INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 4: CONTADOR DE 0 A 9

ALUMNO: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 30 DE ABRIL DE 2023

OBJETIVO

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador de 0 a 9 mostrado en un display activado con un Push Button.

INTRODUCCIÓN TEÓRICA

CONSTANTES EN LENGUAJE C

En C las constantes se declaran con la directiva #define, esto significa que esa constante tendrá el mismo valor a lo largo de todo el programa. Al contrario que las variables, las constantes mantienen su valor a lo largo de todo el programa. Para indicar al compilador que se trata de una constante, usaremos la directiva #define:[1]

#define <identificador> <valor>

Observa que no se indica el punto y coma de final de sentencia ni tampoco el tipo de dato. La directiva #define no sólo nos permite sustituir un nombre por un valor numérico, sino también por una cadena de caracteres. El valor de una constante no puede ser modificado de ninguna manera.[1]

EJEMPLO USANDO #DEFINE

```
/* Uso de las constantes */

#include <stdio.h>
#define pi 3.1416
#define escribe printf
main() /* Calcula el perímetro */
{
	int r;
	escribe("Introduce el radio: ");
	scanf("%d",&r);
	escribe("El perímetro es: %f",2*pi*r);
}
```

PULSADORES (PUSH BUTTON)

Un pulsador o interruptor, es un dispositivo simple con dos posiciones, ON y OFF, un ejemplo es el interruptor de la luz.[2]

Estos pequeños pulsadores son de un 1/4 por cada lado, tienen 4 pastillas por lo que se puede pensar que hay 4 cables, pero son dos de cada lado unidos, por tanto, este pulsador es solamente de 2 cables.[2]

CARACTERÍSTICAS

- Utilizado como switch o interruptor al momento de ser presionado.
- Funciona como contacto normalmente abierto (NA).
- Infinito número de aplicaciones.
- Aguanta hasta 50 A.

- Voltaje: 120 VDC/ 220 VAC.
- Tamaño muy reducido.
- 4 pines amigables para usar en el protoboard.

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 1 Display cátodo común
- ✓ 7 Resistores de 330 Ω a 1⁄4 W
- ✓ 1 Push Button

DESARROLLO EXPERIMENTAL

1. Diseñe un programa colocando en el Puerto B un Display. Coloque un Push Button en la terminal 0 del Puerto D para incrementar su cuenta del 0 al 9.

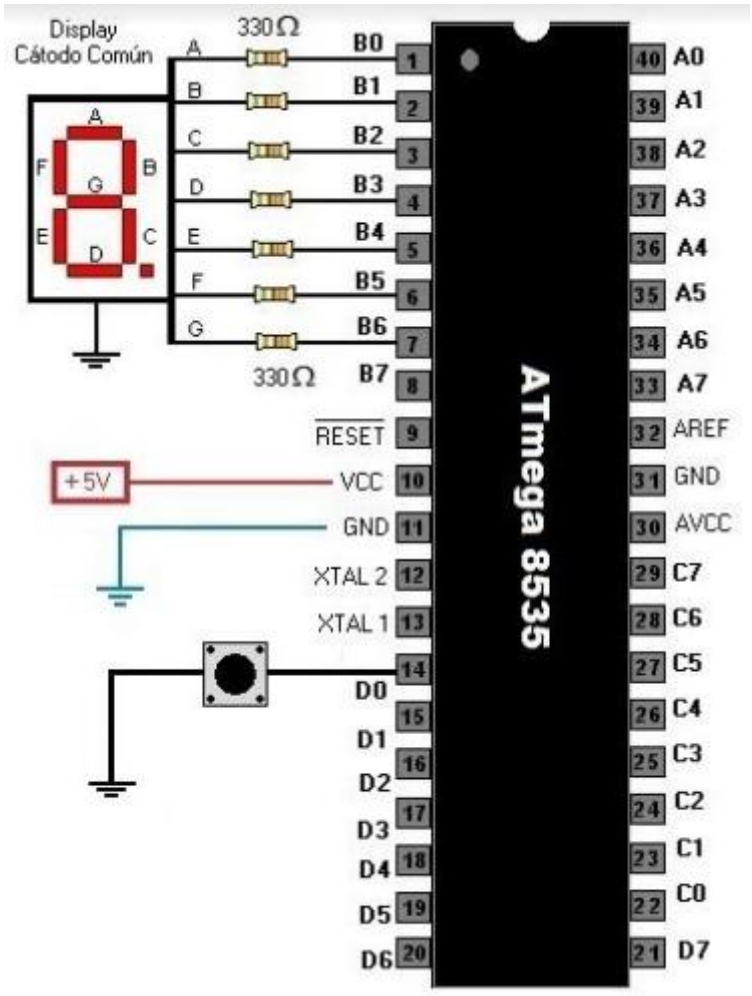


Figura 1. Circuito para el contador de 0 a 9

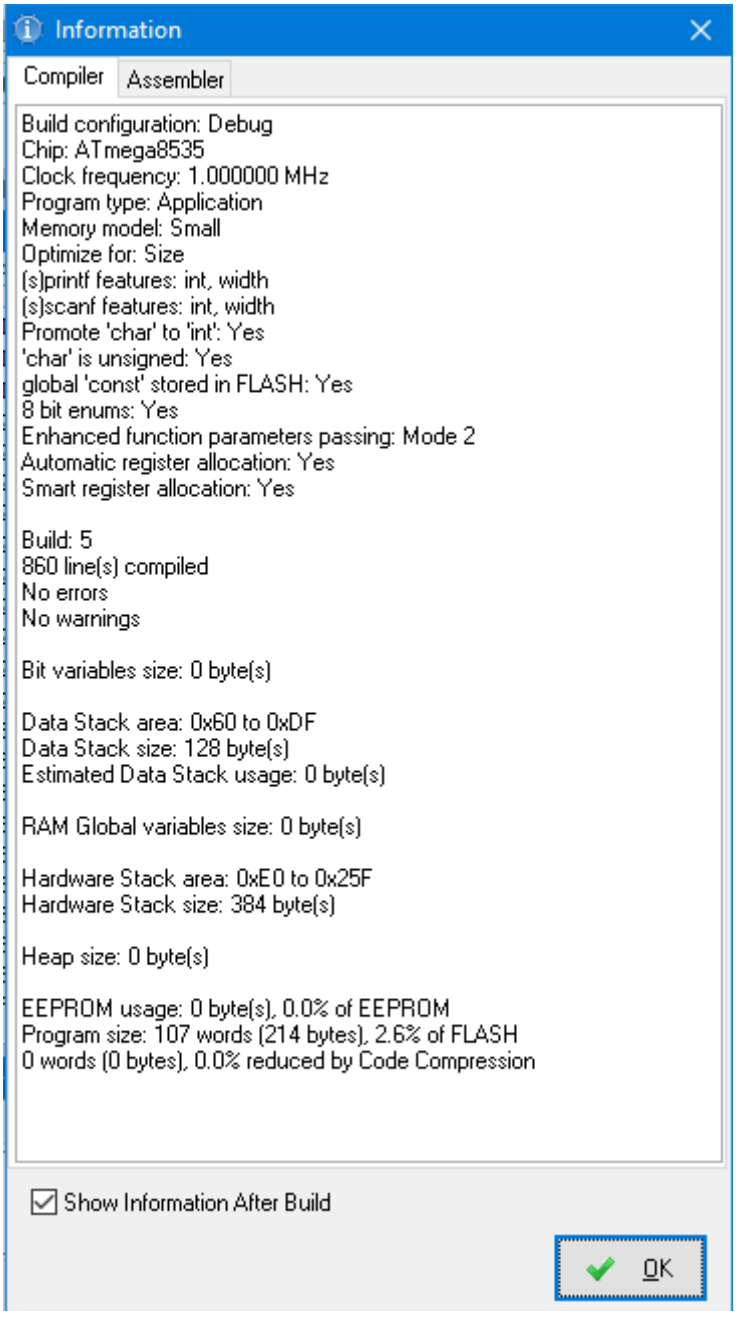


Figura 2. Compilación exitosa en CodeVision

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    :
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>
#define boton PIND.0   //Definición de un puerto mediante una
etiqueta

const char tabla7segmentos[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66,
0x6d, 0x7d, 0x07, 0x7f, 0x6f};
unsigned char var1;

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

// 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=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<<DDD1) | (0<<DDD0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
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);
```

```
while (1){
        if(boton == 0){
            var1++;
        }
        if(var1 == 10){
            var1 = 0;
        }
        PORTB = tabla7segmentos[var1];
      }
}
```

CIRCUITO ELECTRICO EN PROTEUS

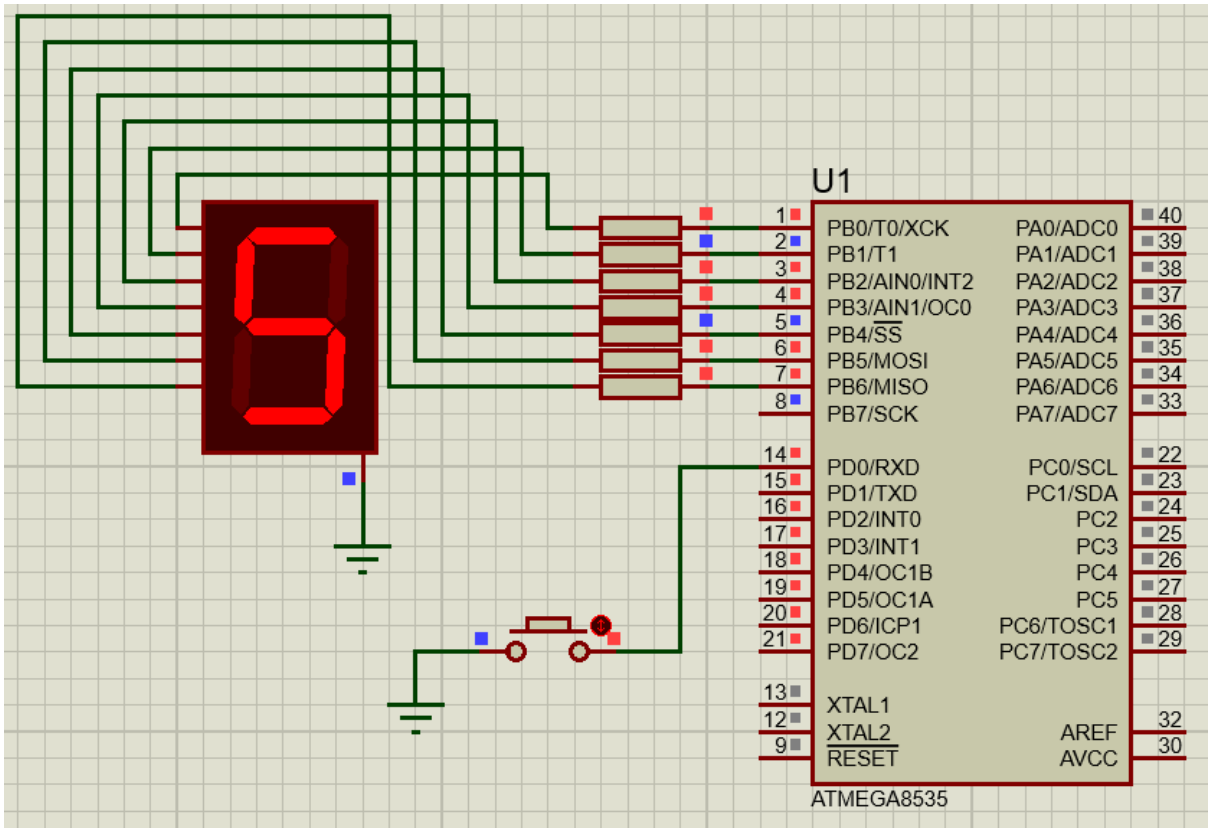
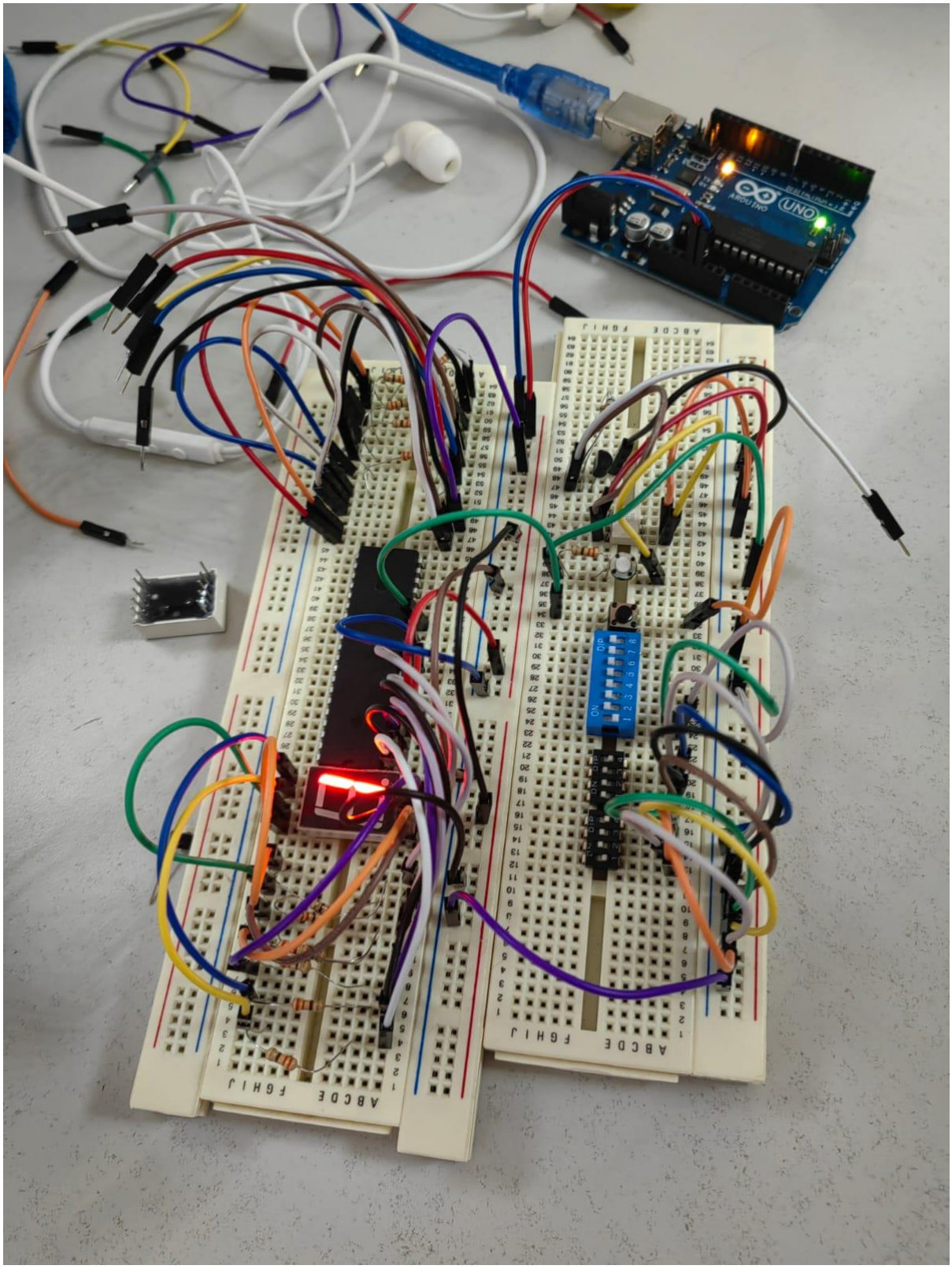


Figura 3. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO



OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrián

El uso de constantes en el lenguaje C nos sirve como un orden de valores que utilizaremos en el desarrollo de nuestros programas, de tal forma que estos valores tengan una etiqueta única y que estos puedan ser utilizados con la etiqueta en lugar de escribir todo el valor. Al ser valores constantes tenemos entendido que será un valor que nunca cambiará en el desarrollo del programa, por lo que no podremos hacerle modificaciones. Estas constantes nos sirven para definir números o incluso cadenas de caracteres. En el caso de la práctica nos sirvió para poder identificar y guardar el número de puerto en el que está conectado el push botón,

Martínez Chávez Jorge Alexis

El push botón es un elemento que nos sirve como un interruptor. Normalmente se usa como interruptor abierto, por lo que se programa un funcionamiento en específico cuando este es presionado. El push botón es un dispositivo muy pequeño pero muy útil en los circuitos electrónicos. Basta con programar el cambio de estado a través de un ciclo if en lenguaje C para poder programar el funcionamiento que queremos que tenga nuestro circuito cuando presionamos el push botón. Para nuestro contador, el push botón sirve como el interruptor que le indica al circuito cuando comenzar el conteo del 0 al 9.

Un problema que se presentó en el circuito simulado es que, al tener una frecuencia muy alta, el conteo se lleva de forma rápida, lo que hace difícil ver los cambios consecutivos del 0 al 9 como se esperaba. Este problema se puede solucionar con el uso de subrutinas de retardo de tiempo, con el fin de visualizar la información en el display de 7 segmentos antes de que cambie; sin embargo, para el desarrollo de esta práctica no se solicitó usar la subrutina de retardo de tiempo.

BIBLIOGRAFÍA

[1] D. Rodríguez and H. Adrián, “Constantes Lenguaje en C - Directiva #define y ejemplo,” *Diseño Web akus.net*, Mar. 25, 2020. https://disenowebakus.net/constantes.php (accessed Feb. 17, 2022).
[2] UAEH, “Pulsadores (Push Button) | Arduino,” *ceca.uaeh.edu.mx*, 2021. http://ceca.uaeh.edu.mx/informatica/oas_final/OA4/pulsadores_push_button.html (accessed Feb. 17, 2022).