



**Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica**



Laboratorio Microcontroladores

**Práctica N.º 04 - Visualización de números en el
Display de 7 Segmentos**

N.º de Equipo: 09

Nombre	Matrícula	NL	Brigada	Día	Hora
Gerardo Enrique Doria Villanueva	2077968	#10	315	Miércoles	V4
Carlos Adrián Guillén Benítez	2077519	#8	515	Viernes	V4
Marcelo Fabrissio Martínez Parga	1793911	#26	315	Miércoles	V4

Nombre del profesor: Jesús Daniel Garza Camarena

Semestre en curso: Agosto – Diciembre 2023

San Nicolás de los Garza, N.L.

Fecha: 20/09/2023

Objetivo

Al finalizar esta práctica, el estudiante será capaz de configurar y controlar un Display de 7 Segmentos utilizando un microcontrolador, para mostrar números en el rango de 0 a 9 y así visualizar información de forma efectiva

Introducción a la práctica: ¿Cómo funciona los display 7 segmentos?

Un display de 7 segmentos es un componente electrónico que permite visualizar dígitos y algunos caracteres del alfabeto, se utiliza mucho gracias a su simplicidad. Su nombre se debe a que está compuesto por siete diodos LED, nombrados de la A a la G, en una forma rectangular, como se observa en la Fig. 1, algunos pueden tener uno extra para mostrar un punto.

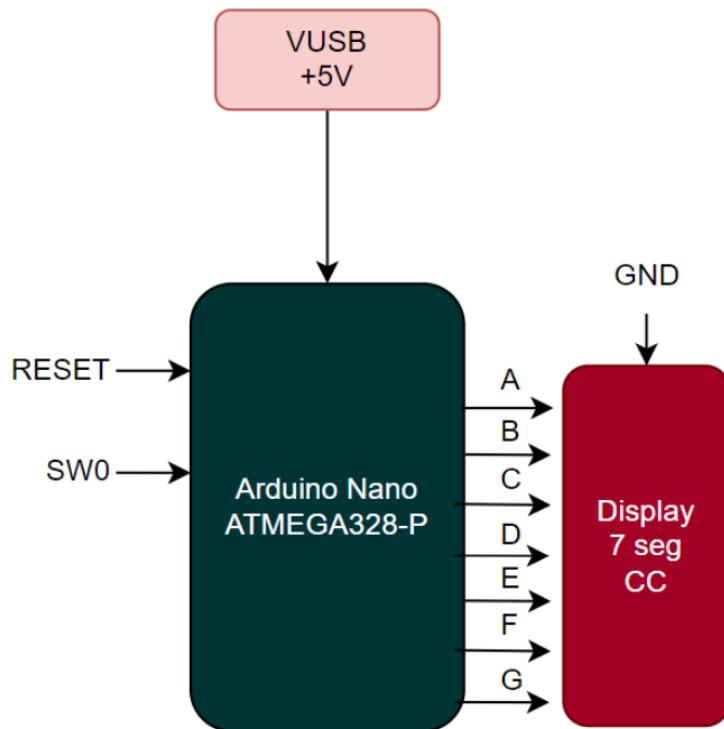
Cuentan con 10 terminales, donde 7 son para los respectivos segmentos, 1 para el punto y 2 de alimentación (ya sea VCC o GND). Los dígitos y letras se generan a partir de las combinaciones de encendido y apagado de los LEDs. Para encender un LED, se debe alimentar con voltaje o conectar a tierra, dependiendo del tipo de display; existen 2 tipos: cátodo común y ánodo común.

Cátodo común: Como se observa en la Fig. 1, sus LEDs se encuentran unidos en su terminal negativa, es decir, en el cátodo. Por lo tanto, para encender los segmentos, sus terminales se deben alimentar con voltaje o 1 lógico.

Ánodo común: Al contrario de cátodo común, sus LEDs se encuentran unidos en su terminal positiva, es decir, en el ánodo, como se observa en la Fig. 2. Para encender los segmentos, se deben conectar a tierra o 0 lógico.

Generalmente, la corriente para los LEDs debe ser de 15 mA, por lo que en un circuito digital donde VCC es 5 V, cada segmento debe ser conectado a una resistencia de 220 Ω. Existen circuitos integrados conocidos como decodificadores BCD a 7 segmentos, que controlar el display para ser contadores. Se pueden utilizar varios displays para mostrar números de varias cifras, palabras, mensajes, etc.

Imagen del diagrama de bloques



Lista de materiales utilizados (5 puntos).

- Arduino UNO
- Protoboard
- Display 7 seg. Cátodo común
- 7 resistencias 220 Ohm
- 1 Dip switch
- 1 resistencia 10k Ohm

Marco teórico (detalles de los elementos utilizados dentro de la práctica)

Un microcontrolador es un circuito integrado digital programable que cumple la función es controlar o automatizar un proceso, posee un CPU, memoria, puertos de entrada/salida, un timer y otros periféricos, y como es un sistema embebido, el usuario no puede modificarlos. El CPU lee y ejecuta las instrucciones programadas, almacenadas en la memoria Flash, los lenguajes más utilizados para programarlos son lenguaje ensamblador y C. El ATMEGA328P es un microcontrolador poderoso de bajo consumo de 8bits-AVR que cuenta con arquitectura RISC, esta implementado en algunas de las tarjetas de desarrollo Arduino. La tarjeta de desarrollo Arduino Nano está basada en el microcontrolador ATmega328P, fue desarrollada por la

compañía Arduino en Italia en el año 2008 cuenta con 14 pines digitales, 8 pines analógicos, 2 pines para reset y 6 pines de poder, se puede programar fácilmente en lenguaje C con una computadora con el programa ARDUINO IDE, cuenta con un voltaje operativo de 5 V y puede soportar corrientes de hasta 40 mA.

Una protoboard o placa de pruebas es una herramienta para como su nombre lo indica, realizar pruebas de funcionamiento de los circuitos electrónicos que creamos. La protoboard permite interconectar componentes electrónicos sin la necesidad de soldar componentes.

Un display de 7 segmentos está compuesto internamente por 7 LEDs, cada uno representa un segmento nombrado de la A a la G. Un LED o diodo emisor de luz funciona a partir de un semiconductor que, al ser atravesado por un voltaje, emite luz gracias al efecto de electroluminiscencia. En el display de cátodo común, sus pines de A-G deben conectarse a una fuente de voltaje, y tiene 2 pines más que deben conectarse a tierra, debido a que sus LEDs están unidos en estas terminales. Su función es poder mostrar los dígitos o letras, con las posibles combinaciones de encendido de sus segmentos. Un resistor es un componente electrónico que limita el flujo de electrones que conforman la corriente eléctrica, son utilizados para evitar la sobrecarga de corriente en otros componentes. Un push button, es un interruptor diseñado para cerrar o abrir un circuito eléctrico cuando se presiona su botón o una perilla, y para volver a su posición normal cuando se suelta; dependiendo si es normalmente abierto o normalmente cerrado. Por lo tanto, esto se puede utilizar para cambiar el modo operativo o la función de un dispositivo electrónico. Se aplican, entre otras cosas, en placas de circuito impreso o protoboard y se utilizan para configurar ciertos ajustes básicos. Los interruptores producen un efecto de rebote en las entradas del microcontrolador.

Imagen del diagrama esquemático del sistema diseñado en un software EDA, con los nombres de los integrantes del equipo

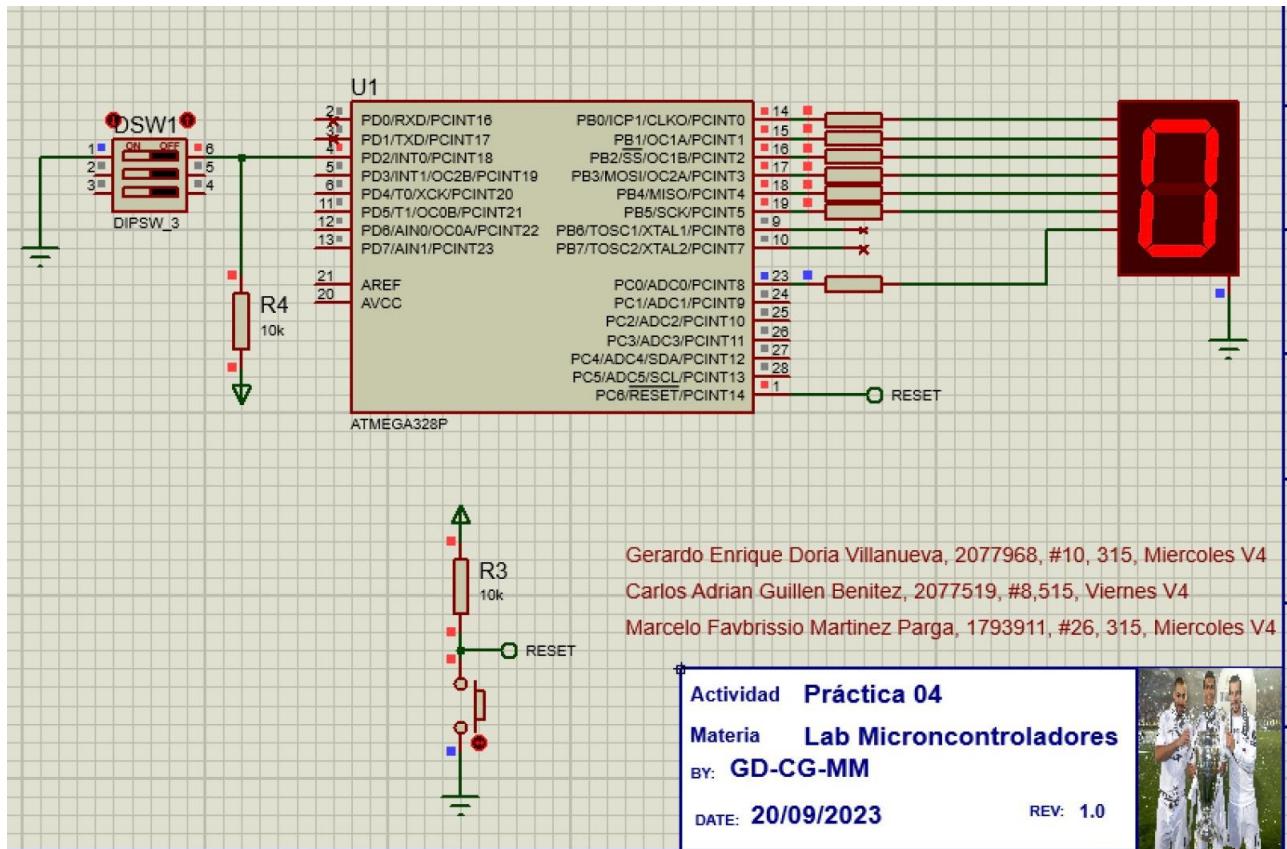
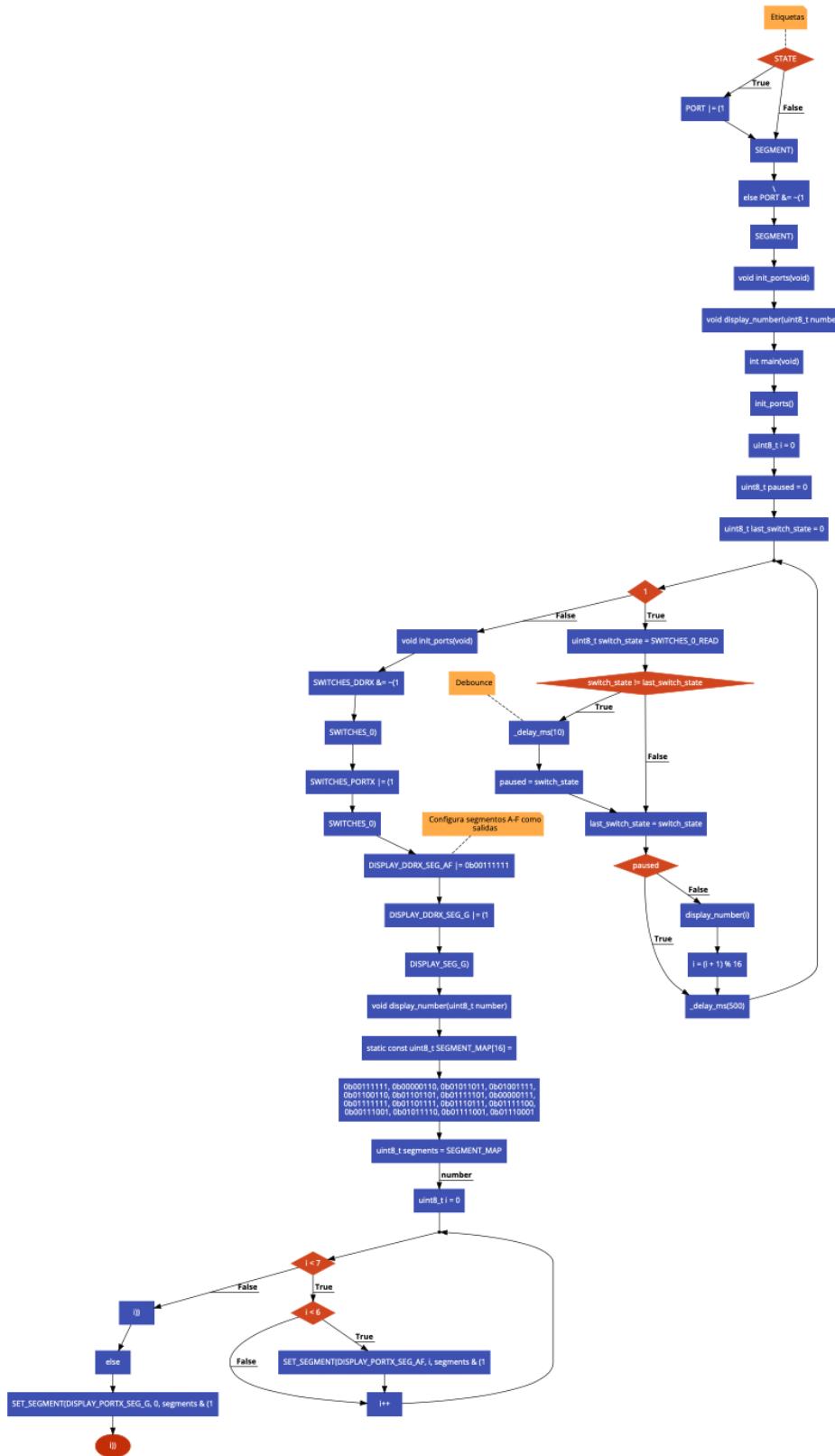


Imagen del diagrama de flujo del programa



Código en lenguaje C o ASM utilizado en la práctica, copiado y pegado directamente del IDE

```
//-----
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
//-----
// Etiquetas
// Inputs
#define SWITCHES_DDRX DDRD
#define SWITCHES_PORTX PORTD
#define SWITCHES_PINX PIND

// Dip Switch
#define SWITCHES_0 PIND2
#define SWITCHES_0_READ !(SWITCHES_PINX & (1 << SWITCHES_0))

// Outputs
#define DISPLAY_DDRX_SEG_AF DDRB
#define DISPLAY_PORTX_SEG_AF PORTB
#define DISPLAY_DDRX_SEG_G DDRC
#define DISPLAY_PORTX_SEG_G PORTC

//Segmento
#define DISPLAY_SEG_A PORTB0
#define DISPLAY_SEG_A_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_A)
#define DISPLAY_SEG_A_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_A)
//Segmento
#define DISPLAY_SEG_B PORTB1
#define DISPLAY_SEG_B_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_B)
#define DISPLAY_SEG_B_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_B)
//Segmento
#define DISPLAY_SEG_C PORTB2
#define DISPLAY_SEG_C_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_C)
#define DISPLAY_SEG_C_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_C)
//Segmento
#define DISPLAY_SEG_D PORTB3
#define DISPLAY_SEG_D_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_D)
#define DISPLAY_SEG_D_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_D)
//Segmento
#define DISPLAY_SEG_E PORTB4
#define DISPLAY_SEG_E_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_E)
#define DISPLAY_SEG_E_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_E)
//Segmento
#define DISPLAY_SEG_F PORTB5
#define DISPLAY_SEG_F_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_F)
```

```

#define DISPLAY_SEG_F_OFF DISPLAY_PORTX_SEG_AF &=~(1 << DISPLAY_SEG_F)
//Segmento
#define DISPLAY_SEG_G PORTC0
#define DISPLAY_SEG_G_ON DISPLAY_PORTX_SEG_G |= (1 << DISPLAY_SEG_G)
#define DISPLAY_SEG_G_OFF DISPLAY_PORTX_SEG_G &=~(1 << DISPLAY_SEG_G)
//-----
void init_ports(void);
void display_number(uint8_t number);

void init_ports(void);
void display_number(uint8_t number);

int main(void)
{
    init_ports();
    uint8_t i = 0;
    uint8_t paused = 0;
    uint8_t last_switch_state = 0;

    while (1)
    {
        uint8_t switch_state = SWITCHES_0_READ;

        if (switch_state != last_switch_state)
        {
            _delay_ms(10); // Debounce

            if (switch_state == 0) // Si cambió a "off," inicia/reanuda el conteo
            {
                paused = 0;
            }
            else // Si cambió a "on," pausa el conteo
            {
                paused = 1;
            }
        }

        last_switch_state = switch_state;

        if (!paused)
        {
            display_number(i);
            i++;
        }

        if (i > 15)

```

```

{
    i = 0; // Reinicia el conteo si llega al final
}
}

{
    _delay_ms(500);
}
}

//-----
void init_ports(void)
{
    //Inputs
    SWITCHES_DDRX &= ~(1 << SWITCHES_0);
    //Pull-up
    SWITCHES_PORTX |= (1 << SWITCHES_0);

    //Outputs
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_A);
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_B);
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_C);
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_D);
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_E);
    DISPLAY_DDRX_SEG_AF |= (1 << DISPLAY_SEG_F);
    DISPLAY_DDRX_SEG_G |= (1 << DISPLAY_SEG_G);

}

//-----
void display_number(uint8_t number)
{
    switch (number)
    {
        case 0: //Número 0
            DISPLAY_SEG_A_ON;
            DISPLAY_SEG_B_ON;
            DISPLAY_SEG_C_ON;
            DISPLAY_SEG_D_ON;
            DISPLAY_SEG_E_ON;
            DISPLAY_SEG_F_ON;
            DISPLAY_SEG_G_OFF;
            break;
        case 1: //Número 1
            DISPLAY_SEG_A_OFF;
            DISPLAY_SEG_B_ON;
            DISPLAY_SEG_C_ON;
            DISPLAY_SEG_D_OFF;
    }
}

```

```
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_OFF;
break;
case 2: //Número 2
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;
case 3: //Número 3
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;
case 4: //Número 4
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 5: //Número 5
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 6: //Número 6
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
```

```
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 7: //Número 7
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_OFF;
break;
case 8: //Número 8
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 9: //Número 9
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 10: //LETRA A
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 11: //LETRA B
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
```

```
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;

case 12: //LETRA C
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_OFF;
break;

case 13: //LETRA D
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;

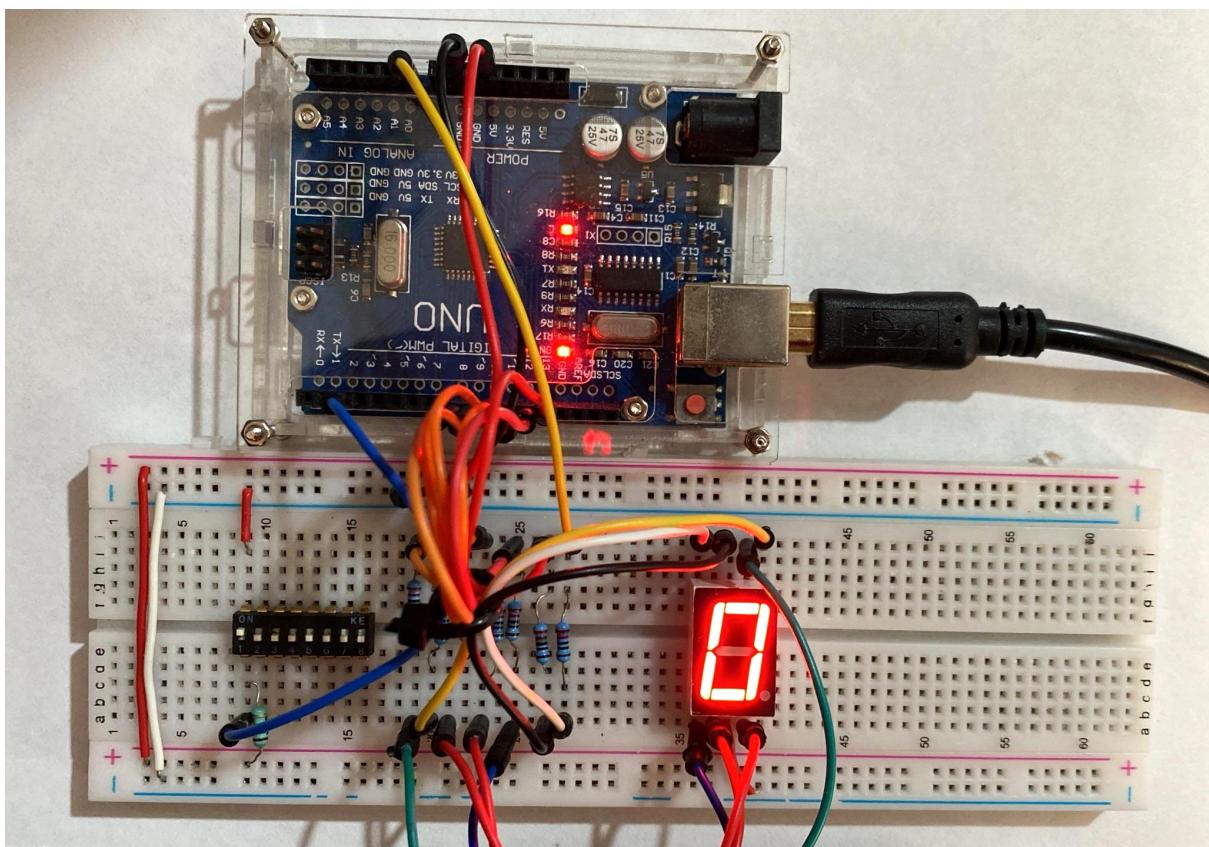
case 14: //LETRA E
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;

case 15: //LETRA F
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
}
}
```

Simulación de la actividad en un video en formato .mp4, proporcionando un archivo aparte en Tinkercad, SimulIDE o Proteus VSM

ANEXADO EN TEAMS

Fotografía del prototipo armado, mostrando solo el área útil y permitiendo el uso de jumpers solo de la tarjeta de desarrollo al protoboard



Cuestionario de la práctica contestado

1. ¿Cómo funciona la multiplexación de display?

La multiplexación para displays se utiliza para conectar varios de ellos sin la necesidad de un gran número de puertos. Varios displays se conectan en paralelo, utilizando un oscilador, se prende intermitentemente los segmentos necesarios para mostrar el mismo número en todos los displays, pero solamente se conecta el ánodo o cátodo del display donde realmente se quiere mostrar ese dígito, de esta manera se selecciona solo 1, y en el resto el dígito no se observa. Como esto ocurre a frecuencias muy altas, el ojo humano no puede detectar la intermitencia, y percibe los displays como si estuvieran todo el tiempo encendidos, cada uno mostrando dígitos distintos.

2. ¿Qué diferencia existe entre colocar una sola resistencia al pin común a conectar una resistencia a cada segmento?

Al colocar una sola resistencia, al encender varios segmentos del display, se pierde luminosidad en este porque la corriente se limita, mientras que, si se conecta una resistencia por segmento, este problema no ocurre. Además, teóricamente todos los LEDs son iguales y la corriente debería distribuirse equitativamente, pero en la práctica, cada segmento se iluminaría con distinta intensidad porque sus resistencias internas no son iguales.

Conclusiones por cada integrante (obligatorio)

Gerardo Enrique Doria Villanueva:

En esta práctica aprendimos como controlar un display utilizando el microcontrolador. Seguimos practicando como declarar sentencias lógicas en el lenguaje C, esto para poder realizar la secuencia ascendente y la pausa dependiendo del estado del push button. También aprendimos a utilizar la estructura switch, para definir que segmentos deberían estar prendidos para cada dígito o letra. Igual continuamos mejorando nuestra manera de optimizar el código, utilizando funciones, macros y etiquetas.

Carlos Adrian Guillen Benitez:

Gracias a esta práctica pudimos aprender a programar un display de 7 segmentos utilizando lenguaje C en combinación con la placa de desarrollo Arduino NANO, además pudimos reafirmar lo aprendido anteriormente como son las funciones, las etiquetas y las máscaras.

Marcelo Fabrissio Martinez Parga:

En esta práctica, profundizamos en el control de un display de 7 segmentos mediante el uso del microcontrolador y la placa de desarrollo Arduino NANO. A través del lenguaje C, reforzamos nuestro conocimiento en la declaración de sentencias lógicas, lo que nos permitió implementar una secuencia ascendente y pausarla según el estado del push button. Además, empleamos la estructura switch para determinar qué segmentos debían iluminarse para cada dígito o letra. Esta

experiencia no solo consolidó nuestra comprensión de conceptos previamente aprendidos, como funciones, etiquetas y máscaras, sino que también mejoró nuestra habilidad para optimizar el código.

Referencias Bibliográficas en formato APA (mencionar los datasheet utilizados, libros, páginas web, etc.) (obligatorio)

- Arduino. (s. f.). Arduino® Nano Product Reference Manual. Arduino Docs. <https://docs.arduino.cc/static/3d8dd6c23e960a38d9df35d0afd5765b/A000005-datasheet.pdf>.
- MICROCHIP Technology. (2019). megaAVR® Data Sheet. Microchip <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/Product/Documents/DataSheets/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf>
- Ingeniería Mecafenix. (2018). Display de 7 segmentos (como se usa). <https://www.ingmecafenix.com/electronica/display-de-7-segmentos/>