**INSTITUTO POLITÉCNICO NACIONAL**


# ESCUELA SUPERIOR DE CÓMPUTO.

# INTRODUCCIÓN A LOS MICROCONTROLADORES.

## REPORTE PRÁCTICA 8.

**PROFESOR: AGUILAR SANCHEZ FERNANDO**

**Grupo: 3CM6**

**Alumno: Rojas Alvarado Luis Enrique**

**Boleta: 2014010995**

**Link de vídeo explicativo: https://youtu.be/GuQbo36t4YU**

# Cronómetro de 59.9 segundos

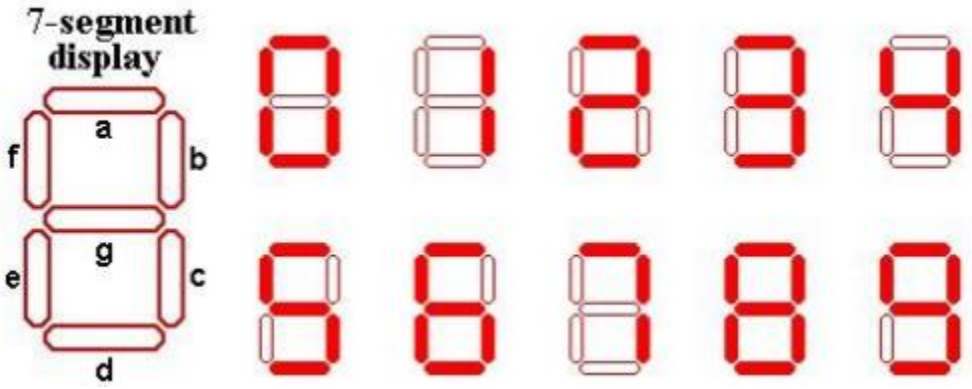
## Introducción Teórica

Display de 7 segmentos o visualizador de 7 segmentos:

Es un componente que se utiliza para la representación de números en muchos dispositivos electrónicos debido en gran medida a su simplicidad. Aunque externamente su forma difiere considerablemente de un diodo LED (diodos emisores de luz) típico, internamente están constituidos por una serie de diodos LED con unas determinadas conexiones internas, estratégicamente ubicados de tal forma que forme un número 8. A cada uno de los segmentos que forman el Display se les denomina a, b, c, d, e, f y g y están ensamblados de forma que se permita activar cada segmento por separado consiguiendo formar cualquier dígito numérico.

A continuación, se muestran algunos ejemplos: Si se activan o encienden todos los segmentos se forma el número "8". Si se activan sólo los segmentos: "a, b, c, d, e, f," se forma el número "0". Si se activan sólo los segmentos: "a, b, g, e, d," se forma el número "2". Si se activan sólo los segmentos: "b, c, f, g," se forma el número "4". En esta práctica utilizaremos 3 de ellos para que se pueda visualizar el cronómetro e indicar milésimas de segundos, unidades de segundos, décimas de segundos hasta llegar a 60.0 segundos, una vez llegado al 60.0 podemos reiniciar el conteo o terminarlo, dependiendo de lo que se quiera representar.

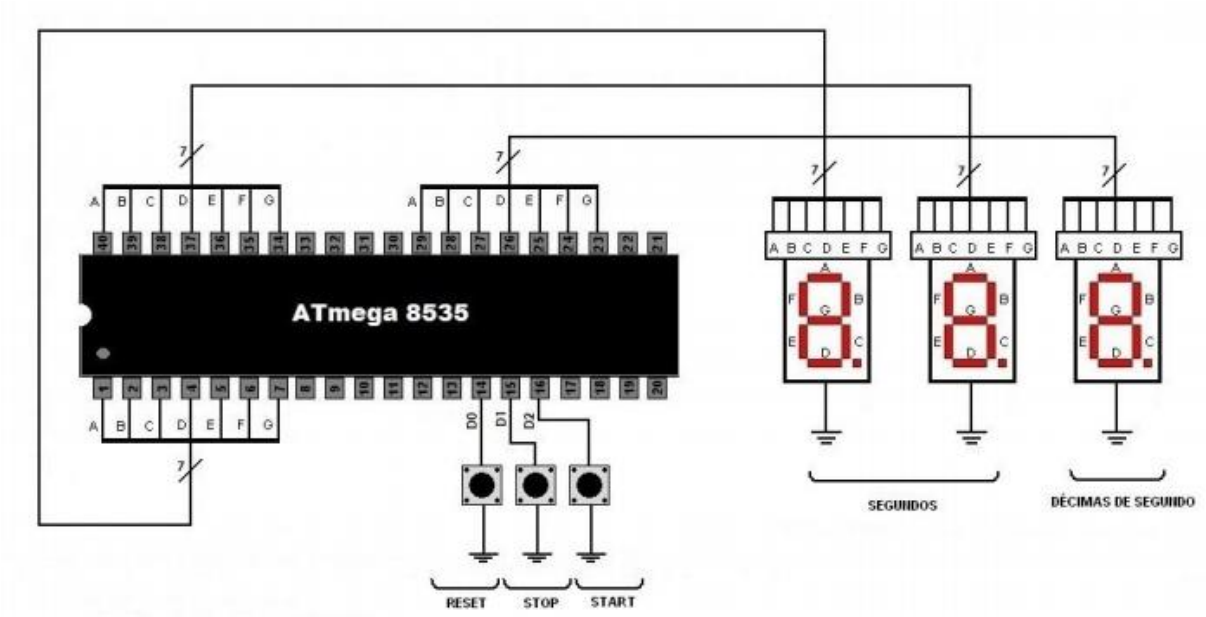
Es importante mencionar que los displays de 7 segmentos, dado que están construidos con diodos LED, requieren una corriente máxima. En otras palabras, se requiere colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED así como de la caída de voltaje. Para calcular la resistencia usamos la Ley de Ohm.




Se utilizó como base el primer contador con el display de 7 segmentos por la facilidad de uso y se modificó para usarlo con 3 displays al mismo timepo con un push button que hará una funcionalidad diferente como son las de stop, start y reset.

## Desarrollo Experimental

1.- Diseñe un programa en el que coloque dos Displays, uno en el Puerto A y el otro en el Puerto B y con una terminal del Puerto D. detecte la cuenta a través de un par infrarrojo.




## 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    : 19/11/2020
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>
#include <delay.h>
```

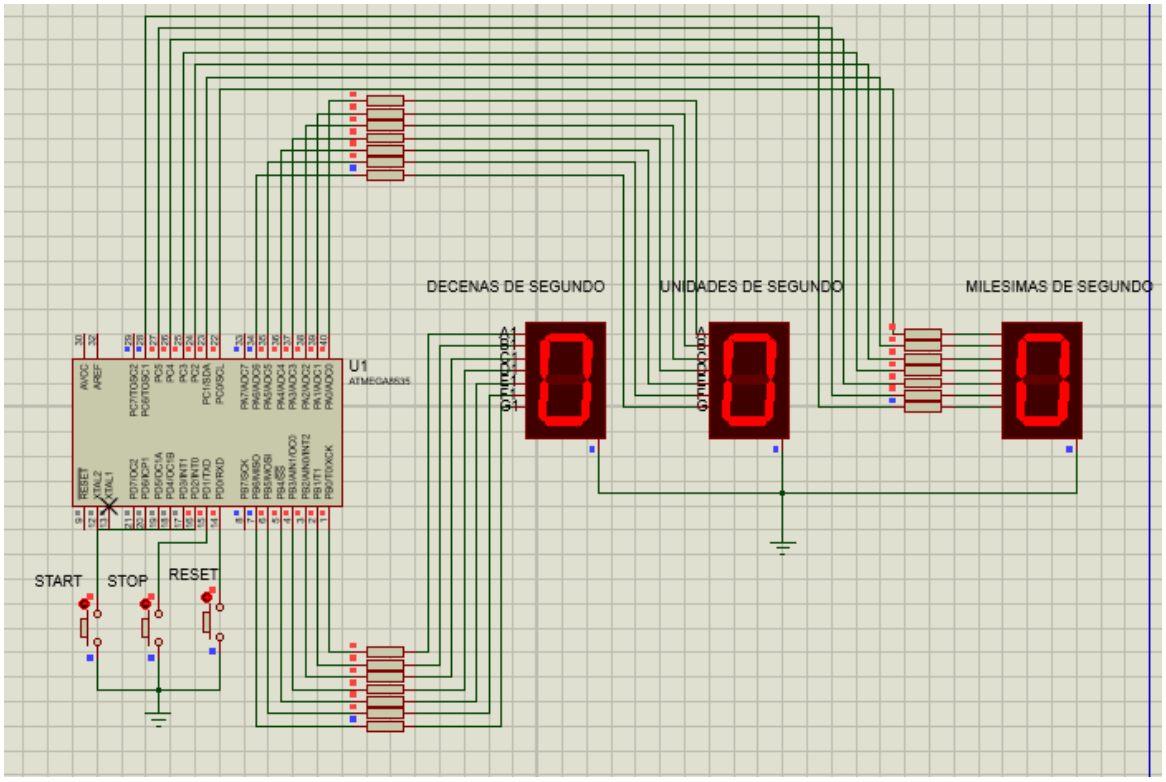
```
#define boton_reset PIND.0
#define boton_stop PIND.1
#define boton_start PIND.2
bit botonp_reset,botonp_stop,botonp_start;
bit botona_reset,botona_stop,botona_start;
bit estado=0;
unsigned char var=0,var2=0,var3=0;
const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f
};

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

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
ut
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<D
DA1) | (1<<DDA0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
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=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
ut
DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<D
DC1) | (1<<DDC0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
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=P Bit1=P Bit0=P
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (1<<PO
RTD2) | (1<<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
```

```
74. // Clock source: System Clock
75. // Clock value: Timer1 Stopped
76. // Mode: Normal top=0xFFFF
77. // OC1A output: Disconnected
78. // OC1B output: Disconnected
79. // Noise Canceler: Off
80. // Input Capture on Falling Edge
81. // Timer1 Overflow Interrupt: Off
82. // Input Capture Interrupt: Off
83. // Compare A Match Interrupt: Off
84. // Compare B Match Interrupt: Off
85. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WG
    M10);
86. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
     (0<<CS10);
87. TCNT1H=0x00;
88. TCNT1L=0x00;
89. ICR1H=0x00;
90. ICR1L=0x00;
91. OCR1AH=0x00;
92. OCR1AL=0x00;
93. OCR1BH=0x00;
94. OCR1BL=0x00;
95.
96. // Timer/Counter 2 initialization
97. // Clock source: System Clock
98. // Clock value: Timer2 Stopped
99. // Mode: Normal top=0xFF
100.        // OC2 output: Disconnected
101.        ASSR=0<<AS2;
102.        TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<
    CS21) | (0<<CS20);
103.        TCNT2=0x00;
104.        OCR2=0x00;
105.
106.        // Timer(s)/Counter(s) Interrupt(s) initialization
107.        TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
    (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
108.
109.        // External Interrupt(s) initialization
110.        // INT0: Off
111.        // INT1: Off
112.        // INT2: Off
113.        MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
114.        MCUCSR=(0<<ISC2);
115.
116.        // USART initialization
117.        // USART disabled
118.        UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<U
    CSZ2) | (0<<RXB8) | (0<<TXB8);
119.
120.        // Analog Comparator initialization
121.        // Analog Comparator: Off
122.        ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
    (0<<ACIS1) | (0<<ACIS0);
123.        SFIOR=(0<<ACME);
124.
125.        // ADC initialization
126.        // ADC disabled
127.        ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<AD
    PS2) | (0<<ADPS1) | (0<<ADPS0);
```

```
128.
129.       // SPI initialization
130.       // SPI disabled
131.       SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA)
    | (0<<SPR1) | (0<<SPR0);
132.
133.       // TWI initialization
134.       // TWI disabled
135.       TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
136.
137.       while (1)
138.             {
139.             //BOTON RESET
140.             if (boton_reset==0){
141.               botona_reset=0;
142.             }else{
143.               botona_reset=1;
144.             }
145.             //BOTON STOP
146.             if (boton_stop==0){
147.               botona_stop=0;
148.             }else{
149.               botona_stop=1;
150.             }
151.             //BOTON START
152.             if (boton_start==0){
153.               botona_start=0;
154.             }else{
155.               botona_start=1;
156.             }
157.
158.             //BOTON RESET
159.             if ((botonp_reset==1)&&(botona_reset==0)){ //hubo cambio de flanco d
    e 1 a 0
160.               var=0;
161.               var2=0;
162.               var3=0;
163.               estado=0;
164.             delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
165.             }
166.             if ((botonp_reset==0)&&(botona_reset==1)) //hubo cambio de flanco de
     0 a 1
167.               delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
168.             //BOTON STOP
169.             if ((botonp_stop==1)&&(botona_stop==0)){ //hubo cambio de flanco de
    1 a 0
170.               estado=0;
171.             delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
172.             }
173.             if ((botonp_stop==0)&&(botona_stop==1)) //hubo cambio de flanco de 0
     a 1
174.               delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
175.             //BOTON START
176.             if ((botonp_start==1)&&(botona_start==0)){ //hubo cambio de flanco d
    e 1 a 0
177.               estado=1;
178.             delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
179.             }
180.             if ((botonp_start==0)&&(botona_start==1)) //hubo cambio de flanco de
     0 a 1
181.               delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
```

```
182.
183.             //CRONOMETRO
184.             if(estado==1){
185.               var3++;
186.
187.               if(var3==10){
188.                   var3=0;
189.                   var2++;
190.               }
191.               if(var2==10){
192.                   var2=0;
193.                   var++;
194.               }
195.               if(var==6){
196.                   estado=0;
197.                   delay_ms(1000);
198.                   var=0;
199.                   var2=0;
200.                   var3=0;
201.               }
202.
203.             delay_ms(100);
204.
205.             }
206.
207.             PORTC = tabla7segmentos[var3];
208.             PORTA = tabla7segmentos[var2];
209.             PORTB = tabla7segmentos[var];
210.
211.             botonp_reset = botona_reset;
212.             botonp_stop  = botona_stop;
213.             botonp_start = botona_start;
214.
215.             }
216.       }
```

## Circuito en Proteus:



# Conclusiones

En la realización de esta práctica adquirimos la habilidad de realizar un cronómetro de 59.9 segundos. Fue una práctica interesante, aplicamos los conocimientos adquiridos previamente con los delay y los displays ya que no deja de ser un contador y queremos que se detengan en cierto número, podemos delimitarlo y con la programación en C es muy sencillo hacer la funcionalidad que pide la práctica con los botones de reset, stop y start, ya que cada uno puede ser una variable distinta y así trabajarlos por separado.