INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

INTRODUCCIÓN A LOS MICROCONTROLADORES

PROFESOR:

Aguilar Sanchez Fernando

ALUMNOS:

Arruti Sánchez Alondra
Carrillo Soto Cristian Eduardo

Grupo: 3CM14

Fecha de Entrega: 05/10/22

PRÁCTICA 4

Contador de 0 a 9

Introducción

En electrónica digital, un contador es un circuito digital capaz de contar sucesos electrónicos, tales como impulsos, avanzando a través de una secuencia de estados binarios. El contador electrónico digital es muy útil por ello en la actualidad estamos rodeados de dispositivos que disponen de algún tipo de contador digital, incluso en la mayoría de los electrodomésticos vienen equipados con uno.

Se denomina ruido eléctrico, a todas aquellas señales de interferencias, de origen eléctrico, no deseadas y que están unidas a la señal principal, o útil, de manera que la pueden alterar produciendo efectos que pueden ser más o menos perjudiciales.

Cuando la señal principal es analógica, el ruido será perjudicial en la medida que lo sea su amplitud respecto a la señal principal. Cuando las señales son digitales, si el ruido no es capaz de producir un cambio de estado, dicho ruido será irrelevante.

Sin descartar que el ruido nunca se puede eliminar en su totalidad. La principal fuente de ruido es la red que suministra la energía eléctrica, y lo es porque alrededor de los conductores se produce un campo magnético a la frecuencia de 50 o 60 Hz. Además, por estos conductores se propagan los parásitos o el ruido producido por otros dispositivos eléctricos o electrónicos.

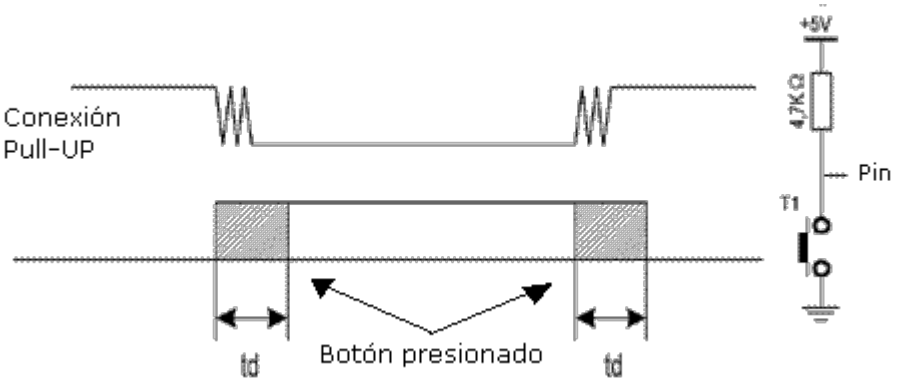
De todas formas, las perturbaciones más perjudiciales son las que se producen interiormente o muy cerca de la instalación. Normalmente son picos y oscilaciones de tensión causados por bruscas variaciones de intensidad en el proceso de conexión y desconexión de los dispositivos de mayor consumo.

Pulsadores o interruptores, hay toneladas de ellos en el hogar. Un interruptor es un dispositivo simple con dos posiciones, EN y AP (ENcendido y APagado). Una clase de interruptor que se usa cada día es el interruptor de la luz. Cuando se conecta, dentro del interruptor, dos cables son unidos, lo que permite fluir a la corriente que enciende la luz o la licuadora. Cuando este se desconecta, los dos cables son desunidos y se corta el flujo de la corriente.

En definitiva, se trata de un mecanismo simple (los hay muy sofisticados), constituido por un par de contactos eléctricos que se unen o separan por medios mecánicos. En electricidad, los falsos contactos que se producen al ser utilizados normalmente, en algunos casos producen una chispa debido a la corriente que atraviesa los contactos, provocando que se quemen en partes y ennegreciendo los contactos eléctricos, lo que a la larga acaba deteriorando dichos contactos.

La chispa se produce siempre al separar los contactos (desconectar), en ocasiones parece que también salta al conectarlos, eso es debido a los rebotes mecánicos que se producen al cambiar de estado.

Esto que en electricidad se considera normal, en electrónica es un verdadero nido de problemas, debido a dichos falsos contactos. Por su propia naturaleza, al cambiar de posición un interruptor, los contactos chocan entre sí y esto significa una serie de falsos contactos que se reproducen de un modo sin control, por lo que se generan los temidos rebotes (debounce en inglés), estos rebotes, se producen incluso cuando unimos dos cables desnudos, simulando un interruptor o pulsador.



Códigos

1. Código de la configuración de los periféricos

```
1.  // I/O Registers definitions
2.  #include <mega8535.h>
3.  #define boton PIND.0
4.  const char tabCatodo [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
5.  unsigned char var;
6.  // Declare your global variables here
7.  void main(void)
8.  {
9.  // Declare your local variables here
10. // Input/Output Ports initialization
11. // Port A initialization
12. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
13. DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) |
    (0<<DDA0);
14. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
15. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) |
    (0<<PORTA1) | (0<<PORTA0);
16. // Port B initialization
17. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
18. DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) |
    (1<<DDB0);
19. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
20. PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
    (0<<PORTB1) | (0<<PORTB0);
21. // Port C initialization
22. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
23. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
    (0<<DDC0);
24. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
25. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
    (0<<PORTC1) | (0<<PORTC0);
26. // Port D initialization
27. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
28. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
    (0<<DDD0);
29. // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
30. PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PORTD2) |
    (1<<PORTD1) | (1<<PORTD0);
31. // Timer/Counter 0 initialization
32. // Clock source: System Clock
```

```
33. // Clock value: Timer 0 Stopped
34. // Mode: Normal top=0xFF
35. // OC0 output: Disconnected
36. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
37. TCNT0=0x00;
38. OCR0=0x00;
39. // Timer/Counter 1 initialization
40. // Clock source: System Clock
41. // Clock value: Timer1 Stopped
42. // Mode: Normal top=0xFFFF
43. // OC1A output: Disconnected
44. // OC1B output: Disconnected
45. // Noise Canceler: Off
46. // Input Capture on Falling Edge
47. // Timer1 Overflow Interrupt: Off
48. // Input Capture Interrupt: Off
49. // Compare A Match Interrupt: Off
50. // Compare B Match Interrupt: Off
51. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
52. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
53. TCNT1H=0x00;
54. TCNT1L=0x00;
55. ICR1H=0x00;
56. ICR1L=0x00;
57. OCR1AH=0x00;
58. OCR1AL=0x00;
59. OCR1BH=0x00;
60. OCR1BL=0x00;
61. // Timer/Counter 2 initialization
62. // Clock source: System Clock
63. // Clock value: Timer2 Stopped
64. // Mode: Normal top=0xFF
65. // OC2 output: Disconnected
66. ASSR=0<<AS2;
67. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
68. TCNT2=0x00;
69. OCR2=0x00;
70. // Timer(s)/Counter(s) Interrupt(s) initialization
71. TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0)
    | (0<<TOIE0);
72. // External Interrupt(s) initialization
73. // INT0: Off
74. // INT1: Off
75. // INT2: Off
76. MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
```

```
77. MCUCSR=(0<<ISC2);
78. // USART initialization
79. // USART disabled
80. UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) |
    (0<<TXB8);
81. // Analog Comparator initialization
82. // Analog Comparator: Off
83. // The Analog Comparator's positive input is
84. // connected to the AIN0 pin
85. // The Analog Comparator's negative input is
86. // connected to the AIN1 pin
87. ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
88. SFIOR=(0<<ACME);
89. // ADC initialization
90. // ADC disabled
91. ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) |
    (0<<ADPS0);
92. // SPI initialization
93. // SPI disabled
94. SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
    (0<<SPR0);
95. // TWI initialization
96. // TWI disabled
97. TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
```

2. Código del programa principal en C

```
while (1)
      {
      if (boton==0)
        var++;

      if (var==10)
        var=0;

      PORTB = tabCatodo [var];

      // Place your code here

      };
```

Circuitos

1. Circuito simulado en Proteus

En la figura 1 se puede observar el contador que en este caso es ascendente y por ende inicia en 0, pero no tendrá el comportamiento deseado, puesto que un "push button" genera algunos errores de pulso y además se debe tener en cuenta la afectación que ocasiona el ruido en el circuito.

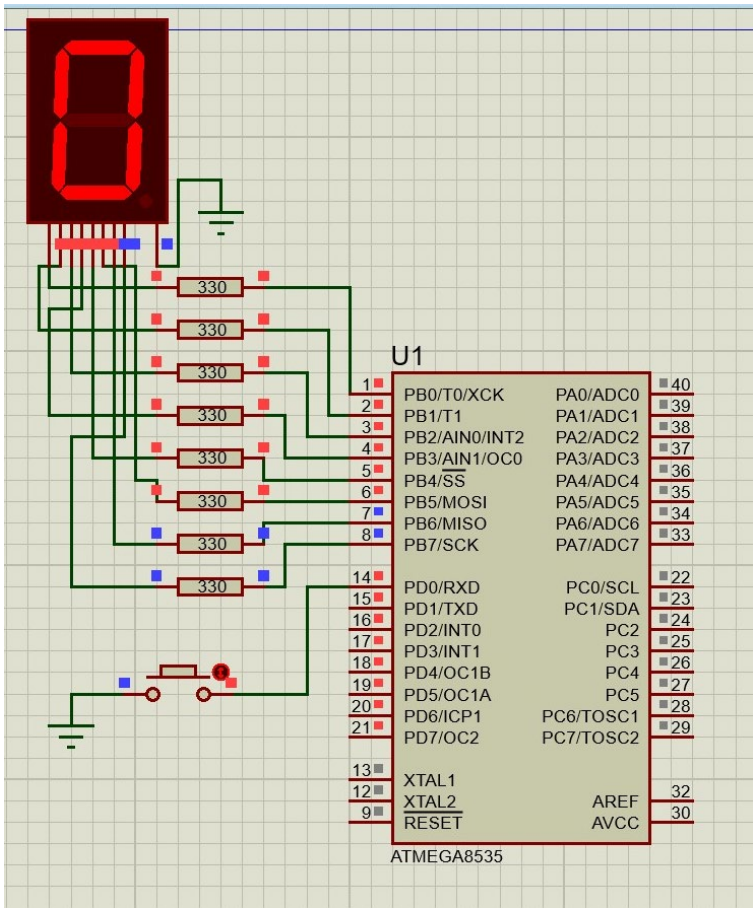


Figura. 1 Inicio de la simulación

Por ende, al realizar una pulsación el incremento no se visualiza de uno en uno, sino que adquiere un incremento aleatorio hasta cierto punto, pues dependerá del tiempo que se pulse el botón y si al momento de soltarlo se genera algún rebote, cabe mencionar que si se mantiene presionado dicho botón se verán todos los segmentos del display encendidos, pues el ojo no es capaz de percibir los cambios.

El botón que se utiliza en la simulación tiene un tiempo determinado cada que se realiza una pulsación por lo que los valores tienden a repetirse en un ciclo mostrando 0 y 5 cada que se realiza una pulsación (en ocasiones con repetición) como se muestra en la figura 2.

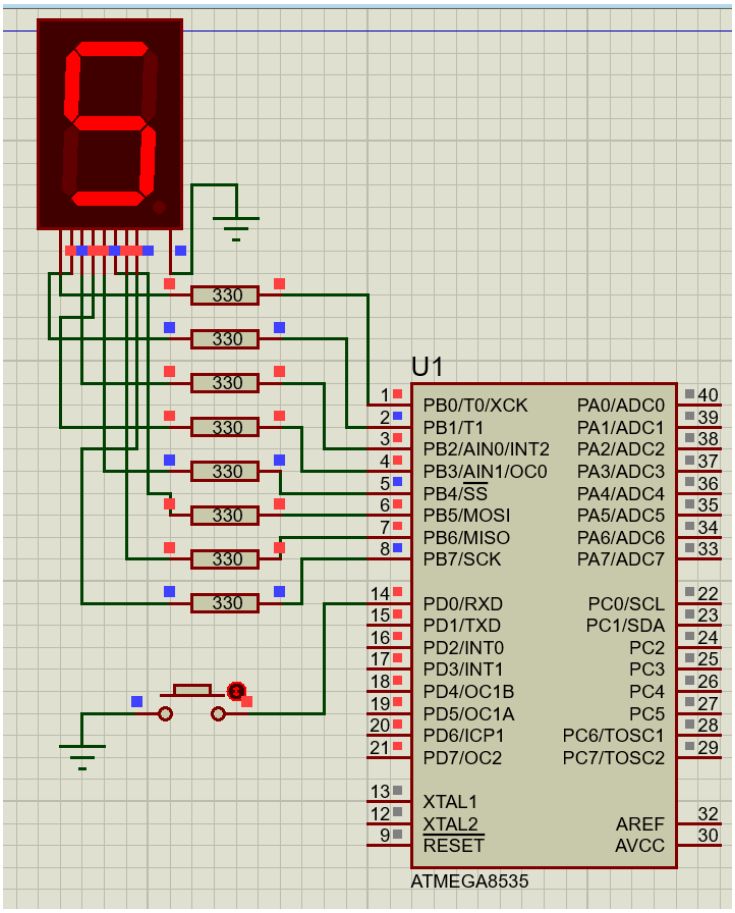


Figura. 2 Resultado tras una pulsación

Por otro lado, en la figura 3 se pueden observar todos los segmentos encendidos puesto que el botón no se ha soltado.

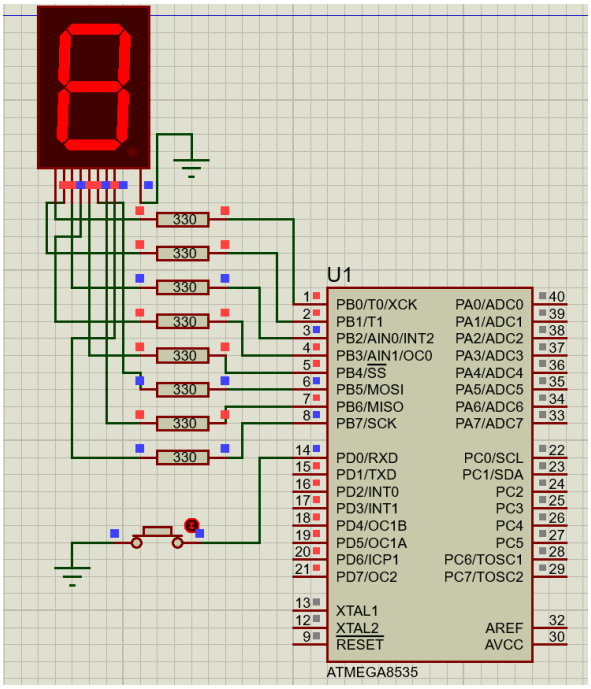


Figura. 3 Circuito con el botón sin soltar

1. Circuito armado en la Proto

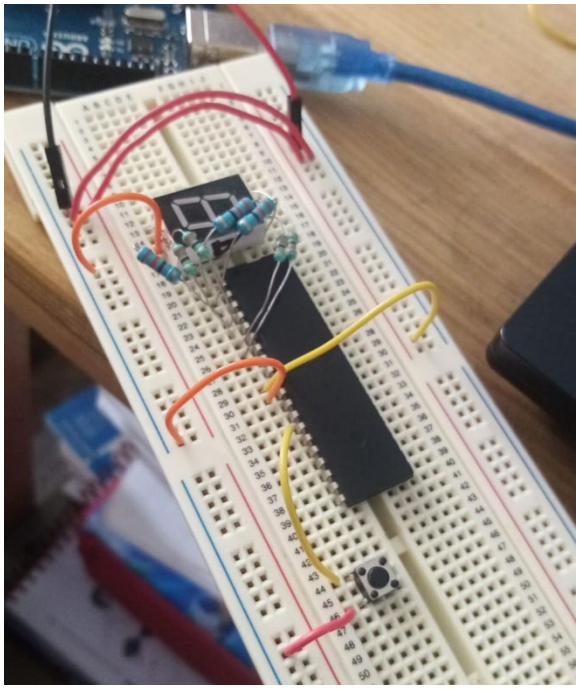


Figura. 4 Push Button

Conclusiones

La realización de la presente práctica permite observar todos los errores o interferencias que generan los componentes utilizados y los cuales se resolverán en prácticas posteriores, además de que se reutiliza parte de la implementación de la práctica anterior, en general no hay mucho que decir de está práctica pues consiste en armar un circuito con errores por lo que no requiere mucho esfuerzo.

No obstante, de los errores ya mencionados se debe comentar que la parte simulada dista en gran medida del circuito en físico, puesto que no es posible asegurar que el tiempo que se presiona el botón sea el mismo en cada caso, a diferencia del simulado que siempre se repite, por lo que se obtienen valores realmente aleatorios o bien puede no presentarse un cambio debido a alguna interferencia (ruido). Por otro lado, en la parte simulada no se presentan los rebotes, por lo que solo es posible visualizarlos al realizar pruebas en la parte física, además de que no se tiene un control sobre los incrementos ya que si el botón no se suelta estos se realizan sin que se perciba a la vista.

Arruti Sánchez Alondra

La realización de la siguiente practica nos ayudó para saber los errores que luego se encuentran en los circuitos ya que a veces nosotros como estudiantes realizamos los circuitos sin saber cómo funcionan o porque están fallando y luego no vemos que hasta el mismo circuito es lo que está diseñado para dar ese error, como nos pasó en la practica 1 de este mismo curso, pero que a la vez supimos cómo se podía arreglar de manera eficiente y que podía seguir funcionando.

El ruido en los circuitos es muy común y estos surgen por diferentes cuestiones ya que a veces hasta el mismo cable es el que genera el ruido y por lo mismo no entra bien la señal de los datos o como el mismo voltaje no entra porque lo que le da al circuito es muy poco o como vimos en esta práctica es por el botón push.

Carrillo Soto Cristian Eduardo

Bibliografía

[1] Martinez, A. J. S. (2014, 6 noviembre). Contador BCD de 0-9 con temporizador 555 (Automatización). Monografias.com. Recuperado 5 de octubre de 2022, de https://www.monografias.com/trabajos102/contador-bcd-0-9-temporizador-555-automatizacion/contador-bcd-0-9-temporizador-555-automatizacion

[2] Corazto, C. (s. f.). *Capítulo 4 temporizador contador*. Scribd. Recuperado 21 de septiembre de 2022, de https://es.scribd.com/document/208475344/Capitulo-4-temporizador-contador

[3] García, V. (2010, 13 noviembre). Pulsadores – Electrónica Práctica Aplicada. Recuperado 5 de octubre de 2022, de https://www.diarioelectronicohoy.com/blog/pulsadores-sin-rebotes