Sparvio Toolbox

Scripts to manage Sparvio sensor systems

Sparvio Toolbox is a set of text-based scripts to access Sparvio systems via the SA1 USB adapter or the SKH1 USB port. Eventually the scripts will evolve into a graphical application.

You need to use the Windows Command line (cmd.exe), Cygwin or some Linux shell to use the scripts. All commands assume the user is running a command prompt in the directory unpacked from the Toolbox zip archive.

Table of Contents

Installation

Depending on your operating system, see install_linux or install_windows.

Python Serial v 3.4

Install with

```
python -m pip install pyserial
```

or through https://pypi.org/project/pyserial/#files

ptpython *(optional)*

Ptpython is only used by `interactive.py` to get a more convenient interface. It is optional. To enable ptpython in `interactive.py`, install the prerequisites:

```
python -m pip install wheel prompt_toolkit requests
```

And install Jedi 0.11 or later

```
python -m pip install jedi --upgrade
```

Reading log files

There are two ways to connect SKH1 to a computer. Either via SA1, using

```
read_log.bat --port <port>
```

or via the SKH1 USB port, using

```
read_log.bat --port <port>:115200bps
```

To use SA1, connect SKH1 to SA1 using the bottom SSP connector. Connect the SA1 micro-USB port to the PC.

Find the COM port used by SA1 by running

```
read_log.bat --ports
```

Create a file named `default_port.txt` only containing the name of the COM port. This identifies what peripheral is SA1 and only needs to be done once.

On Windows, run `read_log.bat` from the command line.

```
read_log.bat
```

On Linux, run `read_log.py` from the command line. For example

```
./read_log.py
```

Usage

```
read_log.py [-h] [--optimize] [--retain] [--noread] [--format]
                 [--verbose] [--ports] [--dir DIR]
                 [--port PORT]

optional arguments:
-h, --help   show this help message and exit
--port PORT  the serial port where SA1 is connected (ex: COM3 in Windows or
             /dev/ttyACM0 in Linux)
--optimize   Faster download, but doesnt work for everyone
--retain     Avoids clearing the log after reading it
--noread     Avoids reading any log files. Just clears or formats the log.
--format     Not only clears the log, but formats the whole storage (after
             any other specified action)
--verbose    Prints traffic
--ports      Lists all reasonable values for 'port' and exits
--dir DIR    Choose a directory to put log files in
```

Real-time graphs

Visualization is done by a program reading data from RR1/RR2 and inputting it to a database. The database is used by Grafana, that creates a web site on the local machine. By visiting the local web page with a browser, the user can view and customize real-time plots of the data.

Four different applications are thus needed:

- Sparvio `telemetry.py` or `interactive.py`
- InfluxDB database service
- Grafana service
- Web browser

For installing and starting InfluxDB and Grafana, see install_windows or install_linux.

Running Grafana and InfluxDB in Windows

Run `influxd.exe`. If Windows warns about "unrecognized application", choose "More info" and "Run anyway". The application will print a page of text, then be silent. Minimize the text window.

If Windows pops up a window asking for permission to run the application, grant it.

InfluxDB needs to run the whole time that data is gathered and visualized. To exit, press Ctrl-C.

Start `grafana-server-exe` in a separate window.

Running Grafana and InfluxDB in Linux

Run these in any terminal:

```
sudo service influxdb start
sudo service grafana-server start
```

Configure Grafana

After installing and starting Grafana and InfluxDB, run `interactive.py` once to set up the InfluxDB database. Without that, the next step can trigger an error "database not found: live".

Point a web browser to `localhost:3000` and use `admin / admin` to login to Grafana. (No need to change the password when queried about it). The browser can be closed and restarted without loss of data.

A one-time setup of Grafana is needed. Click "Add your first data source" and enter this data:

```
Choose "InfluxDB"

Name: Sparvio
URL: http://localhost:8086 (already suggested)
Access: Server
Auth: No checkboxes
Database: live
User: admin
Password: admin
Min time interval: 0.5s

Click "Save and test"

A green box "Data source is working" should be printed.

go to Dashboard
```

If plotting over geographical area is desired, install the Plotly plugin:

https://grafana.com/plugins/natel-plotly-panel/installation

Grafana can show a variety of different views, called *dashboards*. Sparvio provides some ready-made dashboards, added by the "import dashboard" button (In Grafana, choose "dashboards" in the left menu, then "manage" and the button "import".) The dashboards are available as `.json` files in the `grafana/` subdirectory under `sparvio_toolbox`.

Grafana customization

Here is a short introduction to creating and customizing dashboards. For more information, see the Grafana documentation.

A dashboard can contain a number of *rows*, where each row can contain one or more *panel*, especially *graphs*. Each graph shows a set of measurements.

Other useful panel types is *singlestat* to show the latest value of a chosen type, and *Plotly* x/y plots to visualize how measurements spread over a geographical area using GPS coordinates.

Sensor data is configured in the Metrics tab of graphs. Use data source "Sparvio" and select the parameter in From. Use "Group by: tag(sensor)" if multiple sensors report the same parameter -- for example, temperature is reported by both SKH1, SKS1 and SKS2.

If dashboards are changed, they need to be saved with the save button at the top of the page. They can also be shared with others by Share dashboard (icon in upper right corner), Export, then Save to file.

Usage

Connect RR1 or RR2 to a computer. Make sure InfluxDB and Grafana are running. Run `telemetry.py` with `--port` to specify the serial port of RR1/RR2.

Example under Windows:

```
telemetry.bat --port COM1
```

Example under Linux:

```
./telemetry.py --port /dev/ttyACM0
```

Pressing Enter will exit.

Map data visualization

Data can be visualized on a 3D map, on which datapoints are plotted in their corresponding position, with color corresponding to the measurement value. The map also shows a UAV model on the last received datapoint. The map is displayed in a local web page.

Two different applications are needed:

- Sparvio `interactive.py` or `telemetry.py`
- Web browser

Viewing the map

When the `interactive.py` or `telemetry.py` application is running, the map may be viewed by visiting http://localhost:8080/3d.html in the web browser.

Displaying different measurement types

The web page immediately loads all data from the session so far and is updated every two seconds when new data arrives. Datapoints of five different measurement types can be displayed. These are:

- Temperature
- Relative humidity
- CO2
- CH4
- CO

The type of measurement to display can be selected using the buttons at the top of the web page. When a measurement type is selected, the corresponding button is marked with a green background. As data arrives, a counter on each button will indicate the number of measurements for that particular measurement type.

Viewing data in different ways

The user can view the data in several different ways. The *Jump to* buttons will focus the camera on a specific point. In this mode, the camera may be zoomed in and out and rotated around that point. Three different choices are available:

- *Start*: Camera focuses on the first datapoint received.
- *Data from above*: Camera focuses on the center of the data bounding sphere, seen from directly above.
- *Data from south*: Camera focuses on the center of the data bounding sphere, seen from south towards north.

This way of viewing the data can be described as locking the data on a specific point. The *Camera lock* radio boxes control if and how the camera should be locked. There are four different choices available:

- *None*: The camera is not locked and may be moved freely across the landscape.
- *Point*: The camera is locked on a specific point as described above. This radio box cannot be selected by the user, but will be automatically selected when the user clicks a *Jump to* button.
- *Follow UAV*: The camera will jump to the UAV every time a new datapoint is received.
- *All data*: Every 5 seconds, the camera is moved so that all data is seen in the window. The previous camera angle is retained, but the zoom level will be reset.

Color thresholds

When a particular measurement type is selected, the minimum and maximum values used for the color coding are displayed. Measurements below the minimum value or above the maximum value will be displayed with the color of the minimum and maximum values, respectively. The background color of the input boxes indicate the color coding for the corresponding values.

The minimum and maximum values can be changed through the input boxes. New values take effect when the user presses enter or leaves the input box.

The color thresholds of the temperature measurement type are treated somewhat differently than other measurement types. For temperature, the color gradient will go from blue at the minimum value to red at the maximum value, passing through white at the midpoint between the minimum and maximum values.

Altitude offset

The terrain height in the map data may not always match the actual terrain height at the location. For this reason, the user can adjust the displayed altitude of all datapoints by changing the *Altitude offset* input. A positive value causes the datapoints to be raised to a higher altitude. This does not change the underlying datapoints in any way and will not be present in log files.

An automatic correction occurs when the first datapoint is received, by assuming the first datapoint is at ground level. The user may override the automatic offset by entering a different value in the input box.

Data radius

Datapoints on the map are indicated by a colored cube centered on the geographical location of the measurement. The cube is a visual approximation of the extent in space where each measurement is considered valid.

The user may change the cube size by selecting among the radio boxes under *Data radius*. The value sets the length of each side of the cubes. The cubes are drawn with each side as double the selected radius. Spheres could give a truer view of the measurement extents, but cubes are used to improve the graphical performance.

Specifications and performance

The map visualization has been tested using an Acer Swift 5 running Windows with 16 GB RAM and Intel Core i7 1165G7 processor. It has been developed for Google Chrome Version 90.

On this specific system, a stress test has been performed, in which approximately 8400 measurements for each of three different measurement types were sent to the visualization with a speed of 3 data messages per second. This corresponds to a measuring session lasting 45 minutes. The visualization was able to cope well, with only some delay observed if the user changed data box size during the time that data was being transmitted. When all data had been transmitted, this delay was much decreased.

When the speed of incoming data is increased, when the number of datapoints are increased, or when using the visualization application on a different computer platform, the drawing of new datapoints may

cease after some time. In this case, the remaining datapoints are observed to be rendered on the map when data transmission pauses or finishes.

Future development

Here are some ideas for further features/improvements:

- Displaying wind speed and direction: These measurements may be displayed as cone-shaped datapoints with direction and length corresponding to the values. At the moment, we are experiencing performance issues when using cones instead of boxes, and so more time is needed to ensure a good user experience.
- Allowing user to decide color gradient: At the moment, the color gradient for each measurement type is hard-coded. Future work may allow the user to decide the gradient. In this way, the user can select colors that display the datapoints with the most contrast towards the terrain.
- Cleaner layout

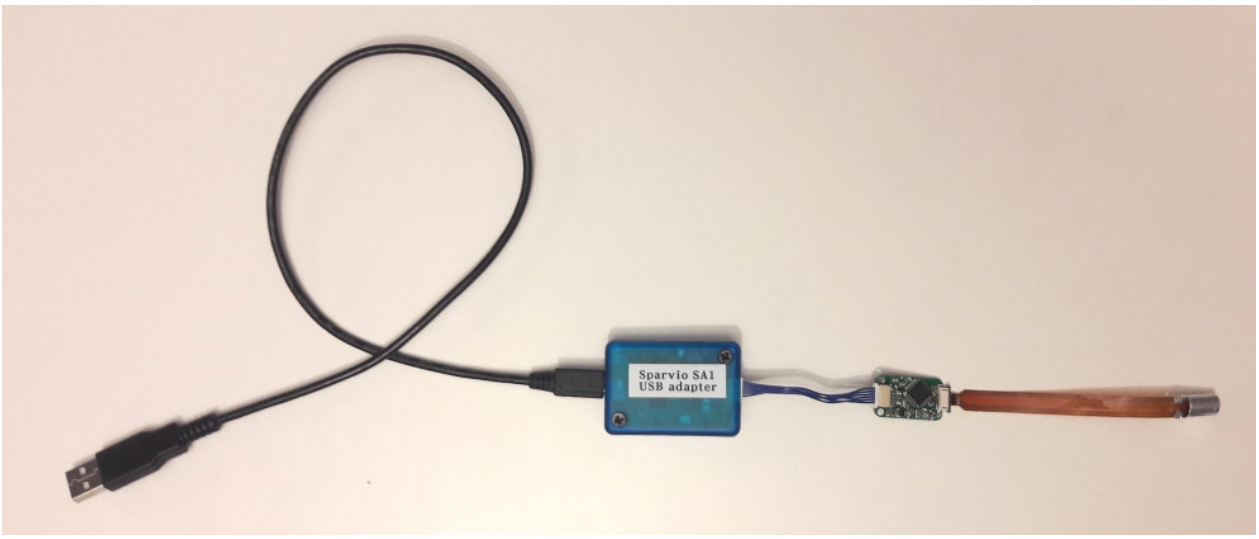
Third-party device access

The script `bridge.py` gives direct terminal access to third-party devices, for example to perform calibration of the sonic anemometer Anemoment Trisonica Mini.

Firmware upgrade

The behavior of each Sparvio module is determined by its *firmware*; a software specifically written for that module type. From time to time, the firmware of the modules may need to be updated. This is only necessary for new features or if the system doesn't work as intended. Upgrading is done with the script `upgrade.py`.

Connect SA1 to a computer and directly to the device to upgrade:



To upload the latest firmware from internet:

```
./upgrade.py
```

This will download the latest firmware for the connected module from internet, and write the firmware to the module. Both the SA1 and the sensor will show a purple color while doing this. A number of lines will be printed, ending with ”Success upgrading firmware!”.

Firmware versions before December 2018 might need an additional parameter to identify what firmware is used. Contact Sparv Embedded if upgrade.py returns an error message.

SKH1 can only be upgraded through the bottom connector.

Sparvio configuration

Many modules can be configured, calibrated or provide trouble-shooting.

Getting and setting individual variables is done with `get.py` and `set.py`. `get.py` can also show the stream of data of a selected variable.

Function calls are done with `call.py`.

For interactive "REPL"[1] access, use the script `interactive.py`.

All components of the system can be accessed as Python objects in an interactive Python session, entering commands at the >>> Python prompt. This exposes the full SSP object model used in Sparvio. Any data from the system can be inspected, variables can be read and changed and all functions can be called.

Some useful SKH1 variables:

Variable	Type	Description
externalGps	true/false	Must restart SKH1 for changes to take effect.
baud	integer	Must restart SKH1 for changes to take effect.
samplingInterval	integer	Interval in millisec between log entries
enableSht2x	true/false	If built-in T and RH should be logged

For example `s.SKH1.externalGps=1` enables using a GPS connected to the UART pads on SKH1.

Serial line access

SA1 can be accessed as a virtual COM port from any application or script, to read the data stream or do configuration. The port settings are 115200 bps, 8N1, no flow control.

Some general terminal programs under Windows are Realterm and Tera Term. Under Linux, minicom.

Python scripting

This tool isn't released yet.

By doing `import sparvio`, Python scripts can dynamically access all aspects of a Sparvio system as native Python data types, objects and functions.

See `python_manual.rst` for details.

---

1 REPL means read-eval-print loop and is a way to interact with a programming language in real-time.