Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Practicas 05 y 06 “Contador de 0 a 9 Activado por flancos y Sin rebotes”

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 11/12/2022

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador activado por flancos de 0 a 9 mostrado en un display activado con un Push Button.

Introducción Teórica

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

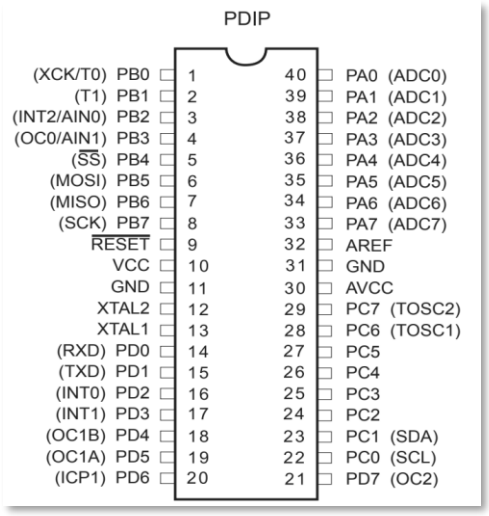

Figura 1. Configuración de pines ATmega8535

El microcontrolador utiliza una arquitectura cerrada, es decir, aquel que es inmodificable por los programadores ajenos a la compañía propietaria del código fuente. Por lo tanto, a este sistema no se le pueden colocar dispositivos periféricos, solo se usa el hardware de la compañía propietaria ya que los dispositivos ajenos a dicha compañía no son compatibles.

El microcontrolador ATMEGA8535 utiliza un encapsulado DIP-40, común en la construcción de circuitos integrados que consiste en un bloque con dos hileras paralelas de pines, observar la figura 2.

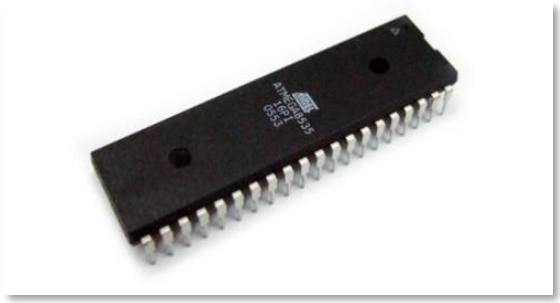

Figura 2. Microcontrolador atmega8535

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Display Cátodo y Ánodo Común

El display de 7 segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9 y o algunas letras. El dispositivo cuenta con 7 leds, uno por cada segmento, a cada uno de estos se le asigna una letra que va de la "a" a la "g" y se encuentran organizados como indica la figura 3.

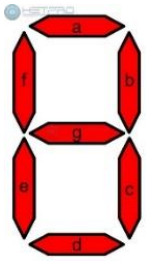

Figura 3. Disposición de los leds en un display de 7 segmentos.

Existen dos tipos de displays de 7 segmentos, de ánodo y cátodo comunes cuya diferencia se encuentra en la forma en que van conectados como lo indica la figura 4.

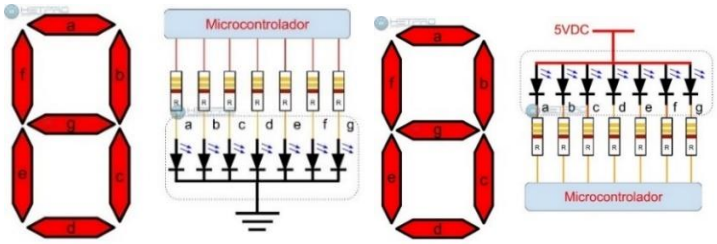

Figura 4 . Conexiones en display de 7 segmentos.

Es importante mencionar que los display de 7 segmentos, dado que están construidos con diodos LED, requieren colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED así como de la caída de voltaje.

Contador Binario

Un 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. Nuestro contador digital tiene un campo muy amplio para su aplicación de forma rotacional (cuantas vueltas efectúa un objeto) o de forma secuencial (Ej. en una empresa para el conteo de cajas de producto, etc.)

Este dispositivo permite visualizar la formación numérica de manera ascendiente desde 0 hasta 9 en un display. Consta de pocos componentes electrónicos y es alimentado por una fuente regulable que suministra 5 volts, es ampliamente utilizado a niveles más complejos para fines comerciales en artefactos electrónicos. Contiene circuitos integrados que permiten dicha función.

Señal activada por flanco

Cuando las señales se validan por un estado lógico (nivel alto o bajo) de la señal de reloj se dice que son activadas por nivel. Cuando se produce las validaciones de las señales cuando la señal de reloj cambia de estado, se dice que son activadas por flanco: flanco de subida (cambio de nivel bajo a alto) y flanco de bajada (cambio de nivel alto a bajo).

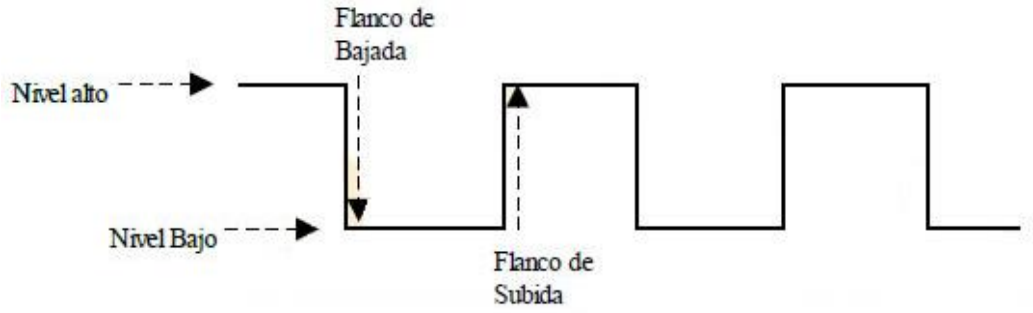

Figura 5. Representación de flancos de subida y de bajada.

Rebotes

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.

Imaginemos que pretendemos aumentar el contador de tantos de un marcador, en un punto y el pulsador no está protegido contra los rebotes, es fácil suponer que, no sería posible añadir un único tanto al mencionado marcador, con las consecuencias que acarreará tal efecto.

En la siguiente figura, se aprecia lo expuesto.

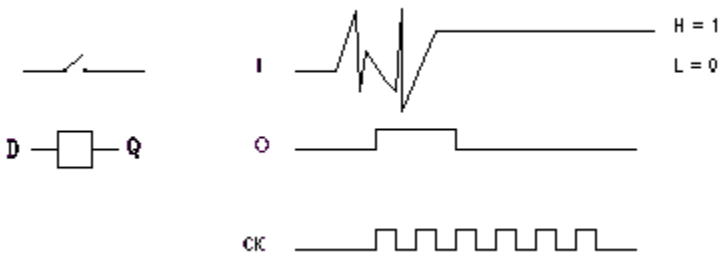

Figura 6. Representación de una señal modificada debido al rebote mecánico del botón.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 Display ánodo o cátodo común
- 7 Resistores de 330 Ω a ¼ W
- 1 Push Button

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 activado por flancos.

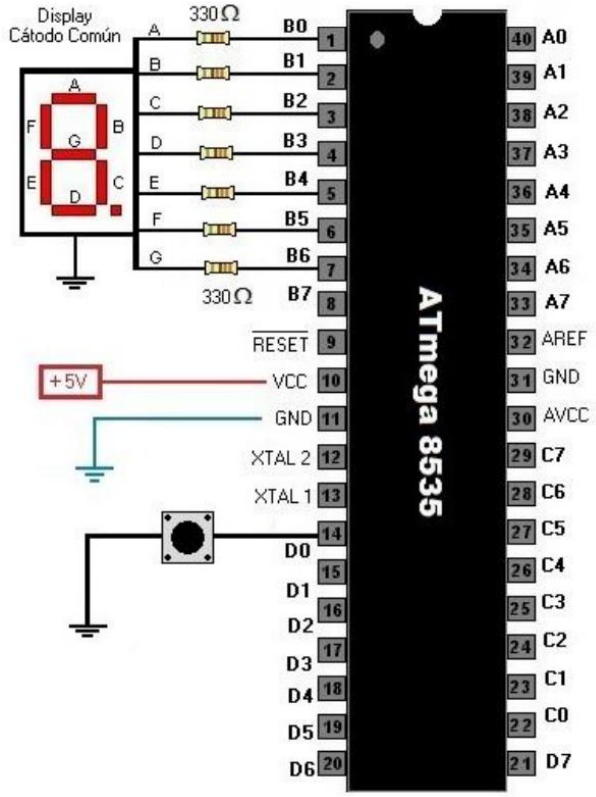

Figura 7. Circuito para el contador de 0 a 9 activado por flancos.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, correspondiente a la **práctica 5** proporcionados por el IDE de CodeVision

```
#include <mega8535.h>
#define boton PIND.0
bit botonp;
bit botona;
unsigned char variable;
const char
tabla7segmentos[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x
58,0x5e,0x79,0x71};
void main(void)
{
// Input/Output Ports initialization
// Port A initialization
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
// Port B initialization
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (1<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);
// Port C initialization
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);
// Port D initialization
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);
TCNT0=0x00;
OCR0=0x00;
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;
```

```
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);
TCNT2=0x00;
OCR2=0x00;
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
(0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);
SFIOR=(0<<ACME);
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
while (1)
      {
      if(boton == 0)
        botona = 0;
      else
        botona = 1;
      if((botona == 0) && (botonp == 1)) //Flanco de 1 a 0
        variable++;
      if(variable == 10)
        variable = 0;
      PORTB = tabla7segmentos[variable];
      PORTA = ~tabla7segmentos[variable];
      botonp = botona;
      }
}
```

Código de la configuración de los periféricos utilizados y código del programa principal en C, correspondiente a la **práctica 6** proporcionados por el IDE de CodeVision

```
#include <mega8535.h>

#include <delay.h>

#define boton PIND.0

bit botonp;

bit botona;

unsigned char variable;
```

```
 const char
tabla7segmentos[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x
58,0x5e,0x79,0x71};

 void main(void)

 {

 // Input/Output Ports initialization

 // Port A initialization

 DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);

 PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

 // Port B initialization

 DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (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<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

 // Port C initialization

 DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);

 PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

 // Port D initialization

 DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);

 PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);

 TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);

 TCNT0=0x00;

 OCR0=0x00;

 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;
 ASSR=0<<AS2;
 TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);
 TCNT2=0x00;
 OCR2=0x00;
 TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1)
| (0<<OCIE0) | (0<<TOIE0);
 MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
 MCUCSR=(0<<ISC2);
 UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);
 ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1)
| (0<<ACIS0);
 SFIOR=(0<<ACME);
 ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);
 SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1)
| (0<<SPR0);
 TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
 while (1)
       {
       if(boton == 0)
         botona = 0;
       else
         botona = 1;
       if((botonp == 1) && (botona == 0)){ //Flanco de 1 a 0
         variable++;
```

```
        if(variable == 10)
            variable = 0;
        delay_ms(40);
      }
      if((botonp == 0) && (botona == 1)) //Flanco de 0 a 1
        delay_ms(40);
      PORTB = tabla7segmentos[variable];
      PORTA = ~tabla7segmentos[variable];
      botonp = botona;
      }
}
```

Simulaciones

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

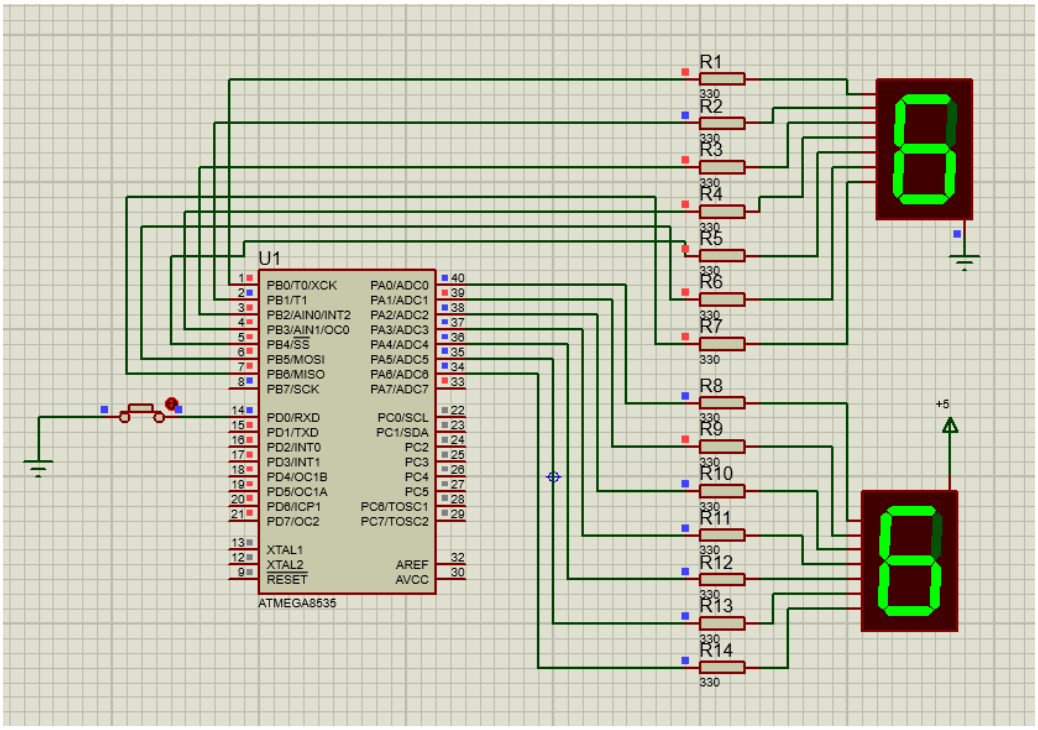

Figura 8. Simulación de las prácticas 5 y 6 (mismo circuito).

Fotografía del circuito armado

A continuación, se muestra la figura 9 la evidencia sobre la realización y prueba de la practica 5.

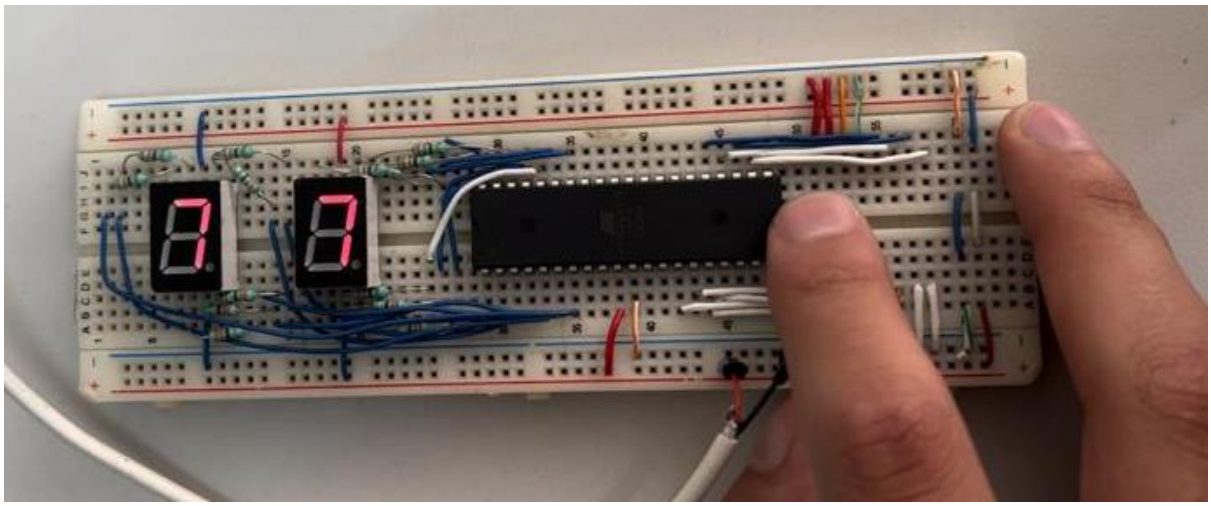

Figura 9. Circuito armado que sirvió para las prácticas 5 y 6.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

Para estas prácticas se tomó como base la práctica 4, en la cual ya teníamos implementado el contador de 0 a 9, siendo el objetivo, en esta ocasión, modificar el anterior código para hacer posible la visualización del conteo en los displays.

El cambio correspondiente a la práctica 5, fue implementar en el software una condicional que fungiera como detector de flancos, nuestra variable auxiliar estará entre el estado 0 y 1, dependiendo si está siendo presionado el push button o no, respectivamente. Por tanto, el efecto deseado se logra en el momento exacto cuando nuestro estado anterior haya sido 1 y el actual sea 0, esto es, la detección de un flanco de bajada, logrando así que el display cambie únicamente hasta que esto suceda, y por tanto se vuelve totalmente distinguible para nosotros, sin embargo, esta detección nos mostró a ligeros errores de ‘falsos flancos’ al momento de pulsar el botón en el circuito, provocando que al presionar una vez llegara a cambiar más de un número en el display.

Para evitar ese cambio no deseado, en la práctica 6, se agregó al código un retardo de 40 ms por flanco, evitando de esta forma que el software detecte un falso cambio provocado por vibraciones, ya sea, cuando se presione el botón o después de haberlo presionado y este se encuentre volviendo a su estado inicial. Logrando de esta forma que por cada botonazo solo nos hiciera un cambio de número en display. Con lo que queda aprendido que, en un sistema de alta velocidad (como este de 1 MHz) y cuyos componentes físicos sean mecánicos, es necesario aplicar correcciones mediante software para evitar errores en el funcionamiento final.

Patlani Mauricio Adriana

En las presentes prácticas, se puede analizar las diferentes maneras en las que se van mejorando el uso de los displays de la práctica 4. En estas prácticas se modificaron partes del código para solucionar algunos errores que se observan al mostrar el contador del 0 al 9

El primer punto a solucionar fue debido a que la detección de flanco por los push botón, y debido a ello se mostraban más números; para solucionarlo, se implementó un código condicional que detectara flancos. Sin embargo, surgió el problema de "falsos flancos" que se solucionaron en la práctica 6.

Para corregir los "falsos flancos", se agregó un retardo de 40ms, así para cuando se oprimiera el botón, se visualizará en el display, número por número.

Ruvalcaba Flores Martha Catalina

La practica 5 fue la corrección de la practica 4, puesto que la practica 4 tuvo como objetivo realizar un contador de 0 a 9 mostrado en un display activado con un Push Button, sin embargo, debido a la velocidad de ejecución del programa, se muestra en el display unos saltos de números, esto debido a que se ejecuta a la velocidad del microcontrolador (oscilador interno), la cual es de 1 MHz, por lo tanto, cada instrucción del código se ejecuta aproximadamente en un microsegundo. Causando que no exista un control sobre el conteo. Tales errores que se reflejaron en la práctica, en esta se solucionaron por medio de la activación de flancos y la transición completa del pulso.

El objetivo de la práctica 5 se logra exitosamente, utilizando el código y diseño del circuito proporcionado por el profesor con el fin de realizar un contador activado por flancos de 0 a 9 mostrado en un display y este siendo activado por un Push Button. Para ello se implementaron en programa una condicional para detectar el estado de 0 y 1 a través de la activación del push-botton. La detección del flanco sucedía cuando era de bajada, es decir, del estado 1 al estado 0. Solo de esta forma el display reflejaba un cambio, sin embargo, llegó a reflejarse en el display más de un numero al momento de activar el push-button, esto se debe a que necesita de un retardo por flanco. Por lo tanto, en esta práctica se destaca la aplicación de la activación por flancos la cual permite la lectura del contenido de un registro, enviar su valor a través de una lógica combinacional y escribir ese registro en el mismo ciclo de reloj, la importancia de la frecuencia de operación del microprocesador (1MHz) y los rebotes, ya que estos son falsas pulsaciones que se producen al hacer falsos contactos en el interruptor, y una forma de solucionarlo es aplicando un retardo (delay).

Sandoval Hernández Eduardo

La práctica 5 y 6 se lograron implementar sin problemas, resultaron más sencillas debido a que el circuito utilizado fue el mismo similar a con la práctica 4, logramos comprender la importancia de saber implementar los flancos de subida y de bajada como lo fue en la práctica 5, así como el sistema sin rebotes de la práctica 6 la cual corrige posibles errores asociados a problemas físicos de los botones lo cual será de mucha utilidad para las prácticas futuras debido a que seguiremos haciendo uso de botones y por tanto es necesario saber prever cualquier tipo de error asociado al uso de estos.

Bibliografía

• F. Aguilar. (2020, Octubre). 05 y 06 Práctica de flancos y antirrebotes. Introducción a los Microcontroladores. [Online]. Disponible en: https://youtu.be/TniMUjaD1BA

• Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

• HetPro (2019). Display 7 Segmentos ánodo y cátodo común: [En línea]. Disponible en https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/

• A. Solano. (2014). Contador BCD de 0-9 con temporizador 555 (Automatización): [En línea]. Disponible en: https://www.monografias.com/trabajos102/contador-bcd-0-9-temporizador-555-automatizacion/contador-bcd-0-9-temporizador-555-automatizacion

• R. Cárdenas. (2020, Octubre). Eliminador de Rebotes en contactos Eléctricos Mecánicos. [Online]. Disponible en: https://edublogcircuitosac.blogspot.com/2020/10/eliminador-de-rebotes-en-contactos.html