Prospects for higher sample rates?

Winslow_Strong

April 2016 edited May 2016 in Hardware

I'm interested in any means to increase the sample rate of OBCI. The application is real-time neurofeedback, so I'm unsure that any solution involving an SD-card, e.g. http://openbci.com/forum/index.php?p=/discussion/546/solution-for-session-freezing-overruns-2khz-sample-rate-with-sd-only will be useful. Are there any hacks now, or any prospects in the future, for higher sample rates for OBCI? Boosting the 32-bit 16-channel board up from 125 Hz would be particularly awesome.

Comments

« 1 2 3 »

wjcroft

May 2016

Winslow, hi.

There is currently a speed bottleneck in the serial link between the chipKIT and the mainboard RFduino. As well as another speed limit in the RFduino's radio link. There are a number of older threads which discuss the details and possible workarounds, such as: Bluetooth 2 or 3 module, wifi, compression, etc.

Categories

Recent Discussions

Activity

Categories

All Categories 737

Hardware 231

Ganglion 42

Software 242

OpenBCI_GUI 88

http://openbci.com/forum/index.php?p=/discussion/342/adding-wifi-faster-bt-and-other-connections-for-v4

http://openbci.com/forum/index.php?p=/discussion/248/constraints-on-data-rates-thru-dongle

http://openbci.com/forum/index.php?p=/discussion/94/using-dpcm-to-save-bandwith

Possibly one of the 'easier' approaches which has already been done, is to use a tiny external Bluetooth 2 or 3 module, attached to the mainboard. The links below show this being done with an external breadboard, but you could just as easily attach the module to the mainboard with double stick foam tape, then run your jumpers.

Bluetooth v2 or v3 has MUCH higher throughput than the Bluetooth 4 / BLE (or Gazelle on the RFduino) currently used. You would bypass the firmware that is sending to the RFduino, and instead pass the data to the BT 2 module over the SPI pins. This approach would also eliminate the dongle, and you could go direct to devices which support standard Bluetooth. The Bluetooth 4 / Bluetooth Low Energy (BLE) stack was intended for lower speed / low power applications, hence is not as adaptable for higher data rates.

http://openbci.com/forum/index.php?p=/discussion/434/openbci-to-android-app-using-external-bluetooth-module-on-breadboard
http://openbci.com/community/openbci-with-android-as-graduation-project/

William

Winslow_Strong
May 2016

Thanks, William. That is extremely helpful.

wjcroft

May 2016

For others following this thread, Winslow and I have been in touch via email with Momen (author of the OpenBCI Community / Android post mentioned above.) And he has provided zip files with his source code.

One disadvantage of the Bluetooth 2 or 3 add on modules, is that most of them use a serial port RX TX pins for connection to the uC. And not SPI. This can impose its own bottleneck on serial port data rates.

Regarding tiny uC breakout modules that allow high speed radio connection, just ran across this new wifi module yesterday, based on the ATWINC1500 module from Atmel,

https://learn.adafruit.com/adafruit-atwinc1500-wifi-module-breakout/overview

This uses SPI so should run blazing fast (uC to ATWINC) and with no radio link bottlenecks.

Only downside is that Adafruit is currently out of stock on these little boards. Must be popular. I'm guessing you could find similar add on tiny board modules containing the same wifi chip from other manufacturers. Or ask Adafruit their estimated time for having more stock.

Actually the only thing the Adafruit board does is 5v to 3.3 v level shifting. And since the 32 bit board is already 3.3 v, you should be able to interface directly to a ATWINC1500, if you can find another PC breakout for the chip.

https://www.google.com/search?q=ATWINC1500+breakout

ATMEL has their own breakout, if you need another supplier,

http://www.atmel.com/Images/Atmel-42388-ATWINC1500-Xplained-Pro_UserGuide.pdf

William

Winslow_Strong

May 2016

I'm wondering why no one seems to be using a wired connection between the OBCI board and a computer? Is there a simple way to connect them via USB to get higher sample rates?

biomurph Brooklyn, NY

May 2016

We are not recommending that the board be wired to the computer because of a number of reasons.
First, safety. We are using a wireless connection to avoid the complications and regulations around the complexity of designing galvanic separation into the board and getting it passed. We also have a very clean power supply with the battery. USB even when your computer is not connected to it's power supply offers a very noisy power supply. For these reasons a wired connection is not ideal.

Winslow_Strong

May 2016

As of right now, Adafruit is still out of stock of their ATWINC1500 WiFi Breakout, but I picked up one from Amazon here: http://www.amazon.com/Adafruit-2999-ATWINC1500-WiFi-Breakout/dp/B01BZRWWNY/

I noticed that awhile back biomurph mentioned he was waiting on this "photon" wifi chip (which now is available): https://store.particle.io/?product=spark-photon&redirected=true

Any reason to prefer it over the ATWINC1500?

wjcroft

May 2016

Winslow, hi.

Joel @biomurph may comment here as well. My impression is that the Photon is a much more general purpose IoT microcontroller; many more IO pins, complete programability. Intended to operate as the central CPU in an IoT device. So the SPI pins there may be setup as a master vs slave operation. The ATWINC1500 is setup as an SPI slave, to be driven from the OpenBCI chipKIT.

I also read that the current consumption on the Photon is higher than the ATWINC, something like 60ma vs. 12ma. And the Photon is expecting 5V input power, whereas the ATWINC is fine with the 3.3v of the OpenBCI.

I did some searches on Google, trying to find more recent reviews comparing Photon and ATWINC directly, but came up blank. I think this is because of the different audiences intended for the products (wifi peripheral vs. general IoT controller).

William

wjcroft

May 2016 edited May 2016

Here's the API call to set the ATWINC1500 RF output power level,

http://atmel.force.com/support/articles/en_US/FAQ/What-is-the-supported-Tx-output-power-for-

ATWINC1500

https://en.wikipedia.org/wiki/DBm

The TX_PWR_LOW -> 6 dBm (4 mW) should be plenty. And is close to the 4 dBm (2.5 mW) used by Bluetooth class 2 (10 meter range).

Winslow_Strong

May 2016

Thanks @wjcroft. My board should arrive today. I'd be up for writing a walk-thru of how to get this working, once I figure it out myself. From a high level, Im still not totally clear on what the steps are (my electronics chops aren't that strong).

Conerns:

- Connect via SPI pins. Aren't these already used by the daisy module in 16 channels OBCI? How would I extend them physically out of the daisy module to be accessible by the breakout board also? I only really care about boosting the 16-channel's 125 Hz rate to >=500 Hz.
- From the adafruit ATWINC1500 RF breakout page: "Right now the Atmel-supplied library works great with Arduino Uno or Zero, but may not work on other Arduinos." I got the impression from reading other posts that Arduino Uno is the language for the V3-32-bit boards running in 8-bit mode, but that to run in 32-bit mode is different. Is it compatible?

Steps:

1. Connect Wifi breakout board to SPI pins
2. Make API call to set ATWINC1500's power level as per William's instructions above.
3. Instruct OBCI to send data out via ATWINC1500. How?
4. Test above with OBCI Processing.

5. Since I ultimately want LSL out, I need to mod the OpenBCI_Python code https://github.com/OpenBCI/OpenBCI_Python to find OBCI via wifi instead of via serial port (COM).

Are any of these off-base? Am I missing any high-level steps?

wjcroft

May 2016 edited May 2016

Winslow, hi.

First let me congratulate you for the courage in jumping into this. :-)

I would take as an example the previously mentioned code from Momen. Even though that is using serial port output, the idea is similar to what you'll be doing, as far as pumping out the data on another port. In Momen's case he had BOTH data streams going (one through dongle and one through Bluetooth). Since you want to avoid speed bottlenecks, you might end up just going strictly through the wifi.

So as far as your step #3 above, use Momen's code as an example for diverting the data stream to another device, and add the necessary setup calls to initiate the ATWINC and it's wifi connection.

Wiring: The SPI is a 'bus', meaning multiple devices can share it, including the Daisy and the ATWINC. The shared pins are the MISO, MOSI, and SCK. You'll also take DVDD (+3.3v) and GND from the same daisy header pins. There is also a RST pin, I think your ATWINC supports that as well.

https://www.arduino.cc/en/reference/SPI

You will use one of the unused digital output pins on the main board (header pins) as your "chip select" for the ATWINC. That's up to you which one you use.

As far as the Adafruit library, you'll have the source, so should be able to tweek it if needed. I would take a look at the Adafruit ATWINC example programs. They have several there. With the wifi link up, you'll have the ability to initiate a tcp/ip connection to your laptop. The data will be sent over that.

It's possible by doing some searches on Google, that you could locate a prototype tcp server code that would run on the laptop; that would take an incoming socket connection, and then connect that (route the data streams) into what is called a pseudo-tty, an emulated COM port. On Unix / Linux this doesnt take much code at all. Not sure what is out there for Windows.

OpenBCI_GUI and the existing LSL stuff expects such a serial port. If you can emulate the COM port over tcp, then you could use those without modifications.

https://www.google.com/search?q=emulate+com+port+over+ethernet

https://en.wikipedia.org/wiki/COM_port_redirector

William

Winslow_Strong

May 2016

@wjcroft, thanks so much for sketching out the details to the walkthru. Before taking credit for being courageous, I have to say this sounds rather beyond my skill level, unfortunately.

Which makes me wonder if BT might be easier? Following the path that Momen blazed could be more straightforward. First thing to note is that BLE is totally out of the question. E.g. the adafruit chip (https://www.adafruit.com/product/2633) maxes out at < 3kb / s

https://forums.adafruit.com/viewtopic.php?f=31&t=83783&start=30#p460592

whereas we would need ~ 400 kb / s to achieve 16 ch @ 1k Hz.

The LMX9838 that you mentioned earlier nominally can reach that w/radio link of 700 kbps. I don't easily find a breakout of that though, and don't know enough about hardware to judge which boards containing it would be compatible with OBCI. Adafruit doesn't have it. Amazon has a few, this being the most promising looking:
http://www.amazon.com/INSTRUMENTS-LMX9838SB-BLUETOOTH-MODULE-704KBPS/dp/B00HRNL9A8
Wouldn't arrive for ~20 days though.

What about the HC-06? e.g. http://www.amazon.com/JBtek-Bluetooth-Converter-Serial-Communication/dp/B00L08GA4Q/
 A comment: http://www.amazon.com/review/R1OZPW6GJ2LCKI/ref=cm_cr_dp_title?ie=UTF8&ASIN=B00L08GA4Q&channel=detail-glance&nodeID=541966&store=pc claims it can run at 460k baud, which would nominally get the job done. However it uses Rx-Tx. What data rate does that bottleneck impose?

I'm grateful for your help, as always.
Winslow

wjcroft

May 2016 edited May 2016

I forgot to mention your physical connection to the SPI header pins.

Currently the Daisy has pins extending from the bottom that plug into the mainboard. So there are a couple ways you can do this.

(1) The 'hack' way. Just solder tack some wires onto the pins you need on top of the Daisy board. This is a bit ugly, but would work. A downside is that the wires would not be very strain

reliefed, so flexing many times could break off.

(2) use one or the other type of 'stacking' header pin. The female type is shown here,

https://www.adafruit.com/products/85

You would just use one of the 8 pin headers, this kit has 2 of those. Removing the existing header would require some unsoldering skills. A "solder sucker" tool helps here in removing most of the solder, then eack individual pin can be eased out. Once the holes are clean, the new header can be soldered in.

Alternately "stacking" male headers are also available, these have long pins on each side.

https://www.adafruit.com/products/400

For connecting to the stacked header, you can use jumper wire plugs,

https://www.adafruit.com/products/1954

One side at the ATWINC would be soldered, and the other plugged into the Daisy header. Same idea for your chip select line to the mainboard.

As far as attaching the ATWINC breakout board physically. I'd use a small piece of double stick foam tape. Cut to size, then stick it a few times to your hand / skin to reduce the aggressiveness of the adhesive. This is so you can pull it off in the future from the mainboard when you need to.

You could stick it on the front surface of the mainboard, on the opposite side from where the RFduino is located. This is a bit away from the ADS1299 A/D chip and more sensitive analog sections of the board. An alternate location would be on the Daisy itself, say over the OpenBCI

logo area, off the right edge away from the ADS1299. Much less surface area here on the Daisy, so mainboard probably best.

wjcroft

May 2016

Hmm, I was writing my last post about headers as you were typing about HC-06. Just saw that now. I dunno, the current bottleneck is a serial port link between the chipKIT and RFduino. Limited on how fast the bit rate can be received by the RFduino. You'd have to check the spec on the HC-06 max RX bit rate.

The wiring of the ATWINC vs Bluetooth is not that much more challenging. SPI is way faster than serial ports, and the data rates over the air also much faster.

wjcroft

May 2016 edited May 2016

Winslow, hi again.

Here's a way you can connect with an [optically]* isolated wired USB serial cable. The isolator has medical grade isolation specs, so safety should be just as good as a wireless connection. But in this case you can up the bit rate.

> **[although opto isolators exist in certain applications, this Analog Devices part uses their iCoupler® technology; which combines high speed CMOS and monolithic air core transformer technology.]*

The current OpenBCI serial bit rate (set in the FTDI serial device COM port driver) is 115200 bps. These are 10 bit bytes, 8 data bits, 1 start and 1 stop bit. The FTDI serial chips also support

230400, 460800, and 921600. I would use the lowest one that can support your data stream.

Isolator board:

https://www.adafruit.com/product/2107

This Adafruit isolator runs at USB rates of 'Low' (1.5 Mbps) or 'Full' (12 Mbps), there is a small switch on top of the board; just set at Full.

My suggestion would be to at first just keep with the battery power supply to the OpenBCI board. That guarantees the cleanest DC power supply. However, this isolator also can supply 100ma of DC power, through a DC/DC Converter. That might be a bit noisier. You could always run that through a linear 5v regulator to cleanup the DC if needed.

I don't have the mainboard power draw numbers right in front of me, but I recall the 8 channel (mainboard) was less than 60ma, that's including the RFduino which you won't be using. I think the Daisy board only pulls something like 15ma. The DC/DC converter spec:

http://www.analog.com/en/products/interface-isolation/isolation/isopower/adum5000.html

The isolator spec:

http://www.analog.com/en/products/interface-isolation/isolation/standard-digital-isolators/adum4160.html
http://www.analog.com/media/en/technical-documentation/data-sheets/ADuM4160.pdf
[IEC 60601 safety specs on page 4]

If you can find a 'thin' type USB cable to go from the isolator to the OpenBCI, that may be an advantage if the mainboard is head mounted. Such as,

http://www.usbfirewire.com/ultra_thin_usb_cable.html

While it is possible to emulate a USB serial port in Arduino software, it's much easier to just use this tiny breakout, again from Adafruit. Mount on the mainboard with some double stick foam tape,

https://www.adafruit.com/product/2264

When powered up this just presents as a serial COM port. You don't need to use all the other fancy features. Tutorial:

https://learn.adafruit.com/adafruit-ft232h-breakout/serial-uart

I don't think you'll need to use the flow control lines, as the chip has built in buffering that should smooth out usb traffic. But it does support CTS/RTS hardware flow control lines if you should need them.

FTDI Chip itself described here,

http://www.ftdichip.com/Products/ICs/FT232H.htm

So basically you'll be using Momen's code, but dropping the Bluetooth setup section. My suggestion is to use the dual position power switch to indicate your mode of operation. In the usual 'PC' position, it operates as normal with the dongle. In the 'BLE' position, you don't use the RFduino, instead all serial IO traffic goes through the hard wired USB link.

This has the advantage now that all your laptop software that expects a COM port, operates normally. Of course you will need to adjust the baud rate of the COM port.

Regards,

William

Winslow_Strong

May 2016

Thanks so much, William. The isolated USB sounds way easier, and we have no particular need for a wireless connection.

Are you recommending the ultrathin USB cables on the basis of reduced weight? Or some other reason?

Metta,
Winslow

wjcroft

May 2016

re: thin usb cable, yes just for reduced weight on your headset. Any cable should work.

yj France , Bordeaux

May 2016 edited May 2016

Hello,

@Winslow_Strong,

Do you realy need 1kHz to performe neurofeedback?
Which kind of EEG "high frequency" feature dou you want to use/study?

Have you thought about data compression?

Most of the time 16bits are enough, then using 16 instead 24 allows you to access to 16 channels - 250Hz
(if the signal drifts you can implement a high pass filter by mean subtraction.)

Moreover a simple "u-Law" compression might allow you to reach 500Hz...

The current bluetooth frame length being 31 octes , a 15 channels / 500Hz seems accessible by software modification.

Regards,
y.j.

Winslow_Strong
May 2016

@yj : interested in gamma around 50 Hz, and possibly higher frequencies. 125 Hz sampling frequency is not really workable for characterizing the amplitude of 1s epochs of 50 Hz content. We are doing exploratory analysis to characterize meditation states, and I would prefer not to rule out potentially relevant signals.

1kHz indeed may be beyond what's necessary, but I figured while we are at it, why not reach that level? I would stop at 500 Hz if it were substantially easier to reach than 1k Hz. 500-1000 Hz are pretty standard recording frequencies in research systems. 256 is more common in clinical systems.

16-bits may be good enough. I have no idea how to implement compression, but have seen it mentioned on the forum a few times. Do you know of a post that outlines a successful implementation? Id be interested to try, if it's not too technically demanding.

wjcroft

May 2016

re: compression. You'd probably want to go with a "loss-less" compression that reproduces an output signal that exactly matches the input signal. Some of the algorithms (u-Law I think) are not loss-less. The DPCM mentioned on the 2nd post from the top can be loss-less. One downside of the current 31 byte radio packets is that they may not be ideal for compression schemes that have variable length coding blocks. For compression you pretty much want a contiguous byte stream that can be parsed arbitrarily.

re: 1K sample rates. Others interested in EMG and realtime feedback EMG have requested faster rates. So it's not just EEG applications that could benefit.

wjcroft

May 2016

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.6378

EEG Data Compression Techniques (1997)

Abstract

In this paper EEG and Holter EEG data compression techniques which allow perfect reconstruction of the recorded waveform from the compressed one are presented and discussed. Data compression permits one to achieve significant reduction in the space required to store

signals and in transmission time. The Huffman coding technique in conjunction with derivative computation reaches high compression ratios (on average 49 % on Holter and 58 % on EEG signals) with low computational complexity. By exploiting this result a simple and fast encoder/decoder scheme capable of real time performance on a PC was implemented. This simple technique is compared with other predictive transformations, vector quantization, discrete cosine transform and repetition count compression methods. Finally, it is shown that the adoption of a collapsed Huffman tree for the encoding/decoding operations allows one to choose the maximum codeword length without significantly affecting the compression ratio. Therefore, low cost commercial microcontrollers and storage devices can be effectively used to store long Holter EEGs in a compressed format. Keywords--- Data Compression, Huffman Code, EEG Signal. I.

For comparison, a high compression, high computation cost EEG compression scheme,

http://www.hindawi.com/journals/ijta/2012/302581/

A High-Performance Lossless Compression Scheme for EEG Signals Using Wavelet Transform and Neural Network Predictors

yj France , Bordeaux
May 2016

@wjcroft,

Thank you very much William for these references to papers (not easy to read, at least for me) and discussion which I discover.

The u-Law is not loss less…

When the signal increases small details disappear.
I wonder how it would affect the gamma band which is the small amplitude desynchronized eeg.
May be u-law would be a good filter... (joke)

@Winslow_Strong,

if the pic32 can extract from a 1kHz flux the features you are looking at for the feedback, this could be the best way to compress datas.
Biomurph is working on an "interrupt Version" which will facilitate this computation...

If you want to try u-law the algorithm is rather simple : **u-Law**

Regards,
y.j.

Winslow_Strong
May 2016

One other issue re:sample rate on the 16-channel system that just came to my attention is that the board is setup to alternate packet transmission between updating channels 1-8 and 9-16.
 That info comes from:

"we are sending data packet at the same rate of 250SPS, and alternating sample packets between the on Board ADS1299 and the on Daisy ADS1299. The method takes an average of the current and most recent channel values before sending to the radio. On odd sample numbers, the Board ADS1299 values are sent, and on even sample numbers, the Daisy ADS1299 samples are sent. When running the system with 16 channels, it is highly recommended that you use an SD card to store the raw (un-averaged) data for post processing."

http://docs.openbci.com/software/02-OpenBCI_Streaming_Data_Format#openbci-v3-data-format-

16-channel-data

Would it be easy to change this asynchronus packet content to a synchronus one when we have higher bandwidth? It would make it less problematic to calculate coherence values.

wjcroft

May 2016

Winslow, yes, that should be straightforward.

Your data stream processing loop will check the status of the PC/BLE power switch. When in PC (default mode), the current RFduino scheme will be used which alternates. When in BLE (usb hardwired) mode, your data stream processing will just mirror what is done in the SDcard portion of the code, with no averaging.

The power switch state can be sampled in your init startup code. Then saved in a variable so you don't need to keep checking that IO pin.

yj France , Bordeaux

May 2016

@Winslow_Strong,

just a note : averages are sent alternatively (or successively , sequentially, serially, [1:8,9:16], like any bits...) but the 16 channels have been sampled synchronously @ 250Hz.

y.j.

Winslow_Strong
May 2016

@yj Yes, so actually the 125 Hz is really downsampled w/linear interpolation from 250 Hz, which should be slightly better than a mere raw 125 Hz sampling. However, as I understand it, that averaging procedure results in the ch1-8 vs 9-16 data that arrives at your computer never having the same time stamp.

E.g. data packet #3 contains channels 1-8 data. It's averaged from t=3 and t=2 (@250 Hz), so its effective time stamp should t=2.5. Data packet #4 contains channels 9-16, and is averaged from t=4 and t=3 (@250 Hz), so its effective time should be t=3.5. And so on...
So you end up with no time points where all data from channels 1-16 is present. This is important to know if one wanted to calculate phase-dependent metrics like coherence between a channel from 1-8 and one from 9-16. A 4ms differences is 1/5 of a wavelength of a 50 Hz wave, which is a really substantial phase shift. It would also really mess up the higher frequency content of a source localization calculation, which assumes all the data points have the same associated time.

It is of course correctable with another interpolation, but its really important to know this about the raw data coming from the daisy module, or one could make big mistakes in naively calculating those types of downstream phase-dependent quantities.

yj France , Bordeaux
May 2016

@Winslow_Strong,

you make me doubt,

I was thinking that :

```
at t1= 0mS   channels 1-16 are  sampled
at t2= 4mS   channels 1-16 are  sampled again   average t1,2  ch 1- 8
are sent in frame 1 (odd ) time stamp =  2mS
at t3= 8mS   channels 1-16 are  sampled again   average t1,2  ch 9-16
are sent in frame 2 (even)  "     "   =  2mS
at t4=12mS   channels 1-16 are  sampled again   average t3,4  ch 1- 8
are sent in frame 3 (odd ) time stamp = 10mS
at t5=16mS   channels 1-16 are  sampled again   average t3,4  ch 9-16
are sent in frame 4 (even)  "     "   = 10mS
...

The critical point is to know if channels 1-16 are sampled by the two
ads1299, at the very same time or not.
Biomurph can probably answer.

Regards,
y.j.
```

wjcroft

May 2016

@yj, hi.

Yes, the mainboard ADS1299 and daisy ADS1299 are synchronized and sampling at the same time. The mainboard sends it's ADS1299 CLK output signal to the input CLK pin of the daisy ADS1299. Then they are both started at the same time using the START SPI command, with both chips being simultaneously selected (through their CS chip select pins.)

http://openbci.com/forum/index.php?p=/discussion/161/daisy-and-main-board-sample-synchronization

Winslow_Strong

May 2016

@yj @wjcroft I had understood that the two ADSs were synchronized and sampling at 250 Hz. What's still unclear to me is in YJ's example, whether the t=3 packet contains the avg of the t=1 and t=2 data points. That runs counter to the literal wording from the docs:

"The method takes an average of the current and most recent channel values before sending to the radio. "

I.e. the current there is t=3, and most recent would be t=2.

So either the doc wording should be changed, or YJ's scheme is not the way the packets are encoded, and we do have an asynchronus time stamping issue of the data at the dongle (rather than on the main boards/daisy boards).

wjcroft

May 2016

Winslow,

I think Conor and Joel @biomurph have been on the road (XTech, Maker Faire, Asilomar), but Joel will probably comment at some point. The actual code for inspection is at,

https://github.com/OpenBCI/OpenBCI_32bit_Library/blob/master/OpenBCI_32_Daisy/OpenBCI_32_Daisy.cpp

Functions updateBoardData and updateDaisyData, lines 611, 658.

yj France , Bordeaux
May 2016

@Winslow_Strong,

Sorry , I am afraid that you are probably right ! ! !

The boolean "firstDataPacket" is true at start : line 175 & 590
and after that always false...

Strange affair...

The code seems to do an average at each sampling step and not one over two

To toogle it I expected to find at the end of OpenBCI_32_Daisy::updateDaisyData() something looking like :
 firstDataPacket = ! firstDataPacket ;
But nothing...
may be I am tired or too old...

I already found this strange , look in my version on GitHub for example line 744
It is funny, I commented lines using this boolean "always false" or suppressed it from tests...
But since I do not use a daisy card , I did not bothered about adverse consequences...

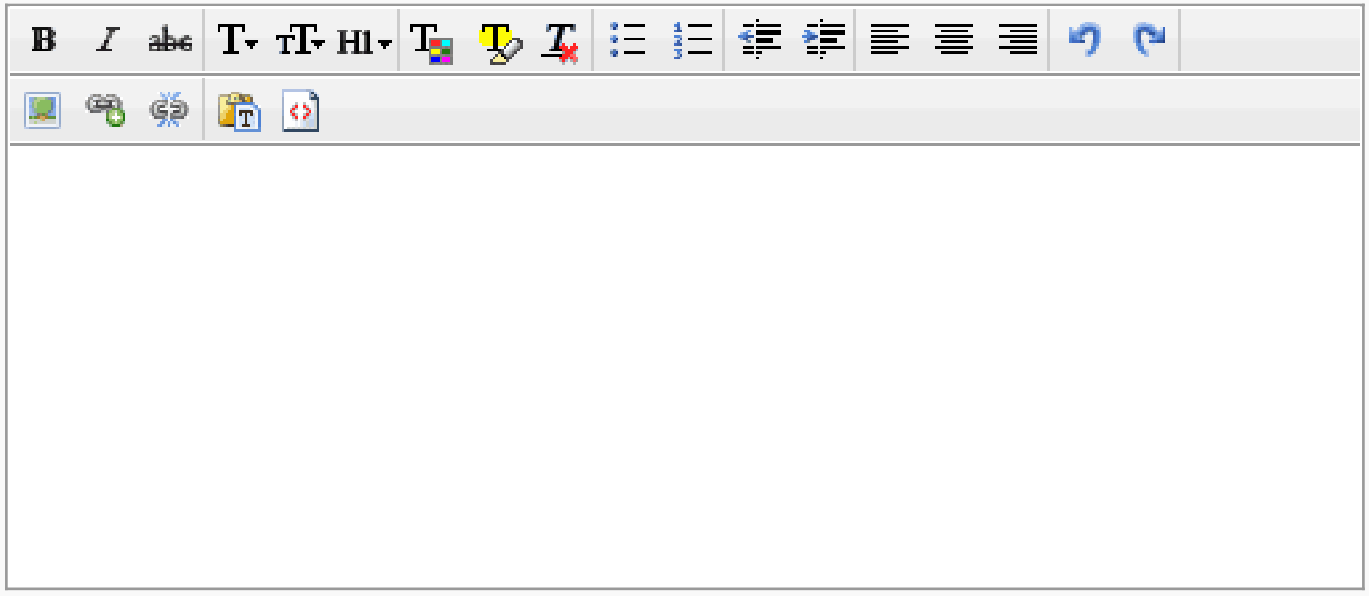
And as William says, let us wait for Joel's and Conor's explanations.

Winslow , run the card with the squarewave test and look if there is a shift of ADSs at the transition or if they are sync...
Use TEST SIGNAL CONTROL COMMANDS : '='

Anyway if it is really buggy, it is easy to correct.

Regards,
y.j.

Leave a Comment



Comment As ...