Table of Contents

Experiment 2: Sampling Consistency and Release-Delay Characterization Report

To obtain the delay offset of the Tethered-Release System under various configurations. The main focus is the timing of the release event. We have the following assumptions of the delay of the tethered-release system.

1. The ground truth of release event is when the tether is released. The ground truth can be detected by a sudden drop in the tether tension - load cell (force sensor) readouts.
2. When using the LiveAmp setuo, the ground truth is acquired by connecting the load cell to the LiveAmp, requiring a wire in addition to the tether connect to the patient.

Based on our setup, there are other approaches of determining the release event besides the force sensor. We design separate sub-experiments to gather and analyze data from them. They are:

1. Experiment 2a: Release trigger signal, connects to trigger box that is not on the patient, does not require an additional wire.
2. Experiment 2b: Accelerometer signal, LiveAmp on-board sensor, does not require an additional wire.

We used the Lab Streaming Layer (LSL) to setup the multi-sensor recording environment. LSL is advertised to have accurate and consistent sampling up to ms level. We want to confirm this by first analyzing the sampling and timestamp consistency of the recordings from LSL and then investigating the optimal hardware and LSL setup for our experiment and further brain and balance research.

The goal of this experiment is to evaluate the distribution (via observing the spread and statistical measures such as mean and STD) of the release delay of the release trigger signal and accelerometer signal approaches. Based on the fact that balance N1, appearing after the release event, is subcortical and cortical -> 50~150ms

after the release -> up to 100 ms wide, we can determine if the sensors can reliably determine the release event.

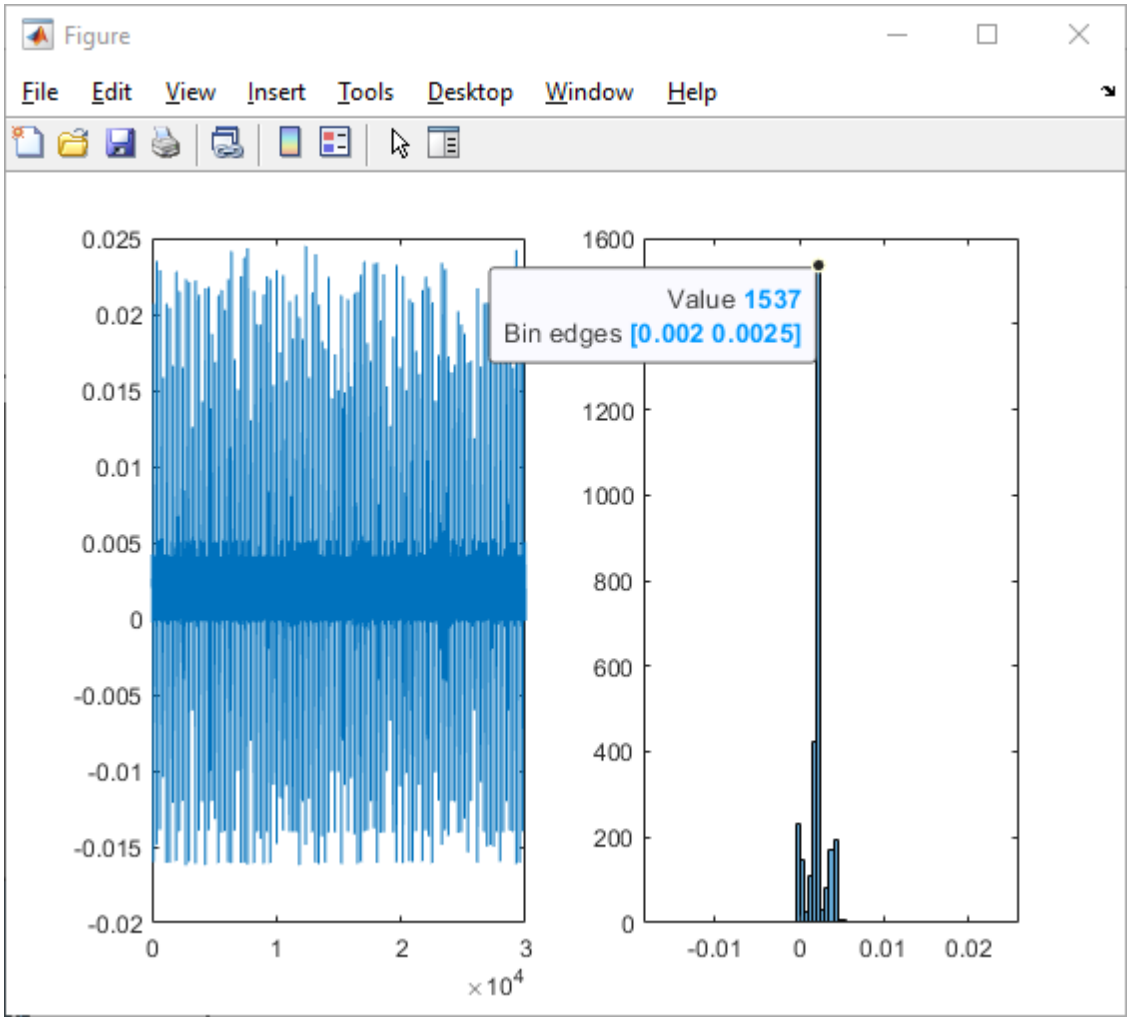
1. Mean: not really any threshold for that as long as the STD is good
2. STD: measures the repeatability or jitter of the latency, should be within 10ms to be research practical.

Note that for the following experiments, the recorded timestamps are in seconds, if no conversion is specified.

Experiment 2o: TimeStamps/Sampling Consistency

First we evaluate the sampling consistency of the LSL system.

Priliminary studies on ActiCHamp data show that the effective sampling intervals (measured by taking the difference between two consecutive timestamps) varies across each recording. There are even a small fraction of the sampling intervals negative, meaning that a later timestamp has a value smaller than the earilier timestamp. As shown in the figure below, the left subplot is the sampling interval for each sampe in the data recording (y axis unit in sec), the right subplot is the distribution of the sampling interval.



There are many possible sources that cause inconsistent sampling intervals. For instance, it could be caused by the LSL library's inherent nature, it could be caused by the Brain Product's connection program, or it could

be caused by my implementation of the LSL-based recorder that retrieves streamed data. Since there are many stages in the signaling pathway, we have to rule them out separately. The stages can be roughly broken down to the sensor stage, the connector stage, and the recorder stage.

1. **The Sensor Stage:** Sensor (microphone) -> Onboard Amplifier -> Windows (COM port, usb bus, driver, etc)
2. **The Connector Stage:** Connector Program (Audio Capture for laptop onbard microphone / ActiCHamp Connector for ActiCHamp / LiveAmp Connector for LiveAmp)
3. **The Recorder Stage:** Recorder Program (Lab Recorder / MATLAB Script using LSL functions / MATLAB GUI using LSL functions)

The LSL library and tools invole in the connector stage and the recorder stage.

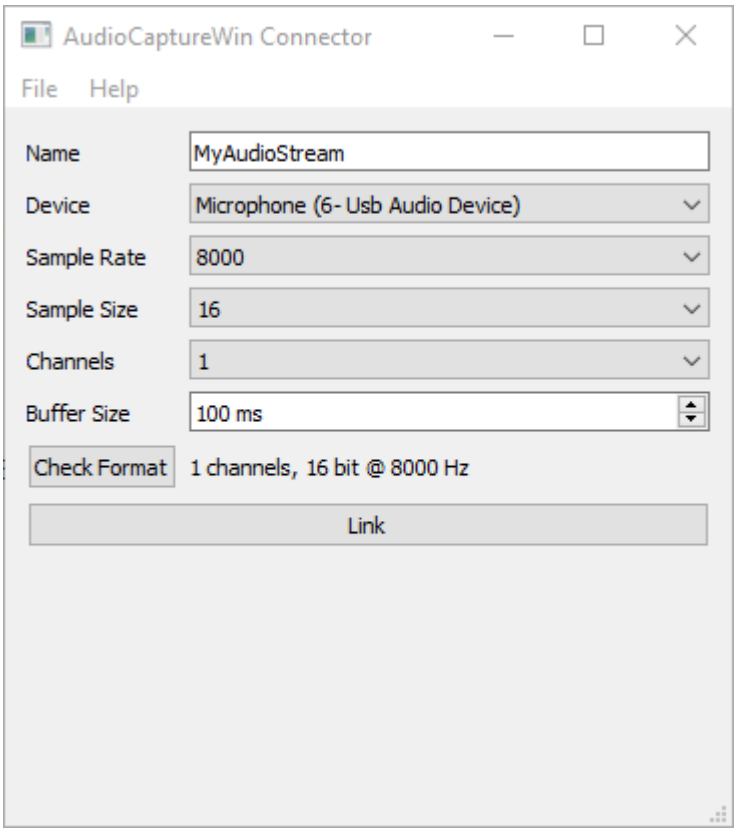
First, we investigate the timestamps consistency of data recordings taken with a different sensor to see if the sensors are the cause of the inconsistent sampling interval. A easy to get sensor is the laptop onboard microphone. By using the Audio Capture app avaliable in the LSL library, we compare the computer onboard microphone and the ActiCHamp setup.

Analysis

First we analyze the audio capture pathway.

Audio Capture

The Audio Capture program is setup with the following configurations. Sample rate is set up 8000 Hz, sample size is set to 16 bit.



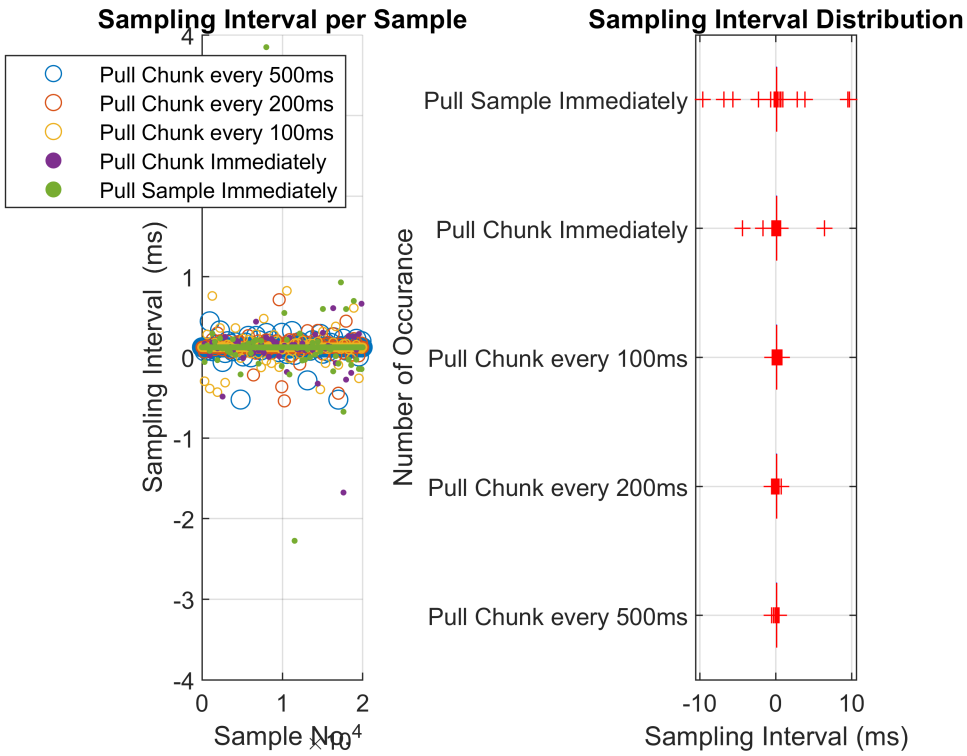
Unfortunely, the Lab Recorder is not compatiable with the sampling frequency of the Audio Capture. So we have to use the LSL function to retrieve data streamed by Audio Capture. The functions are pull_sample() and pull_chunk(). pull_sample() pulls 1 sample for each call and pull_chunk() pulls all new samples since last call for each call. The functions are called in a loop. To minimize the influence of the MATLAB related operations, the other lines in the loop are for storing and displaying the pulled samples. A pause function is called after each pull for testing the frequency of pulling data. My hypothesis is that if data is pulled too fast, it might interfere with the normal functions of the connector app and worsen the timestamp consistency, thus worsening the sampling interval consistency.

```
% loop to pull data
while true
    % get data from the inlet
    [pos,ts] = inlet.pull_chunk();
    % store
    stored = vertcat(stored, [pos', ts']);
    pause(0.5)
    % display
    if ~isempty(ts)
        fprintf('%.2f\t',pos);
        fprintf('%.5f\n',ts);
    end
end
```

Signal Pathway:

Sensor (microphone) -> Onboard Amplifier -> Windows -> Connector Program (Audio Capture) -> Recorder (MATLAB Script)

By setting the pause time to 0, 0.1, 0.2, and 0.5 sec (roughly simulating pulling samples immediately, every 100 ms, every 200 ms, and every 500 ms), we can calculate for the mean, STD, and distribution of the sampling interval from each setup. The plots and table shown below are the results.



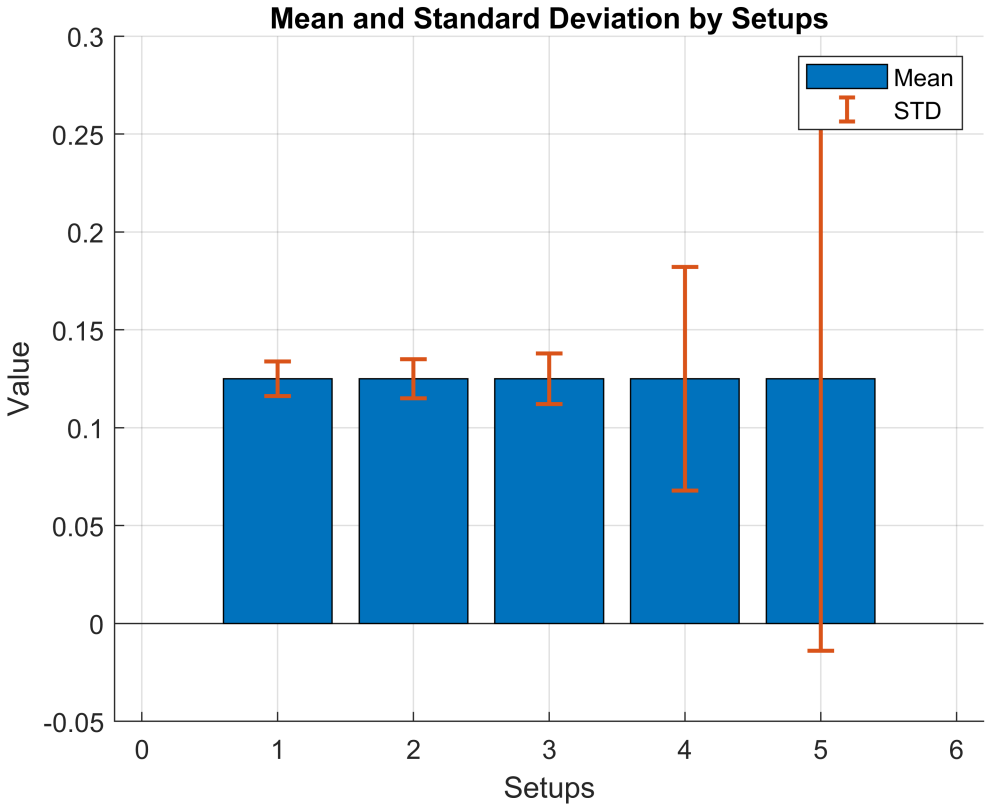
In the figure shown above, the left subplot are the sampling intervals from each experiment config, using either pull_sample() and pull_chunk(), the majority of the sampling interval is located 0.125 ms, corresponds to the 8000 Hz sampling rate. However, if we look at the box plot for each setup, the pull sampling immediately and pull chunk immediately configs have more outliers (marked by the red plus sign) that could be up to 10 ms away from the mean.

Mean and STD of the MATLAB Script at different configurations:

The mean and STD of the different configs are shown below, we can clearly see that pulling immediately resulted in significantly worse standard deviations.

T = 5×3 table

	Setup	Mean (ms)	Standard Deviation (ms)
1	"Pull Chunk every 500ms"	0.1250	0.0088
2	"Pull Chunk every 200ms"	0.1250	0.0100
3	"Pull Chunk every 100ms"	0.1250	0.0129
4	"Pull Chunk Immediately"	0.1250	0.0572
5	"Pull Sample Immediately"	0.1250	0.1390



Setups 1 to 5 corresponds to -> "Pull Chunk every 500ms", "Pull Chunk every 200ms", "Pull Chunk every 100ms", "Pull Chunk Immediately", "Pull Sample Immediately"

Comments: The above results have shown that the sensor stage is probably not the source of the inconsistent sampling intervals. That's because the same problem persist for different sensors. The use of the LSL function does influence the sampling intervals. Next, we switch back to the ActiCHamp system to test for the recorder stage in this pathway.

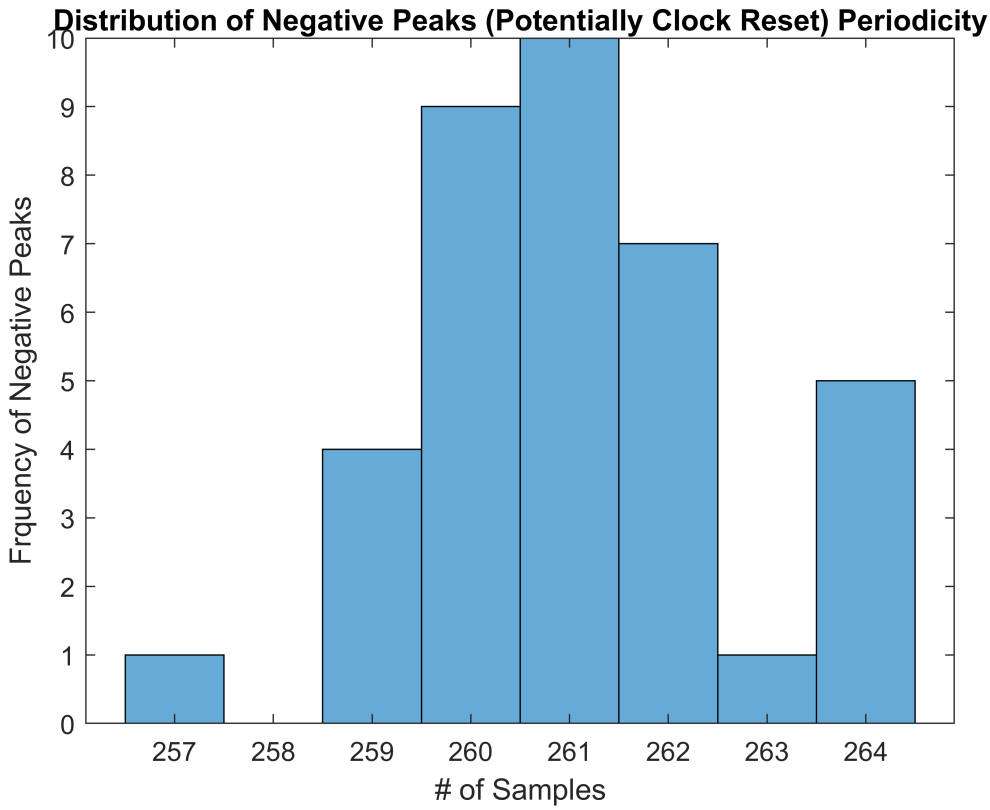
ActiCHamp

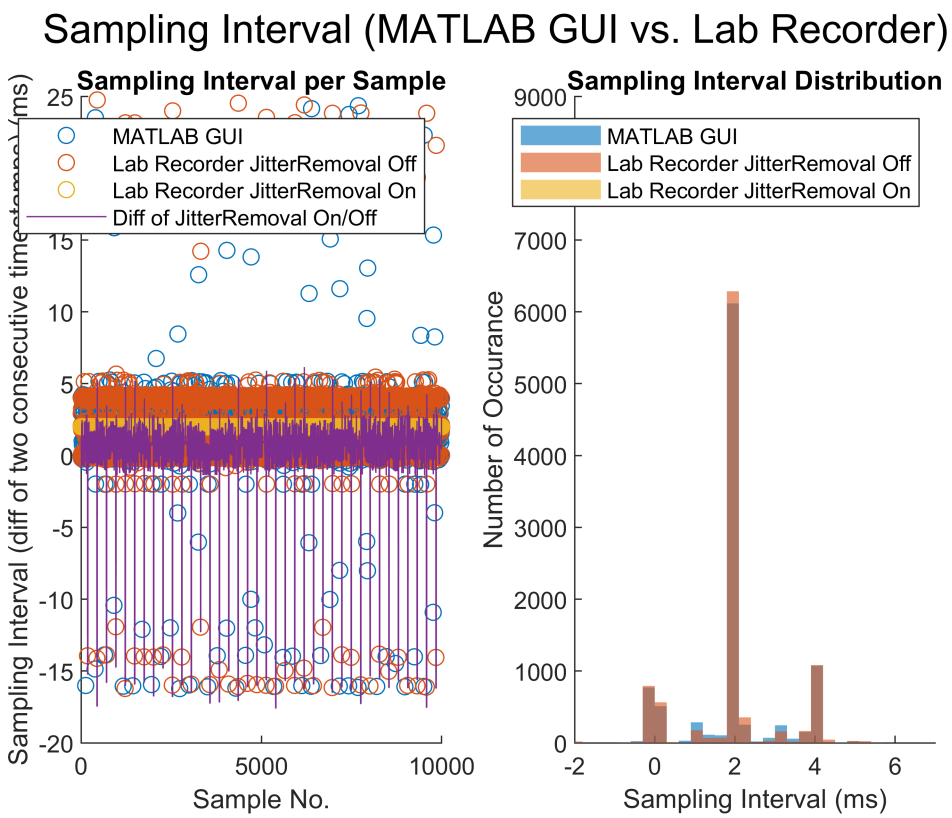
Here, we look at the ActiCHamp data recorded via the MATLAB GUI (which also uses pull_chunk in a very large loop) and Lab Recorder. When loading the Lab Recorder recorded data to MATLAB workspace via load_xdf(), we have a setting 'HandleJitterRemoval' set to true by default. This setting automatically remove jitter by repositioning the timestamps in an ascending order. We tested for both the jitter removal on and off when loading the xdf data recorded by Lab Recorder.

Signal Pathway:

Sensor (EEG/Conditioned Force) -> Amplifier (ActiCHamp/LiveAmp) -> Windows (USB Wired ActiCHamp / Bluetooth Wireless LiveAmp) -> Connector Program (ActiCHamp Connector / LiveAmp Connector) -> Recorder Program (Lab Recorder / MATLAB Script / MATLAB GUI)

```
sample_duration = 19.9967

num_negative_peaks = 38

nominal_negative_peak_periodicity = 263.1316
```


Distribution of Negative Peaks (Potentially Clock Reset) Periodicity
Frquency of Negative Peaks
# of Samples
0
1
2
3
4
5
6
7
8
9
10
257
258
259
260
261
262
263
264


Sampling Interval (MATLAB GUI vs. Lab Recorder)
Sampling Interval per Sample
Sampling Interval (diff of two consecutive timestamps) (ms)
Sample No.
MATLAB GUI
Lab Recorder JitterRemoval Off
Lab Recorder JitterRemoval On
Diff of JitterRemoval On/Off
Sampling Interval Distribution
Number of Occurance
Sampling Interval (ms)
MATLAB GUI
Lab Recorder JitterRemoval Off
Lab Recorder JitterRemoval On

In the figure shown above, the left subplot is the sampling intervals for each condition. The yellow circles indicate the jitter removed Lab Recorder recording. They located exactly at the 2 ms position without any outliers. However, the same Lab Recorder recording loaded without jitter removal, shown as the orange circles, are nearly as inconsistent as the MATLAB GUI recordings indicated by the blue circles. However, the orange circles are located mainly at the 0, 2, and 4 ms positions with outliers located near the extremeties (25 and -15 ms). Where as the outliers for the blue circles are more evenly distributed. The left subplot is the distribution of the sampling intervals for each condition. The distributions from the MATLAB GUI recording and the Lab Recorder recording without jitter removal are very similar.

Mean and STD of MATLAB GUI and Lab Recorder:

The mean sampling interval for both the MATLAB GUI and Lab Recorder are 0.0020 s = 2 ms which agrees with the 500Hz nominal sampling rate of the ActiCHamp Connector program. The sampling interval STDs from the MATLAB GUI recordings and that of the Lab Recorder (without jitter removal) are not different either.

T = 3×3 table

	Setup	Mean (ms)	Standard Deviation (ms)
1	"MATLAB GUI"	2.0001	1.8898
2	"Lab Recorder JitterRemoval Off"	2	1.9523
3	"Lab Recorder JitterRemoval On"	2	0

Comments: The implementation of retrieving data from LSL EEG Streams in the MATLAB GUI could be suboptimal. However, the inconsistent sampling intervals exist for both the MATLAB GUI, MATLAB Script, and the Lab Recorder. Thus, the inconsistency possibly does not come from the recorder stage either. Changing implementation (shown by pausing differently in the previous section) might help improve the consistency, but cannot solve the problem entirely. The sampling interval inconsistency could be an LSL inherent problem since the timestamps are generated. A good news is that LSL don't change the order of the data samples and don't miss samples under normal network conditions. We are expected to get all the data samples even though the timestamps could be inconsistent.

Function Generator

The result of Experiment 2o-1 demonstrated that the timestamps recorded by LSL are inconsistent, disregarding the choice of the sensors or recorders. The true source of the sampling inconsistency is still unknown.

Based on the LSL documents/guides,

1. The LSL system will not change the order of the sampled data.
2. The LSL system will not change the value of the sampled data, only the timestamps.
3. The LSL system only considers the timestamps of the data when performing de-jittering or clock sync.
4. The LSL system might have irregular sampling interval due to multiple reasons, including intrinsic device delay, periodic clock drift sync, and jitter.
5. The LSL system interpolate the data based on the raw timestamps by considering the nominal sampling rate and any identified gaps in the recording.

6. The dejittering algorithm is in load_xdf() and is listed on the side. Note that this function only changes timestamps, not values. This function does not reorder sample sequence.

To experimentally determine the sampling order and sampling interval consistency, we design this experiment. In this experiment, we use a function generator to generate periodic triangular wave with maximum acceptable slope magnitude.

Based on the ActiCHamp and LiveAmp limitations, the triangular wave is configured as follows:

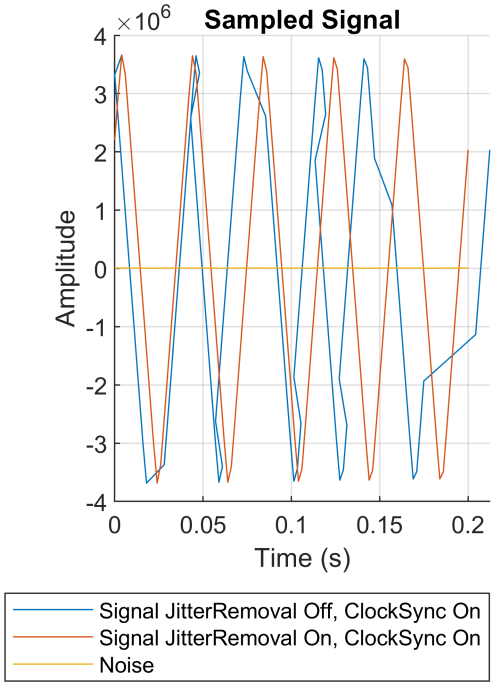
1. Frequency (Hz): 25 Hz, 12.5 Hz, 6.25 Hz, 3.125 Hz
2. Amplitude (peak-to-peak): 8V, 7.8~7.9 V read by the oscilloscope
3. Duty Cycle: 50%
4. Offset: 0 V
5. Attenuation: 0dB

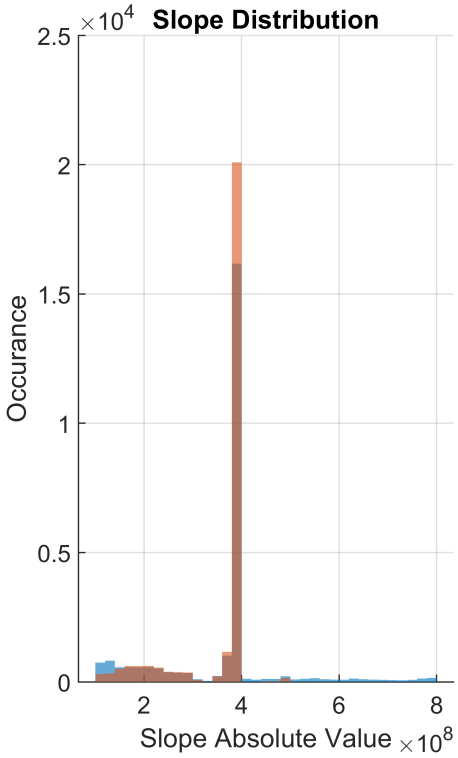
Now we obtain the slope of the sampled signals.

We have the following understandings of the sampled signals:

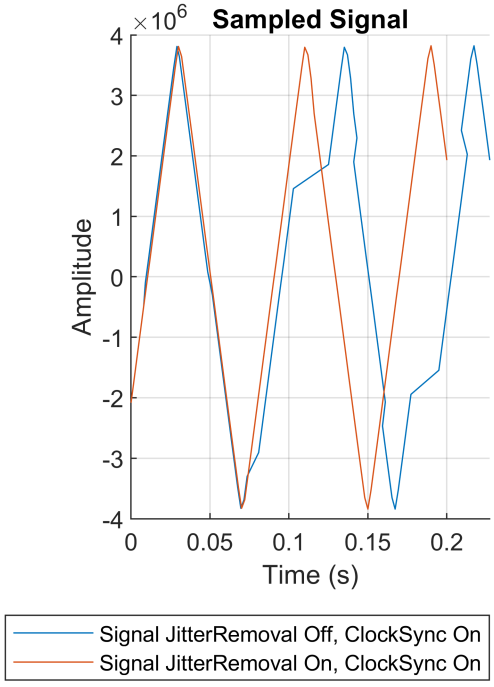
1. If the signal is sampled with consistent and uniform sampling intervals by the ActiCHamp or LiveAmp, but the timestamps are inconsistently generated by the LSL when the sampled signal reaches the LSL connector program, the calculated signal slope from the raw signal will have a wider distribution than the signal slope from the dejittered signal. And we should trust and use the dejittered signal.
2. If the signal is sampled with inconsistent sampling intervals by the whole ActiCHamp/LiveAmp - LSL system, the calculated signal slope from the raw signal will have a narrower distribution than the signal slope from the dejittered signal. And we should not use the dejittered signal. And LSL is not to be trusted.
3. If LSL actually re-orderes signals or results in non-monotonical timestamps during timestamps generation or dejittering stages. We can see these non-monotonical instances on the plot of the sampled signal. And LSL is not to be trusted.

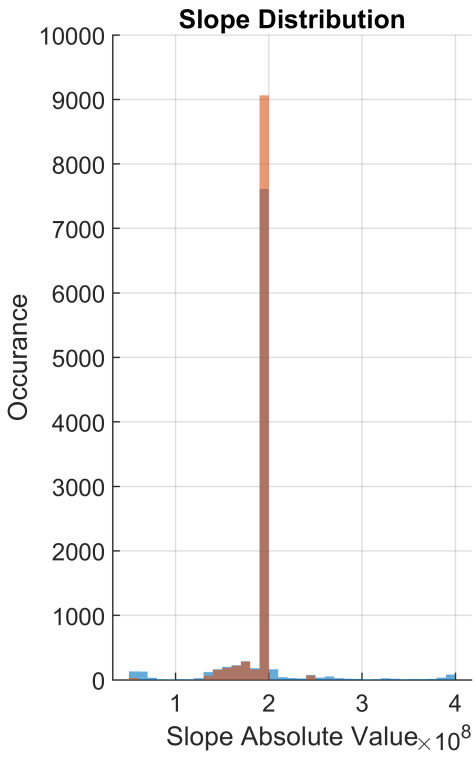
Sampled 25Hz Signal

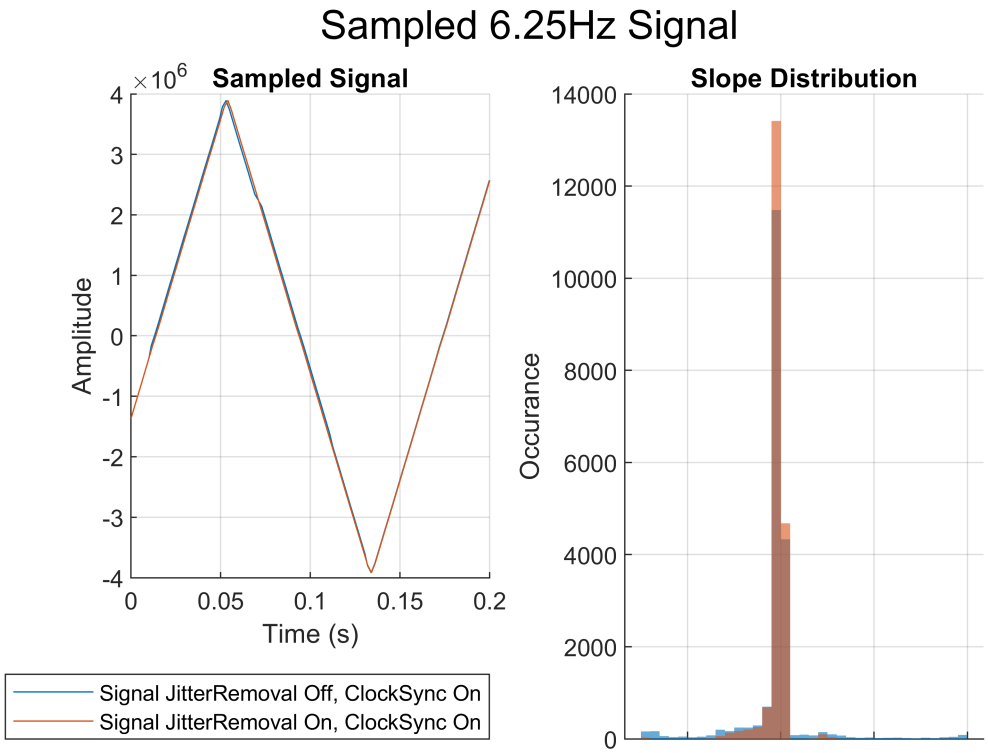


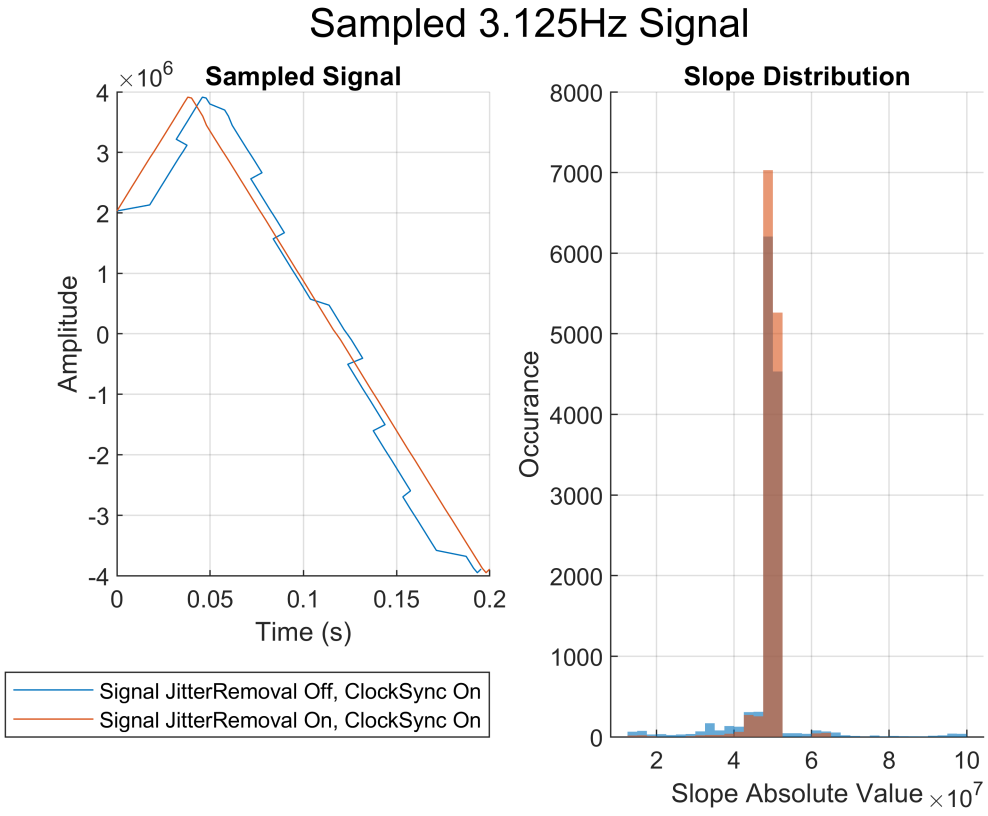


Sampled 12.5Hz Signal







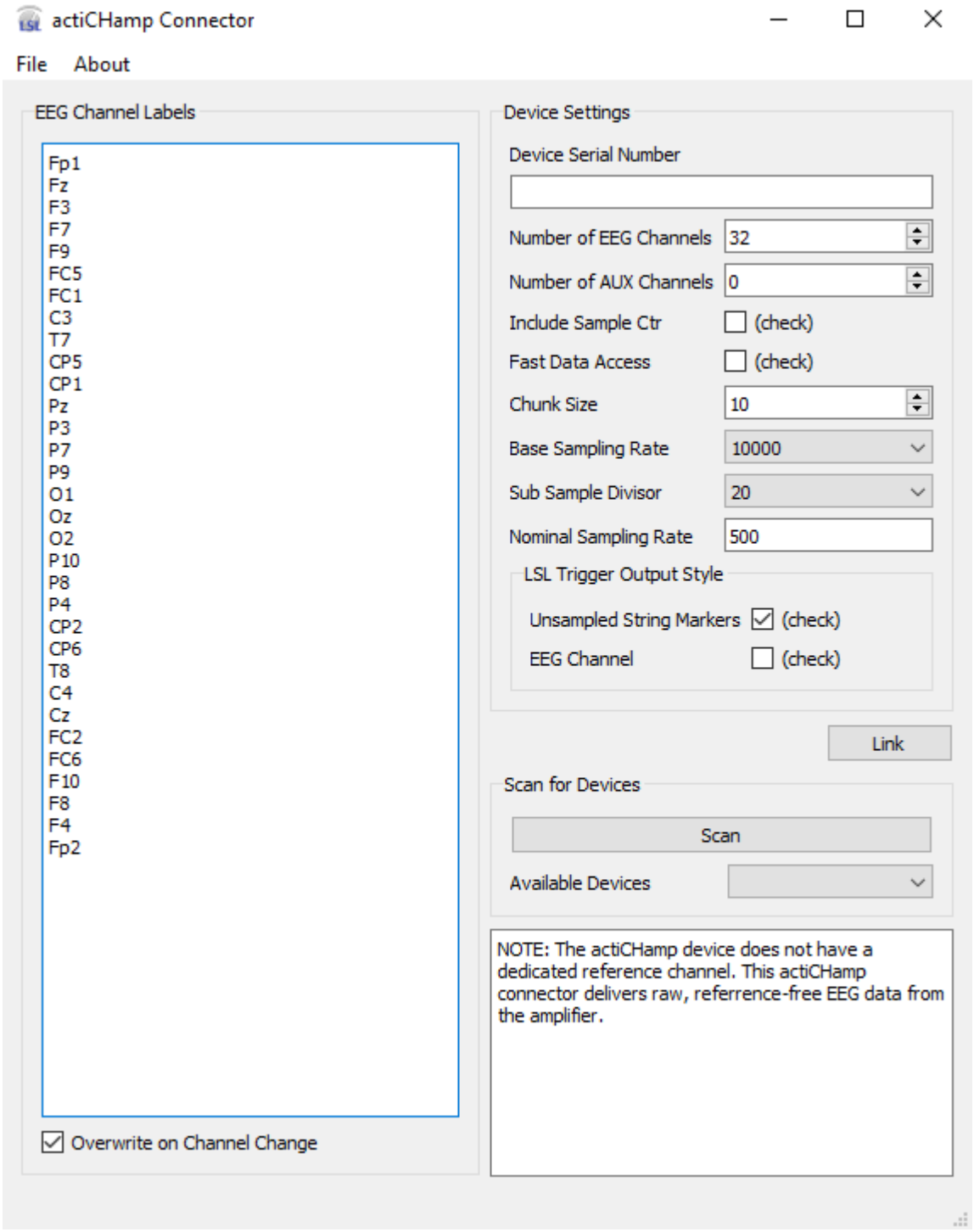


If we calculate and plot the mean and STD of the signal slope with 'HandleJitterRemoval' On and Off. We have:

T3 = 4×5 table

	Signal Frequency (Hz)	Mean (Off)	Mean (On)	STD (Off)	STD (On)
1	"25 Hz"	4.2788e+08	3.5253e+08	1.9701e+09	8.8913e+07
2	"12.5 Hz"	1.0799e+09	1.8628e+08	5.4133e+10	3.9095e+07
3	"6.25 Hz"	2.0496e+08	9.6193e+07	5.6221e+09	1.4266e+07
4	"3.125 Hz"	2.0281e+08	4.8899e+07	6.2476e+09	5.1190e+06

Comments: The results shows that the assumption of LSL dejittering function, the sampling interval of the ActiCHamp sensors are reliable/consistent, is valid. We should enable the 'HandleJitterRemoval' feature of load_xdf() to ensure data quality. Besides, we should check the "EEG Channel" box under the LSL Trigger Output Style to ensure that the trigger outputs can also be dejittered by the load_xdf().



Conclusion

The sampling inconsistency could be an inherent nature of the LSL system in the connector stage, when the timestamps are generated. But as long as the physical sensor system is reliable, we can trust the auto jitter removal of the LSL system. Given all the above analysis, we should still prioritize the Lab Recorder for data recordings to ensure timestamp consistency as we can perfrom jitter removal and it is the most supported by the Brain Products. For all the subsequent experiments except the 2a Weight vs. Human Subject, we will use the Lab Recorder for recording.

Experiment 2a: Trigger Button vs. Load Cell (Force Sensor)

Here we look at how well the trigger button signal can be used to replace the force sensor signal in detecting the release event onset. The trigger button is connected to the Trigger Box and is a marker signal. For each release, the trigger button is pressed and released. When the button is pressed, the relay on the metal plate connects and drive the solenoid to trigger the bow release. Thus, the main delay comes from the electro and mechanical structure of the tethered release system, while a small portion of the delay could come from the hardware of the Trigger Box and the ActiCHamp. We measure the sum of two differences here.

Analysis

Weight vs. Human Subject (ActiCHamp, recorded via MATLAB GUI)

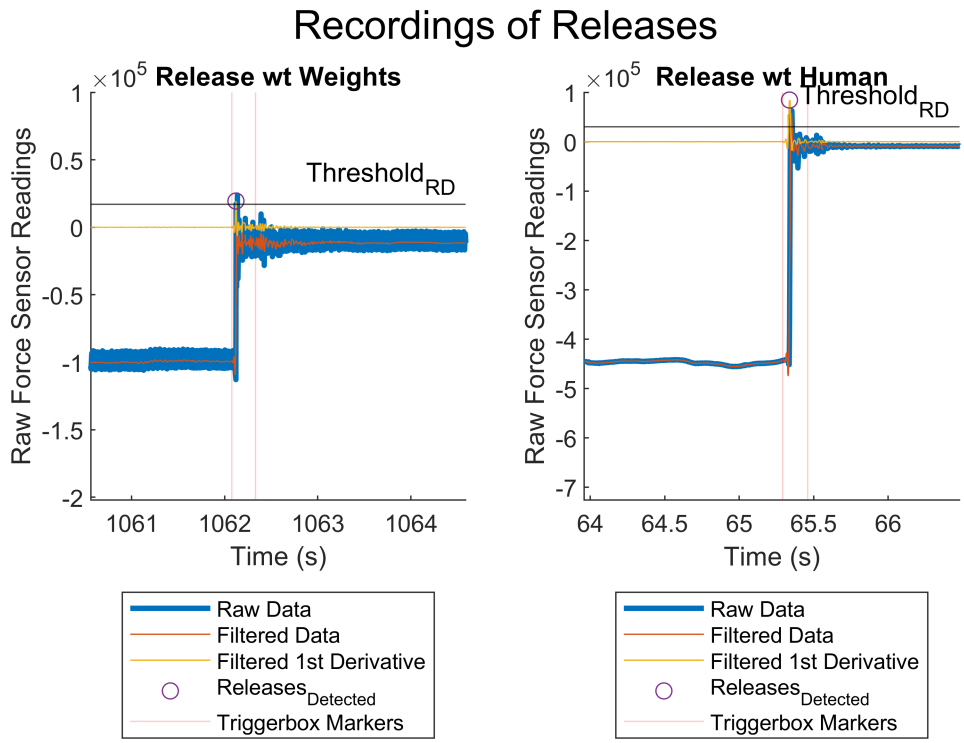
First, we look at a few release trials taken by the MALAB GUI, the GUI uses pull_chunk() in a very large loop (the other task for each loop includes updating the plots, checking status of each user input, saving the new data to workspace, etc). The exact time for each loop is not investigated but could be around 500 to 1000 ms.

Below are some basic information of the recording:

```
The mean sampling interval of the recording is: 2.8585 ms

The pre-set nominal sampling rate of the Stream Connector is:500 Hz

The release detection thresholds are: 17000, 30000

The number of recorded button presses: 22   9
```

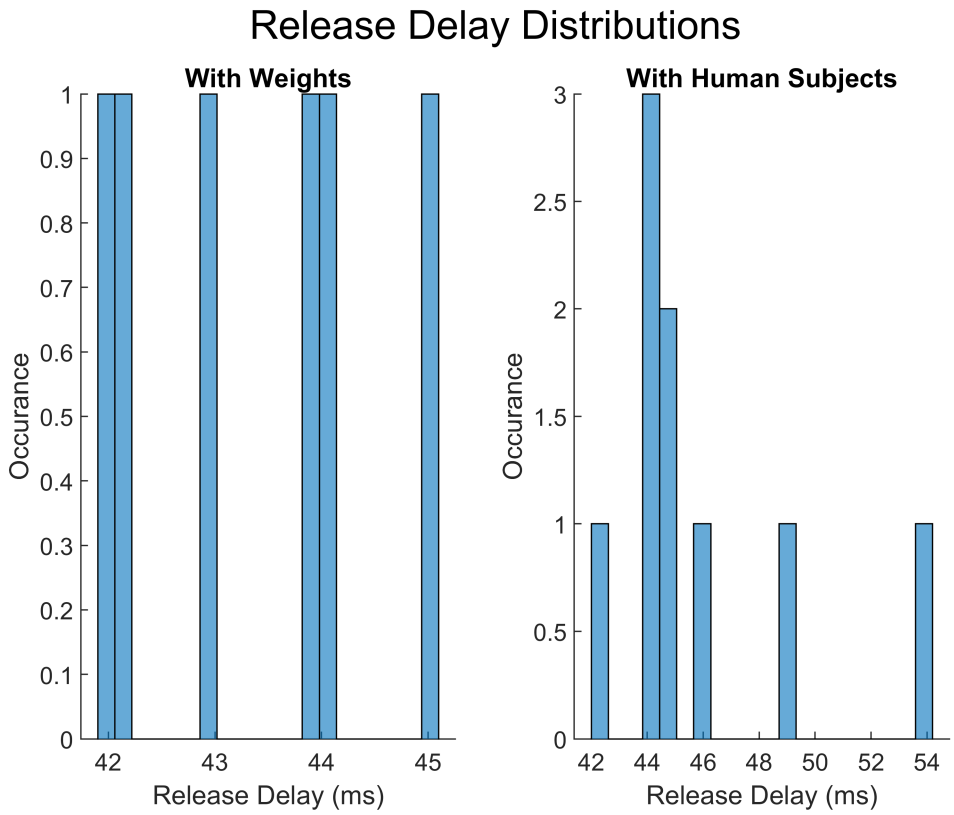
The standard processing pipeline for determining the release delay onset is broken down as follow:

1. The raw force data is filtered by a low-pass minimum-order filter to reduce noises from powerline and other sources.
2. Numerical derivatives (acceleration) are calculated from the filtered force data. This centered the data around 0 except release onset, making thresholding easier.
3. Peaks are detected from the derivatives given a peak threshold as the min peak height, and peak prominance (ended up not used here).
4. Timestamp of detected peak is taken as the release event. Timestamp of the trigger button press is taken as the trigger signal.
5. Difference between two timestamps are taken as the release delay.



The plot shown above are two releases. The left subplot is recorded with weights acting on the tether, the right subplot is recorded with Human Subjects leaning. The raw force data, the filtered force data, the derivative of the filtered force data, and the peak (release onset, indicated as the purple circle) detected are shown in each subplot. The release delay is the time difference between the left pink marker line and the purple circle.

```
The number of detected releases (tension drops) is: 6, 9
```



The distribution of the release delay recorded by MATLAB GUI is shown below:

```
pd_3 =
  NormalDistribution

  Normal distribution
       mu = 45.8815   [43.1357, 48.6273]
    sigma =  3.5722   [2.41287, 6.84351]
```

Comments: When recording via the MATLAB GUI, the release delay has a mean value of around 45 ms with a STD of 3.6 ms. The mean release delay is about the same as the ActiCHamp release delay measured via the Lab Recorder, as shown in next section, reasons are unknown. The standard deviation of the delay is about the same as that of the release delay via the Lab Recorder. To ensure the sampling consistency and mininal influence to the sampling, the Lab Recorder will be used for the rest of the analysis.

Due to the limited number of releases here, we cannot draw meaningful conclusions yet. In the next section, we will analyze more trials recorded from the Lab Recorder.

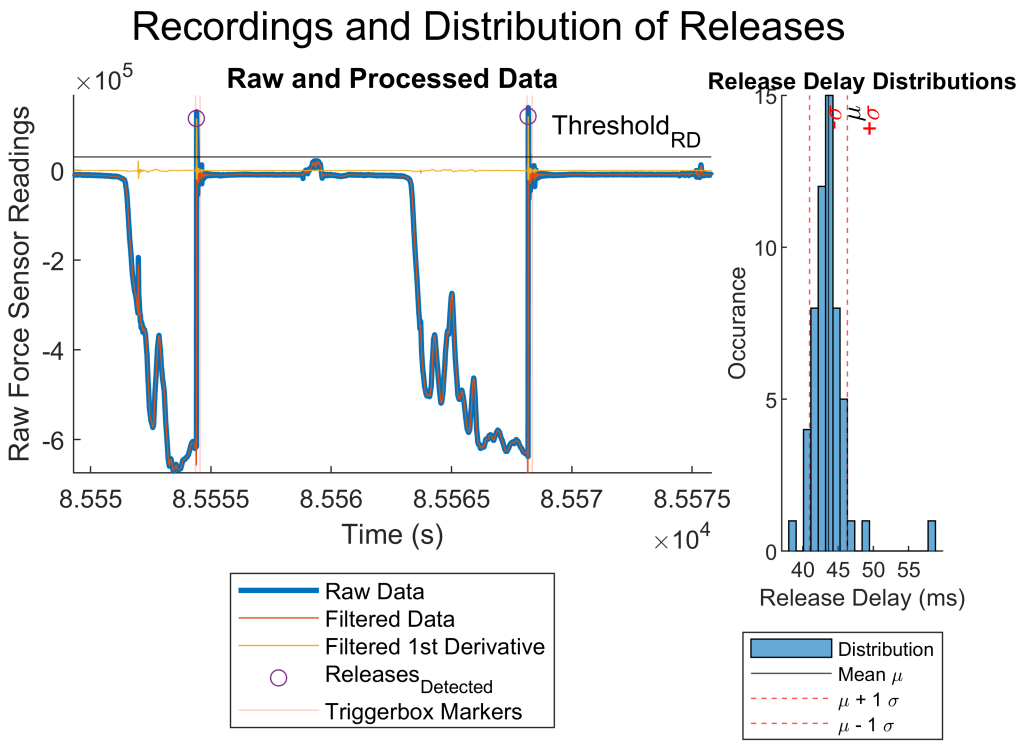
Human Subject Releases (ActiCHamp, recorded via Lab Recorder)

50+ release trials on ActiCHamp and the Trigger Box system are recorded via the Lab Recorder.

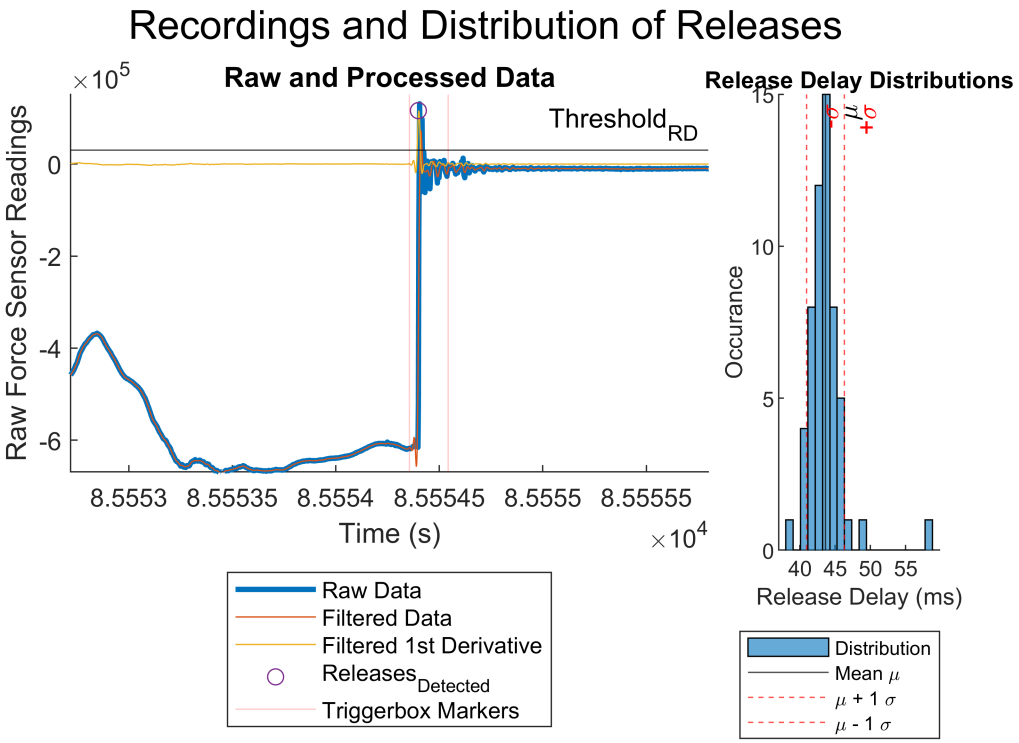
```
The mean sampling interval of the recording is: 2 ms

The pre-set nominal sampling rate of the Stream Connector is: 500 Hz

The release detection thresholds is: 30000

The number of recorded button presses: 56
```

```
The number of detected releases (tension drops) is: 56

pd_LAB =
  NormalDistribution

  Normal distribution
       mu = 43.6337   [42.9145, 44.3529]
    sigma = 2.68566   [2.26421, 3.30136]
```

The distribution of the release delay between the trigger button and the force sensor recorded via the Lab Recorder is shown above. The mean delay is 43.6 ms and the STD is 2.6 ms.



If we zoom in and show 1 perturbation, the raw data is relatively clean and the peaks are well identifiable. Only 1 peak is detected for each release trigger (red yline).



With more samples, the distribution of the release delays are shown. It is clear that the distribution roughly follows a skewed distribution? But this could also be due to the limited number of samples.

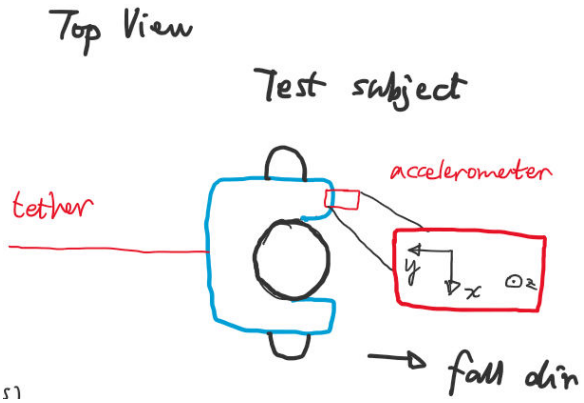
Comments: Note that the Lab Recorder recorded release delays has a mean value of around 43.7 ms, about the same as the MATLAB GUI recorded release delays (mean = 45 ms). The STD of the release delays are around 2.68 ms, with a 95% confidence interval around 2.2 to 3.3 ms. The release delay is mostly normally distributed (with 1 outlier > 55 ms, not investigated, could be due to peak detection algorithm). Note that the nominal sampling interval of Lab recorder is 2 ms. So the fluctuations of the STD of release delays are within 1-2 data samples. Considering that the N1 potential is 50-150 ms after the release, the release delay between the button press and the force sensor tension drop that falls within 15.3 ms $\pm$ 3.3 ms ($\pm 1\sigma$, 68% of cumulative probability, assuming normally distributed) should give accurate enough indicator of release.

Conclusion

The mean delay between the trigger button press and the force sensor drop is around 44 ms with an std of 3 ms. This makes the trigger button a quite reliable indicator of the release event, considering that the N1 potential could be 50-150 ms after the release and is wide itself.

Experiment 2b: Accelerometer vs. Load Cell (Force Sensor)

Now, we look at the accelerometer. I used ActiCHamp system for this experiment. Thus I used an external accelerometer connected the the Aux 5, 6, 7 channels to the ActiCHamp. The accelerometer is taped to the harness in front of the left chest. The orientation of the accelerometer is indicated below:



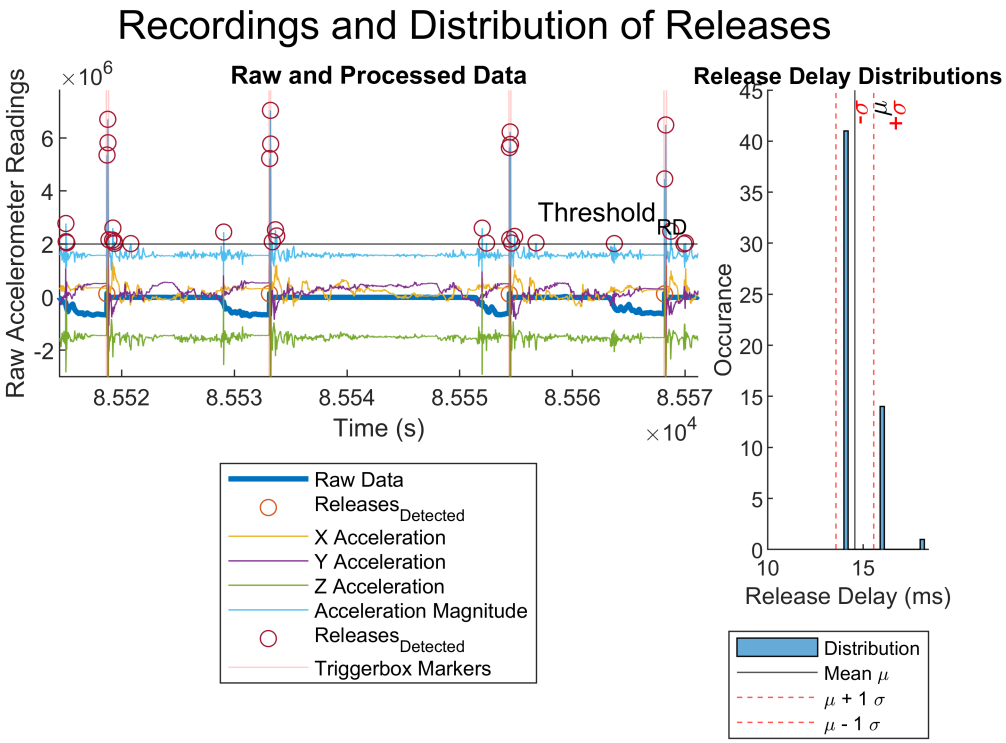
I tested the accelerometer's x, y, and z directions (Rajashree confirmed that this accelerometer is used for measuring head movement) connected to ActiCHamp. I haven't tested the on-board accelerometer orientations on LiveAmp. Signals on x, y, and z channels are all adequate to find the offset (delay). However, I am using the acceleration magnitude sqrt(x^2+y^2+z^2) for peak detection (to determine movement onset) so that the orientation of the accelerometer doesn't matter. For reference, I took notes on the ActiCHamp accelerometer orientation for the trials. We can still obtain similar delay results if we only use 1 channel.

Analysis

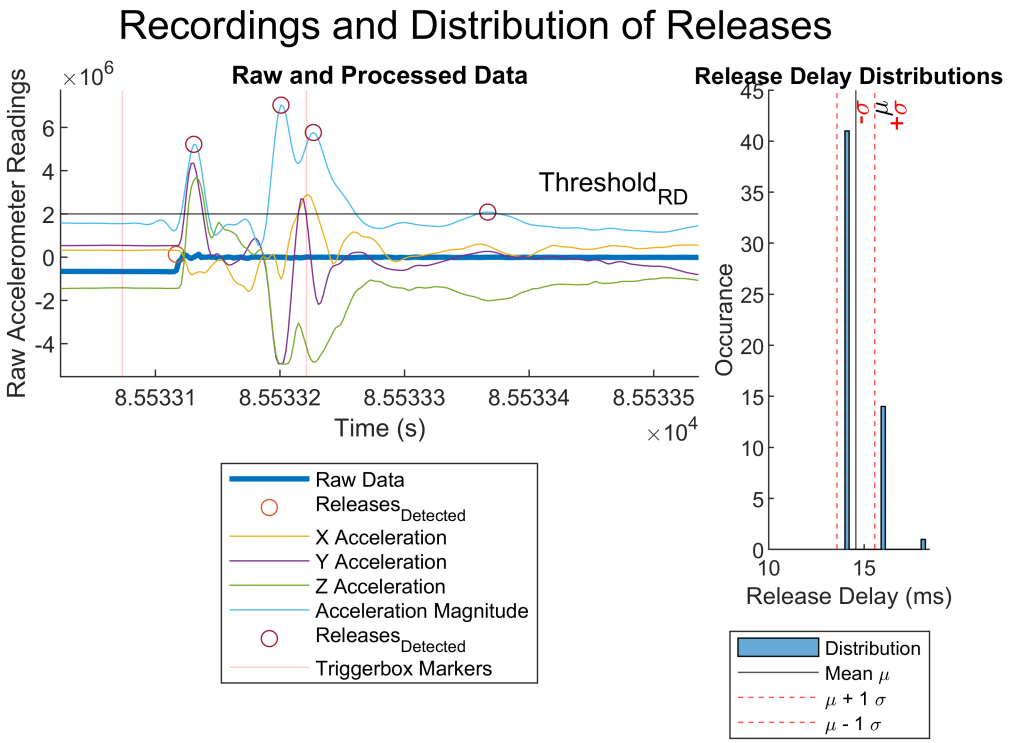
Accelerometer Delay (ActiCHamp, recorded via Lab Recorder)

```
RD_minheight = 2000000

The mean sampling interval of the recording is: 2 ms

The pre-set nominal sampling rate of the Stream Connector is: 500 Hz

The release detection thresholds is: 2000000

The number of recorded button presses: 56

pd_LAB =
  NormalDistribution

  Normal distribution
       mu =  14.5716   [14.307, 14.8363]
    sigma = 0.988256   [0.833173, 1.21482]
```

The accelerometer indicated release is very close to the force sensor indicated release. With a mean delay of 14.6 ms and a STD of 1.0 ms. The STD is slightly better than that of the trigger button. Actually, if we look into a few releases, we easily find out that the first acceleration magnitude peak after the trigger button press (causes by the release of tether, indicates the release event) is about 7 samples (if nominal sampling interval = 2 ms, 2 * 7 = 14 ms) after the force drop from the force sensor. And the result is farely consistent.



If we zoom in to show 1 release, we can identify that multiple peaks are detected for each release trigger.Multiple peaks are detected in close proximity after the release trigger (the red xline befor the first detected peak).



Comments: Looking at the distribution of the release delay of the accelerometer, we can see that 41 out of the 56 releases have an acceleration peak exactly 7 samples (14ms) after the force drop, 14 releases have the acceleration peak 8 samples after the force drop. The distribution of the acceleration peak delay proves that the accelerometer can be a very reliable indicator for the release event. Consider its ease of use, setup simplicity, and wiring, accelerometer is a even better indicator than the trigger button. However, we have to consider the multiple peaks detected after each release. We have to invest more on algorithms to accurate detect the release-related acceleration peak when given only the accelerometer data. Or we can use the trigger button marker to assist the release-related peak detection. The release-related acceleration peak is after the trigger button press and followed by a more prominant peak. The release-related acceleration peak is usually smaller in magnitude or prominance (determined by the placement of the accelerometer) when compared to the later stepping response peak as the stepping response cause a more profound impact on the subject's body.

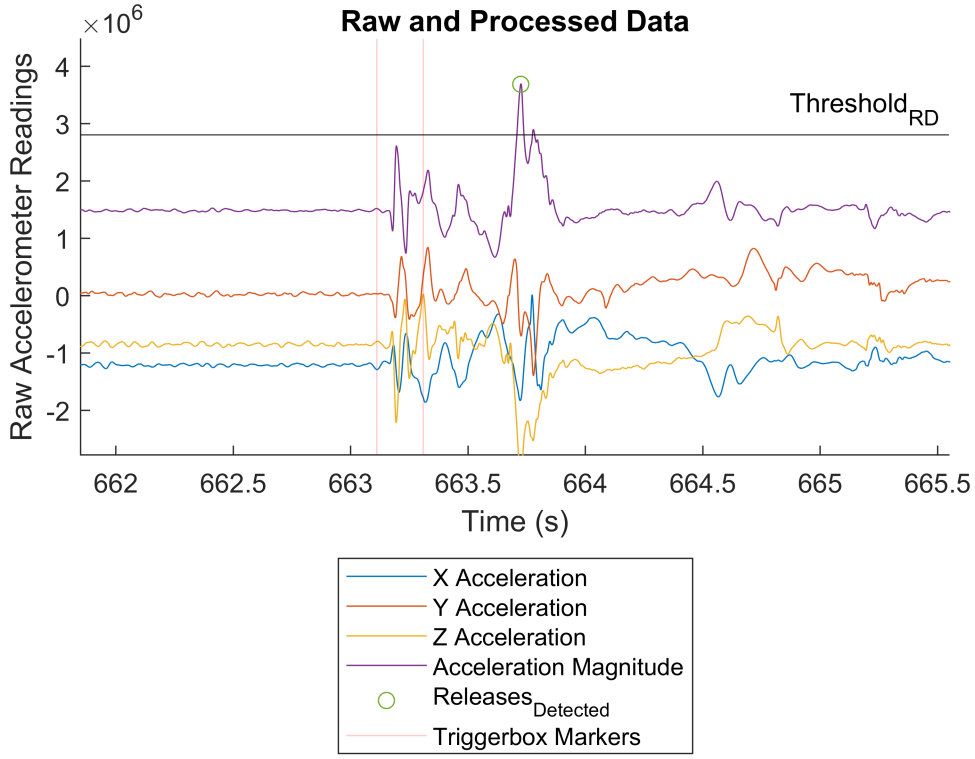
Here the accelerometer is attached to the harness around chest area. The accelerometer is attached with its x-y-z axis direction label facing up. The tether is aligned with the y-axis. The fall direction is on the negative y direction. Readings from this setup has shown clear and prominant peaks after each release trigger. However, multiple peaks are detected, as shown in the above zoomed in plot of the raw data that could be due to the oscillation of the harness and the attached accelerometer after the release. Only the first peak about the threshold after each release trigger is used for calculating the release delay to ensure consistency. By looking at x, y, z axis differently, we can see that the undesired oscillation comes from a combination of x, y, and z axis. The combinations of x, y, and z readings in the desired peak and the undesired peak are distinguishable. The z acceleration is positive in the desired peak and is negative in the undesired peak. However, more in-depth analysis is required for yielding a definitive conclusion or ultilizing the acceleration readings from individual axis.

Accelerometer Position

The accelerometer position is also very crucial for detecting the release event onset. Here acceleration data is recorded with the accelerometer attached at different positions on the test's subject's body. 5 tethered release via (force sensor, trigger button, and accelerometer attached) ActiCHamp Connector and Lab Recorder, 1st release acc on shoulder, 2-3rd releases acc on forearm, 4-5th releases acc on chest. There are noises in-between releases due to repositioning of the accelerometer, thus we only present the acceleration data around 1 typical release.

```
The mean sampling interval of the recording is: 0.002 s
The pre-set nominal sampling rate of the Stream Connector is:500 Hz
The release detection min peak height is: 2800000
The release detection peak prominance is: 1500000
The number of recorded button presses: 5
```

If we zoom in to the 3rd release (when the accelerometer is attached to the forearm):



If we compare this release data to the previous release, we can clearly see that the amplitude of the release peak (in-between the two pink trigger button marker lines) is smaller than that when the accelerometer is attached to the shoulder/chest area. Plus, the release peak is followed by muleiple peaks. The peak detection algorithm detected the largest peak, whcih should corresponds to the impact of stepping to recover balance.

Comments: The placement of the accelerometer affects the amplitude and delay of its event peaks. Generally, the more directly the accelerometer is attached to the tether (for instance, the harness vs. the shoulder), the less delay and bigger magnitude of the perturbation. The detection of release event solely on accelerometer

data could unreliable when the release acceleration peaks are small in magnitude with current algorithm. A working hypothesis of the different amplitudes and delays of the accelerometer readings during the release is that the human body segments and clothes are spring damper systems that causes delay and soften the sudden impact at the moment of release. Thus, a rigid connection between the accelerometer and the harness is crucial for minimizing the delay (from the release event to the accelerometer peak detection) and maximize the amplitude of the peak (as shown in the previous section).

Conclusion

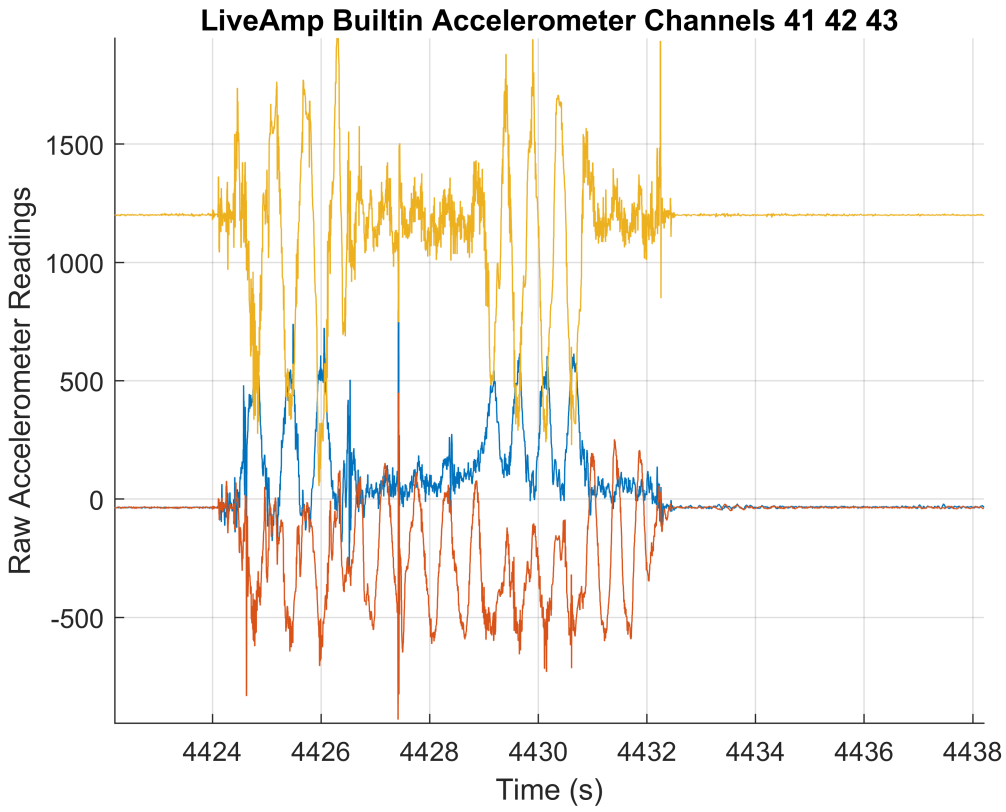
The mean delay between the force sensor drop and the accelerometer peak is close to 14 ms with an std of 1 ms. This also makes the accelerometer a very good indicator of the release event with small delay and better reliability when compared to the trigger button (when properly placed on the subject). However, the placement of the accelerometer (in the ActiCHamp case) or the LiveAmp is crucial for resulting in a clear signal (large magnitude, large prominance) and a small delay.

Accelerometer (LiveAmp Built-in vs. Stand-alone)

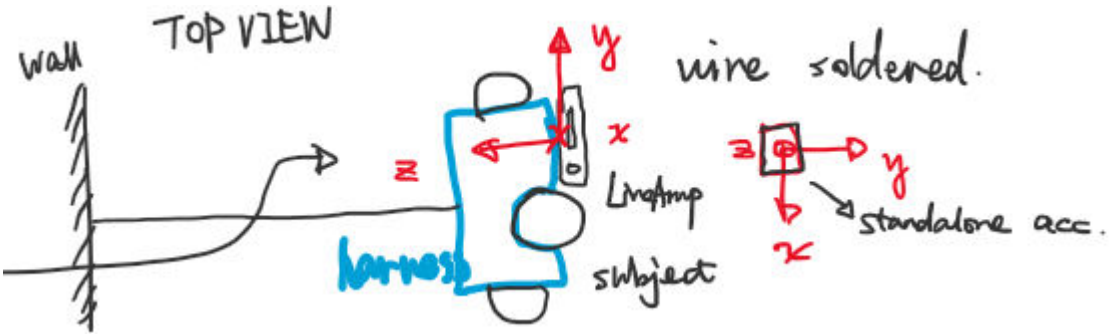
As in the previous section we concluded that the accelerometer is a good indicator of the release event when placed properly. LiveAmp has a built-in accelerometer that is very convenient but offers less flexibility in the placement when compared to a stand-alone accelerometer that can be taped almost everywhere on the test subject. Thus, we want to investigate whether the LiveAmp built-in accelerometer is sufficient for determining the release event, or a stand-alone accelerometer is necessary. In this experiment, we connected an external stand-alone accelerometer to the LiveAmp via the LiveAmp Sensors & Triggers Extention and performed 50+ releases. The results are analyzed below after determining the corresponding channels for the accelerometers.

LiveAmp Builtin Accelerometer Test

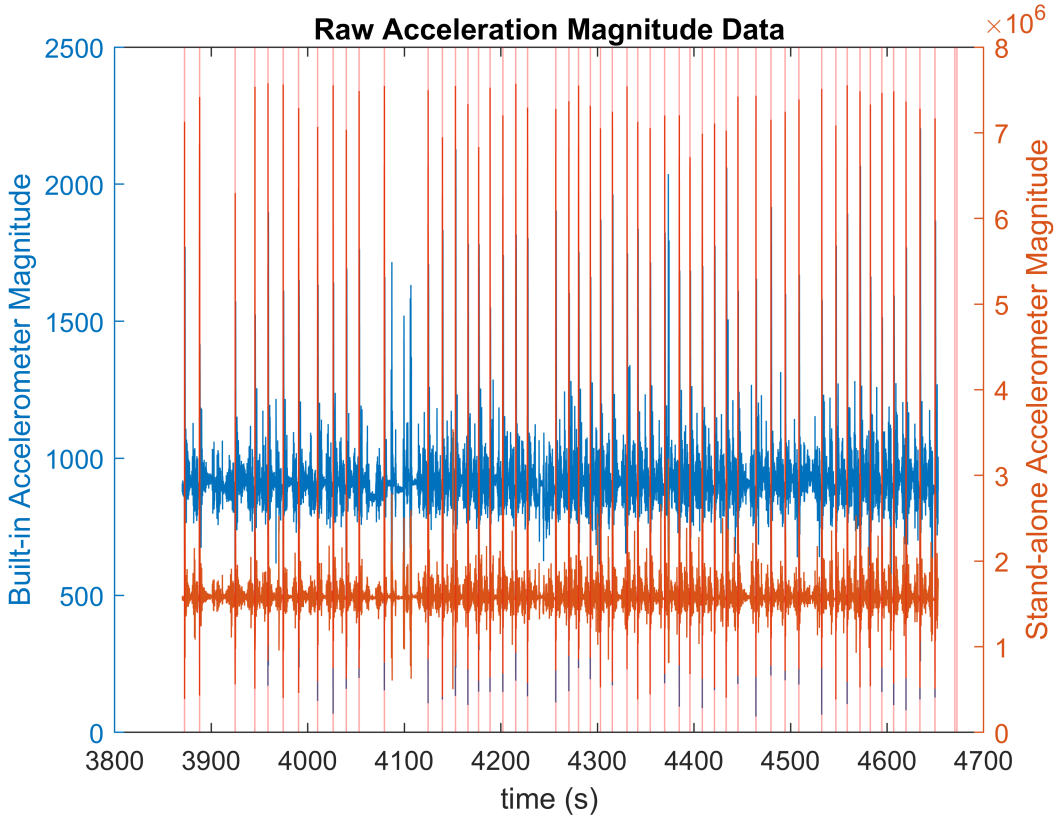
LiveAmp has a builtin accelerometer. However, the LSL LiveAmp connector program does not specify which channels do the accelerometer signals belong to. Thus, I record a test data with EEG electrodes hang in air and manipulate the LiveAmp in three directions. The resulted .xdf data recording contains 43 channels. There are 32 EEG and 8 AUX channels, leaving the additional channels for accelerometer. The plot below showing the raw readings from channel 41, 42, and 43, confirms that these 3 channels are for the LiveAmp onboard accelerometer.



Through the above plot, we have identified that the built-in accelerometer of LiveAmp has dedicated channels 41, 42, and 43 (when configured 32 EEG channels + 8 AUX channels). The channels should correspond to x, y, and z axis. But further verification is needed to comfirm the axis-to-channel correspondance. The experiment configurations are shown below:

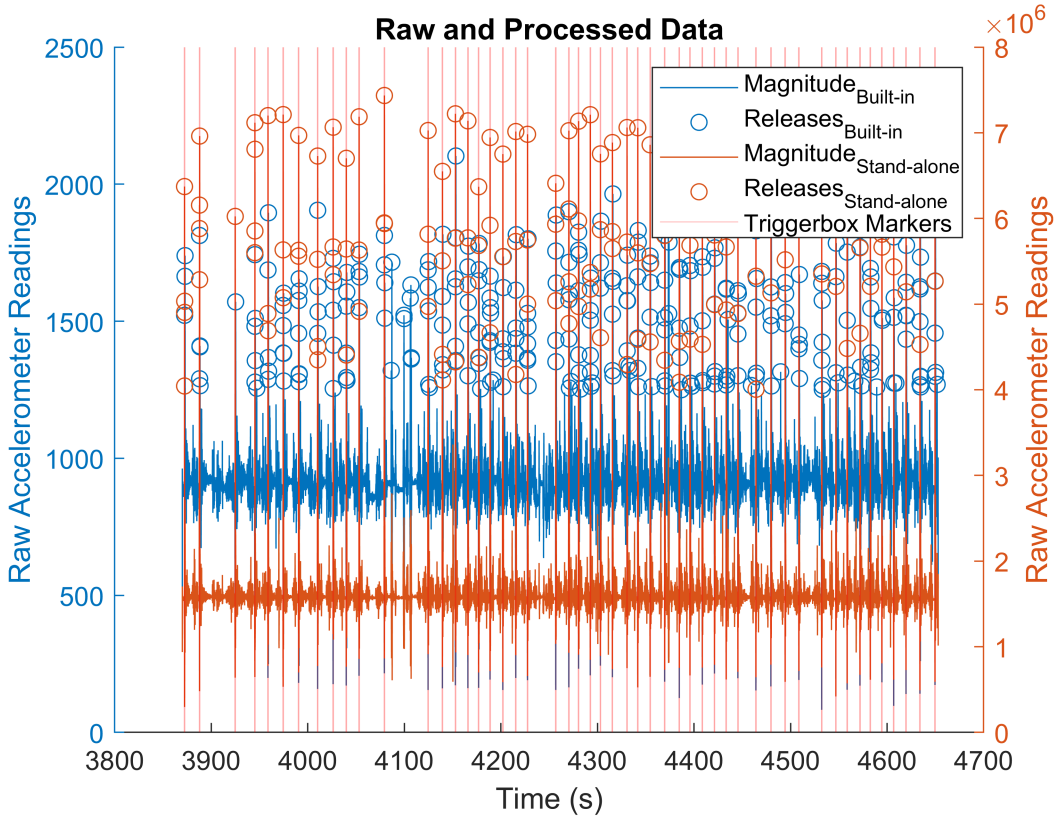


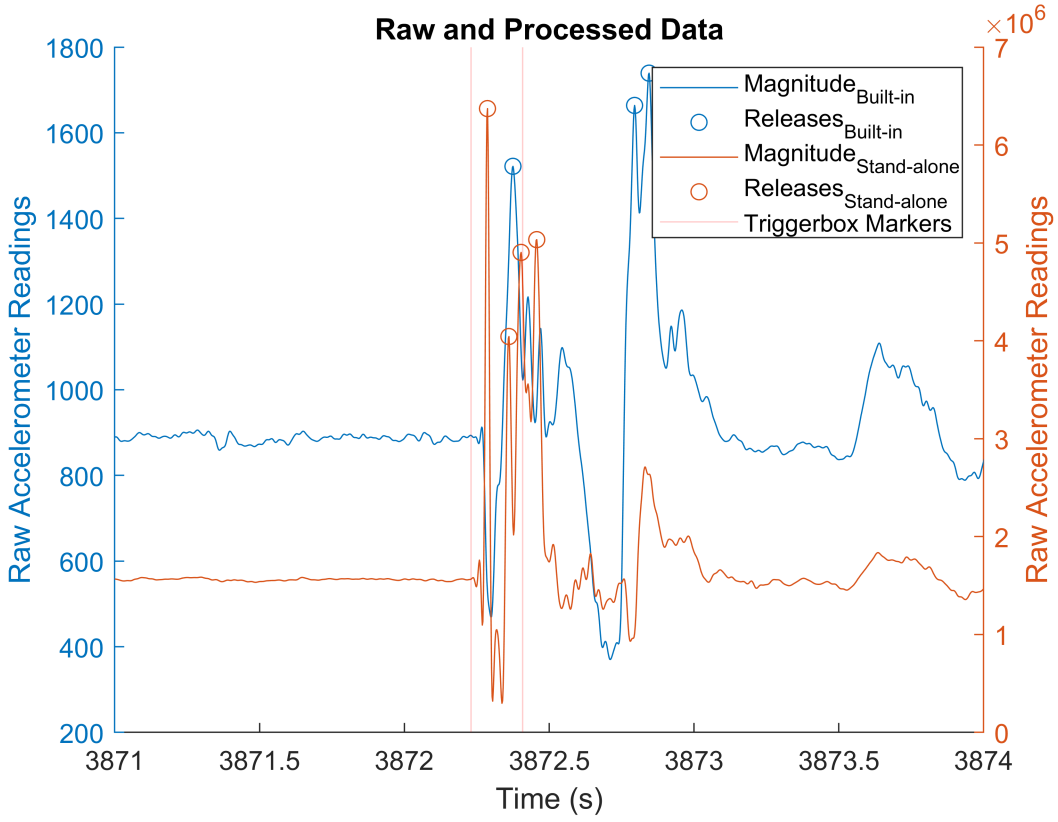

```
channel_acc = 1×3
    37    38    39
```

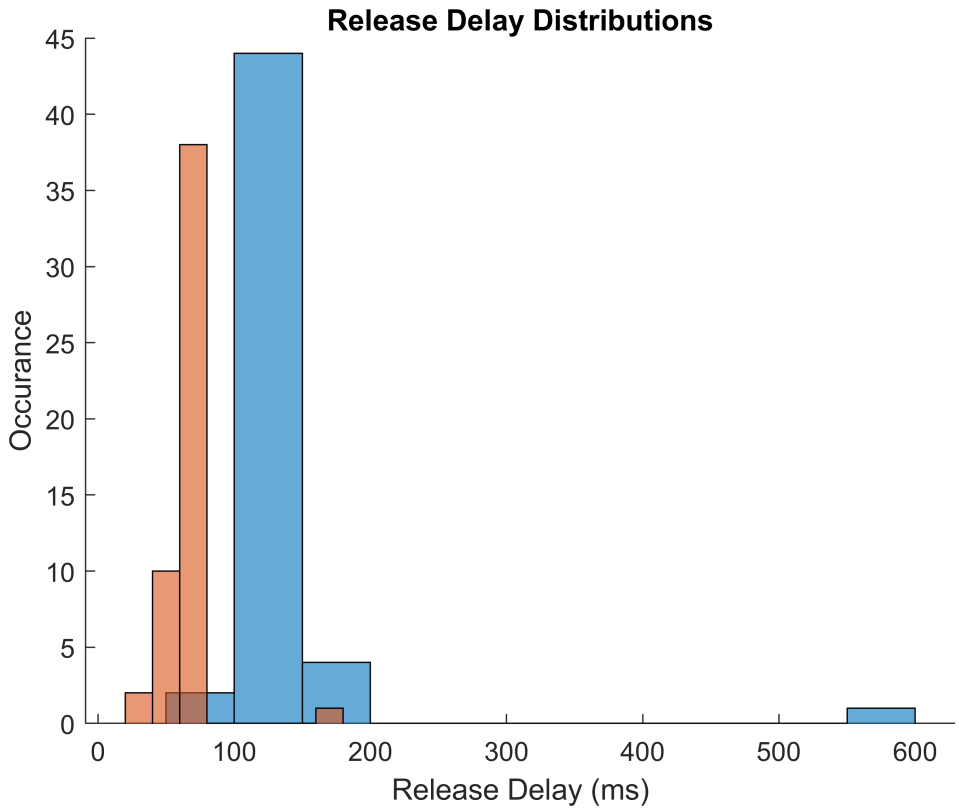


Based on the plot, we identify that at time from 3870s to 4652s are the section of experimental trials in the recording. We need to exclude the 4 undesired button press-release at the end as they are not associated with release events.

```
The mean sampling interval of the recording is: 2 ms
The pre-set nominal sampling rate of the Stream Connector is:500 Hz
The release detection thresholds is: 1250  4000000
The number of recorded button presses: 51
```


Raw and Processed Data
×10^6
Magnitude Built-in
Releases Built-in
Magnitude Stand-alone
Releases Stand-alone
Triggerbox Markers
Raw Accelerometer Readings
Raw Accelerometer Readings
Time (s)
2500
2000
1500
1000
500
0
8
7
6
5
4
3
2
1
0
3800
3900
4000
4100
4200
4300
4400
4500
4600
4700


Raw and Processed Data
×10^6
Magnitude Built-in
Releases Built-in
Magnitude Stand-alone
Releases Stand-alone
Triggerbox Markers
Raw Accelerometer Readings
Raw Accelerometer Readings
Time (s)
1800
1600
1400
1200
1000
800
600
400
200
7
6
5
4
3
2
1
0
3871
3871.5
3872
3872.5
3873
3873.5
3874




```
pd_LAB =
  NormalDistribution

  Normal distribution
       mu = 62.0492   [57.3019, 66.7966]
    sigma = 16.8792   [14.123, 20.9822]
```