26.Rotary_Encoder

Introduction

A rotary encoder is an electro-mechanical device that converts the angular position or motion of a shaft or axle to analog or digital code. Rotary encoders are usually placed at the side which is perpendicular to the shaft. They act as sensors for detecting angle, speed, length, position, and acceleration in automation field.

Hardware Required

- ✓ 1 * Raspberry Pi
- ✓ 1 * Breadboard
- ✓ Several Jumper Wires
- ✓ 1 * Rotary Encoder Module
- ✓ 1 * T-Extension Board
- ✓ 1 * 40-pin Cable

Principle

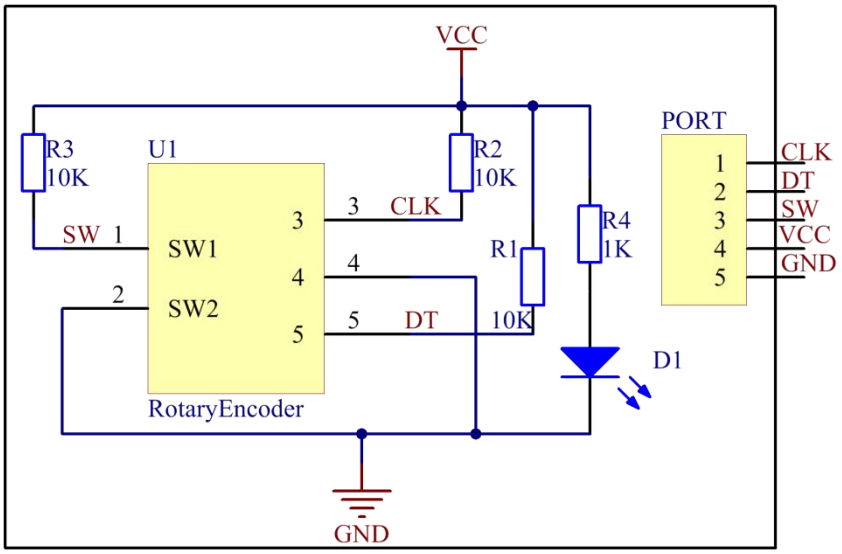
A rotary encoder is a type of position sensor which is used for determining the angular position of a rotating shaft. It generates an electrical signal, either analog or digital, according to the rotational movement.

There are many different types of rotary encoders which are classified by either Output Signal or Sensing Technology. The particular rotary encoder that we will use in this tutorial is an incremental rotary encoder and it's the simplest position sensor to measure rotation.

This rotary encoder is also known as quadrature encoder or relative rotary encoder and its output is a series of square wave pulses.

It shows that if output 1 is high and output 2 is high, then the switch rotates clockwise; if output 1 is high and output 2 is low, then the switch rotates counterclockwise. As a result, during SCM programming, if output 1 is high, then you can tell whether the rotary encoder rotates left or right as long as you know the state of output 2.

26.Rotary_Encoder



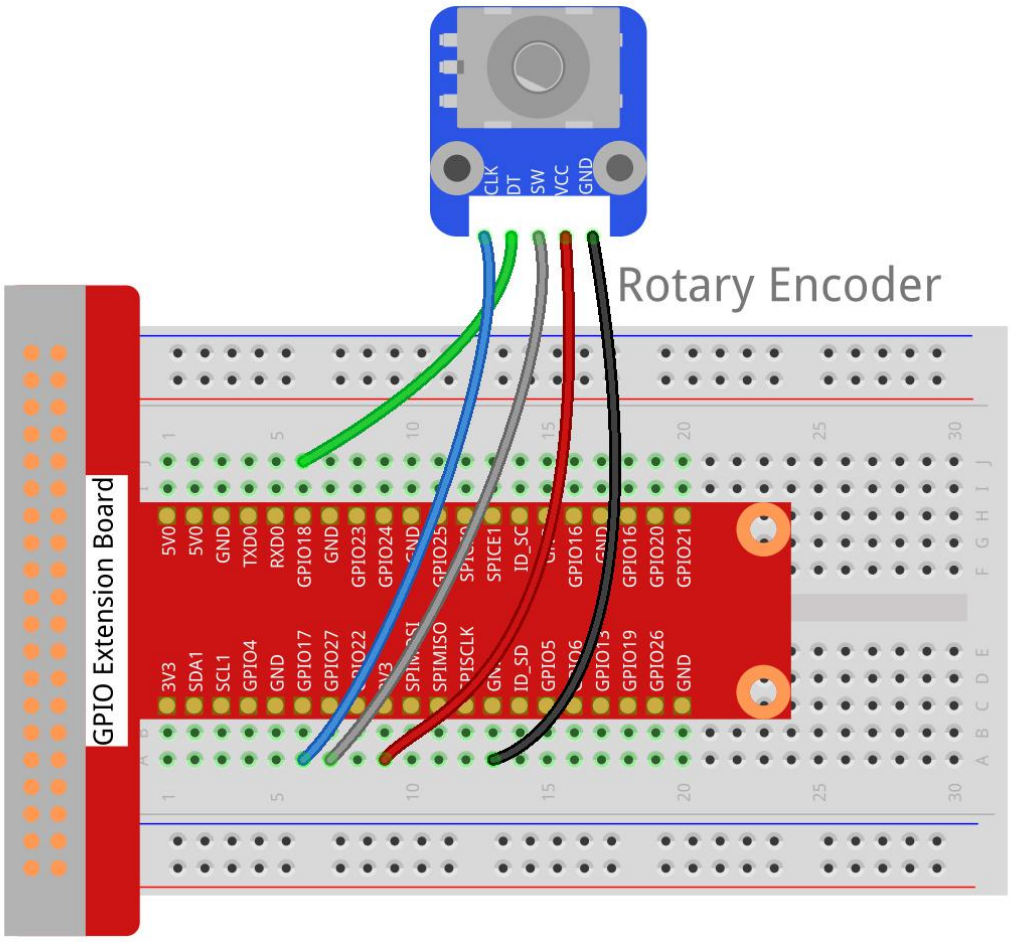
It is summarized by using oscilloscope to observe the output waveform of CLK and DT and operating the rotary encoder. You can try yourself.

Experimental Procedures

Step 1: Build the circuit

Raspberry Pi	T-Cobbler	Rotary Encoder Module
GPIO0	GPIO17	CLK
GPIO1	GPIO18	DT
GPIO2	GPIO27	SW
5V	5V0	VCC
GND	GND	GND

26.Rotary_Encoder



For C Language Users

Step 2: Open the code file.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/26.Rotary_Encoder
```

Step 3: Compile the code.

```
gcc 26.Rotary_Encoder.c -o Rotary_Encoder.out -lwiringPi
```

Step 4: Run the executable file above.

```
sudo ./Rotary_Encoder.out
```

Code

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
```

26.Rotary_Encoder

```
#include <wiringPi.h>

#define    RoAPin      0     //clk
#define    RoBPin      1     //dt
#define    SWPin       2     //sw

static volatile int globalCounter = 0 ;

unsigned char flag;
unsigned char Last_RoB_Status;
unsigned char Current_RoB_Status;

void btnISR(void)
{
    globalCounter = 0;
}

void rotaryDeal(void)
{
    Last_RoB_Status = digitalRead(RoBPin);

    while(!digitalRead(RoAPin)){
        Current_RoB_Status = digitalRead(RoBPin);
        flag = 1;
    }

    if(flag == 1){
        flag = 0;
        if((Last_RoB_Status == 0)&&(Current_RoB_Status == 1)){
```

```
            globalCounter ++;
        }
        if((Last_RoB_Status == 1)&&(Current_RoB_Status == 0)){
            globalCounter --;
        }
    }
}

int main(void)
{
    if(wiringPiSetup() < 0){
        fprintf(stderr, "Unable to setup wiringPi:%s\n",strerror(errno));
        return 1;
    }

    pinMode(SWPin, INPUT);
    pinMode(RoAPin, INPUT);
    pinMode(RoBPin, INPUT);

    pullUpDnControl(SWPin, PUD_UP);

    if(wiringPiISR(SWPin, INT_EDGE_FALLING, &btnISR) < 0){
        fprintf(stderr, "Unable to init ISR\n",strerror(errno));
        return 1;
    }

    int tmp = 0;

    while(1){
```

```
        rotaryDeal();
        if (tmp != globalCounter){
            printf("%d\n", globalCounter);
            tmp = globalCounter;
        }
    }

    return 0;
}
```

For Python Language Users

Step 2: Open the code file.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 3: Run.

```
sudo python3 26.Rotary_Encoder.py
```

Now rotate the shaft of the rotary encoder, and the value printed on the screen will change. Rotate the rotary encoder clockwise, the value will increase; Rotate it counterclockwise, the value will decrease; Press the rotary encoder, the value will be reset to 0.

Code

The code here is for Python3, if you need for Python2, please open the code with the suffix py2 in the attachment.

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

RoAPin = 11       # CLK Pin
RoBPin = 12       # DT Pin
BtnPin = 13       # Button Pin
```

26.Rotary_Encoder

```
globalCounter = 0


flag = 0
Last_RoB_Status = 0
Current_RoB_Status = 0


def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(RoAPin, GPIO.IN)     # input mode
    GPIO.setup(RoBPin, GPIO.IN)
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)


def rotaryDeal():
    global flag
    global Last_RoB_Status
    global Current_RoB_Status
    global globalCounter
    Last_RoB_Status = GPIO.input(RoBPin)
    while(not GPIO.input(RoAPin)):
        Current_RoB_Status = GPIO.input(RoBPin)
        flag = 1
    if flag == 1:
        flag = 0
        if (Last_RoB_Status == 0) and (Current_RoB_Status == 1):
            globalCounter = globalCounter + 1
        if (Last_RoB_Status == 1) and (Current_RoB_Status == 0):
            globalCounter = globalCounter - 1
```

```
def btnISR(channel):
    global globalCounter
    globalCounter = 0

def loop():
    global globalCounter
    tmp = 0 # Rotary Temperary

    GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=btnISR)
    while True:
        rotaryDeal()
        if tmp != globalCounter:
            print ('globalCounter = %d' % globalCounter)
            tmp = globalCounter

def destroy():
    GPIO.cleanup()                    # Release resource

if __name__ == '__main__':        # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:    # When 'Ctrl+C' is pressed, the child program
destroy() will be    executed.
        destroy()
```

Phenomenon Picture

26.Rotary_Encoder