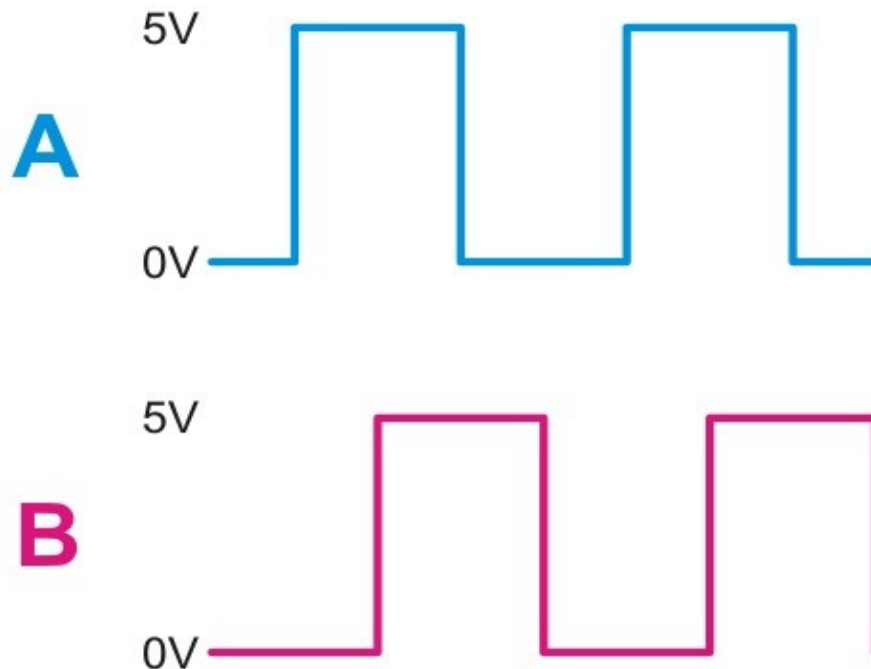


## How to use a quadrature encoder



A quadrature encoder, also known as an incremental rotary encoder measures the speed and direction of a rotating shaft. Quadrature encoders can use different types of sensors, optical and hall effect are both commonly used. The photo shows inside of a Rover 5 gearbox. There are two IR sensors on the PCB that look at the black and white pattern on one of the gears. No matter what type of sensors are used the output is typically two square waveforms  $90^\circ$  out of phase as shown below.



If you only wish to monitor the speed of rotation then you can use either output and simply measure the frequency. The reason for having two outputs is that you can also determine the direction of shaft rotation by looking at the pattern of binary numbers generated by the two outputs.

Depending on the direction of rotation you will get either:

- 00 = 0
- 01 = 1
- 11 = 3

10 = 2

or

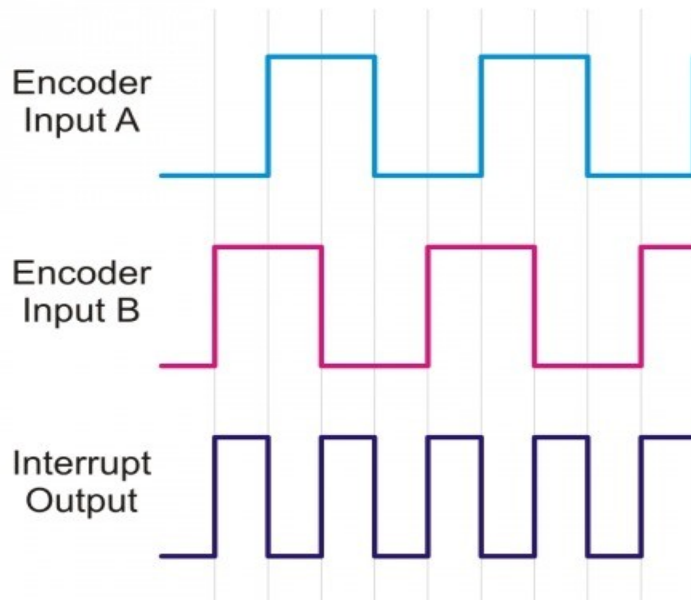
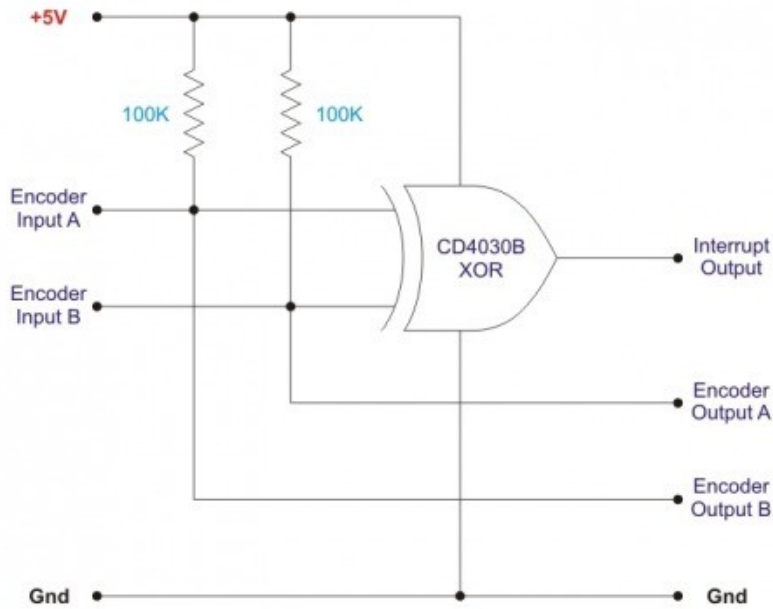
00 = 0

10 = 2

11 = 3

01 = 1

By feeding both outputs into an XOR gate (exclusive OR) you will get a square wave with twice the frequency regardless of direction. This can be useful as it allows one interrupt pin to monitor both encoder inputs.



I was looking at how to write efficient code to convert these binary inputs into a simple "forward or backward" output. I ended up with a 2 dimensional array (matrix) that made the code quick and easy.

The binary values above convert to 0,1,3,2 or 0,2,3,1 depending on the direction. This pattern repeats continuously. By using the current value from the encoder to index one dimension of the array and the previous value to index the other dimension you can quickly get a -1, 0, or +1 output. My array looks like this.

		Current Value			
		0	1	2	3
Old Value	0	0	-1	+1	X
	1	+1	0	X	-1
	2	-1	X	0	+1
	3	X	+1	-1	0

As you can see, if the value has not changed then the output is 0.

The sequence of 0, 1, 3, 2 gives an output of -1.

The sequence of 0, 2, 3, 1 gives an output of +1.

X represents a disallowed state and would most likely occur if the encoder outputs are changing too quickly for your code to keep up. Normally this should not happen. In my code I put a 2 here. When I get an output of 2 I know that I got an error, perhaps due to electrical noise or my code being too slow. If you replace X with 0 then the disallowed state will be ignored.

In my Arduino code I make this a 1 dimensional array. that looks like this:

```
int QEM [16] = {0,-1,1,2,1,0,2,-1,-1,2,0,1,2,1,-1,0}; // Quadrature Encoder Matrix
```

To read the array my index is: Old \* 4 + New

So my code reads like this:

```
Old = New;
```

```
New = digitalRead (inputA) * 2 + digitalRead (inputB); // Convert binary input to decimal value
```

```
Out = QEM [Old * 4 + New];
```

Good luck and enjoy.

Brought to you by <http://letsmakerobots.com>