VikiLABS

Working with an Incremental Rotary Encoder

www.vikipedialabs.com

May 27, 2017

1 Introduction

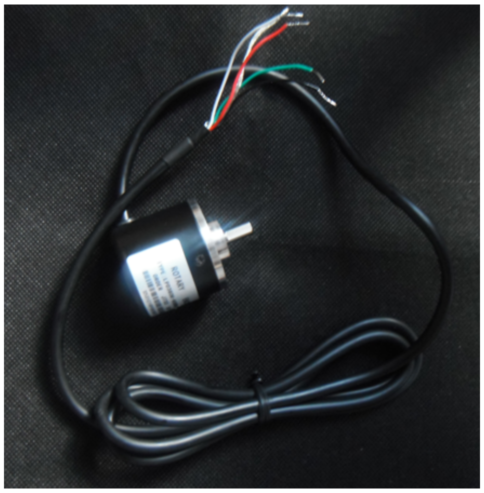

Figure 1: The LPD3806 600BM G5 24C Rotary Encoder

A rotary encoder is used to measure rotational speed, angle and acceleration of an object. It is used to precisely measure rotation of a joint or motor. Rotary encoders are used to provide direct physical feedback of motor position, joint position, and speed of rotation. Unlike potentiometers, it can turn infinitely with no end stop. Rotary encoders come in various kinds of resolutions. The number of pulses or steps generated per complete turn varies from 16 - 1024 pulses/revolution.

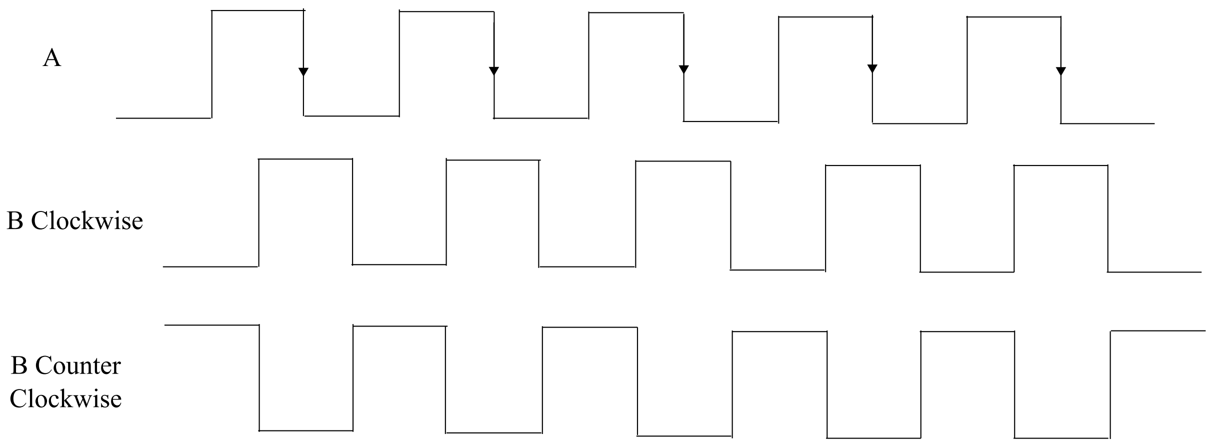

Figure 2: Incremental Rotary Encoder: two square wave outputs (A and B) which are 90° out of phase

A rotary encoder has two square wave outputs (A and B) which are 90° out of phase with each other. Every time the A signal pulse is on the falling edge, the value of the B pulse is read. From Fig 2, when the encoder is turned clockwise, the B pulse is always positive. The B pulse is negative when the encoder is turned counter-clockwise. By connecting both outputs with a microcontroller, it is possible to determine the direction of turn. By counting the number of A pulses, we can determine how far it has turned. The two outputs (A and B) represent the motion of the encoder disc as a quadrature modulated pulse train. By adding a third index signal that pulses once for each revolution, the exact position of the rotor can be known.

2 Specifications:

The LPD3806 600BM G5 24C Rotary Encoder has the following specifications:

- 600 pulses/revolution for a single phase. Therefore, two-phase output leads to 2400 pulses/revolution
- Maximum mechanical speed: 5000 Revolutions/minute
- Response frequency: 0-20KHz

3 Connections:

Colour − Connection

1. Red − 5-24V DC
2. Black − Ground
3. Green − A phase
4. White − B phase

The rotary encoder is open collector encoder, so it is necessary to connect pull-up resistors of 10KΩ to the A and B phase outputs, otherwise it will not function properly. Note that A and B phase outputs must not be directly connected with Vcc, otherwise, will burn the output triode. The connection is shown in Fig. 3.

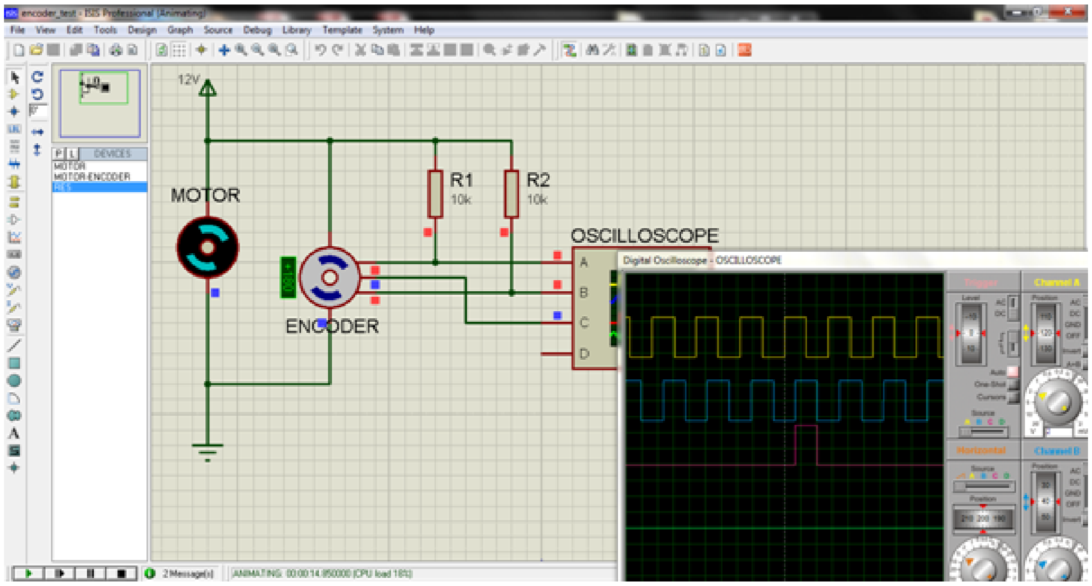

Figure 3: Incremental Rotary Encoder: two square wave outputs (A and B) which are 90° out of phase

4 Measuring RPM Using an Encoder

Revolutions Per Minute (RPM), or how fast something turns, can be sensed in a variety of ways. Two of the most common sensors used to determine RPM are optical encoders and Hall effect sensors. These sensors output one or more pulses per revolution (depending on the sensor). There are differences in waveforms for varying RPMs. As RPM increases, the period (T) and pulse width (W) becomes smaller. Both period and pulse width are proportional to RPM. The frequency (or period) of either A or B signal gives the RPM of the joint. Counting the number of pulses gives motor position. A-B phase provides motor direction.

5 Arduino Code:

The Arduino Code gotten from Arduino Playground is given below. I included a line in the void loop() function to measure the angle of the joint or motor. Since I am using the LPD3806 600BM G5 24C Rotary Encoder, and it produces 2400 pulses for one complete revolution which is 360°, Therefore, the angle is given by:

$$\text{Angle} = \text{Pulse Count} \times \frac{360}{2400} \tag{1}$$
$$\therefore \text{Angle} = \text{Pulse Count} \times \frac{3}{20}$$

Note that in the code, the pulses keep counting to $2^{16} = 65536$ after which it resets. In order to avoid this, the pulse count needs to be reset to 0 after it gets to 2400. For my application, the joint would not make a complete 360° so there was no need to reset it.

Another important point when installing the encoder is that the direction of positive rotation matters. If the encoder is installed such that its rotation is in the counter clockwise position, the variable encoder0Pos will start from 65536 and start counting down.

Listing 1: Arduino Code

```
#define encoder0PinA 2
#define encoder0PinB 3

volatile unsigned int encoder0Pos = 0;

float angle = 0.0;

void setup() {
  pinMode(encoder0PinA, INPUT);
  pinMode(encoder0PinB, INPUT);

  void doEncoderA();
  void doEncoderB();

  // encoder pin on interrupt 0 (pin 2)
  attachInterrupt(0, doEncoderA, CHANGE);

  // encoder pin on interrupt 1 (pin 3)
  attachInterrupt(1, doEncoderB, CHANGE);

  Serial.begin (9600);
}


void loop(){
  Serial.println (encoder0Pos, DEC);
  angle = encoder0Pos * (3.0/20.0);
  Serial.println (angle);
}

void doEncoderA(){
  // look for a low-to-high on channel A
  if (digitalRead(encoder0PinA) == HIGH) {
    // check channel B to see which way encoder is turning
    if (digitalRead(encoder0PinB) == LOW) {
      encoder0Pos = encoder0Pos + 1; // CW
    }

    else {
      encoder0Pos = encoder0Pos - 1; // CCW

    }
  }

  else {  // must be a high-to-low edge on channel A
    // check channel B to see which way encoder is turning
    if (digitalRead(encoder0PinB) == HIGH) {
      encoder0Pos = encoder0Pos + 1; // CW
    }
    else {
      encoder0Pos = encoder0Pos - 1; // CCW
      }
   }
}


void doEncoderB() {
   // look for a low-to-high on channel B
   if (digitalRead(encoder0PinB) == HIGH) {
      // check channel A to see which way encoder is turning
      if (digitalRead(encoder0PinA) == HIGH) {
         encoder0Pos = encoder0Pos + 1; // CW
      }
      else {
         encoder0Pos = encoder0Pos - 1; // CCW
      }
   }

   // Look for a high-to-low on channel B
   else {
      // check channel B to see which way encoder is turning
      if (digitalRead(encoder0PinA) == LOW) {
         encoder0Pos = encoder0Pos + 1; // CW
      }
      else {
         encoder0Pos = encoder0Pos - 1; // CCW
      }
   }
}
```