspbian Chromium 72.0.3626.121 and Windows 10 Chrome version 83.0.4103.97 (64-bit).

It seems to work fine also on:

- Win 10 with Edge 83.0.478.45 (64-bit),
- Win 10 with Firefox 68.9.0-esr (64-bit),
- Raspbian 9.11 with Chromium Version 72.0.3626.121,
- Raspbian 9.11 with Firefox-esr 52.9.0 (32-bit).

Version 0.1.x was mostly developed on Raspbian Chromium 72.0.3626.121 and Windows 10 Chrome version 83.0.4103.97 (64-bit).

Version 0.2.x was mostly developed on Raspbian Chromium 72.0.3626.121 and Windows 10 Chrome version 87.0.4280.88 (64-bit).

↥ back to TOC

CPU and memory requirements on the client side

On a Raspberry Pi 3B+ with signalk-stripcharts executing on a Chromium browser: 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 numbers of charts and paths, the memory footprint is between 60 MB and 80 MB when the InfluxDB option is not used, and between 80MB and 120 MB when it is.

(measurements above are from Chromium browser task manager)

↥ back to TOC

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 browser console as needed.

Some tracing options are provided in stripcharts_options.js. Tracing occurs below the displayed charts. Alternatively, tracing can be activated in setting the tracing option in the charts specs file as e.g. `options.logTypes: "sbi";`.

↥ back to TOC

Limitations

- Signalk-stripcharts will only chart paths which return a scalar value, not those returning an object value with multiple properties such as environment.current, navigation.position, navigation.attitude;
- Signalk-stripcharts is meant to run on the local network; so far it supports only http and ws (not https, wss)

To-do functional

- [ ] Provide a more friendly way to create/edit/manage custom specs files and preserve them when installing new versions of the package
- [ ] Units, add some missing units and conversions: Volt, Ampere, Watt, Joule, Coulomb, Ratio(0-n), Ratio(0%-100%), Cubic_meter, Cubic_meter_per_second, ...
- [ ] enumActivePaths.html: list time stamps
- [x] InfluxDB retrieval: automatically select the RP (if any) which contains some data for the past period selected and with the smallest sampling period even when the latter is larger than avgInterval (in such case data may be plotted as dots as the query results will contain some null values)

To-do technical

- [ ] Refactor code using ES6 modules and bundle with webpack
- [ ] Retrieval of boat data is inspired from a simple Signal K websocket example. Possibly it could be replaced by signalk-js-client
- [ ] Test on other client platforms (IOS, Androïd)

Any hints are most welcome.

↥ back to TOC

Dependencies included and licences

- c3, licence
- d3, licence
- influx client, licence
- simplepicker, licence

↥ back to TOC

Credits

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

- NavMonPC stripcharts
- Signal K
- c3.js

Thank you to all who have shown the way!

↥ back to TOC