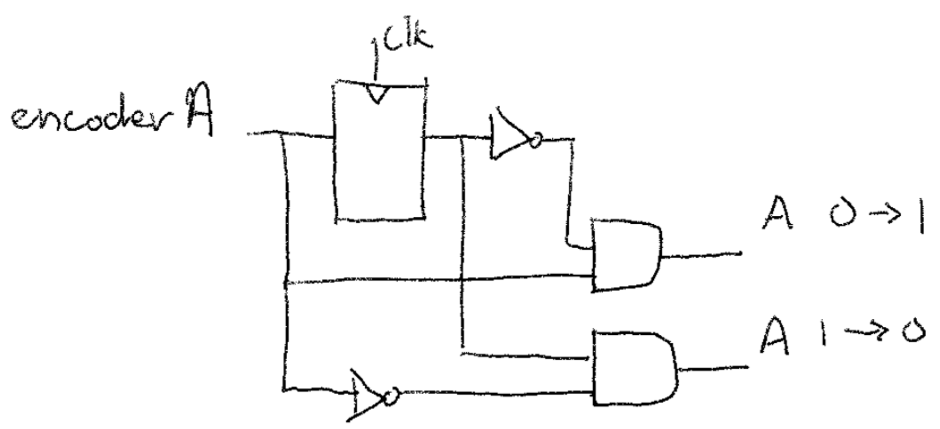
ES 4 Using rotary encoders as input

Tracking encoder pulses

Look at the datasheet for the Bourns PEC12R-4217F-N0024 encoder (posted on the course website). Connect it to a multimeter or the Digital Discovery and play around with it until you understand how it works. In particular, how can you tell which direction it's being turned?

To count, we essentially watch for an edge on one signal, check whether the other signal is high or low, and increment or decrement the count appropriately. This diagram may help you think about the technique:



if	A 0→1	and	B=1	count=?
	"	"	B=0	count=
if	A 1→0	and	B=1	count=
	"	"	B=0	count=

However, if we want to watch for both rising and falling edges, we can simplify the design by using an XOR gate instead of AND. This page describes the design: https://www.fpga4fun.com/QuadratureDecoder.html

Wire the encoder to the UPduino. The encoder essentially contains two switches that open and close as you turn the knob, so you can connect the encoder between the UPduino pin and ground and use the internal pullups just like in previous labs.

The encoder is a bit annoying to put into the breadboard. We suggest you bend the mounting tabs slightly inward, and place it into the breadboard with the mounting tabs in the center groove of the breadboard.

Dealing with metastability

You probably noticed that your knob doesn't work well at all: sometimes it skips, sometimes it even counts backward. This terrible behavior is due to metastability[1]. If the input switches on a clock edge, it might not be a well-defined 1 or 0. This means we've broken the binary abstraction, and all bets about the behavior of our circuit are off.

[1]Unless there are bugs in your design, in which case it's due to metastability *and* bugs.

This will *always* be a hazard when you're dealing with inputs from the outside world that are not synchronized to the internal clock. To prevent metastability (or more correctly, to make it exponentially less likely), we can use a series of flip-flops as a synchronizer. Two flip-flops is generally enough.

```vhdl
process(clk)
begin
  if rising_edge(clk) then
    sync1 <= encoder_input; -- encoder_input is a signal from the outside world
    sync2 <= sync1;
    -- Now you can use sync2 to compute logic; it should be a stable 0 or 1
  end if;
end process;
```