INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 2: CONTADOR ASCENDENTE Y DESCENDENTE

ALUMNO: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 16 DE ABRIL DE 2023

OBJETIVO

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador descendente y ascendente utilizando funciones de retardo.

INTRODUCCIÓN TEÓRICA

CONTADORES

Es todo circuito o dispositivo que genera una serie de combinaciones a sus salidas sincronizadas por una señal de reloj externa. Se puede clasificar de acuerdo con:[1]

- El comportamiento con la señal de reloj: Asíncronos o síncronos.
- El formato de salida del conteo: Binario, BCD (Decimal codificado en binario) o Arbitrario.
- Sentido del conteo: Ascendente (Progresivo) o Descendente (Regresivo).

CONTADOR ASÍNCRONO

Formado en principio por Flip-Flops y lógica combinatoria adicional. Se llaman así ya que la señal eterna de reloj en general se conecta a la entrada de un solo Flip-Flop y se propaga luego internamente. La ventaja es su sencillez. Su principal desventaja es su limitada velocidad de respuesta que depende fuertemente de la cantidad de bits que maneje.[1]

CONTADOR SÍNCRONO

Formado en principio por Flip-Flops y lógica combinatoria adicional. Se llaman así ya que la señal externa de reloj en general se conecta a las entradas de reloj de todos los Flip-Flop simultáneamente. La ventaja es su mayor velocidad de respuesta respecto al asíncrono. Su relativa desventaja es su mayor complejidad en la elaboración del circuito y consumo de energía.[1]

En el diseño lógico, salvo algunas excepciones, se utilizan los contadores síncronos ya que esto permite, además de la velocidad, un mejor control en la propagación de retardos a otros dispositivos.[1]

CONTADOR ASÍNCRONO BINARIO PROGRESIVO

Dado que la señal de reloj de cada FF se obtiene de la salida del FF anterior existe una cadena de retardos en las respuestas. El primer FF es que reacciona más rápido al cambio del CLK y así sucesivamente por lo que el último FF es el que define la velocidad de respuesta del contador. A mayor número de bits, mayor retardo. Los peores casos son para los conteos en donde cambia el FF más significativo (el último).[1]

CONTADOR ASÍNCRONO BINARIO REGRESIVO

Se emplea el mismo circuito del progresivo con la diferencia que se deben de conectar cada salida /Q a la entrada de reloj del FF siguiente mientras que el conteo se sigue tomando de las salidas Q.[1]

SUBRUTINA DE RETARDO DE SOFTWARE

La mayoría de las aplicaciones se diseñan para el usuario, este requiere ver en tiempo real las componentes de control y como el microcontrolador es muy rápido (hablando con respecto al ojo humano) se necesita esta rutina básica para generar esperas y poder visualizar o ver el control que realiza el microcontrolador al elemento controlado.[2]

Una desventaja de las subrutinas de retardo de software es que el microcontrolador pierde tiempo haciendo "nada", para un mejor desempeño o control se recomienda el uso del temporizador, pero para fines prácticos muchas veces es necesario tener a mano una rutina básica como la de retardo de software.[2]

SUBRUTINA

Es una secuencia de instrucciones que se utiliza más de una vez en el programa. Cuando el CPU encuentra un llamado a subrutina, este desvía la atención del flujo del programa principal hacia otra parte del código donde ejecuta el contenido de esta subrutina; finalizada la ejecución de dicha subrutina, esta retorna en la siguiente instrucción del código donde fue invocada la subrutina. Es importante el uso de subrutinas pues acelera el proceso de desarrollo de una aplicación, es decir podemos tener bloques de código que controlan dispositivos y de estos solo hacemos un llamado.[2]

SUBRUTINA DE RETARDO

La subrutina "Delay" es capaz de generar retardos desde 1 ms hasta 65.535 seg., pero esto no asegura de que sean exactamente la cantidad de tiempo precisa, este es solo un estimado. Su llamado se realiza cargando el tiempo en el registro HX y luego invocándola.[2]

En el lenguaje de programación C podemos invocar esta subrutina llamando a la función "delay_ms(t)" o "delay_us(t)", en dónde "t" es el tiempo que se desea retardar la ejecución del programa, además de que la primera función se encarga de generar un retardo expresado en milisegundos mientras que la segunda genera retardos expresados en microsegundos. Para utilizar estas funciones en nuestros programas es necesaria la importación de la librería "delay.h" en el código fuente.

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 16 LED's
- ✓ 16 Resistores de 330 Ω a 1⁄4 W

DESARROLLO EXPERIMENTAL

1. Diseñe un contador binario ascendente en el Puerto B (0-255), y un contador descendente en el Puerto D (255-0). Use un retardo de un segundo para visualizar la información.

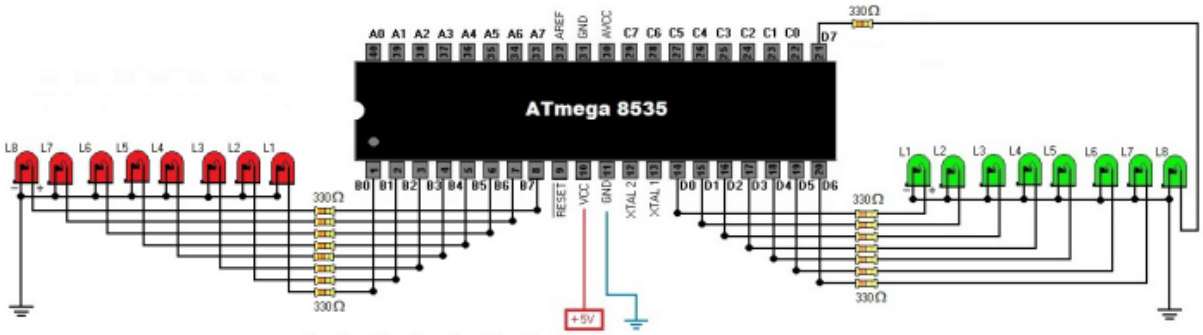


Figura 1. Circuito para el contador ascendente y descendente

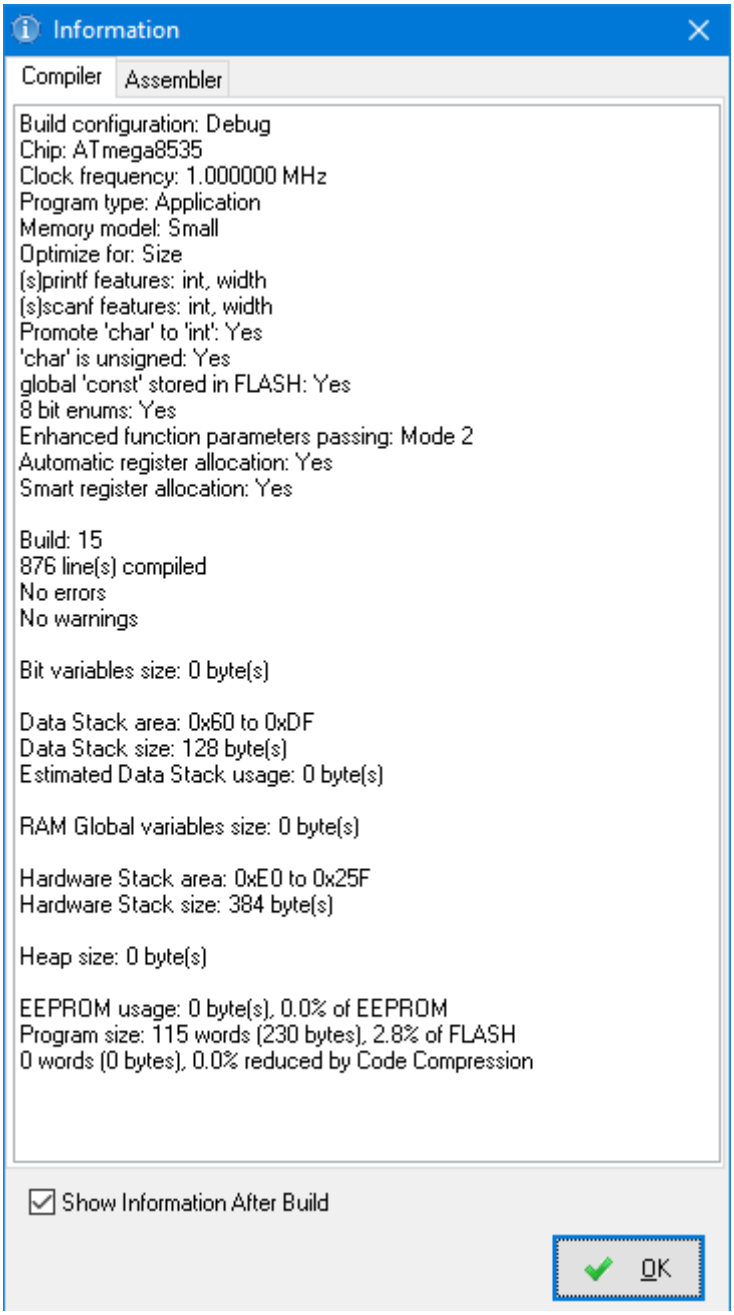


Figura 2. Compilación exitosa en CodeVisionAVR

CÓDIGO GENERADO POR CODEVISION

```
/*****************************************************
This program was created by the CodeWizardAVR V3.47
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
```

```
Date    : 06/02/2022
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

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

// Declare your global variables here

void main(void)
{
// Declare your local variables here
    int tiempo = 750; //tiempo del delay

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out
Bit1=Out Bit0=Out
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
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) |
(0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
```

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

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out
Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) |
(1<<DDD2) | (1<<DDD1) | (1<<DDD0);
// State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) |
(1<<PORTD3) | (1<<PORTD2) | (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
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
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;

// Timer/Counter 2 initialization
// Clock source: System Clock
```

```
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22)
| (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
(0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) |
(0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) |
(0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) |
(0<<CPHA) | (0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
```

```
PORTB = 0x00;            //Inicialización Puerto B = 0
PORTD = 0xFF;            //Inicialización Puerto D = 255

while (1)
      {
      // Place your code here
      delay_ms(tiempo);  //Delay para visualizar información
      PORTB ++;          //Incremento Puerto B
      PORTD --;          //Decremento Puerto D
      }
}
```

CIRCUITO ELECTRICO EN PROTEUS

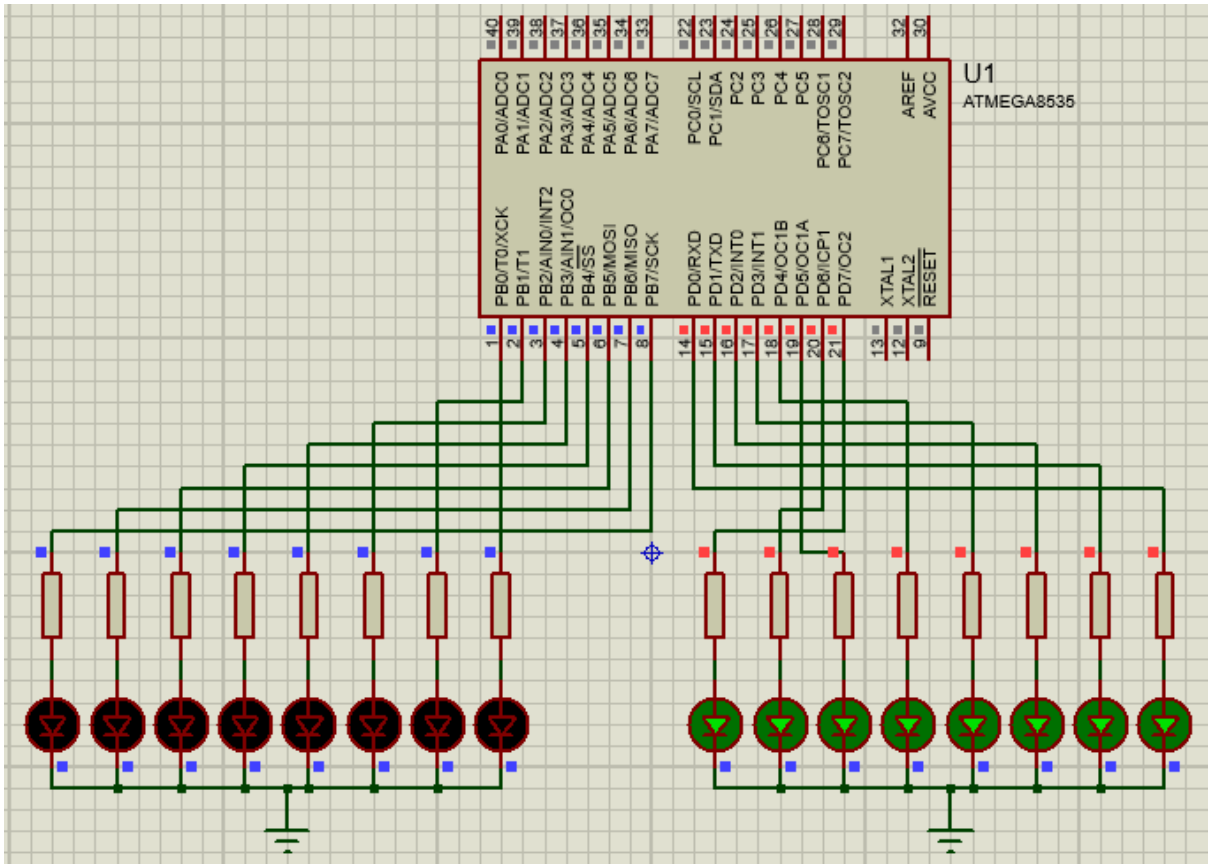


Figura 3. Circuito simulado en Proteus cuando B = 0 y D = 255

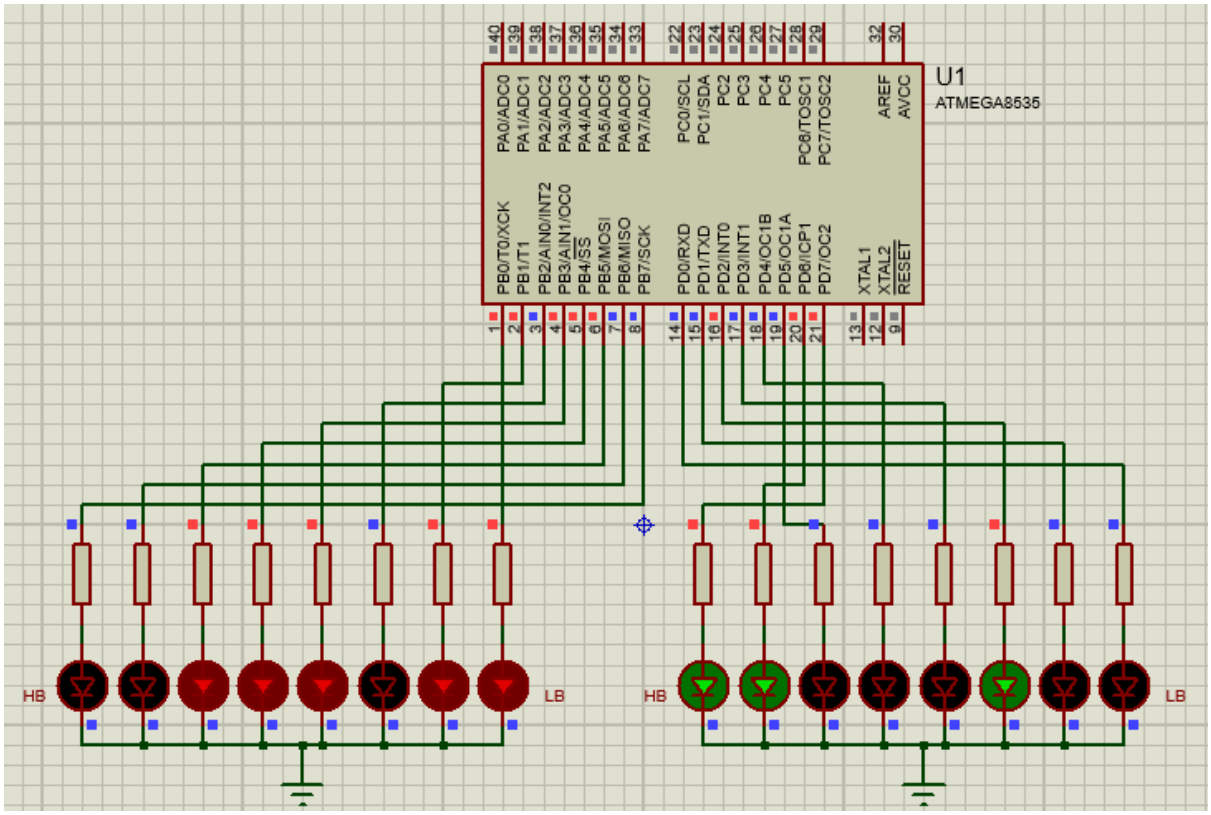
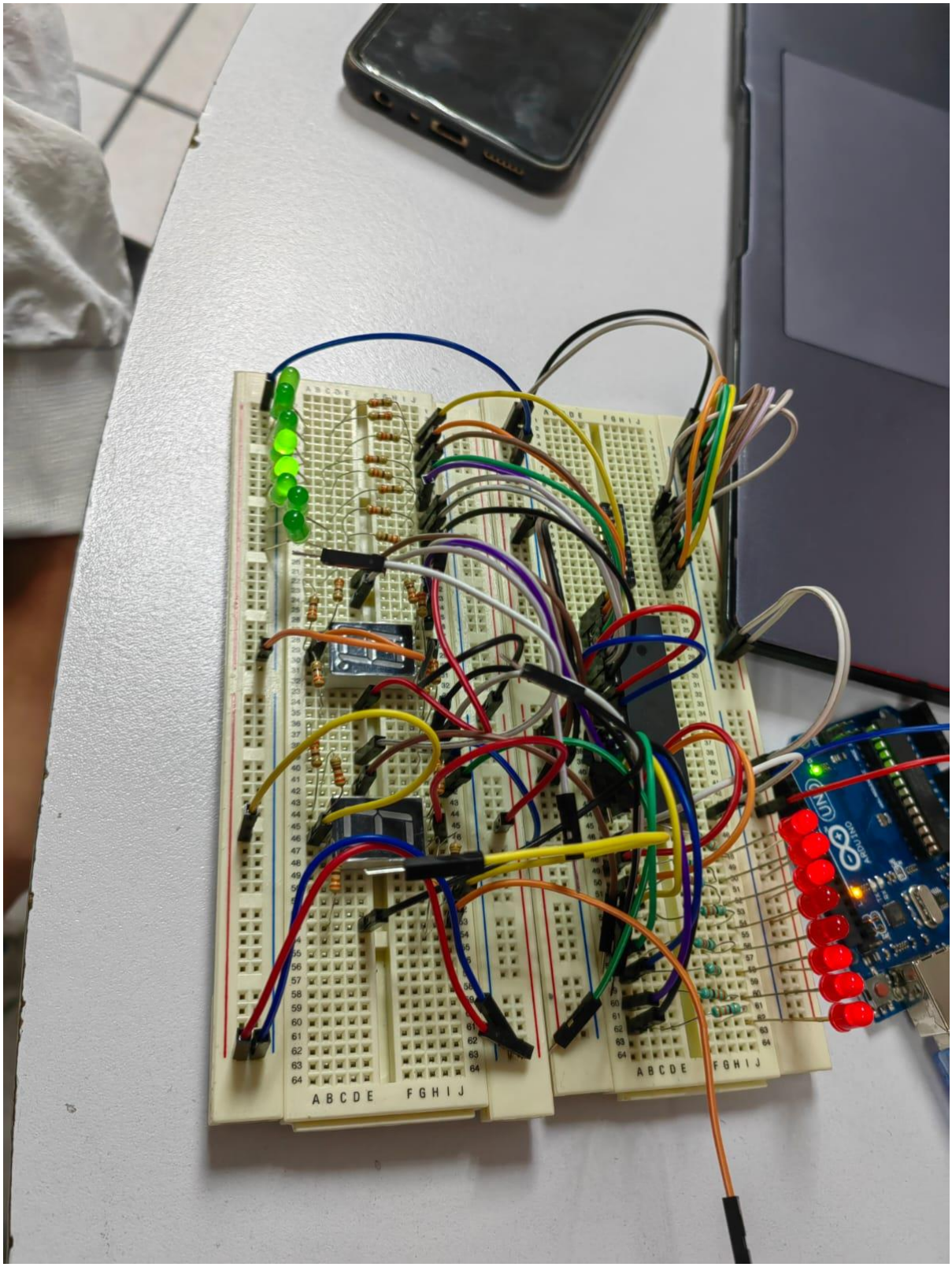


Figura 4. Circuito simulado en Proteus cuando B = 59 y D = 196

CIRCUITO EN EL PROTO ARMADO



OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrián

Los contadores binarios nos sirven para poder llevar una cuenta progresiva o regresiva según sea el caso. Podemos realizar el incremento o decremento del valor que mostraremos mediante el uso de las operaciones básicas de suma y resta. Los incrementos y decrementos suelen ser de 1 unidad en 1 unidad, sin embargo, podemos determinar por código el conteo que se desea realizar. En la realización de esta práctica pude desarrollar un contador binario ascendente para la salida del puerto B y un contador binario descendente para la salida del puerto D.

Para la realización de los contadores es muy importante el inicializar correctamente los puertos de la siguiente manera: el puerto B, al ser un contador ascendente, tiene que estar inicializado con el valor 0x00; mientras que el puerto D, al ser un contador descendente, tiene que estar inicializado con el valor 0xFF. Los puertos tienen que ser inicializados para posteriormente realizar el incremento y decremento en ambos puertos y llevar al cabo los conteos simultáneamente.

Martínez Chávez Jorge Alexis

El microcontrolador es muy rápido para que el ojo humano pueda percibir los cambios que se efectúan en los puertos de salida, por lo que fue necesaria la implementación de una subrutina de retardo de software mediante el uso de la función “delay_ms” de la librería “delay.h”, con lo que se generó un retardo de 750 ms con el fin de que se pudieran observar los números en los puertos de salida antes de realizar el incremento y decremento.

El manejo de contadores y de subrutinas de retardo de tiempo nos son de mucha utilidad para algunas aplicaciones que se pueden desarrollar con los microcontroladores.

BIBLIOGRAFÍA

[1] S. Noriega, “Contadores”, 2009. Accessed: Feb. 07, 2022. [Online]. Available: https://catedra.ing.unlp.edu.ar/electrotecnia/islyd/apuntes/Tema%205%20Contadores%202010.pdf.
[2] R. Alvarado, “Generación de subrutina de retardo de software”, 2005. Accessed: Feb. 07, 2022. [Online]. Available: https://docplayer.es/85809014-Generacion-de-subrutina-de-retardo-de.html.