Rotary Encoder

Divyashree Ayyagari and Riddhi Dalmia

July 2024

0.1 Rotary Encoder:

A rotary encoder is an electromechanical device used to measure a rotating shaft's rotation angle, position, or movement. It converts the angular position of a shaft into an analog or digital signal.

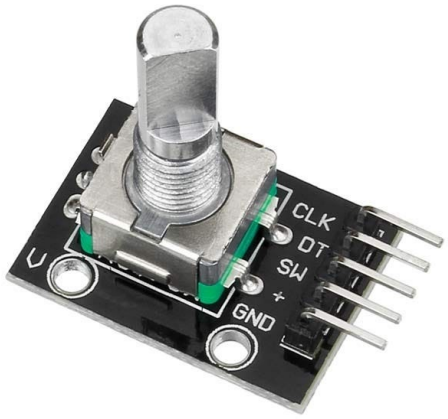

Figure 1: Rotary Encoder

0.1.1 Types of Rotary Encoder:

Types of Rotary Encoders are:
1. Incremental Rotary Encoders: Measure the relative position, typically counting pulses.
2. Absolute Rotary Encoders: Provide a unique code for each shaft position, measuring the exact position.

0.1.2 Working Principle:

The encoder has a disk with evenly spaced contact zones that are connected to the common pin C and two other separate contact pins A and B. When the disk starts rotating step by step, the pins A and B will start making contact with the common pin and the two square wave output signals will be generated accordingly. Any of the two outputs can be used for determining the rotated position if we just count the pulses of the signal. However, if we want to determine the rotation direction as well, we need to consider both signals simultaneously.

We can notice that the two output signals are displaced at 90 degrees out of phase from each other. If the encoder is rotating clockwise the output A will

be ahead of output B. So if we count the steps each time the signal changes, we can notice that the two output signals have opposite values. Vice versa, if the encoder is rotating counterclockwise, the output signals have equal values. So considering this, we can easily program our controller to read the encoder position and the rotation direction.

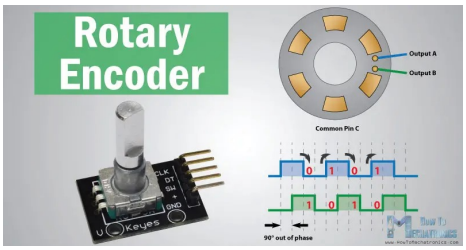

Figure 2: Working Principle

0.1.3 Connections:

A typical rotary encoder has the following pins:
1. GND (Ground)
2. VCC (Power Supply)
3. Output A
4. Output B
5. SW (Switch) - Optional, if the encoder includes a push button.

Connections to an Arduino:
1. GND: Connect to one of the Arduino's GND pins
2. VCC: Connect to the 5V pin on the Arduino
3. Output A: Connect to a digital input pin on the Arduino
4. Output B: Connect to another digital input pin on the Arduino
5. SW (Switch): If you have a switch, connect it to another digital input pin

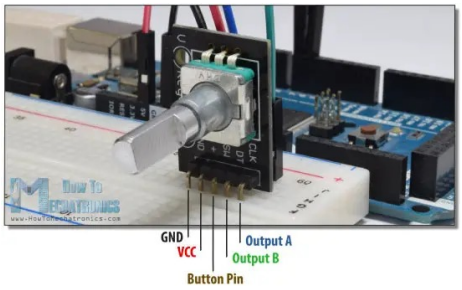

Figure 3: Pins of Rotary Encoder

0.2 Source Code:

```
#define outputA 3
#define outputB 4

int counter = 0;
int aState;
int aLastState;


 void setup()
 {
   pinMode (outputA,INPUT);
   pinMode (outputB,INPUT);

   Serial.begin (9600);
   aLastState = digitalRead(outputA);
 }

 void loop()
 {
   aState = digitalRead(outputA);

   if (aState != aLastState)
   {
     if (digitalRead(outputB) != aState)
     {
       counter ++;
     } else
     {
       counter --;
     }
     Serial.print("Position: ");
     Serial.println(counter);
   }

   aLastState = aState;
 }
```

This code is for reading the position of a rotary encoder using an Arduino. It sets up two pins (8 and 9) to read signals from the encoder. In the setup function, it initializes these pins as inputs and starts the serial communication. It then reads the initial state of one of the encoder outputs (output A). In the loop function, it continuously reads the current state of output A. If this state has changed from the previous state, it checks the state of the second output (output B) to determine the direction of rotation. If output B is different from output A, it increments a counter, indicating clockwise rotation. If output B is the same as output A, it decrements the counter, indicating counterclockwise rotation. The counter's value, representing the encoder's position, is printed to the serial monitor. The previous state of output A is then updated for the next loop iteration.

0.3 Controlling Motor with Rotary Encoder:

0.3.1 Components:

1. DC Motor
2. Motor Driver
3. Arduino Nano
4. Breadboard
5. Connecting Wires
6. Rotary Encoder
7. Battery (12V)

0.3.2 Connections:

Rotary Encoder to Arduino: The connections are as follows:
1. Common Ground (GND): Connect the GND pin of the rotary encoder to the GND pin on the Arduino.
2. Power (VCC): Connect the VCC pin of the rotary encoder to the 5V pin on the Arduino.
3. Output A (CLK): Connect the CLK (or A) pin of the rotary encoder to a digital input pin on the Arduino (e.g., pin 3).
4. Output B (DT): Connect the DT (or B) pin of the rotary encoder to another digital input pin on the Arduino (e.g., pin 4).
5. Switch (SW) [optional]: If your rotary encoder has a built-in switch, connect it to another digital input pin if you plan to use it.

Motor Driver to Arduino: The connections are as follows:
1. Enable Pin (ENA): Connect the ENA pin of the motor driver to a PWM output pin on the Arduino (e.g., pin 5).

2. Input Pins (IN1, IN2): Connect the IN1 and IN2 pins of the motor driver to digital output pins on the Arduino (e.g., pins 8 and 9).

Motor to Motor Driver: Connect the motor terminals to the motor output terminals on the motor driver (usually labeled as OUT1 and OUT2 or M1).

Power Supply: Connect the power supply to the motor driver's power input terminals (usually labeled as VCC and GND). Make sure to connect the GND of the power supply to the GND of the Arduino to ensure a common ground.

0.3.3 Controlling Motor within an Interval:

```
#define outputA 3
#define outputB 4

int counter = 0;
int aState;
int aLastState;

const int pwm = 2 ;
const int in_1 = 8 ;
const int in_2 = 9 ;

void setup()
{
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);

  pinMode(pwm,OUTPUT) ;
  pinMode(in_1,OUTPUT) ;
  pinMode(in_2,OUTPUT) ;

  Serial.begin (9600);
  aLastState = digitalRead(outputA);
}

void loop()
{
  aState = digitalRead(outputA);
  if (aState != aLastState)
  {
    if (digitalRead(outputB) != State)
    {
      counter ++;
    } else {
      counter --;
    }

    Serial.print("Position: ");
    Serial.println(counter);
```

```
  }

  aLastState = State;

  if (counter >= -10 && counter <= 10)
  {
    analogWrite(pwm, 128);
    digitalWrite(in_1, HIGH);
    digitalWrite(in_2, LOW);
  }
  else
  {
    analogWrite(pwm, 0);
    digitalWrite(in_1, LOW);
    digitalWrite(in_2, LOW);
  }
}
```

This code is designed to control a motor based on the position feedback from a rotary encoder. It initializes two pins as inputs to read the encoder signals (output A and output B) and three pins as outputs to control the motor (pwm, inp1, and inp2). In the setup function, it sets the pin modes and reads the initial state of output A. In the loop, it continuously reads the current state of output A and compares it to the last state to detect changes, indicating encoder pulses. Depending on the state of output B, it determines the rotation direction (clockwise or counterclockwise) and updates a counter. The motor speed and direction are then controlled based on the counter value: if the counter is between -10 and 10, the motor runs, otherwise, the motor stops. This setup allows for precise motor control based on the encoder's feedback.

0.3.4 Controlling speed of the Motor:

```
#define outputA 3
#define outputB 4

int counter = 0;
int aState;
int aLastState;

const int pwm = 2 ;
const int in_1 = 8 ;
const int in_2 = 9 ;

void setup()
{
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);

  pinMode(pwm,OUTPUT) ;
  pinMode(in_1,OUTPUT) ;
  pinMode(in_2,OUTPUT) ;

```

```
  Serial.begin (9600);
  aLastState = digitalRead(outputA);
}

void loop()
{
  aState = digitalRead(outputA);
  if (aState != aLastState)
  {
    if (digitalRead(outputB) != aState)
    {
      counter ++;
    }
    else
    {
      counter --;
    }

    Serial.print("Position: ");
    Serial.println(counter);
  }

  aLastState = aState;

  int motorspeed= map(counter, 0, 50, 0, 255);
  analogWrite(pwm, motorspeed);
  digitalWrite(in_1, HIGH);
  digitalWrite(in_2, LOW);
}
```

This code controls a motor's speed based on the position feedback from a rotary encoder. It initializes two pins (‘outputA‘ and ‘outputB‘) as inputs to read the encoder signals and three pins (‘pwm‘, ‘inp1‘, and ‘inp2‘) as outputs to control the motor. During setup, it sets the pin modes and reads the initial state of ‘outputA‘. In the ‘loop‘, it continuously reads the current state of ‘outputA‘ and compares it to the last state to detect changes, indicating encoder pulses. Depending on the state of ‘outputB‘, it determines the rotation direction (clockwise or counterclockwise) and updates a counter accordingly. The motor speed is then controlled by mapping the counter value (ranging from 0 to 50) to a PWM value (ranging from 0 to 255). The mapped PWM value is written to the motor control pin (‘pwm‘), setting the motor speed, while the motor direction is set to run forward. This setup allows the motor speed to be proportional to the encoder's position feedback.