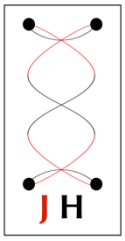

JAGA software Python open source

v.4.1c

1. Install Anaconda Python:

Install the Anaconda Python distribution, which contains Python and other packages we depend on (matplotlib, numpy).

- Find Anaconda at https://www.anaconda.com/download/
  Use the Python 2.7 distribution
  See our document how to install anaconda

- After installation, find where Anaconda is installed, run

  - For Windows
    `[path to Anaconda]\python jaga_display.py`
  - For Linux
    `[path to Anaconda]/python jaga_display.py`

- If you add the Anaconda directory to your execution path you can just type
  `python jaga_display.py`

2. Python codes for capturing JAGA data without display:

- Go to the directory : `cd ~/Directory_Name/Jaga_Python`
- Type in the directory `>> python capture.py`

3. Python codes for displaying and capturing JAGA data:

- Type
  `python jaga_display.py`
- For more options type
  `python jaga_display.py --help`
- For example, to display and capture data from two JAGA devices:
  `python jaga_display.py --num_devices 2`

4. Output file:

The output file name will be like `yyyy-mm-dd_HH-MM-SS_jaga.dat` and will be saved into a subdirectory "JAGA_data" created under the directory you ran `jaga_display.py`

- The output data from JAGA16 (our open data format) is a stream of binary data that contains Headers (packet info and hardware settings) + neural recording data.

- Note: If your file is really long, you can truncate it by typing
  >> split -b 1396* multiple_integer your_filename your_outputfile (Each of our packet is 1396 bytes, so, you will need to split the file into each with multiple of 1396 bytes.
  e.g. >> split -b 1396000 your_filename smaller_file returns smaller_filea, smaller_fileb, smaller_filec, …, smaller_fileaa with each 1396000 bytes.

- The JAGA data is recorded as digital samples from an ADC (analog to digital converter) with 16-bit resolution, with values ranging from 0-65535. 0 µV input leads to the ADC value of 32768 (half of 65535) ± a small channel-specific offset.

5. Generating timestamps for obtained neural recording data

- `data2csv.py` generates by default timestamps (in ISO time format) for each sample in the data (each channel in one row)

  ```
  >> python data2csv.py yyyy-mm-dd_HH-MM-SS_jaga.dat > output_filename.csv
  ```

- If you want to generate timestamps as Epoch time format more readable in MATLAB (a floating number in seconds since Unix Epoch (1970-01-01 00:00:UTC) to microsecond precision)
  ```
  >> python data2csv.py --matlab yyyy-mm-dd_HH-MM-SS_jaga.dat > output.csv
  ```

- To obtain packet statistics (packet drop rate)
  ```
  >>python file_stats.py yyyy-mm-dd_HH-MM-SS_jaga.dat
  ```

- To display the neural data in timestamps in a matlab file.
  ```
  >>run jaga_csv_read.m and import output.csv into this .m file.
  ```

- To display the neural data with timestamps & plot fft in a matlab file.
  ```
  >>run jaga_csv_read_fft.m and import output.csv into this .m file.
  ```

6. Simple viewing of neural recording data with sample counter (no time stamps)

This is a quick way to view the neural recording without converting the counter to timestamps

- `generate_jaga_data_array.m` generates a MATLAB vector array for all channels without timestamps. This simple approach ignores the data gap, since you did not properly

incorporate timestamps, however you can quickly view what you recorded post-data acquisition.

- If (for example), you want to plot just the i^{th} channel,
  `plot(data_array (:,i))`

7. Trouble Shooting: Are you a Mac user?

Error: Files are not written on disk or display does not occur

- Check your computer is accepting jaganet by typing
  `>> ifconfig`

The output should contain the line: **`inet 192.168.8.100 netmask 0xffffff00 broadcast 192.168.8.255.`** If you cannot find this line, the computer is not listeing to jaganet.

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
     options=10b<RXCSUM,TXCSUM,VLAN_HWTAGGING,AV>
     ether ac:87:a3:1d:50:fc
     nd6 options=201<PERFORMNUD,DAD>
     media: autoselect (none)
     status: inactive
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
     ether 88:63:df:c6:e2:dd
     inet6 fe80::875:2b7a:383c:7ffd%en1 prefixlen 64 secured scopeid 0x5
     inet 192.168.8.100 netmask 0xffffff00 broadcast 192.168.8.255
     nd6 options=201<PERFORMNUD,DAD>
     media: autoselect
     status: active
en2: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
     options=60<TSO4,TSO6>
            ether 0a:00:00:52:16:a0
```

- To Check whether jaga data is arriving.
  `>> sudo tcpdump -i en1 -l -n udp port 8888`

  The output should contain the line like below `192.168.8.100.55000: UDP, length 138`

```
09:43:43.927404 IP 192.168.8.10.8888 > 192.168.8.100.55000: UDP, length 1388
09:43:43.970318 IP 192.168.8.10.8888 > 192.168.8.100.55000: UDP, length 1388
09:43:44.013281 IP 192.168.8.10.8888 > 192.168.8.100.55000: UDP, length 1388
09:43:44.056362 IP 192.168.8.10.8888 > 192.168.8.100.55000: UDP, length 1388
```

- JAGA device uses 8888 as our source port.
- Depending on your computer, it can be en1, or en0. Check the result of ifconfig.
- -l will suppress line buffer and will deliver data as soon as it comes in.
- -n will suppress reverse look -up for DNS

Acknowledgements for code contributors: Jordan Sorokin