INSTITUTO POLITÉCNICO NACIONAL

ESCOM

# Instituto Politécnico Nacional

## Introducción a los Microcontroladores


Alumnos:

Contreras Cardoso Adolfo

Martínez Alvarado Bryan Alexis

Maya Martínez Alonso Rubén

Pérez Gómez Santiago

Profesor:

Ing. Pérez Pérez José Juan


3CM15

IPN

# Planteamiento del Problema

Escribe un programa para el circuito dado en proteus para que realice lo siguiente:

Al inicio deberá mostrarse “- - - - - - - - “en los 8 displays, al pulsar algún botón, deberá mostrarse la información correspondiente en el display más a la derecha, de la siguiente forma en el caso de presionar los botones en el siguiente orden: “2,6,8,9,- “.

“ - - - - - - - -“

“- - - - - - - 2“

“- - - - - - 2 6“

“- - - - - 2 6 8“

“- - - - 2 6 8 9“

“- - - 2 6 8 9 - “

El programa puede ser .asm ó .C

# Desarrollo

Para la solución al problema planteado, se utilizó como base algunos de los programas realizados en trabajos anteriores, empezando por el manejo de la EEPROM y también el contador de anillo requerido para visualizar valores distintos en los displays.

Básicamente se almacena en las primeras 8 direcciones de la EEPROM los guiones a mostrar al inicio, posteriormente, se modifican esos valores de acuerdo con cada acción del usuario, si presiona una tecla, se recorren a la izquierda y se almacena el nuevo valor, si utiliza la función de borrado, se recorren a la derecha y se almacena un guión.

Cabe mencionar que, cuando se presiona el teclado es necesario detener el contador de anillo hasta que deje de presionarse la tecla, esto para que no se guarde en la EEPROM un valor muchas veces, sino sólo una.

El diagrama de flujo correspondiente a la solución propuesta es el siguiente:

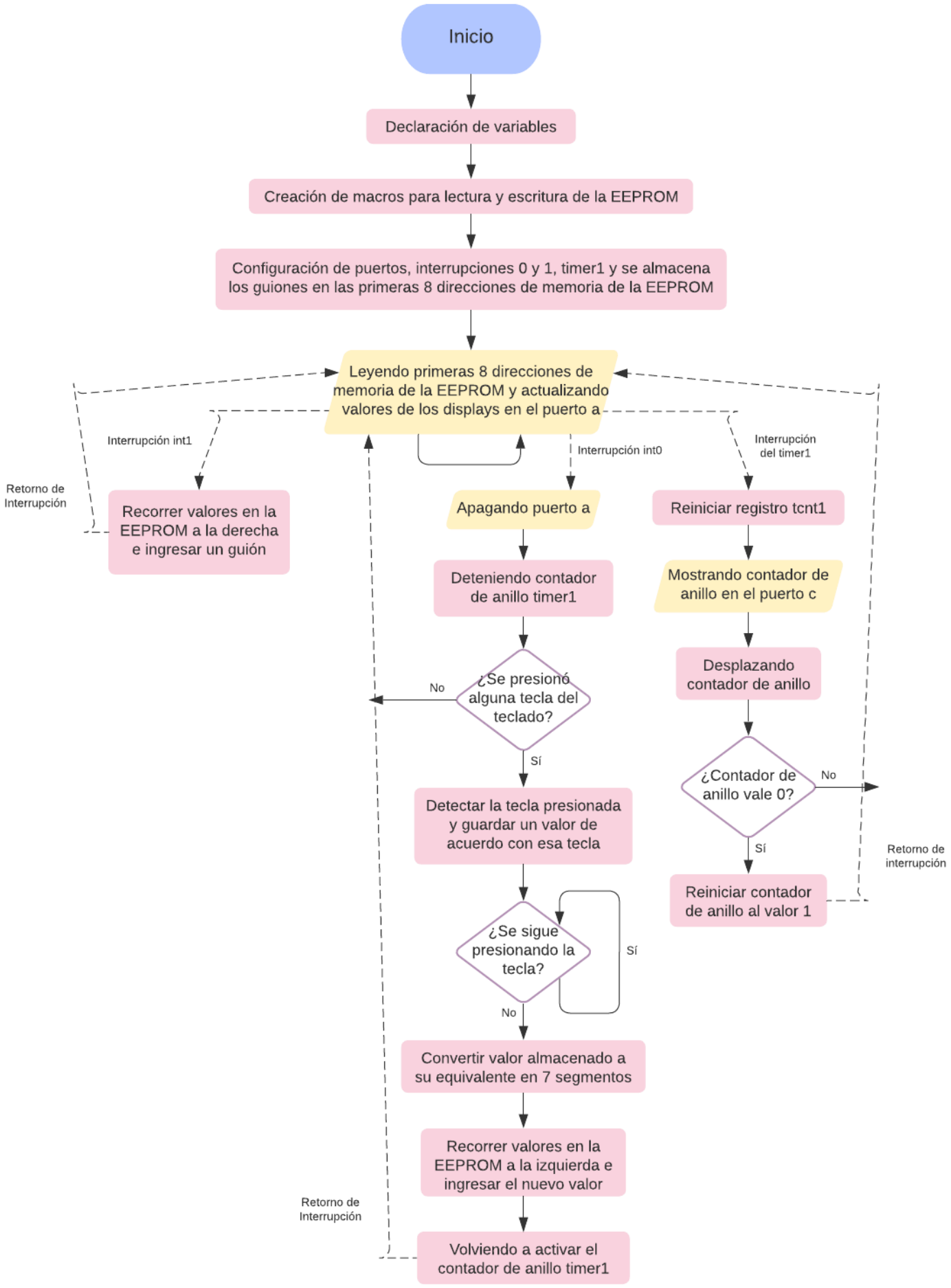



**Figura 1.** Diagrama de Flujo

# Implementación

El código en ensamblador correspondiente al diagrama de flujo es el siguiente:

```
;codigo para detectar teclado matricial y mostrar la tecla
;pulsada en displays 7 segmentos, con una funcion de borrado

;importando libreria inicial
.include "m8535def.inc"

;variables de apoyo
.def aux = r16
.def anillo = r17

;para displays
.def display = r18
.def flag = r19

;para eeprom
.def dirh = r20
.def dirl = r21

;macro para reducir codigo de lectura y escritura eeprom, se
;espera a que eeprom este lista y despues se carga el valor
.macro eeprom_memory

    ;guardando direccion para cargar datos en la eeprom
    ldi dirh,@0
    ldi dirl,@1

    ;ciclando hasta que eeprom este lista
    rcall eeprom_loop
```

**Figura 2.** Primera Sección de Código

```
    ;EEAR = registro de direccion
    ;direccion de eeprom a leer o escribir
    out EEARH,dirh
    out EEARL,dirl

.endm

;para escritura en la eeprom
.macro eeprom_write

    ;cargando direccion de memoria
    eeprom_memory @0,@1

    ;EEDR = registro de datos
    ;para guardar o recibir datos de lectura o escritura
    out EEDR,@2

    ;EECR = registro de control para lectura o escritura
    ;EEMWE = indica que se habilite la escritura
    sbi EECR,EEMWE

    ;EEWE = indica que se inicie la escritura
    sbi EECR,EEWE

.endm
```

**Figura 3.** Segunda Sección de Código

```
;para lectura en la eeprom
.macro eeprom_read

    eeprom_memory @0,@1

    ;EECR = registro de control para lectura o escritura
    ;EEMWE = indica que se inicie la lectura
    sbi EECR,EERE

    ;guardando valor leido de la eeprom
    in @2,EEDR

.endm

;codigo principal
rjmp main

;codigo int0
rjmp teclado

;codigo int1
rjmp borrar

;codigo timer1 para contador de anillo
.org $008
rjmp contadorAnillo
```

**Figura 4.** Tercera Sección de Código

```
main:
    ;configurando apuntador de pila
    ldi aux,low(ramend)
    out spl,aux
    ldi aux,high(ramend)
    out sph,aux

    ;puertos a y c como salidas
    ;puerto b y d como entradas
    ser aux
    out ddra,aux
    out ddrc,aux
    out portb,aux
    out portd,aux

    ;timer1 sin preescalado
    ldi aux,1
    out tccr1b,aux

    ;habilitando desborde timer1
    ldi aux,4
    out timsk,aux

    ;detectando los flancos de subida
    ;para int0 e int1 en el registro mcucr
    ldi aux,0b00001111
    out mcucr,aux
```

**Figura 5.** Cuarta Sección de Código

```
;habilitando int0 e int1 con gicr
ldi aux,0b11000000
out gicr,aux

;habilitando interrupciones globalmente
sei

;cargando guiones en la eeprom para que
;posteriormente se muestre en los displays
ldi aux,$40
eeprom_write $00,$00,aux
eeprom_write $00,$01,aux
eeprom_write $00,$02,aux
eeprom_write $00,$03,aux
eeprom_write $00,$04,aux
eeprom_write $00,$05,aux
eeprom_write $00,$06,aux
eeprom_write $00,$07,aux

;ciclo infinito
loop:
    ;se muestra lo que hay en la eeprom en los displays
    rcall actualizarDisplays
    rjmp loop
```

**Figura 6.** Quinta Sección de Código

```
;interrupcion int0
;detecta una tecla y la guarda en la eeprom
;recorriendo el valor en los displays
teclado:
    ;apagando los displays
    ldi aux,0
    out porta,aux

    ;deteniendo contador de anillo
    ldi aux,0
    out tccr1b,aux

    ;revisando si fue presionada la fila 1 del teclado
    sbic pinb,3
    rjmp sig1

        ;si se presiona la columna 1 se muestra el 7
            sbic pinb,7
            rjmp val0
            ldi display,7
            rjmp finTeclado

        ;si se presiona la columna 2 se muestra el 8
        val0:
            sbic pinb,6
            rjmp val1
            ldi display,8
            rjmp finTeclado
```

**Figura 7.** Sexta Sección de Código

```
    ;si se presiona la columna 3 se muestra el 9
    val1:
        sbic pinb,5
        rjmp val2
        ldi display,9
        rjmp finTeclado

    ;si se presiona la columna 4 se muestra el %
    val2:
        sbic pinb,4
        brne sig1
        ldi display,10
        rjmp finTeclado

;revisando si fue presionada la fila 2 del teclado
sig1:
    sbic pinb,2
    rjmp sig2

    ;si se presiona la columna 1 se muestra el 4
        sbic pinb,7
        rjmp val3
        ldi display,4
        rjmp finTeclado
```

**Figura 8.** Séptima Sección de Código

```
    ;si se presiona la columna 2 se muestra el 5
    val3:
        sbic pinb,6
        rjmp val4
        ldi display,5
        rjmp finTeclado

    ;si se presiona la columna 3 se muestra el 6
    val4:
        sbic pinb,5
        rjmp val5
        ldi display,6
        rjmp finTeclado

    ;si se presiona la columna 4 se muestra la X
    val5:
        sbic pinb,4
        brne sig2
        ldi display,11
        rjmp finTeclado

;revisando si fue presionada la fila 3 del teclado
sig2:
    sbic pinb,1
    rjmp sig3
```

**Figura 9.** Octava Sección de Código

```
;si se presiona la columna 1 se muestra el 1
    sbic pinb,7
    rjmp val6
    ldi display,1
    rjmp finTeclado

;si se presiona la columna 2 se muestra el 2
val6:
    sbic pinb,6
    rjmp val7
    ldi display,2
    rjmp finTeclado

;si se presiona la columna 3 se muestra el 3
val7:
    sbic pinb,5
    rjmp val8
    ldi display,3
    rjmp finTeclado

;si se presiona la columna 4 se muestra el -
val8:
    sbic pinb,4
    brne sig3
    ldi display,12
    rjmp finTeclado
```

**Figura 10.** Novena Sección de Código

```
;revisando si fue presionada la fila 4 del teclado
sig3:
    sbic pinb,0
    rjmp finSubrutina


    ;si se presiona la columna 1 se muestra el ON/C
        sbic pinb,7
        rjmp val9
        ldi display,13
        rjmp finTeclado

    ;si se presiona la columna 2 se muestra el 0
    val9:
        sbic pinb,6
        rjmp val10
        ldi display,0
        rjmp finTeclado

    ;si se presiona la columna 3 se muestra el =
    val10:
        sbic pinb,5
        rjmp val11
        ldi display,14
        rjmp finTeclado
```

**Figura 11.** Décima Sección de Código

```
    ;si se presiona la columna 4 se muestra el +
    val11:
        sbic pinb,4
        brne finSubrutina
        ldi display,15
        rjmp finTeclado

;si alguna tecla fue presionada entonces
;se ejecuta siempre esta seccion de codigo
finTeclado:

    ;se espera a que se deje de presionar la tecla pulsada
    esperarBoton:

        clr flag

        sbis pinb,0
        ldi flag,1

        sbis pinb,1
        ldi flag,1

        sbis pinb,2
        ldi flag,1

        sbis pinb,3
        ldi flag,1
```

**Figura 12.** Décimo Primera Sección de Código

```
        cpi flag,0
        brne esperarBoton

    ;cuando se deja de presionar el boton se
    ;convierte el valor a mostrar en 7 segmentos
    rcall conversion

    ;se guarda el valor a mostrar, recorriendo
    ;los que ya se tenian hacia la izquierda
    rcall recorrerDisplays

    ;se vuelve a iniciar el contador de anillo
    ldi aux,2
    out tccr1b,aux

;si no se presiona ninguna tecla se
;ejecuta esta seccion de codigo
finSubrutina:
    reti
```

**Figura 13.** Décimo Segunda Sección de Código

```
;este codigo se ejecuta en int0 para guardar un
;nuevo valor en la eeprom y mostrarlo en los displays
recorrerDisplays:
    ;recorriendo un valor a la izquierda
    eeprom_read $00,$01,aux
    eeprom_write $00,$00,aux

    ;recorriendo un valor a la izquierda
    eeprom_read $00,$02,aux
    eeprom_write $00,$01,aux

    ;recorriendo un valor a la izquierda
    eeprom_read $00,$03,aux
    eeprom_write $00,$02,aux

    ;recorriendo un valor a la izquierda
    eeprom_read $00,$04,aux
    eeprom_write $00,$03,aux

    ;recorriendo un valor a la izquierda
    eeprom_read $00,$05,aux
    eeprom_write $00,$04,aux

    ;recorriendo un valor a la izquierda
    eeprom_read $00,$06,aux
    eeprom_write $00,$05,aux
```

**Figura 14.** Décimo Tercera Sección de Código

```
    ;recorriendo un valor a la izquierda
    eeprom_read $00,$07,aux
    eeprom_write $00,$06,aux

    ;almacenando la tecla pulsada hasta la derecha
    eeprom_write $00,$07,display

    ret

;interrupcion int1
;detecta un push button y borra un valor al ingresar un
;guion desde la derecha recorriendo el valor en los displays
borrar:
    ;recorriendo un valor a la derecha
    eeprom_read $00,$06,aux
    eeprom_write $00,$07,aux

    ;recorriendo un valor a la derecha
    eeprom_read $00,$05,aux
    eeprom_write $00,$06,aux

    ;recorriendo un valor a la derecha
    eeprom_read $00,$04,aux
    eeprom_write $00,$05,aux

    ;recorriendo un valor a la derecha
    eeprom_read $00,$03,aux
    eeprom_write $00,$04,aux
```

**Figura 15.** Décimo Cuarta Sección de Código

```
    ;recorriendo un valor a la derecha
    eeprom_read $00,$02,aux
    eeprom_write $00,$03,aux

    ;recorriendo un valor a la derecha
    eeprom_read $00,$01,aux
    eeprom_write $00,$02,aux

    ;recorriendo un valor a la derecha
    eeprom_read $00,$00,aux
    eeprom_write $00,$01,aux

    ;almacenando el guion en la posicion de hasta la izquierda
    ldi aux,$40
    eeprom_write $00,$00,aux

    reti

;timer1
;contador de anillo para los displays
contadorAnillo:
    ;reiniciando timer1 para que siempre vaya rapido el conteo
    ldi aux,255
    out tcnt1h,aux
    ldi aux,0
    out tcnt1l,aux
```

**Figura 16.** Décimo Quinta Sección de Código

```
    ;cargando valor del contador de anillo
    out portc,anillo

    ;desplazando bit
    lsl anillo

    ;verificando si es necesario reset
    cpi anillo,0
    brne saltarReinicio

    ;reset del contador de anillo
    ldi anillo,1

    saltarReinicio:
        reti

;se muestra lo que hay en la eeprom en los displays
actualizarDisplays:
    ;si el bit 7 del contador de anillo esta encendido
    ;se carga el valor 1 almacenado en la eeprom
    cpi anillo,$80
    brne bitDos
    eeprom_read $00,$01,aux
    rjmp salir
```

**Figura 17.** Décimo Sexta Sección de Código

```
;si el bit 6 del contador de anillo esta encendido
;se carga el valor 2 almacenado en la eeprom
bitDos:
cpi anillo,$40
brne bitTres
eeprom_read $00,$02,aux
rjmp salir

;si el bit 5 del contador de anillo esta encendido
;se carga el valor 3 almacenado en la eeprom
bitTres:
cpi anillo,$20
brne bitCuatro
eeprom_read $00,$03,aux
rjmp salir

;si el bit 4 del contador de anillo esta encendido
;se carga el valor 4 almacenado en la eeprom
bitCuatro:
cpi anillo,$10
brne bitCinco
eeprom_read $00,$04,aux
rjmp salir
```

**Figura 18.** Décimo Séptima Sección de Código

```
;si el bit 3 del contador de anillo esta encendido
;se carga el valor 5 almacenado en la eeprom
bitCinco:
cpi anillo,8
brne bitSeis
eeprom_read $00,$05,aux
rjmp salir

;si el bit 2 del contador de anillo esta encendido
;se carga el valor 6 almacenado en la eeprom
bitSeis:
cpi anillo,4
brne bitSiete
eeprom_read $00,$06,aux
rjmp salir

;si el bit 1 del contador de anillo esta encendido
;se carga el valor 7 almacenado en la eeprom
bitSiete:
cpi anillo,2
brne bitOcho
eeprom_read $00,$07,aux
rjmp salir
```

**Figura 19.** Décimo Octava Sección de Código

```
    ;si el bit 0 del contador de anillo esta encendido
    ;se carga el valor 0 almacenado en la eeprom
    bitOcho:
    cpi anillo,1
    brne salir
    eeprom_read $00,$00,aux

    salir:
    out porta,aux
    ret

;para macros de lectura y escritura de eeprom
eeprom_loop:
    ;EECR = registro de control
    ;EEWE = bit que indica si eeprom esta lista para escritura
    ;esperar a que el bit EEWE tenga un cero
    sbic EECR,EEWE
    rjmp eeprom_loop
    ret
```

**Figura 20.** Décimo Novena Sección de Código

```
;funcion para convertir una tecla pulsada
;a su equivalente en display 7 segmentos
conversion:
    ;si es el cero se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,0
    brne saltarCero

    ldi display,$3f
    ret

    saltarCero:

        ;si es el uno se carga el valor en
        ;7 segmentos y se termina la subrutina
        cpi display,1
        brne saltarUno

        ldi display,6
        ret

    saltarUno:

        ;si es el dos se carga el valor en
        ;7 segmentos y se termina la subrutina
        cpi display,2
        brne saltarDos
```

**Figura 21.** Vigésima Sección de Código

```
     ldi display,$5b
     ret

saltarDos:

     ;si es el tres se carga el valor en
     ;7 segmentos y se termina la subrutina
     cpi display,3
     brne saltarTres

     ldi display,$4f
     ret

saltarTres:

     ;si es el cuatro se carga el valor en
     ;7 segmentos y se termina la subrutina
     cpi display,4
     brne saltarCuatro

     ldi display,$66
     ret
```

**Figura 22.** Vigésimo Primera Sección de Código

```
saltarCuatro:

    ;si es el cinco se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,5
    brne saltarCinco

    ldi display,$6d
    ret

saltarCinco:

    ;si es el seis se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,6
    brne saltarSeis

    ldi display,$7d
    ret

saltarSeis:

    ;si es el siete se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,7
    brne saltarSiete

    ldi display,7
    ret
```

**Figura 23.** Vigésimo Segunda Sección de Código

```
saltarSiete:

    ;si es el ocho se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,8
    brne saltarOcho

    ldi display,$7f
    ret

saltarOcho:

    ;si es el nueve se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,9
    brne saltarNueve

    ldi display,$6f
    ret

saltarNueve:

    ;si es el % se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,10
    brne saltarDivision

    ldi display,$52
    ret
```

**Figura 24.** Vigésimo Tercera Sección de Código

```
saltarDivision:

    ;si es la X se carga el valor H en
    ;7 segmentos y se termina la subrutina
    cpi display,11
    brne saltarMultiplicacion

    ldi display,$76
    ret

saltarMultiplicacion:

    ;si es el - se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,12
    brne saltarResta

    ldi display,$40
    ret

saltarResta:

    ;si es el ON/C se carga el valor C en
    ;7 segmentos y se termina la subrutina
    cpi display,13
    brne saltarOnC

    ldi display,$39
    ret
```

**Figura 25.** Vigésimo Cuarta Sección de Código

```
saltarOnC:

    ;si es el = se carga el valor en
    ;7 segmentos y se termina la subrutina
    cpi display,14
    brne saltarIgual

    ldi display,$48
    ret

saltarIgual:

    ;si es el + se carga el valor E en
    ;7 segmentos y se termina la subrutina
    cpi display,15
    brne fin

    ldi display,$79

fin: ret
```

**Figura 26.** Vigésimo Quinta Sección de Código

El código en lenguaje C correspondiente al diagrama de flujo es el siguiente:

```
//libreria inicial
#include<avr/io.h>

//libreria para interrupciones
#include<avr/interrupt.h>

//libreria para eeprom
#include<avr/eeprom.h>

//variables de apoyo
int anillo = 1;
int display;
int aux;

//funcion para configurar puertos y variables
void config_io(void){
    //configurando como salida
    DDRA = 0xff;

    //configurando como salida
    DDRC = 0xff;

    //configurando como entrada
    DDRD = 0x00;

    //configurando como entrada
    DDRB = 0x00;
```

**Figura 27.** Primera Sección de Código

```
    //configurando pull up
    PORTB = 0xff;

    //habilitando interrupcion desborde timer1
    TIMSK = _BV(TOIE1);

    //iniciando conteo timer1 sin preescalado
    TCCR1B = _BV(CS10);

    //int1 e int0 incrementan con flancos de subida
    MCUCR = 15;

    //habilitando int1 e int0
    GICR = _BV(INT1)|_BV(INT0);
}

//funcion para inicializar eeprom
//cargando guiones en la eeprom para que
//posteriormente se muestre en los displays
void cargarValoresIniciales(void){
    eeprom_is_ready();
    eeprom_write_byte(0x00,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x01,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x02,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x03,0x40);
```

**Figura 28.** Segunda Sección de Código

```
    eeprom_is_ready();
    eeprom_write_byte(0x04,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x05,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x06,0x40);
    eeprom_is_ready();
    eeprom_write_byte(0x07,0x40);
}

//funcion para actualizar displays
void actualizarDisplays(void){
    eeprom_is_ready();
    switch(anillo){
        case(128):
            PORTA = eeprom_read_byte(0x01);
            break;
        case(64):
            PORTA = eeprom_read_byte(0x02);
            break;
        case(32):
            PORTA = eeprom_read_byte(0x03);
            break;
        case(16):
            PORTA = eeprom_read_byte(0x04);
            break;
        case(8):
            PORTA = eeprom_read_byte(0x05);
```

**Figura 29.** Tercera Sección de Código

```
            break;
        case(4):
            PORTA = eeprom_read_byte(0x06);
            break;
        case(2):
            PORTA = eeprom_read_byte(0x07);
            break;
        case(1):
            PORTA = eeprom_read_byte(0x00);
            break;
    }
}

//main
int main(void){
    //configurando puertos y variables
    config_io();

    //habilitando interrupciones globalmente
    sei();

    //escritura eeprom valores iniciales
    cargarValoresIniciales();

    //ciclando el programa
    while(1)
        //actualizando displays
        actualizarDisplays();
}
```

**Figura 30.** Cuarta Sección de Código

```
//funcion para desplazar manualmente
//un bit del contador de anillo
void desplazarContadorAnillo(void){
    switch(anillo){
        case(1):
            anillo = 2;
            break;
        case(2):
            anillo = 4;
            break;
        case(4):
            anillo = 8;
            break;
        case(8):
            anillo = 16;
            break;
        case(16):
            anillo = 32;
            break;
        case(32):
            anillo = 64;
            break;
        case(64):
            anillo = 128;
            break;
        case(128):
            anillo = 1;
            break;
    }
}
```

**Figura 31.** Quinta Sección de Código

```
//funcion del timer1 para contador de anillo
ISR(TIMER1_OVF_vect){
    //reiniciando timer1 para que siempre vaya rapido el conteo
    TCNT1H = 255;
    TCNT1L = 0;

    //cargando valor del contador de anillo
    PORTC = anillo;

    //desplazando bit
    desplazarContadorAnillo();
}

//funcion de int1 para borrar un valor
ISR(INT1_vect){
    eeprom_is_ready();
    aux = eeprom_read_byte(0x06);
    eeprom_is_ready();
    eeprom_write_byte(0x07,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x05);
    eeprom_is_ready();
    eeprom_write_byte(0x06,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x04);
    eeprom_is_ready();
    eeprom_write_byte(0x05,aux);
```

**Figura 32.** Sexta Sección de Código

```
    eeprom_is_ready();
    aux = eeprom_read_byte(0x03);
    eeprom_is_ready();
    eeprom_write_byte(0x04,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x02);
    eeprom_is_ready();
    eeprom_write_byte(0x03,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x01);
    eeprom_is_ready();
    eeprom_write_byte(0x02,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x00);
    eeprom_is_ready();
    eeprom_write_byte(0x01,aux);

    //almacenando el guion en la posicion de hasta la izquierda
    eeprom_is_ready();
    eeprom_write_byte(0x00,0x40);
}
```

**Figura 33.** Séptima Sección de Código

```
//este codigo se ejecuta en int0 para guardar un
//nuevo valor en la eeprom y mostrarlo en los displays
void recorrerDisplays(void){
    eeprom_is_ready();
    aux = eeprom_read_byte(0x01);
    eeprom_is_ready();
    eeprom_write_byte(0x00,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x02);
    eeprom_is_ready();
    eeprom_write_byte(0x01,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x03);
    eeprom_is_ready();
    eeprom_write_byte(0x02,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x04);
    eeprom_is_ready();
    eeprom_write_byte(0x03,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x05);
    eeprom_is_ready();
    eeprom_write_byte(0x04,aux);
```

**Figura 34.** Octava Sección de Código

```
    eeprom_is_ready();
    aux = eeprom_read_byte(0x06);
    eeprom_is_ready();
    eeprom_write_byte(0x05,aux);

    eeprom_is_ready();
    aux = eeprom_read_byte(0x07);
    eeprom_is_ready();
    eeprom_write_byte(0x06,aux);

    //almacenando la tecla pulsada hasta la derecha
    eeprom_is_ready();
    eeprom_write_byte(0x07,display);
}

//funcion para convertir una tecla pulsada
//a su equivalente en display 7 segmentos
void conversion(void){
    switch(display){
        //para el cero en 7 segmentos
        case(0):
            display = 0x3f;
            break;
        //para el uno en 7 segmentos
        case(1):
            display = 6;
            break;
```

**Figura 35.** Novena Sección de Código

```
//para el dos en 7 segmentos
case(2):
    display = 0x5b;
    break;
//para el tres en 7 segmentos
case(3):
    display = 0x4f;
    break;
//para el cuatro en 7 segmentos
case(4):
    display = 0x66;
    break;
//para el cinco en 7 segmentos
case(5):
    display = 0x6d;
    break;
//para el seis en 7 segmentos
case(6):
    display = 0x7d;
    break;
//para el siete en 7 segmentos
case(7):
    display = 7;
    break;
//para el ocho en 7 segmentos
case(8):
    display = 0x7f;
    break;
```

**Figura 36.** Décima Sección de Código

```
        //para el nueve en 7 segmentos
        case(9):
            display = 0x6f;
            break;
        //para el % en 7 segmentos (/)
        case(10):
            display = 0x52;
            break;
        //para la X en 7 segmentos (H)
        case(11):
            display = 0x76;
            break;
        //para el - en 7 segmentos
        case(12):
            display = 0x40;
            break;
        //para el ON/C en 7 segmentos (C)
        case(13):
            display = 0x39;
            break;
        //para el = en 7 segmentos
        case(14):
            display = 0x48;
            break;
        //para el + en 7 segmentos (E)
        case(15):
            display = 0x79;
            break;
    }
}
```

**Figura 37.** Décimo Primera Sección de Código

```
//funcion de int0 para cuando se pulsa una tecla
ISR(INT0_vect){
    //apagando los displays
    PORTA = 0;

    //deteniendo contador de anillo
    TCCR1B = 0;

    //revisando si fue presionada la fila 1 del teclado
    if(bit_is_clear(PINB,3)){
        //si se presiona la columna 1 se muestra el 7
        if(bit_is_clear(PINB,7))
            display = 7;
        //si se presiona la columna 2 se muestra el 8
        else if(bit_is_clear(PINB,6))
            display = 8;
        //si se presiona la columna 3 se muestra el 9
        else if(bit_is_clear(PINB,5))
            display = 9;
        //si se presiona la columna 4 se muestra el %
        else if(bit_is_clear(PINB,4))
            display = 10;
    }
```

**Figura 38.** Décimo Segunda Sección de Código

```
//revisando si fue presionada la fila 2 del teclado
else if(bit_is_clear(PINB,2)){
    //si se presiona la columna 1 se muestra el 4
    if(bit_is_clear(PINB,7))
        display = 4;
    //si se presiona la columna 2 se muestra el 5
    else if(bit_is_clear(PINB,6))
        display = 5;
    //si se presiona la columna 3 se muestra el 6
    else if(bit_is_clear(PINB,5))
        display = 6;
    //si se presiona la columna 4 se muestra la X
    else if(bit_is_clear(PINB,4))
        display = 11;
}
//revisando si fue presionada la fila 3 del teclado
else if(bit_is_clear(PINB,1)){
    //si se presiona la columna 1 se muestra el 1
    if(bit_is_clear(PINB,7))
        display = 1;
    //si se presiona la columna 2 se muestra el 2
    else if(bit_is_clear(PINB,6))
        display = 2;
    //si se presiona la columna 3 se muestra el 3
    else if(bit_is_clear(PINB,5))
        display = 3;
    //si se presiona la columna 4 se muestra el -
    else if(bit_is_clear(PINB,4))
        display = 12;
}
```

**Figura 39.** Décimo Tercera Sección de Código

```
//revisando si fue presionada la fila 4 del teclado
else if(bit_is_clear(PINB,0)){
    //si se presiona la columna 1 se muestra el ON/C
    if(bit_is_clear(PINB,7))
        display = 13;
    //si se presiona la columna 2 se muestra el 0
    else if(bit_is_clear(PINB,6))
        display = 0;
    //si se presiona la columna 3 se muestra el =
    else if(bit_is_clear(PINB,5))
        display = 14;
    //si se presiona la columna 4 se muestra el +
    else if(bit_is_clear(PINB,4))
        display = 15;
}
```

**Figura 40.** Décimo Cuarta Sección de Código

```
    //se espera a que se deje de presionar la tecla pulsada
    loop_until_bit_is_set(PINB,0);
    loop_until_bit_is_set(PINB,1);
    loop_until_bit_is_set(PINB,2);
    loop_until_bit_is_set(PINB,3);

    //cuando se deja de presionar el boton se
    //convierte el valor a mostrar en 7 segmentos
    conversion();

    //se guarda el valor a mostrar, recorriendo
    //los que ya se tenian hacia la izquierda
    recorrerDisplays();

    //se vuelve a iniciar el contador de anillo
    TCCR1B = 2;
}
```

**Figura 41.** Décimo Quinta Sección de Código

Para la simulación en Proteus, primero se muestra los valores iniciales, posteriormente se ingresan algunas teclas y luego se eliminan otras, de manera que los resultados son los siguientes:

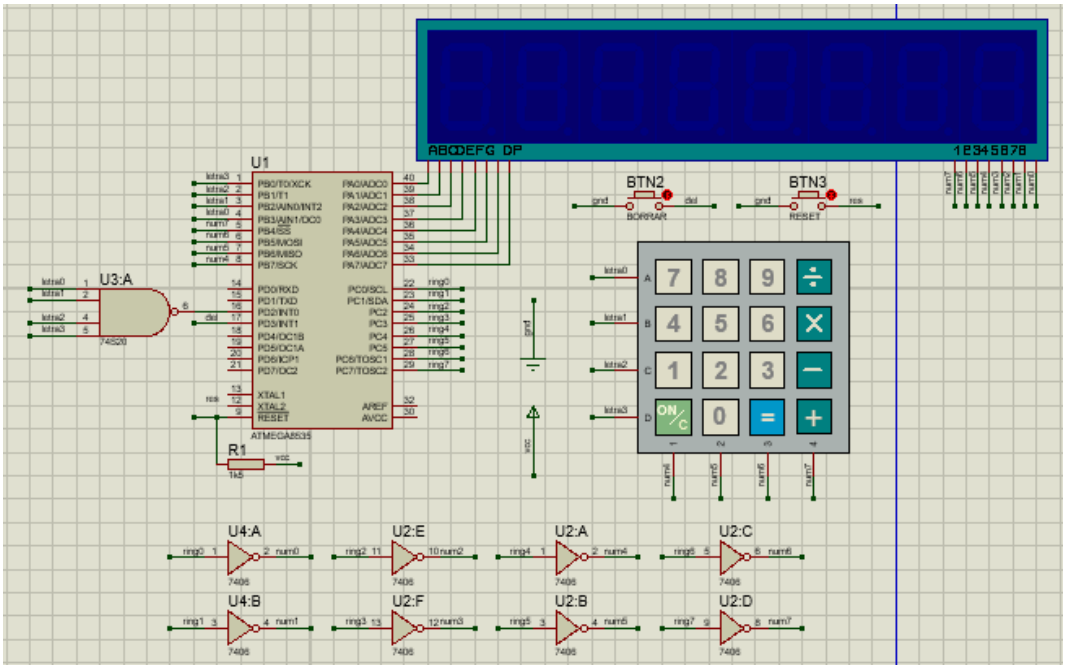



**Figura 42.** Circuito en Proteus

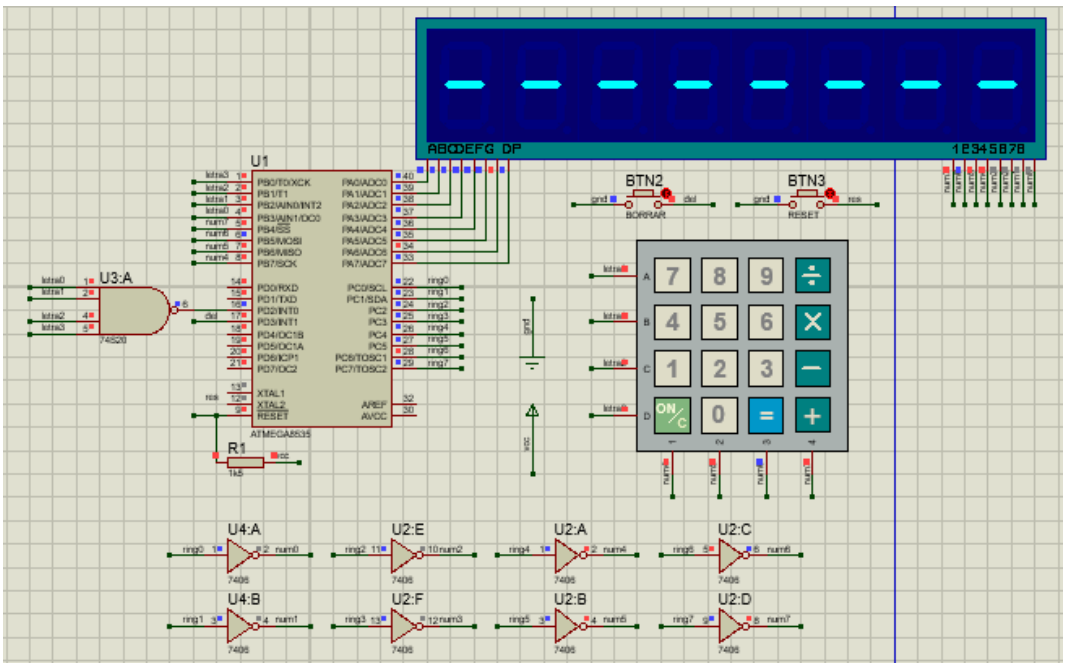



**Figura 43.** Valores Iniciales en Displays

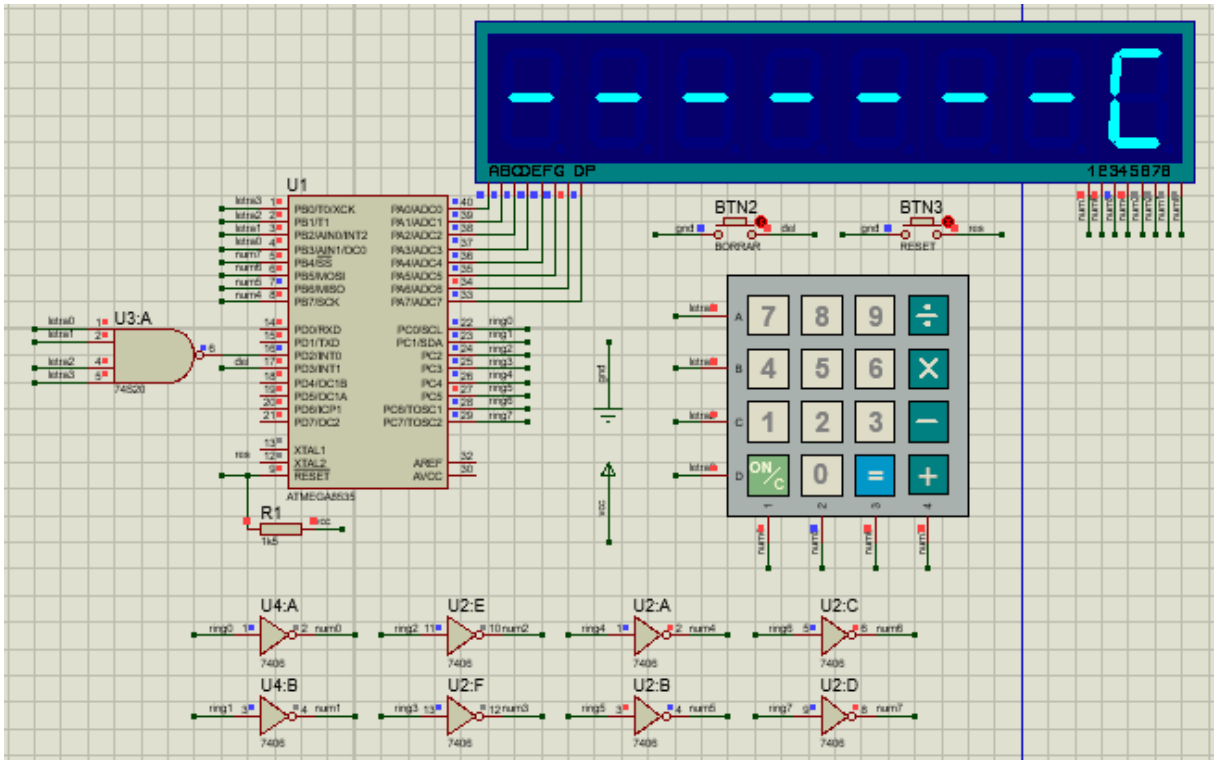



**Figura 44.** Se Presiona la Tecla ON/C

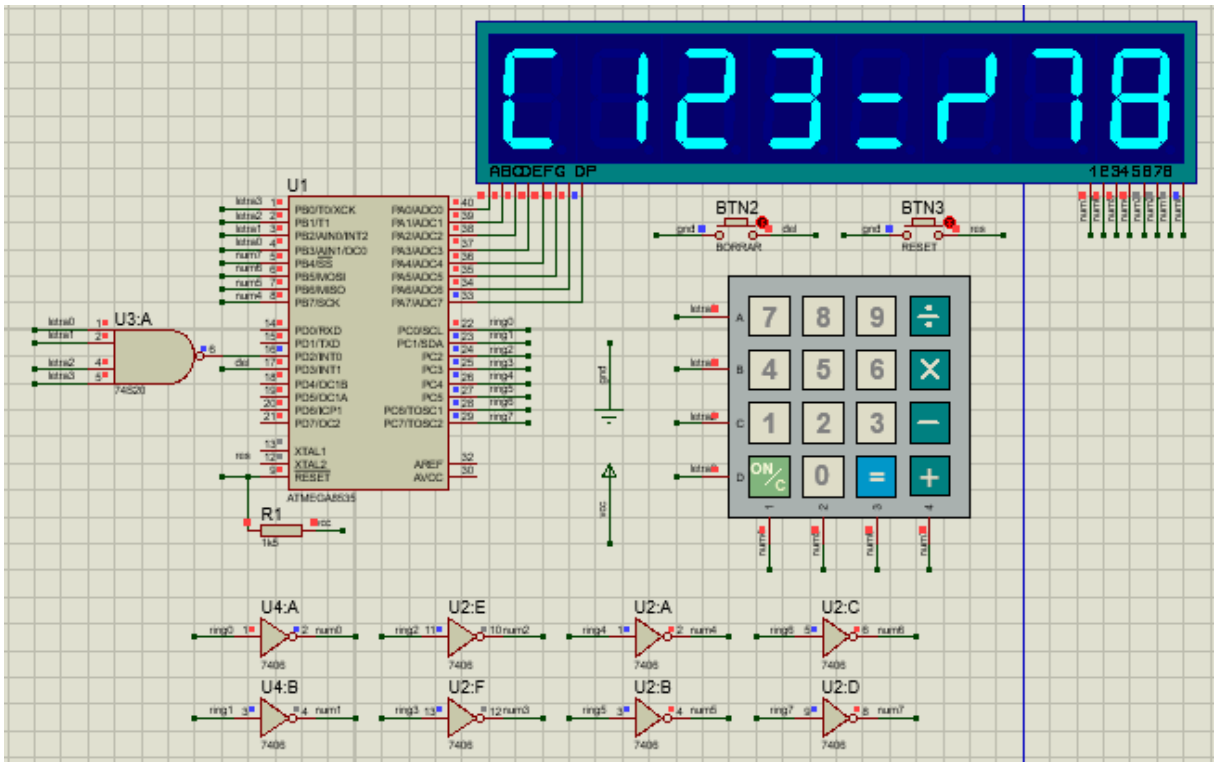



**Figura 45.** Se Presionan Teclas Hasta Llenar Todos los Displays

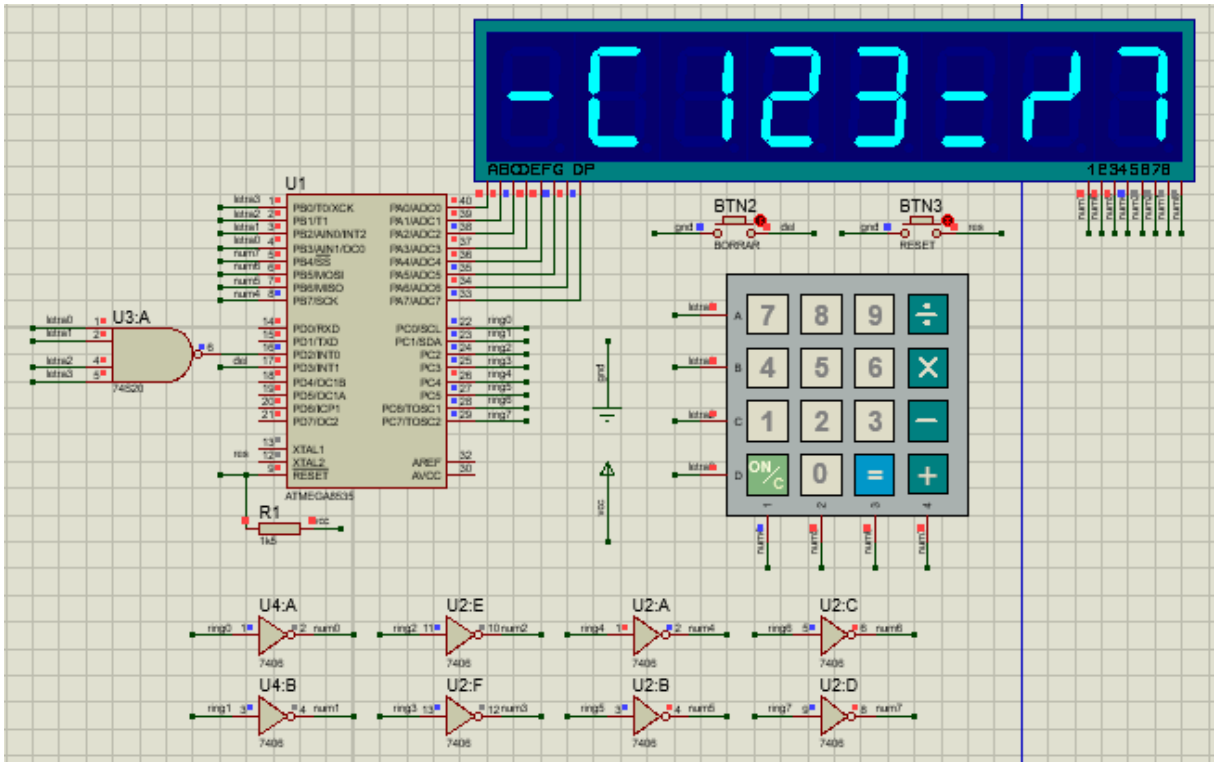



**Figura 46.** Usando la Función de Borrado

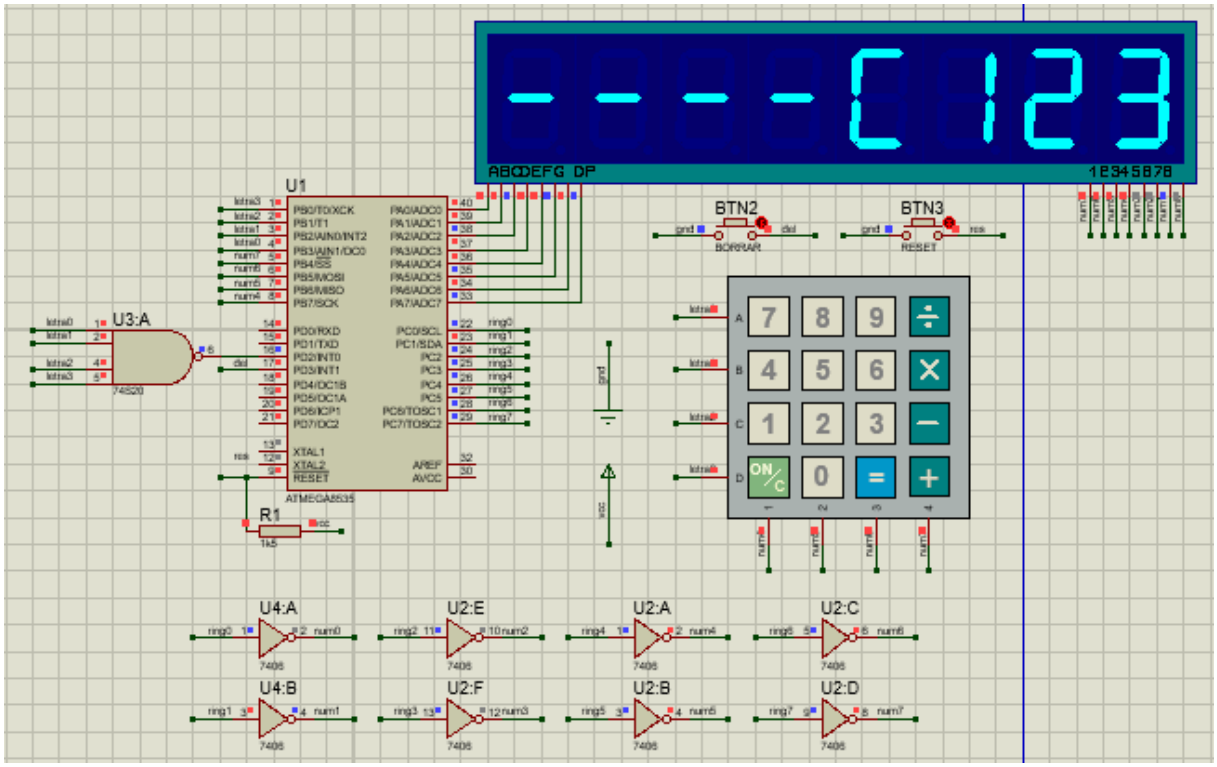



**Figura 47.** Borrando Múltiples Veces

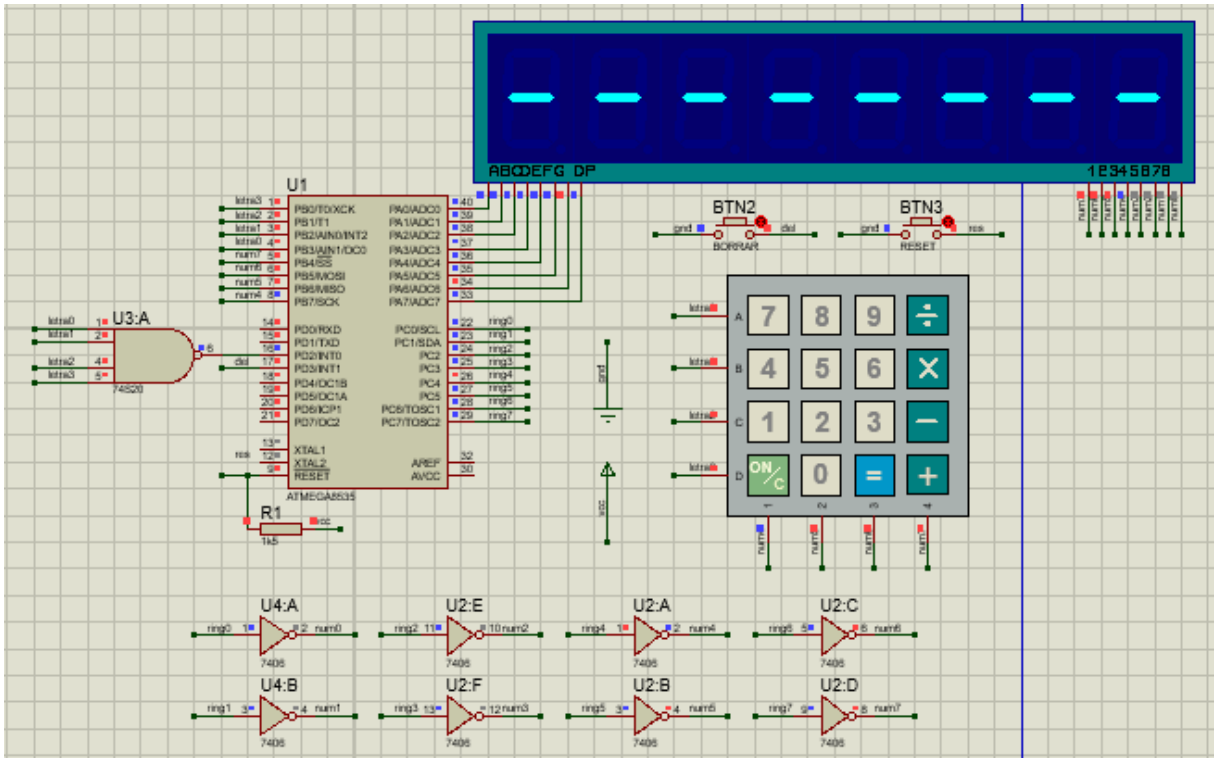
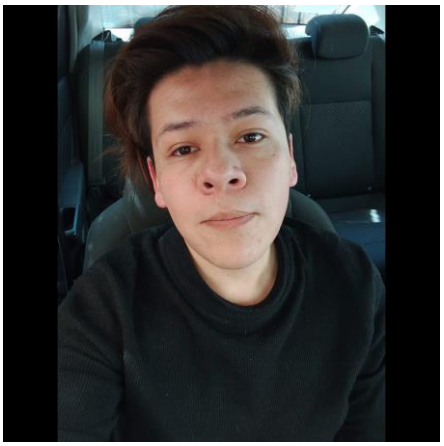



**Figura 33.** Borrando Hasta Limpiar Todos los Displays

# Conclusiones Individuales

## Contreras Cardoso Adolfo



Para finalizar la realización de esta práctica fue un reto para mis compañeros y para mí, ya que se emplearon todos los conceptos y temas que se vieron durante todo el semestre, el objetivo de la práctica se logró, gracias al profesor y un buen equipo se logró entender algunos funcionamientos que nos proporcionan los microcontroladores, en lo personal las asignaturas de éste tipo no me agradan y ha resultado ser un reto a lo largo de la carrera, agradezco a mis compañeros de trabajo y al profesor por su constancia en las clases y el poco apoyo que brindó.

## Martínez Alvarado Bryan Alexis

La realización de este proyecto ha sido fundamental para aplicar, utilizar y analizar todo lo que se vio durante el semestre de la Unidad de Introducción a los Microcontroladores. Se ha podido concluir con esta práctica, en donde se empleó el microcontrolador acompañado de más componentes para lograr crear un circuito con una funcionalidad específica, que sería de gran utilidad. De esta manera concluimos esta Unidad de Aprendizaje de manera satisfactoria.

## Maya Martínez Alonso Rubén

La realización de este proyecto fue complicada ya que usamos todo lo que vimos en el semestre y logramos juntar todos esos conocimientos para hacer un sistema más complejo como lo es esta calculadora también me hace pensar que tan poderosos son los microcontroladores que comandan calculadoras más complejas y como estos son programados. Es interesante pensar en todas las posibles aplicaciones a este teclado matricial y también pensar en cómo funcionan los teclados de computadora siendo muchísimas más teclas. Fue interesante el curso y sobre todo retador.

## Pérez Gómez Santiago

El desarrollo e implementación de este trabajo resulto el mayor reto hasta ahora, pues el problema planteado requiere emplear todos los conocimientos adquiridos en el curso de Introducción a los Microcontroladores, razón por la cual, podemos concluir con dicha asignatura de manera satisfactoria. Personalmente, me siento satisfecho con todos los resultados obtenidos hasta ahora, así como con el tiempo y esfuerzo que se empleó en cada cosa.