24/9/22

CW ECCU binning (1 to 1) of spike train see if different

Global error minimisation function (what is the cost function)

MHZ sample test.

Spin test

More experimental results

Error quantification per cluster (convert to steps) % error.

Switching map generation

CW / CCW deviation from ideal. Spacing error.

Redo of ZC displacement / switching zone size

$|^{0}_{e} = \frac{2}{poles} |^{0}_{m}$ * Number of zones = 3.poles

Full elebrical cycles is 6 switching steps 360°_{e} is 6 steps

$360^{\circ}_{e} / 6 = 60^{\circ}_{e}$

* $60^{\circ}_{e} = \frac{2}{poles} 60^{\circ}_{m} = 1^{step}_{e} = \frac{120^{\circ}m}{poles}$

in steps $\frac{2^{14}}{360^{\circ}} \times \frac{120^{\circ}}{poles} = \frac{2^{14}}{3.poles}$ "Number of digital angles per electrical switching step

For pdes = 14 we get $\frac{2^{14}}{3.14} \sim 390$ steps is the switching zone size.

The ZC displacement is 30°_{e} $d_{ZC} = \frac{2^{14}}{6.pdes}$

Teensy 4.0 2Mbyte flash, 1Mbyte ram

Thinking about speed control loop

The IRQ can interrupt the

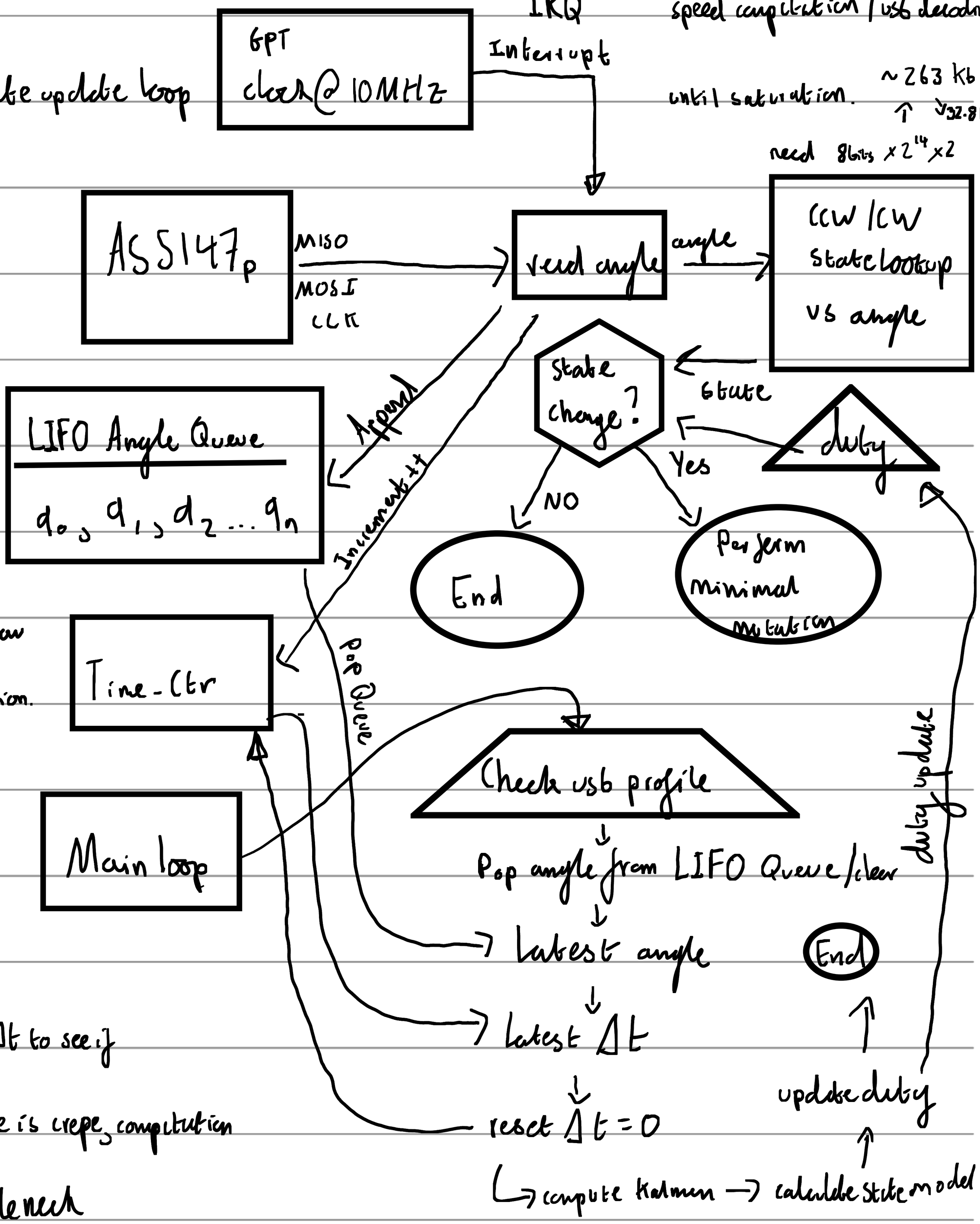
speed computation / usb decoding

until saturation. ~263 kb

↑ ↓32.8kB

need 8bits $\times 2^{14} \times 2$

State update loop



use Δt to see if

there is creep, computation

bottleneck

26/9/22

Will attempt another cw/ccw run of about 10s with node memory flag set.

Update combine reports to declare mean & error & relative error.

Displacements from ideal now that I have derived the formulas, global ✓ error functions.

→ displacements from ideal, new_mean: {"zc_chunk": {0: tuple, per chunk get key, value tuple disp→ tuple + check ✓

Interesting oscillatory pattern! 30 disp max

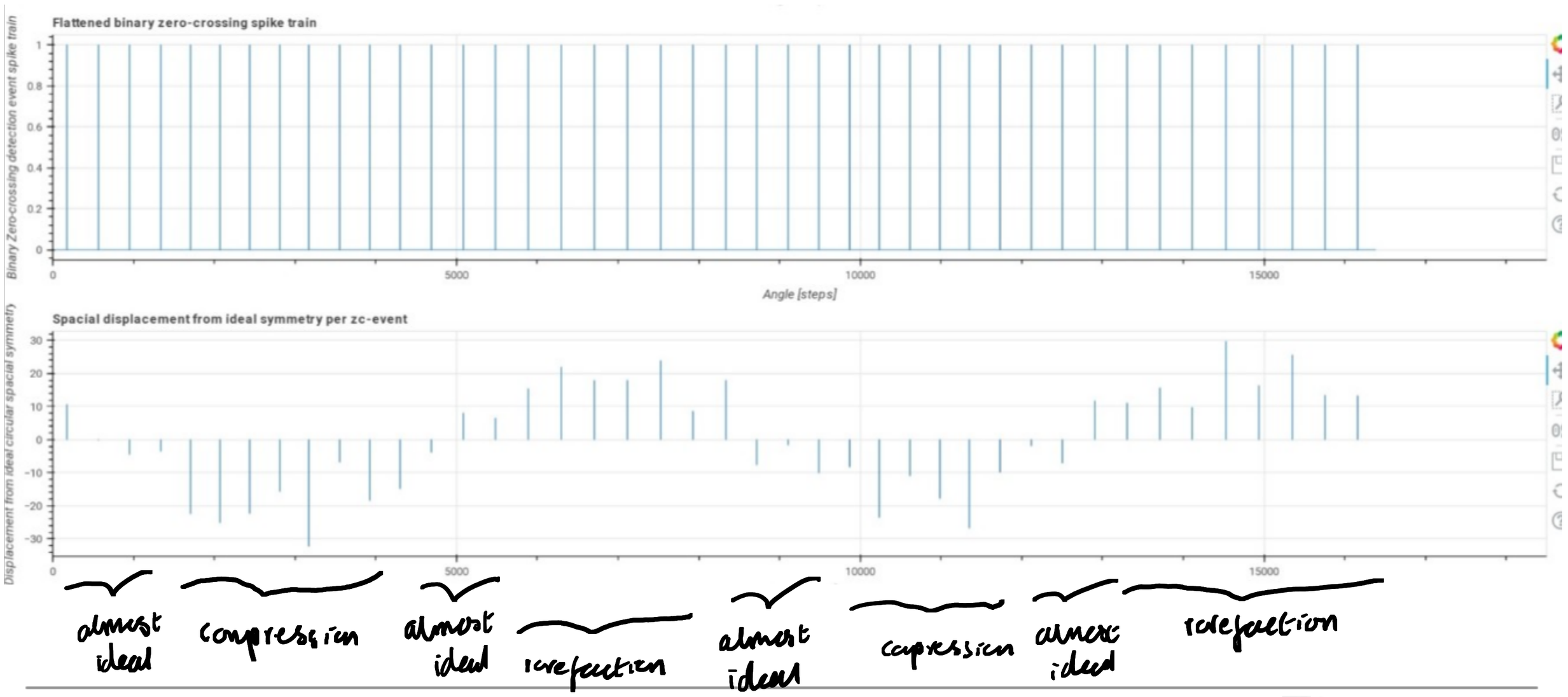
→ cluster error

chunk mc cluster 1 ... cluster N ✓

zc-rise... 10 ± 0.1 ...

✓ channel error avg. of cluster errors min max

Global errors based of displacement from ideal. ; do errors for abs as well as raw value analysis. ✓



Thinking about the above displacement away from $\frac{2^{14}}{3.14} \sim 390$ steps

Ideal spacing

chart above, interesting it seems to oscillate like a sinusoid with a period of 2x the circular period. As the motor has circular symmetry ↓ 8.57°

it make sense that the sum of displacements $\sum d_i = 0$, as

if not the pattern would shift spacially on the next turn & it would not be periodically

stable over the circle, its not obvious to me why we get a 2 period sinusoidal like

pattern, perhaps this is evidence of some error based on angle, or

we have revealed some real manufacturing asymetry for the AZ21Z motor.

Max displacement from ideal on the order of ~ 32.2106 angular steps (4dp)

Conversion factor is $\frac{360^\circ}{2^{14}\text{ steps}}$ [degrees per step] → 0.71° (2dp)

Average displacement from ideal spacing ~ 16.1303 → 0.35° (4dp → 2dp)

Smallest displacement from ideal ~ 0.3578 → 0.0079° (4dp → 2dp)

Best cluster error 6.3774 → 0.14°

Worse cluster error 87.8996 → 1.93°

Would be nice to know how cluster error correlates to error of the deviation from ideal symmetric spacing.

Thinking about ZC displacement, if it turns out the displacement from ideal symmetry has a real physical pattern, the displacing the ZC by 30° or globally would be a bad choice, instead take the middle point of adjacent ZC-events.

29/9/22

Next step long duration cw/ccw run ~10s +, see what length of time is stable.

iteration.

What naming convention for runs XX _ Sept _ A2212 _ cw _ X

date month motor direction

→ 29_sept_A2212_cw_1 99.95% match

rate

→ 29-sept-a2212_ccw-1 99.96% match

rate

Combined reports for these two cluster error went up, decision from ideal went down. Interesting that the error (cluster) does appear dependent on runtime! Big insurance to make sure cw/ccw runs are of equal length!

How can we ensure run lengths are nearly the same?

Have host PC send the number of iterations required to

teensy (ADC), have that teensy only poke the slave n times,

have master teensy send termination serial print to host

so host can auto exit, giving the user some indication

of termination.

Then with source program for ADC teensy, have a flag for

run size., lots of short runs should minise duster error, & labs

of these runs. → Better capture → Auto analyse command!

We probably can't prevent skew from mis matches!

NuForce DDA 100

eink capacitive

Ohm meter
VU meter

Blue dot matrix

orange (drilled)

CI 30

chord 30

OLED

USB

laptop

Inputs

Outputs

on/off button

pitch (linear)

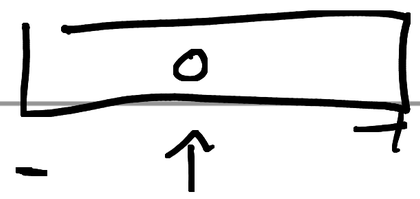
speed (button / dial)

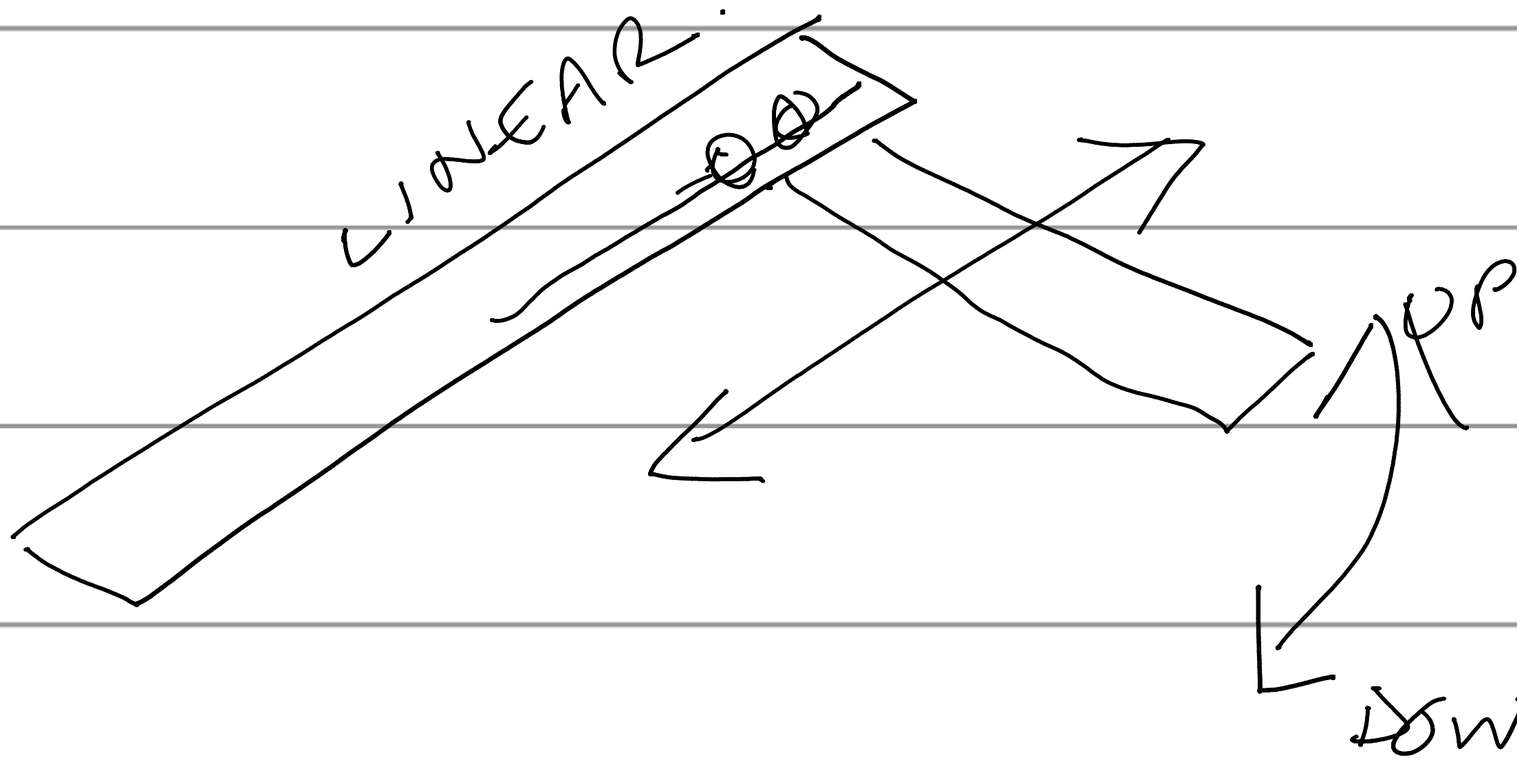
coarse grain vs fine grain.

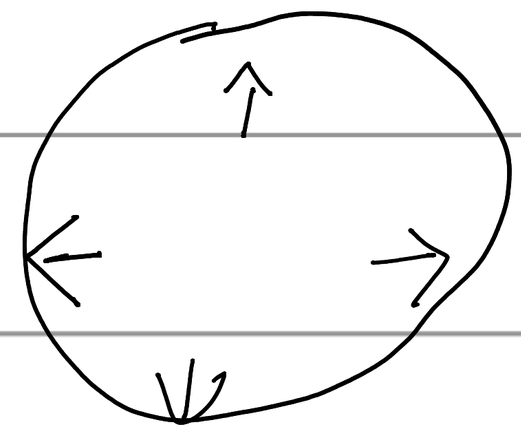
Target Speed 33.3 rpm 33.X

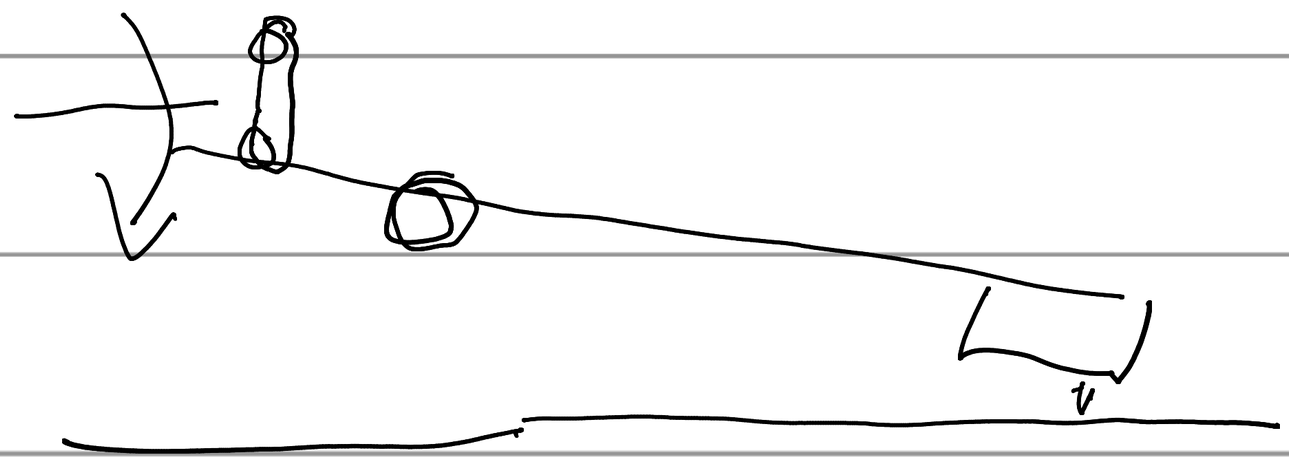
True Speed 33.2 (dignostic usb

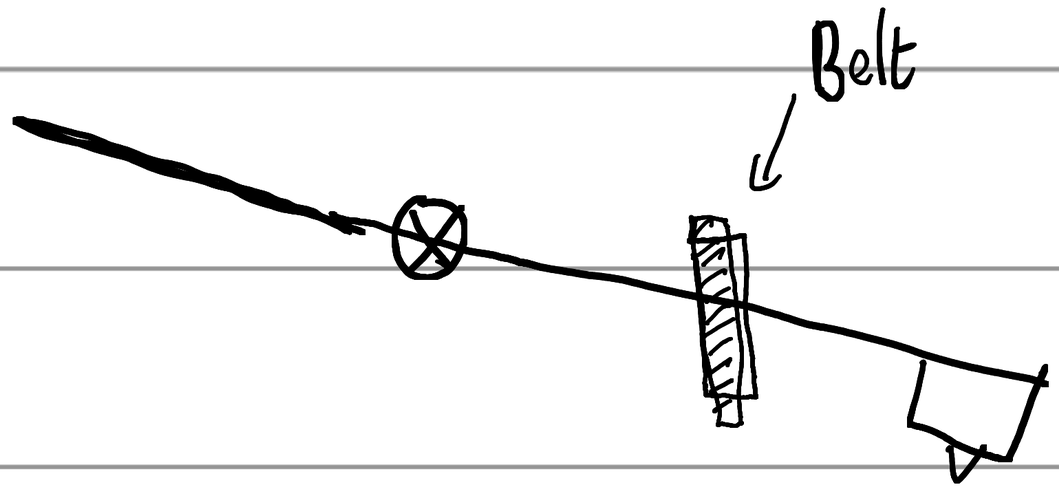
Pitch









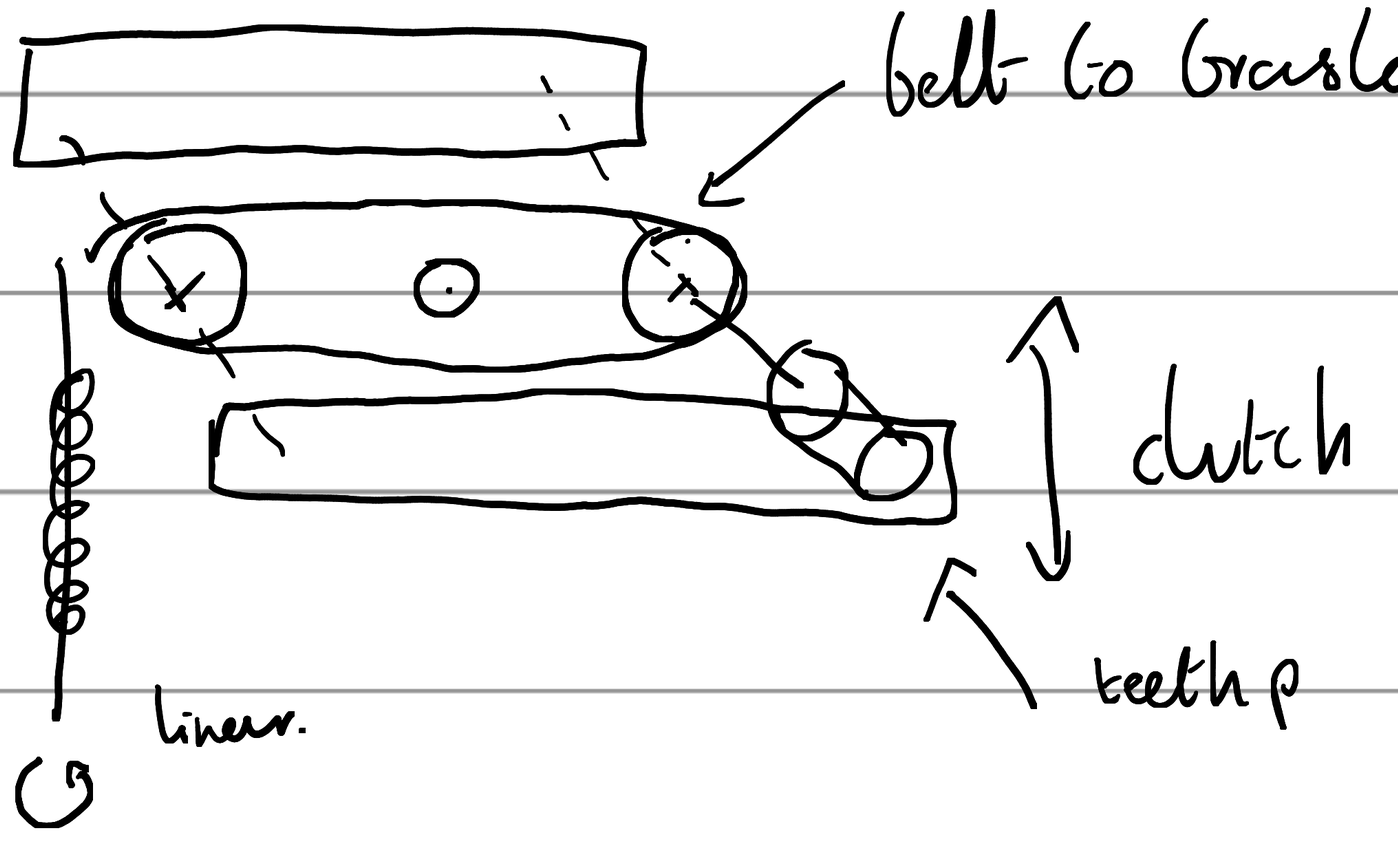


low friction
cirder arrangement

Raise arm

to pivot bit

needle dumps.



DC xV

16.8V

thermocouple

SBIDC

DC

current shunt

10V

SBIDC

PD Buck

Teensy

SCN

CLK

MISO

MOSI

Plug for calibration Module.

A B C

a b c

VN

Feule

USB diagnostic

ribbon

PCB

Motor

AS5174p

S N

N/S radial poles

braided cable

Power conveying

SBIDC

Teensy

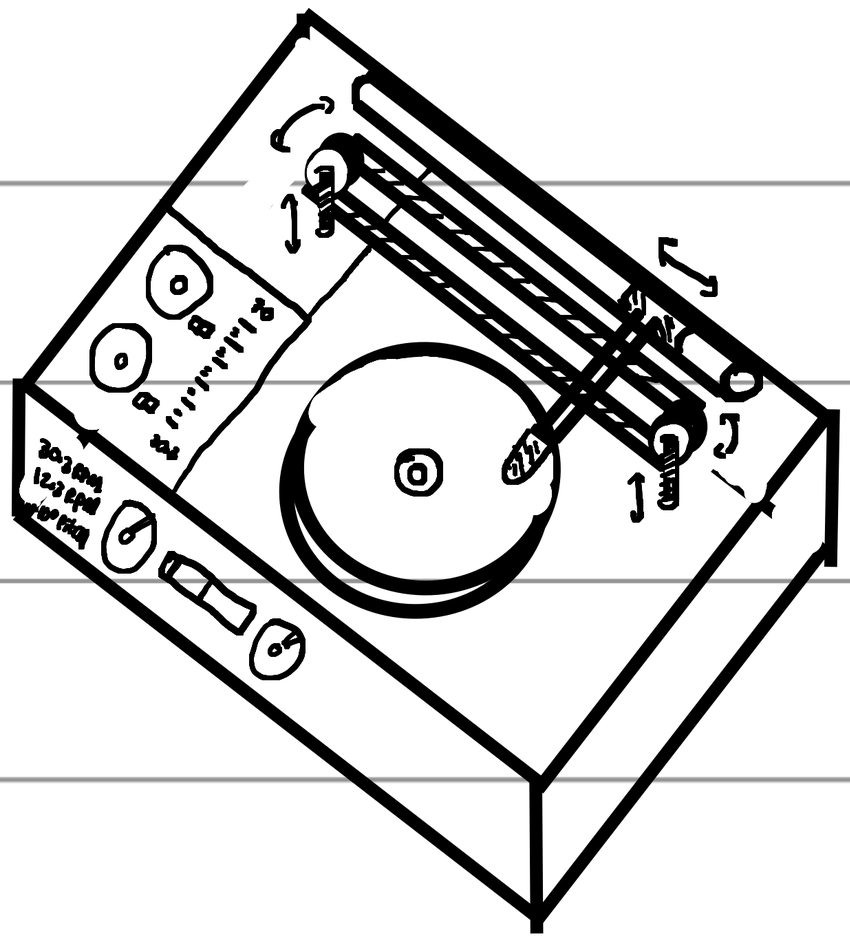
PC PS

NIG

AS5147p

NG wires

ethernet



28/9/22

Created sample rate program, capturing cycles @

1,041,611 Hz ... need about 5x! or at minimum 2.84MHz

running nano second delay (3ns) ... 1.51506 MHz

running parity check 1,744 MHz 6 cycles

running at 816MHz ~~2,371,990~~ Hz 6 cycles.

Remove ns delay checks reading cycle

try 2ns fail

try 3ns 814MHz fail

3ns 720MHz → 71.4 12 MHz

2ns 720MHz → fail but maybe missing 8000-16000

Sample Rate = $S_R = \frac{2^{14}}{60} \cdot \omega[rpm]$ $\frac{60 \cdot S_R}{2^{14}} = \omega(rpm)$

3ns 720MHz $S_R = 1,411,700$

$\omega(rpm) = 5,169.8$ Hz

Code optimised for "fastest" 1,451 MHz

720MHz / Fastest / 3ns

goal 10,416 rpm

1,451,500

$5,31 | 5.55$ Hz

↳ 0.51 steps so

best case sample every $\frac{2}{2^{14}} \times 360^\circ = 0.04^\circ$

this is for hard mode

10,416 rpm.

For azzie:

target 16,800 Hz $\frac{60 \cdot S_R = w(rpm)}{2^{14}}$

$w(rpm) = 5,169.8$ Hz

$\frac{}{16,800} = 0.307$

$\downarrow$

Sample every $\leftarrow$ $1/0.307$

0.671 $\leftarrow$ 3.2496 steps

Min cluster error is about 6.377 steps. X

Trying cross loop optimization 1.4783704 MHz

1.8% improvement from desc

720 MHz, parity removed.

Trying remove 3ns tuning delay on last 2 bites.

7.8% improvemb

1.565 MHz

~5731.73 rpm.

Remove all logic from last 26 bits, apart from

value shift.

13% improvemb

1.651 MHz

57%
improvemb
from strategy

6047 rpms

goal 2.84 MHz

Convert 16 → 32 bits mask 14 1's

2.3% improvement from 16 bits 1.69 MHz

alternate 3/2 ns @ 1.743 MHz

all 2ns @ 1.888 MHz

Try no op for wait 1.8996 MHz

6956.73 rpm

→ 800 secs
slowly possible.

1.9 MHz With fewer first no-ops

goal was
2.84MHz

C
720MHz

1.693595

7687.3 rpm

~86% improvement.

10,416 1.46 steps.

@ 16,800 rpm $\frac{16.8\text{k}}{7087.37} \sim 2.37$ steps.

Re-reading the data sheet 2^{14} is not the accuracy of the encoder it's 2^{12}. I can see ±4 steps only

so this makes sense.

$\frac{60 \cdot S_R = w(rpm)}{2^{12}}$ S_R best 1.9353395 MHz

$1{,}935{,}395$

$w(rpm) = 28{,}350$ rpm.

4096

looking at uncombined single direct duster errors they were on the order of 6 steps, as we are only getting ± 4 steps, the error is only slightly more than min step size. So worse

Target

worse case 16

0 4 8 12 16 so ± 8

Another thing the cache map was going to be

$16,384 \times 8 \times 2 = 262\,144$

But in reality there are really only 4096 (2^{14})

so $4096 \times 8 \times 2 = 65,536$

would need to build a 2^{12} map for cw/ccw, should divide

We could get by 4_3 `>>2` bitshift. CANNOT DO THIS | get odd axes, not huge speed → interpolation

Should see how fast we can get with overclock using delay nanoseconds. (try 1ns)

NO NEED

Create different versions based on different MHz (easy clock.

NO NEED
stable @ 600 MHz

Create a way to combine cluster errors (generic systematic error (multi scale)

29/9/22

Retesting just one encoder method.

	Smallest freq	Rpm max	Round down
@ 720 Mhz (max possible)	1,924,956	28,197	
@ 600 Mhz	1,604,086	23,497	
@ 528 Mhz	1,411,581	20,677	

To be safe if we are going to do a loop for this we should back off the max speed by atleast 10%. even @ 600 Mhz, with 10% reduction we are over 21 kipm.

	10% reduction		Removing clocks after edge measured.
720	1732460	25377	304
600	1443677	21147	184
528	1270422	18609	112

Important observation, raw sample data does include odd numbers! So the ADS127L11 does interpolate at high speed! Neat! Will have to keep ZC map as a 2^{24} bit numbers, as bit shifting odd numbers will loose precision. Fine.

So what is left to do:

→ Combined cluster errors + host offset shifting codes from different runs over the top of each other to reveal a cluster error divorced from the systemic displacement.

→ Programable run-times & energy ADC exit cmd to host.

→ Perhaps improvements to sync/serve so that multiple runs are easier to perform.

→ Find cw/ccw displacement maps.

→ ESC high speed program

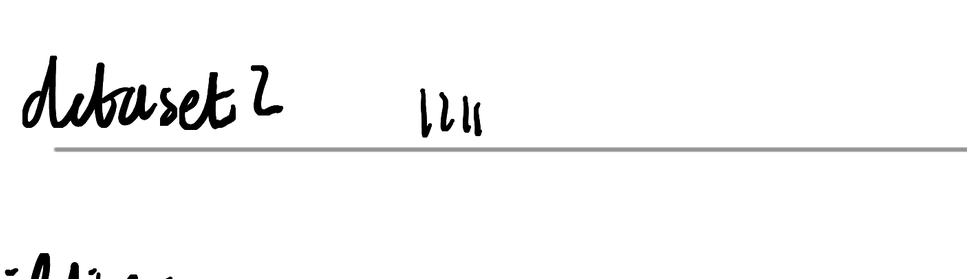
Thinking about how to get cluster errors without systematic errors.

We have the uncombined histograms & identifier

dataset 1	1111	1211	111
dataset 2	1211	1211	111
identifier	00000000	11111111	2222222

Convert this to

dataset channel : cluster x → [angles,] → dataset channel cluster Circular mean



Shift all dataset channel clusters to be centered on C_m, move points by mean difference of cluster mean center to dataset cluster center

dataset cluster counts per normalization

↓

we have weighted hists

don't need

→ merge datasets.

Then recreate histogram (reverse init) with errors

Mostly done just need to plot values + tests.

For one of 16 Sept - ccw

24_Sept-a2212_CW-1

Error looks somewhat high, need to check the histogram to check.

2/10/22

Plan for tomorrow:

→ Plot histogram + errors for merged translated datasets ✓

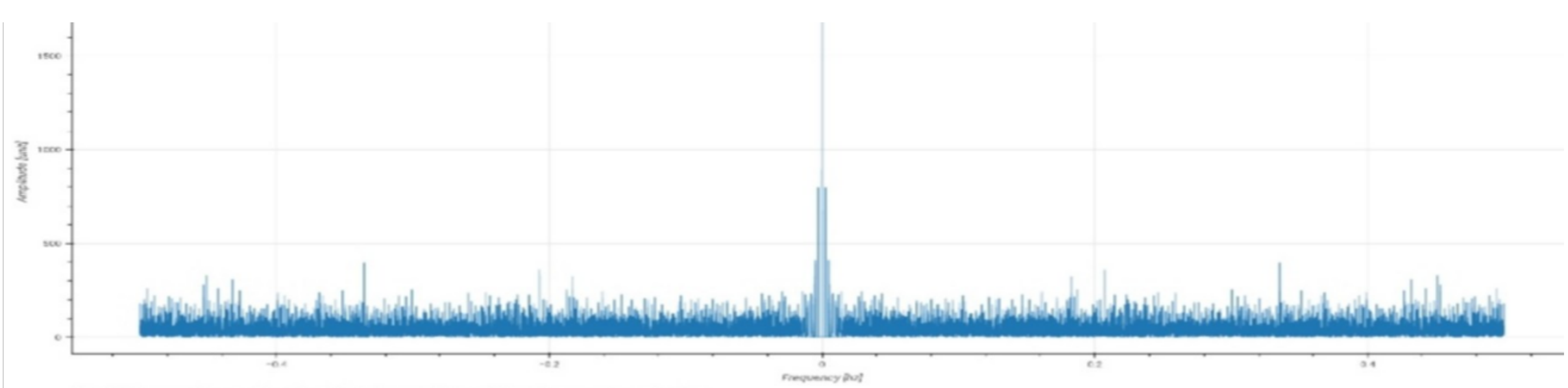
→ Fix bugs ✓

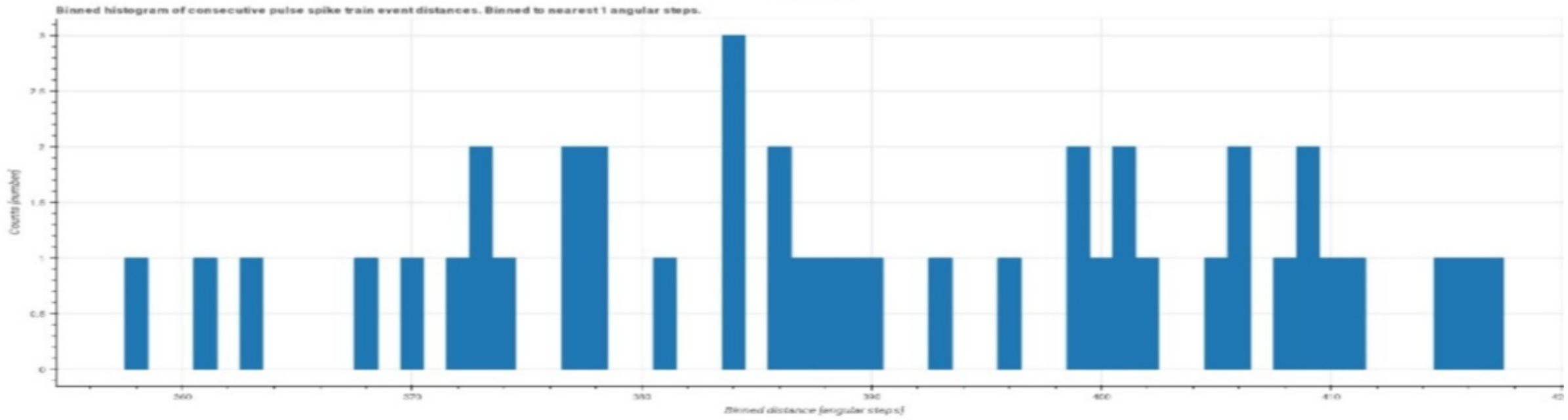
→ Create routine for running TDC collection for a fixed number of ticks. Host program multiplication, send the number of ticks via arguments. □

→ Finish routine which creates the ESC state up ☐ using circular queue of adjacent ZC events.

3/10/22

Finished translated histogram, need to negate translation rather than adding it.





Quantitative error analysis:

Cluster circular means and error

Mean cluster values and error indicates how well we know the location of the zero-crossing events, measured in [Angular steps]

Channel name	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Min error	Max error	Avg±std error	Min error % of ideal	Max error % of ideal	Avg±std error % of ideal
zc_channel_ar_data	14112.6170±124.0370	7121.9251±119.2398	2444.4419±109.8574	9494.9233±109.1508	179.2747±115.4289	4690.4734±121.2828	11734.8838±113.1370	109.1508	124.0370	116.0191±5.2937	27.9805	**31.7966**	29.7412±1.3570
zc_channel_af_data	15345.7793±187.3166	5883.7740±182.8516	10612.4580±173.4280	1334.7423±177.1812	8333.1955±184.0374	3549.4929±175.1113	12896.4734±178.1880	173.4280	187.3166	179.7306±4.7174	44.4579	**48.0182**	46.0735±1.2093
zc_channel_br_data	3172.1043±5.3437	10240.3975±5.5178	14936.5873±4.9961	7933.1039±5.9645	5484.5342±7.2939	12502.8851±5.7892	954.3577±5.2396	4.9961	7.2939	5.7350±0.7048	1.2807	**1.8698**	1.4701±0.1807
zc_channel_bf_data	6707.1475±41.8562	13707.0745±37.5573	2071.0931±34.6641	9109.3889±38.0131	4300.5924±32.8782	16158.1768±35.4017	11350.5429±39.2377	32.8782	41.8562	37.0869±2.7999	8.4282	**10.7297**	9.5071±0.7177
zc_channel_cr_data	10989.9241±5.7299	1708.2128±3.8161	6300.5940±5.7999	15756.7832±5.1319	13304.6125±5.4324	8722.7535±5.5221	3926.7893±4.5237	3.8161	5.7999	5.1366±0.6705	0.9783	**1.4868**	1.3168±0.1719
zc_channel_cf_data	566.9576±37.0307	9871.7574±35.2135	5085.7268±38.5561	14528.6790±38.7479	7531.6080±40.5783	12119.4142±37.7958	2813.8005±33.0995	33.0995	40.5783	37.2888±2.2909	8.4850	**10.4021**	9.5589±0.5873

Global error:

Global error from ideal symmetry indicates how consecutive zc-event distances deviate from ideal symmetry

- Ideal distance and average displacement from ideal value error: 390.0952±16.0629 [Angular steps]
- Relative error of zc-events from ideal: **4.1177** [relative error percentage]
- Actual measured mean and stdev of displacements: **390.0952±16.0629** | **100.0000±4.1177** [percentage of ideal]

Absolute displacement from ideal symmetry

- Min value: 0.2142 [Angular steps] | **0.0549** [percentage of ideal]
- Max value: 31.7914 [Angular steps] | **8.1497** [percentage of ideal]
- Average value: 13.8715±8.0991 [Angular steps] | **3.5559±2.0762** [percentage of ideal]

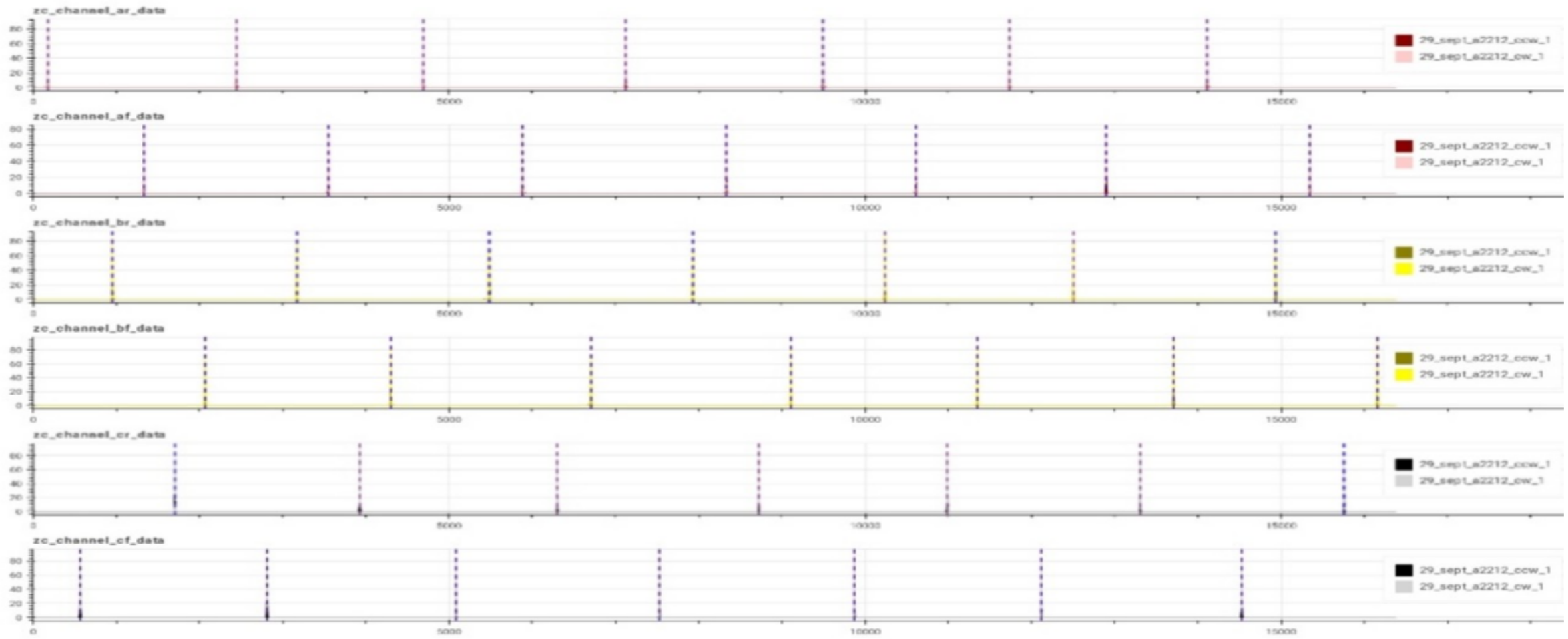
Displacement from ideal symmetry

- Min value: -31.7914 [Angular steps] | **-8.1497** [percentage of ideal]
- Max value: 26.7248 [Angular steps] | **6.8508** [percentage of ideal]
- Average value: 0.0000±16.0629 [Angular steps] | **0.0000±4.1177** [percentage of ideal]

After translating the cw/ccw runs the max error is dramatically reduced

Translated histograms

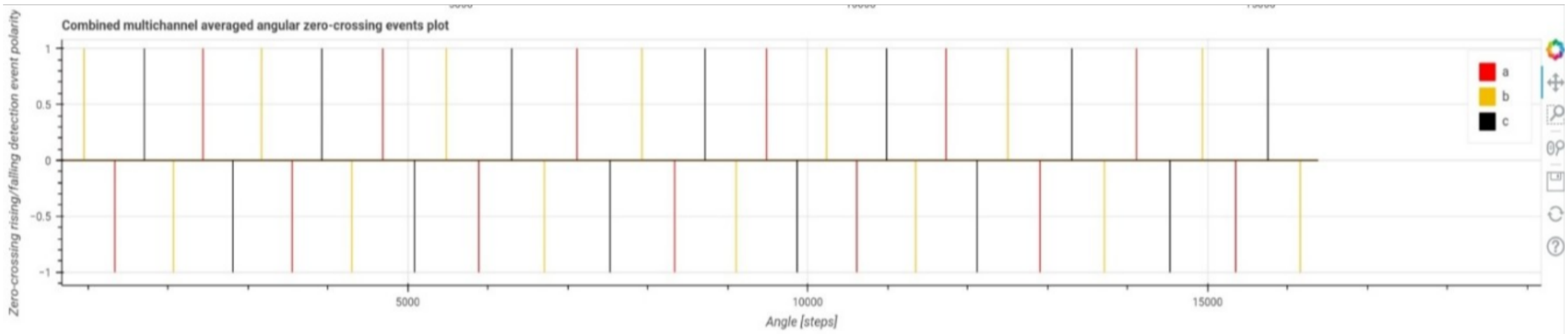
By calculating the distance from the mean centers of each datasets channel cluster to the mean center of the merged datasets channel cluster. The histogram values are translated and error recalculated, thus eliminating the systemic error.

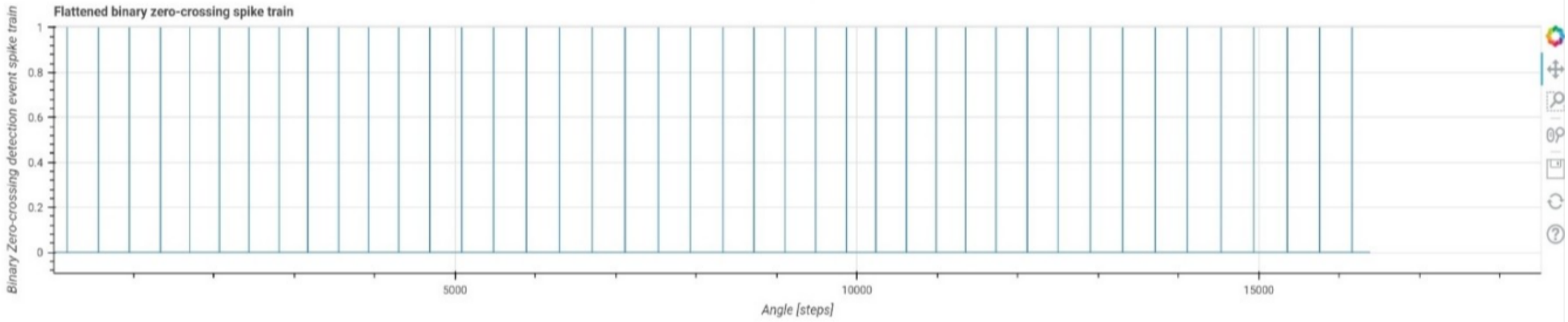


Quantitative error analysis of translated histograms

Errors calculated by translating each cluster.

Channel name	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Min error	Max error	Avg±std error	Min error % of ideal	Max error % of ideal	Avg±std error % of ideal
zc_channel_ar_data	14112.6170±5.6622	7121.9251±5.4831	2444.4419±4.9791	9494.9233±5.3468	179.2747±5.0677	4690.4734±5.1063	11734.8838±4.9239	4.9239	5.6622	5.2241±0.2570	1.2622	**1.4515**	1.3392±0.0659
zc_channel_af_data	15345.7793±5.5577	5883.7740±5.4595	10612.4580±5.3530	1334.7423±4.9853	8333.1955±5.2960	3549.4929±4.7732	12896.4734±4.7588	4.7588	5.5577	5.1691±0.3033	1.2199	**1.4247**	1.3251±0.0778
zc_channel_br_data	3172.1043±4.4547	10240.3975±4.4723	14936.5873±4.4923	7933.1039±4.8778	5484.5342±5.6964	12502.8851±4.5472	954.3577±4.7070	4.4547	5.6964	4.7497±0.4116	1.1420	**1.4603**	1.2176±0.1055
zc_channel_bf_data	6707.1475±5.4183	13707.0745±4.2709	2071.0931±5.0494	9109.3889±4.8261	4300.5924±4.9341	16158.1768±5.2075	11350.5429±4.6332	4.2709	5.4183	4.9056±0.3503	1.0948	**1.3890**	1.2575±0.0898
zc_channel_cr_data	10989.9241±4.8258	1708.2128±3.5278	6300.5940±4.9521	15756.7832±5.0658	13304.6125±4.6338	8722.7535±4.8334	3926.7893±4.4001	3.5278	5.0658	4.6055±0.4836	0.9043	**1.2986**	1.1806±0.1240
zc_channel_cf_data	566.9576±4.3949	9871.7574±4.7886	5085.7268±4.6613	14528.6790±4.5797	7531.6080±5.0926	12119.4142±5.0504	2813.8005±4.0328	4.0328	5.0926	4.6572±0.3436	1.0338	**1.3055**	1.1939±0.0881





Thinking about final ZC map (state map).

Need function to identify the state State (channel, rising/falling, direction)

If going CW then angular step ++, CCW --

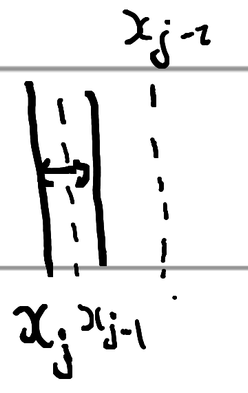
Midpoint is circular mean.

If CW

midpoint $(X[i], X[i+1 \bmod len])$ or $i-1$ $+ x_{i-1}$

If CCW

$-$ midpoint $(X[j], X[i=0])$ or $j-1$ $+ x_{j-1}$



5/10/22

Working on converting mean map into ZL CW/CCW map.

→ typing mean {"cloud": {0: 179.0}}

↓ ordered zip

[("cloud", 0, 179.0), ...]

take reverse for other direction.

Take midpoints of consecutive pairs, don't forget about idx=0 get last for the pair from end of sequence.

For range (distance) careful of -ve treatment

Mark angles displaced from c-v alue (angle) away & save

State

plot binary state map for cw/ccw ✓

Think we made a mistake with like state_map. I think each needs to be shifted by 1 step (respectively cw +ve ccw -ve)

Counter-clockwise rotation (-ve)

ESC	5	4	3	2	1	0
IN	C	C	B	B	A	A
SD	B	A	A	C	C	B
RISING	A	X	C	X	B	X
FALLING	X	B	X	A	X	C

This is expected to trigger this state

5 4 3 2 1 0

clockwise rotation (+ve)

ESC	0	1	2	3	4	5
IN	A	A	B	B	C	C
SD	B	C	C	A	A	B
RISING	C	X	A	X	B	X
FALLING	X	B	X	C	X	A

This is expected to trigger this state

0 1 2 3 4 5

(W correct ✓!

Rising / Falling indicates the NEXT expected edge.

Check code!

ccw

cw

state	5	4	3	2	1	0
rise/fall	↓	↑	↓	↑	↓	↑
phase detect	C	A	B	C	A	B

state	0	1	2	3	4	5
rise/fall	↓	↑	↓	↑	↓	↑
phase detect	A	C	B	A	C	B

think current version is correct.

Next steps:

→ Validate state map

→ Create cpp file with statemaps for ESC

→ Fix measurement in ticks 32 bits? (msg), end signal sync escape.

→ ESC program

→ New even measurements cw / ccw

→ ESC vib decode logic (interrupts!) correct, perhaps fix max duty electronically.

18/10/22

looking at how to do fixed length runs.

→ Need to modify state logic / sync program.

So on start procedure ... store number of seconds / ticks on each trigger then compare to final ticks, exit triggers & print END to host.

On the host we have the same program, need to take an argument for seconds to collect, encode this & send to teensy, but for each parsed line check for end message & force exit.

Had a go, need to test.

19/10/22

Created new branch FEATURES/uniform-run-time,

Moving away from merge-runs.

Checkout both computers to new feature branch.

→ To start only need to test ADC (easy)
sync / save program.

→ Flash new Keesy ADC code.

Circuit is broken... checking wires.

Main → HC-adc-scan1step

↓

ADC

a2212 _ Oct_19_NO-spin

~

a2212 _ Oct_21_cw_1

CW 3 CCW 5

uniform-run-time

urt2
is BROKEN!

Oct 22 cw 1

ccw 1

New Logic with uniform run 2 branch is very broken.
(but 2 decent run with a uniform run 1 uniform run 2 branch.) weird!

Need to try multiple runs with uniform run 2 branch, does seem to exhacerbate the systemic error. Things to try reduce serial writing to minimum, check escape condition 2 max ticks.

Need a run with few outliers, looking for short/long or long/short pattern.

Need to try plugged vs unplugged power supply

long run time needed ~ 5(s)

Simplify END print condition.

Ss, uniform run, keep going until we have long/short pattern.

Combine multiple times [3s, 5s, 8s] (watch out for 8s having multiple gaussians because of time slippage.

OCt-XX-CW-1S-2

XX-CCW-2S-1 etc.

Compare to best previous run, look for similar sinusoidfrom ideal spacing.

Make outlier detection more aggressive with longer runs.

22/10/22

Reverting to merge_runs branch as uniform_runs has very bad signal, much higher phase A. Trying to physically isolate circuit better... Need to see what diff from merge_runs branch to uniform_runs causes such an issue... could be the saved stuff, could be the exit condition. Baseliens seem lower quality than before, clutter error has gone up.

23/10/22

Revert to merge_runs → FEATURES/uniform_run_time3
Trying hard coding 4s (manual unplug after 6s)
See if this does not mess up the ADC measurements

a2212-oct-23_4s_cw_1, some noise, fails to cluster.

a2212-oct-23_4s_cw_2, no visible glitching this time, some bad clustering on a rising, attempting to re-run analysis. Failed

uniform-run-time 3

a2212-oct-23_4s_cw_3, good data ~ 7.9% error ✓

swap to uniform_run_time [1], remove terminal check & re-run. unplugged 3S_cw_1, list out of range, re-run analysis, plugged again

3S_cw_2 (won't cluster), 3S_cw_3 (more bad clustering!)

2S_cw_1 (unplugged!) worked ok (some outliers)

2S_ccw_1 (unplugged! still spun back (a few outliers)

Oct-23 25-CW-1 couldn't worked but not

25-CCW-1 amazing results.

checkpoint uniformation time C1] pushed dufcu84...

Always try unplugged go for 6s? make obbier

elimination more aggressive (number of student

deviation currently 6.3 → 101)

oct-23-6S-CW-1 unplugged! fail

CW-2

6s is not really working... bad clustering... revert to

45.

23 OCG- 4S-CW-4 unplugged!

unplugged? CW-5 good! unlucky point

Oct 23

_ HS_ccw_1 Not clustering

ccw-2 Not clustering

_3 bad result

_4 Bad result

_5

_6 good! (slight clump shew!

filtering idea remove all combs ≤ 2, most of the spurious points have very low combs, with a 4s run would be fairly unlikely to have a bad run. ✳

otherwise need to make Kalman more constrained.

Bad clustering is happening frequently, not enough clusters * should check this.

Thinking about it should being slightly higher. Could try & remove Time CIR prints & index the measurements within the same program just before transmission.

24/10/22

* Try & take new measurements as bit by filtering low counts ≤ 2 first.

Probably do this after zero-cross-detection, filter histograms & check distributions, all. ish ✓

24-Oct-45_cw_1 unplugged looks good
45_ccw_1 unplugged looks good
) combined ok!

Histogram appears quite truncated close significa
outliers from |6| -> 3

oct-24-4S-cw-2 unplugged combine ✓

ccw-2 unplugged

oct-24-6S-cw-1 unplugged combine

6 - 6S - ccw - 1 unplugged

oct-24-8S-cw-1 - similar

ccw-1 - similar

oct-24-6S-ccw-2 noise!

↳ by dijk

ccw-2
↳ bad result!

Repeat! 6S-ccw-3 ✓ 8S-ccw-2 ✓

6S-cw-3 ✓ 8S-cw-2 ✓

6 s - ccw - 4 ✓

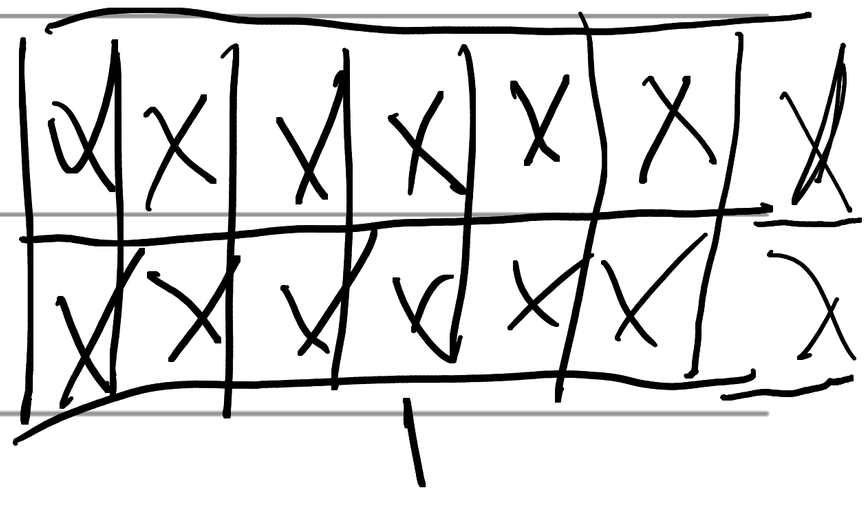
6 s - cw - 4 ✓

composed by results by more 3 s.

unplugged forced neck cpr

OCt-24-3S - cw_1 → 7

-ccw-1 - 7



reply

25/10/22

looking at Oct-24-3S results:

For cw: 1 & 2 are inconsistent with 3, 4, 5, 6, 7,

For ccw: 1, 2, 3, 4, 5, 6, 7 seem consistent.

The 3s results are much more consistant & less prone to artifacts (encoder / voltage glitches), so today collect more data. I still want to try multi collection times so rather than longer run times try more shorter runtimes. Later will try 10x cw 2s & 10x ccw 3s.

Before I attempt the 20x runs I wish to try the setup with the jupter driver for the encoder.

→ Backup (checkpoint code) ✓

→ Increase signal → SoO (oscilloscope) ✓ allow more

→ Use jupyter method + flush with optimised compile for speed Fastest

encoder

→ Trial 2s OCT_2S_CCW_1 closed chrome to see run unplugged + full cpu freq

... looks good proceed with large collection ✓

Oct-25-25

	1	2	3	4	5	6	7	8	9	10
CW	X	X	X	X	X	X	X	X	X	X
CCW	X	X	X	X	X	X	X	X	X	X

Set freq!

vuply!

26/10/22

(10)

Time to combine results, all results from Oct-25-25, from Oct-24-3s 3 → 7 (5).

Looking at Oct-24-4s 3, 4 (2).

Looking at Oct-24-6s all dippers

8s all dippers.

Time	2s	3s	4s	6s	8s
CW					
Pattern	SL				
consistency	10/10				
CCW					
Pattern					
consistency					
Note	Capture CW/CCW				

d - red

6 - yellow

C - black

R↓ B↑ Y↓

what should we combine,

1

cw3, ccw3

10/10 5/7 1/2

20 12 2/2

R↑ B↓ Y↑ (2s, 3s, 4s, ~4s) → sept 16

combined

not [illegible]

6 stable maps

combined

---

1 1

1/2 1/1

(inverted [illegible] / jelly)

2/2 1/1 → cw1, ccw1

R↓ B↑ Y↓ (6s, 8s)

cw1, ccw1

both

3 stable maps

B↓ Y↑ R↓ (6s) cw-4 1 stable

ccw-4 map

↑RIBT (6s) → cw-3

ccw-3

↓ (8s) cw-2

ccw_2

27/10/22

Implementing state map to cpp file 16,384

2Mbyte flash code

32 bit ints would be the fastest for 1024KB?

computation.

512K fast ~2^{22}

512K slow

$2^{21.9}$ lower

bit

$32 \times 2^{14} = 524,288$

$= 65,536$ bytes $= 2^{16}$ bytes

we have x2 of these $= 2^{17}$ bytes

If this does not compile do 8 bit char array & cast:

$\times 2^{14} = 16,384$ bytes

32 KB

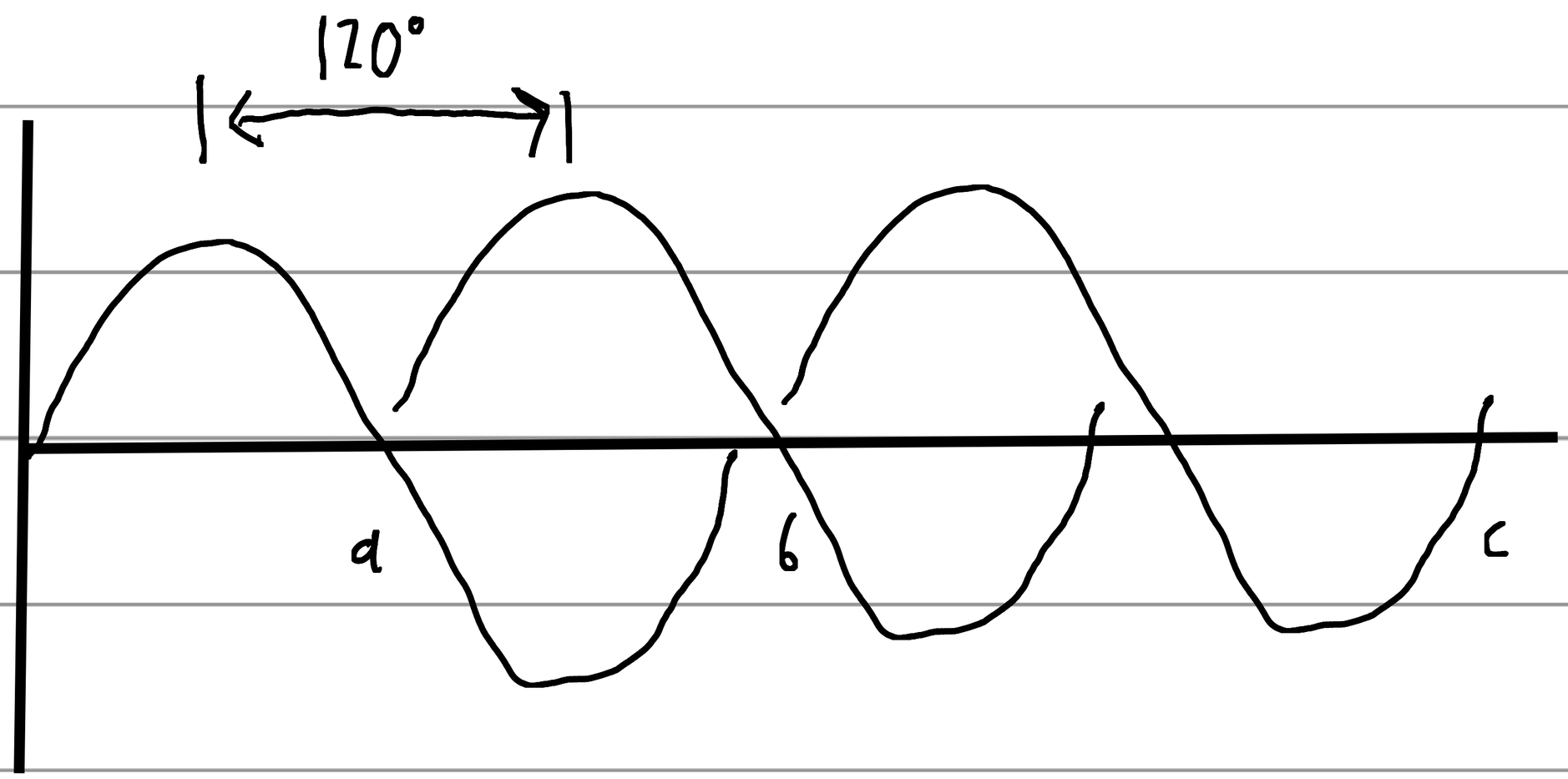
if <512 Kb

easily enough

131 kB

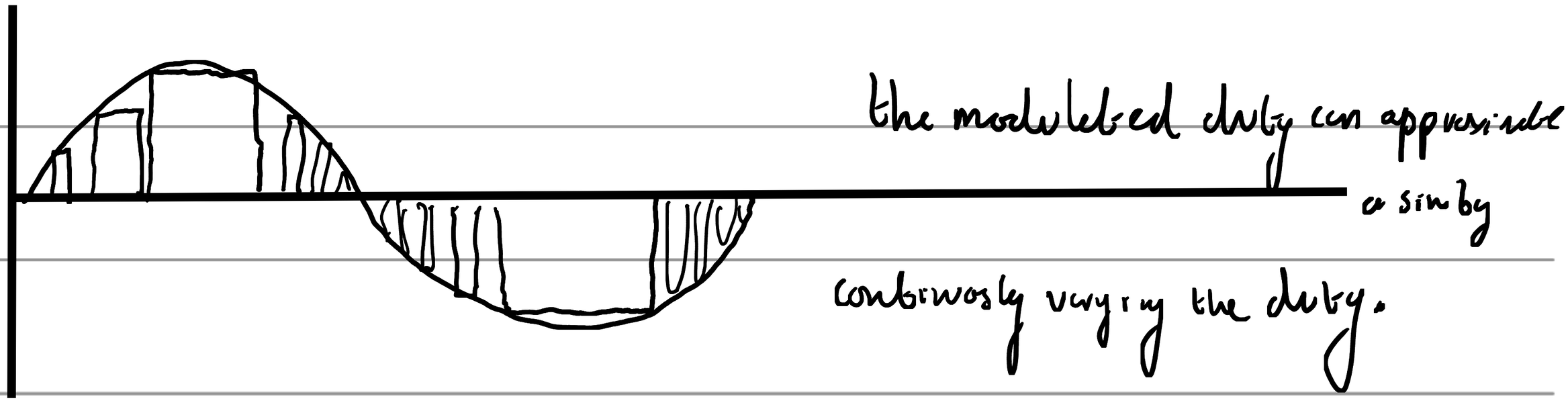
64 kB

Alternative simple sinusoidal commutation



change position of pulse A, both pulse A, B, C 120° separation.
to change position *

Constructing a sinusoid via duty cycle.



Where does the negative polarity come from?

*