

4/6/22

Config map

ADC library → check old lib with serial plot

Next:

- 1) Add hardware timer
- 2) Add trig 0 interrupt (TDC prior)
- 3) Attempt to hook up cmp as well
- 4) For each direction do a manual spin look for >0 Virtual neutrons
- 5) Binning software for calculating Zero Crossing angles.
- 6) Software to find transitions from ADC with Kernel

5/6/22

ADC -

→ Remove block commutation related code, except
single PWM

ADC	ACMP	Phase
14	21	A
15	22	B
16	23	C
17	18	VN

It has 3 identical chains

Reverse seems to work.

Does not seem to work with ADC & ACMP both at the same time
↳ investigate why

Trigger chain 4 readings via etc 0, 1, two readings each.

3 trigger chains for 3 phases.

Work todo:

1 GPT timer @ $n \times f_{adc}$ (n at least 2)

Only need 1 pwm @ frequency f_{adc}

Only need 1 trigger chain with etc 0 l 1

SPI read angle

GPT callback \rightarrow increments timer

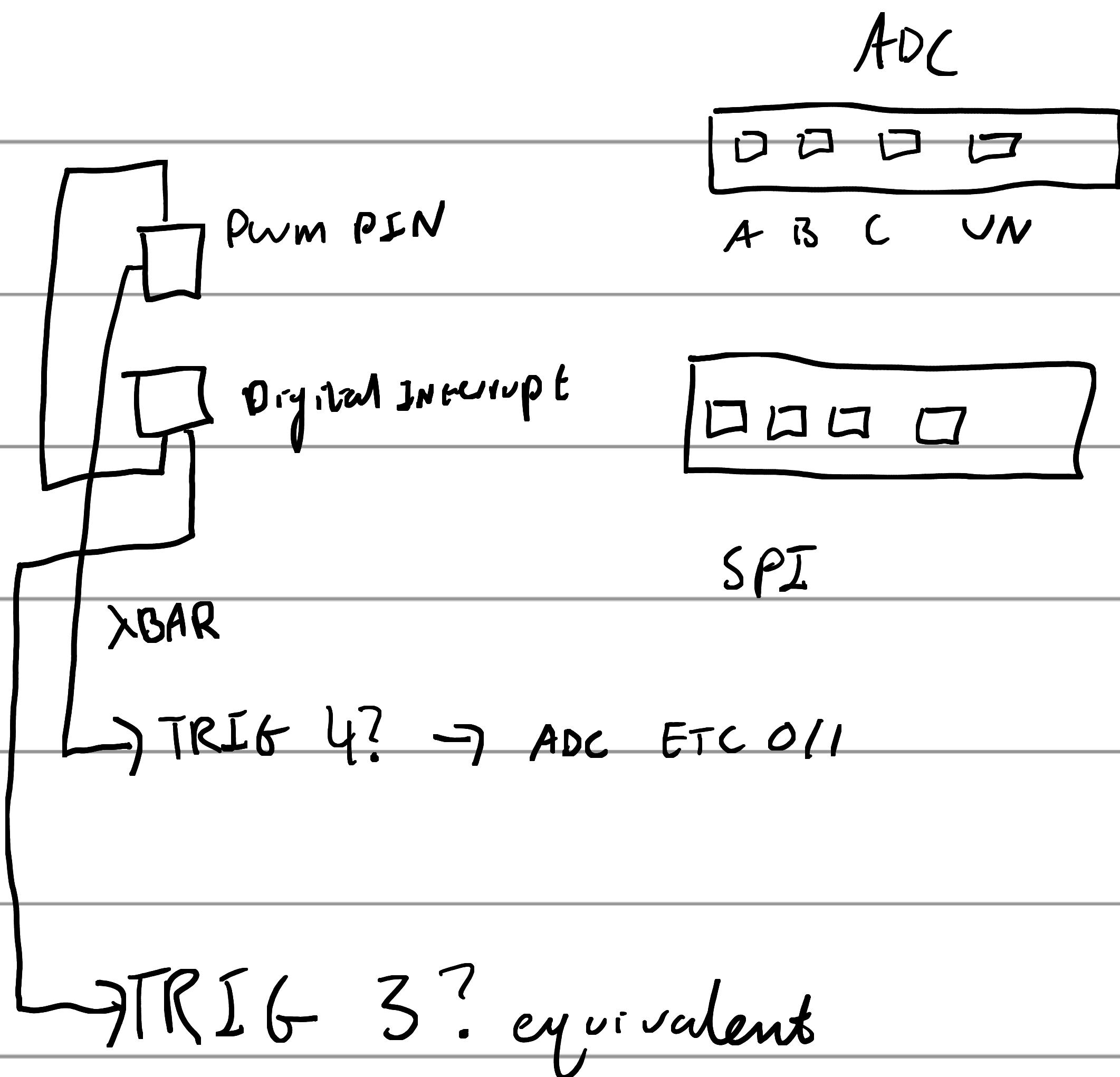
ETC-0 callback \rightarrow collect A & B voltages & record time

ETC-1 callback \rightarrow collect C & UN voltages & record time

Output

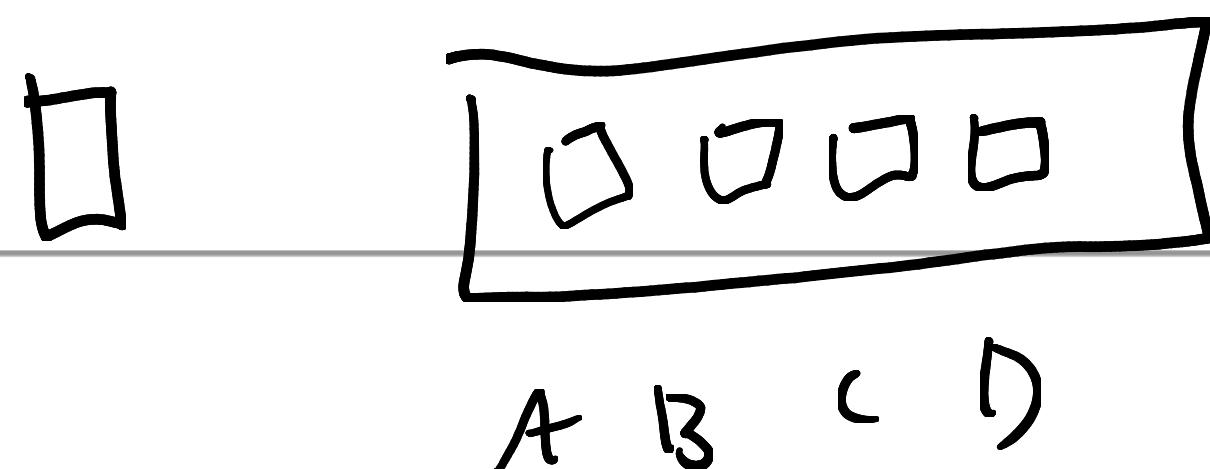
Voltages / time

SPI read in loop (perhaps avoid timeouts & only print to serial here) angle / time



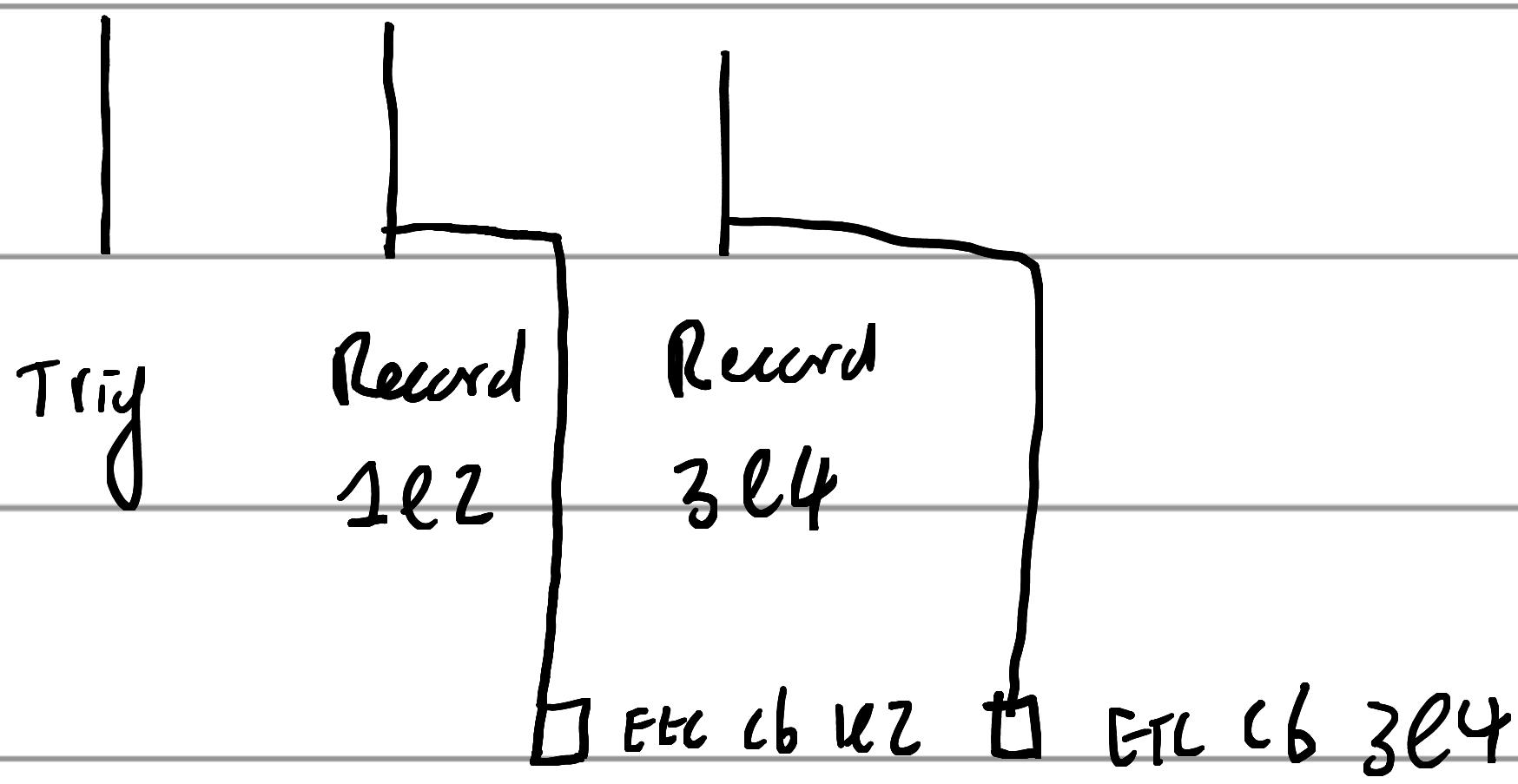
→ Record time before

PWM



Time callbacks

||||| | | | | | | | | | | | | |



↓
Record time

↓?
send time & voltages

(maybe on send in the
loop to avoid halting
interrupts)

loop

→ Send read command

SPI

→ record time just before clock pulse (closest temporally)
Q

voltages from latest ADC + its time.

→ print values to serial out

6/6/22

ADC: Simplest setup trigger ADC via PWM trigger, have time (tr increments on the 2nd ADC_EOC callback.

Implemented code to take ADC measurements ✓
Find Encoder
max sample rate
Add PS4 input code (controller)
Freq to high overloads
serial plot & monitor
... write custom code
Find highest serial out data rate

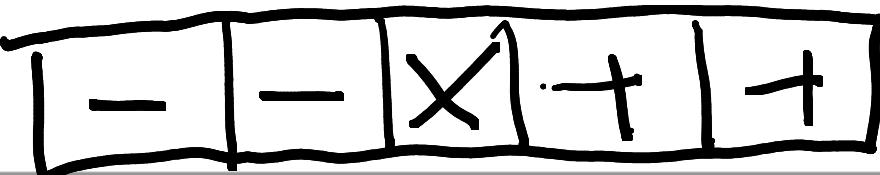
Refactored repository README.md in each directory.

7/6/22

Created assembly for a power drill to directly drive a blade motor. So i can take temp measurements & determine zero crossing points from a motor & create a calibration map

Next → Test ADC at higher frequency (custom logger?)
driven by a power drill.... hopefully the voltages are strong
enough.

→ write up SPI interface for encoder

→ Write code to detect & classify zero-crossings,
2D kernel?  Validate magnitude to
avoid noise... low value filter?

Need to validate to 6 steps follow in a consistent
pattern for forward or reverse directions.

Determine sequence has VN consistently > 0

→ Eventually would be nice to have an angle measurement
ADC
at about 30khz if voltages permit, or about 2 samples unique angle

longer term when encoder is finished build a histogram

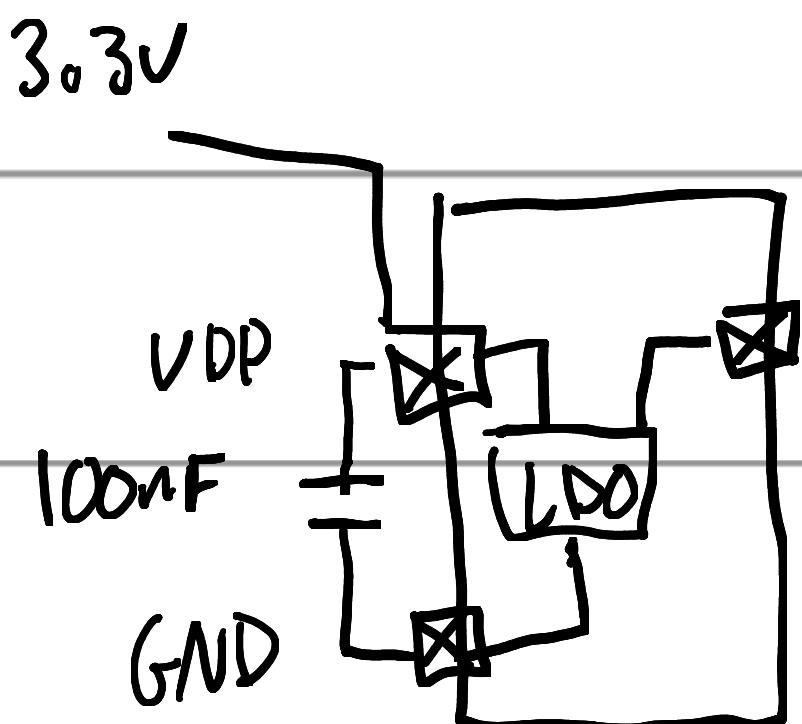
of angle vs zero crossing points, validate sequence, approximate error. etc...

8/6/22

14 bit encoder AS S147 P

		Pin	Name	Type (I/O)	Description
CSn	—	✓ 1	CSn	I	SPI chip select
CLK	—	✓ 2	CIK	I	SPI clock
MISO	—	✓ 3	MISO	O	SPI master data input slave out
MOSI	—	✓ 4	MOSI	I	SPI master data output slave in
TEST	—	X 5	Test	/	
B	—	X 6	A	O	Increment signal A
A	—	X 7	B	O	Increment signal B
		X 8	W1PWM	O	commutation W
Reserved	Available	X 9	✓	O	commutation V
swm	CS 10 green	X 10	U	O	commutation U
A _{in} 2	SCK 13 Blue	✓ 11	VDD	Power	5v power
A _{SD} 1	MISO 12 gray	✓ 12	VDD 3V	Power	3.3v power
B _{in} 9	MOSI 11 Purple	✓ 13	GND	Power	Ground
B _{SD} 0	SV is VDD	X 14	I	O	Increment signal I or PWM.
C _{in} 8	as labeled on encoder				
C _{sd} 7	Replace B?				
ADC	too noisy!				
A 14					
B 15					
C 17					
VN 16					
GND					

3.3 V operation

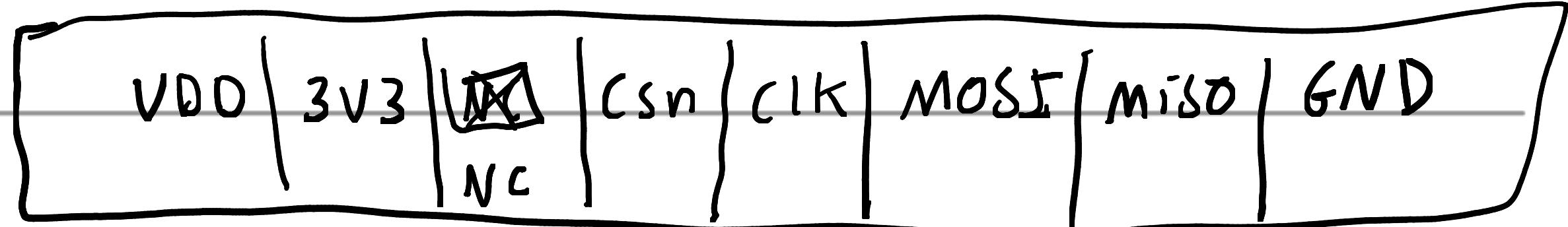


(SV)

100nF between VDD & GND

VDD & VDD3V bridged.

Breakout
pin layout

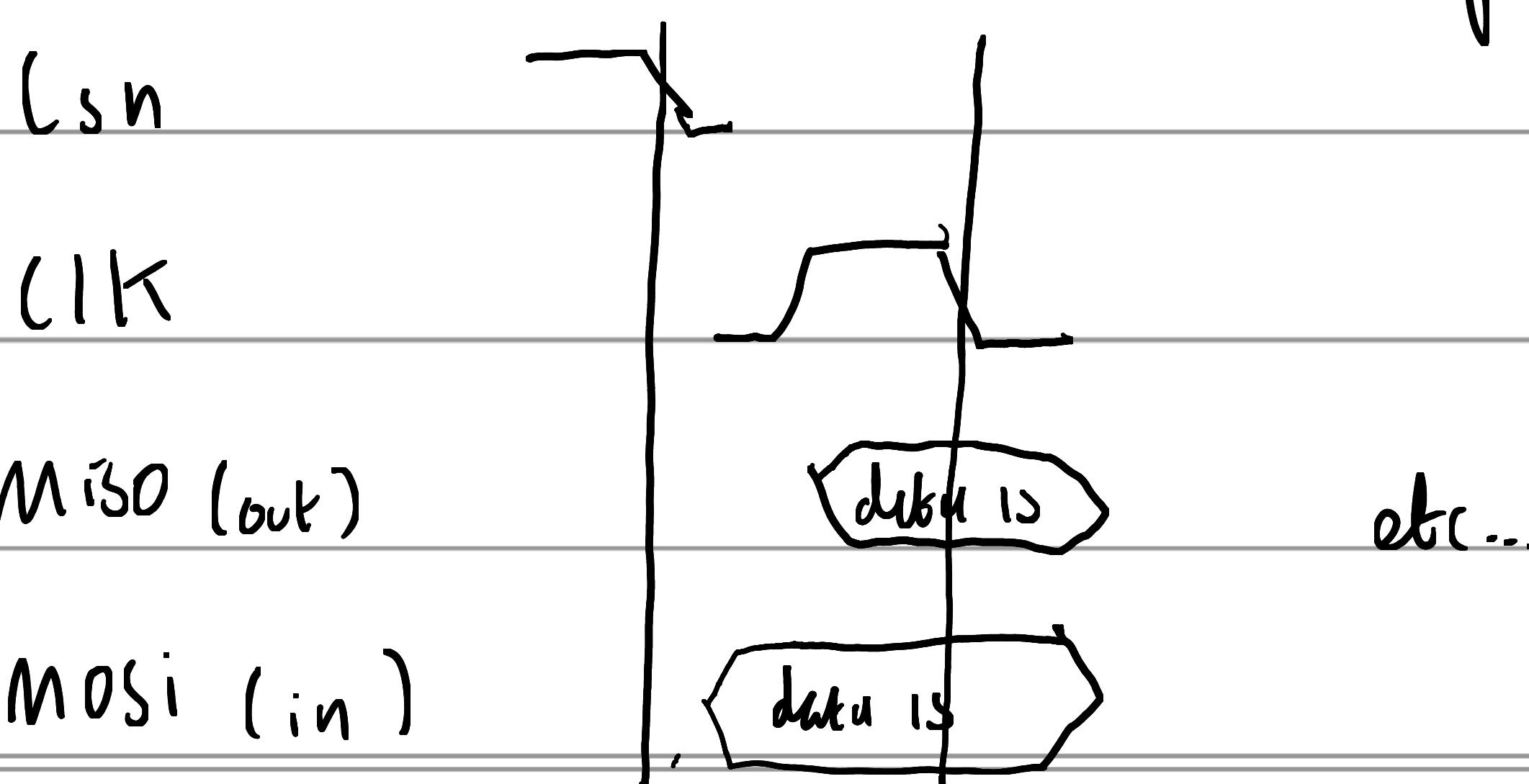


Data transfer:

Data transfer starts with falling edge of CSn (clk low)

MOSI (Master Output Slave Input) sampled on falling edge of CLK

Commands are executed at the end of the frame (CSn rising)



Procedure

Each frame CSn high → low

Each bit of command

- 1) Set CLK High → wait → low
- 2) Set MOSI high (Keep as command is just FFF...)
- 3) Read MISO bit

After all at end of frame CSn low → high.

Dont forget the value is only 14 bits!

Building little assembly to hold the magnet above the sensor, with screw nuts to adjust the standoff distance from the encoder. I can rotate the magnet by twisting the screw.

Sensor header pins soldered to the encoder PCB.

Sample rate

The drill drives the bldc motor at about $\sim 8 \text{ Hz}$

at $2^{14} = 16,384$ steps per cycle it would be nice to have an ADC sample at each step, or more if we also record the time to differentiate the order. So say we sample twice we would get $16 \text{ Hz} * 16,384$ we need a pwm triggering the ADC at $262,144 \text{ Hz}$.

What are we transmitting

time	phase A	phase B	phase C	phase VN	total Step
32 bits	12 bits	12 bits	12 bits	12 bits	14 bits

$$94 \text{ bits} \times 262,144 \text{ Hz} = 24,641,536 \text{ bits/second}$$

24.64 Mbps! $\xrightarrow{\text{(50% greater)}}$

without the time $62 \text{ bits} \times \dots = 16,252,928 \text{ bps} = 16.25 \text{ Mbps}$

In ASCII for convenience (without time) $4096, 4096, 4096, 4096, 16384 /$
 $20 + 6 \text{ bytes}$
 $32 \text{ bits } 4,294,967,296 \text{ 10 chars}$ 2086 bits
36 bytes 288 bits [75.4 Mbps] YIKES 54.82 MBps (maybe!)

Will depend on if the host can keep up otherwise it will make like

Teensy 4.0 wait. Will it buffer? Try & redirect bty to ram file, which
is 6Bps region. (stty raw; cat) </dev/ttys0xx something like that.

9/6/22

Finished assembly setup today, soldered together the cable which
connects the drill driven test bldc motor & finished the encoder
container which allows for a height adjustable screw which suspends
a magnet above the encoder.

Next...

- 1) Check wiring (Encoder, SMT \rightarrow Motor cables, SMT \rightarrow Teensy) ✓
- 2) check ADC code & hardware works. ✓
- 3) write encoder SPI code ✓
- 4) Test encoder works (debug etc) ✓
- 5) Find the limit of usb serial transfer (can we do it with ASCII at $\sim 200\text{ kHz}$?) ✓
- 6) Combine the ADC code with Encoder code ✓
- 7) Write some code to find the 6 states of the zero crossing & their repetition around entire rotation of the motor (angle), this code must validate, that the sequence is continuous, UN never drops too low, noise is ignored & the direction the motor was spinning during one run.
- 8) Write code which takes two valid datasets of a given motor, in opposite directions. Rotates each map from the zero crossing point to the ideal switching point ($\sim 30^\circ$ electrical) given the spinning direction. Superimpose these two maps & plot on a chart to check for symmetry.

10/6/22

Finished 1, 2, 3 & 4. Next a little refactor so I can do ⑥,
need a single configurable sample rate. Read angle in ADC interrupt?

Perhaps even printing to serial, need to fix clocking / sampling ADC to
ensure sync is not lost between readings EA, B3 & EC, UN3 which occur separately

Perhaps in ADC code need to subtract a zero point offset? check data
quality near just before drill pulse.

⑤ got encoder delay down to 3ns per bit to read otherwise parity fails.

serial times of value, parity \n in ~1 second of 373,120
↑
↓

payload

01e16384 \n

1111111 8 bytes 64 bits \times 373,120 s⁻¹

> 300 kHz

but only with
angle.

need ~ 262 kHz

23,879,680 23.9 Mbps. If we assume this
is the teary data output rate, then non ascii is under
at 16.25 Mbps & with time is close!

lets see, probably with
custom encoding.

Payload

can just write a buffer using `Serial.write(buffer, len-buffer)`

14 bit angle uint (probably 16 bit for convenience)

1 bit encoder parity

(4) x 12 bit ADC uint 48 bits

63 bits or 65 bits with 16 bit angle

Say we log at 300 kHz then 19.5 Mbps, prepare for lots of noise!

11/6/22

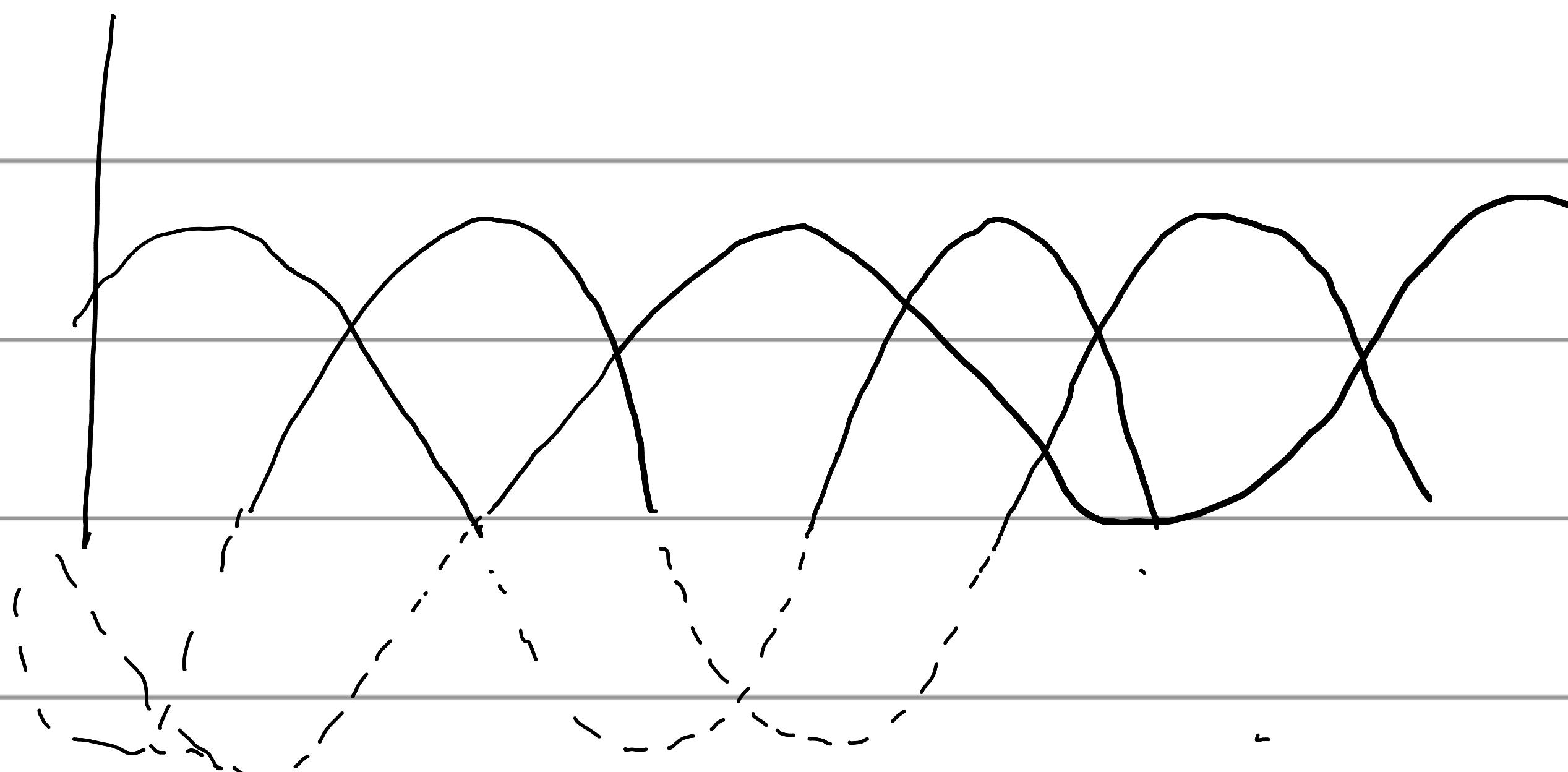
ADC + Encoder combined

Implemented ⑥ I refactored scripts thinking about ⑦
need a new python library to look at zero crossings.

ADC VN channel just looked like noise... try swapping VN to another channel

So V_N is very noisy depending on the change of position of the encoder. Guess as the data transfer via SPI is creating noise in ground & thus V_N .

One could perhaps approximate V_N as $V_N = \frac{V_a + V_b + V_c}{3}$ so long as encoder noise is not affecting signal of phase A, B or C but i think V_N is most sensitive to noise in ground.



12/6/22

Coded up graph square to plot data gathered by adc + encoder library.

One channel C is lower than A & B, virtual neural is almost non-existent. Encoder is generating noise on ground.

Trying to estimate v_n as $\frac{A+B+C}{3}$, attempting to normalise A, B & C by dividing by their means.

Doing a Kalman filter for normalised signals, & non-normalised signals minus $\frac{A+B+C}{3}$.

Signal is pretty crap, removing the encoder does not stop channel C being lower... bug?

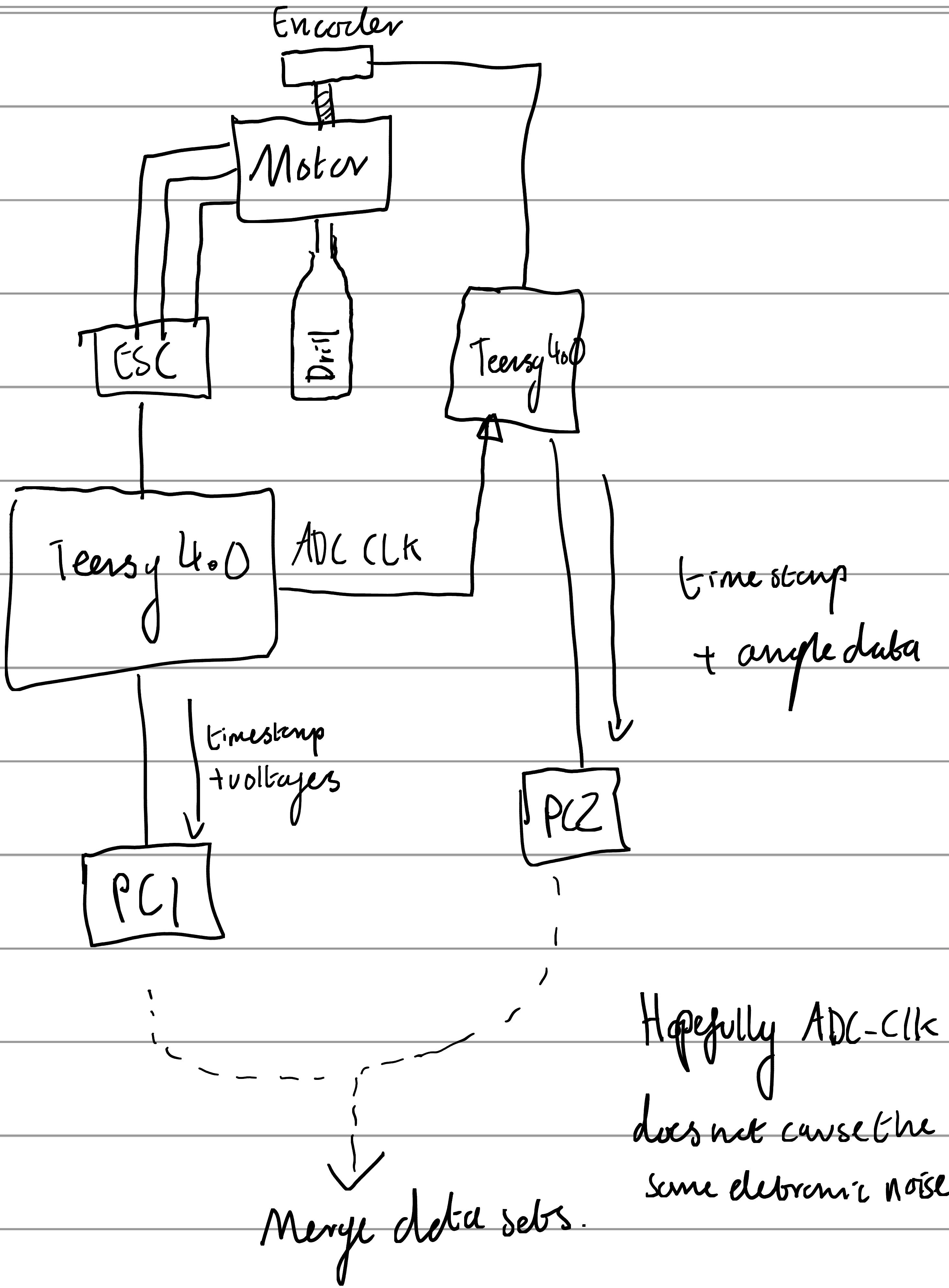
13/6/22

Trying old ADC code... it works! channels are balanced,
VN is valid and without noise if the encoder is absent,

After many hours debugging... channels are unbalanced
when serial logging the parity bit! weird! if only printing
angle, A, B, C & VN it works, angle is blank with
encoder driver code disabled it has good signal.

Created a new grapher to show phaseX_i & VN vs time,
phaseX_i - VN_i, VN_i & kalman of PhaseX_i - VN_i,
good results! Just needs the angle data & can have a go at the 1D kernalic

Conclusion... Need a second Teesry 4.0 to read the
encoder & ensure grounds are not shared. Use one Teesry 6.0
to trigger the other. Merge the dataset with software.



Need to order an additional Teensy 4.0

14/6/22

Thinking about processing dataset, presume
angle, d, b, (LVM) is the incoming data
Kalman filter all channels.

Next Kernel $\begin{bmatrix} - & - & + \end{bmatrix}$ rising kernel

$\begin{bmatrix} + & - & + & - \end{bmatrix}$ falling kernel

len kernel = 5 0 1 2 3 4 101 102 103
More complex - - 0 + + - - 0 + +

104 len data

Need at least 5 data points so first attempt

at $idx = 2$ final attempt at idx

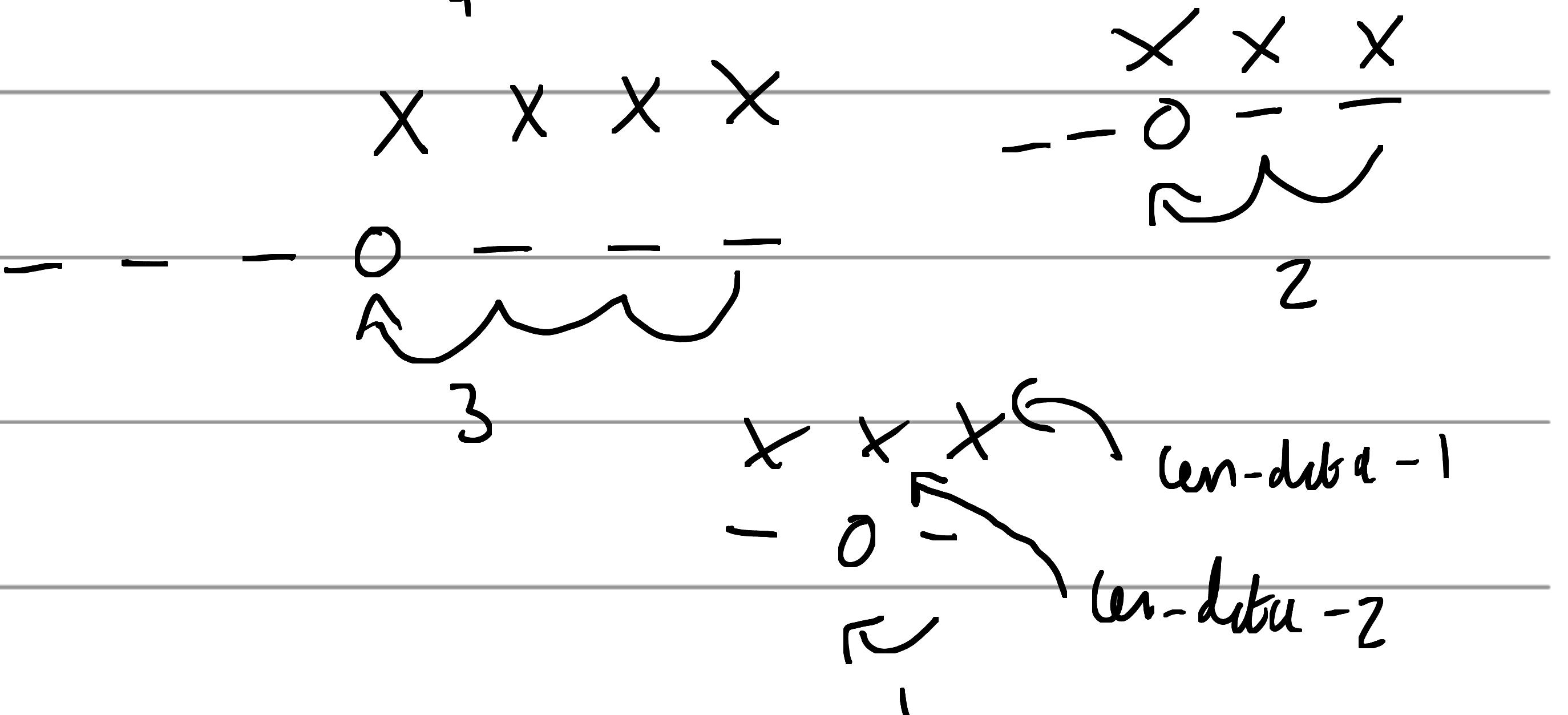
$idx > 1$

$idx > 101$

$idx > \text{len-data} - 2$

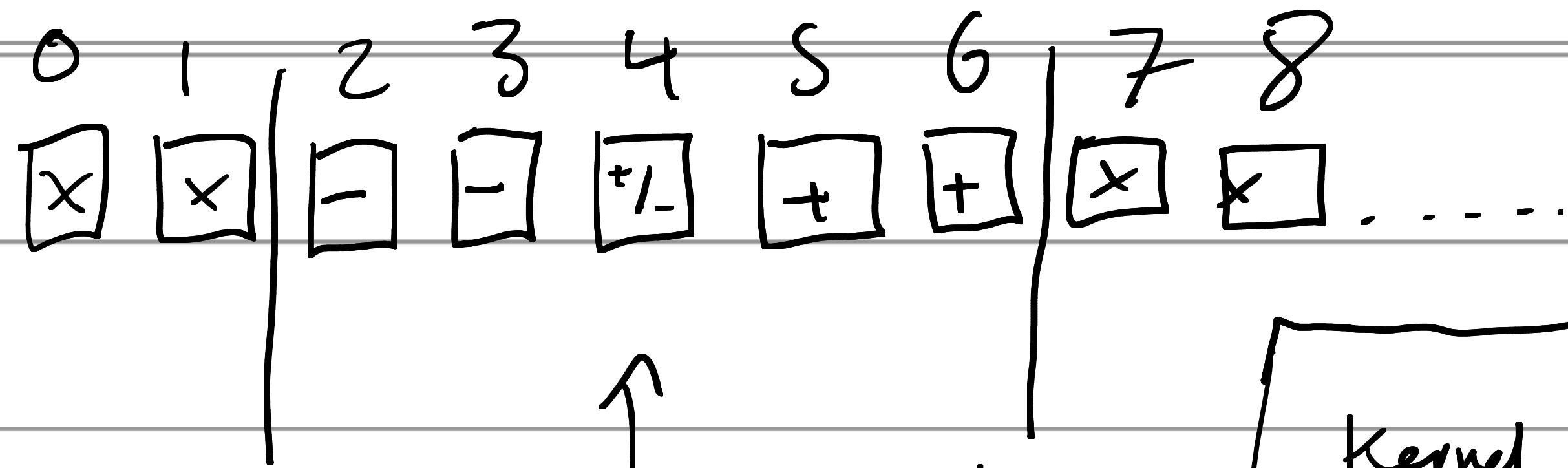
3	$idx \geq 0$	$idx \leq (\text{len-data} - 2)$
5	$idx \geq 1$	$idx \leq (\text{len-data} - 3)$
7	$idx \geq 2$	$idx \leq (\text{len-data} - 4)$

$\text{len-data} - 4$



3	1	$\underline{\text{len-kernel} - 1}$
5	2	= 2
7	3	

$\underline{\text{len-kernel}}$	$\frac{\text{len-kernel} - 1}{2} \quad \textcircled{A}$	start	end
3	1	$idx > \textcircled{A} - 1$	$idx \leq \text{len-data} - (\textcircled{A} + 1)$
5	2	$idx > \textcircled{A} - 1$	$idx \leq \text{len-data} - (\textcircled{A} + 1)$
7	3	$idx > \textcircled{A} - 1$	$idx \leq \text{len-data} - (\textcircled{A} + 1)$



↑ we are here

Kernel size = 5

midpoint = 2

idx = 4

2 3 4 5 6 data space

[0 1 2 3 4] Kernel space

Kernel
Input
idx=0

data - input idx = 4

idx

$$(0, 4) + \text{midpoint} \stackrel{?}{=} 2$$

$$K_{idx} (K_{idx} + \text{idx}) - K_m$$

$$0 (0+4)-2 = 2$$

$$1 (1+4)-2 = 3$$

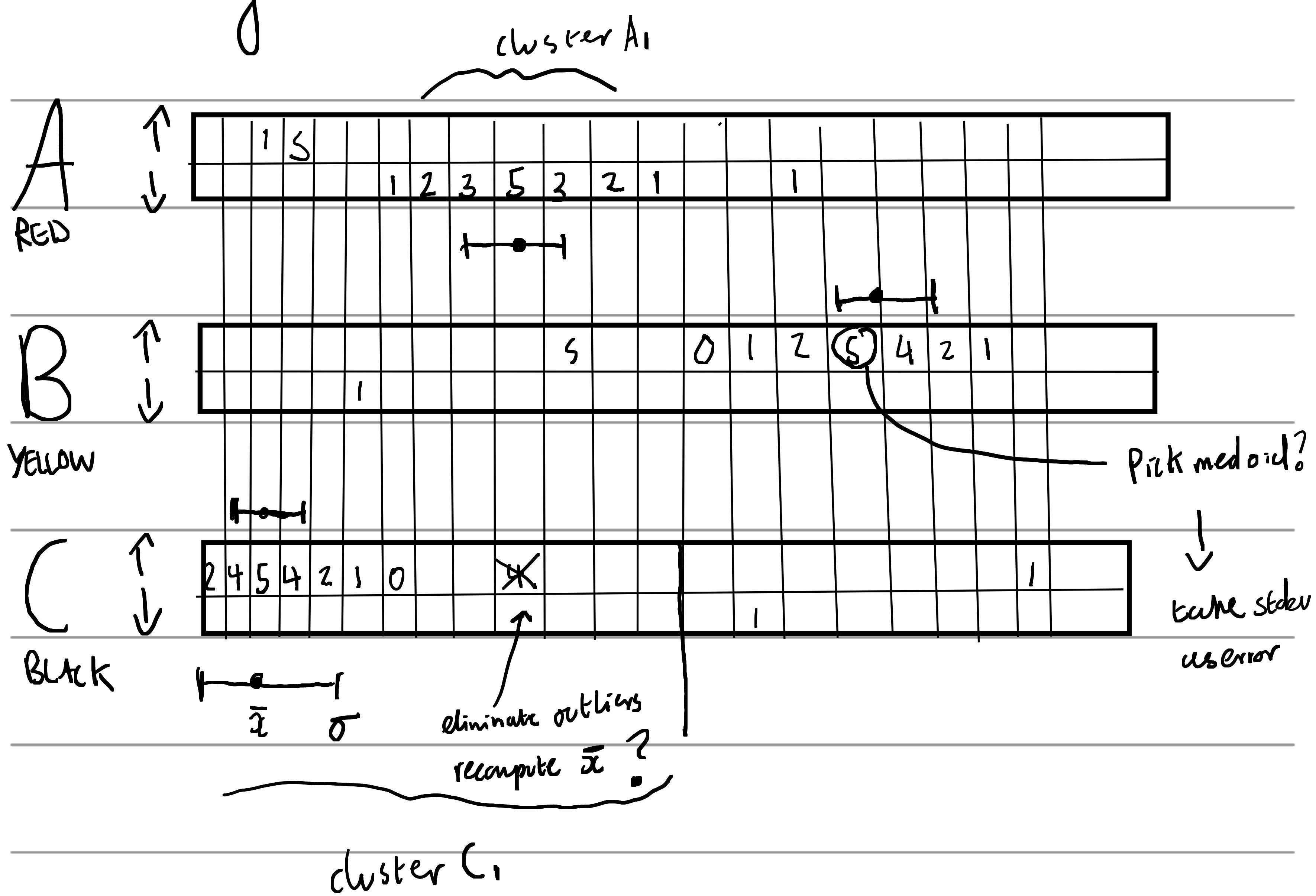
$$2 (2+4)-2 = 4$$

3

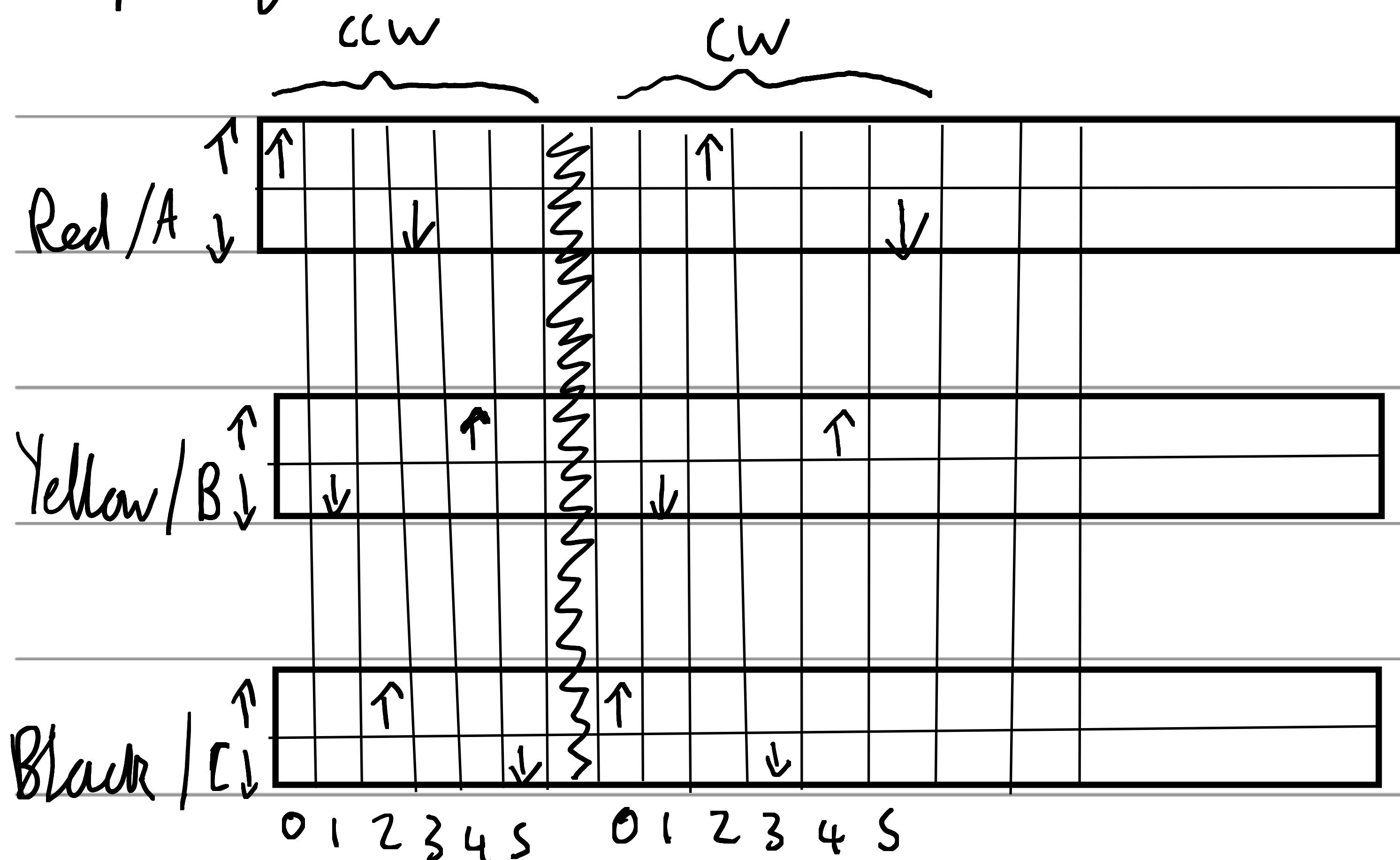
wrote draft of ZC kernel code,

Updated scripts & improved graphing. Estimating the angle as proportional to the time + white noise, have to wait for additional microcontroller to get real data.

Thinking about zero cross detection

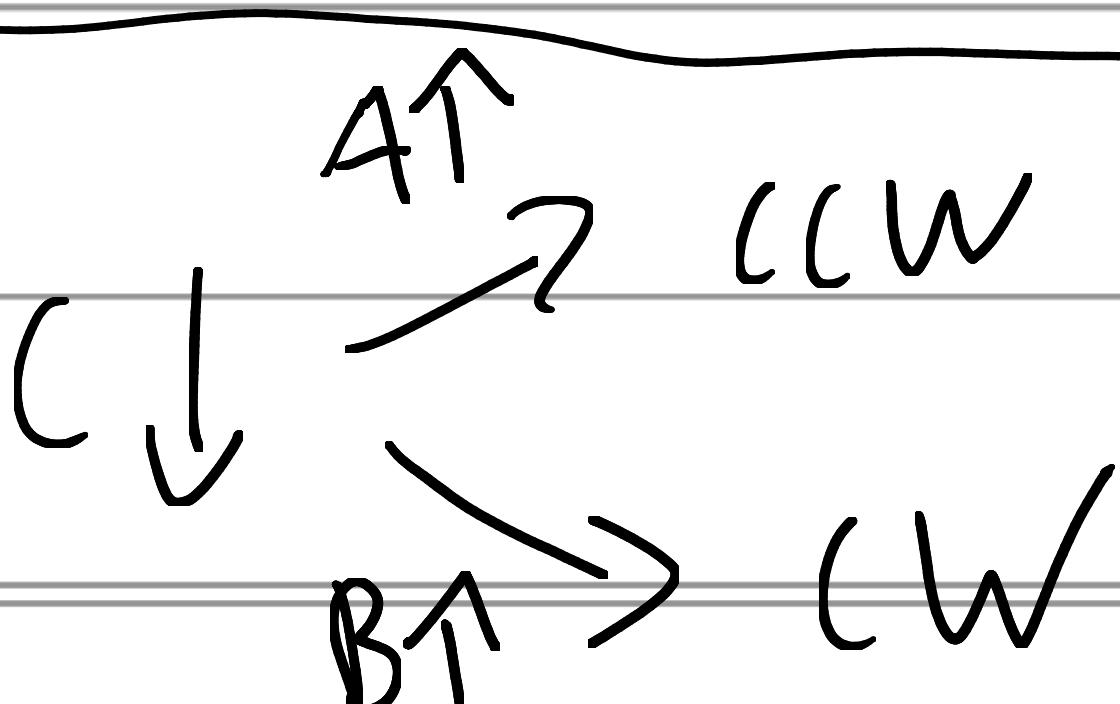
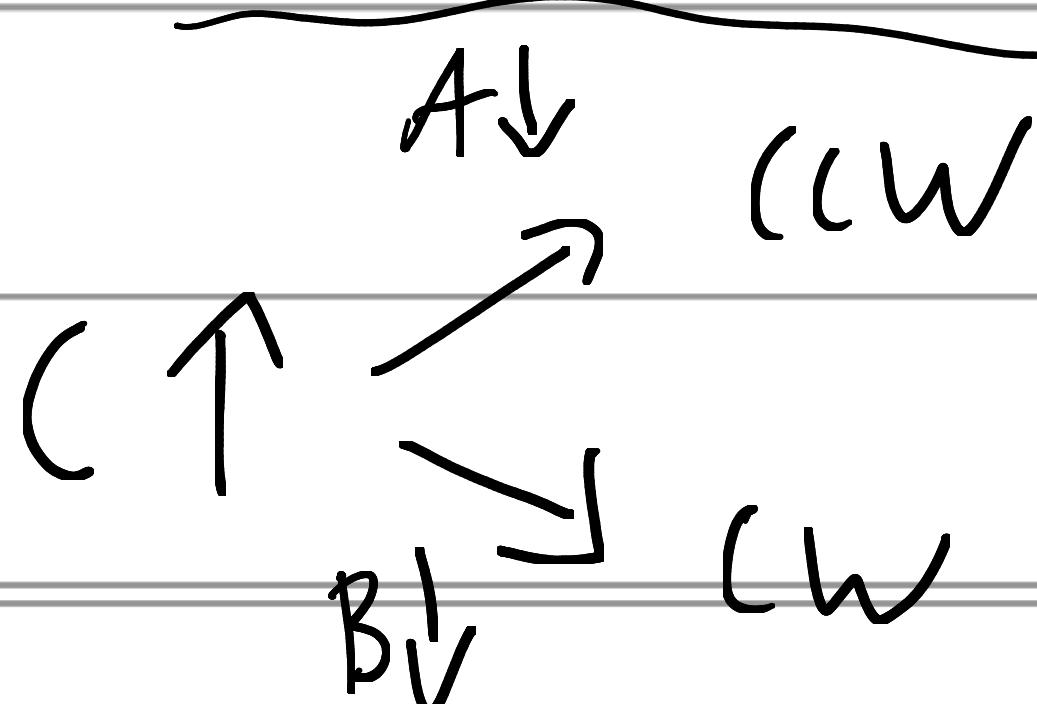
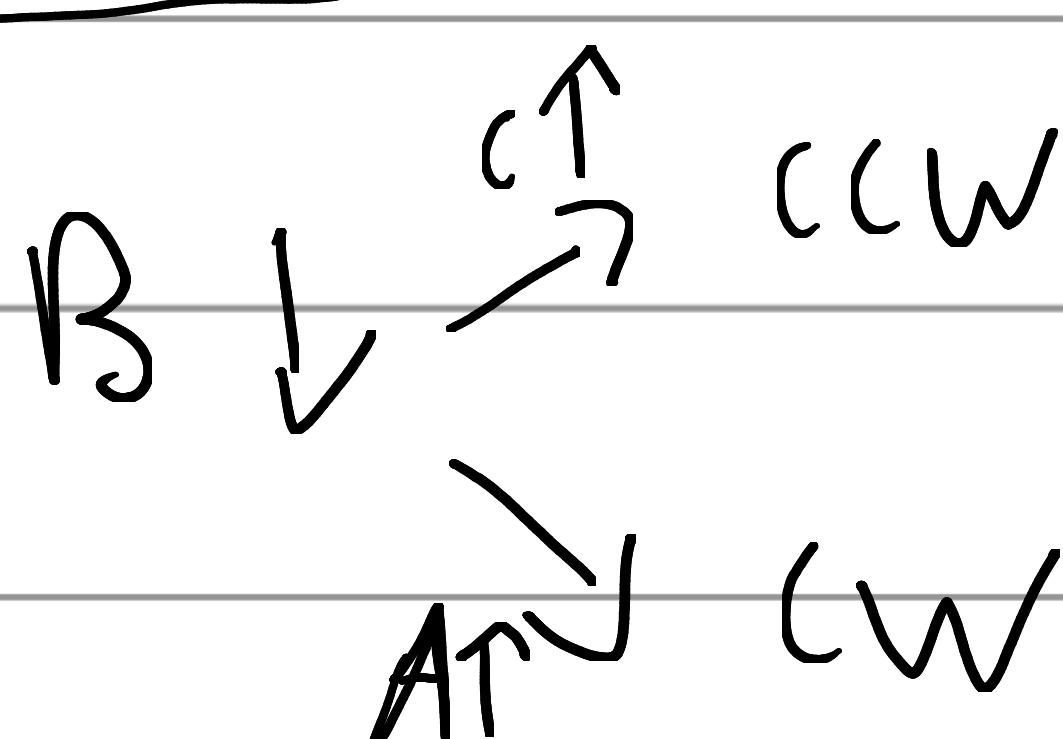
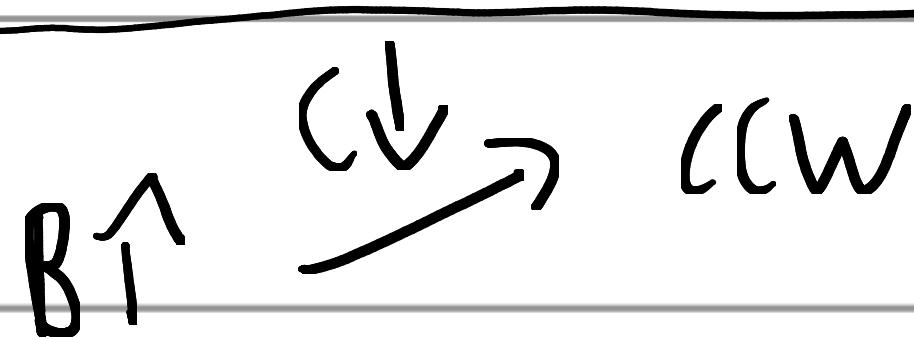
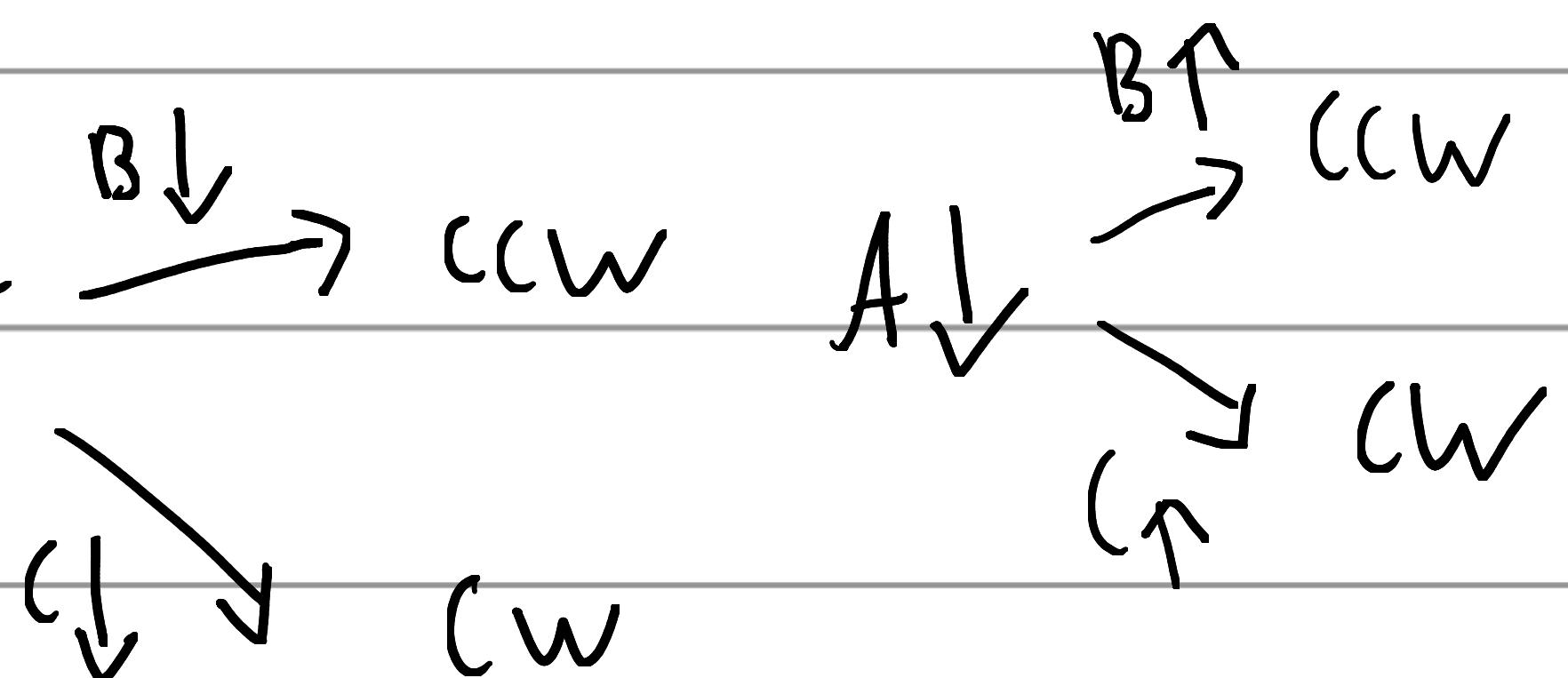


Complete sequence



How to validate:

if we find $A \uparrow$, what's next



So we have 6 crossings per electrical cycle $\frac{\text{Poles}}{2}$
electrical cycles per mechanical cycle. $6 \times \frac{12}{2} = 36$ total crossings.

$$36 / 3 = 12 \text{ crossings per phase}$$

We have 3 phases A', B', C' & two channels
+ve cross, -ve cross

$$12 / 2 = 6 \text{ crossings per phase crossing channel}$$

channel-ZC-cluster-number = $\frac{\text{Poles}}{2}$

1516122

Refactored ZC deblocking code, re-written in pyspark in case
data is very large.

(cleaned up script / file names.)

Clustering code is next... chart this data? Sanity check that ZC is in good place vs raw X_i -VN.

After clustering finalise the ZC angles, validate the sequence cw or ccw. Chart & display results.

16/6/22

Just cleaning up code today... chores. Had a look at spark on Windows, might need to install spark within Linux VM. WSL pyspark cannot use my existing network cluster as the VM has an IP separate from the host & thus network node cannot reply, bridge networking, add wsl to fire up?

17/6/22

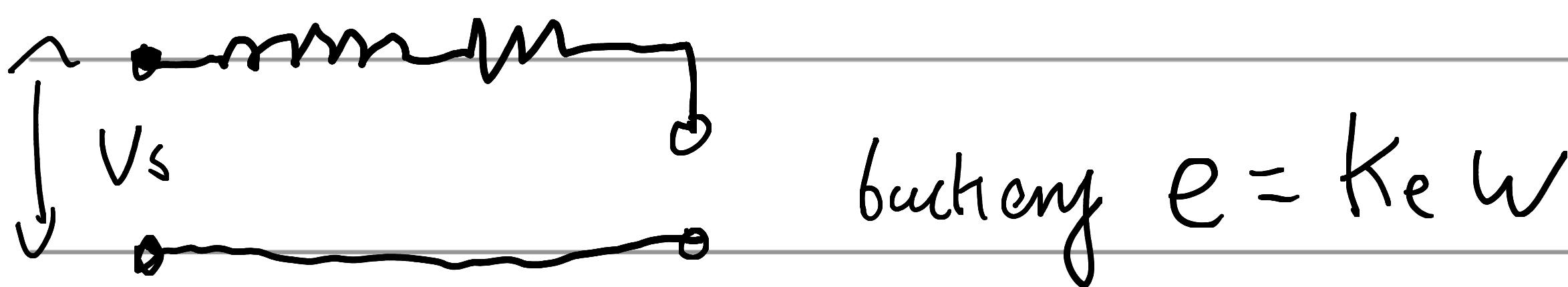
PID control parameters

Parameter	Rise time	Overshoot	setting time	steady state error
K_p	↓	↑	small change	↓
K_I	↓	↑	↑	eliminate
K_D	small change	↓	↓	small change

The goal is to obtain a fast rise, with minimal overshoot & almost no steady state error.

- 1) obtain open loop response
- 2) Add K_p to improve rise time.
- 3) Add K_d to improve overshoot percentage
- 4) Add K_I to reduce steady state error.

Transfer function



$$e = -Ri - L \frac{di}{dt} + Vs \quad *$$

rotor
Inertia load (J) Angular velocity w (T_i : torques)

$$J \frac{dw}{dt} = \sum_i^N T_i \quad *$$

TL mechanical torque
Te electrical torque

friction constant

$$Te = k_f w + J \frac{dw}{dt} + TL$$

* what is $\frac{di}{dt}$ $L \frac{di}{dt} = -e - Ri + Vs$

$$\frac{di}{dt} = \frac{1}{L} (-e - Ri + Vs)$$

$$= \frac{1}{L} (-Ke w - Ri + Vs)$$

$$\frac{di}{dt} = -\frac{iR}{L} - \frac{k_e w_m}{L} + \frac{1}{L} v_s$$

$$\frac{\partial w}{\partial t} = \frac{1}{J} \sum T_i \quad T_e = k_t w$$

$$\sum T_i = \bar{T}_e + \bar{T}_L$$

$$T_e = k_f w + J \frac{dw}{dt} + \bar{T}_L - \frac{1}{J} (T_e - k_f w - \bar{T}_L) = \cancel{\sum} \frac{dw}{dt}$$

$$+ J \frac{dw}{dt} = -k_f w + \bar{T}_L + T_e$$

$$\frac{dw}{dt} = \frac{1}{J} (-k_f w - T_L + T_e) \quad T_e = k_t w$$

$$= \frac{1}{J} (-k_f w - T_L + k_t w)$$

$$k_t w = 2T_L - k_{tw} + ik$$

Don't follow derivation but the result is

$$\frac{dw_m}{dt} = i \frac{k_t}{J} - K_f \frac{w_m}{J} + \frac{1}{J} T_L$$

$$\frac{di}{dt} = -i \frac{R}{L} - K_e \frac{w_m}{L} + \frac{1}{L} V_S$$

$$\int u \, dv = uv - \int v \, du$$

Laplace transformations by parts identity

$$\mathcal{L} \left(\frac{df(t)}{dt} \right) = \int_0^{\infty} \frac{df(t)}{dt} e^{-st} dt$$

$$\begin{aligned} \frac{du}{dt} &= -se^{-st} & \uparrow & u \\ du &= -se^{-st} dt & u' : & v = f(t) \quad dv = \frac{df(t)}{dt} dt \end{aligned}$$

$$\int_0^{\infty} e^{-st} \frac{df(t)}{dt} dt = \left[e^{-st} f(t) \right]_0^{\infty} - \int_0^{\infty} f(t) (-s)e^{-st} dt$$

$$= [e^{-st} \cdot f(\infty) - e^{0t} f(0)] + s \int_0^{\infty} f(t) e^{-st} dt$$

Laplace again (1st derivative)

complex

$$\mathcal{L} \left\{ \frac{df(t)}{dt} \right\} = \int_0^\infty e^{-st} \underbrace{u}_{u} \frac{df(t)}{dt} dt = F(s)$$

$$u = e^{-st} \quad \frac{du}{dt} = -se^{-st} \quad du = -se^{-st} dt$$

$$\int uv' = uv - \int u'v$$

$$\frac{dv}{dt} = \frac{d f(t)}{dt} \quad v = f(t)$$

$$\int_0^\infty e^{-st} \frac{df(t)}{dt} dt = \int f(t) e^{-st} \Big|_0^\infty - \int_0^\infty f(t) (-s)e^{-st} dt$$

$$= \left[\underbrace{f(\infty) e^{-\infty}}_0 - \underbrace{f(0^+) e^0}_1 \right] + s \underbrace{\int_0^\infty f(t) e^{-st} dt}$$

$$F(s) = \mathcal{L} \{ f(t) \}$$

$$\mathcal{L} \{ f'(t) \} = sF(s) - f(0) \quad \text{NEAT!}$$

$$\int_0^\infty e^{-st} \frac{dw}{dt} dt \quad \int u du = uv - \int v du$$

$$\frac{dw_m}{dt} = i \frac{k_t}{J} - k_f \frac{w_m}{J} + \frac{L}{J} T_L$$

$$U' = \frac{dw}{dt} \quad dv = \frac{df(t)}{dt} dt \quad v = f(t)$$

$$U = w ?$$

$$\int_0^\infty e^{-st} \frac{dw}{dt} dt = \left[e^{-st} w(t) \right]_0^\infty$$

$$- \int_0^\infty w(t) (-s) e^{-st} dt$$

$$w(0) + s \int_0^\infty w(t) e^{-st} dt = L \left(\frac{dw}{dt} \right)$$

$$= w(0) + s \int_0^\infty \frac{d\theta}{dt} e^{-st} dt = w(0)$$

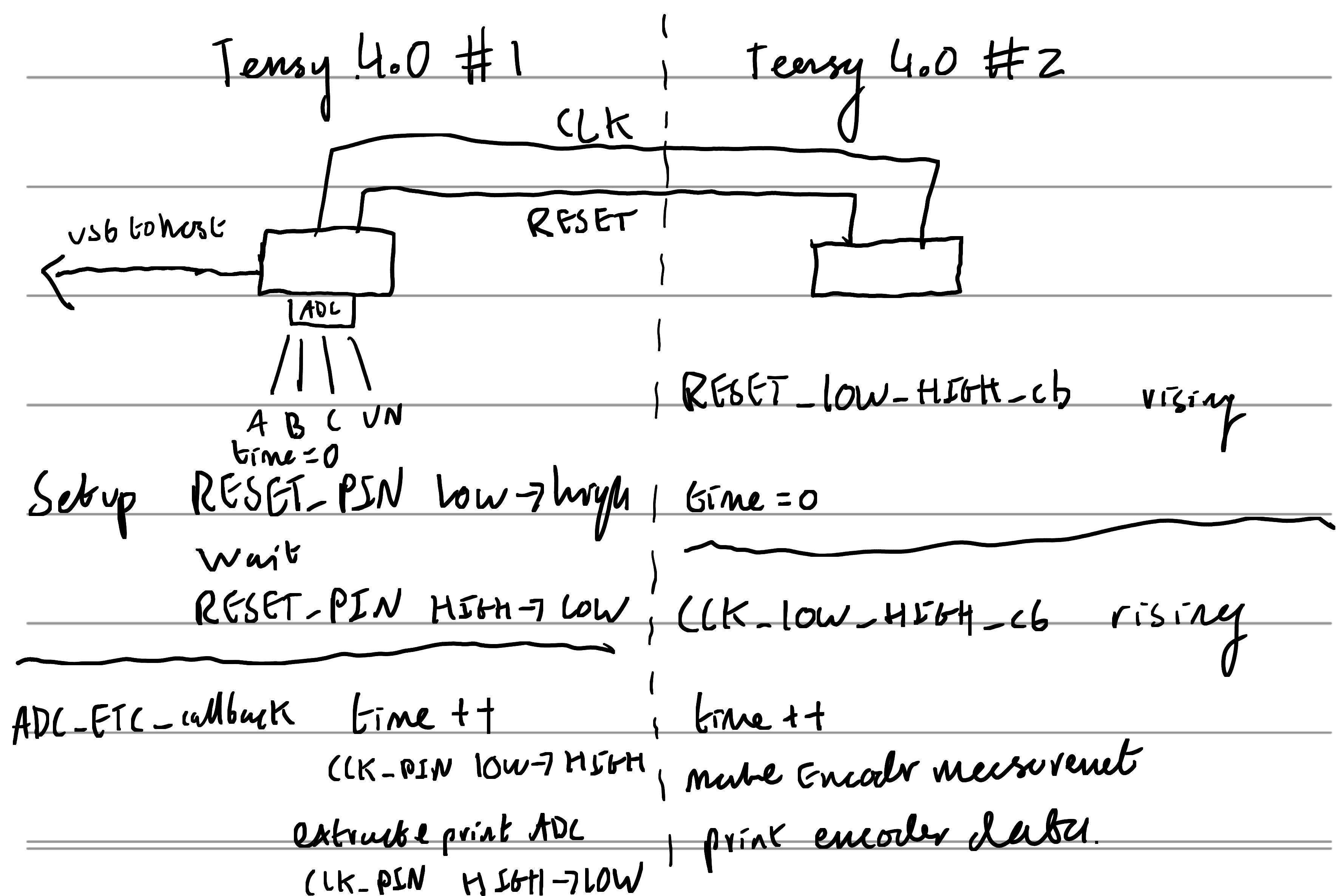
Inverse Laplace

$$f(x) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{st} F(s) ds$$

reminds me of the Cauchy integral.

2016/22

New teensy is here. Need to remove encoder code from ADC. ADC by itself but in callback send a pulse to 2nd teensy, need to have an integer representing the tick of time. we want a 2nd channel to send a pulse so when the first Teensy powers on send a pulse to reset the timer counter.



Draft code done. Both teensy's flushed with
correct hex.

Encoder is on /dev/ttyACM0
ADC on /dev/ttyACM1

Procedure

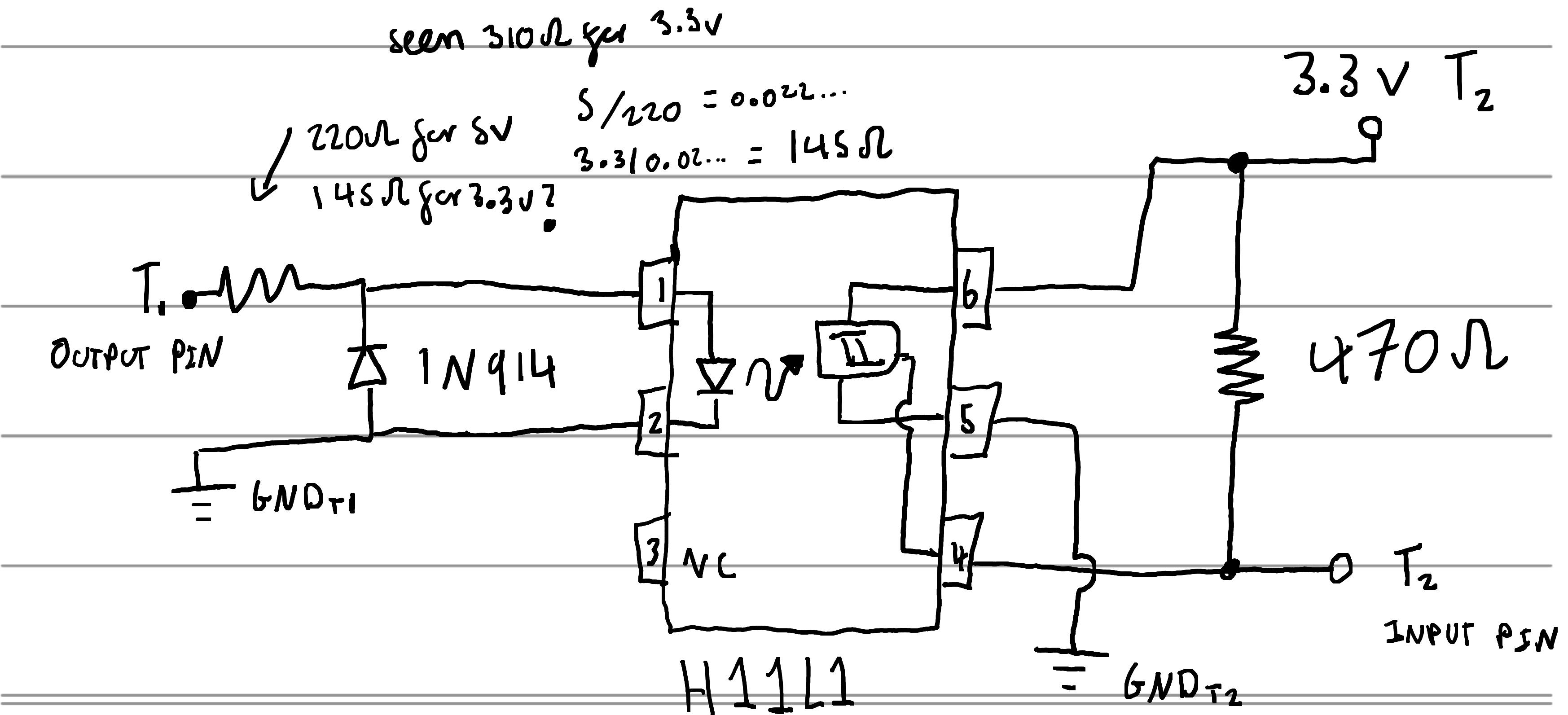
- Connect Encoder teensy
- Start 2 instances of serial Plot
- Bind first serial plot to /dev/ttyACM0
- Disconnect ADC teensy
- Connect ADC teensy
- Bind 2nd serial plot to /dev/ttyACM0

Results & thoughts

Noise from Encoder still present in TDC measurements if both
beersys are plugged into same laptop USB & grounds interact.

Noise is gone if using second laptop but digitally interrupting
the slave beersy does not work. Need to trigger slave beersy
without sharing ground, galvanic isolation. Best option is
probably opto-electrically.

Isolator circuit using H11L1



21/6/22

Shopping list

H12L1 opto-isolator x 2

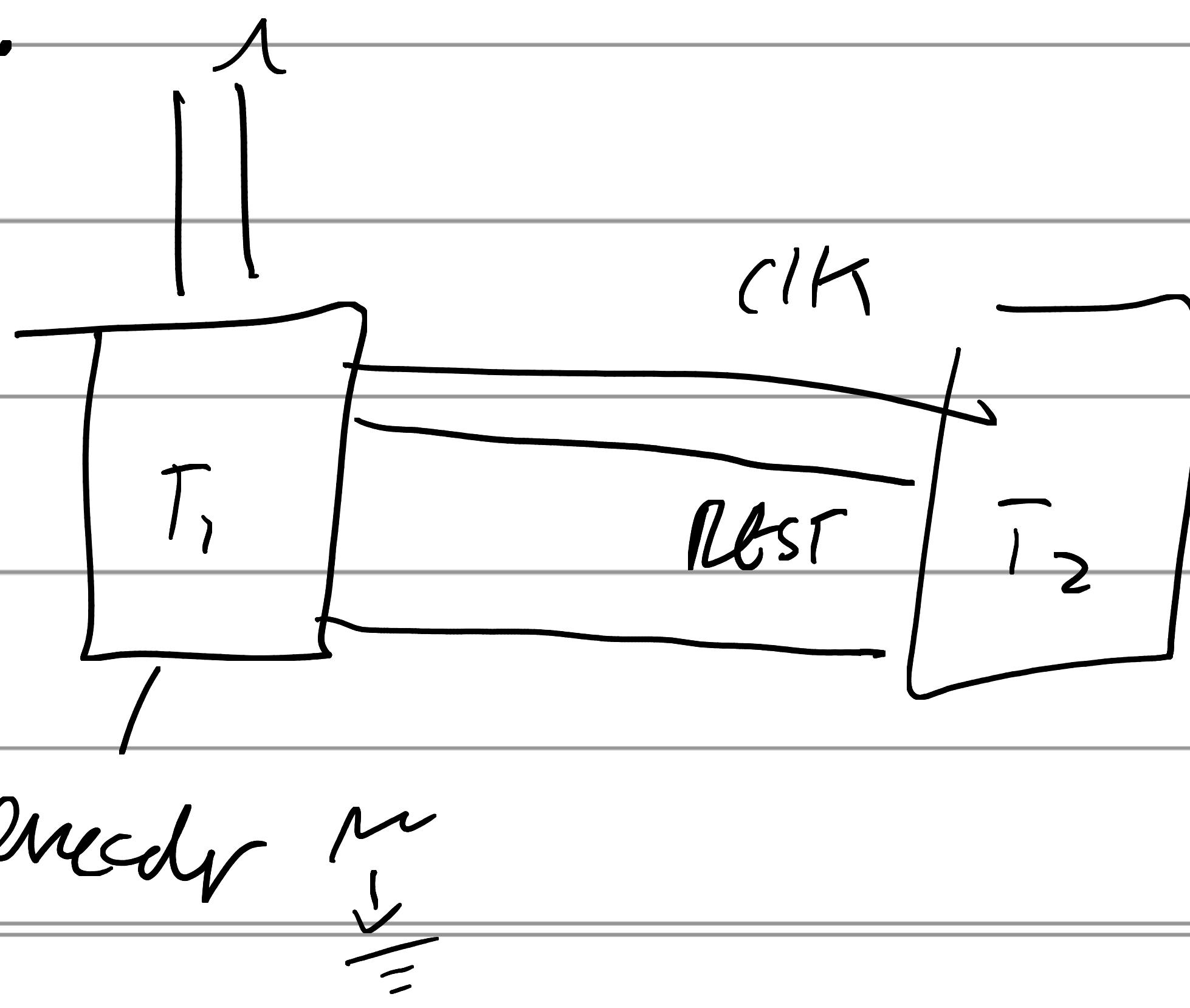
1N914 diode x 2

470Ω resistor x 2

220Ω (5V) or 145Ω (3.3V) x 2

Attempted to fix some obvious bugs, ctr for time was tripping twice for each high signal to slave.

Shopping list ordered.



29/6/22

Optoelectronics have arrived. Have multiple resistors

To try should try highest Ω resistors first & only
if they don't work try lower resistances.

Until resistances are known then hardware is at risk.

To avoid destroying microcontroller the plan would be to
get one teensy 4.0 communicating with itself. This
could be a good way to pre-validate that the CLK
pulse is not firing erroneously via noise & introducing error &
the time getting out of sync.

Timer 1 GPT $\sim 60,000$ Hz

↓ SET CLK Low (opto inverts input)

wait $\sim 4\text{ns}$

SET CLK High

tick T_1

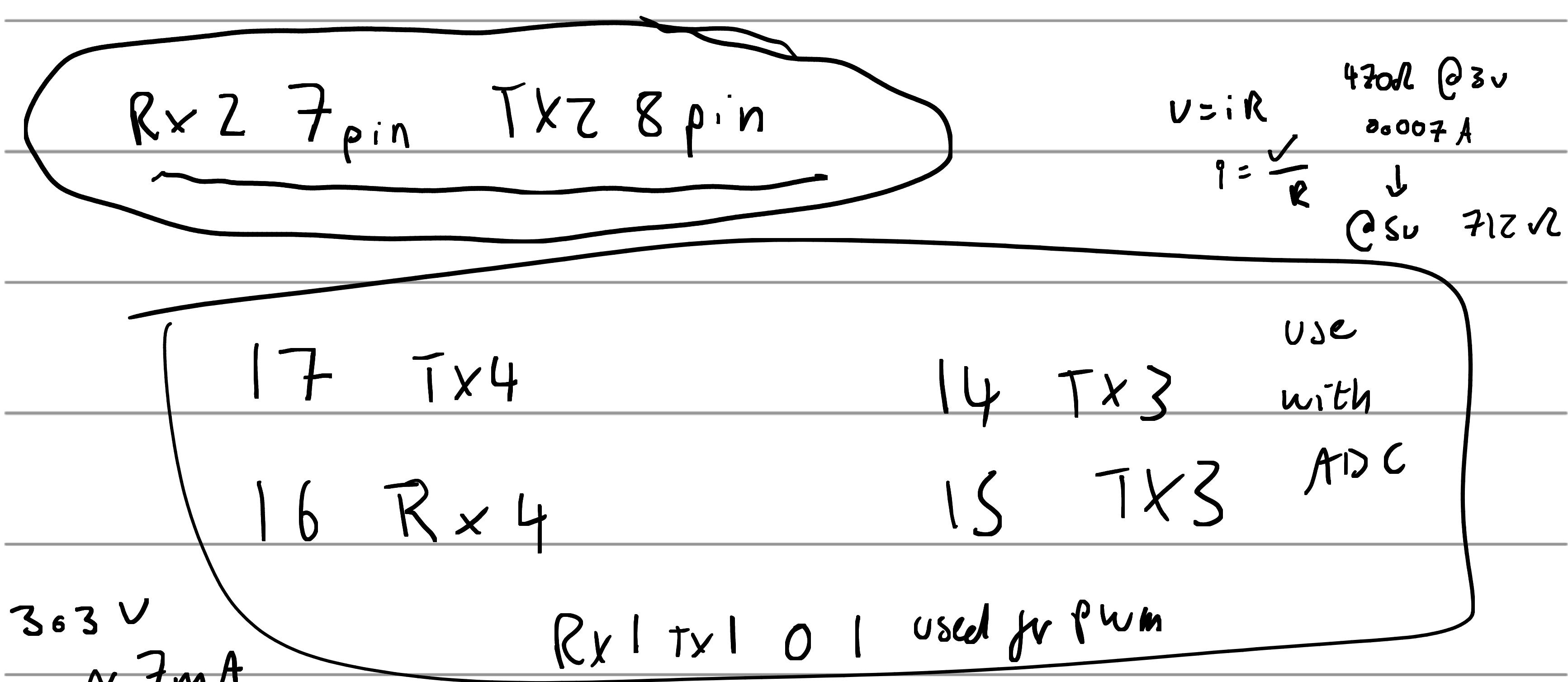
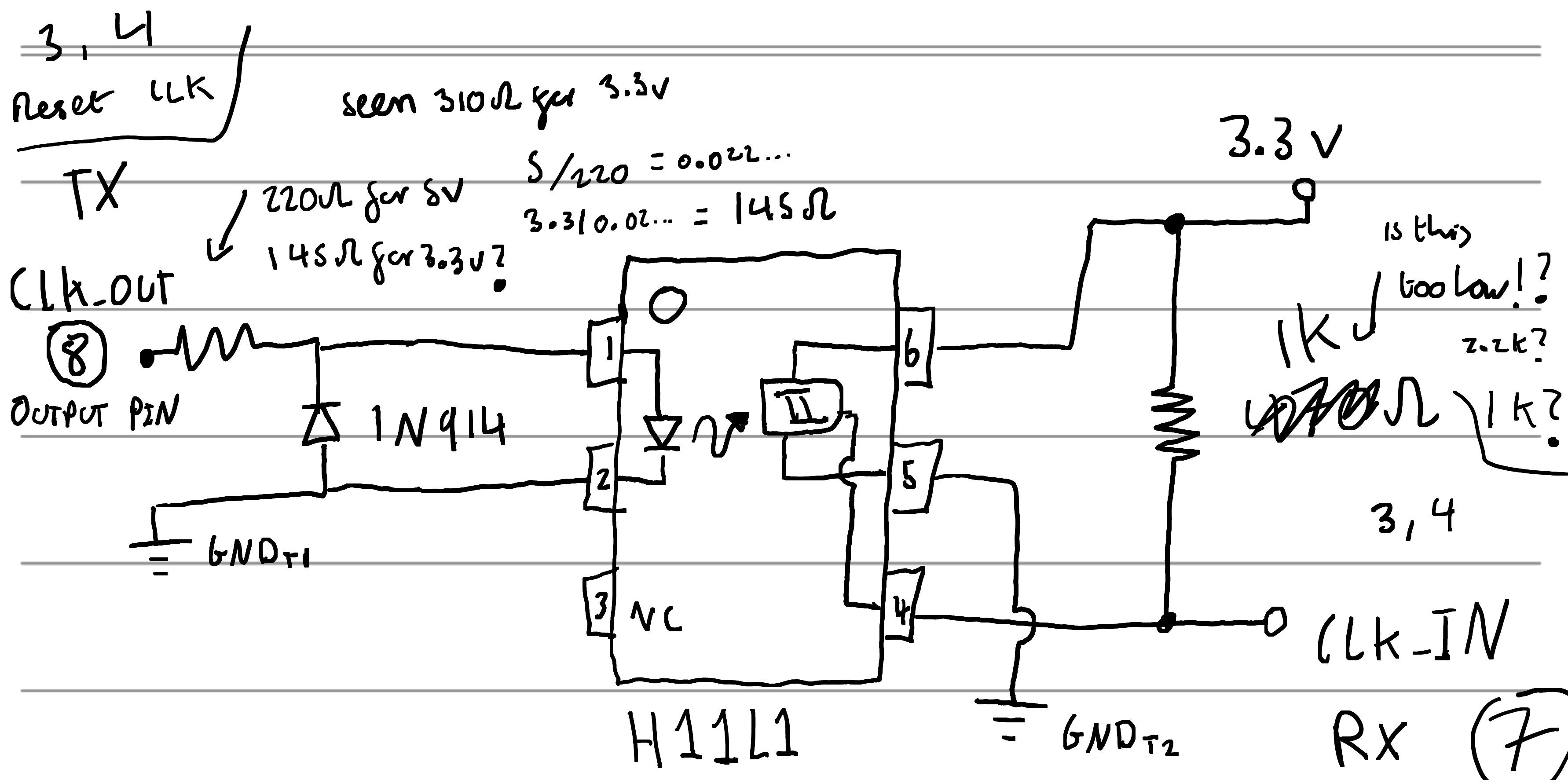
ISR Digital Interrupt

CLK_IN Rising?

debounce?

tick T_2

comp $T_1 \text{ < } T_2$

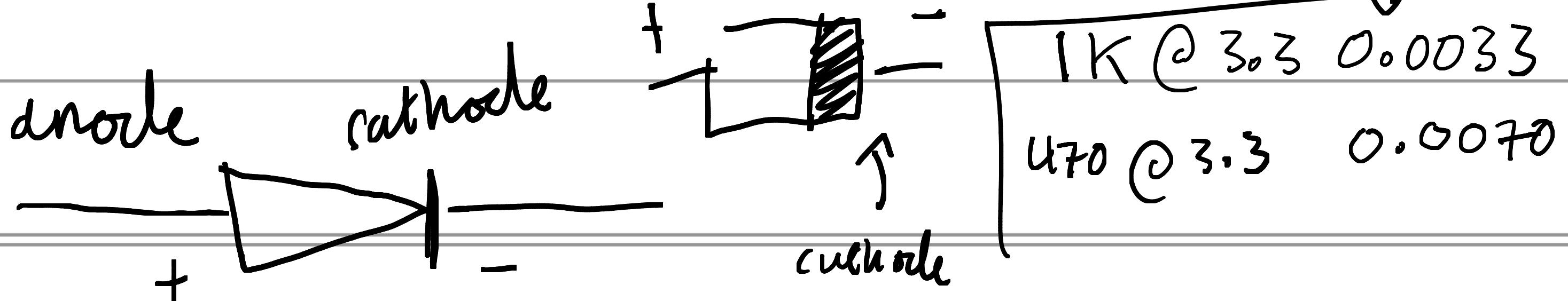


Rx/TX 5 21 20 } clear i think
Rx/TX 6 25 24 }

Perhaps use Rx 2 & Tx 2 for CLK as maybe

higher switching rate as used for serial uart.

try 1k first



order to try:

	220Ω	1kΩ	J danger teensy
danger opto	↓	220Ω	470Ω
	180Ω	470Ω	

5/7/22

Tested single Teensy 4.0 opt-coupling to itself ok.
pulse rate @ 120 kHz, pulse duration 3,000 ns / 3 μs.

Fairly stable seems to miss between 0 → 3 ticks, but
remains stable, no determined the reason for missed clocks
perhaps some kind of initialisation issue.