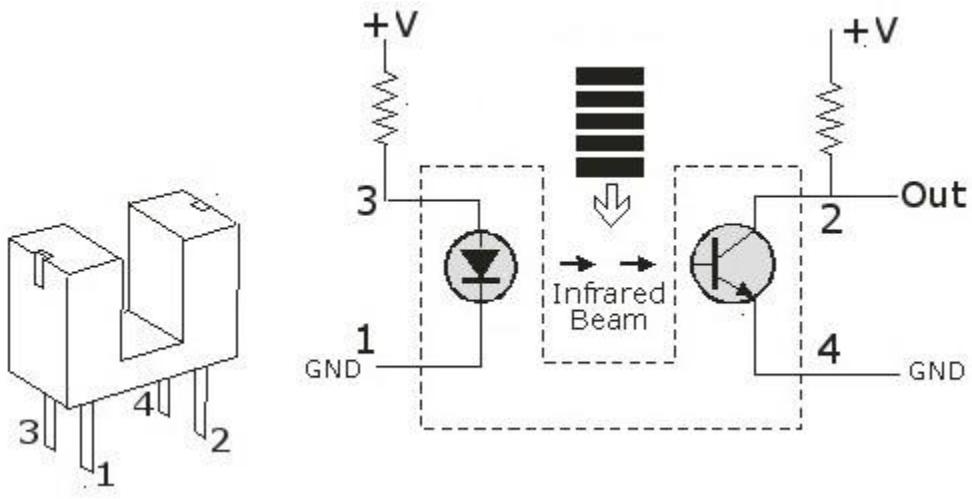
Optical Position Encoder with Arduino

Now a days, optical position encoders/rotary encoders are widely used even in hobby robotics. Common applications of position encoders are:

- DC motor position / velocity control
- Servo-mechanism Position / velocity control
- Computer printer
- Numerically controlled machinery
- RPM sensors in robotics

Photo-Interr-upter

Front end of an ordinary optical position encoder, used for these tasks, is a slotted Photo Interrupter /Opto-Interrupter module with an IR LED & a Photo Transistor/Diode mounted facing each other enclosed in plastic body. When light emitted by the IR LED is blocked because of the alternating slots of the encoder disc (also known as index disc), conduction level of the photo transistor/diode changes. This change can be detected by a discrete hardware or by a microcontroller. In short, a photo-interrupter is composed of an infrared emitter on one side and an infrared detector on the other By emitting a beam of infrared light from one side to the other, the photo-interrupter can detect when an object passes between them, breaking the beam.



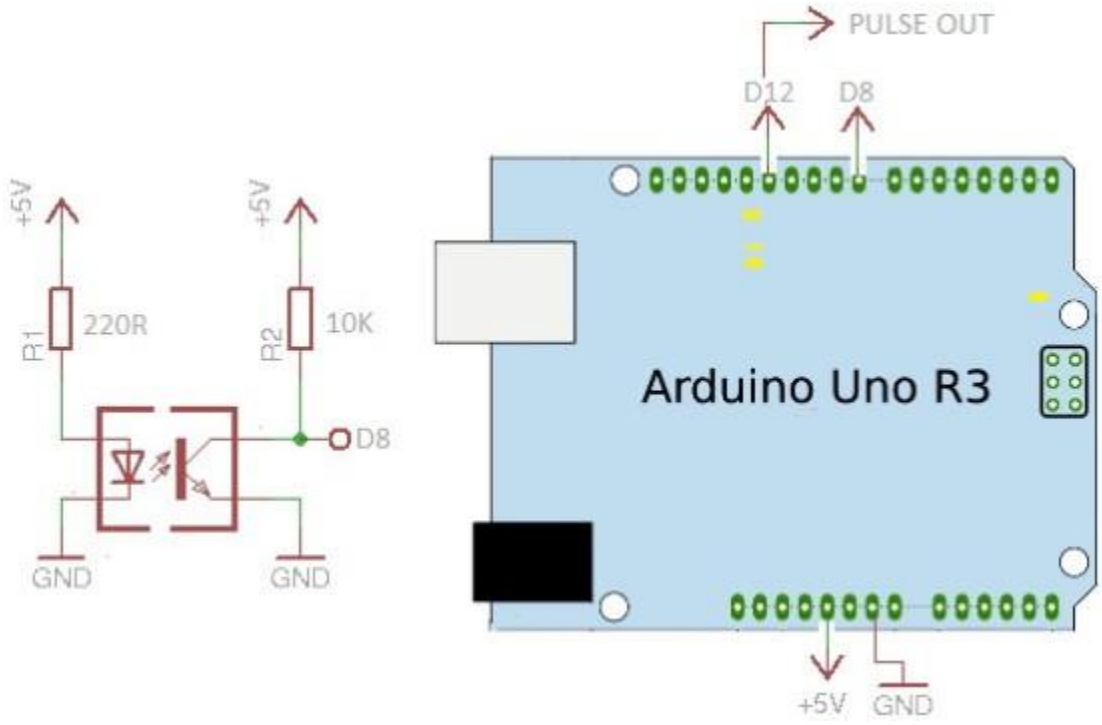
Encoder/Index Disc

As we need to create pulses, an encoder disc/index disc is very necessary here. An easy way to make it is with a thin transparent acrylic, thin transparent adhesive transparency sheet and a laser printer. Take a drawing software and draw the black stripes as shown in the picture. Print it to the transparency sheet and fix it on the round-shaped thin transparent acrylic. As an alternative, you can make the disc from black acrylic and cut out the white spaces.

An important note: Physical width of stripes and spaces is a most critical factor. Look at the datasheet of the photo-interrupter to find its slit width. It is better to set minimum witdh of the stripes (and also of the spaces) two times (x2) the slit width of the photo-interrupter. For example, if the slit width is 1mm, the width of the stripes and spaces should be 2 mm. If the RPM of the disc is 60, then we have 1 disc turn/second. If the disc has 36 stripes then the pulse frequency is 36Hz, which can easily be handled by the photo-interrupter.

The Hardware

To get started, just create a small test circuit as shown here with a photo- interrupter and an Arduino. This allows you to experiment and make sure all things works as per your expectation. The 10K resistor (R2) is a pull-up resistor. Value of the first resistor (R1) depends on the photo-interrupter you use. In this set-up, the onboard LED (at D13) of the Arduino board is usually in off mode, and when the beam broken the LED goes on. Auxillary output available from D12 of the Arduino can be used to monitor the encoded signal on an oscilloscope, for example.



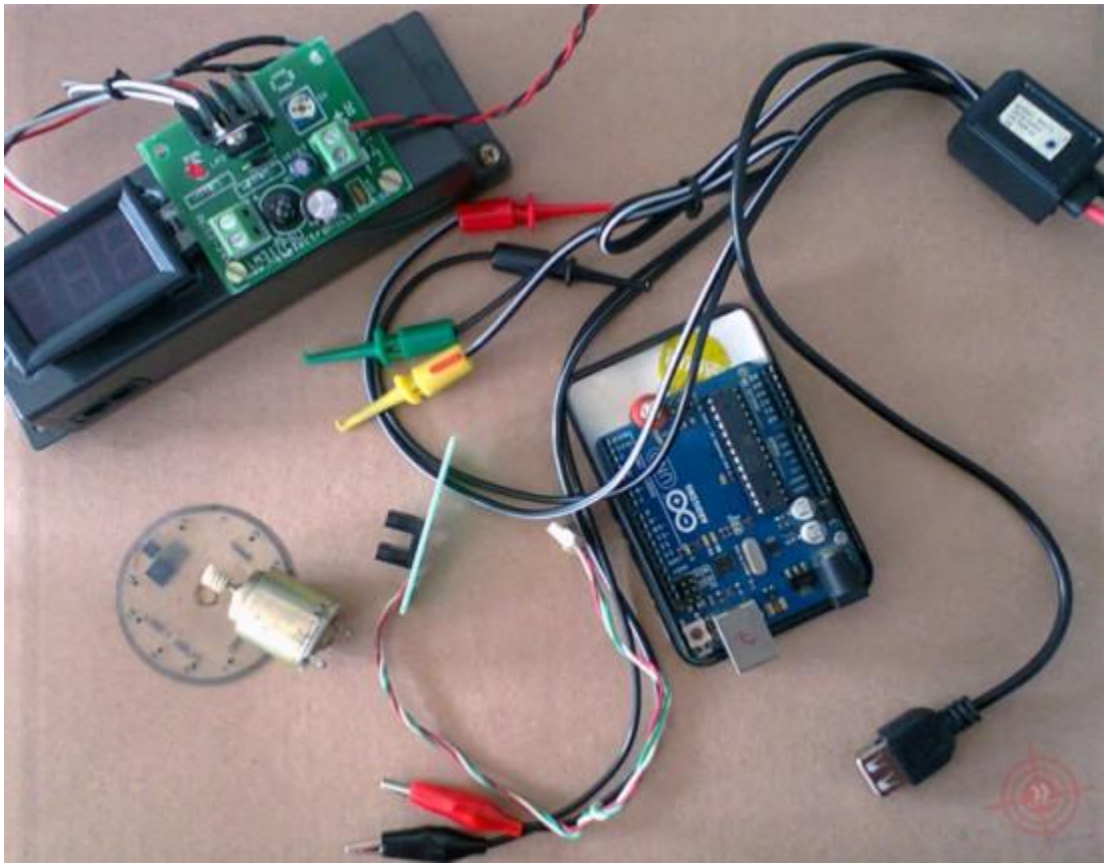
The Arduino Sketch

```
/*
-Arduino Position Encoder
-Using a generic photo-interrupter
-Basic Test Sketch 1 / June 2014
-Tested at TechNode Protolabz
-www.electroschematics.com/
*/
const int encoderIn = 8; // input pin for the interrupter
const int statusLED = 13; // Output pin for Status indicator
const int pulseOutput = 12; // Pulse output pin for external interfacing
int detectState=0; // Variable for reading the encoder status
void setup()
{
   pinMode(encoderIn, INPUT); //Set pin 8 as input
   pinMode(statusLED, OUTPUT); //Set pin 13 as output
   pinMode(pulseOutput, OUTPUT); // Set Pin 12 as output
}
void loop() {
   detectState=digitalRead(encoderIn);
   if (detectState == HIGH) { //If encoder output is high
      digitalWrite(statusLED, HIGH); //Turn on the status LED
      digitalWrite(pulseOutput,HIGH); // Give a logic-High level output
   }
   else {
      digitalWrite(statusLED, LOW); //Turn off the status LED
      digitalWrite(pulseOutput,LOW); // Give a logic-Low level output
   }
}
```

Test Procedure

Bring the photo-interrupter connected with the hardware to your encoder disc and do the test. Connect the D12 output to an oscilloscope and run your encoder disc by hand, or by using a low rpm dc motor. If you don’t have an oscilloscope, watch the on-board LED (D13) to note the pulse output. In this case, try to turn the disc slowly by hand to see the pulse activity directly.

This is the very basic code you need. We have the hardware and the basic software up and running. Now you can upgrade it to whatever you want it to do.



Initial set up of the testing process @ author's lab

Improving the Code

There are two basic ways to read a microcontroller's digital input; Polling, and Interrupt. With polling, the system reads the input all the time inside a loop (as used in this sketch). Main drawback of this polling method is that it is difficult to do other things while polling. The controller does the only job of reading the input and processing the data. But, when using interrupts the system can do any other job without any mishaps. When an input pulse arrives, the

system stops its job, jumps to the interrupt routine and then returns to the job before. So, working with interrupts is what better when it comes to this type of encoding tasks. In a forthcoming article you can see many more about this idea!