0.0.1 Quadrature Encoder

In order to understand the design of the Quadrature Decoder a small explanation, of how a Quadrature Encoder works, is needed.
A Quadrature Encoder uses two channels to sense the position and direction of, typically, a rotating disk/shaft or a linear strip. The disk or strip has two paths on it, positioned 90 degrees out of phase with each other, see figure 1 for a rotary encoder and figure 2 for a strip encoder. Stationary sensors are typically placed on top of the encoder[1], so when a track moves in relation to a sensor, it outputs a logic high or low output signal depending on what part of the track is visible to the sensor. An encoder has two output signals, one for each channel, typically called A and B, see figure 3 for a representation of those for both encoder types. These two signals, A and B, are what allows the decoder to determine the position and direction of the encoder. [?]

fjern those for

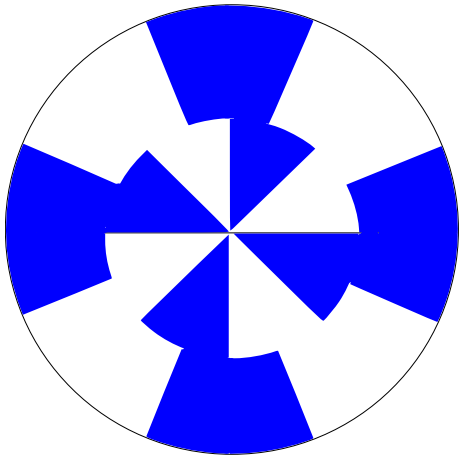

Figure 1: *Tracks of a rotary encoder.*

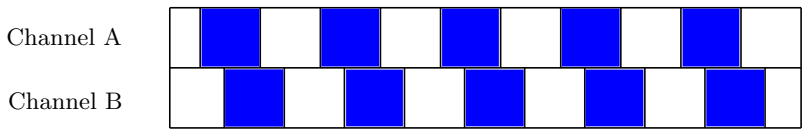

Figure 2: *Tracks of a Linear encoder*

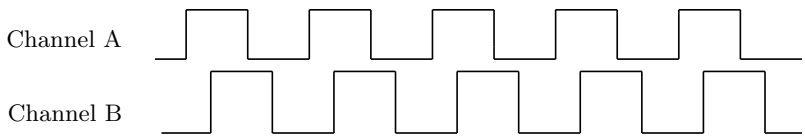

Figure 3: *The logic output of an encoder*

0.0.2 Decoder principle

The founding principle of a Quadrature decoder relies on decoding the relationship between the two outputs and deduce a position change. The idea is to observe both encoder outputs. By counting the transitions from high to low and low to high on just one of the outputs, it can be determined how far the encoder has rotated. However, by adding the second

[1] The Stationary sensors are part of the whole component called an encoder

output, the direction of the encoder can be computed. The encoder used in the project is a rotary encoder with a resolution of 360[**?**], which means that the Quadrature Decoder will count 360 "ticks" per shaft turn. The resolution can not be mapped directly to the system itself as its motors are geared. The motors do three full rotations of their shafts before the system itself has completed a full rotation. Hence the resolution for the system is 1080 ticks. This gives a $\frac{1}{3}$ of a degrees position precision, allowing a very precise measurement.

0.0.3 State machine and reset

To use the transitions between encoder output A and output B, a truth table of every possible combination of the two signals must be computed. It is necessary to include a third signal, called reset, to allow for resetting of the position counter when a new home position is desired. The truth table computed from all three signals is shown in figure 4.

A_prev	A_new	B_prev	B_new	Reset	Direction	Position
0	1	0	0	1	Forward	+ 1
1	1	0	1	1	Forward	+ 1
1	0	1	1	1	Forward	+ 1
0	0	1	0	1	Forward	+ 1
0	0	0	1	1	Backward	- 1
1	0	0	0	1	Backward	- 1
0	1	1	1	1	Backward	- 1
1	1	1	0	1	Backward	- 1
x	x	x	x	0	No change	0

Figure 4: *This figure shows the truth table for the Quadrature decoder*

The truth table shows eight possible scenarios that can occur from tracking the transitions of the outputs. Four of these results in a forward direction and the counter will increment, four scenarios results in a backwards direction and the counter will decrements. The last scenario occurs if the reset flag is set low[2], and the counter is set to zero regardless of what it was on before. From these observations a state machine containing 4 states has been computed. The states are: AB_low with the gray code 001[3], AB_high with gray code 111, A_high with gray code 101 and B_high with gray code 011. The state machine diagram is shown below in figure 5. There is no reset state as it would not meet the criterias of a state. Thereforth the reset signal will be treated as synchronous reset signal and every state will check for it.

The decoder works by first initializing and go to a state depending on the level of signal A and B. The decoder will then use the debounced signals to determine the next state. If the starting state is AB_low, there is three correct outcomes: the reset signal goes low and the counter will go to zero, the signal A goes high and the state changes to A_high or the last possible outcome, that does not give an error, is B_high. By looking at the truth table from the above figure 4, it can be observed that if the state goes from AB_low to A_high, the direction will be forward and the counter will increment where the counter would decrements and direction set to backwards if the next state is B_high.

The counter keeps track of the position of the encoder. The counter counts position zero

[2]Reset is implemented as active low, as can be observed in the truth table 4.
[3]The first digit is signal A, second is signal B and third is the reset signal

Nu er greycode kun 2 bits ie. "11"

Syntes dette er fyld, behøver ikke være der, blot beskriv at reset bliver bru som synkron si nal

Følgende afsnit syntes jeg ikke behøver være der, truthtable forklare godt n hvad der sker.

Lyder lidt mærkeligt, hva med: Since the counter value i relative to the initial position, it is conds ider

as the angle the system was in as the encoder was initialized. It is therefore considered useful to add a reset function, so a new home position can be added as desired. The state machine approach has been implemented and the process described in section **??** under subsubsection **??**

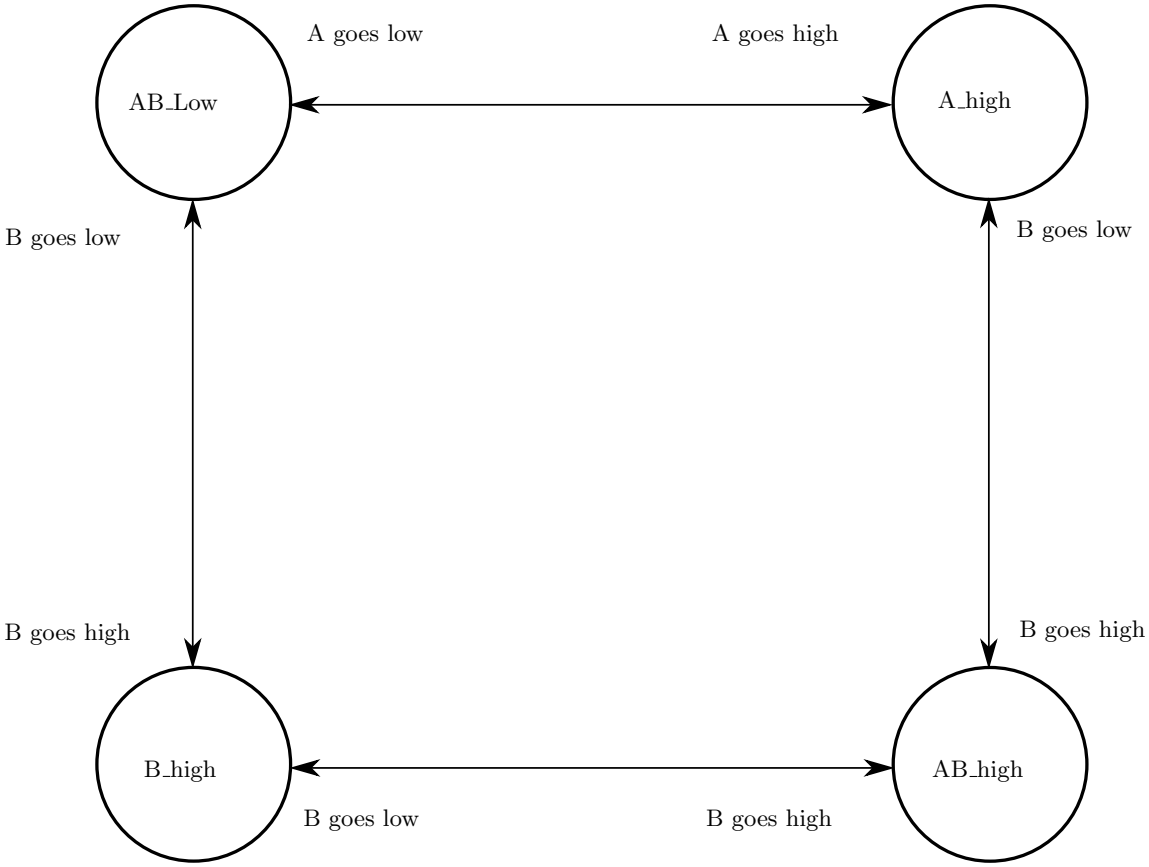

Figure 5: *Shows the different states the decoder can be in*

As mentioned above a reset is needed for the application as this allows a new home position to be set as desired. The reset can be implemented as either a synchronous or an asynchronous reset. The difference between the two is that the synchronous reset depends on a clock signal, either rising or falling egde, and will not be effective before next actove edge whereas the asynchronous reset is independent of the clock and triggers immediately when enabled.
The advantages of a synchronous reset is that it gives a synchronous circuit and provides a natural filter for most giltches.[4] The disadvantages are that it requires a clock signal to reset and the response time depends on the clock frequency, which produces a further issue with the length of the reset signal. This paticular issue however is not relevant in this system as the decoder component will run many times faster than the component that provides the reset signal.
The asynchronous reset has the advantages of the reset being of higest priority and it has an almost immediate response. The disadvantages are that it is very sensitive towards any noise as just a milisond of a "fake" reset signal can set it off and it can cause issues if any synchronous resets are used in the system.

[4] If a glitch/noise happens on an active clock edge the fake reset will go through

Jeg syntes lidt følgende afsnit mere er en slag teori, angående asynkro- n/synkron re- set, fremfor hva vi har valgt og hvorfor? Evt. lige spørg om der er andre de syntes det?

Vil mene det er ligegyldigt om det er synkront eller asynkront mht støjfølsomhed,

Due to the fact that the state machine itself is using an active clock edge and that the decoder component will be the fastest, the synchronous reset has been chosen.

0.0.4 Velocity measure

The decoder is not only responsible for computing the position of the system but also the velocity. The simplest way to estimate the velocity is by dividing the change in position with the change in time. The most classic options are either to keep the Δt fixed, measure the position between the interval and then take the difference between x(n) and x(n-1) as shown in equation 0.1. The other is to keep the position fixed and compute how long it takes to reach the fixed Δx like in equation 0.2.

$$v(n) = \frac{x(n) - x(n-1)}{T} \tag{0.1}$$

$$v(n) = \frac{X}{t(n) - t(n-1)} \tag{0.2}$$

The second approach 0.2 is considered the most reliable at slow speeds, which arguably the system does fall under. However, this approach depends on the output to be a "fixed interval pulse train"[**?**]. The system does not fit this criteria, because it generates different frequency waves depending on the supplied PWM value.

The first approach using equation 0.1 is not hindered by the difference in frequency it has been chosen.

As the chosen approach requires a fixed time interval one such must be decided. The issue to take into account here is to make sure the interval is neither too short or too long. A too short interval might yield scenario where the velocity seems to be zero even if the system does move. However, on the opposite side, a too long interval could yield a scenario that gives a wrong velocity, if the system changes velocity a lot.
This can be illustrated using a system that during a 1 second sample period changes velocity more then once, the result will always be a average over the period and if the system changes velocity a lot can be very misleading.
In this case it is important to capture the system speeding up and slowing down, so a low interval time of $10ms$ has been chosen.

lidt talesprog?

behøver det væ
der? syntes det
er forklaret fint
ovenfor.