Contents

Side Panel 1 - rotary Encoder

How does it work?

I was fumbling around with the Totem Mini Lab's Side Panel 1, especially with the rotary encoder that s mounted on this side board. At first, I could not make heads or tails from it, but, using my oscilloscope, I've figured it out now.

Figure 1 - Rotary Encoder

Contacts

It appears that both connections - labelled 'Contact A' and 'Contact B' give pulses. In the pictures below, Yellow indicates Contact A, and Blue indicates Contact B. The contacts are normally open and, when closed, they pull the outputs to ground.

Pulses

When turning the know clockwise, the pulse from contact B comes just before the pulse from contact A and just the other way around when turning counter-clockwise (have a look at the images below). Now we can do some coding to decode this behaviour.

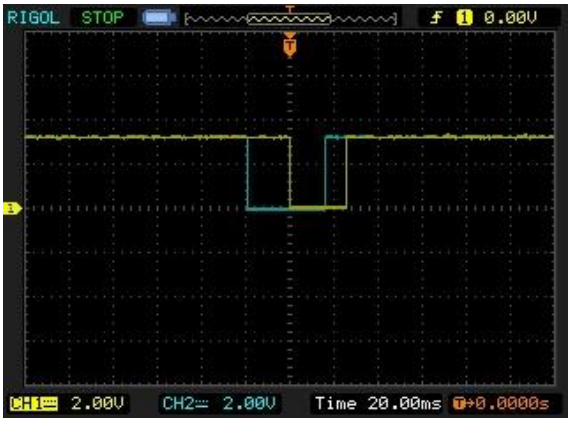

Figure 2 - Rotary Encoder, turning clockwise

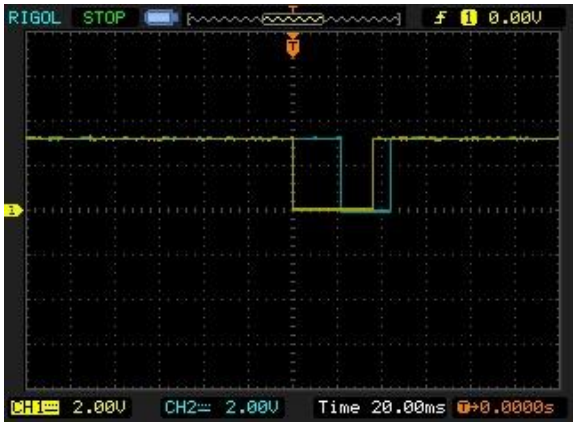

Figure 3 - Rotary Encoder, turning counter-clockwise

Algorithm

So, what does this code have to do? It has to detect the direction of motion, right (clockwise) or left (counter-clockwise). We are going to have a closer look at what happens exactly.

What happens?

Let's call the contacts A and B. When we don't do anything, both pins will be high, so the value on the pins will be binary '11', or decimal 3. We are going to use two pins on our Arduino to connect to these contacts, I used:

- pin 16 (A2) for Contact A
- pin 17 (A3) for Contact B

Values

How do we create a decimal value from two individual pins? We do that in a calculation:

Decimal value = 2 x (pin 16 value) + pin 17 value

This involves a multiplication and an addition. We can change the calculation to make it faster:

*Decimal value = (pin 16 value)***<<1 | *pin 17 value***

In this calculation '***<<1***' means: shift the value 1 bit to the left (which is equivalent to a multiplication by 2), and the '***|***' stands for the logical OR operation (in this case, because we just shifted a value 1 bit to the left, the rightmost bit has become a zero and OR'ing the value of one other bit will in effect add it to the other value).

Possibilities

In the table below we can see all possible binary combinations and their values.

A (16)	B (17)	Decimal
0	**0**	**0**
0	**1**	**1**
1	**0**	**2**
1	**1**	**3**

Figure 4 - Rotary Encoder, possible combinations

When turning the button these values change. The pins are normally high, so the value is decimal 3 (11) when we don't do anything.

Clockwise

When we turn clockwise, contact B is made first, resulting in a value of 1, and then contact A is made, giving us the value 0, then contact B is released, giving us value 2 and at last contact B is released, then we're back at value 3. So, the sequence of changing values is 3, 1, 0, 2, 3.

Counter-clockwise

When turning counter-clockwise the sequence of events is reverse, so the values will be reverse as well: 3, 2, 0, 1, 3.

Explanation

We can see this in these pictures (remember, B is blue, A is yellow):

Figure 5 - Rotary Encoder, clockwise values

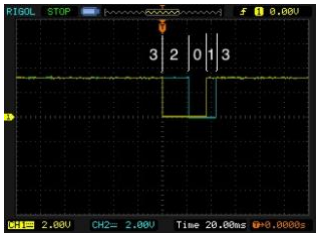

Figure 6 – Rotary Encoder, counter-clockwise values

Transitions

The sequences we mentioned each signify one click of the knob. When we take a closer look, the first two values of each sequence are unique per direction. So we only have to look at the transitions from 3 to 1 (indicating a clockwise turn) and from 3 to 2 (indicating counter clockwise turn).

Detection

To be able to detect a transition, we would have to store the value of the two pins between two loops. Then, next time, we can compare them. We are only interested in situations where:

- the previous value is 3
- the current value is different from the previous value
- the current value is 1 or 2

If all that is true and the current value is 1, this indicates a clockwise turn, whereas when the current value is a 2 this indicates a counter-clockwise turn.

Okay, what can we do with it?

Example program

Now we know how the Rotary Encoder works and how we can detect which way it turns, we can put it to work. To show this in code, I took a program that displays a binary counter using eight bits, eight LED's attached to the Arduino. The speed was set to change at a regular interval using a 'wait-time'. Now let's use the Rotary Encoder to be able to change the speed.

Actions

When turning clockwise, we want the speed (or frequency of the counting) to go up, so the wait-time has to go down. And of course, when turning counter-clockwise, we want the speed (or frequency of the counting) to go down, so the wait-time has to go up. I chose an increment / decrement value of 10, because otherwise the change in speed was barely noticeable and we had to turn very long (many clicks) to achieve a significant change in speed.

You can find the code of the sketch on the next page(s) or download it at my GitHub page.

The Arduino sketch

```
// ---------------------------------------------------------------------------
// Program : TotemDuino_Rotary_Decoder.ino
// Author  : Gerard Wassink
// Date    : December, 2017
//
// Function: count from 0-255 with an 8 bit counter in LEDs
//         use a Rotary Encoder to speed things up or slow them down
//         Used for explanation of the Totem Side Panel 1
//
// ---------------------------------------------------------------------------
//         GNU LICENSE CONDITIONS
// ---------------------------------------------------------------------------
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation; either version 2 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// ---------------------------------------------------------------------------
//      Copyright (C) 2017 Gerard Wassink
// ---------------------------------------------------------------------------

int num = 0;                  // Number will count up (0-255)
int LED[8] = {9,8,7,6,5,4,3,2};     // Pin numbers used for LEDs

unsigned long now = millis();       // Keep track
unsigned long prv = millis();       //  of time
long wait_time = 100;             // initial speed

int contactA = 16;              // Input pin Rotate A
int contactB = 17;              // Input pin Rotate B
int A_val = 0;                  // Input value A
int B_val = 0;                  // Input value B
int cur_val = 3;                // Current value
int prev_val = 3;               // Previous value

// ---------------------------------------------------------------------------------
// the setup function runs only once when you press reset or power the board
// ---------------------------------------------------------------------------------
void setup() {
  Serial.begin(9600);           // Set up Serial
  Serial.println("TotemDuino_Rotary_Encoder"); // Greet our public

  for (int i=0; i<=7; i++) {      // Set
    pinMode(LED[i], OUTPUT);      //  all LEDs
  }// for                        //  to output
  pinMode(A2, INPUT_PULLUP);      // A2 = input, with pull-up activated
  pinMode(A3, INPUT_PULLUP);      // A3 = input, with pull-up activated
} // setup

// ---------------------------------------------------------------------------------
// the loop function runs over and over again, forever
// ---------------------------------------------------------------------------------
void loop() {
  //---
  //--- Code for the eight bit binary counter on output LEDs
  //---
  now = millis();               // Determine time
  if ((now - prv) >= wait_time) {   // if time passed
    int n = num;                // work with n to leave num intact
    for (int i=0; i<=7; i++) {      // For all LEDs
      digitalWrite(LED[i], n & 1);  // Put value of right most bit to LED
      n >>= 1;                  // Shift right, next bit
    }// for
    if (++num > 255) num = 0;       // Increment and roll over when over 255
```

```
    prv += wait_time;              // Save the time
  }// if

  //---
  //--- Code to handle the Rotary Encoder clicks
  //---
  A_val = digitalRead(contactA);     // read Rotate A
  B_val = digitalRead(contactB);     // read Rotate B
  cur_val = B_val<<1 | A_val;        // calculate current value
  if (prev_val == 3) {               // previous value 3?
    if (cur_val == 1) {              // clockwise?
      wait_time -= 10;               // speed up (decrease wait time)
      Serial.print("wait_time: ");   //   display new value
      Serial.print(wait_time);       //   of the
      Serial.println("");           //     wait-time
    } else if (cur_val == 2) {       // counter-clockwise?
      wait_time += 10;               // speed down (increase wait time)
      Serial.print("wait_time: ");   //   display new value
      Serial.print(wait_time);       //   of the
      Serial.println("");           //     wait-time
    }// if
    if (wait_time < 0) {             // make sure
      wait_time = 0;                 // wait time doesn't get under 10
    }// if
  }// if
  prev_val = cur_val;                // save current to previous value

}// loop
```

List of figures