

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

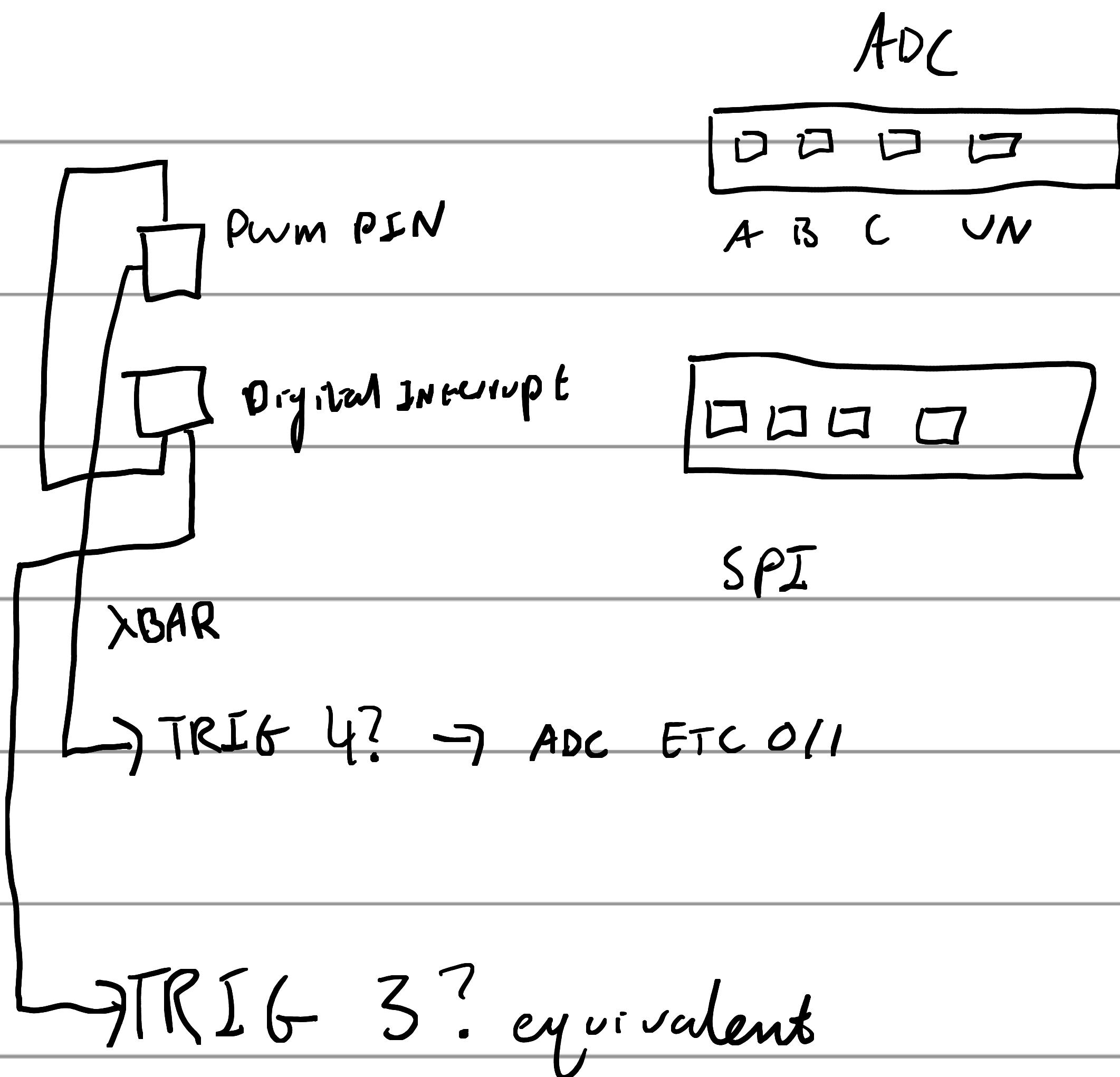
ETC-0 callback \rightarrow collect A & B voltages & record time

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

Output

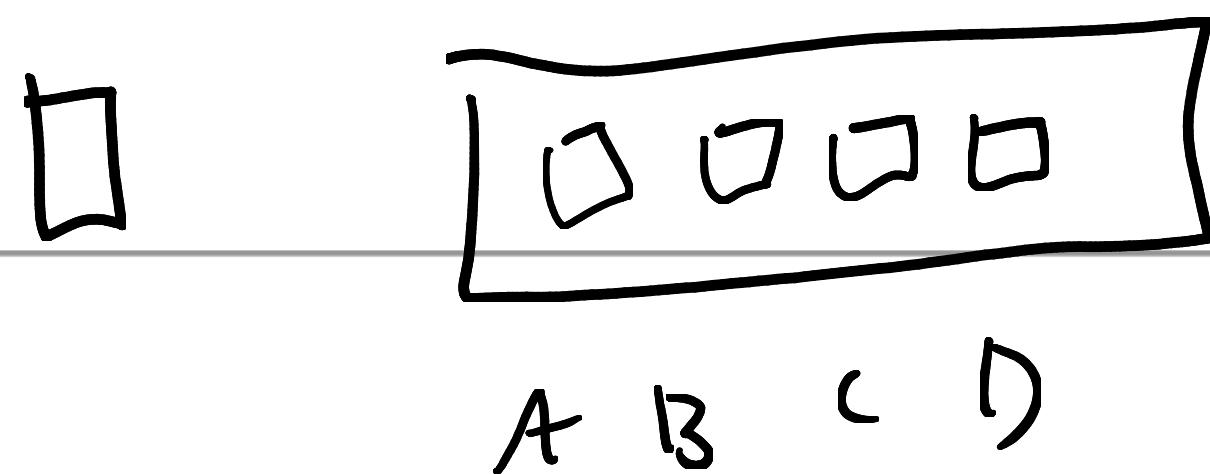
Voltages / time

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



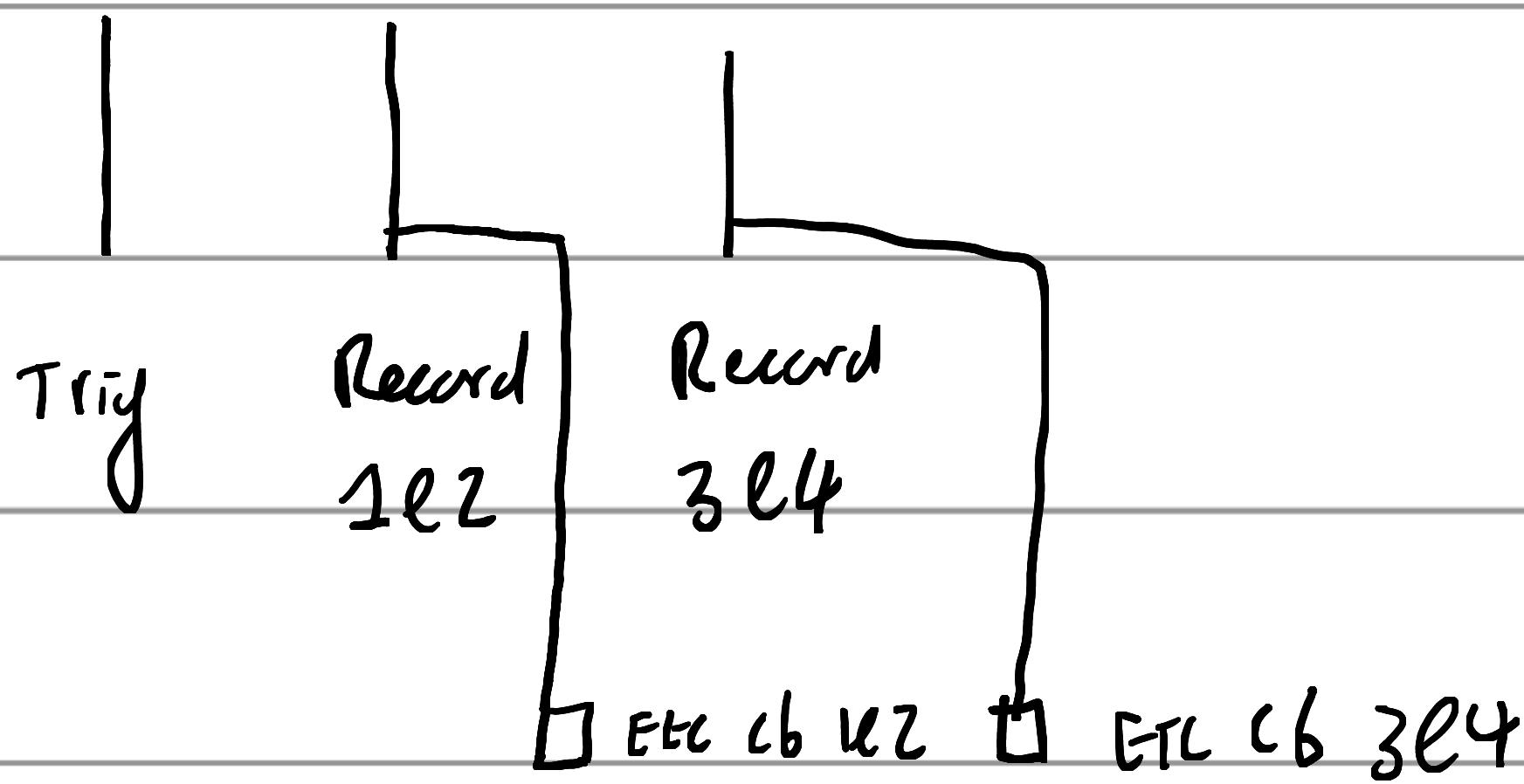
→ Record time before

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

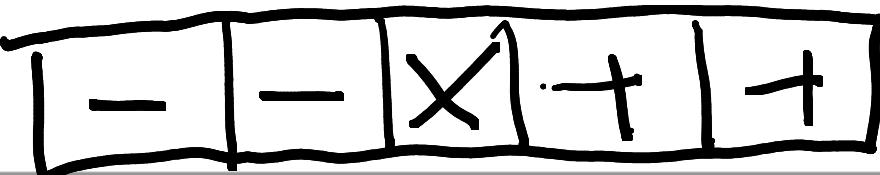
Refactored repository README.md in each directory.

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

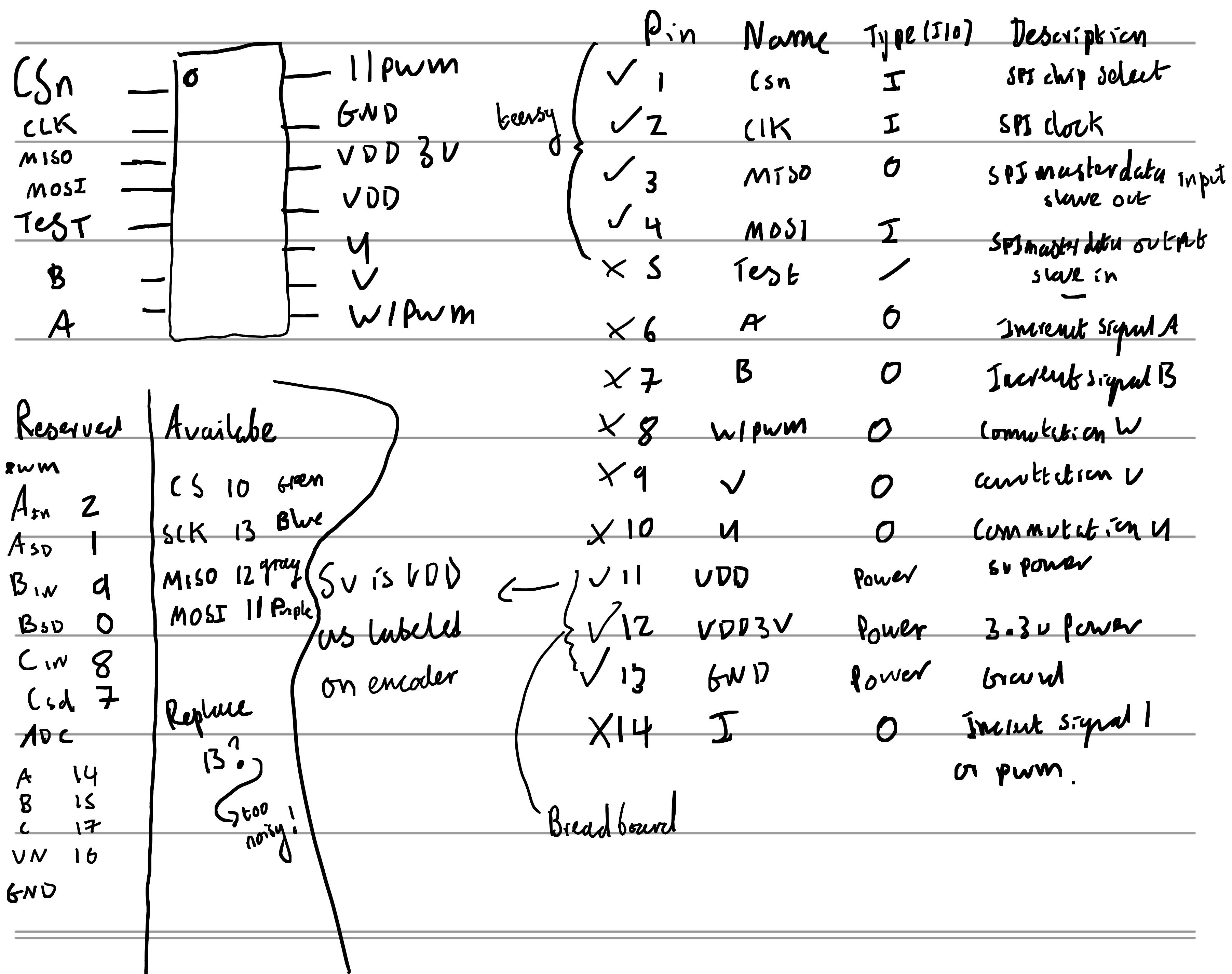
→ Eventually would be nice to have an angle measurement
ADC
at about 30khz if voltages permit, or about 2 samples unique angle

longer term when encoder is finished build a histogram

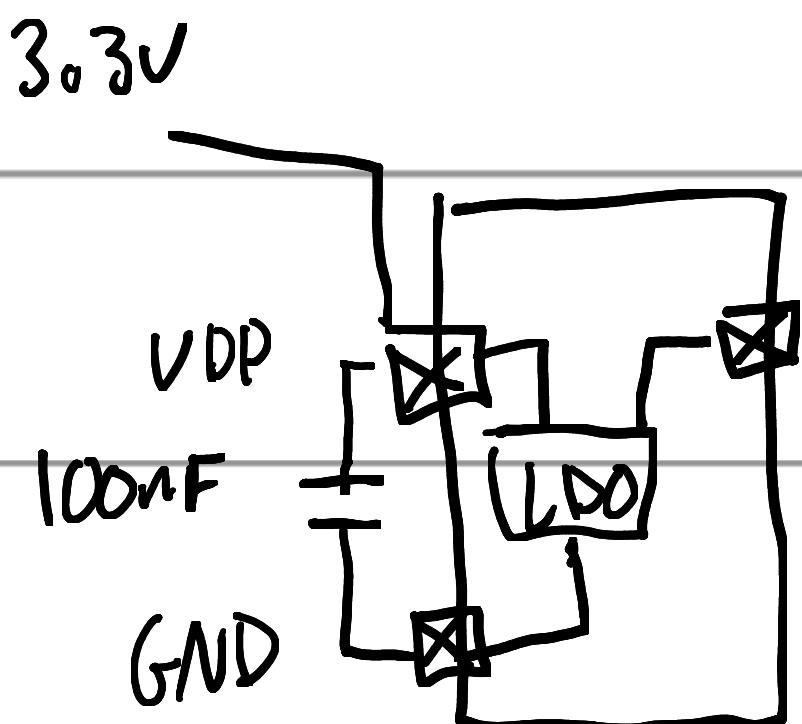
of angle vs zero crossing points, validate sequence, approximate error. etc...

8/6/22

14 bit encoder AS S147 P



3.3 V operation

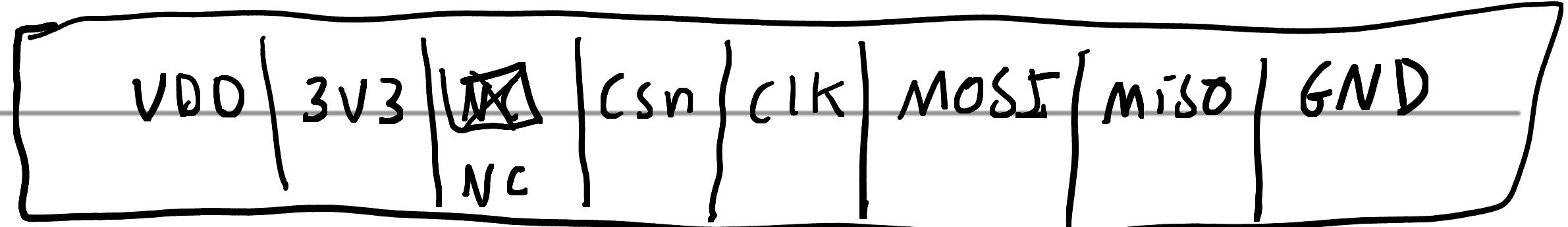


(SV)

100nF between VDD & GND

VDD & VDD3V bridged.

Breakout
pin layout

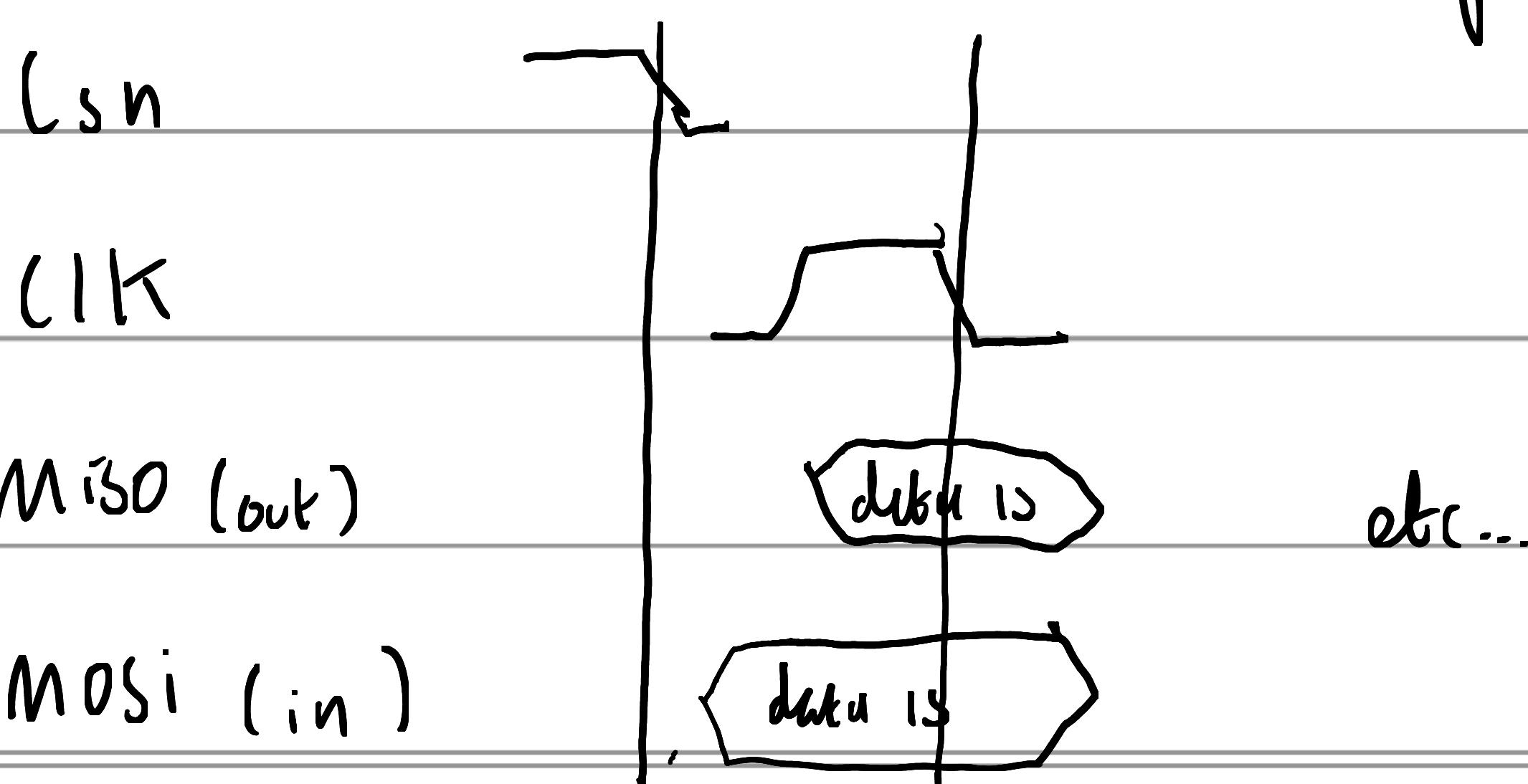


Data transfer:

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 \rightarrow low

Each bit of command

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

After all at end of frame CSn low \rightarrow 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) $0096, 4046, 4096, 4096, 16384$
 \downarrow
(with time) $20 + 6 \text{ bytes}$
32 bits $4,294,967,296$ 10 chars 2086 bits

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

Teensy 4.0 wait. Will it buffer? Try & redirect tty to ram file, which

9 / 6 / 22

Finished assembly setup today, soldered together the cable which connects the drill driven test blade motor I 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

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

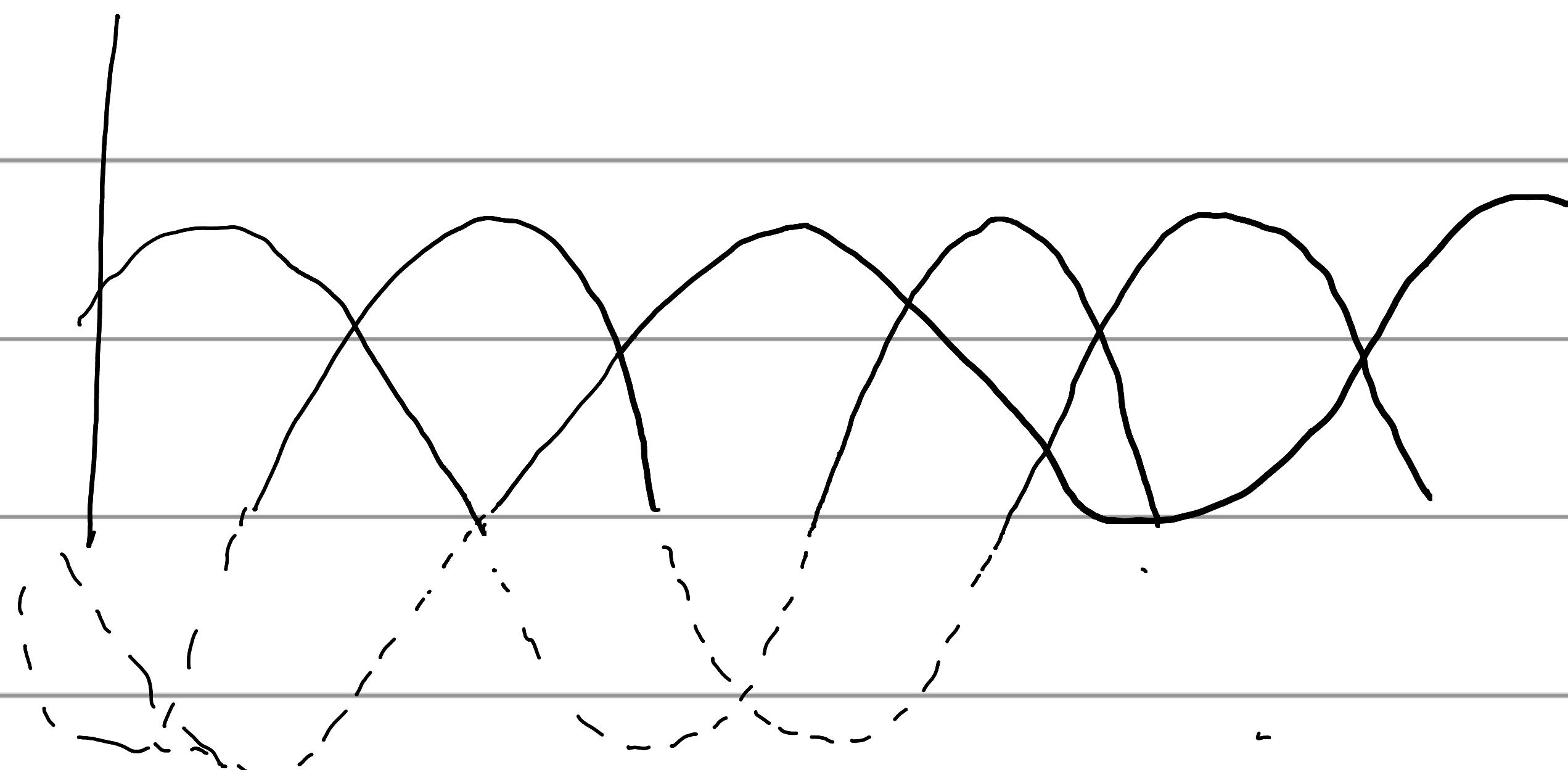
ADC + Encoder combined

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?

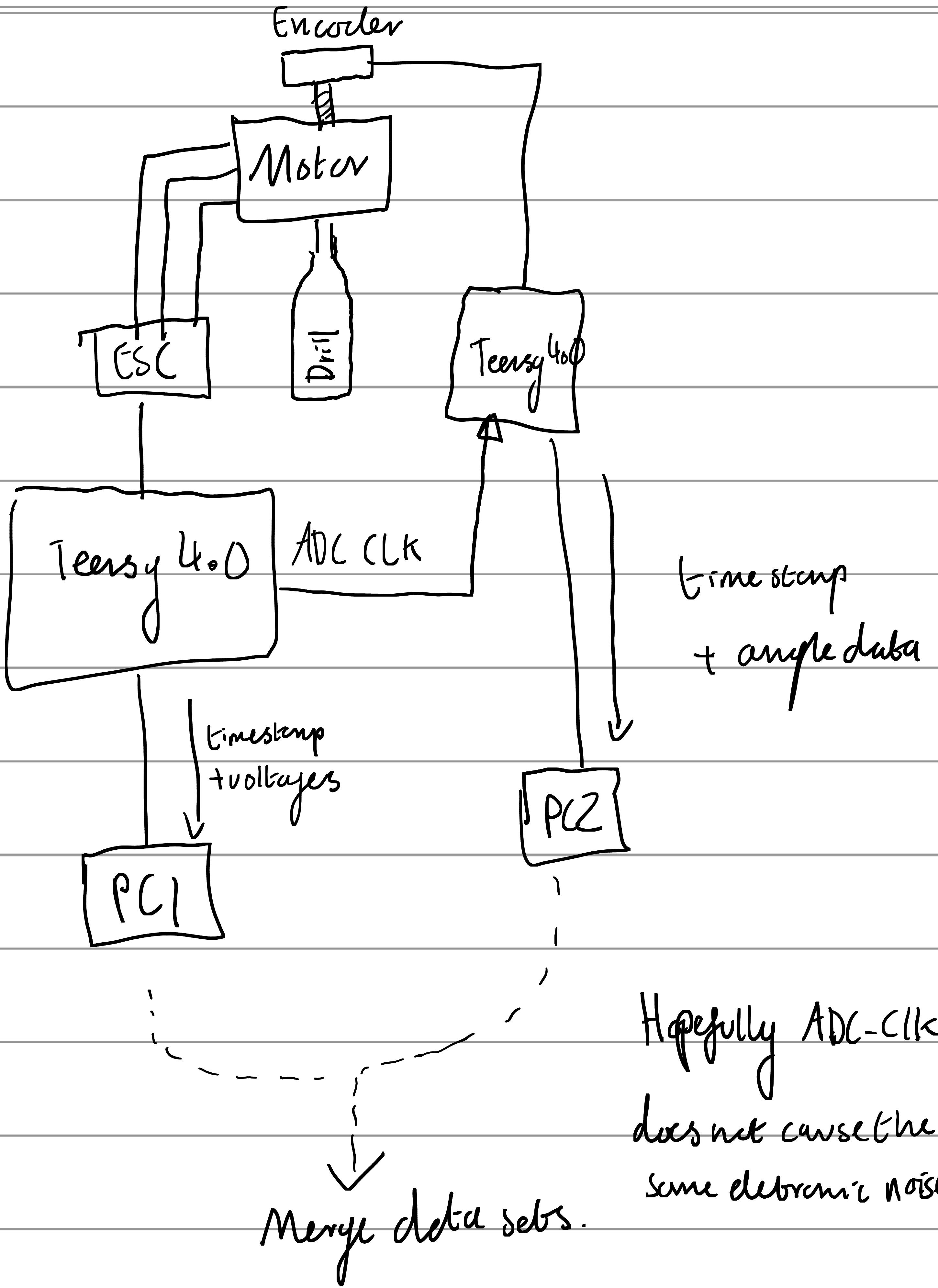
13/6/22

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