



**INSTITUTO POLITÉCNICO NACIONAL**

**“ESCOM”**

**INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**REPORTE 6: CONTADOR UTILIZANDO INTERRUPCIONES Y SU  
RESULTADO APAREZCA EN UN LCD**

**ESQUIVEL PÉREZ JONATHAN ALFREDO**

**HERNÁNDEZ LÓPEZ ÁNGEL ZAIT**

**SALGADO GALLEGOS JESUS**

**SANCHEZ PIZANO IRVING DANIEL**

**INTRODUCCIÓN A LOS MICROCONTROLADORES**

**PEREZ PEREZ JOSE JUAN**

**3CM8**

**28/JUNIO/2020**

## OBJETIVO. –

El objetivo de esta práctica es saber usar el timer de 0 al 99, utilizando interruptores y a su vez habilitando un contador. Esto se verá reflejado en un LCD.

## INTRODUCCIÓN. –

### CONTADOR. -

Un contador es un circuito en el que sus salidas siguen una secuencia fija que cuando acaba vuelve a empezar, o circuitos que reciben sus datos en forma serial ordenados en distintos intervalos de tiempo.

Los pulsos de entrada pueden ser pulsos de reloj u originarse en una fuente externa y pueden ocurrir a intervalos de tiempo fijos o aleatorios.

### INTERRUPCIONES. -

Las interrupciones es una de las características de los microcontroladores, de las más importantes que constituye la capacidad de sincronizar la ejecución de programas con acontecimientos externos; es decir, cuando se produce una interrupción, el micro automáticamente deja lo que esto haciendo, va a la dirección 04h de programa y ejecuta lo que encuentre a partir de allí hasta encontrarse con la instrucción RETFIE que le hará abandonar la interrupción y volver al lugar donde se encontraba antes de producirse dicha interrupción

## MATERIAL Y EQUIPO UTILIZADO

- Protoboard
- push button
- resistencias
- 2 LCD
- microcontrolador ATmega 8535
- software proteus

## QUE SE REALIZARA EN LA PRACTICA

Se realizara un contador del 0 al 99 que cuente con las siguientes interrupciones.

- Int0 controla el contador derecho.
- Int1 controla el contador izquierdo.
- Int2 controla ambos contadores
- Reset pone en ceros ambos contadores.

Los contadores se podrán interrumpir por separado o bien ambos a la misma vez.

# DESARROLLO DE LA PRACTICA INCLUYENDO DIAGRAMAS DE FLUJO, CODIFICACION COMENTADA

CODIFICACION:

```
1. .include "m8535def.inc"
2.     .def aux = r16
3.     .def col = r17
4.
5. .macro ldb      ; Macro lbd
6.     ; @0 registro bajo
7.     ; @1 valor a guardar
8.     ldi aux,@1 ; Se carga el valor ingresado @1 en aux
9.     mov @0,aux ; Se mueve el valor de aux en el registro @0
10. .endm
11.
12. .macro mensaje ; Macro mensaje
13.     ; @0,@1,@2,@3,@4 valores hexadecimales o enteros
14.     ; Se llaman a la macro lbd para guardar los valores
15.     ; en un registro en especial
16.     ldb r4,@0
17.     ldb r3,@1
18.     ldb r2,@2
19.     ldb r1,@3
20.     ldb r0,@4
21. .endm
22.
23. .macro deco     ; Macro deco
24.     ; @0 Es un registro
25.     push zh      ; Se agrega zh en el pila
26.     push zl      ; Se agrega zl en la pila
27.     ldi ZH, high(tabla<<1) ; Se inicializa apuntador ZH
28.     ldi ZL, low(tabla<<1) ; Se inicializa apuntador ZL
29.     add zl,@0    ; suma a ZL el registro entrante
30.     lpm aux,Z    ; Se carga contante del programa
31.     mov @0,aux   ; Se mueve el valor de aux al registro entrante
32.     pop zl       ; Se saca de la pila ZL
33.     pop zh       ; Se saca de la pila ZH
34. .endm
35.
36. .macro inidet   ; Macro ident
37.     ;@0 valor hexadecimal
38.     push aux     ; Se agrega aux a la pila
39.     push col     ; Se agrega col en la pila
40.     in aux,timsk ; aux contendrá el valor del registro timsk
41.     ldi col,@0   ; Se carga el valor entrante a col
42.     eor aux,col  ; Se aplica la or exclusiva en aux y col y se guarda en aux
43.     out timsk,aux ; pone el valor de aux en el registro timsk
44.     pop col      ; Se saca col de la pila
45.     pop aux      ; Se saca aux de la pila
46. .endm
47.
48. reset:
49.     rjmp main ; vector de reset
50.     rjmp stst0;vector INT0
51.     rjmp stst1;vector INT1
52.     .org OVF2addr;$004
53.     rjmp cuenta1; vector timer2
```

```

54.     .org OVf1addr;$008
55.     rjmp cuenta0;vector timer1
56.     rjmp barre;vector timer0
57.     .org INT2addr;$012
58.     rjmp stst2;vector INT2
59.
60. main:
61.     ldi aux,low(ramend)
62.     out spl,aux
63.     ldi aux,high(ramend)
64.     out sph,aux      ; Se inicializa la pila del programa
65.     rcall config_io  ; Se llama a config_io
66.     rcall texto0     ; Se llama a texto0
67.     clr zh           ; Se limpia el apuntador zh
68.     clr zl           ; Se limpia el apuntador zl
69.     ldi col,1        ; Se carga a col en 1
70.     out portc,col    ; Se muestra col en el puerto C
71.     ld aux,z         ; Se carga a aux el valor de Z
72.     out porta,aux    ; Se muestra en el puerto A el valor de aux
73.
74. uno:nop             ; No hay operacion, un ciclo de reloj menos
75.     nop             ; No hay operacion, un ciclo de reloj menos
76.     rjmp uno        ; Salta a la uno
77.
78. config_io:
79.     ser aux          ; Se le signa el valor FF
80.     out ddra,aux     ; Se inicia como puerto de salida A
81.     out portb,aux    ; Se inicia como puerto de entrada B
82.     out ddrc,aux     ; Se inicia como puerto de salida C
83.     out portd,aux    ; Se inicia como puerto de entrada D
84.     ldi aux,1        ; Se carga uno en aux
85.     out tccr0,aux    ; Se selecciona el reloj sin preescala
86.     ldi aux,2        ; Se carga dos en aux
87.     out tccr1b,aux   ; Se selecciona el reloj/128 de preescala de contador uno
88.     ldi aux,$01; 0000 0001
89.     out tmsk,aux; toie0
90.     ldi aux,5        ; Se carga aux con 5
91.     out tccr2,aux    ; Se selecciona el reloj/128 de preescalade 8 bits
92.     ldi aux,$0a; 0000 1010
93.     out mcucr,aux    ; Se activan las interrupciones INT1
94.     ldi aux,$e0; 1110 0000
95.     out gicr,aux     ; Se habilitan las interrupciones
96.     sei              ; pone vantera i en uno
97.     ldi r22,16       ; Se carga el valor 16 al registro
98.     ret              ; Regresa a donde fue llamada
99.
100. texto0:
101.     mensaje 0,0,$40,0,0 ; Se llama a la macro mensaje
102.     ; Mueve el valor de los registros bajos a uno alto
103.     mov r18,r0
104.     mov r19,r1
105.     mov r20,r3
106.     mov r21,r4
107.     rcall conv      ; Llama a conv
108.     ret
109.
110. cuenta0:            ; Cuando se desborde el timer1
111.     inc r18          ; Incrementa en uno el registro
112.     cpi r18,10       ; Compara si el registro es igual a diez
113.     brne sal0        ; Si el registro no es igual a diez, salta a sal0

```

```

114.      clr r18      ; Se limpia el registro
115.      inc r19      ; Incrementa en uno el registro
116.      cpi r19,10    ; Compara si el registro es igual a 10
117.      brne sal0     ; Si no es igual a 10, salta a sal0
118.      clr r19      ; Se limpia el registro
119.
120.  sal0:           ; Si el registro no es igual a 10, entonces
121.      mov r0,r18    ; Mueve el valor de r18 a r0
122.      mov r1,r19    ; Mueve el valor de r16 a r1
123.      deco r0      ; Llama a la macro deco
124.      deco r1      ; Llama a la macro deco
125.      reti         ; Regresa a donde fue llamado
126.
127.  cuenta1:        ; Cuando se desborde el timer0
128.      dec r22      ; Decrementa en uno el registro
129.      brne otro    ; Si la bandera Z es igual a 0, salta a otro
130.      ldi r22,16   ; Se carga el valor 16 al registro
131.      inc r20      ; Se incrementa en uno el registro
132.      cpi r20,10    ; Compara si r20 es igual a 10
133.      brne sal1     ; Si no son iguales, salta a sal1
134.      clr r20      ; Se limpia el registro
135.      inc r21      ; Se incrementa en uno el registro
136.      cpi r21,10    ; Compara si el registro es igual a 10
137.      brne sal1     ; Si no son iguales, entonces salta a sal1
138.      clr r21      ; Se limpia el registro r21
139.
140.  sal1:           ; Si el registro es igual a 10
141.      mov r3,r20    ; Se mueve el r20 a r3
142.      mov r4,r21    ; Se mueve el r21 a r4
143.      deco r3      ; Llama a la macro deco
144.      deco r4      ; Llama a la macro deco
145.  otro:
146.      reti         ; Regresa a donde fue llamado
147.
148.  conv:
149.      ; Para los registros r0 a r4 se decodifica para poderlos postrar en los display
150.      deco r0
151.      deco r1
152.      deco r3
153.      deco r4
154.      ret
155.
156.  barre:          ; Cuando se desborde con counter0
157.      out porta,zh  ; Se carga el valor de ZH al puerto A
158.      inc z1        ; Se incrementa en uno ZL
159.      lsl col       ; Recorre un bit a la izquierda del registro
160.      cpi col,$20    ; Compara si col es igual a 20 hexadecimal
161.      brne dos      ; Si no es igual entonces salta a dos (z = 0)
162.      ldi col,1      ; Carga en el registro el valor de uno
163.      clr z1        ; Se limpia ZL
164.
165.  dos:            ; Si el registro es igual a 32 decimal
166.      out portc,col  ; Muestra en el puerto C el valor de col
167.      ld aux,z       ; Carga en aux el valor del registro Z
168.      out porta,aux  ; Muestra en el puerto A el valor de aux
169.      reti         ; Regresa a donde fue llamado
170.
171.  stst0:
172.      inidet $04     ; Se llama a la macro inidet
173.      reti

```

```

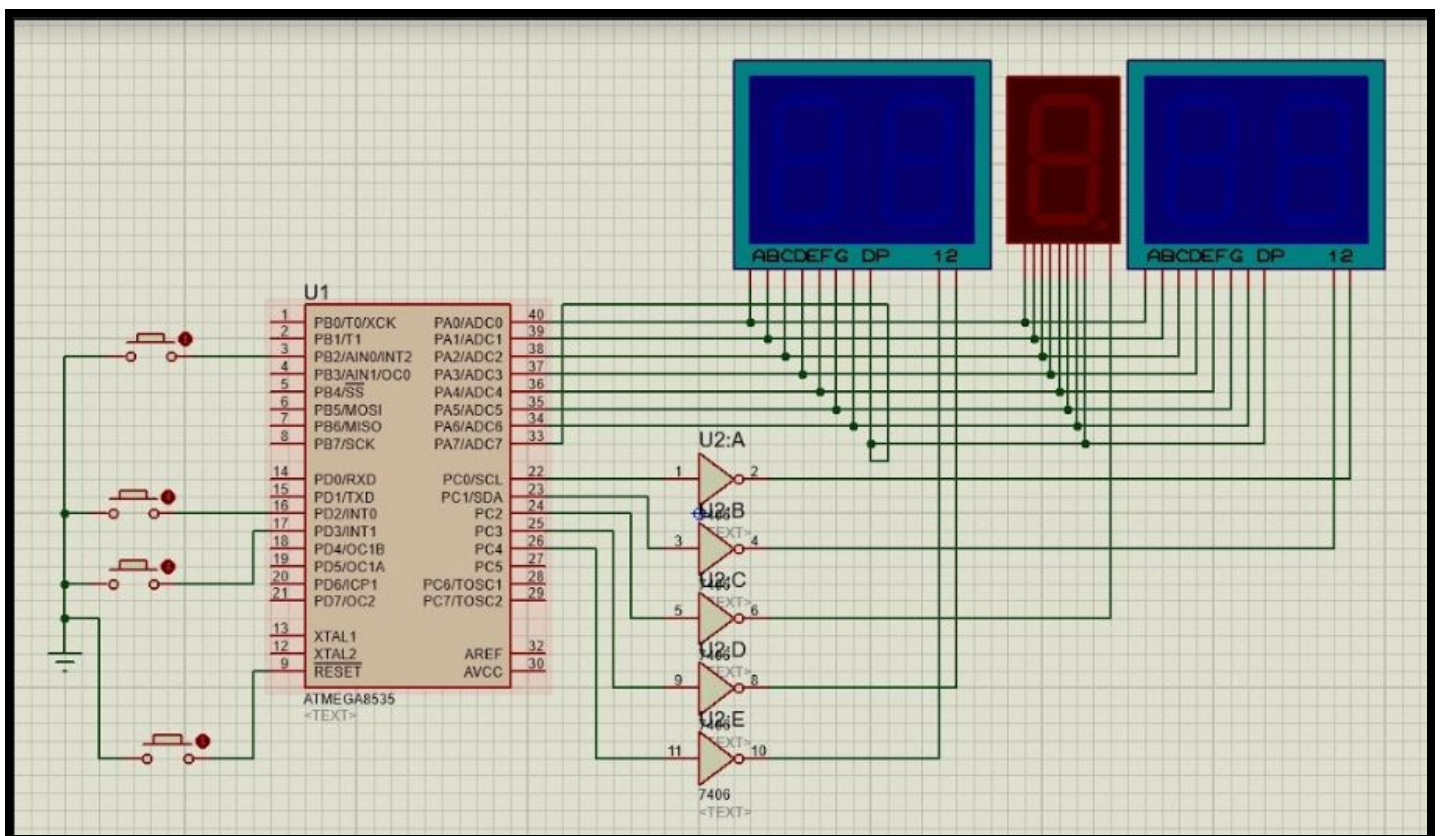
174.
175. stst1:
176.     inidet $40      ; Se llama a la macro inidet
177.     reti
178.
179. stst2:
180.     inidet $44      ; Se llama a la macro inidet
181.     reti
182.
183. tabla:
184.     .db $3f,$06,$5b,$4f,$66,$6d,$7d,$27,$7f,$6f

```

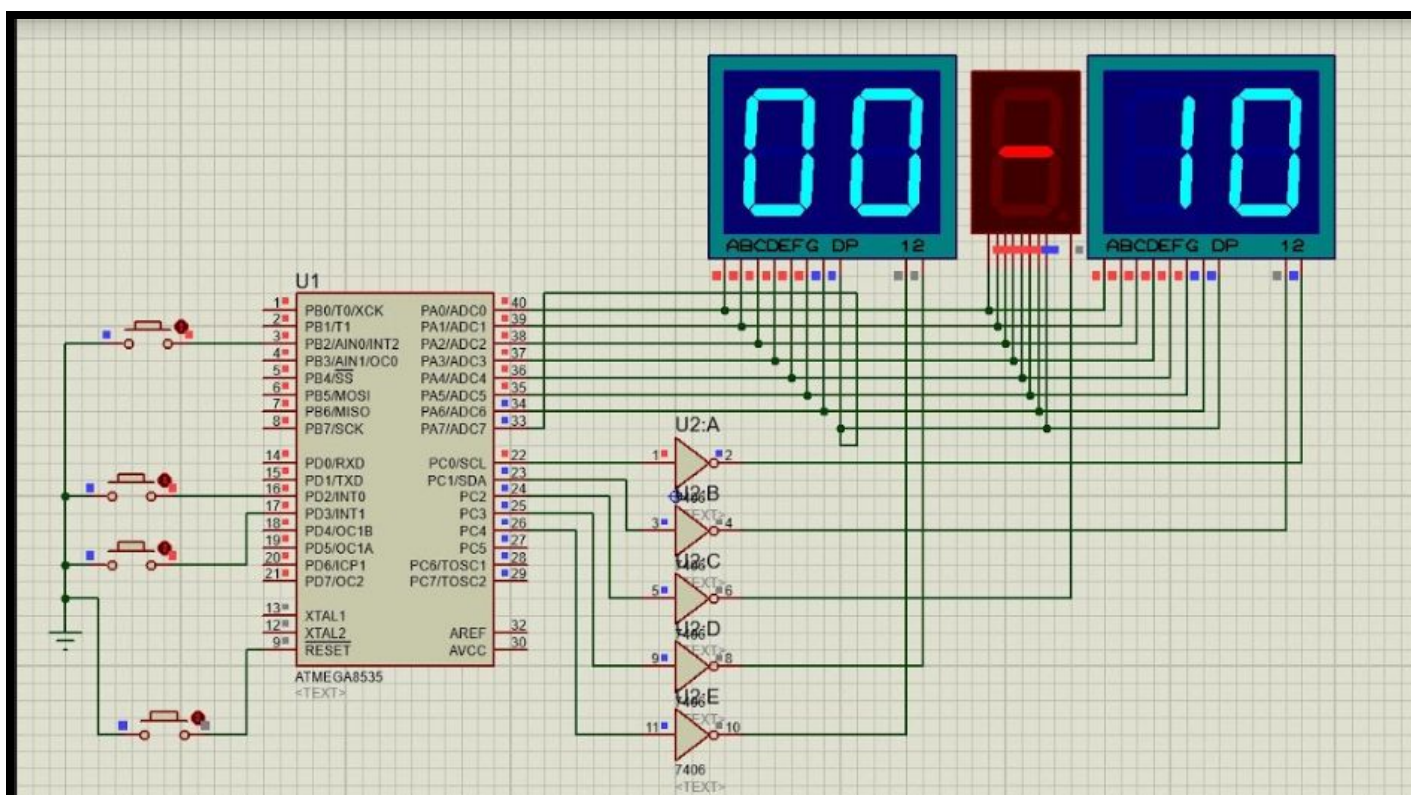
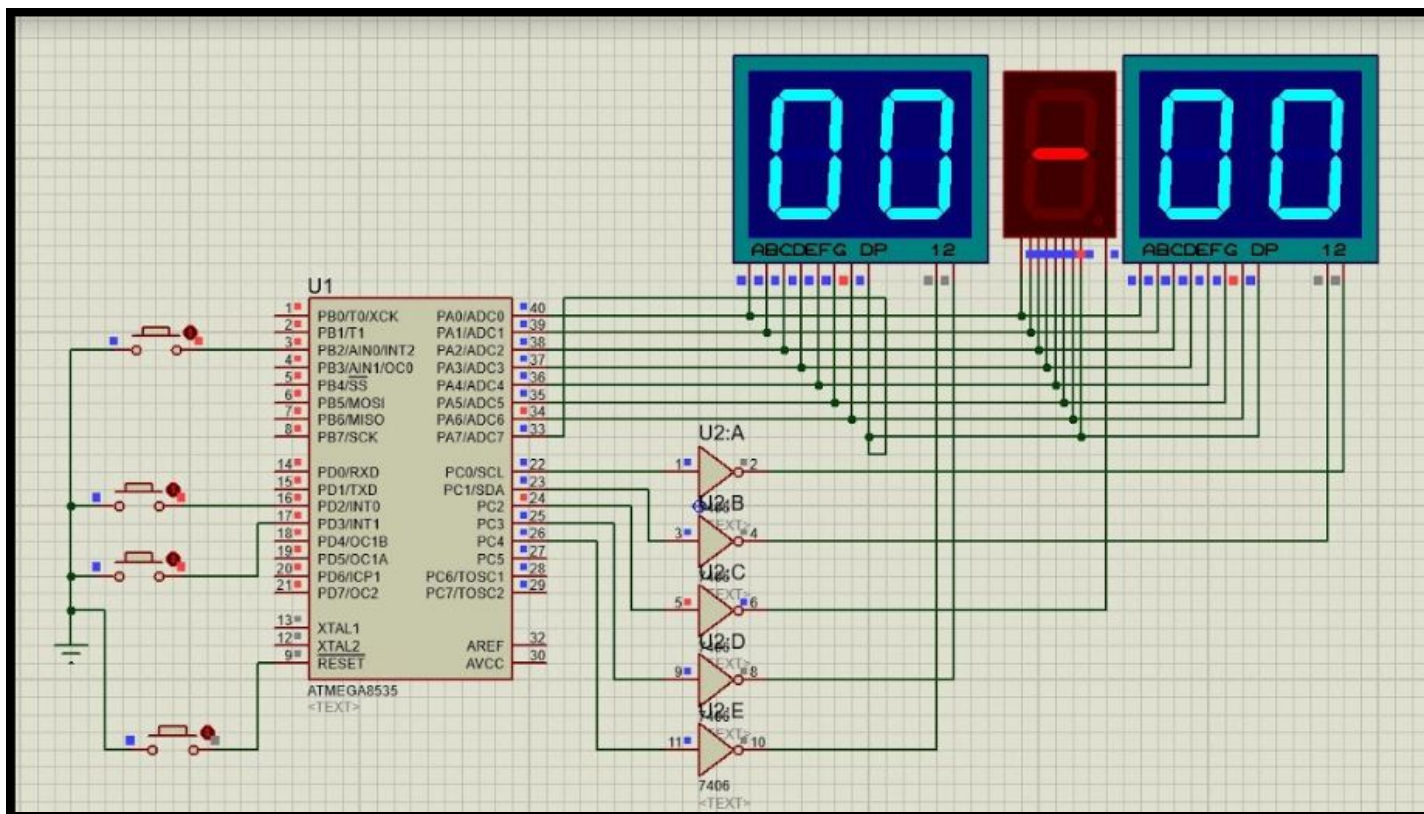
SIMULACION:

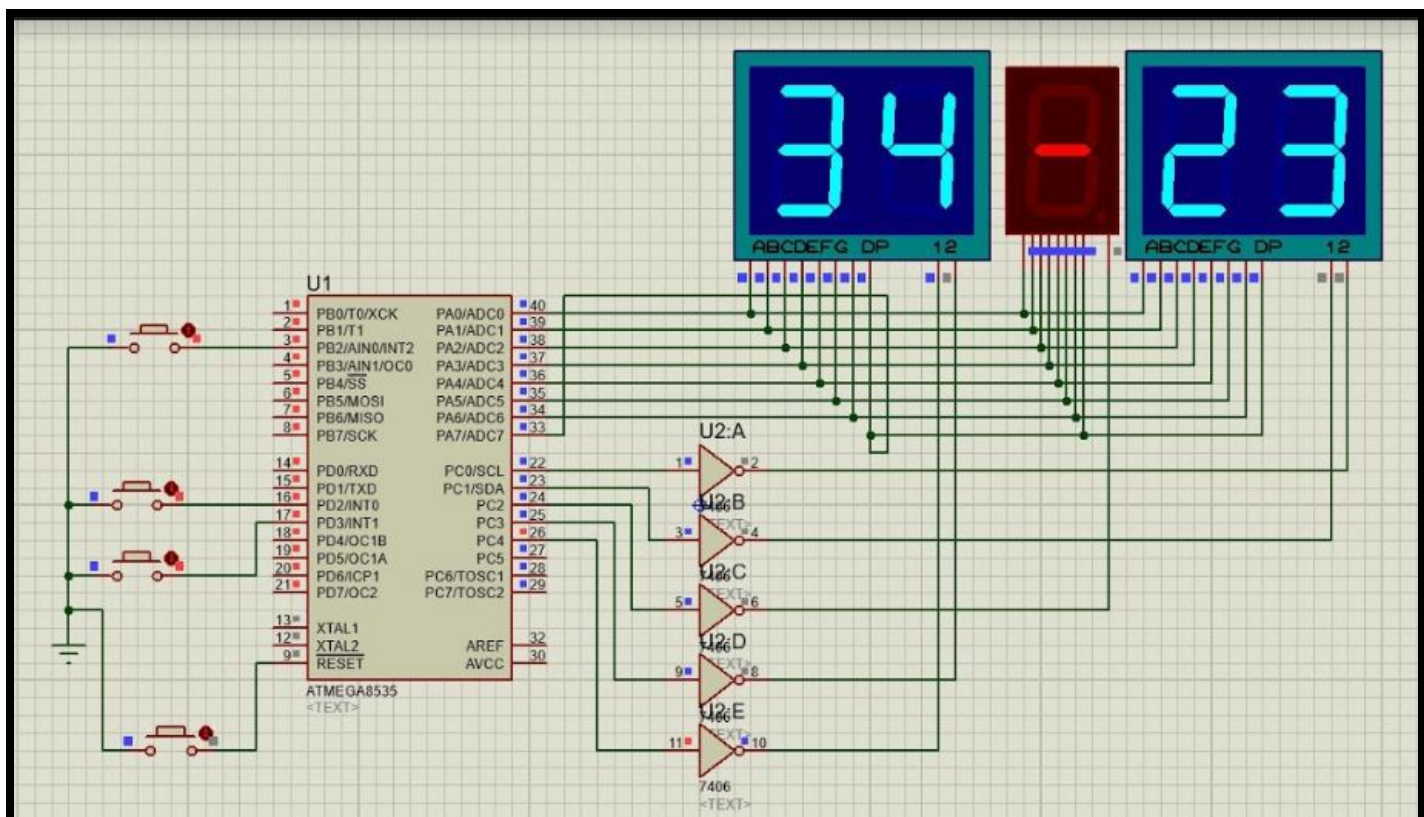
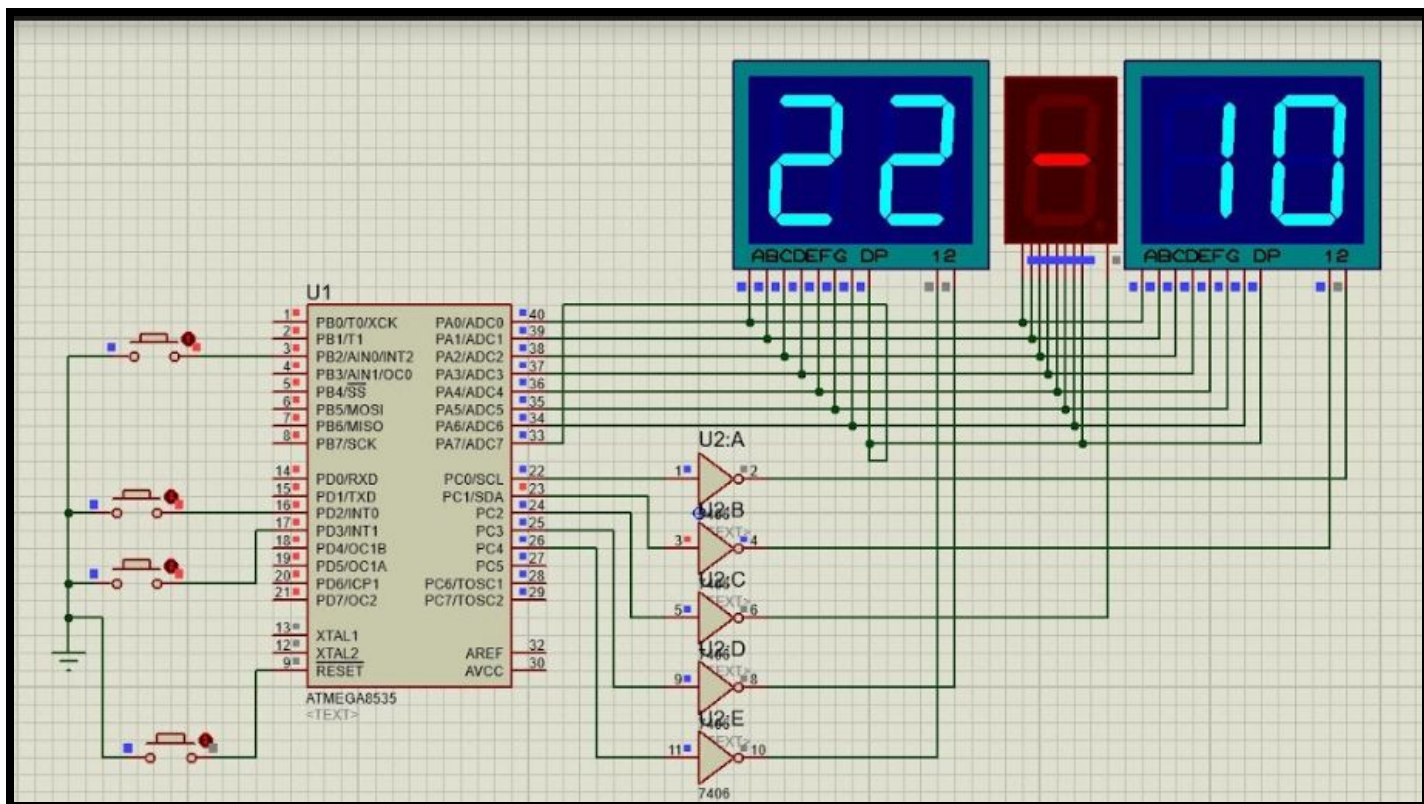
El video se encuentra en:

<https://drive.google.com/file/d/1ikxfYtBwBeJQO8TqiQoOcA0wsbaOJo9v/view?usp=sharing>











## CONCLUSIONES Y OBSERVACIONES INDIVIDUALES

### ❖ **Esquivel Pérez Jonathan Alfredo**

En la presente práctica pudimos implementar nuevamente los conocimientos aplicados en la práctica anterior debido a que solamente implementamos un multiplexor para visualizar la salida del dato correspondiente. Asimismo el control de timers ayudó a hacer un mejor control sobre los contadores del proyecto

### ❖ **Hernández López Ángel Zait**

En esta práctica se pudo apreciar el controlar los diferentes tipos de timers que contiene el microcontrolador. ya que este tiene dos contadores distintos, además de un tercero que se encarga de hacer un tipo de multiplexor para elegir la salida del dato correspondiente y así ver el distinto dato que corresponde a cada uno.

### ❖ **Salgado Gallegos Jesús**

Como se ve en la práctica se implementó y aumentó el uso de las interrupciones para el contador, teniendo instrucciones que modifiquen una o ambas salidas, esto que se necesita de un multiplexor para generar las salidas correspondientes que se quieren mostrar, además de agregar operaciones de incremento y decremento.

### ❖ **Sanchez Pizano Irving Daniel**

Esta práctica fue un tanto de repaso ya que ya habíamos utilizado en otras prácticas los contadores y las interrupciones. Y en esta ocasión fue necesario juntar los conocimientos adquiridos en prácticas anteriores. Lo único que se me sigue complicando un poco es el tema de las interrupciones.