Rotary Encoders

Introduction

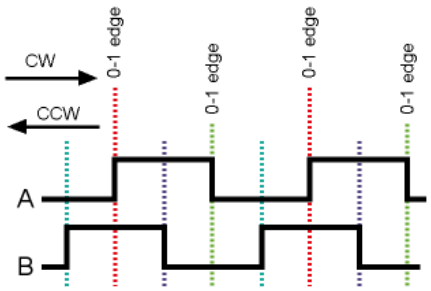
For this guide, we explore on rotary encoders. You may have been interfacing with this device daily as you will find them in your 3D printer knobs and stereo volume control.

A rotary or “shaft” encoder is an angular measuring device. It is used to precisely measure rotation of motors or to create wheel controllers (knobs) that can turn infinitely (with no end stop like a potentiometer has). Some of them are also equipped with a pushbutton when you press on the axis (like the ones used for navigation or volume control on your stereo). They come in all kinds of resolutions, from maybe 16 to at least 1024 “steps” per revolution.

Interfacing with Arduino

An encoder normally has 3 pins. We will refer to them as A and B. These pins contacts with a common pin C. We can determine the speed of rotation of the shaft of the encoder by just counting the “pulses” or the times that either pin comes in contact with pin C.

However, if we want to determine the direction of rotation, we need to consider both the pins.



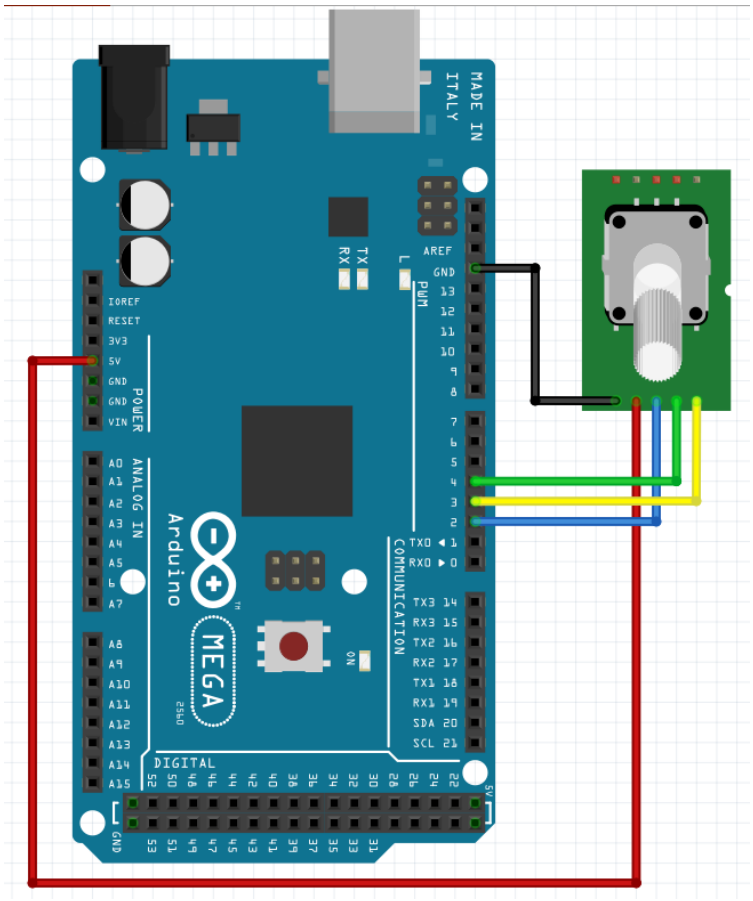
Rotating the switch clockwise will cause pin A to change states first.
Rotating the switch counterclockwise will cause pin B to change state first.

Interfacing with an encoder requires more circuit component that just the encoder itself but luckily, we have a module with all stuff needed built in.

So notice that in our encoder, we have 5 pins.

GND : This pin is to be connected to ground
+ : This pin is to be connected to the supply voltage. Either 3.3V or 5V
SW : Note that some encoders has a pushbutton switch, this pin is connected to the pushbutton part of the encoder. (get a multimeter, set to continuity mode, connect SW and GND) Notice that when you press the button on the encoder, SW is being connected to ground. Otherwise the output of the pin will be 5V (or your supply voltage) due to the presence of the “pull-up” resistor at the back of the encoder module
DT / DATA : Think of this as your encoder “PIN B”
CLK / CLOCK : Think of this as your encoder “PIN A”

As you can see, DT and CLK are just like “switches” and connects to the ground momentarily when you rotate the encoder switch (get a multimeter, set to continuity mode, connect DT or CLK to GND) – otherwise they will output a voltage equivalent to 5V or your supply voltage, also because of the pull-up resistors at the back of the module.



DO NOT FORGET to power off your arduino when wiring and make sure to always double check before powering up
So we connect:

GND	to	Ground
(+)	to	5V
SW	to	PIN 2
DT	to	PIN 4
CLK	to	PIN 3

Upload the code to the Arduino (Rotary Encoders sketch)

```
#define encoderPinA  3  //CLK
#define encoderPinB  4  //DT
#define pushbutton 2    //SW

unsigned int encoderPos = 0;

void setup() {

  pinMode(encoderPinA, INPUT);
  pinMode(encoderPinB, INPUT);
  pinMode(pushbutton, INPUT);
  attachInterrupt(digitalPinToInterrupt(encoderPinA), doEncoder, FALLING);
  attachInterrupt(digitalPinToInterrupt(pushbutton), doButton, FALLING);
  Serial.begin (9600);
}

void loop(){
  Serial.print("Position:");
  Serial.println (encoderPos, DEC);
  delay(500); //a delay so that we wont get overwhelmed
}

void doEncoder() {
  if (digitalRead(encoderPinB)==HIGH) {
    encoderPos++; //counter-clockwise
  } else {
    encoderPos--; //clockwise
  }
}

void doButton() {
  Serial.println("button pressed");
  }
```

So the first 3 lines, only defines the pin configuration. It is important to note that we used pins 3 and 2 here as they are “interrupt-capable pins”. Having an interrupt simply means that the microcontroller will “pause” the execution of the execution of the program and to run a predefined routine. We’ll see more of that later

Then “encoderPos” is just a variable that we increment or decrement depending on the rotation of the encoder. Note that if the variable hits “zero” and we try to decrement it, as it is unsigned, it will just simply roll over to the highest possible value – in this case 65535

Pins 9 to 11 simply configures our pins as input pins.

The “attachinterrupt” here on lines 12-13 attaches our pins to an interrupt. Meaning, if encoderPinA input is on a rising edge (low to high), it will execute the “doEncoder” routine. There are only a handful of pins in your Arduino that can handle an interrupt so make sure to verify first before attaching. In our particular board, pins 2, 3, 18, 19, 20, 21 are the only ones capable.

Same with pushbutton. If the microcontroller sees the voltage go from high to low or a falling edge, it will call the “dobutton” routine.

The loop routine here only displays the value of our counter every 500 milliseconds

The interrupt routine “doEncoder” is called when there is a “rising edge” on encoder “pin A”. when detected, it will check the state of “pin B”. If pin B is high then it means that it got “triggered” first and our encoder is rotating counter-clockwise. If it is low, it is rotating clockwise. The routine then increments or decrements our counter accordingly.

Finally “doButton” just displays when the button is pressed.

Useful links
https://dronebotworkshop.com/rotary-encoders-arduino/