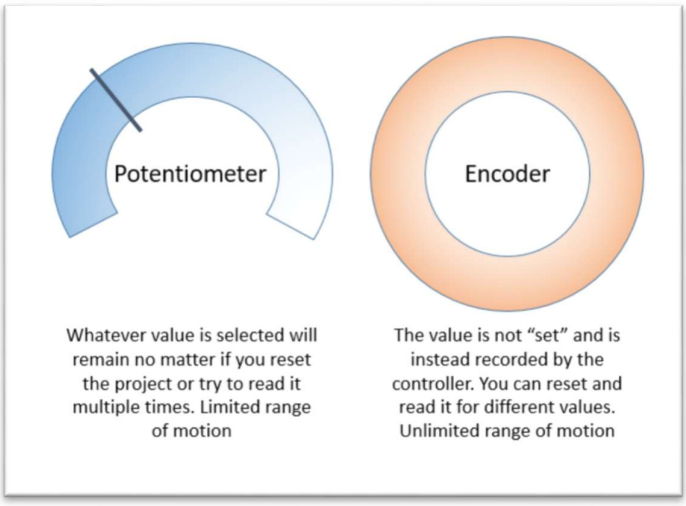
Rotary Encoder

SR1230

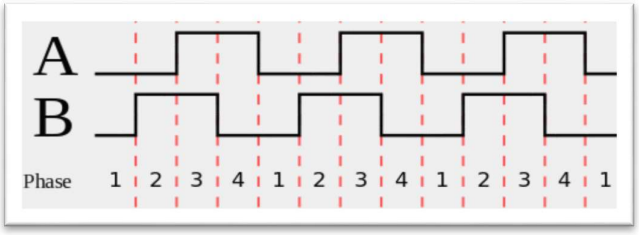
How Encoders work

Rotary encoders detect a change in rotation, and as such cannot be used for detecting a "set" position. That is, each time you restart your arduino project, the encoder is set at 0, no matter how many times you've turned it previously.

This is different to a potentiometer, which will retain the same value when you reset your project. The benefit to this is that the rotary encoder can have an infinite number of turns and works well for adaptive controls such as menu items or code where you want to control many different values from the same knob.



Most encoders work on the same premise. There's 3 main connections of concern, which would be labelled **A**, **B**, and Common or **C**. The task of the microcontroller is to track the changes between A and B to record which way the encoder is turning and how many "clicks."



Starting from phase 1, if we turn right one click, we see that B raises high, another click then rises A to high, then B low, and finally A low, which returns us back to phase 1. In the reverse, A goes high first, rather than B. so we can see what changes first and which way the encoder is turning. In this example, C is connected to ground.

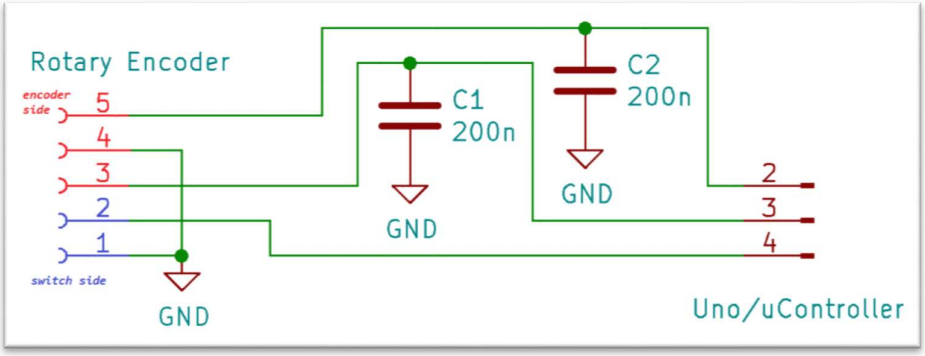
Software implementation

It might sound tricky, but it's easy enough to do in software. We can use "pin change interrupts" to let the UNO count and keep track of the turning. *(for other devices, look at pin-change interrupts or events for your chosen controller).*

(Note, for UNO boards, pin change interrupts can only be on pins 2 and 3)

Rotary Encoder

Connection between the encoder and the Arduino is suggested below.



Note: We recommend 100-200nF capacitors connected between A/B lines to C. You will get unreliable and erratic behaviour without these capacitors.

Each “notch” that you can feel corresponds with one whole phase change, from 1 to 4. To track this, we can look at just the falling edge of one of the pins, and check to see if the other pin matches. Try the below code (swap `enc_a` and `enc_b` if it’s not quite as expected)

```
#define enc_a 2
#define enc_b 3

volatile int encoderValue = 0;

void setup() {
  Serial.begin(9600);
  pinMode(enc_a, INPUT_PULLUP);
  pinMode(enc_b, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(enc_a), encoder, FALLING);
}

void loop() {
  Serial.println(encoderValue);
  delay(100);
}

void encoder() {
  if (digitalRead(enc_a) == digitalRead(enc_b)) {
    encoderValue++;
  }
  else {
    encoderValue--;
  }
}
```

We must use `volatile int` and `digitalPinToInterrupt()` in order to use interrupt functions; if you would like to read more, check out:

https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

SR130