Single photon interference

29/4/2025
~10am

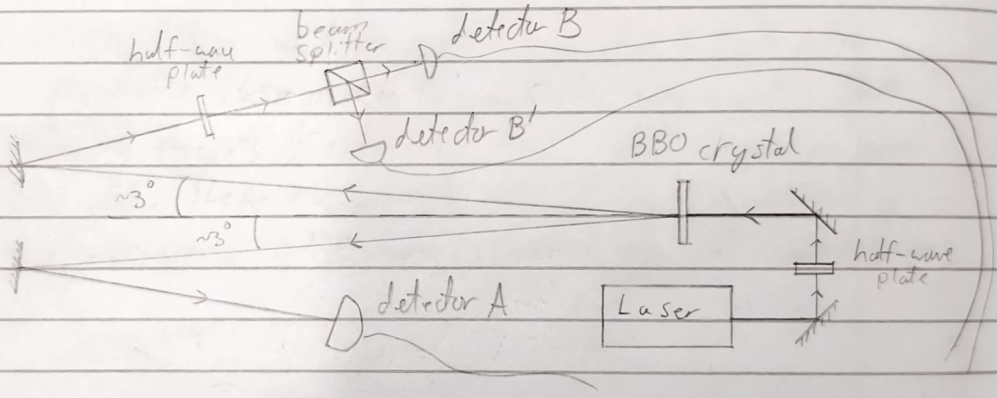
In this experiment, we will use spontaneous parametric down conversion to obtain individual photons, which we can then use for single photon interference and other experiments.

Safety:

In this experiment, we will be using a class 3B laser. Proper precautions have been taken place to avoid ~~exposure~~ eye exposure including wearing appropriate eye protection, and using an interlock switch.

Setup:

The optics setup is as follow:



Time to digital converter

Photon counting module

an detector

Note that there are coloured glass filters ~~not~~ shown to block external light.

The time to digital converter allow us to measure the photon count rate in each detector and the coincidence count between them.

The coincidence counting is important since it allow us to estimate $g^{(2)}$ which we will use to show that we obtain the single photons from this setup.

Spontaneous parametric down conversion:

A key idea in this experiment is how this β-Barium Borate (BBO) crystal produces entangled photon pairs.

to the crystal

The input wave is referred to as the pump (with $\omega = \omega_p$) while the two outputs are referred to as the signal and idler (with $\omega = \omega_s$ and $\omega = \omega_i$ respectively). Spontaneous parametric down conversion is "spontaneous" because there are no input signals, the signal and idler waves are generated spontaneously inside the crystal. The process is "parametric" because it depends on the electric fields and not just on intensities. It is called "down conversion" because $\omega_i < \omega_p$ and $\omega_s < \omega_p$.

Energy conservation requires that:

$$\hbar\omega_p = \hbar\omega_s + \hbar\omega_i$$
$$\Rightarrow \omega_p = \omega_s + \omega_i$$

Momentum conservation is equivalent to a classical condition known as phase-matching, which requires that:

$$\hbar\vec{k}_p = \hbar\vec{k}_s + \hbar\vec{k}_i$$
$$\vec{k}_p = \vec{k}_s + \vec{k}_i$$

The frequencies and wavevectors of the photons are not independent of each other, they are related by a dispersion relation.

$$k_p = \frac{n_p \omega_p}{c}$$

where n_p is the index of refraction for ω_p in the crystal. There are similar expressions for the signal and idler waves.

Recall and note that the indices of refraction depend on the frequency $n(\omega)$, if the pump, signal and idler waves are nearly collinear, energy conservation and momentum conservation

imply that the indices of refraction of all three waves are nearly the same. Most transparent optical materials, n increases with frequency (in the visible spectrum). Since ω_p is normally twice as ω_s and ω_i, it will normally have a very different index of refraction. Thus, we need a "trick" to satisfy $\vec{k}_p = \vec{k}_s + \vec{k}_i$. This trick is to use a birefringent downconversion crystal.

In type 1 downconversion, the polarizations of the signal and idler are determined by the crystal orientation, and are parallel to each other. For maximum efficiency, the pump polarization is perpendicular to the signal and idler, thus its dispersion relation is different to that of the downconverted beams.

We use β-Barium Borate (BBO), so for a laser of wavelength 405 nm, this process produces signal and idler beams of wavelength 810nm. In order to separate the signal and idler, they are chosen to make a small angle ~3° with the pump beam. Since only the relative angles between the pump, signal, and idler are important, the downconverted beams are emitted as cones surrounded the pump beam.
While the photons emitted are allowed to come out in many directions, they always come out in signal-idler pairs. These photons are emitted at the same time to a very high precision, and to distinguish specific photon pairs, we use this fact. If two photons are detected within a narrow time interval (about 8 ns) we say that they are coincident, and assume they constitute a signal-idler pair.

Photon counting and the second-order correlation function:

As a measure of coincidences, we use the second-order correlation function:

$$g^{(2)}(\tau) = \frac{\langle \mathcal{E}^*(t)\mathcal{E}^*(t+\tau)\mathcal{E}(t+\tau)\mathcal{E}(t)\rangle}{\langle \mathcal{E}^*(t)\mathcal{E}(t)\rangle\langle \mathcal{E}^*(t+\tau)\mathcal{E}(t+\tau)\rangle}$$

$$= \frac{\langle I(t)I(t+\tau)\rangle}{\langle I(t)\rangle\langle I(t+\tau)\rangle}$$

where $\mathcal{E}(t)$ and $I(t)$ are the electric field and intensity of the light beam at time t. The $\langle \cdots \rangle$ indicate time average.

It can be interpreted as conditional probability:

"I've detected a photon at time t. What is the probability of detecting another photon at time $t+\tau$?"

or more generally:

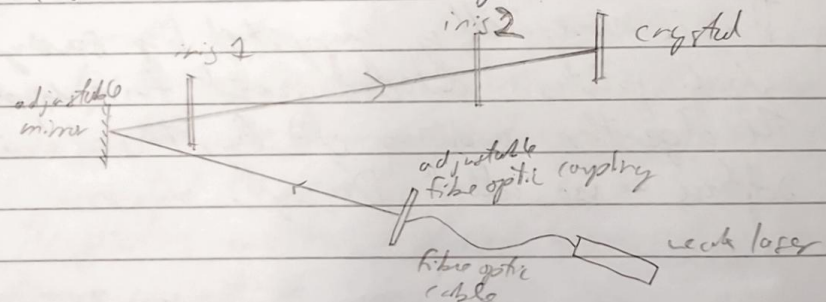
"I've detected n photons at time t. What is the probability of detecting a similar number of photons at time $t+\tau$"

This is numerically computed by saving the times at which photons are detected for each detector, and the algorithm to obtain $g^{(2)}$ from this is built in the software we use called "Daisy".

Software

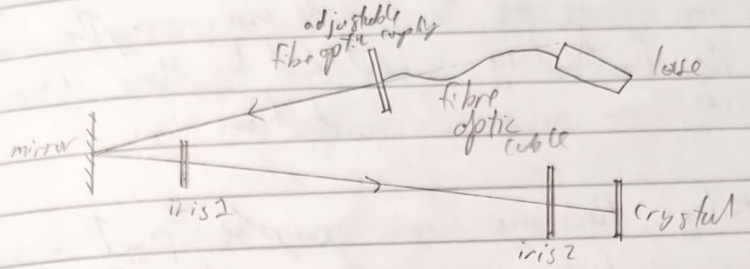
The company quTAG that designed the time to digital converter has a python API for it. The software is written in C with a python wrapper. This allows us to extract count data, histograms and $g^{(2)}$ data directly from the time to digital converter using python

~12pm To calibrate the setup, we first need to confirm that we are receiving signal-idler pairs from the crystal. Then we remove the beam splitter to do cross correlation with detector A and B.
Before we can do that, we first need to align the optics. To do this we first align the BBO crystal. This was done by my demonstrator along with the mirror placements.
Then we need to align detector A. Detector A is connected via a fibre optic cable, so we can shine a red laser backwards through the system to align it so that the red laser hits the BBO crystal.



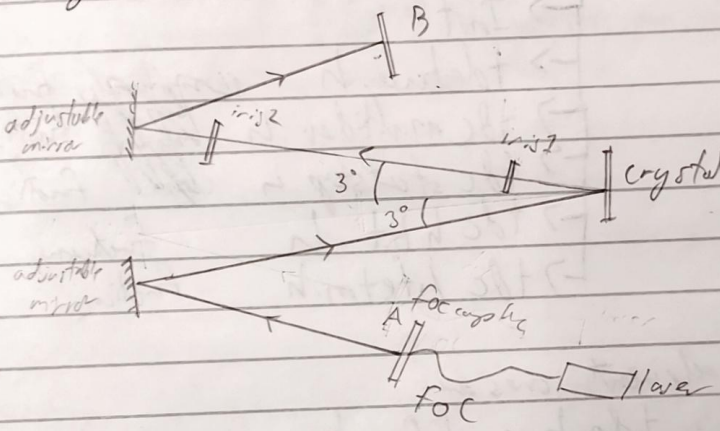
We first adjusted the fibre optic coupler (FOC) so that the laser goes through the centre of iris 1, then we adjust the mirror so that the laser goes through the centre of iris 2. Moving the mirror affects how the laser goes through iris 1, so we repeat this process until convinced the laser is appropriately aligned to the crystal

We do the same for detector B:



~~Once aligned, we need to clean the ends of the fibre optic cables as they ~~crop~~. This can be done with a ~~fibre~~ handheld fibre optic microscope~~

Once aligned, we can ~~adjust the~~ fine tune the setup by reaching a lock laser from A to B:



~2pm The python wrapper is handled by a dll file, which gives me access to the C code using python libary ctypes. All the functionality is handled by the C library TDC1.h which has online documentation in C.

Luckily, there are some example python scripts that I can effectively copy into one file. There is also a python file that effectively acts as a python library for everything:

class QuTAG
- Init
- tdcbase.h — corresponding functions
- tdcmultidev.h — corresponding functions
- tdcstatstep.h — corresponding functions
- tdchbt.h — functions
- tdclifetm.h — functions

The relevant ones are:
- tdcbase (for basic operations)
- tdcstatstep (for collecting counts)
- tdchbt (for obtaining g(2) data)

I can use the given examples to already extract count data and view a live count of counts with python.
There is also an already written file in the examples which essentially converts the C code functions into python ones.
I can effectively copy over this file and some examples to get data from the time to digital converter.
The photon counter is not on so all the values are 0, but we still get results.
I have added a way to print the device settings and generate a live count plot.

~7pm There are no examples of how to get the $g^{(2)}$ data, so I have guessed and ~~tried~~ have written a function:

```
def generate_G2(qutag):
    hbt_func = qutag.createHBTFunction()
    return qutag.calcHBTG2(hbt_func)
```

(qutag is the class mentioned earlier)
I can't test this while at home, so I'll have to wait until tomorrow.

I've also corrected ~~the~~ something I did wrong earlier: many functions have ~~data~~ unspecified units, I had been assuming units to each use case, but now I see that calling ~~qutag.timebase~~ qutag.getTimebase returns the resolution of the machine (10^{-12} s) which are counted as "bins" for which other functions measure time in.

30/4/2025

~9am Now with access to the time-to-digital counter I can test the code I wrote yesterday.
Running it just returns a float of 0. I was expecting an error saying that I can't print a ctypes function or an array? I don't really how this going to be.
Maybe if we turn on the photon counter it might give me more info.
We turned on the photon counter and ran the program with the lights off but it still just gives 0.

I'm going to look into it a bit more and see what's not working.

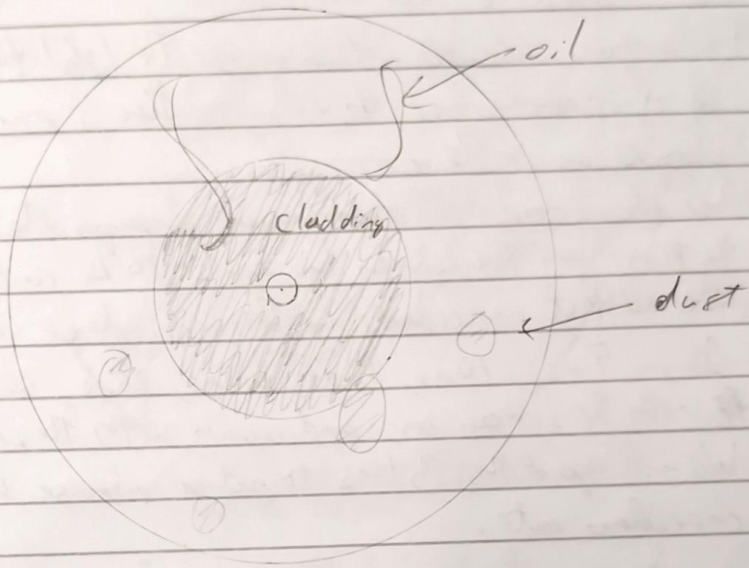
~10:30am Having looked through the documentation, it seems that qutag.create HBT Function() is the $g^{(2)}(\tau)$ function I'm looking for.

~11am The HBT function cannot be accessed directly but only with qutag.analyze HBT Function. After create HBT... is called, the program integrates over time and the state of the function can be extracted with the analyse call.

This idea doesn't work. Even though it is integrating over time (I can tell with qutag.get HBT Integration Time()), I'm just getting an array of 0s.

I currently have an unsuccessful function to generate a live $g^{(2)}$ plot.
Curiously, the array size is twice minus 1 of my specified binCount.

~12:30pm To continue with the optics, we need to clean the fibre optic cable ends. We do this with a handheld fibre optical microscope to see how dirty the end is, and a fibre optic reel cleaner to clean them.
We first check to see how dirty the end is with the microscope and see something like:



To clean them, we push on a lever on the reel cleaner to reveal the reel of cloth material, which we then use to firmly wipe the end of the cable. A very dirty end will make little noise when wiped, but a clean one will squeak. We have to be quite thorough when cleaning as a little dust could reduce our count rate by 20%-40%.

~~Once cleaned,~~
Once the both ends are cleaned, we attatch the cables to the optical system.

~1:30pm Now that the optical setup is completed and aligned, we can begin by taking a background count dataset.

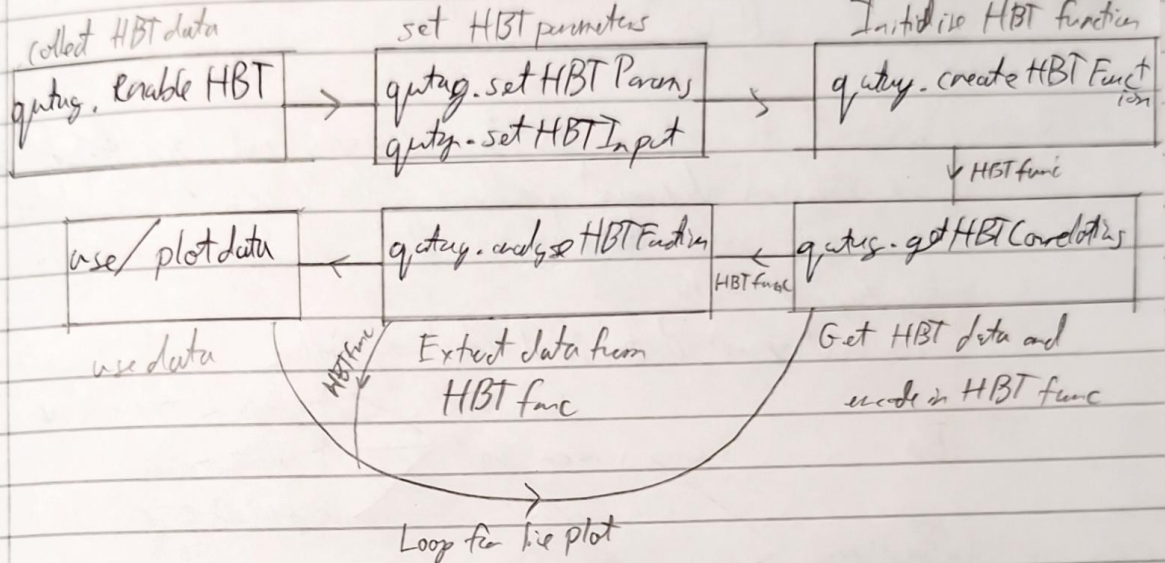
The computer screen in the room which we use to record data seems to be having a significant effect on the count rate, from ~ 1.5 kHz when the screen is off to ~ 30 kHz when it is on.
Furthermore, the light from the screen reflects off of people and objects, so there is more dependence on how people are in the room.
We also need to choose an appropriate coincidence window, the maximum time between two events to be counted as a coincidence.
We will test coincidences with coincidence window
1 ns, 5 ns, 10 ns, 15 ns, 20 ns, 25 ns, 30 ns, 35 ns, 40 ns, 45 ns, 50 ns
with the screen on and again with the screen off.
We will repeat this 3 times to get an average background count and coincidence rate.

I have written code to automate this and take each coincidence window for an exposure time of 20 seconds.

Also note that we've had some aligning problems, the crystal seems to not quite be aligned or something. Alex says he'll get it sorted.

We've two more screen off repeats to do for tomorrow.

11am After looking further into the documentation, I think I found why I was getting all 0s for the $g^{(2)}$ function. I think that calling qutag.createHBTFunction() initializes an "empty" function which I then need to "fill" with the $g^{(2)}$ data by using qutag.getHBTCorrelations:



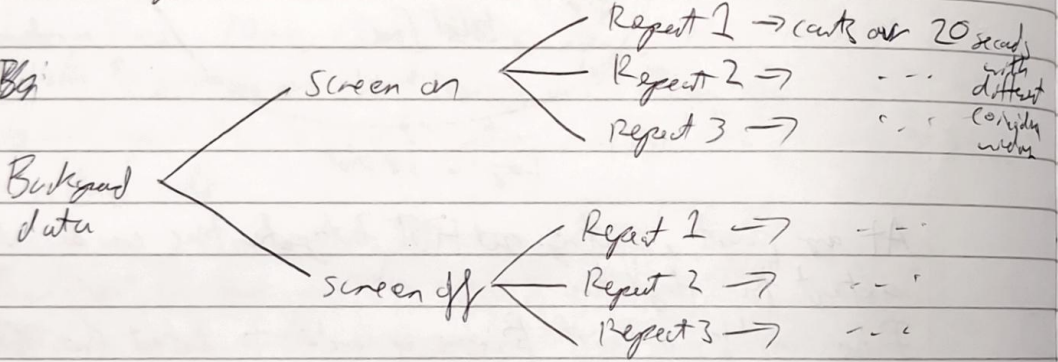
At any point, qutag.getHBTIntegrationTime can be called to extract the integration time.
I have written a brief piece of code to extract the ~~HBTfunction~~ $g^{(2)}$ data, but I cannot test it until tomorrow.

1/5/2025

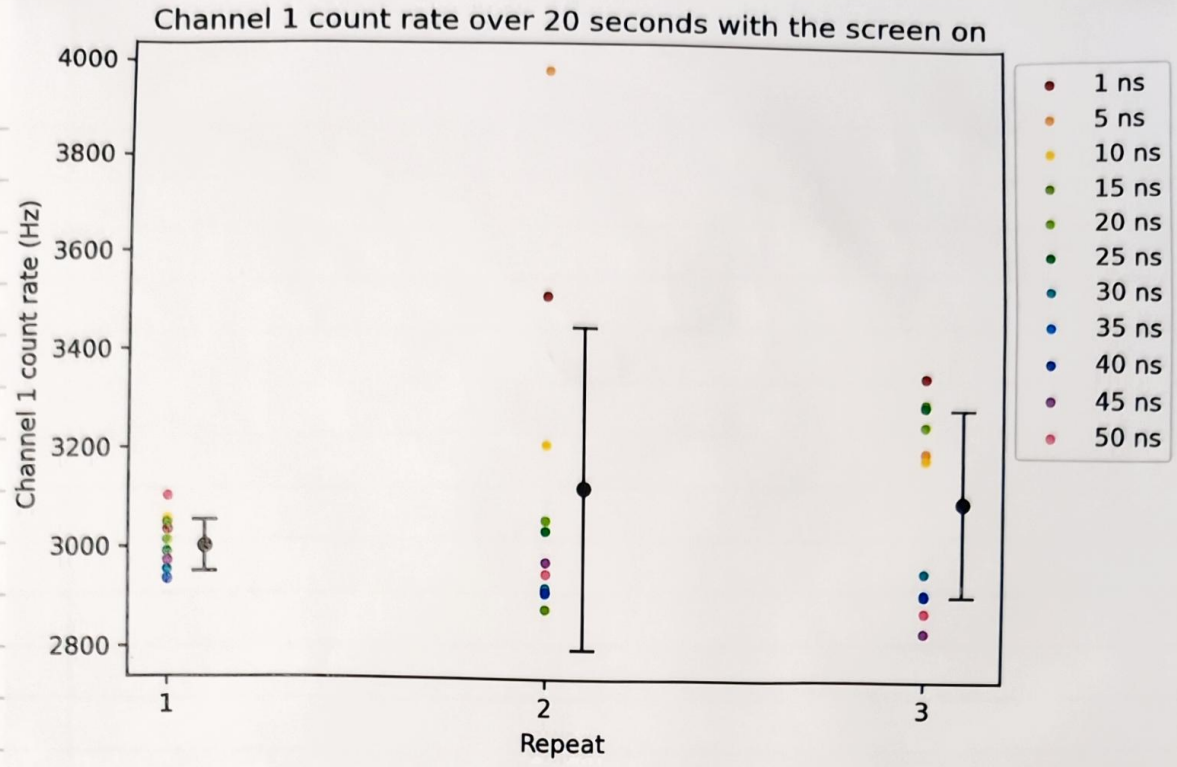
~9 am Before I do anything, we finished our background dataset. We also measured the background $g^{(2)}$ function to perhaps use it to model the noise and better identify useful data later.

After that, I tested my code and it seems to be working. However, not only is the output list $2 \times \text{binCount} - 1$, but each of it is all 0s.

~10 am We want to continue to the next experiment, but ~~the~~ we still have alignment problems with the crystal and the whole setup. While Alex fixes that, I will analyse the Bi207 background data:
The background data is divided as follows



The coinciding windows shouldn't affect the individual channel counts, so we effectively have 11 datasets of individual Bi207 channel counts per repeat.
Starting with Channel 1, screen on:

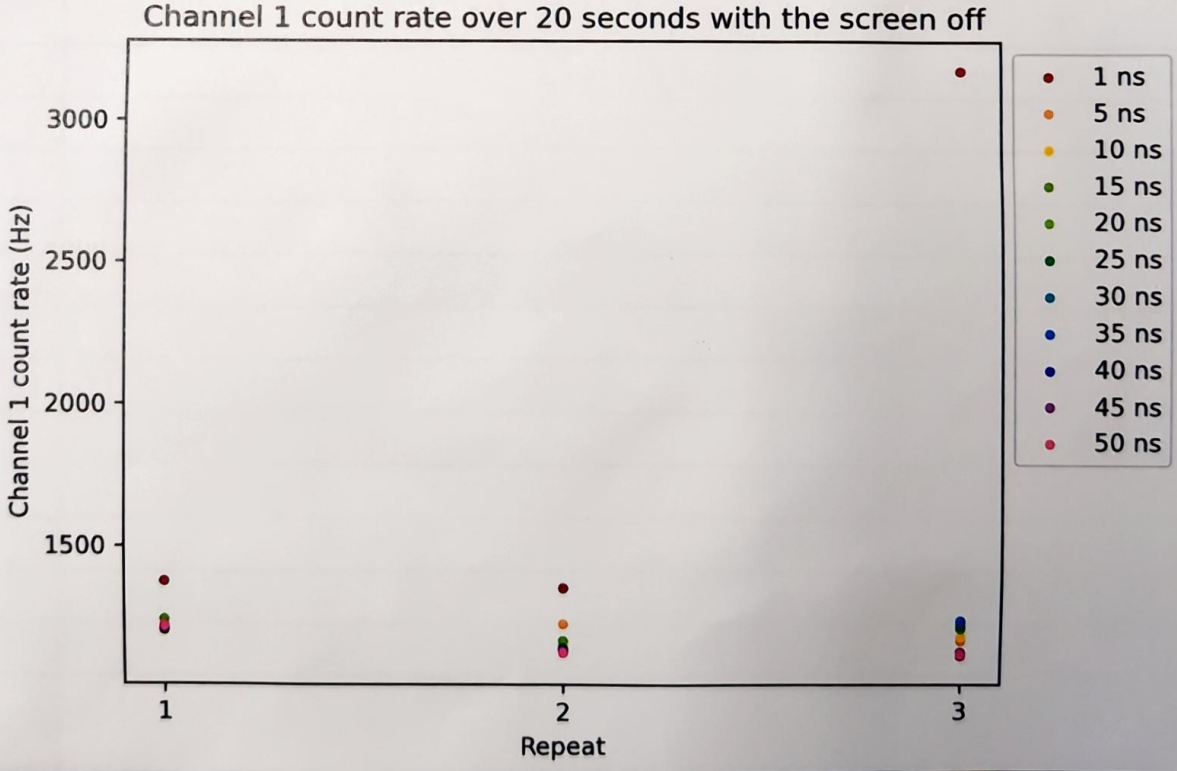


$\langle \mu \rangle = 3072 \pm 29$ Hz

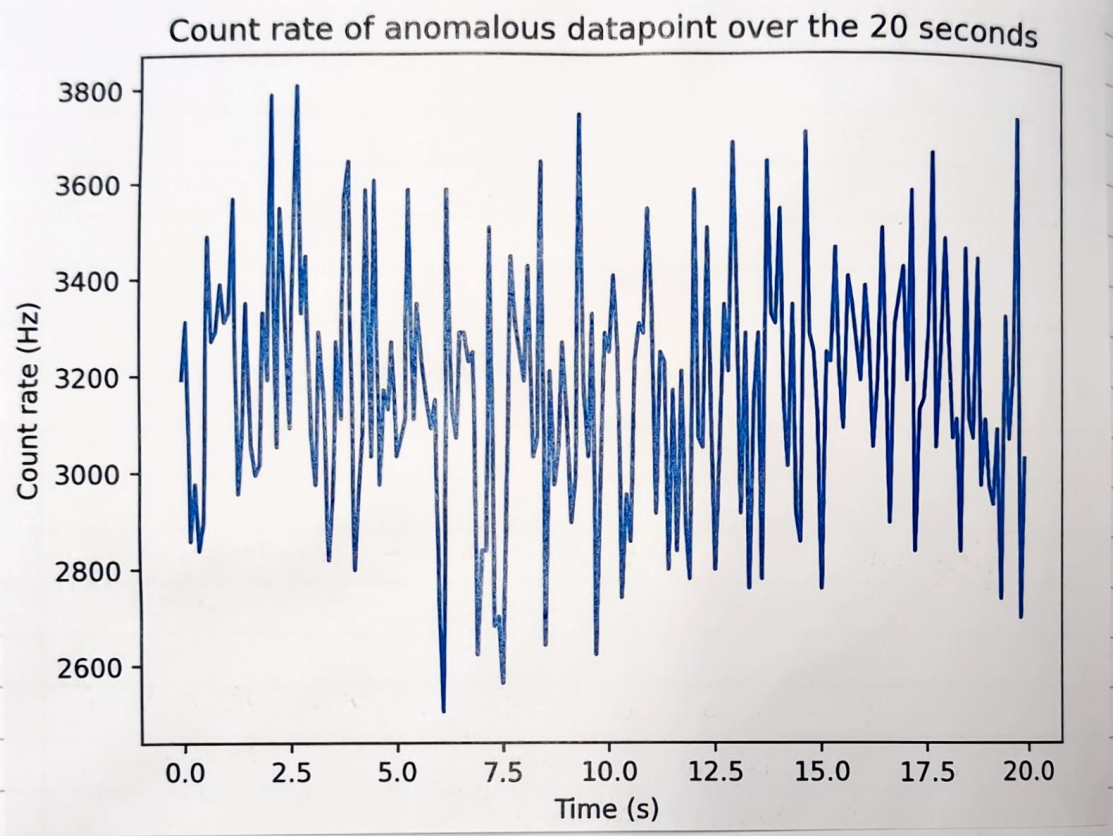
$\langle \sigma \rangle = 187 \pm 63$ Hz

With the screen on, Channel 1 background count is roughly modelled by $N(\mu = 3072, \sigma = 187)$.

The equivalent screen off data gives:

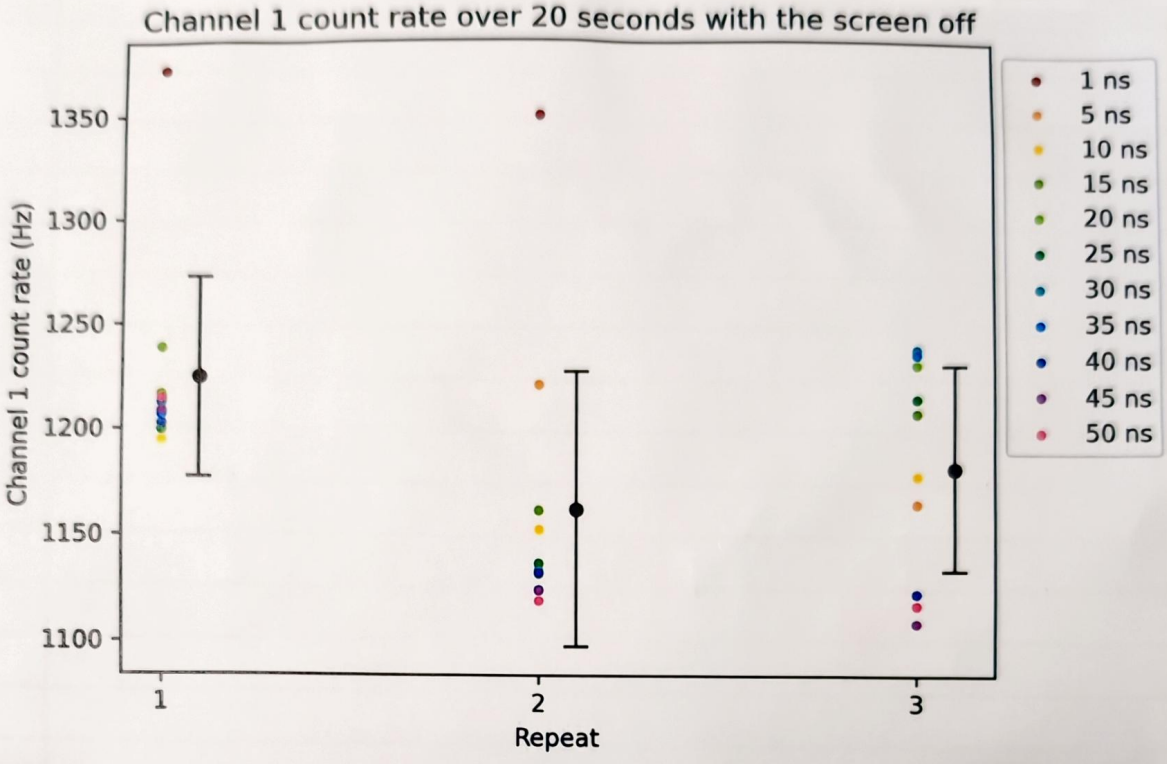


The 1st data point of the 3rd repeat is clearly very off, perhaps due to some other lighting from a phone screen? It is just below what we expect for the screen on data, so it would need to be something really bright for a short time or something as bright as the screen over the whole thing. Looking at the anomalous plot (with count window of 0.1s):

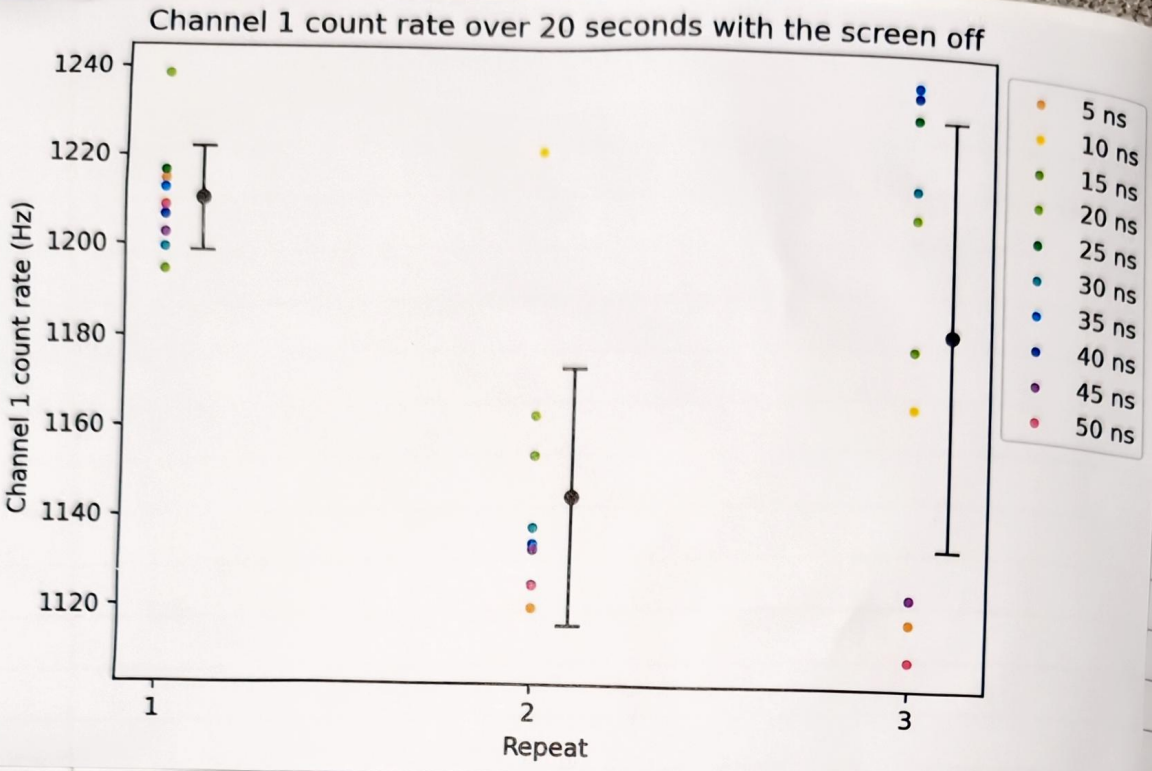


Something was consistently screwing with the dataset, however we can afford to discard this dataset as we have 32 others.

So, rejecting the anomaly: $\langle \mu \rangle = 1190 \pm 15\,Hz$ $\langle \sigma \rangle = 53.8\,Hz$ $\times 4.6$



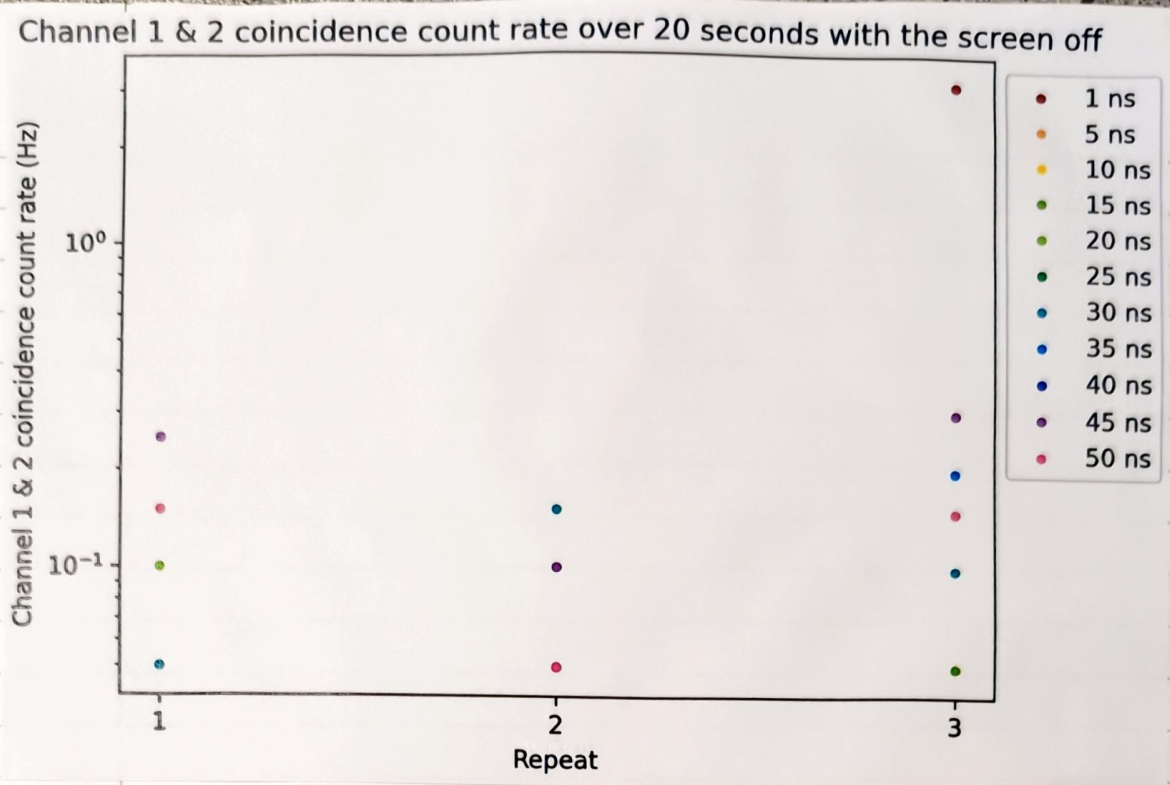
There seems to be a consistent problem with the 1 ns datasets. It could be that less counts are being counted as coincidences, boosting individual channel counts? Though I wouldn't expect the software to be coded that way. We can afford to discard the 1 ns datasets as anomalous, which seems to be the case for only the screen off case.



$\langle \mu \rangle = 1179 \pm 16\,Hz$ $\langle \sigma \rangle = 29.5^{+8.7}_{-} Hz$

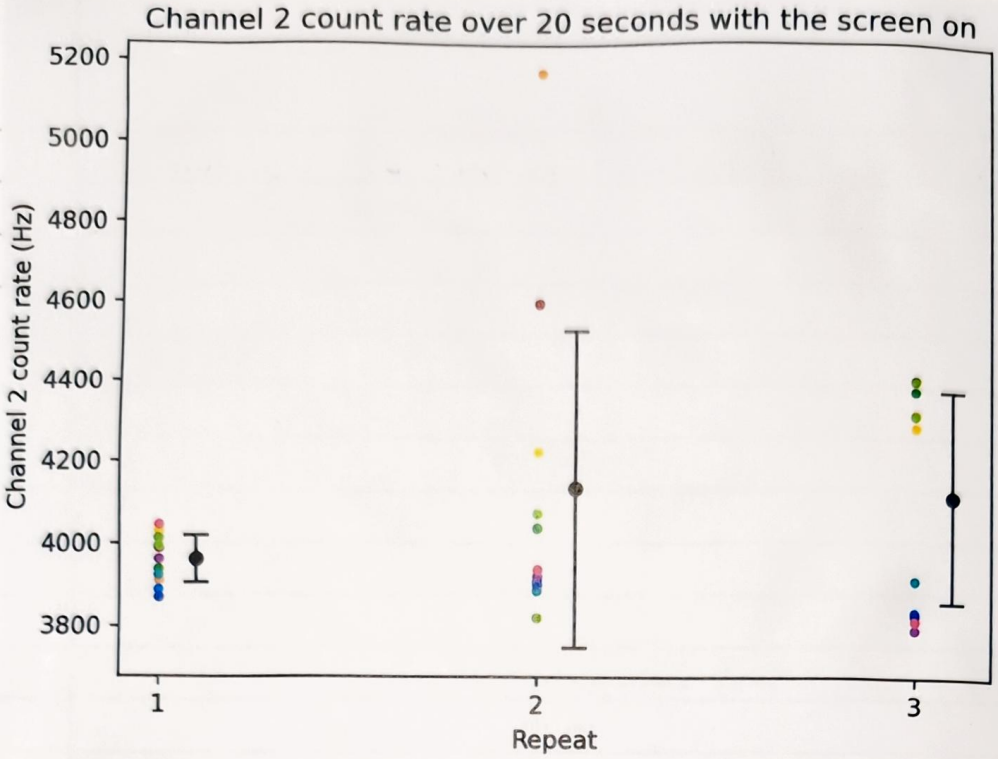
There also some weird spreading going on too across repeats. I think it might be due to impatience on our part during the data taking. e would over avoid which could've changed the reflection of light coming through gaps in the door.

To test the ~~[illegible]~~ theory on the previous page, we can graph the coincidence rates between channel 1 + 2. We should expect the coincidence rates for 1ns to be a low as for them to increase with window size. abnormally

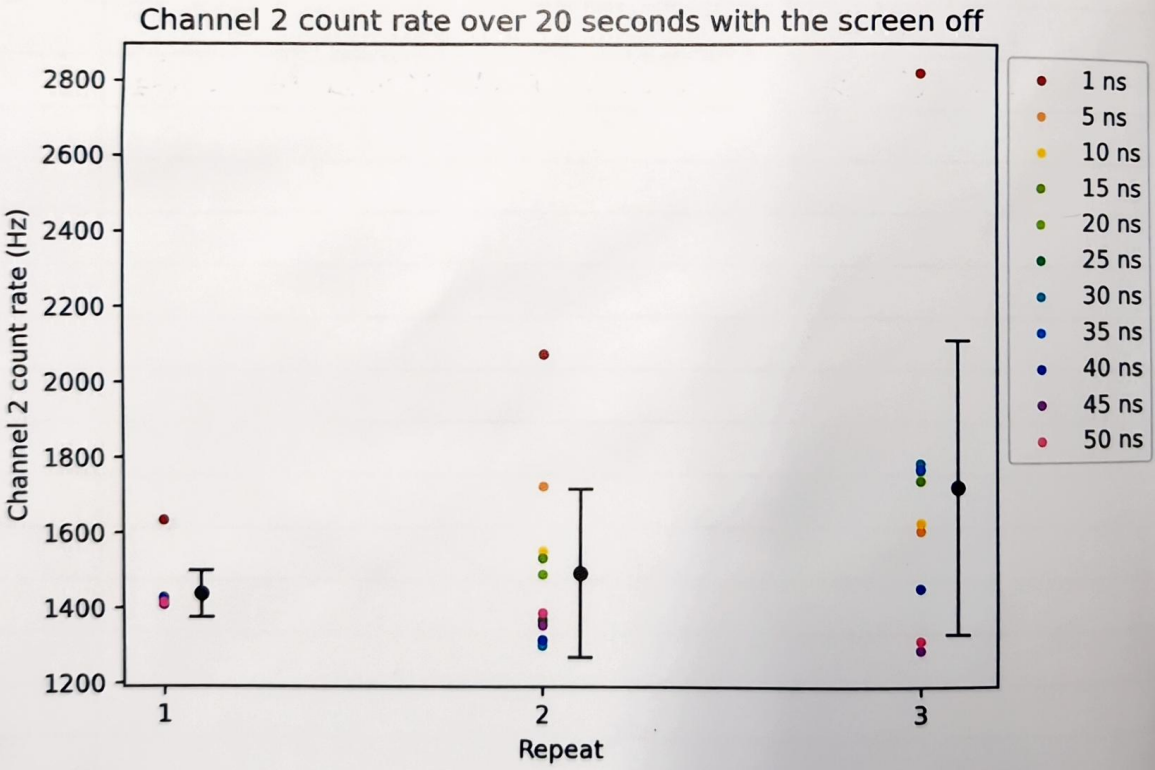


We see that the opposite actually takes place in repeat 3, where the anomaly was most prominent.
Note that the data points lie on exactly the same because we have so little counts that it is effectively discretised.
This disproves the previous theory. I have no idea what might happened.

Anyway, plotting channel 2:

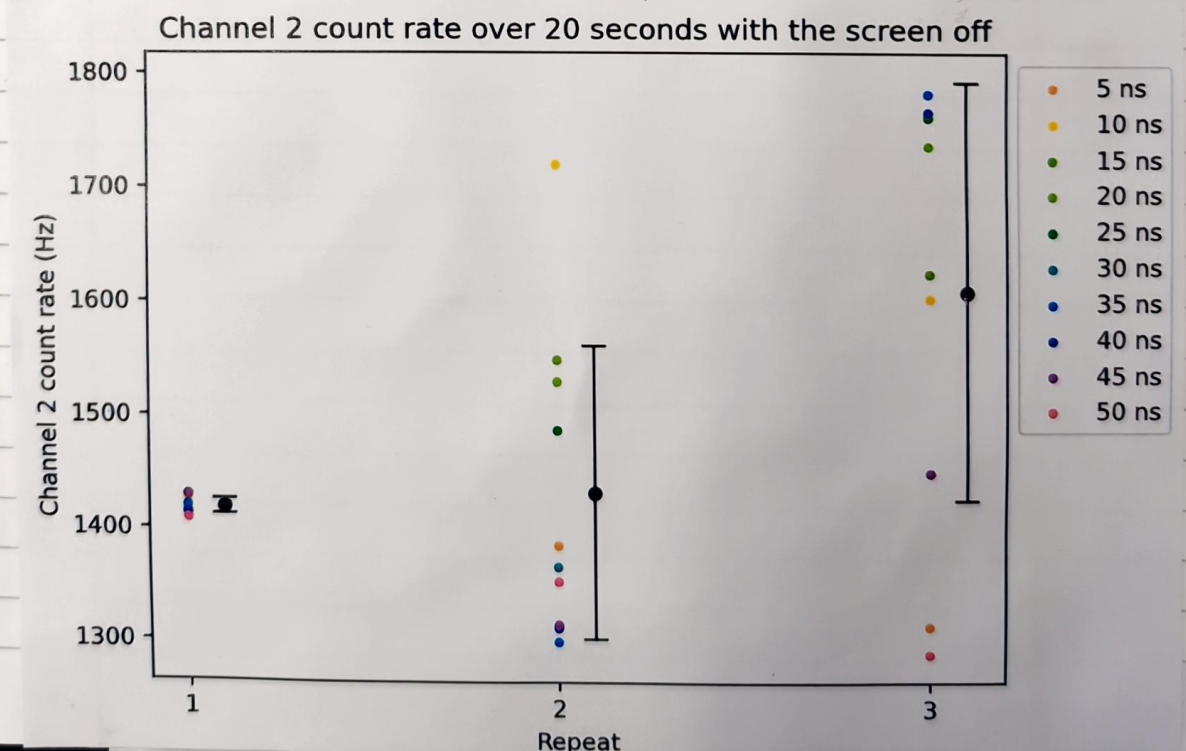


$\langle \mu \rangle = 4071 \pm 45$ Hz $\langle \sigma \rangle = 234 \pm 45$ Hz

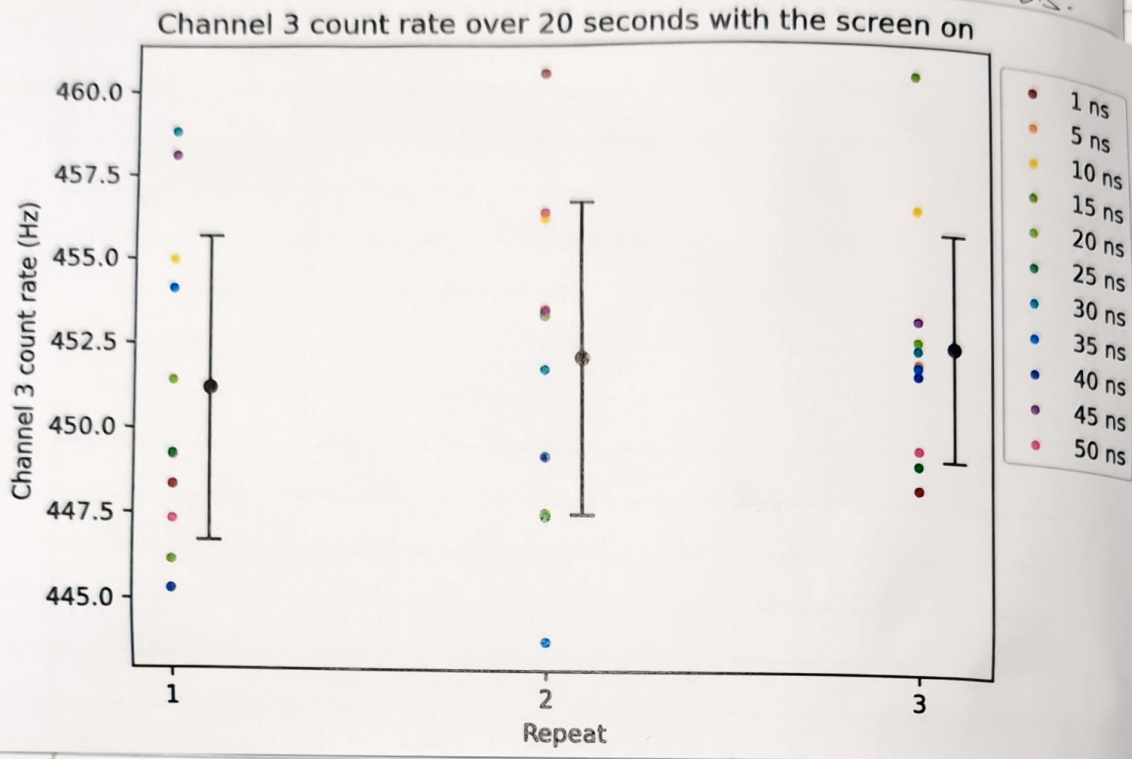


$\langle \mu \rangle = 1549 \pm 70$ Hz $\langle \sigma \rangle = 224 \pm 77$ Hz

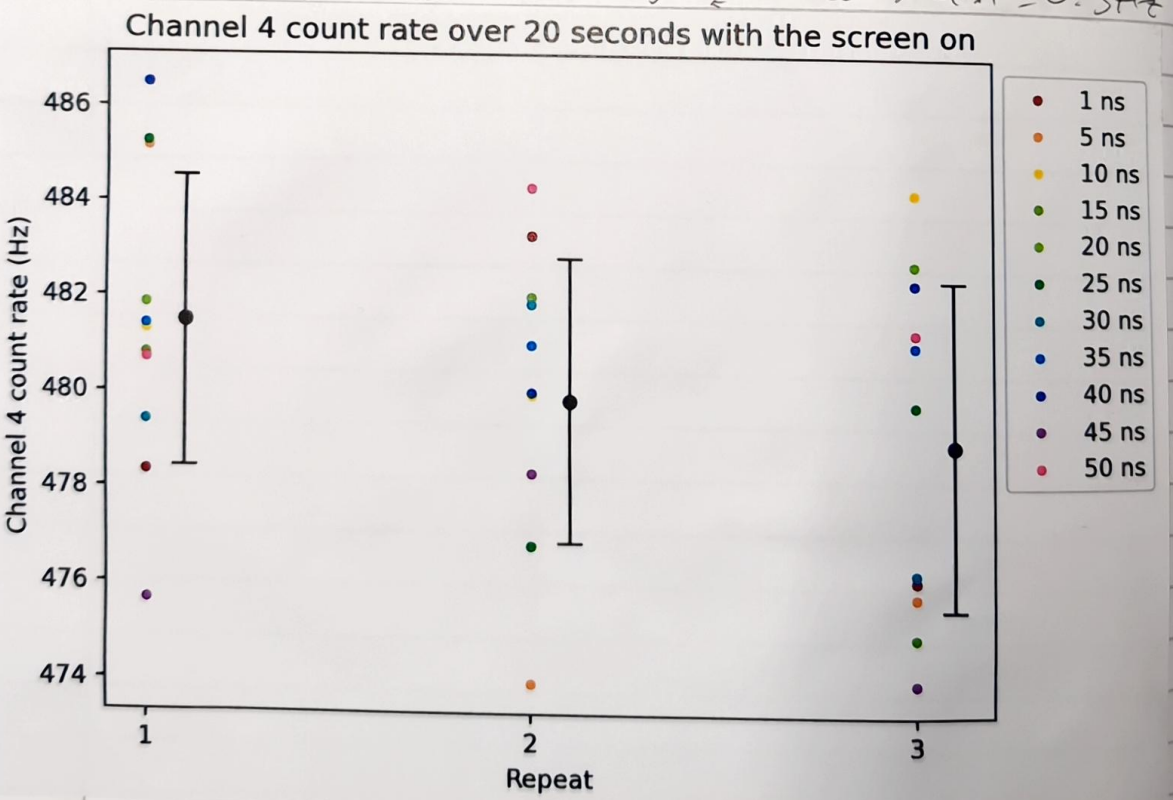
The anomalous 7ns datasets have reappeared in channel 2!
When I was doing the screen off datasets, I was unable to tell when the data-taking program was done. So, because I ran it, I put a stopwatch on my phone for 5 min so that when it went off, I'd know the data taking was done. The 7ns dataset was the first to be taken, so perhaps my phone being on affected the 7ns before I turned it off. I was trying to be quick on turning my phone off, but maybe I wasn't quick enough.
However this isn't represented in the count rate plot over the 20 seconds. But I think that maybe this plot should be discarded. I think this because there are more counts registered than timestamps. Specifically, the fpe-to-digital converter reported 63599 channel 1 counts in 20 seconds, but only saved 32263 channel 1 timestamps. I believe this is because the timestamps are stored in a buffer which probably reset during the 20 seconds, which case we'd lose the data for the beginning of the 20 seconds. I believe the timestamp phone screen data was lost to the buffer reset. $\pm 63^{+\infty}_{-\infty}$
Removing the anomalous 7ns datasets: $\langle \mu \rangle = 1486 \pm 50$ Hz $\langle \sigma \rangle = 106$



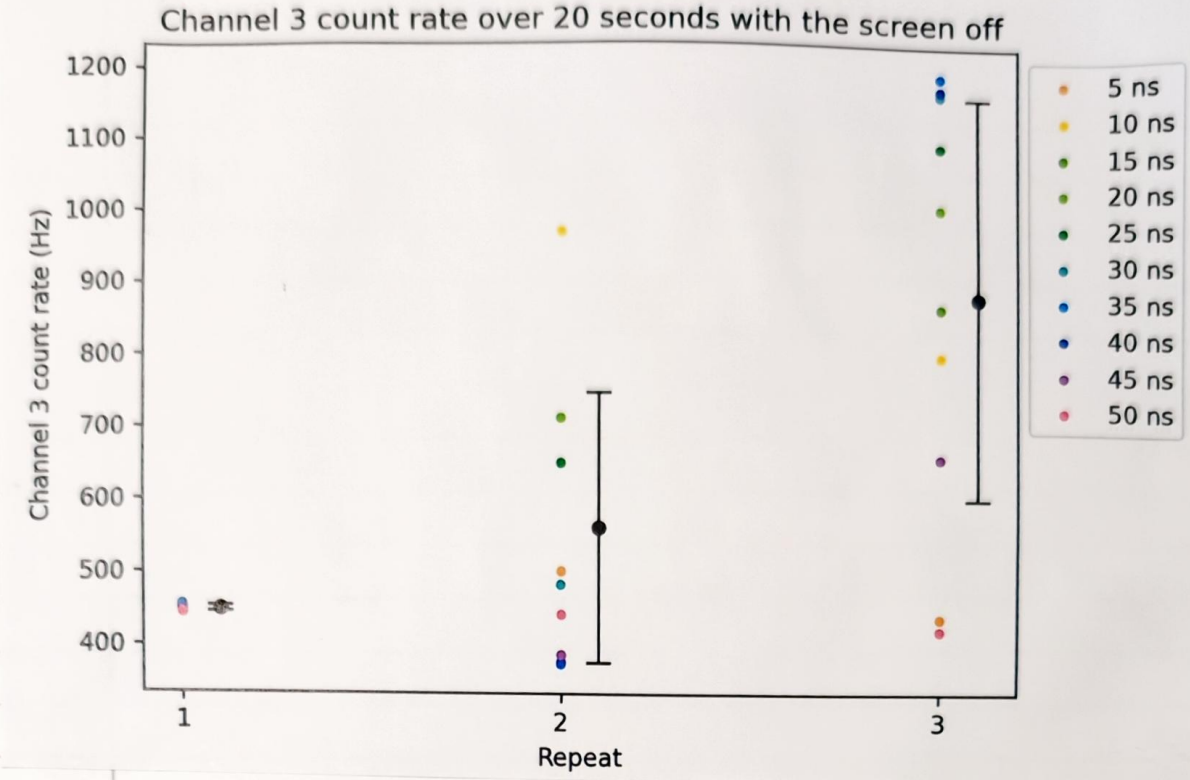
Channel 3 and 4 had ups on so they have similar counts.



$\langle \mu \rangle = 451.9 \pm 0.3$ Hz $\langle \sigma \rangle$ 4.1 ± 0.3 Hz

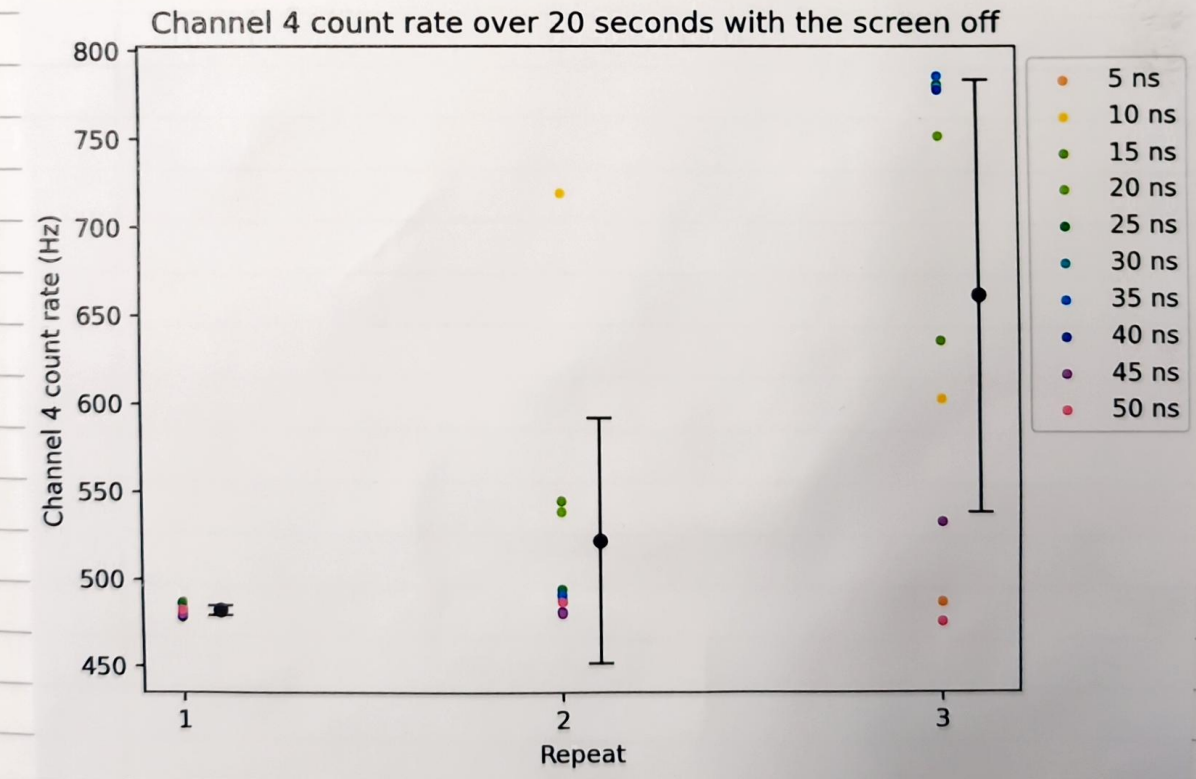


$\langle \mu \rangle = 480.1 \pm 0.6$ Hz $\langle \sigma \rangle = 3.2 \pm 0.1$ Hz



$\langle \mu \rangle = 630 \pm 105$ Hz $\langle \sigma \rangle = 156 \pm 66$ Hz

Repeat 1: $\mu = 469.82$ $\sigma = 4.1$



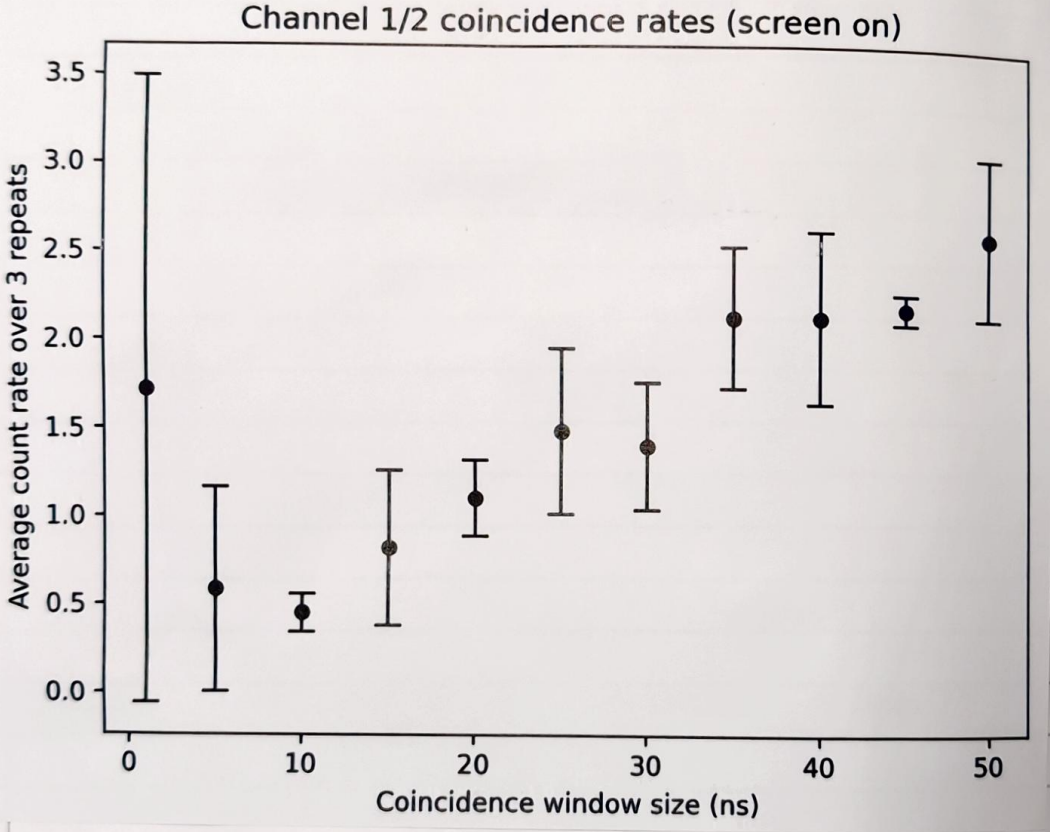
$\langle \mu \rangle = 555 \pm 44$ Hz $\langle \sigma \rangle = 65 \pm 28$ Hz

Repeat 1: $\mu = 482.215$ $\sigma = 2.849$

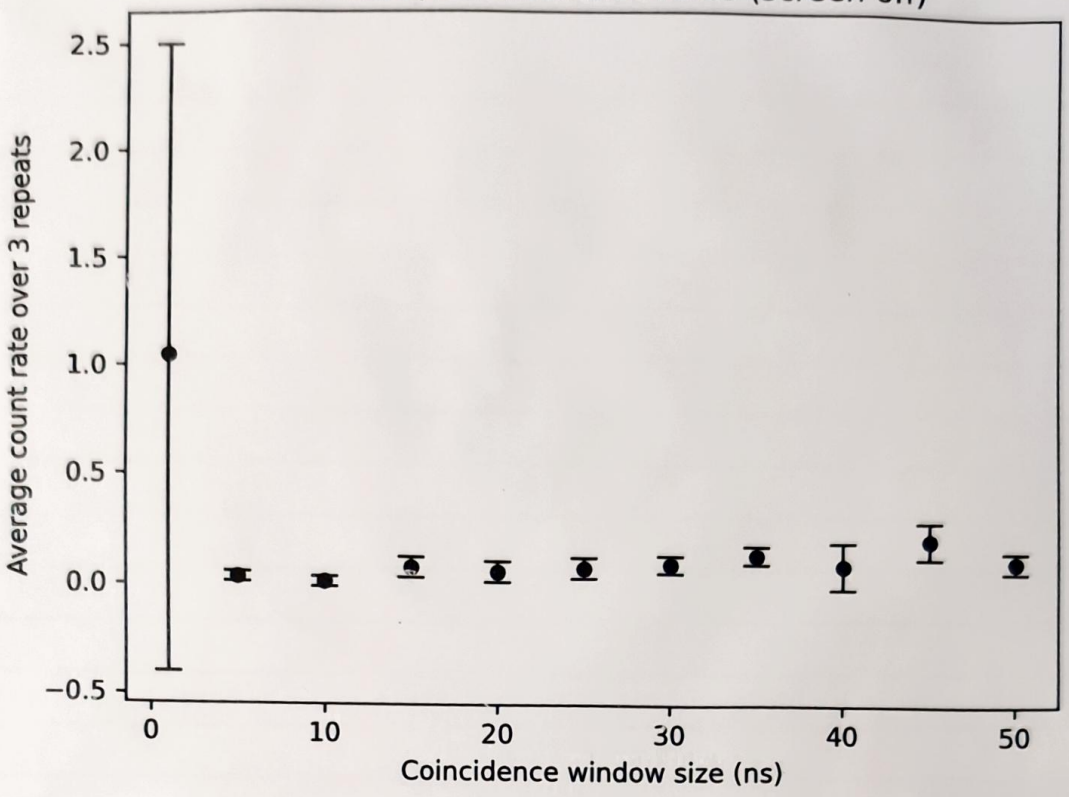
Something weird is going on with the 2nd and 3rd repeats of the screen off data. These are the ones taken the day after all the others, so it's likely something changed. Maybe the caps on the sensors were removed due to ~~realigning~~ Alex maybe doing some realigning.

The last two repeats are obviously wrong, so I will only use data repeat 1. So about 400 counts/second are from photons that pass through the cap.

Looking at coincidence rates of channels 1+2 over different windows.

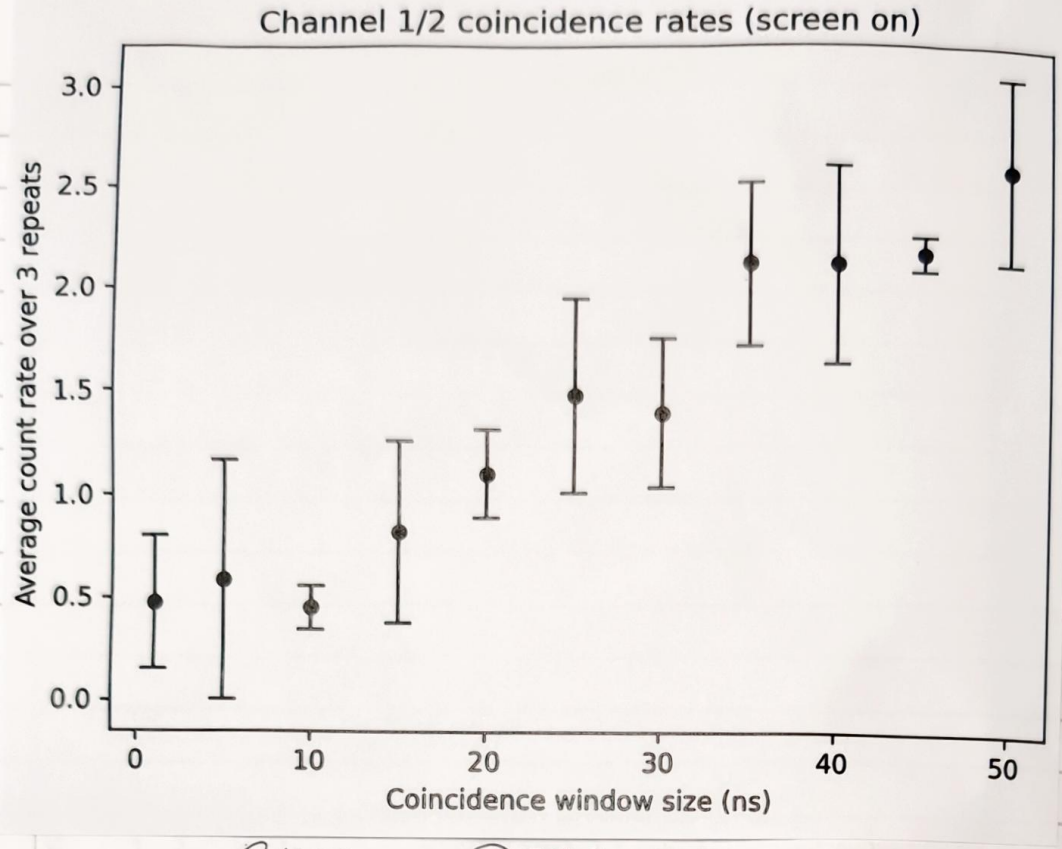


Total 7ns window counts	repeat 1	repeat 2	repeat 3
	3	84	16

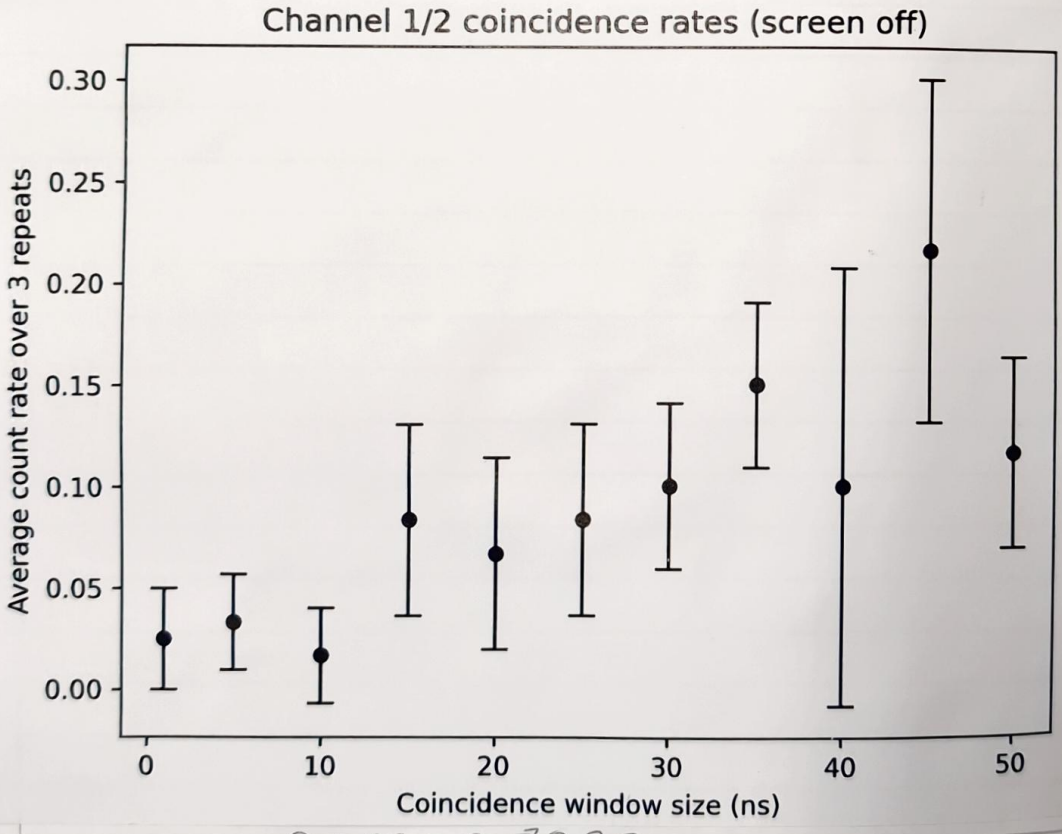


Total 1 ns window counts	repeat 1	repeat 2	repeat 3
	0	1	62

Again some weirdness here consistent with the phase idea in the 1 ns dataset. I'll discard the obvious anomalous counts of 84 and 62.



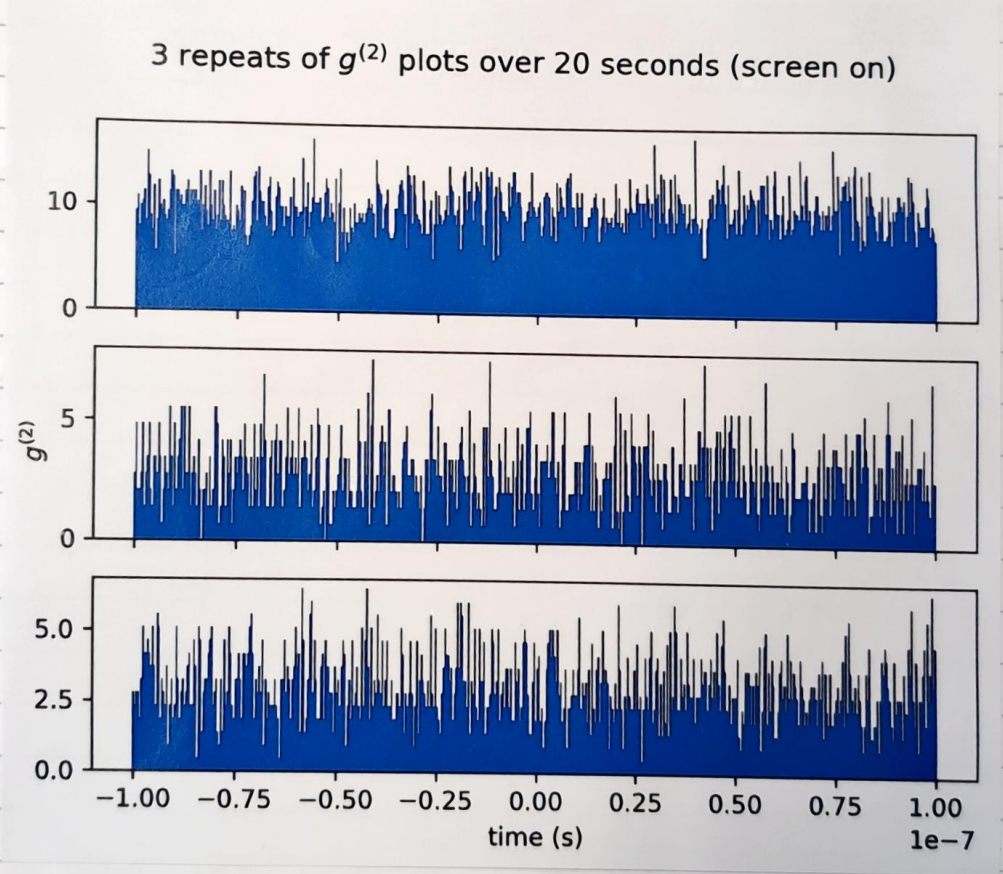
PMCC = ~~0.8323~~ 0.9757

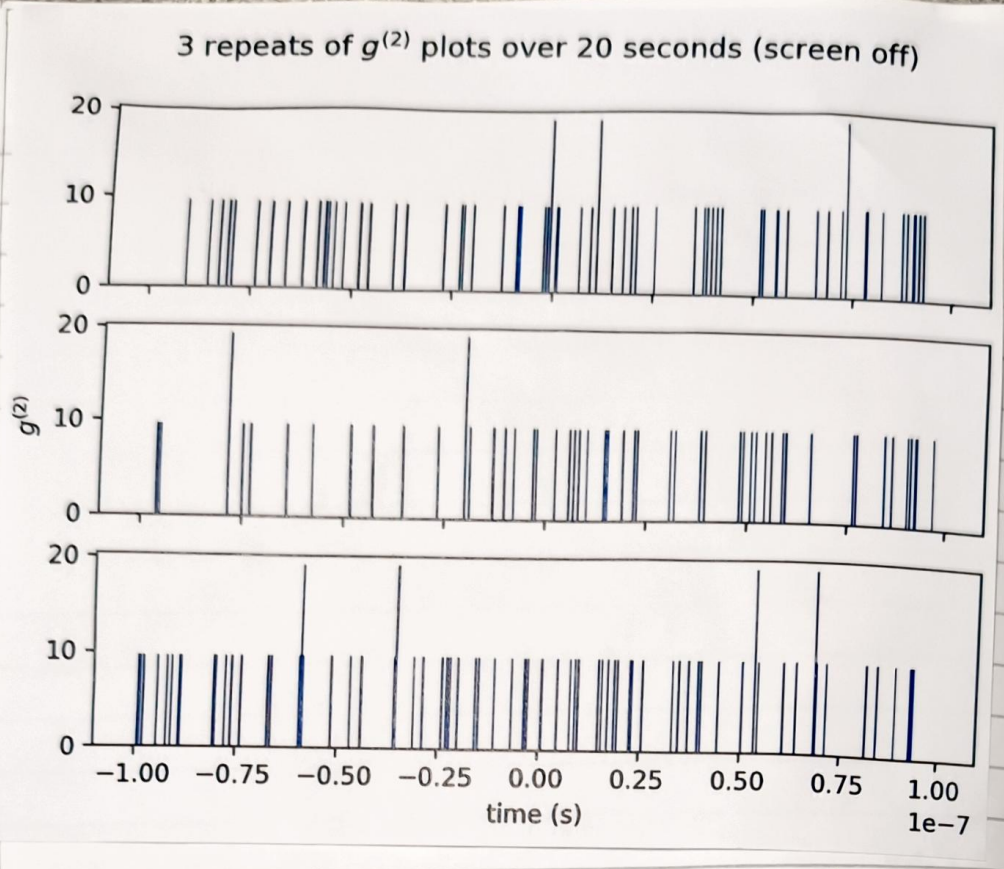


PMCC = 0.8323

As expected, there is a positive correlation between the coincidence window size and the coincidence count rates (as evidenced by the PMCC coefficients). Thus when we do the experiment, we should choose ~~a~~ a large enough to observe coincidences, but not too large to overcount them.

We also decided to save the background $g^{(2)}$ data, however these graphs aren't very useful since they aren't normalized yet and we cannot really "subtract" the background $g^{(2)}$ from the experiment





These $g^{(2)}$ plots are really just us getting used to the software

The $g^{(2)}$ graphs have two parameters: bin count and bin size. These graphs are effectively histograms.
Note that when you input the bin count, the actual number of bins in the histogram is $2 \times \text{binCount} - 1$. From here onwards, I will quote $2 \times \text{binCount} - 1$ as the true bin count.

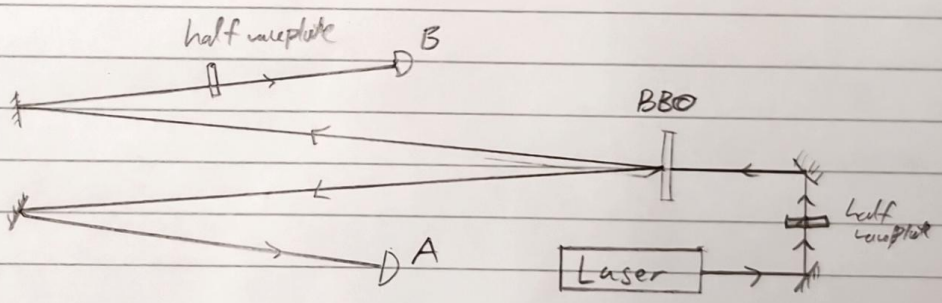
2/5/2025 ~9am

Experiment 1

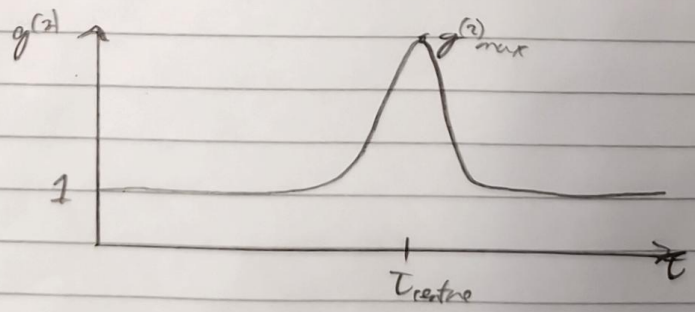
The alignment problems have returned, roughly half our counts are from background light. Eventually gone through the effort of tipping over any bright lights that may increase the noise in our experiment.

~10:30am Alex has managed to align the setup so that we have roughly 30 kHz of counts in each detector. A and B. (optics shown below) To take the $g^{(2)}$ data, we set up the hotkey so that we can trigger the software to record while the screen is off, and then quickly leave the room. Once we've waited long enough, we quickly enter the room and use a hotkey to save the data before too much noise (from the open door) affects the dataset.

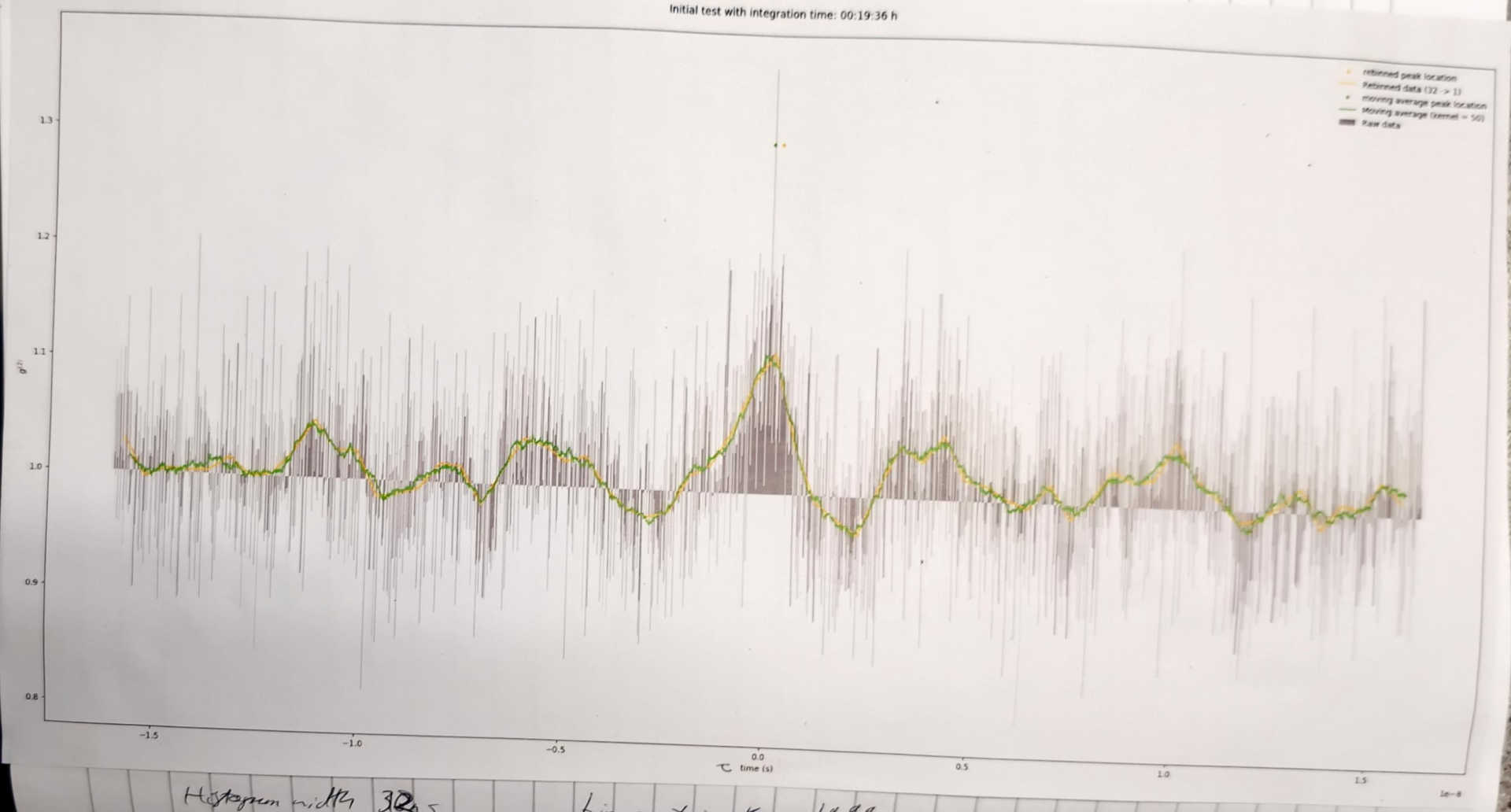
We tested this method on an initial dataset (PTO)



If the dem coincidence works properly, we expect a peak in the $g^{(2)}$ graph near zero. A higher peak means that we obtain more coincidences which means the ~~Raman~~ detector we picking up more dem-correlated photons.



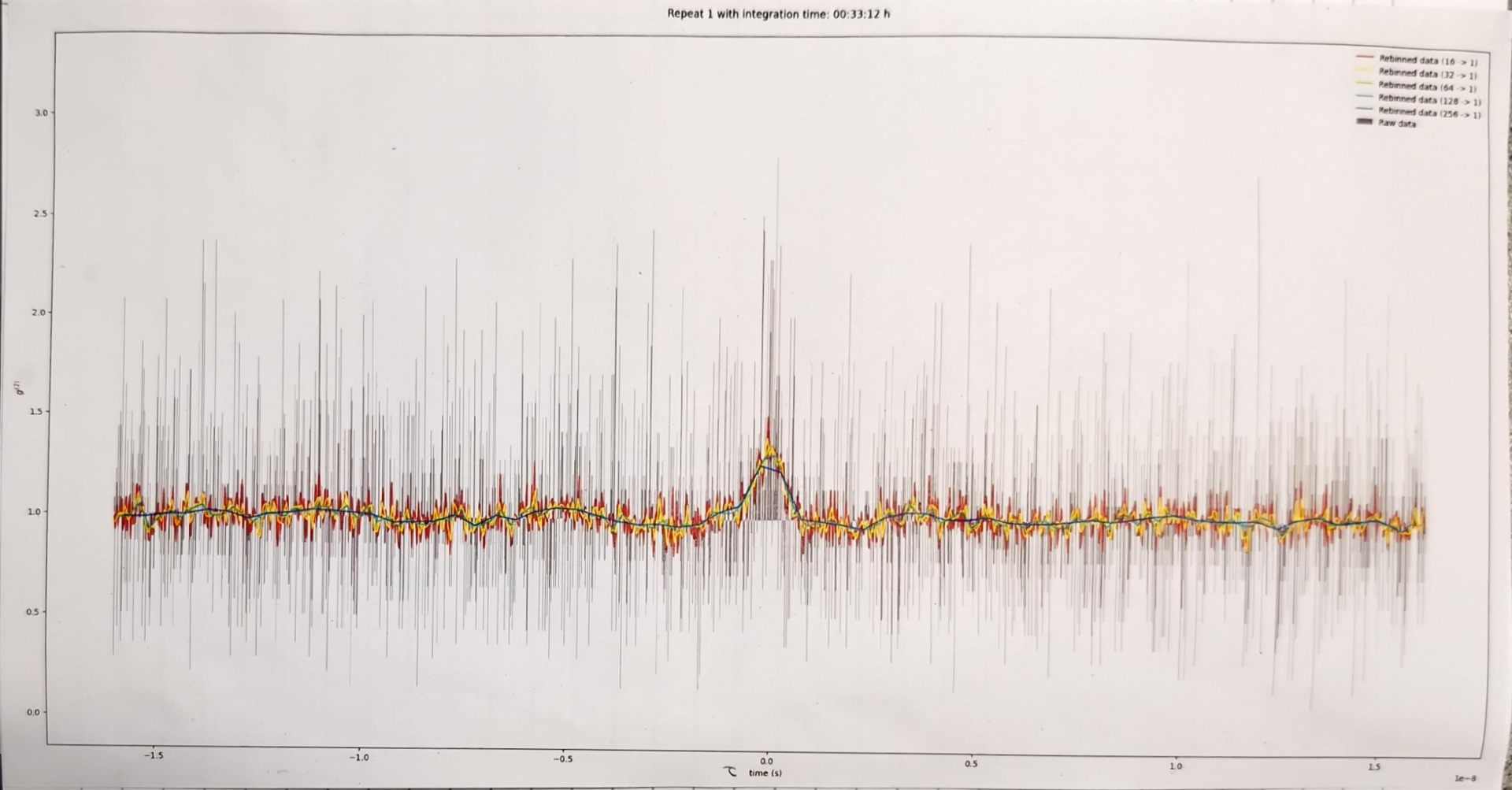
$\tau_{centre} \simeq 0$



Histogram width 3[illegible]5 bin count: [illegible]

The raw data is clearly very noisy, and there are two ways that I've come up with to process it.
The first method is to evaluate a moving average. A moving average with kernel size of 50 bins (about 8×10^{-10} seconds) yields a peak at $\tau = -7.21 \times 10^{-11}$ s, $g^{(2)}_{max} = 1.179$.
The second method takes advantage of the fact that the $g^{(2)}$ graph is effectively a histogram, by re-binning the histogram. This is done by combining consecutive bins by average so as to conserve normalization. Re-binning 32 bins to 1 yields a peak at $\tau = 1.36 \times 10^{-12}$ s, $g^{(2)}_{max} = 1.122$.
These peaks are found with scipy's find_peaks() function.
Doing a moving average is not very physically representative as a binning, whereas re-binning is simply a change of resolution but a moving average is something more complicated. The re-binning makes more sense to use here. Besides, both methods give roughly the same peak, so the moving average should give roughly the same results anyways.

This dataset is too noisy to get anything useful from. There seems to also be some periodicity in the dataset, but could just be a consequence of noise.



Hydrogen width : 32ns

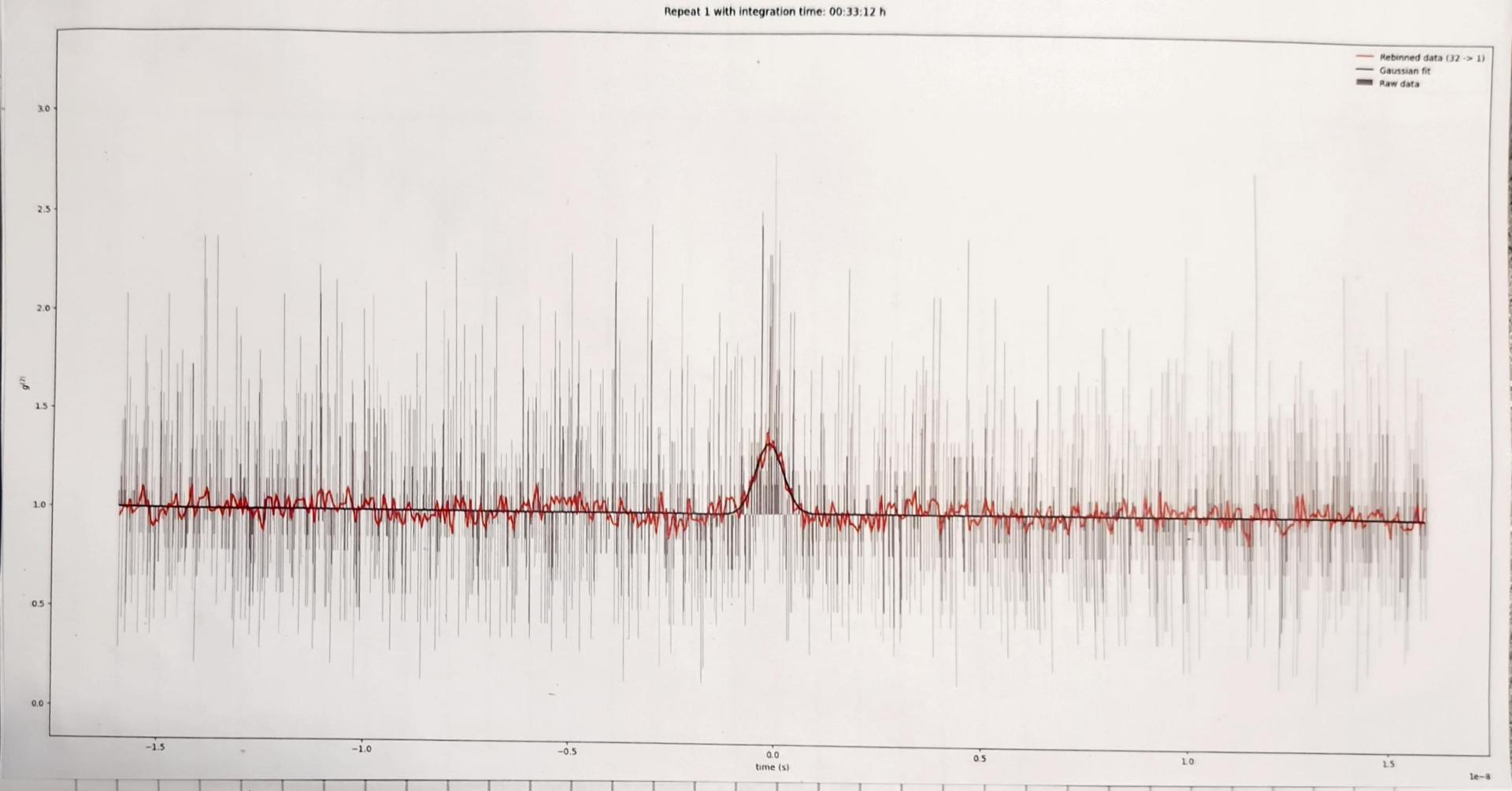
Bin count = 16383

~11am

Doing another dataset for half an hour yields slightly better results. We max out the bin count since it can be reduced in post. However the choice of choosing the Bin count now introduces a degree of freedom for the histogram ~~resolution~~ resolution which in turn affects the height of the $g^{(2)}$ peak.

Bin size (seconds)	$g^{(2)}$ max peak
0.031	1.518
0.062	1.413
~~0.249~~	~~1.359~~
0.125	1.369
0.250	1.321
0.500	1.273

The choice of rebinning affects the peak height.
From now on, the rebinning will be such that there are 511 bins after rebinning (bin width of 6.25×10^{-11} s).
I chose 511 because the histogram defaults to this upon startup so if any mistakes are made, we can still salvage the data. Also 1 → 32 seems like a nice balance between noise reduction and loss of precision.



Histogram width: 32 ns
Bin Count: 16383

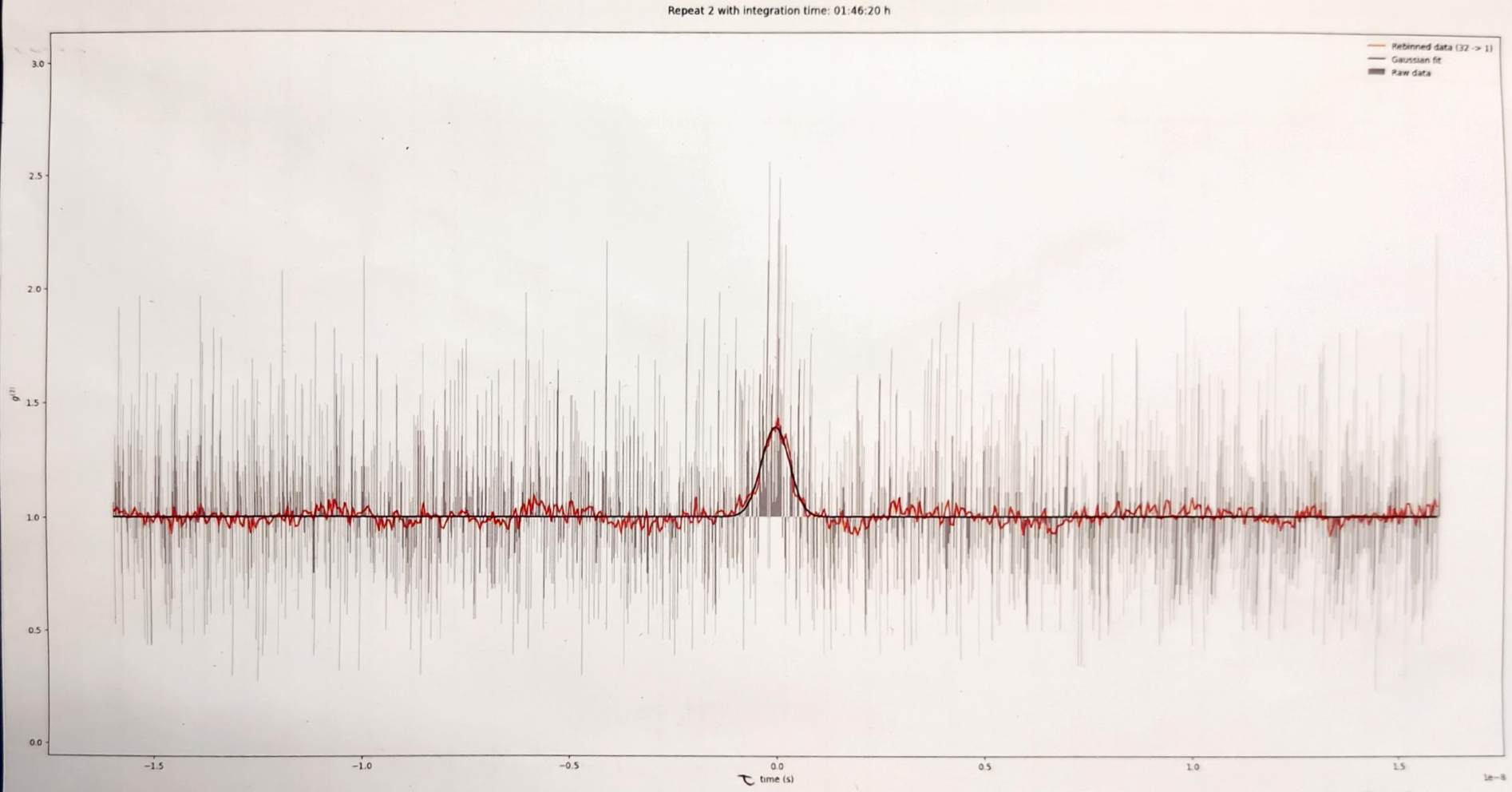
Retrieving this dataset and applying a gaussian fit gives:

$$g^{(2)}_{max} = 1.36 \pm 0.02$$
$$\mu = \tau_{center} = (-5.1 \pm 2.1) \times 10^{-11}\ s$$
$$\sigma = (4.7 \pm 0.3) \times 10^{-10}\ s$$

This gaussian fit does not work well with the raw data, which is why rebinning is necessary.

~12pm Done with the dataset with a larger integration time yielding better dataset.
Fitting a gaussian yields:

$$g^{(2)}_{max} = 1.40 \pm 0.01$$
$$\mu = t_{centre} = (-2.5 \pm 1.3) \times 10^{-11}\,s$$
$$\sigma = (4.8 \pm 0.2) \times 10^{-10}\,s$$



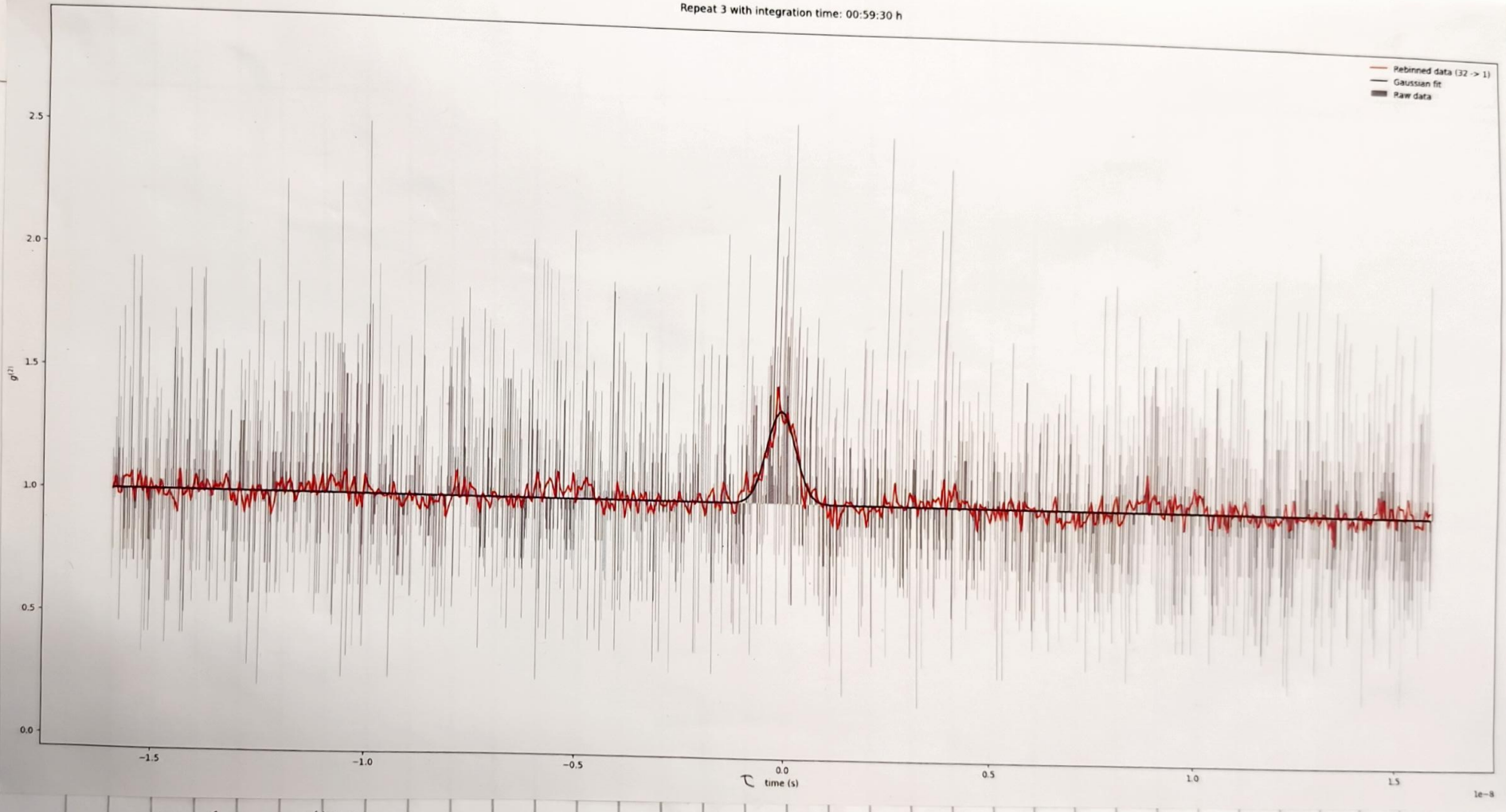
Histogram width: 32 ns
Bin cut : 16383

~2pm

One final run gives the following dataset:

Fitting a ~~gaussian~~ yields:

$$g^{(2)}_{max} = 1,38 \pm 0.02$$
$$\mu = t_{centre} = (-5.0 \pm 1.7) \times 10^{-11} s$$
$$\sigma = (4,9 \pm 0.2) \times 10^{-10} s$$

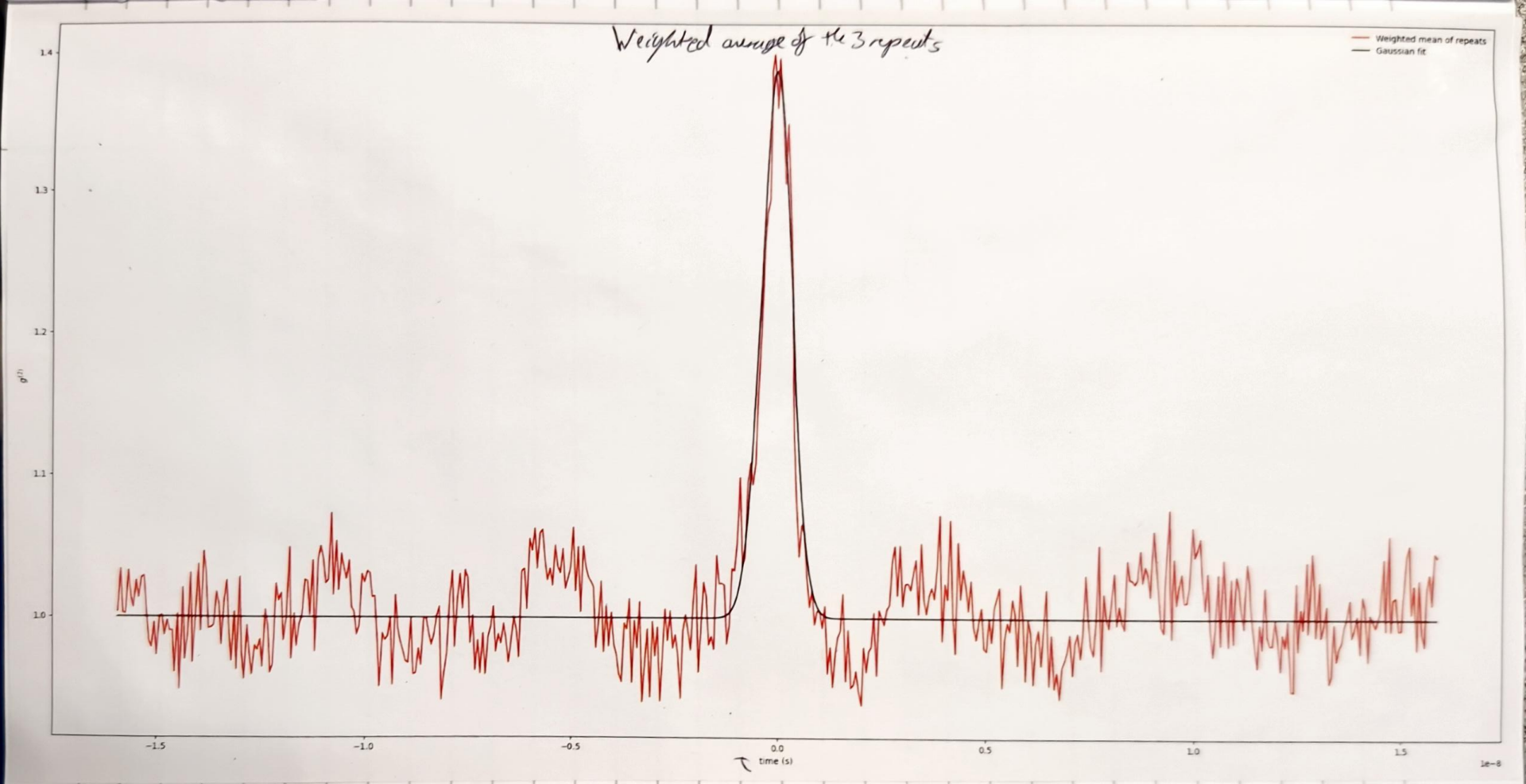


Histogram width: 32 ns
bin Count: 16383

By using all three obtained datasets, we can average and find an error on each datapoint. To combine the datasets while still considering the integration time, we take a weighted average. This gives us one big dataset that we can fit a gaussian to:

$$g^{(2)}_{max} = 1.396 \pm 0.006$$
$$\mu = \tau_{cable} = (-4.3 \pm 0.8) \times 10^{-11}\ s$$
$$\sigma = (5.17 \pm 0.09) \times 10^{-10}\ s$$
$$\chi^2_\nu = 6088 \qquad df = 508$$

This might look like a substantial χ^2_ν, but it is inflated by the strong periodic pattern appearing around the peak. This periodic pattern also appeared in the initial test dataset and seems to have a period of ~6ns. This corresponds to roughly how long it takes light to travel about 2m which is roughly the length that light travels from the BBO crystal to the fiber optic interface, so it's likely there are reflections bouncing off the fiber optic interface, through the crystal and into the next fiber optic interface.


Weighted average of the 3 repeats
Weighted mean of repeats
Gaussian fit
1.4
1.3
1.2
1.1
1.0
$g^{(2)}$
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
time (s)
1e−8

~4pm

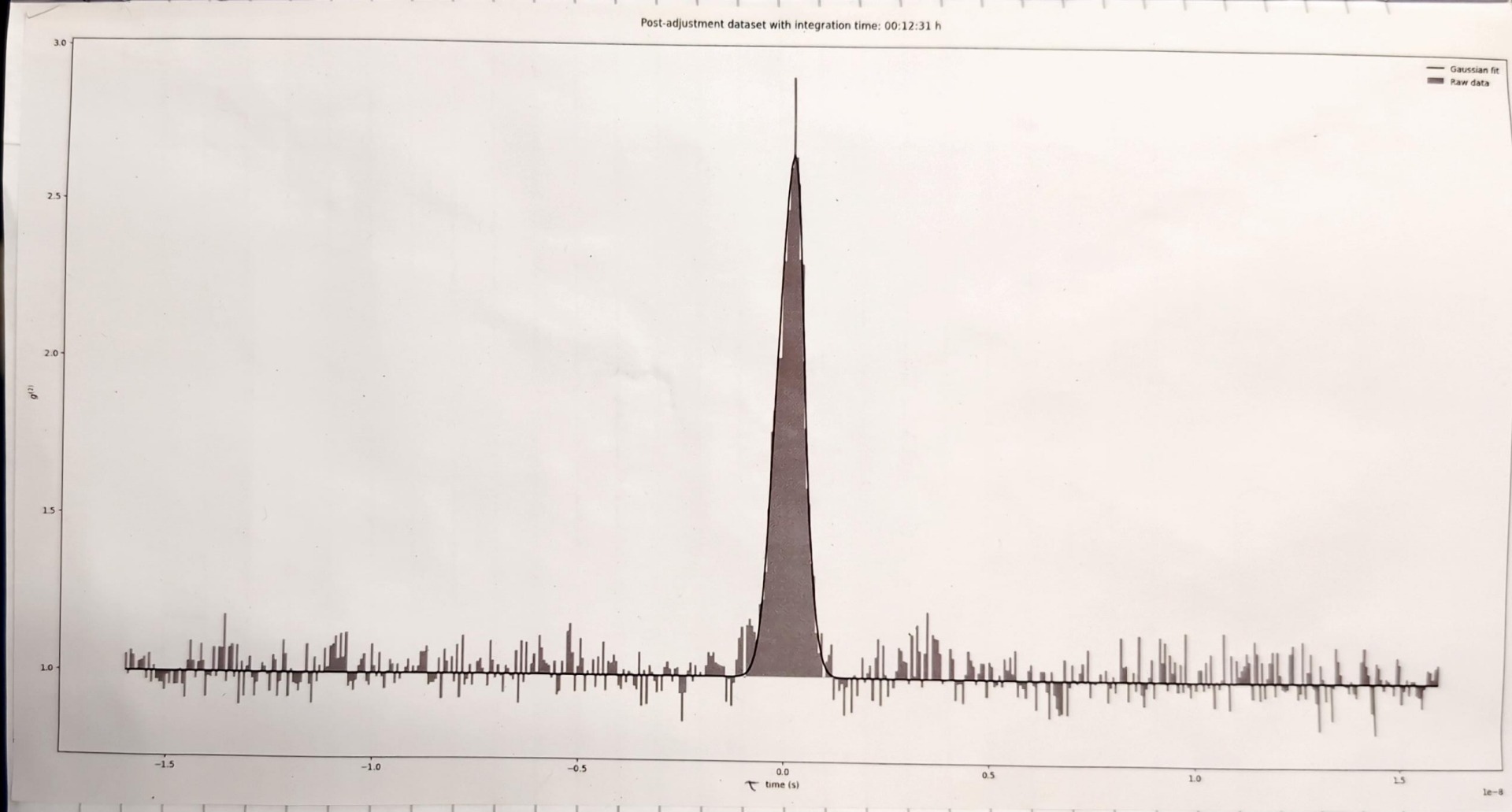
With some adjustments to the setup we successfully managed to increase the count rate of detector A to ~110 kHz and B to ~40 kHz. With this, we obtain the following dataset. We fitted a gaussian to the raw data given:

$$g^{(2)}_{max} = 2.66 \pm 0.03$$
$$\mu = \tau_{cable} = (1.28 \pm 0.06) \times 10^{-10}\ s$$
$$\sigma = (4.62 \pm 0.08) \times 10^{-10}\ s$$

With the maximum of the raw data:

$$\max[g^{(2)}] = 2.912745$$

which is significantly better than the previous alignment!



Histogram width = 32ns
bin count = 511

6/5/2025

~10am

With some further alignment, we managed to get the counts all the way up to ~200 kHz for both A and B! To do this fine alignment, we needed to have two dials at the same time to accommodate for both the absolute rotation of both the mirrors. With these counts, we obtain the following dataset.

Fitting a gaussian gives:

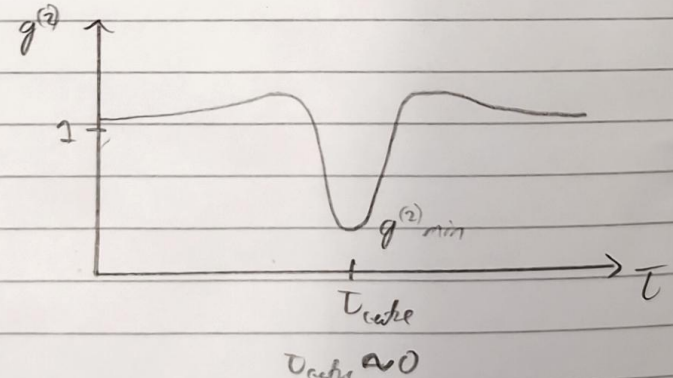
$$g^{(2)}_{max} = 7.81 \pm 0.05$$
$$\mu = \tau_{center} = (5.5 \pm 0.3) \times 10^{-10}\,s$$
$$\sigma = (4.2 \pm 0.4) \times 10^{-10}\,s$$

Significantly better than even the last alignment!

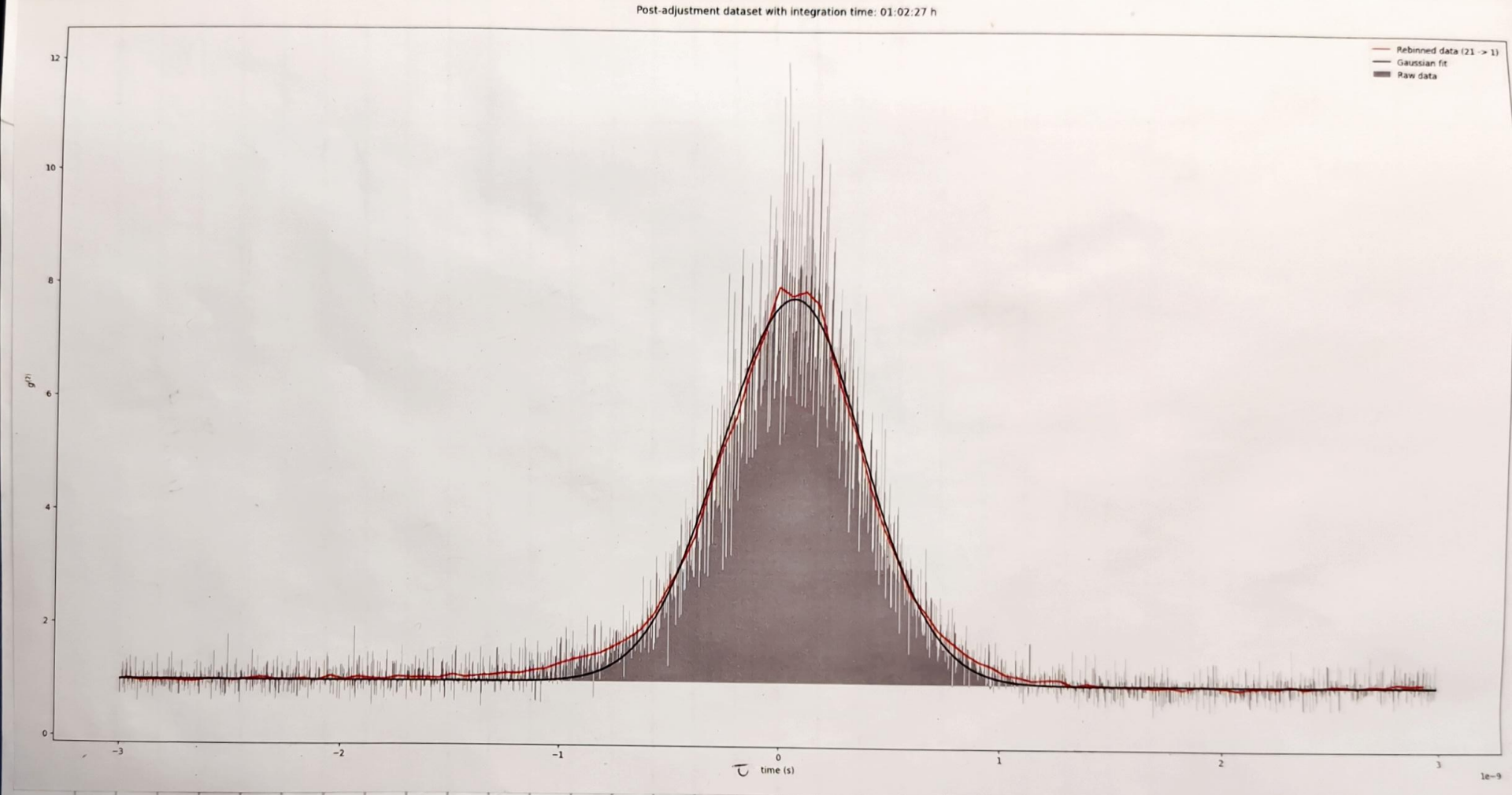
Experiment 2

~12pm ~~We need to measure~~ Now that we have confirmed that the dim conversion is functional, we wish to extract individual photons. This can be done by reintroducing the beam splitter and reproducing the original setup.

If we measure the $g^{(2)}$ graph between B and B', we should expect to see a dip in the distribution since the single photon would have been detected ~~and~~ so the other detector should measure (ideally) nothing.



$\tau_{center} \approx 0$



Histogram width = 6ns
bin count = 2047

Rebinning s.t. bin width = 6.25×10^{-11} s like others

~2pm We're having trouble getting the counts up on detector 3. To align detector 3, we used a laser through the fiber optic interface and aligned it with the path of detector 2's fiber optic interface by making the laser hit the same points from either detector's fiber optic interface.
This is pretty hard to do because there aren't enough degrees of freedom to align it easily.

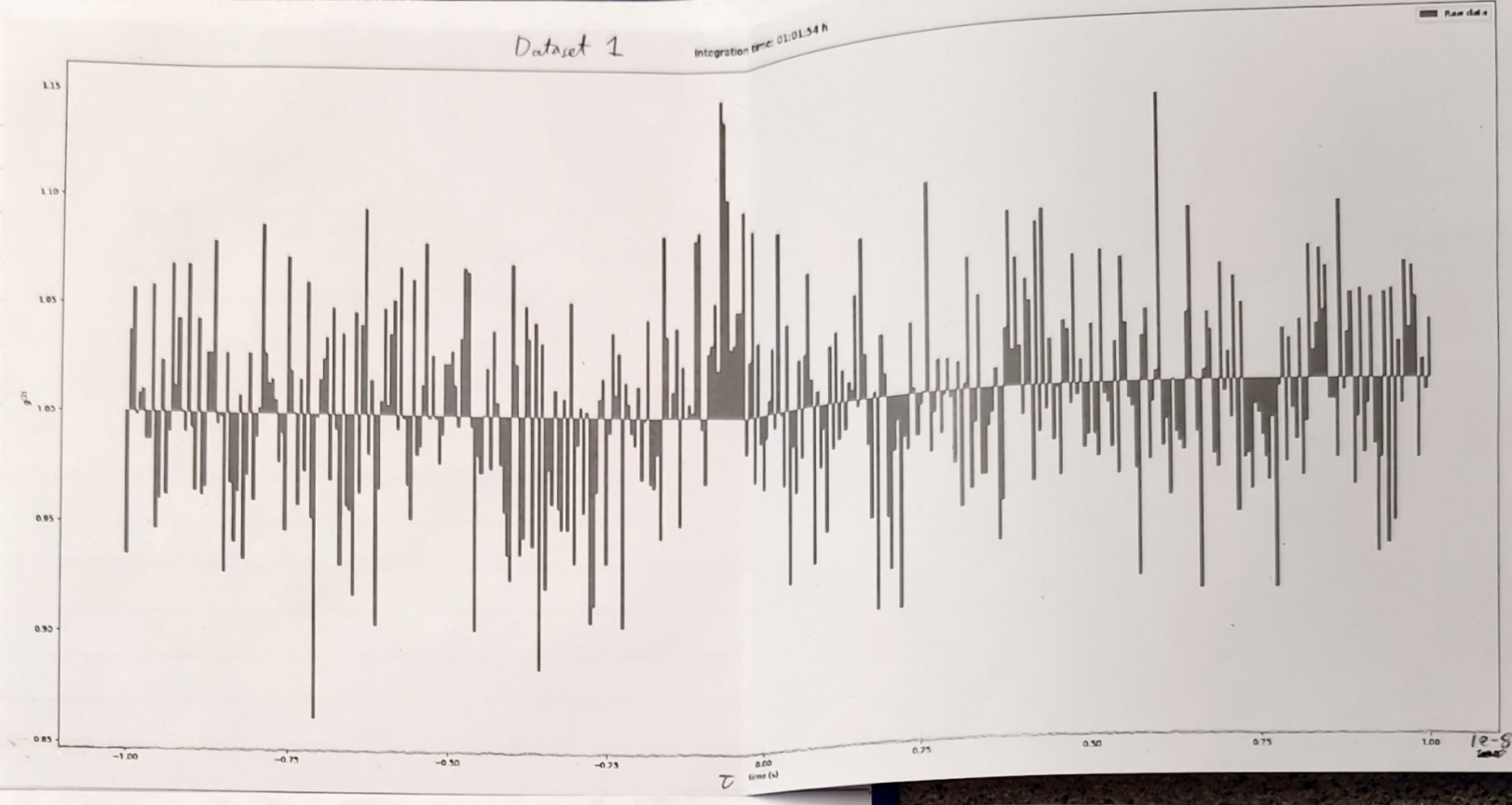
7/5/2025
~9am We didn't manage to get anything but background counts on detector 3 yesterday, so will try again today.

We've noticed that the irises used to align 3° for the ray to the beam splitter from the cavity are not quite aligned, probably causing losses to both B and B'.

After adjusting the irises, we seem to be getting some counts on both B and B'. B has roughly 60kHz, but B' has roughly 8kHz count rates. As the time-averaged count rates, we decided to take a dataset (dataset 1).
~11am This dataset is mostly noise with (weakly) a subtle peak near zero, though this doesn't tell us anything. So we can try adjusting the stop more.
~1pm We haven't touched the half wave plate between the mirror and beam splitter because we thought it was necessary for the setup. But now we question if it's necessary since the B' counts increase when it is removed.
So assuming the half wave plate can make A to have 200 kHz and B and B' to have ~100 kHz count rates, which is exactly how we want it to be (since 2·100kHz = 200 kHz). We want to count only the photons ~~that~~ only a little A detects a photon, & we "herald" the photon at B or B' based on a detection at A. Ground wires Sadly we haven't managed to do this with software. We should hopefully still see something if we have it the way (dataset 2)

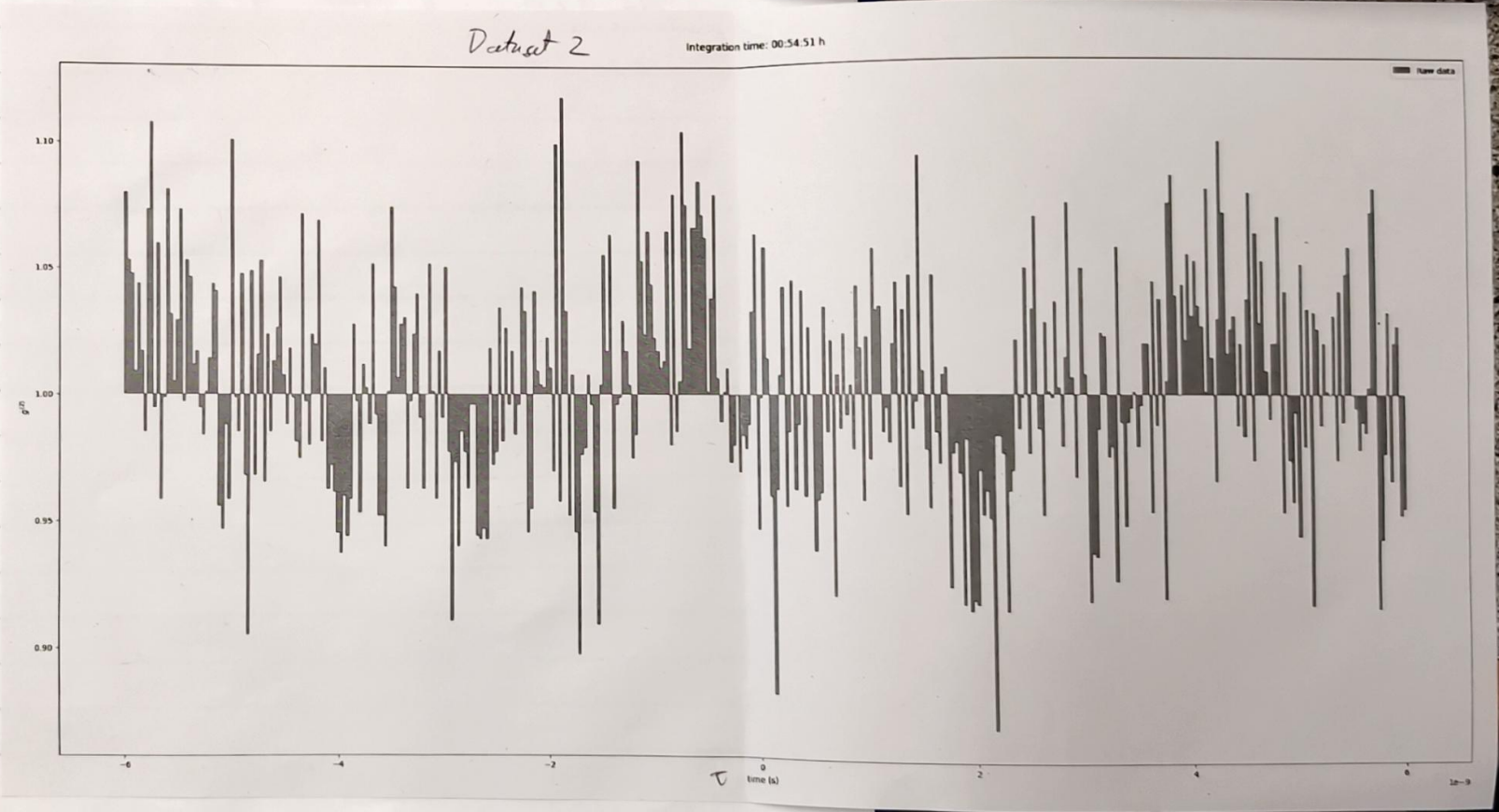
Bin size = 5.03×10^{-11} s

Bin count = 399



Bin size = 3.02×10^{-11} s

Bin count = 399



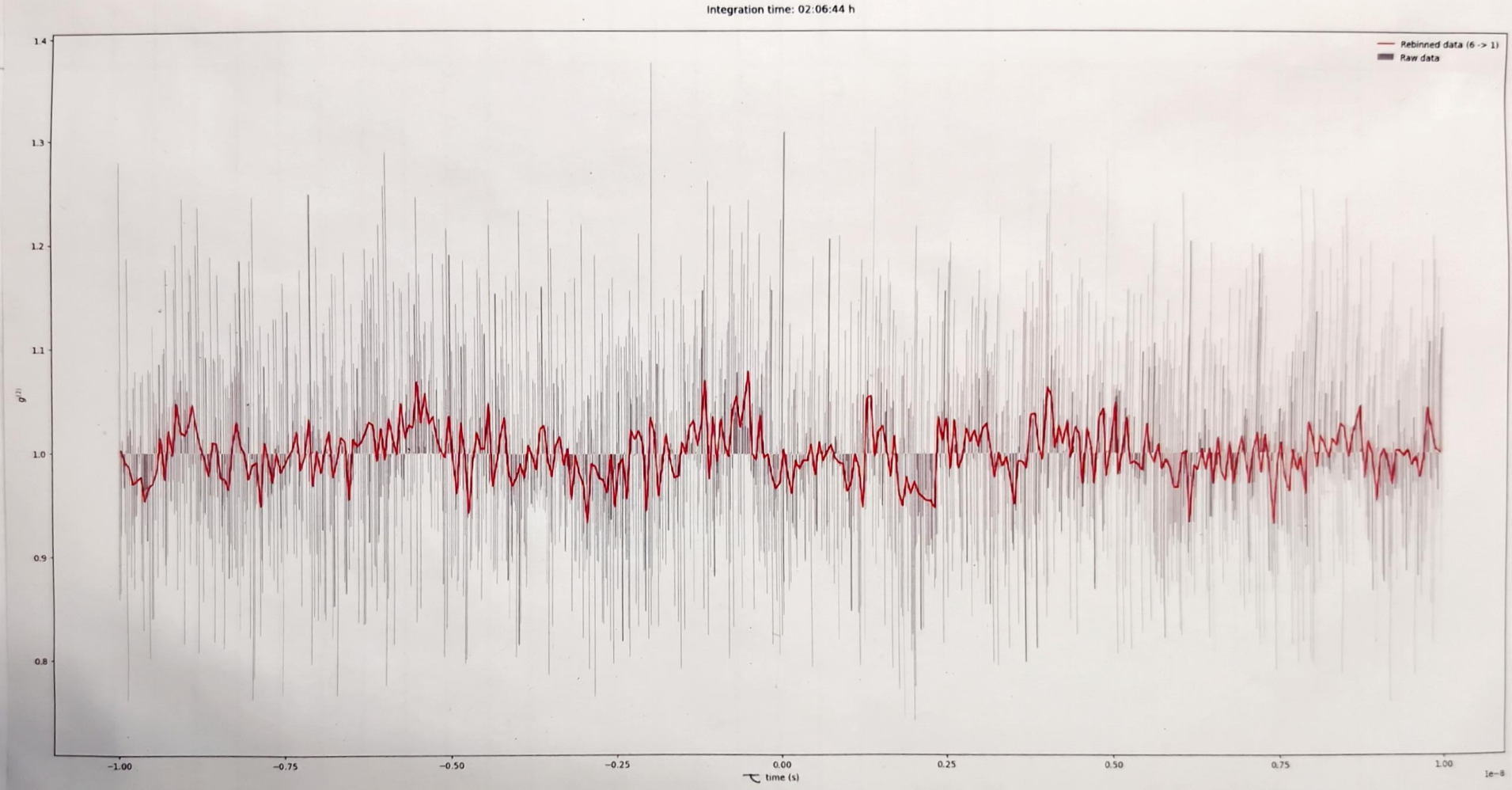
~3pm Detector 2 still gives noise, although there is a very weakly significant dip at around $\tau = 2$ns where it goes down to 0.870. This isn't significant enough to say anything definite, so I'll do an overnight run starting at 4pm to see if that is really a dip. I will set the settings to histogram with 20ns and bin count of 1999.

8/5/2025

~9am Sadly, the computer recording the data overnight logged me out, so the data is lost.
I'll instead start a run now for two hours.

~11am Looking at the data, it seems to again be mostly noise. The dip at about 2 ns does not appear to be significant especially with the shimmering. Even so, the minimum $g^{(2)}$ there is 0.7605

If I had more time, I would use mirrors for more degrees of freedom over detectors B and B', and I would experiment with what the half wave plate does to the count rate. Also, I've found that the python API has capability for 3-way beamsplitting even though the software doesn't. I tried to write code for it yesterday, but failed as I don't completely understand the python wrapper for this bit. We could use a protocol to do beamsplitting via hardware start/stopping the counts based on when A receives a count, but this would cut our count rates in half which is not optimal.



bin count = 1944
histogram width = 10ns
bin width = 1×10^{-11} s

Rebinned to 5 ll bins
bin width ≃ 6×10^{-11}

Conclusion

In conclusion, we successfully demonstrated spontaneous parametric down-conversion, achieving a maximum peak of $g^{(2)}(\tau_{centre}) = 7.81 \pm 0.05$ in the $g^{(2)}(\tau)$ graph using a histogram of re-binned bin width 6.25×10^{-4} s. We attempted to demonstrate the single photon-ness of the down-converted beams by passing one through a single photon detector, but we failed to do so.

In conclusion, we used two detectors A and B to successfully demonstrate and confirm spontaneous parametric down-conversion. We confirmed it by aligning the detectors with the 3° idler and signal beams and, constructed a $g^{(2)}(\tau)$ histogram plot. This plot, at best, had a peak of height 7.81 ± 0.05, (with bin width 6.25×10^{-4} s) which confirms the detection of two consistently near-simultaneous photons. After this, we attempted to show the single photon-ness of the beam by passing one beam through a beam splitter into single photon detectors B and B', and using the A detector to herald the photon in the beam. We failed to demonstrate this, since the $g^{(2)}(\tau)$ plot had no significant dips.* With more time, we could've implemented 3-way heralding code (since the software we used lacks it) and refined the optical setup to perhaps get more counts on B and B'.

* which would've demonstrated the existence of single photons, and we were unable to implement 3-way heralding.