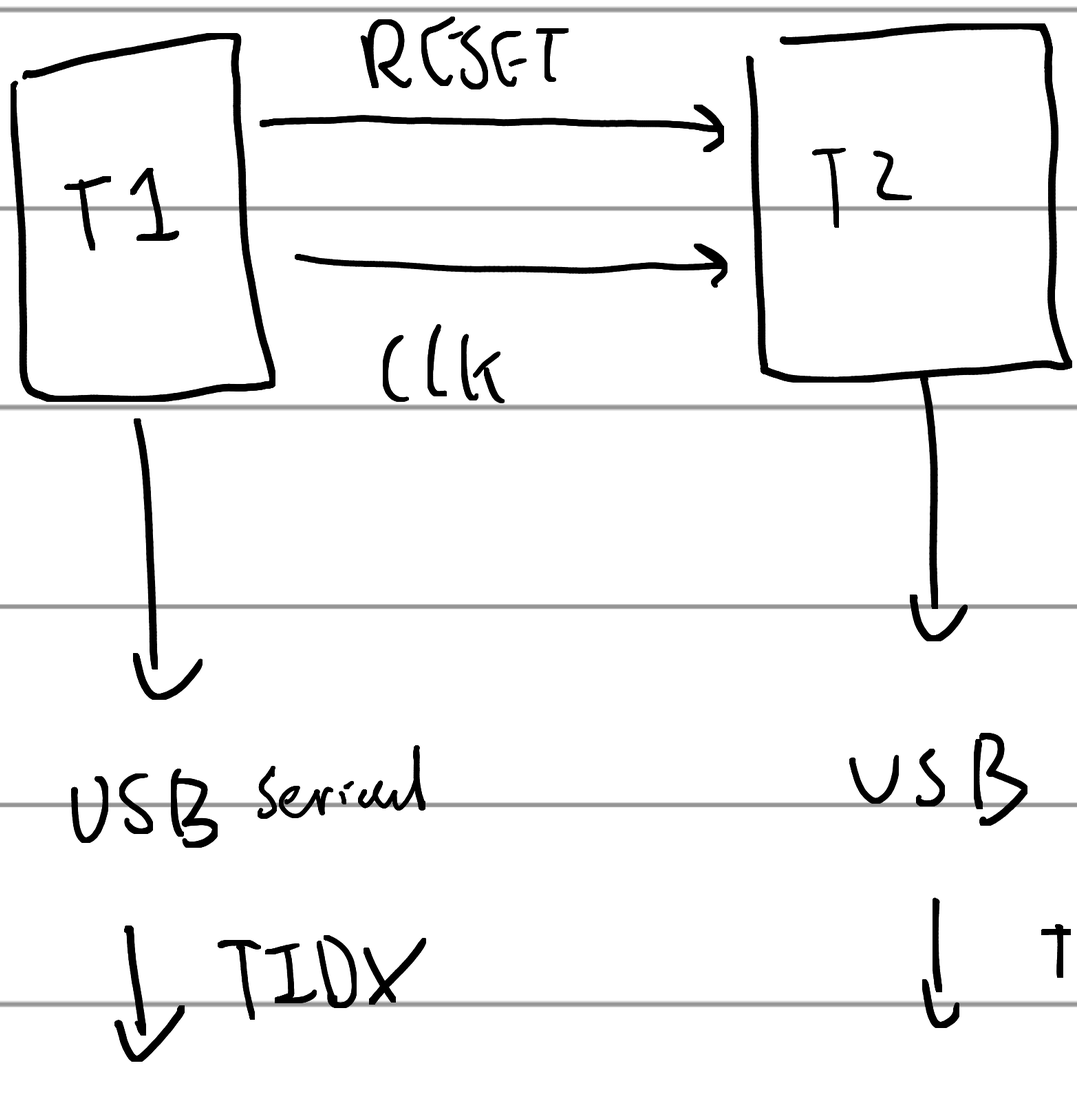
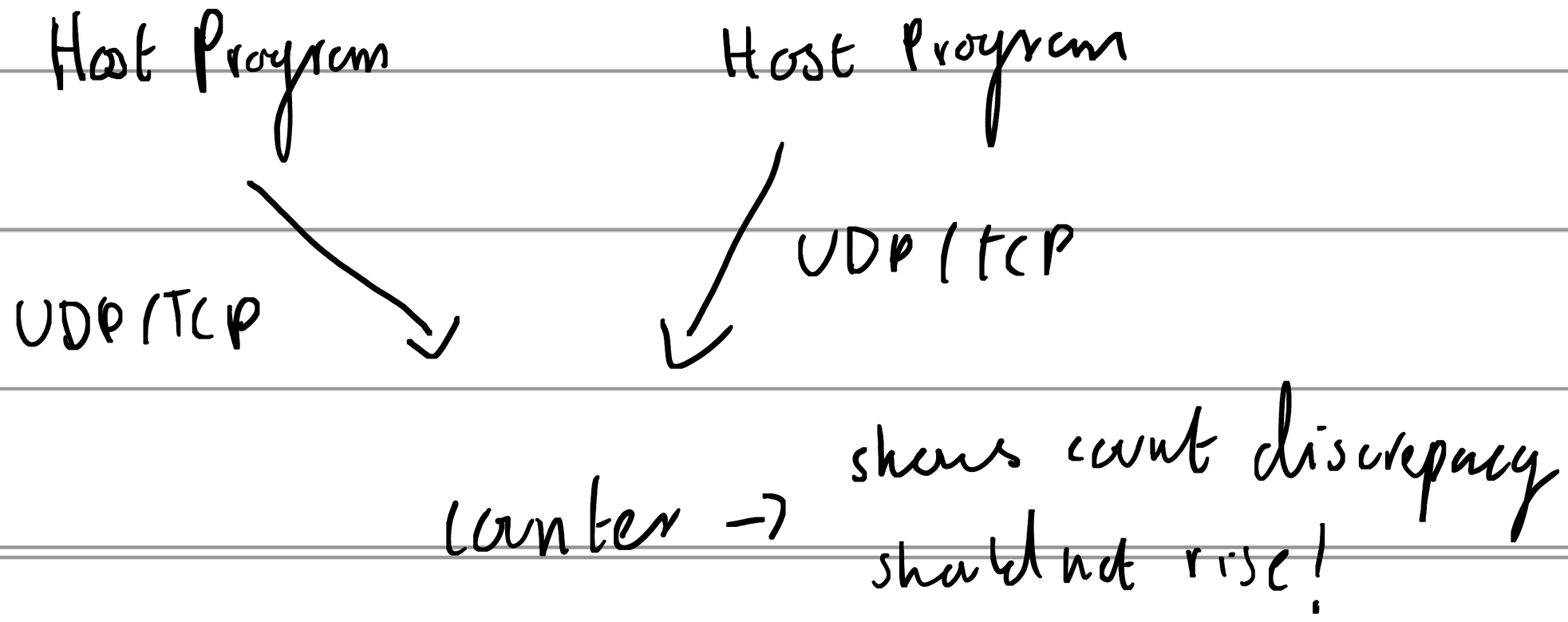
22/7/22

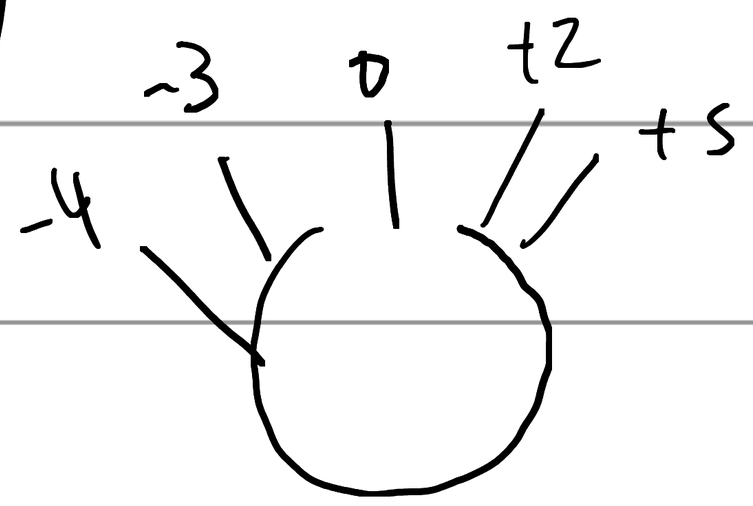
Thinking about synchronisation. I knew from testing that there is a fixed offset in terms of the number of CLK ticks, this indicates some initialisation issues, perhaps the RESET pulse will fix this? It's worth a test.





Could call this service Network Counter.

If the discrepancy remains constant but changes after RESET is pulsed then it would be a good idea to have multiple runs in each direction. Ideally balancing out the errors in each direction.



Could the network counter store the difference? Then we could maybe apply the displacement negating the diff & thus symmetrise after.

I worry that discrepancy is due to Serial.print a different amount of data, if buffering is the issue then could make

the byte message length the same. Although the time to compute each part of the message may still differ atleast we can eliminate one source of miss syncing.

If going to do multiple forward / backward runs need a way to combine datasets.

A B C A B C C B A C B A

So take backward direction & swap channels A & C & then merge based of angular position.

24/7122

Plan for next week → check Teensys for water damage with simple LED program

→ So. current program just pipes data from /dev/tcmp

→ Need a new program something fast maybe c++, bind to /dev/tcmp, open UDP port to CIK computer on the network.

→ Network send & log program ☐

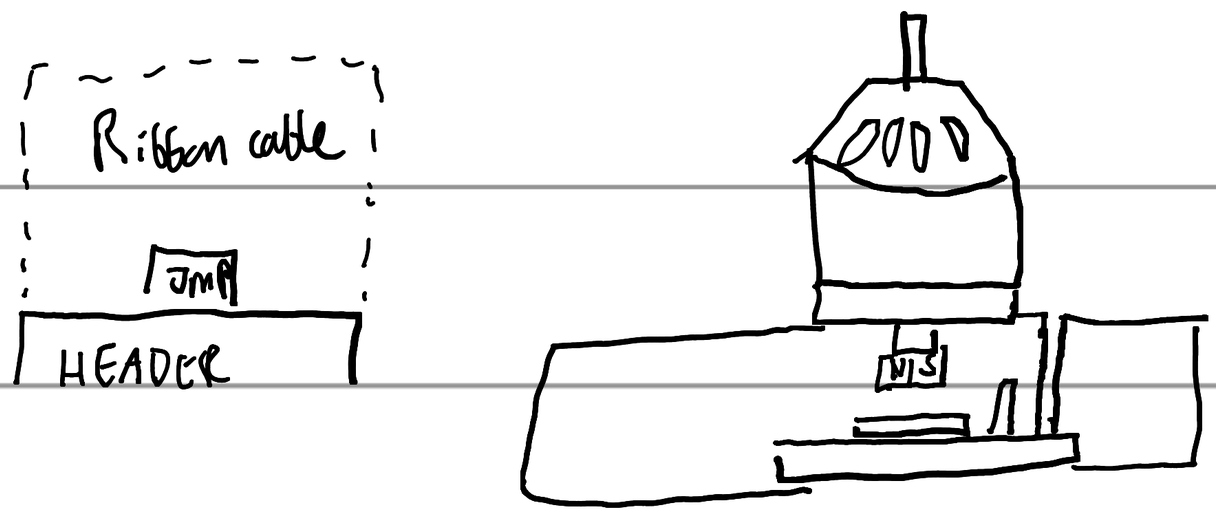
→ Network recieve & comp ☐

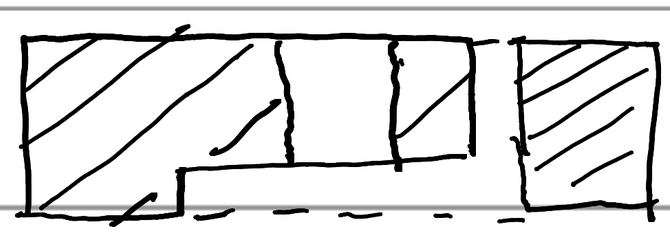
→ check ADC program ☐

→ check Encoder program. ☐

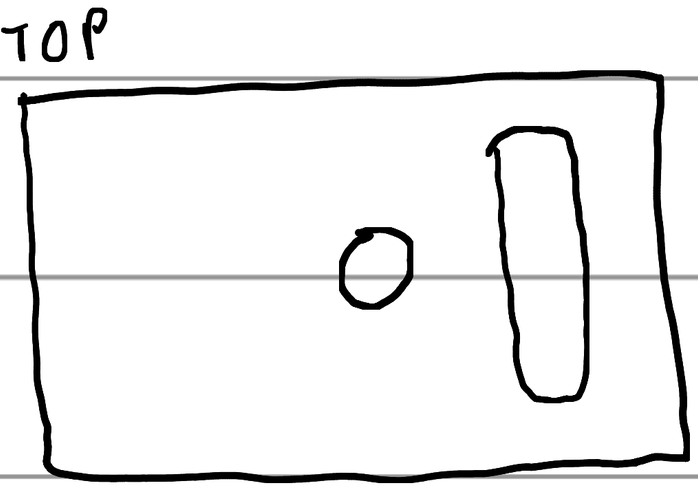
After playing with network sqme, do 1 simulated forward & backwards, test merging these datasets. Swap A & C.

Then need to actually mount the sensor.



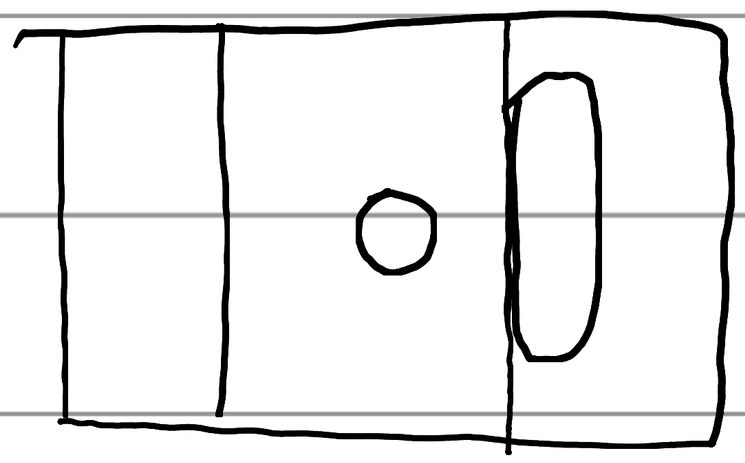


TOP



check where mounting screws could go, the encoder & motor need to be very well secured to avoid introducing displacement.

Bottom



Ask SK to design & manufacture this.

2/8/22

So teensy's appear fine after water damage.

Tried the code I made some observations.

→ Teensy gives the RESET pulse before logging from tty ACM0.

Resolution to this could be wait for first byte I then start the process of resetting.

work on custom logger I wait for init then send a byte to Teensy I start logging.

Python logic needed ... bind to Teensy Acm, open socket (baud serial), send byte ... start logging.

Parsing code done. (wait for bytes from serial)

Two laptops refuse to connect via local wifi, but both can contact my home server, so will load the network-multicapture-sync program.

4/8/22

Testing setup 2 teensys + 2 host computers + 3rd network computer for syncing.

Interesting results; had to reduce the logging rate to 7,500 Hz, remove logging to disk & with power cables disconnected.

Seems susceptable to:

Host computer speed (small laptop logging encoder values) might be the cause of buffering.

Network latency / bandwidth, play with encoder write volume.

Network sync only really needs time so could reduce the data transmitted & UDP packet size (SIZ). (time only packet)

Writing to disk could be a bottleneck, check where we are writing (tmp ram disk) could try buffering in memory in python.

Try a faster computer for the encoder host, put this on ubuntu server where we run the sync program.

If network packet is ok sized then sync program could buffer output.

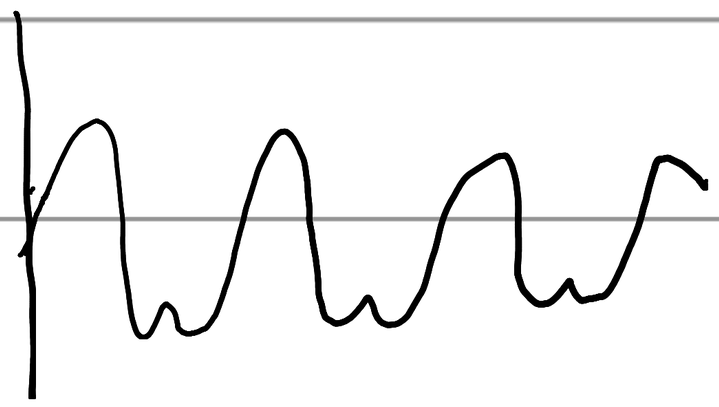
How much Hz do we really need?

$\frac{12}{2} \times 6 = 36$ steps

$\frac{7.5}{36}$ 20% there

rpm estimate

5,000 rpm → 10 k rpm



36,000 sps 18,000 sps gonna need something in this region. 18 → 36 k ideally double 72k + or more.

Step 1 new pc

Step 2 reduce logging

Step 3 retest

Step 4 reduce network packets

Step 5 retest

use ethernet cable? NO!

So eliminating slow laptop helps. Sort of stable

without writing to disk + with sync + service (device 1 encoder)

~50K (stable for about 10 mins)

Sort of stable with writing to stdout @ 50K. (a few mins)

Overclocking laptop hurts sync.

5/8/22

So tried to find the bottleneck

→ Tested network rate ~ 150mbps

→ Tested write to disk speed similar to network

→ Tested python vs nodeJS.

NodeJS is far faster for this sort of operation.

Turns out python serial library will not do a baud rate > 3000 (2048)

Rewrite in nodeJS allow baud rate of 5,000,000 more than enough. Now recording at x2 100 Hz with stability.

Need to fix code for nodeJS as is a mess, add arguments & fix package

write program to process network transmitted data, need to visualise data with drift run to make sure data is free from encoder noise.

Need to update documentation

6/8/22

Updated documentation.

Updated sync program & displaying maximum difference & current between device time.

Bug device 0 prints the baud number while waiting for start ... this messes up the sync program (diff & mdiff). suppress this print.

Took first network capture, affected by not unplugging laptop.

2nd network capture, affected by printing baud rate & start.

Step 1 → Remove bad print ✓

2 → Perform 3rd network capture.

3 → sync combine program

→ take last time from each device.

→ Remove any subsequent readings from device & dataset.

→ organise into two lists

→ from the last point in dataset 2,

find matching data point

→ merge lines into final format

→ print missing combinations & skip them. If something remains in the other list

→ when at end 0 or 1 step if not found in other dataset & no more entries in the other list (its exhausted)

used pandas with groupBy time, aggregate a set of lines, eliminated unmatched ✓

Next checkout combined dataset & try smoothing.

The smoothing parameters need to be command line arguments.

& sensible defaults documented. theta_variance

alpha

Check for noise in VN! Fingers crossed!

7/8/22

Probably better to re-write sync program in JS, worried about the latency causing data to be lost.

In source program need an on('close') handler so that we process exit after Teensy is unplugged.

1718122

Network sync via UDP is too lossy & TCP is too slow for the required transmission rate. Abandoning network transmission. writing via fs (node) was also too slow to show even if writing to /tmp ram disk. writing to /tmp was too slow. New strategy buffer list in node program only write to disk on Ctrl+C or if teensy disconnected, don't send anything by the network. Write to tmp on exit.

Data combiner is finished, created node program to merge the two data files from each host computers /tmp directories.

Testing

Problems with breadboard connectivity.

Problems if the drill is too slow, one channel is bigger than others.

Reminds me of last time with the parity bit.

Tried to remove time print from ADL as it changes very quickly (parity), but this time did not seem to have an effect, it was more to do with drill speed & checking connections. Also reduced log speed to 90kHz.

Got a few good runs with laptop plugged in and not plugged in! Kind makes me doubt how necessary the opto setup was fct: do remember huge frequency noise which in the latest run (run17-4) is totally absent (clean!) Lobbing about 90% data patches from the data combine program.

Todo:

→ use tip to transfer the data from nebula-device-server to the sync. (open connection on teensy unplug or ctrl-c) ✓

→ have sync program combined to data before writing it to the calibration dataset folder (do this after two connections close) ✓ suppress nebula errors → disk

→ chain commands

→ Readme cleanup ✓

→ ssh automation?

→ npm script to detect python env is active ✓

18/8/22

✓

Branch from main → FEATURES/ original - code-working

looking at network sync via tcp. ✓ Finished result

Still looks good, updated readme.

Updated FEATURES/ calibration from FEATURES/

calibration - network 2 ✓

Update main with F/ calibration. ✓

19/8/22

Made kmedoids a class

26/8/22

Noticed several problems with kmedoids, fixed cost kernal not returning an int, fixed broken euclidean metric (still need square root) but clustering with ball from pyclustering seems sensible. Original kmedoids did not include pyspark variant of k++ seeding (taken from old version). Within the fit function; added ability to add bespoke metric as argument, fixed return value... needed to add order by as cluster order vs centeroid order was being shuffled, centroids with INCORRECT clusters after all this time! Added a metric which obeys modular arithmetic so a distance between 0 & 16384 ... compared to 0 & 1000, leave cluster of {0, 16,384} & {1,000} with cluster size 2. Need to add pyclustering to dependencies.

Need to modify zero-cross program to change cache file save location.

Create new clustering program file which takes the result of zero cross detection program.

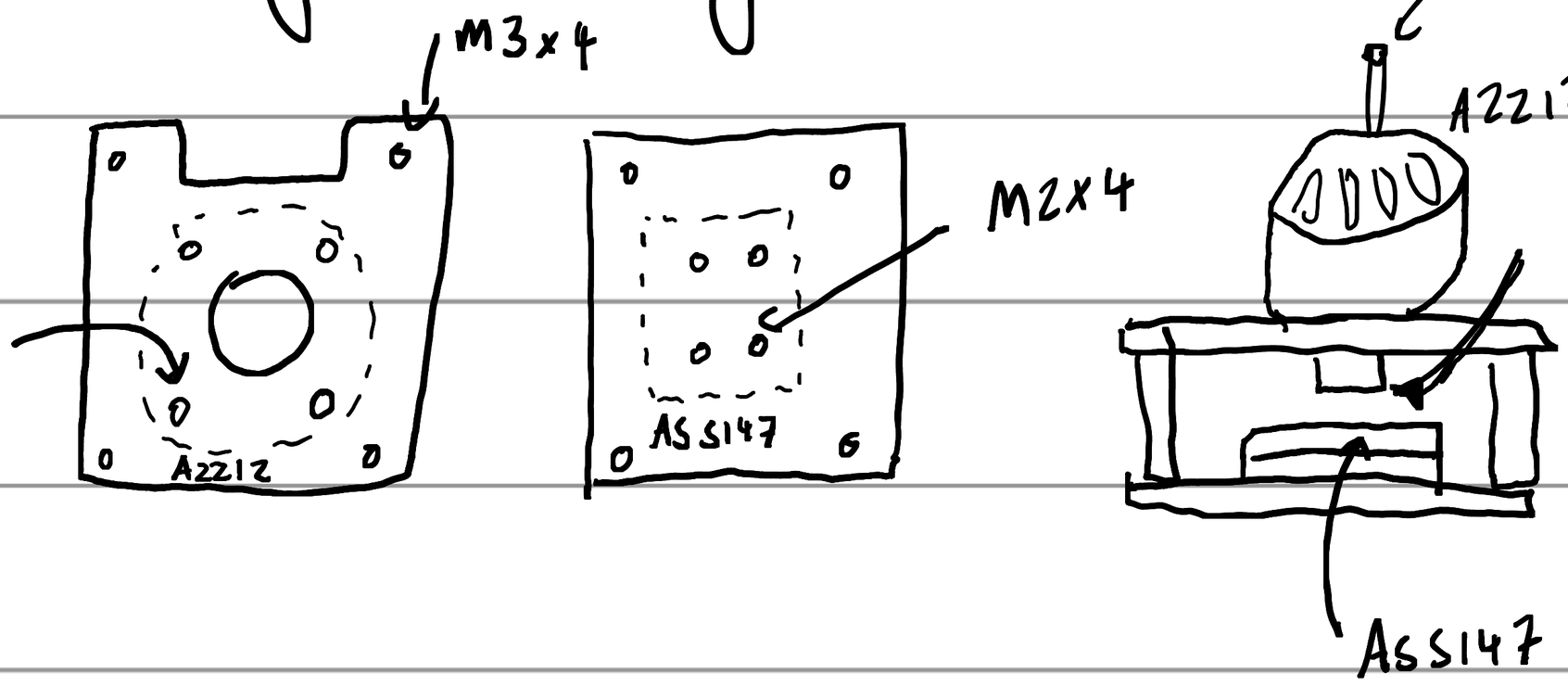
11/9/22

Working on assembling a mount for the test motor (A2212) and the encoder (AS5147 + breakout board).

TS-GK-AB

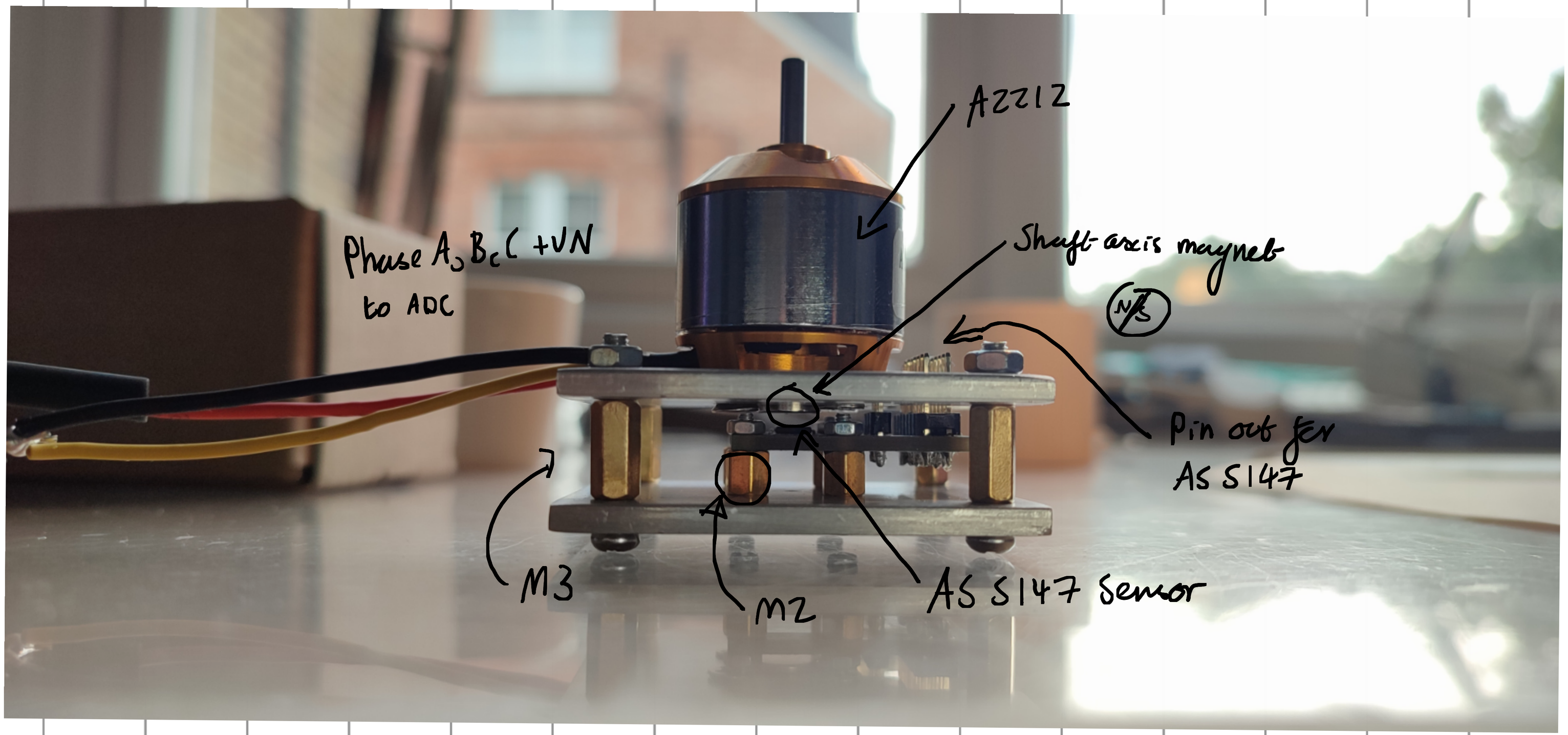
→ Successfully glued magnet to central shaft of the A2212 motor.

→ working on mounting plates.



→ SK finished assembly

Final assembly for AZ212 / AS5147



→ Re-wired Teensy, motor & encoder ready for testing.

Next steps:

→ check wiring + Jumper ✓

→ Run experiment and get capture with new assembly. ✓

→ checkout to FEATURES/calibration-documentation ✓

→ Re-run experiment & debug ✓

→ merge FEATURES/calibration-documentation ✓

to FEATURES/calibration & ✓

Main ✓

→ Create clustering program + graph clusters, mean, stdev & finally verify sequence.

21/1/22

→ change smooth output folder to datasets/data/calibration-data

→ Zero cross, check output / save output to file. Arguments for program.

→ Thinking about graph program, needs two pieces of data ... I like histogram of counts per channel & the clustering data per channel.

Need to per each channel:

→ plot histogram

→ plot each cluster with different colour.

Need a histogram with multiple colours (series)

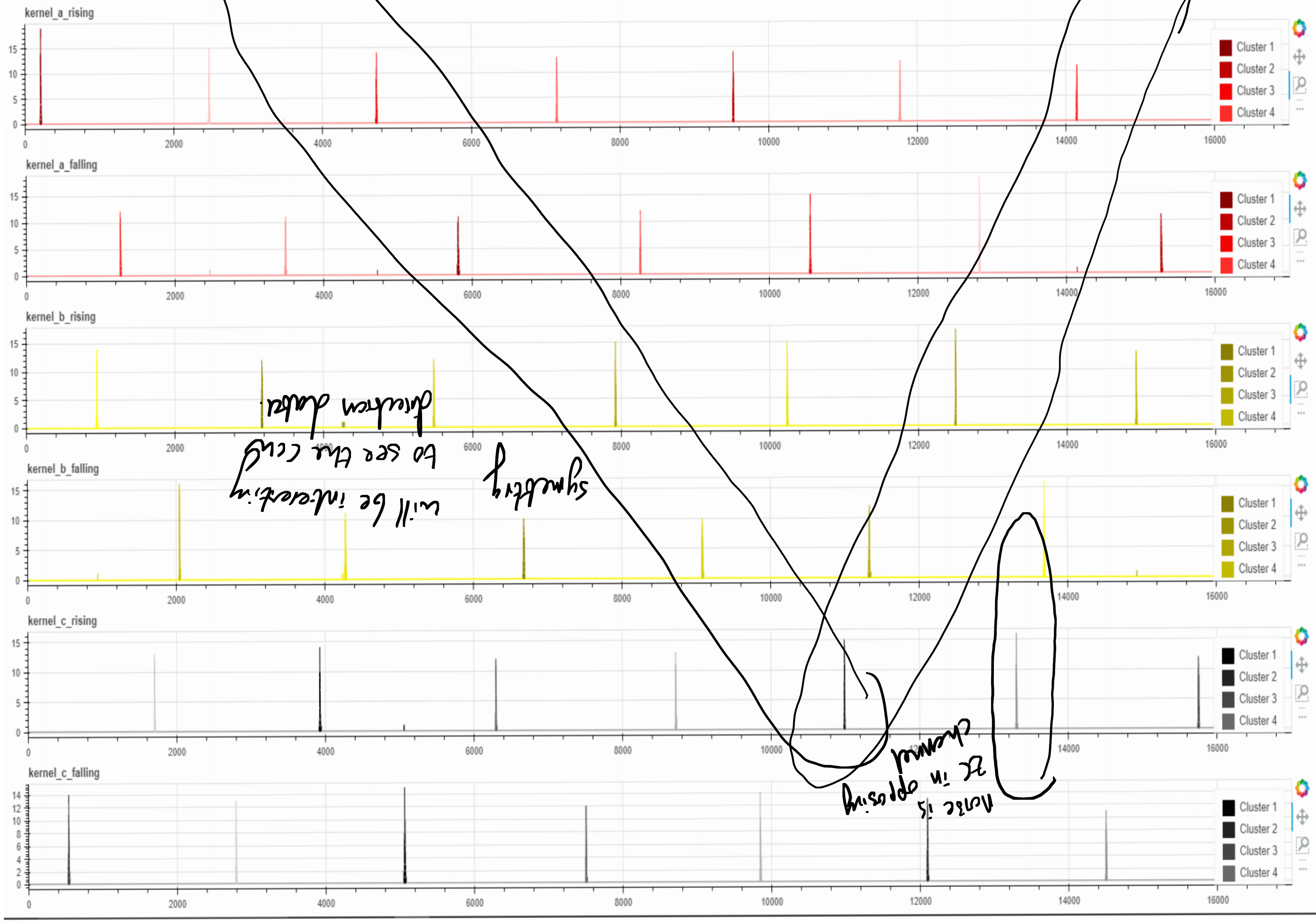
→ Finished a draft of this software

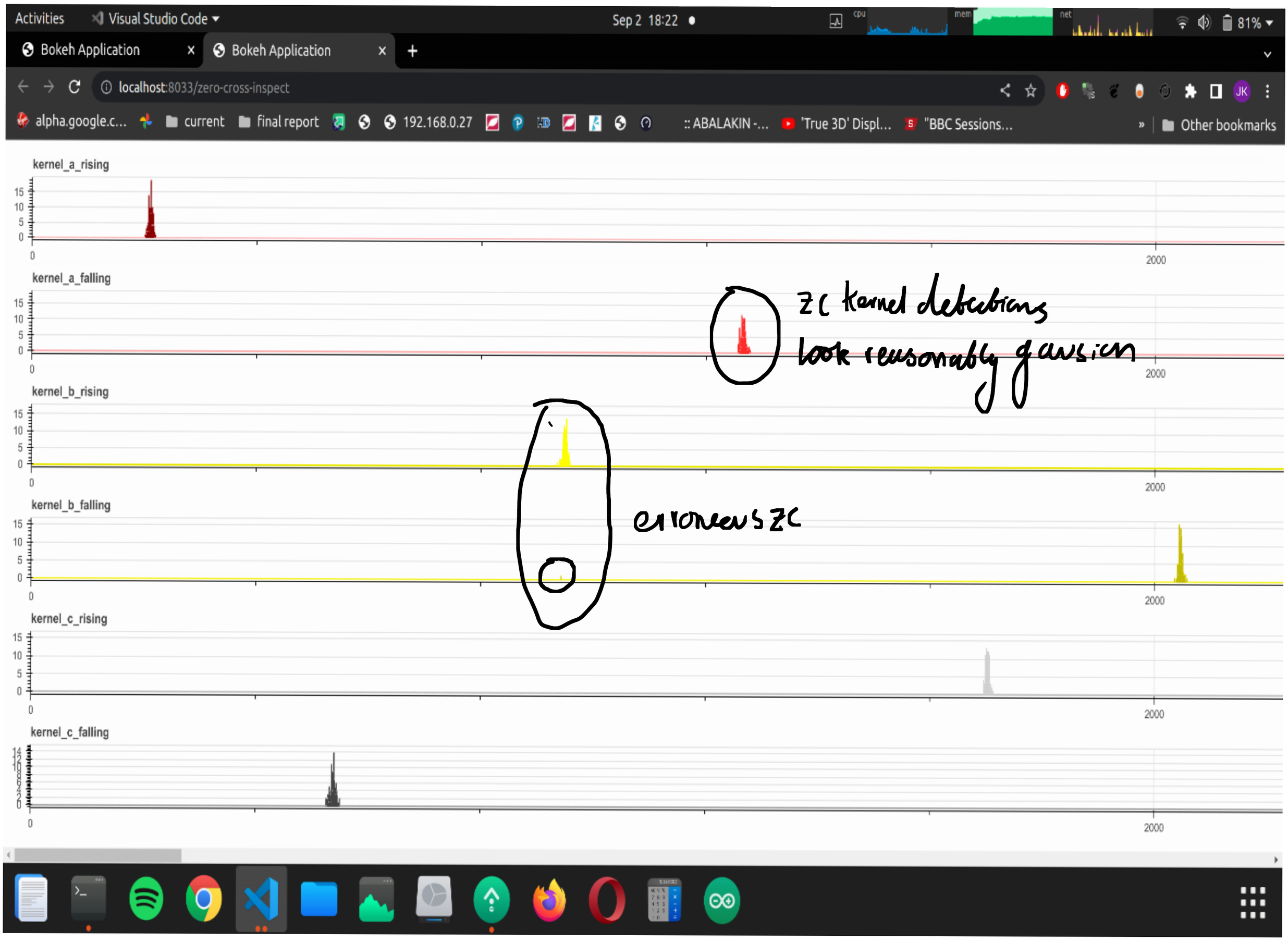
→ Tested the software with a CW test.

Bokeh Application | Bokeh Application | Bokeh Application

localhost:8033/zero-cross-inspect

alpha.google.com | current | final report | 192.168.0.27 | :: ABALAKIN - THE... | 'True 3D' Display Us... | "BBC Sessions (In C... | "Expert Body" To D... | "Kommander's Car"... | "Warm Chill" by Juli... | Other bookmarks




Activities
Visual Studio Code
Sep 2 18:22
Bokeh Application
Bokeh Application
localhost:8033/zero-cross-inspect
kernel_a_rising
kernel_a_falling
kernel_b_rising
kernel_b_falling
kernel_c_rising
kernel_c_falling
ZC kernel detections
look reasonably gaussian
erroneous ZC

Thinking about the mean & standard deviation with Circular statistics. What do we want from a mean value? It is a representative point which minimises the distance from the point & every other point in the set.

$$\mu = \arg\min \left[\sum_{i}^{N} d(x, x_i) \right]$$

x is a set

x_i is a member of x

if this was 1 then you get the median

A good choice for $d(x_0, x_i) = (x_0 - x_i)^2$ (2) sum square error

Other metrics root mean square etc

The variance can be obtained $\sigma^2 = \frac{1}{N} \sum_{i}^{N} (x_i - \mu)^2$

So thinking about it, the centroid we have per cluster is the mean & we can calculate the stdev by doing the sum of the pairwise cost metric over the whole set.

Other metrics

sum of squares $\sum_{i=1}^{N} (x_0 - x_j)^2 p_i$

mean square $\sum_{i=1}^{N} \frac{(x_0 - x_j)^2}{\sum_{i}^{N} p_i} p_i$

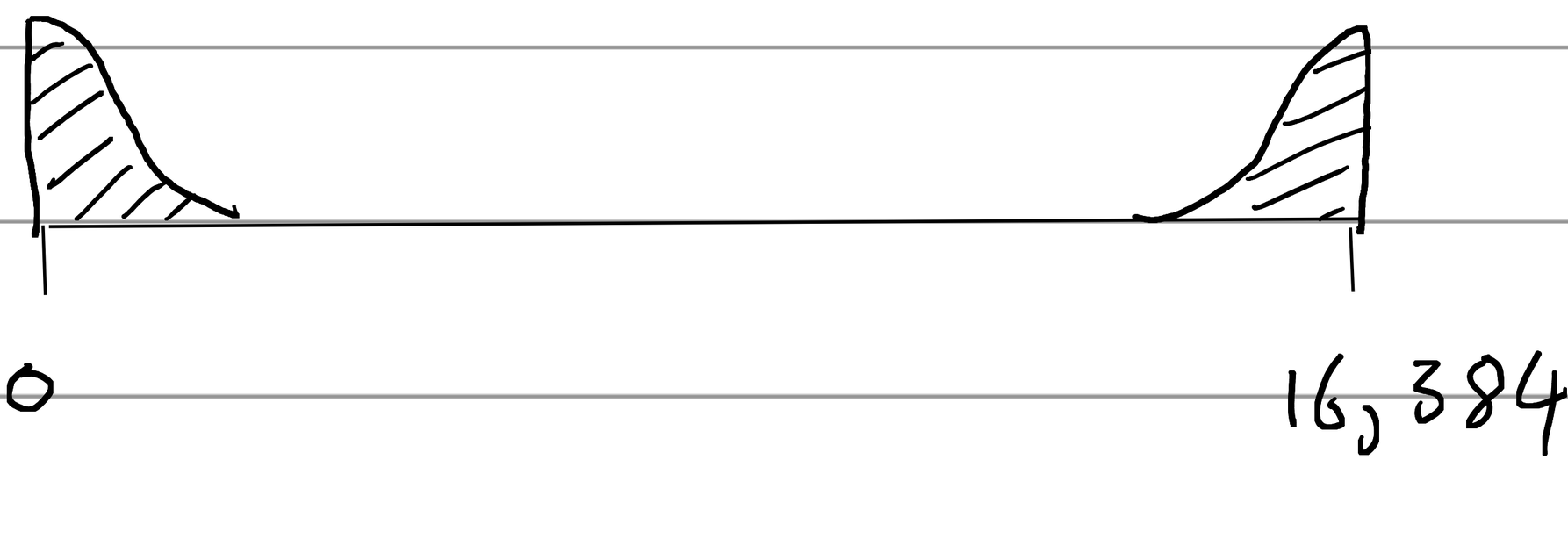
root sum square $\sqrt{\sum_{i}^{N} (x_0 - x_j)^2 p_i}$

root mean square $\sqrt{\frac{\sum_{i}^{N} (x_0 - x_i)^2 p_i}{\sum_{i}^{N} p_i}}$

The quadratic function is strictly monotonic (first derivative does not change sign) for positive arguments, hence does not matter, they will yield the same argmin

Other moments $E(x-b)^n = \sum_i^n \binom{n}{i} E(x-a)^i (a-b)$

Due to the modular/circular nature of the distribution when we unwrap the axis we could get a zero-crossing at precisely angle 0 (12 bit) & we could find our gaussian split after uncurling the axis.



Now from Kmedoids we will get $\mu = 0$, to get σ we can use difference squared sum which represents the error & the distance must again obey modular arithmetic.

$$delta = (angle2 - angle1) \% 16,384$$

$$delta > (16,384 / 2) \text{ then } delta = -(16384 - delta)$$

$$delta <= (16,384 / 2) \text{ then } delta = delta$$

4/9/22

Improvements, would be good to have a folder per each experimental run & then have uniformly named files in this folder for each of the processed files e.g. kadnen smoothed, histogram ZC, clustered ZC etc. Save files as html for each plot.

Additions for graphing software, need to work out statistical variance / std dev, plot error bars & mean.

New graph for finalised ZC, rising edge +ve falling -ve.

Validate square cw/ccw logic (will need to do both a forward & backward run & compare.

Create state map for cw & ccw

Extra plot for smooth (a-vn, b-vn, c-vn) etc; raw data

5/9/22

Need to change data format as to seperate centroid & cluster members. ✓

Calculating mean & stdev (root(sum square/N)) ✓

Plotting error bars on zero-cross - inspect.

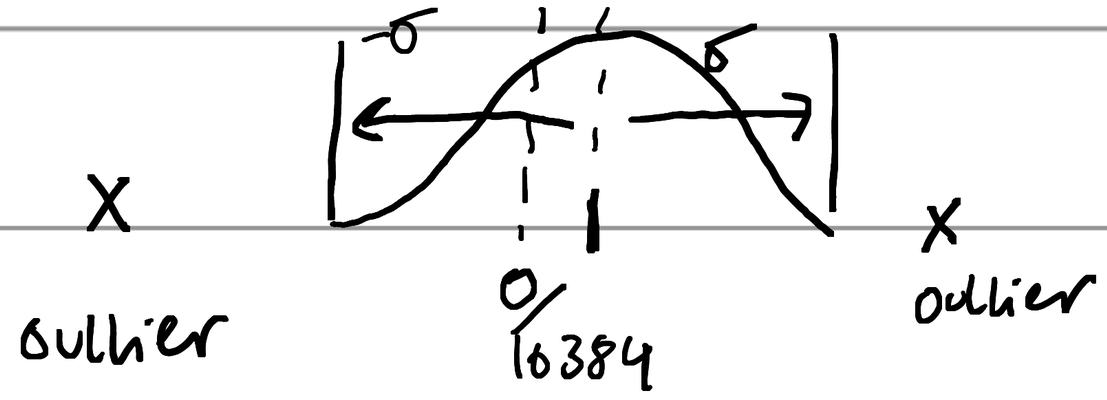
Thinking about outliers & how to eliminate them. We have the centroid & cluster members per channel & we have the mean & stdev

We could define outliers as points whos distance from the

Centroid is greater than $n \cdot \sigma$ where n is a constant

by using the modified metric we can avoid dealing

with crossing over $16384 \rightarrow 0$ line.



$d(X, \mu) > n\sigma(X)$ eliminate

Need to do this per channel.

Implemented successfully & adapted respect to chart

the result.

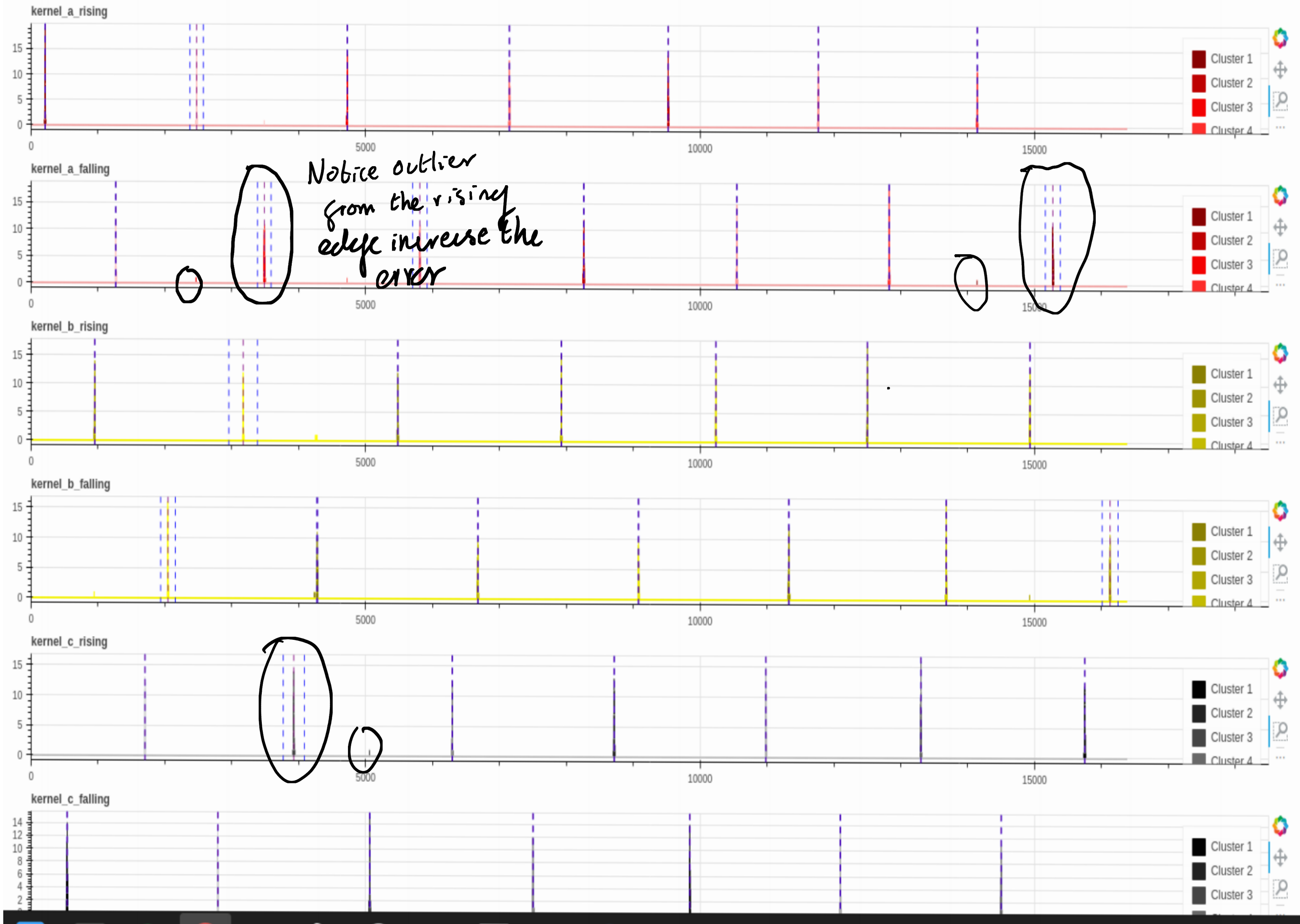
Activities Google Chrome 18:17 22.2 °C 2.40GHz 100%

Bokeh Application | Bokeh Application | gaussian std deviations p

localhost:8033/zero-cross-inspect

Update

Apps | alpha.google.... | current | final report | 192.168.0.27 | ::ABALAKIN -... | 'True 3D' Displ... | "BBC Sessions... | "Expert Body"... | "Kommander"... | Other bookmarks



Clusters with standard deviation error bars (blue)
& mean identified in purple

Error & mean of clustered channel data after outlier 735 elimination.