INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO.

INTRODUCCIÓN A LOS MICROCONTROLADORES.

REPORTE PRÁCTICA 5.

PROFESOR: AGUILAR SANCHEZ FERNANDO

Grupo: 3CM6

Alumno: Rojas Alvarado Luis Enrique

Boleta: 2014010995

Link de vídeo explicativo: https://youtu.be/7EBejp8OICM

Práctica 6: Contador de 0 a 9 sin rebotes

Introducción teórica.

Las láminas metálicas utilizadas en la construcción de las llaves/botones poseen, inherentemente, elasticidad. Por ello al intentar ponerlas en contacto se genera un choque que produce un movimiento en sentido contrario que aleja las láminas. Este proceso se repite hasta disipar la energía cinética adquirida por la lámina móvil.

Este fenómeno es conocido como efecto rebote. El efecto se manifiesta en el hecho que el cierre (apertura) del circuito no es instantáneo. Durante un cierto tiempo la llave (ó el botón) oscila entre cerrado y abierto. El fenómeno se puede dar tanto al cerrar el circuito como al abrirlo. [1]

De todos es conocido que, al cerrar un interruptor, se produce un rebote mecánico de sus contactos que no se puede evitar y consecuentemente, estos saltos son lo que producen más de un cierre del circuito, (esto que en electricidad, tiene una importancia relativa, cuando se trata de electrónica digital, es un problema muy grave), lo que queríamos era un único pulso, o sea que ha aparecido el rebote, produciendo un número indeterminado de pulsos, que serán considerados como datos a tratar. [2]

La implementación de sistemas antirebote, como medio de Þ ltrado para transiciones no deseadas, en señales digitales de entrada a sistemas de automatización se realiza normalmente mediante algoritmos que hacen parte íntegra del cuerpo lógico de control. Independiente del tipo de sensor en la puerta, estando abierta o cerrada, que no implican un cambio de estado y aperturas o cierres sin presencia de rebotes.

Como solución en la eliminación de rebotes se acostumbra a incluir contadores y retardos de tiempo como medios de filtrado, sin embargo, estos medios son un obstáculo para la detección de transiciones que ocurren sin la presencia de rebotes. Se muestra en este artículo una solución íntegra al filtrado de rebotes que tiene presente una detección rápida, cuando los cambios de estado se realizan mediante transiciones limpias o sólidas y sin perjuicio del filtrado.

El filtrado de señales con presencia de rebotes es un procedimiento normal en el diseño y validación de sistemas digitales. Frecuentemente se aborda este objetivo con el empleo de biestables, que mediante una señal de sincronismo de frecuencia adecuada (normalmente baja), hacen sensible un sistema a los cambios en una señal de entrada, únicamente a ciertos intervalos de tiempo. Los intervalos son suficientes para considerar que la señal alcanza un estado estable durante este periodo de tiempo. Por lo que se hace automático dentro de un programa de acuerdo con la arquitectura del microcontrolador que vamos a utilizar.[3]

Desarrollo experimental

1.- Diseñe un programa colocando en el Puerto B un Display. Coloque un Push Button en la terminal 0 del Puerto D para incrementar su cuenta del 0 al 9 sin rebotes.

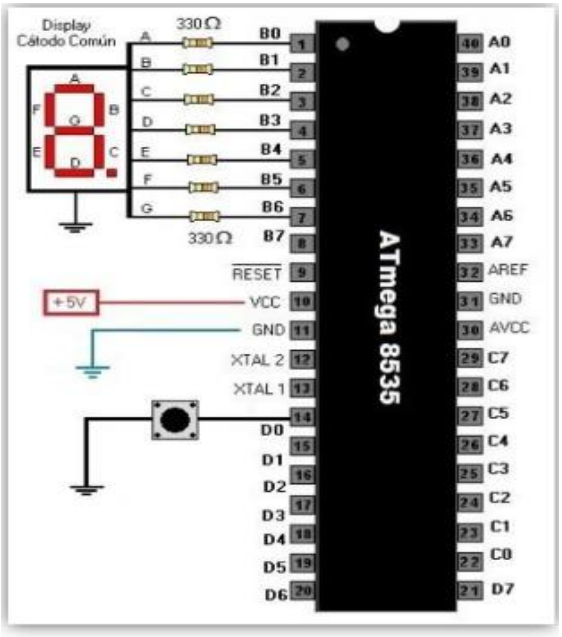


Figura 1. Circuito para el contador de 0 a 9 sin rebotes.

Código

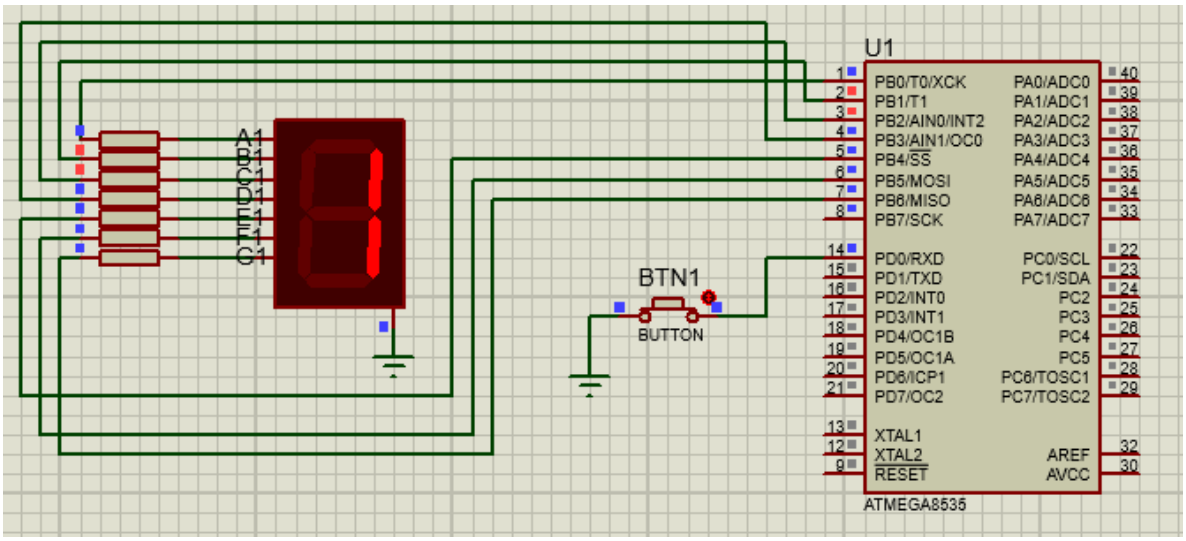
```
/*****************************************************
This program was created by the
CodeWizardAVR V2.60 Evaluation
Automatic Program Generator
© Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 05/11/2020
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>
```

```
#include <delay.h>
#define boton PIND.0
bit botonp;
bit botona;
unsigned char var;
const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f
};


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<D
DA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PO
RTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
ut
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<D
DB1) | (1<<DDB0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PO
RTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<D
DC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PO
RTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<D
DD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=P
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PO
RTD2) | (0<<PORTD1) | (1<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
```

```
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
      TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
      TCNT2=0x00;
      OCR2=0x00;

      // Timer(s)/Counter(s) Interrupt(s) initialization
      TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

      // External Interrupt(s) initialization
      // INT0: Off
      // INT1: Off
      // INT2: Off
      MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
      MCUCSR=(0<<ISC2);

      // USART initialization
      // USART disabled
      UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

      // Analog Comparator initialization
      // Analog Comparator: Off
      ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
      SFIOR=(0<<ACME);

      // ADC initialization
      // ADC disabled
      ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);

      // SPI initialization
      // SPI disabled
```

```
129.       SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA)
    | (0<<SPR1) | (0<<SPR0);
130.
131.       // TWI initialization
132.       // TWI disabled
133.       TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
134.
135.       while (1){
136.           if (boton==0)
137.               botona=0;
138.           else
139.               botona=1;
140.           if ((botonp==1)&&(botona==0)){ //hubo cambio de flanco de 1 a 0
141.               var++; //Se incrementa la variable
142.               if (var==10)
143.                   var=0;
144.                   delay_ms(40); //Se coloca retardo de 40mS para eliminar rebote
    s
145.           }
146.           if ((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
147.               delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
148.
149.           PORTB=tabla7segmentos [var];
150.           botonp=botona;

151.           }
152.       }
```

Simulación



Conclusiones

En esta práctica eliminamos el rebote, que es cuando hay una entrada, esta no es capturada correctamente y por supuesto, se ve reflejado en la salida. Esto lo resolvemos a causa del programa de la práctica anterior, agregando un delay para que de cierta manera los datos tengan un intervalo de entrada y no se genere traslape y por consiguiente errores. Ahora detectamos el flanco de subida para saber cuándo se terminó de pulsar el botón y agregando el delay.

Referencias

[1] https://www.fing.edu.uy/inco/cursos/firmware/teorico/clase05-manejoES.pdf

[2] http://www.geocities.ws/allcircuits/teoriadigital2.html

[3] http://www.scielo.org.co/pdf/tecn/v15n29/v15n29a10.pdf