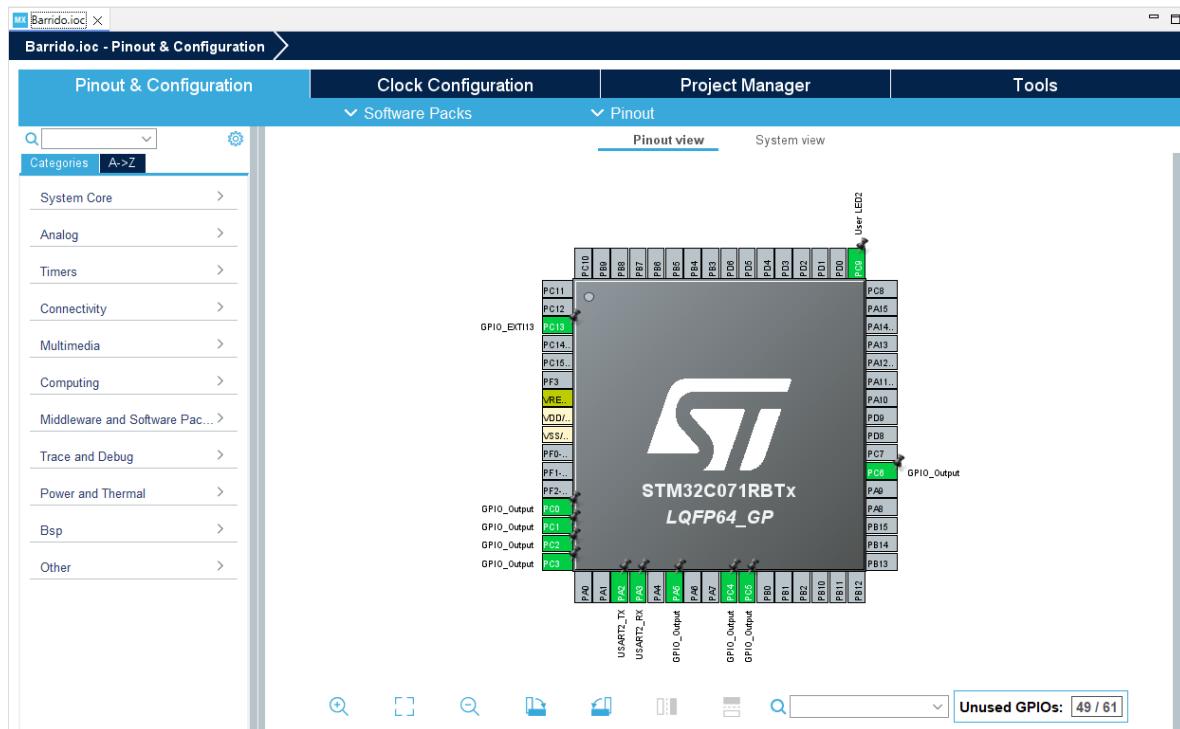


BARRIDO

CONFIGURACION INICIAL



Actividad Fundamental 2.- Barrido en display

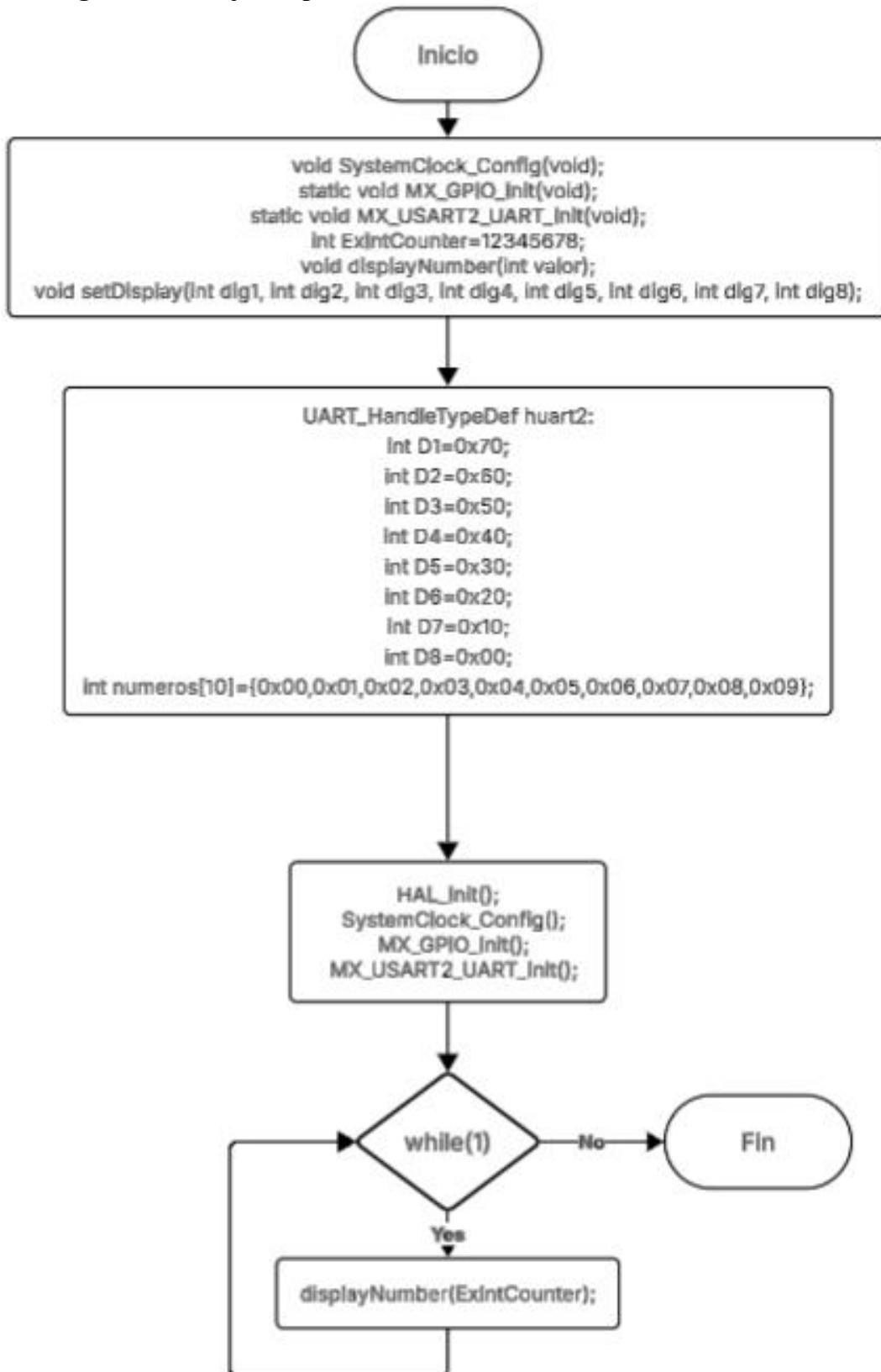
1.- Descripción del problema

Desarrollar un Código que permita escribir en un display de 8 dígitos un numero de 0 a 99999999.

2.- Objetivos

1.- El código del puerto C deberá gestionar el funcionamiento de un 74hc238 y un CI74xx47 de tal forma que generen el fenómeno de persistencia visual en un display de 8 dígitos

3.- Diagrama de Flujo Propuesto



```
displayNumber(ExIntCounter);
```

```
int dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8;  
dig8=valor%10;  
dig7=(valor%100)/10;  
dig6=(valor%1000)/100;  
dig5=(valor%10000)/1000;  
dig4=(valor%100000)/10000;  
dig3=(valor%1000000)/100000;  
dig2=(valor%10000000)/1000000;  
dig1=(valor%100000000)/10000000;
```

```
setDisplay(dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8);
```

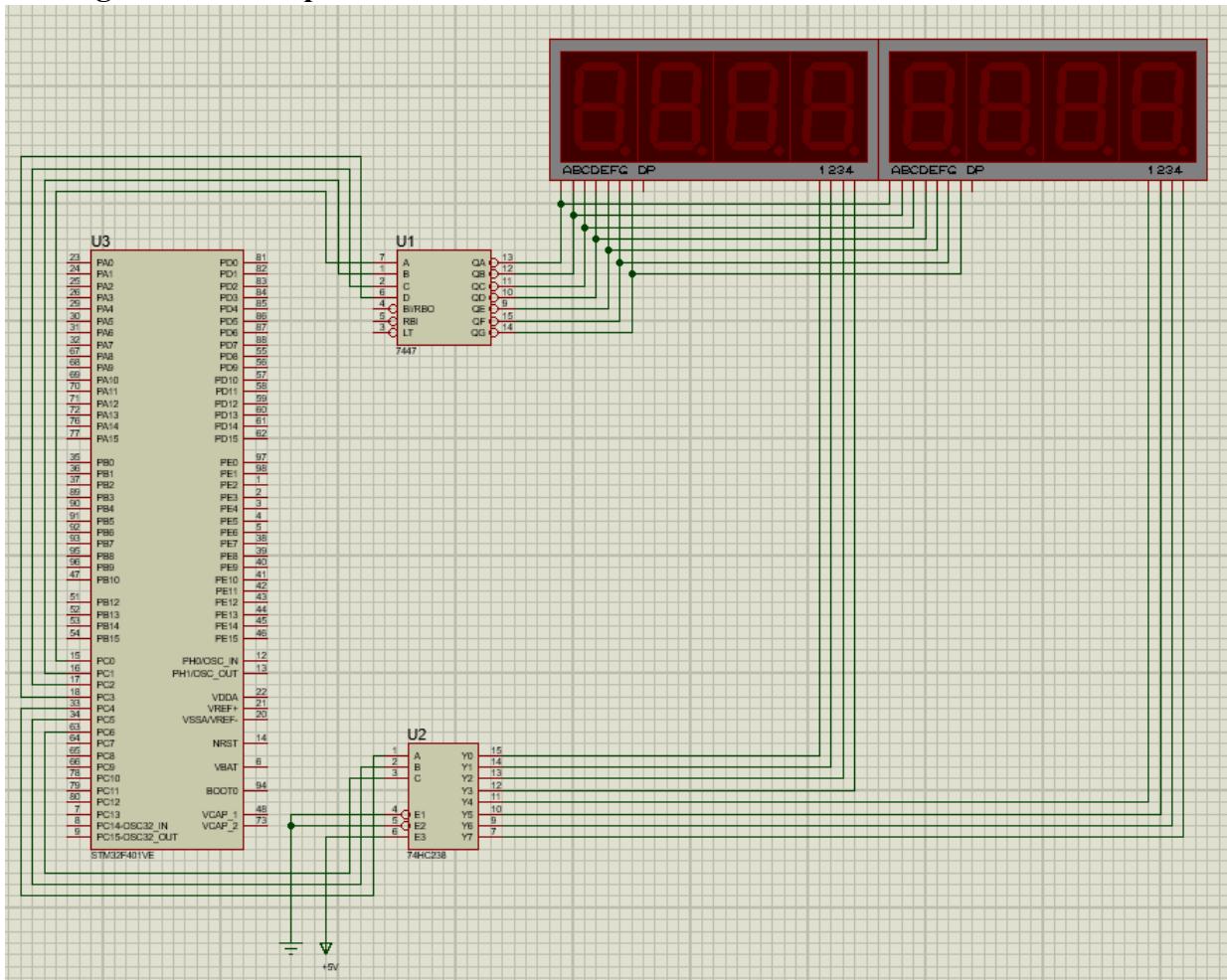
```
Fin
```

```
setDisplay(int dig1, int dig2, int dig3, int dig4, int dig5,int dig6,int dig7,int dig8)
```

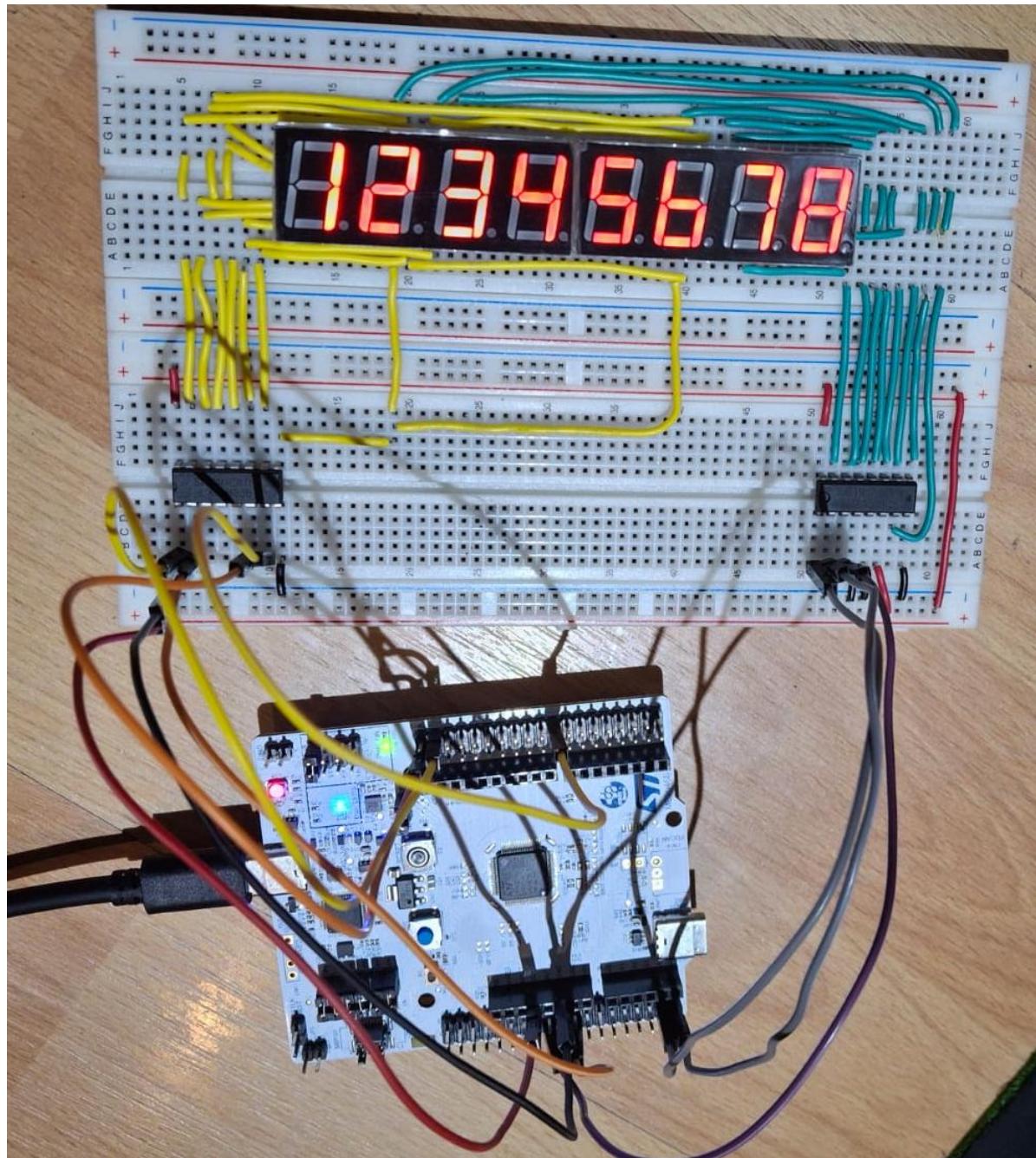
```
    GPIOC->ODR=D8+numeros[dig8];
    HAL_Delay(1);
    GPIOC->ODR=D7+numeros[dig7];
    HAL_Delay(1);
    GPIOC->ODR=D6+numeros[dig6];
    HAL_Delay(1);
    GPIOC->ODR=D5+numeros[dig5];
    HAL_Delay(1);
    GPIOC->ODR=D4+numeros[dig4];
    HAL_Delay(1);
    GPIOC->ODR=D3+numeros[dig3];
    HAL_Delay(1);
    GPIOC->ODR=D2+numeros[dig2];
    HAL_Delay(1);
    GPIOC->ODR=D1+numeros[dig1];
    HAL_Delay(1);
```

```
Fin
```

4.- Diagrama de la Implementación del Circuito



5.- Evidencia Grafica



6.- Programación del Microcontrolador

Prototipos utilizados y su descripción

En la línea 1 está el prototipo de la función que configura las velocidades de trabajo del reloj del sistema.

En la línea 2 se configuran las terminales GPIO utilizadas como entradas o salidas.

En la línea 3 se declara el prototipo de la función que permite mostrar un numero en un display de siete segmentos.

En la línea 4 se define el prototipo de la función que actualiza los valores de cada digito del display de siete segmentos.

1	void SystemClock_Config(void);
2	static void MX_GPIO_Init(void);
3	void displayNumber(int valor);
4	void setDisplay(int dig1, int dig2, int dig3, int dig4, int dig5, int dig6, int dig7, int dig8);

Llamadas utilizadas y su descripción.

En la línea 1, la función displayNumber(12345678); toma el número 12345678 como entrada y lo descompone en sus dígitos individuales, desde las unidades hasta las decenas de millones. Luego, pasa esos dígitos a la función setDisplay() para mostrar cada uno en un display de 7 segmentos, asignando el valor adecuado para cada posición del número.

En la línea 2, la función setDisplay(dig1, dig2, dig3, dig4, dig5, dig6, dig7, dig8); recibe los dígitos descompuestos de un número y los utiliza para actualizar los pines GPIO correspondientes, mostrando cada dígito en un display de 7 segmentos. Cada valor de los dígitos se traduce en el encendido adecuado de los segmentos para visualizar el número completo

1	displayNumber(12345678);
2	setDisplay(dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8);

Llamadas a función

En la línea 1 HAL_Init(); inicializa la capa de abstracción de hardware (HAL) de STM32, preparando el sistema para interactuar con el hardware del microcontrolador. Esta función configura los controladores básicos, como el reloj del sistema, las interrupciones y otros recursos esenciales, asegurando que el entorno de desarrollo esté listo para ejecutar aplicaciones.

En la línea 2, SystemClock_Config(); configura el reloj del sistema del microcontrolador, ajustando las fuentes de reloj, los multiplicadores y divisores necesarios para establecer la frecuencia de operación deseada. Esta función es esencial para asegurar que el microcontrolador funcione a la velocidad adecuada para las aplicaciones que se ejecutarán.

En la línea 3, MX_GPIO_Init(); inicializa los pines GPIO (General Purpose Input/Output) del microcontrolador, configurando sus modos de operación (entrada, salida, alternativo, analógico) y estableciendo parámetros como la velocidad y el estado inicial. Esta función es fundamental para preparar los pines de E/S para su uso en la aplicación.

1	HAL_Init();
2	SystemClock_Config();
3	MX_GPIO_Init();

Funcionamiento y estructura del loop principal

En la línea 1 se ejecuta un condicional o bucle que en este caso es un While, su función es entrar a un ciclo interminable.

En la línea 3 ejecuta la llamada que obtiene los números 12345678 como argumento. Esta función descompone el numero en sus dígitos individuales, desde las unidades hasta las decenas de millones y luego pasa estos dígitos a la función setDisplay() para mostrar cada uno de ellos en los displays

1	While(1)
2	{
3	displayNumber(12345678);
4	}

Desarrollo y descripción de las funciones utilizadas dentro del loop de control principal.

displayNumber(12345678)

En la línea 1 esta el prototipo de la función del displayNumber.

En la línea 3 se declaran las variables privadas de que serán utilizadas en la función.

En la línea 4 utiliza la función del módulo para separar el dígito de las unidades como se muestra en el ejemplo

123456789%10=8

En la línea 5 se efectúa la operación del modulo para separar la decena y la división para eliminar las unidades como se muestra en el ejemplo.

123456789%100=78 $78/10=7$ dig7=7

En la línea 6 se realiza la operación para obtener el dígito de las centenas.

En la línea 7 se realiza la operación para obtener el dígito los millares.

En la línea 8 se realiza la operación para obtener el dígito de las decenas de millar.

En la línea 9 se obtiene el valor de las centenas de millar.

En la línea 10 se obtiene el valor de los millones.

En la línea 11 se obtiene el valor de las decenas de millón.

En la línea 12 se llama a la función setDisplay para mostrar los valores extraídos en el display.

1	void displayNumber(int valor)
2	{
3	int dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8;
4	dig8=valor%10; //Unidades (8)
5	dig7=(valor%100)/10; //Decenas (7)
6	dig6=(valor%1000)/100; //Centenas (6)
7	dig5=(valor%10000)/1000; //millares a (5)
8	dig4=(valor%100000)/10000;
9	dig3=(valor%1000000)/100000;
10	dig2=(valor%10000000)/1000000;

11	dig1=(valor%100000000)/10000000;
12	setDisplay(dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8);
13	}

La función setDisplay es una función privada que se va a encargar de concatenar el valor del digito a escribirse con el ánodo correspondiente del display.

```
setDisplay(dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8);
```

En la linea 3 se concatena el valor de D8=0x00 con dig8=0x08 obteniendo 0x08

En la línea 5 se concatena el valor de D7=0x10 con dig7=0x07 obteniendo 0x18

En la linea 7 se concatena el valor de D6=0x20 con dig6=0x06 obteniendo 0x26

En la línea 9 se concatena el valor de D5=0x30 con dig5=0x05 obteniendo 0x35

En la linea 11 se concatena el valor de D4=0x40 con dig4=0x04 obteniendo 0x44

En la línea 13 se concatena el valor de D3=0x50 con dig3=0x03 obteniendo 0x53

En la linea 15 se concatena el valor de D2=0x60 con dig2=0x02 obteniendo 0x62

En la línea 17 se concatena el valor de D1=0x70 con dig1=0x01 obteniendo 0x71

1	void setDisplay(int dig1, int dig2, int dig3, int dig4, int dig5,int dig6,int dig7,int dig8)
2	{
3	GPIOD->ODR=D8+numeros[dig8];// <u>Unidades</u>
4	HAL_Delay(1);
5	GPIOD->ODR=D7+numeros[dig7];// <u>Decenas</u>
6	HAL_Delay(1);
7	GPIOD->ODR=D6+numeros[dig6];
8	HAL_Delay(1);

9	GPIOD->ODR=D5+numeros[dig5];
10	HAL_Delay(1);
11	GPIOD->ODR=D4+numeros[dig4];
12	HAL_Delay(1);
13	GPIOD->ODR=D3+numeros[dig3];
14	HAL_Delay(1);
15	GPIOD->ODR=D2+numeros[dig2];
16	HAL_Delay(1);
17	GPIOD->ODR=D1+numeros[dig1]; // <u>Decenas de Millones</u>
18	HAL_Delay(1);