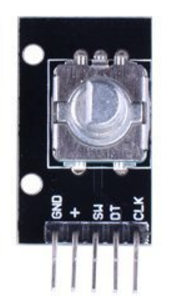
Rotary Encoder  


Overview

An incremental rotary encoder (also called a *shaft* encoder) converts angular motion of a shaft to a series of digital pulses that can be counted to determine how many times (and in what direction) a shaft has been rotated. Combined with control logic, rotary encoders can be used to measure turning speed, distance, and position. Rotary encoders are used in industrial controls, robotics, computer mice and trackballs, and other applications that require precise but unlimited rotation.

Experimental Materials

Raspberry Pi x1

Breadboard x1

Rotary encoder x1

Dupont jumper wires

Experimental Procedure

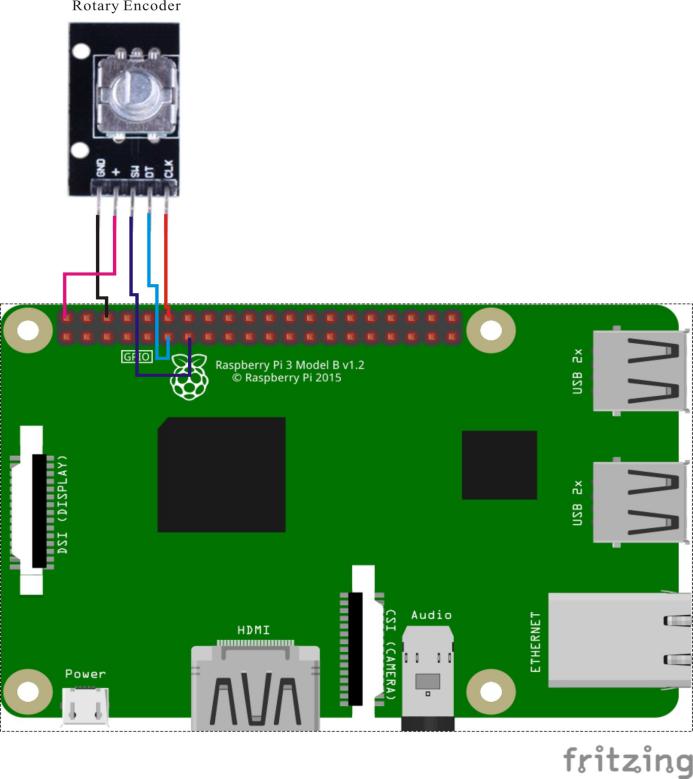
1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GPIO library, and wiringPi library as described in READ\_ME\_FIRST.TXT.
2. Install the rotary encoder on your breadboard and use Dupont jumper wires to connect in to your Raspberry Pi as illustrated in the Wiring Diagram below.
3. Execute the sample code stored in this experiment’s subfolder.

If using C, compile and execute the C code:  
cd Code/C  
gcc rotaryEncoder.c -o rotaryEncoder.out –lwiringPi  
./ rotaryEncoder.out

If using Python, launch the Python script:  
cd Code/Python  
python rotaryEncoder.py

1. Make experimental observations. As you turn the rotating shaft on the sensor in one direction, a counter displayed on Raspberry Pi command line interface increases. As you turn the shaft in the other direction, the counter decreases. You can zero the counter by pressing the small button on the sensor.

Wiring Diagram



Rotary-Encoder pin position:

"DT" ↔ Raspberry Pi pin 11

"CLK" ↔ Raspberry Pi pin 12

"SW" ↔ Raspberry Pi pin 13

"+" ↔ Raspberry Pi +5V

"-" ↔ Raspberry Pi GND

Sample Code

Python Code

#!/usr/bin/env python

import RPi.GPIO as GPIO

import time

RoAPin = 11

RoBPin = 12

BtnPin = 13

globalCounter = 0

flag = 0

Last\_RoB\_Status = 0

Current\_RoB\_Status = 0

def setup():

GPIO.setmode(GPIO.BOARD)

GPIO.setup(RoAPin, GPIO.IN)

GPIO.setup(RoBPin, GPIO.IN)

GPIO.setup(BtnPin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.add\_event\_detect(BtnPin, GPIO.FALLING, callback=btnISR)

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

while True:

rotaryDeal()

if tmp != globalCounter:

print 'globalCounter = %d' % globalCounter

tmp = globalCounter

def destroy():

GPIO.cleanup()

if \_\_name\_\_ == '\_\_main\_\_':

setup()

try:

loop()

except KeyboardInterrupt:

destroy()

C Code

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <stdlib.h>

#include <wiringPi.h>

#define SWPin 2

#define RoAPin 0

#define RoBPin 1

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)

{

int temp =0;

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;

}

while(1)

{

rotaryDeal();

if(temp != globalCounter)

{

printf("%d\n", globalCounter);

temp = globalCounter;

}

}

return 0;

}