# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

## “Conteo de clientes con EEPROM en C”


**Integrantes del Equipo:**

Contreras Cardoso Adolfo

Martínez Alvarado Bryan Alexis

Maya Martínez Alonso Rubén

Pérez Gómez Santiago


**Grupo:**

3CM15

**Profesor:**

Ing. José Juan Pérez Pérez

**Asignatura:**

Introducción a los Microcontroladores

## Desarrollo

Para el circuito usado en el ejercicio de contador de clientes, agregar 2 displays bcd 7 seg. en el puerto D, en estos displays se mostrará la cuenta de clientes ganadores. La primera vez que funcione el circuito mostrará “00” y por cada cliente ganador se irá incrementado en forma decimal hasta “99”. Cada vez que se incrementa esta cuenta se debe guardar en la EEPROM este valor, si se apaga el sistema o se pulsa “RESET” debe iniciar la cuenta con el último valor guardado,si se pulsa INT2 la cuenta deberá iniciarse a “00”.

El diagrama de flujo correspondiente al resultado es el siguiente:

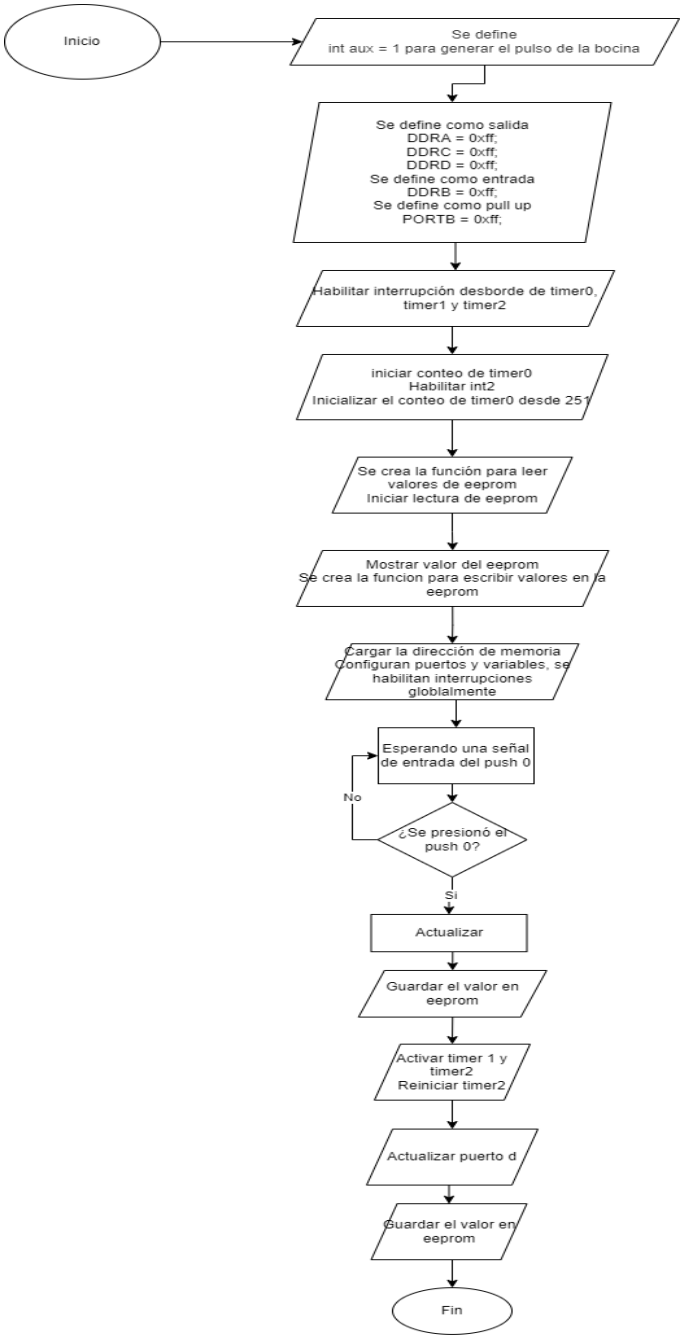


Figura 1. Diagrama de flujo

El respectivo código en ensamblador es el siguiente:

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

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

//para generar pulso de la bocina
int aux = 1;

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

    //configurando como salida
    DDRC = 0xff;

    //configurando como salida
    DDRD = 0xff;

    //configurando como entrada
    DDRB = 0x00;

    //configurando pull up
    PORTB = 0xff;

    //habilitando interrupcion desborde timer0, timer1 y timer2
    TIMSK = _BV(TOIE2)|_BV(TOIE1)|_BV(TOIE0);

    //iniciando conteo timer0
    TCCR0 = _BV(CS02)|_BV(CS01);

    //int2 incrementa con flancos de bajada
    MCUCSR = 0;

    //habilitando int2
    GICR = _BV(INT2);

    //usando notacion decimal para inicializar
    //el conteo de timer0 desde 251
    TCNT0 = 250;
}
```

Figura 2. Primer sección de código

```
//funcion para leer valores de la eeprom
int leer_eeprom(int high, int low){

    //esperando eeprom
    loop_until_bit_is_clear(EECR,EEWE);

    //cargando direccion de memoria
    EEARH = high;
    EEARL = low;

    //iniciando lectura de eeprom
    EECR = _BV(EERE);

    //mostrando valor
    return EEDR;
}

//funcion para escribir valores en la eeprom
void escribir_eeprom(int high, int low, int value){
    //esperando eeprom
    loop_until_bit_is_clear(EECR,EEWE);

    //cargando direccion de memoria
    EEARH = high;
    EEARL = low;

    //escritura de la eeprom
    EEDR = value;
    EECR = _BV(EEMWE);
    EECR = _BV(EEWE);
}
```

Figura 3. Segunda sección de código

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

    //habilitando interrupciones globalmente
    sei();

    //lectura eeprom
    PORTD = leer_eeprom(0,0);

    //ciclando el programa
    while(1)
        //actualizando cada que se presiona el push 0
        if(PB0 == 0)
            PORTC = 255 - TCNT0;
}
```

Figura 4. Tercera sección de código

```
//funcion del timer 2
ISR(TIMER2_OVF_vect){
    TCNT2 = 114;

    PORTA = aux;

    if(aux == 0)
        aux = 1;
    else
        aux = 0;
}

//funcion del timer 1
ISR(TIMER1_OVF_vect){
    TCCR1B = 0;
    TCCR2 = 0;
}

//funcion del timer 0
ISR(TIMER0_OVF_vect){
    //actualizando puerto d
    PORTD = PORTD + 1;

    //guardando valor
    escribir_eeprom(0,0,PORTD);

    //activando timer1 y timer2
    TCNT0 = 250;
    TCCR1B = _BV(CS12);
    TCCR2 = _BV(CS21);

    //reiniciando timer1
    TCNT1H = 0xb3;
    TCNT1L = 0xb5;

    //reiniciando timer2
    TCNT2 = 114;
}
```

Figura 5. Cuarta sección de código

```
//funcion de int2
ISR(INT2_vect){
    //actualizando puerto d
    PORTD = 0;

    //guardando valor
    escribir_eeprom(0,0,PORTD);
}
```

Figura 6. Quinta sección de código

Simulación:

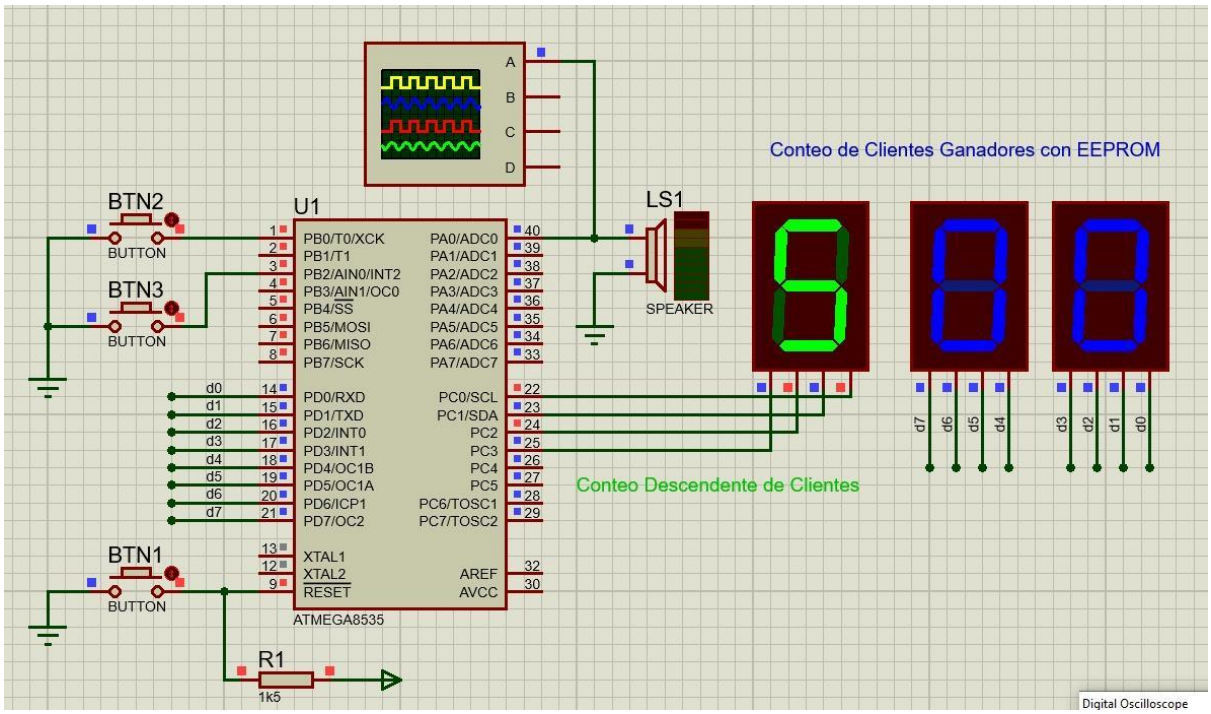


Figura 7. Inicio y primer conteo

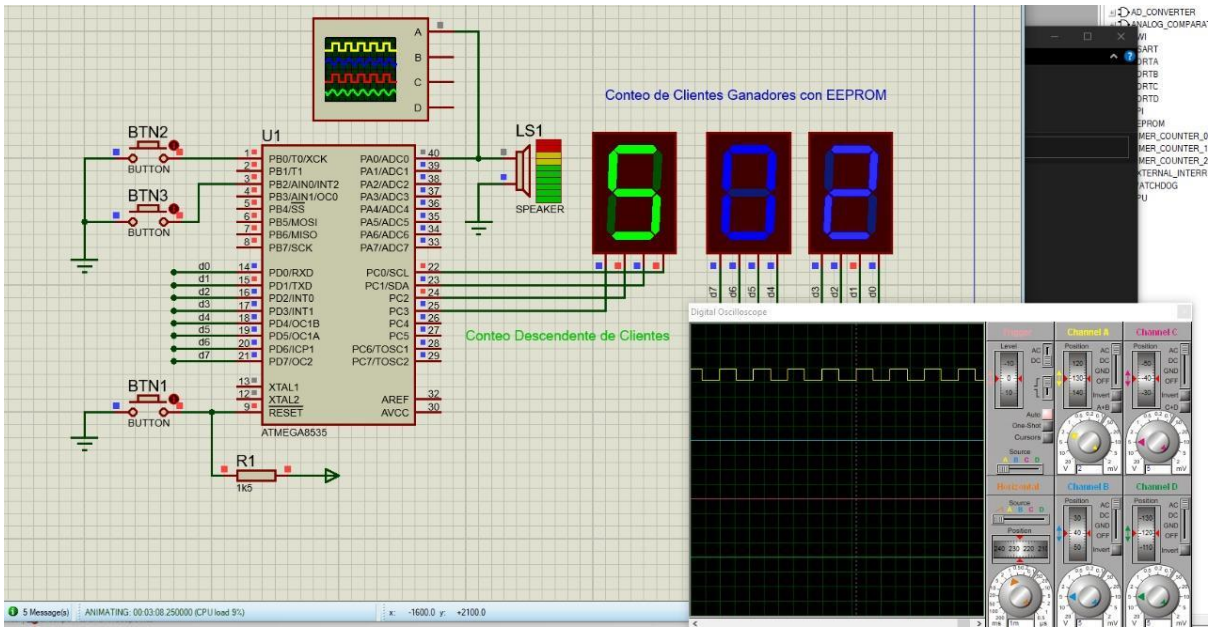


Figura 8. Oscilador con sonido

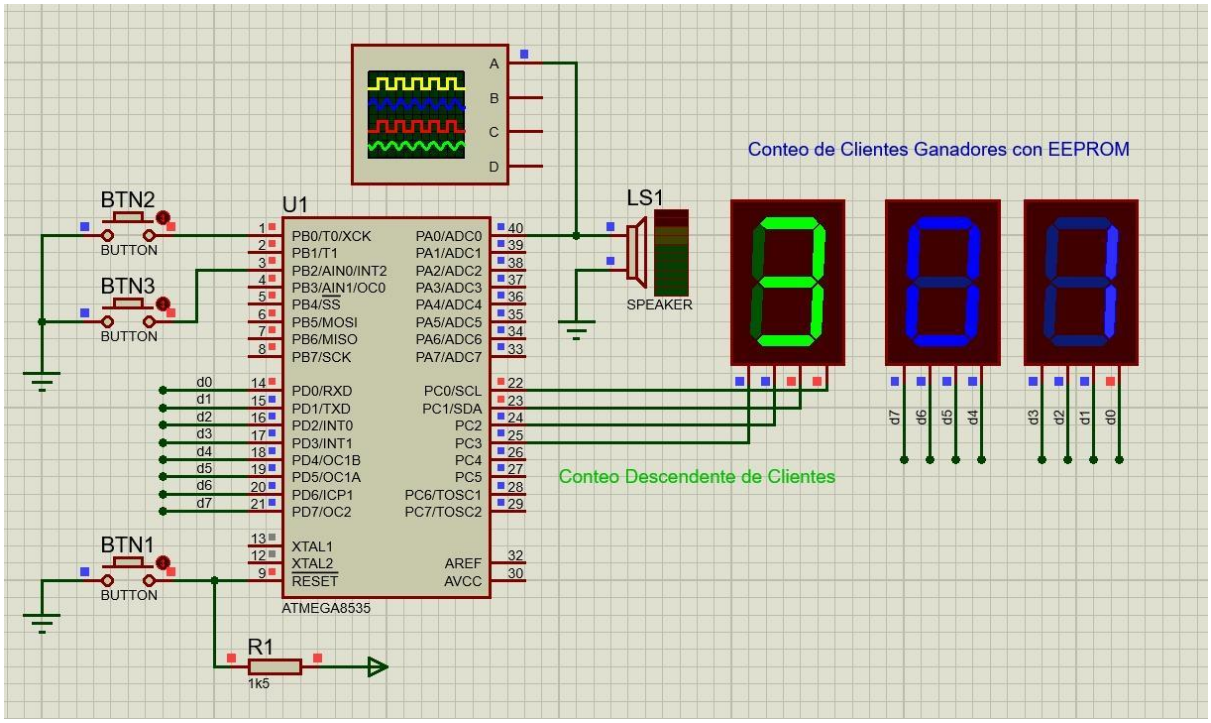


Figura 9. Antes del reset

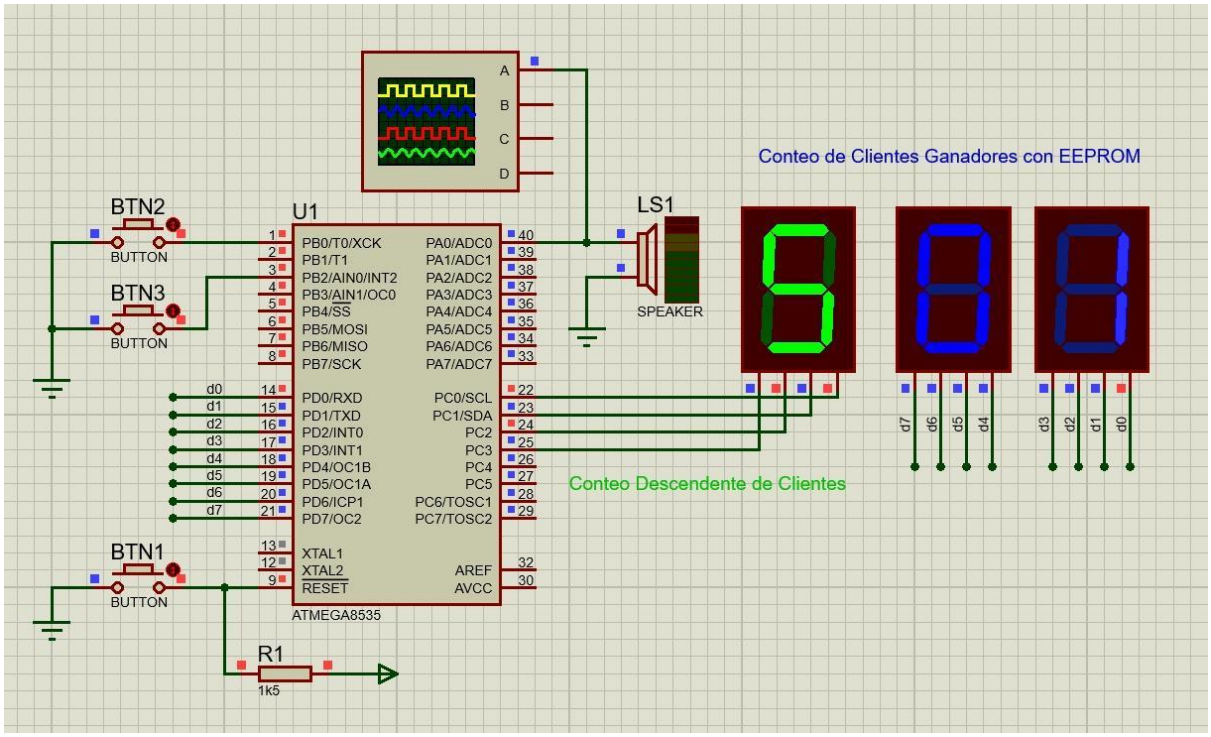


Figura 10. Después del reset

## Conclusiones Individuales

### Contreras Cardoso Adolfo

La realización de esta práctica fue poner en marcha
la traducción del lenguaje ensamblador al lenguaje c, pudimos lograr hacer funcionar esta práctica que fue echa con ensamblador pero ahora con lenguaje c, en lo personal me gustó trabajar más con este lenguaje de programación ya que es más conocido y se facilitan muchas cosas, el estudio de nuestro microcontrolador nos está dando la oportunidad para nuevos funciones del mismo y nos servirá para introducirnos para otros componen

### Martínez Alvarado Bryan Alexis

Durante el desarrollo de esta práctica se puso a prueba el funcionamiento de la memoria eeprom, macros, displays, timer, entradas físicas y salidas, aplicando lo visto durante la clase, se logró poner en práctica los conocimientos adquiridos en la parte teórica y se logró observar el funcionamiento del microcontrolador AtMega8535 junto con otros componentes electrónicos, pudiendo analizar cómo es que funcionan.

### Maya Martínes Alonso Rubén

El uso de lenguaje C facilitó el desarrollo de la práctica comparada con la misma usando asm, es interesante pensar en todas las posibilidades de programas que se pueden hacer en C facilitando el desarrollo haciéndolo más rápido. Es impresionante como con el mismo controlador podemos usar un lenguaje de alto nivel y sigue funcionando bien aunque es como un paradigma completamente nuevo.

### Pérez Gómez Santiago

El realizar la conversión de ensamblador a lenguaje C resultó un gran desafío, en especial debido al desconocimiento del manejo de la memoria EEPROM en lenguaje C, sin embargo, se consiguieron los resultados esperados, empleando los conocimientos adquiridos en las sesiones de clase, aunque fue necesario consultar en variadas ocasiones el manual de ayuda de AVR Studio.