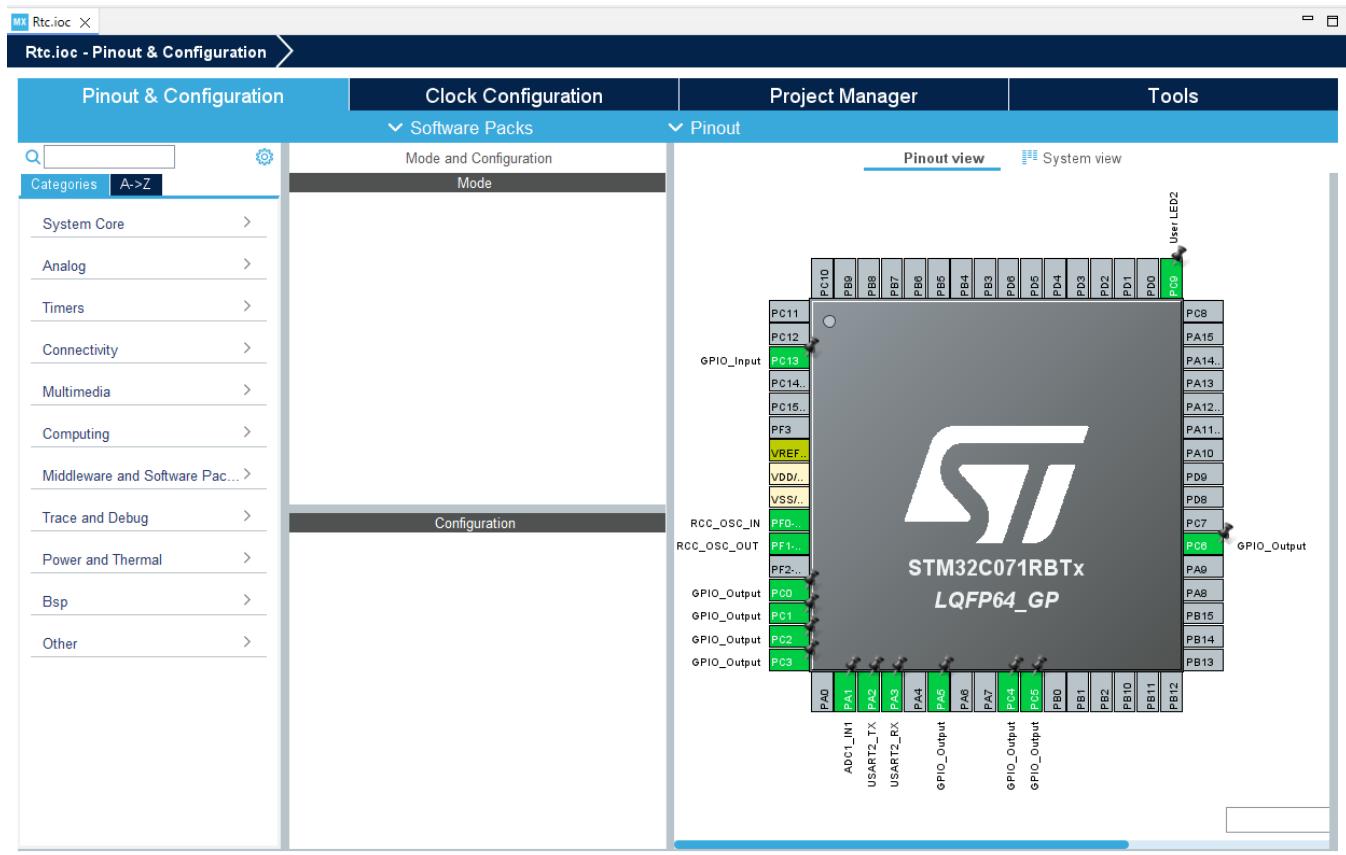


RTC

CONFIGURACION INICIAL



MX Rtc.ioc X

Rtc.ioc - Pinout & Configuration

Pinout & Configuration
Clock Configuration
Project

▼ Software Packs
▼ Pinout

Categories
A-Z

- [System Core >](#)
- [Analog >](#)
- [Timers >](#)
- [RTC](#)

 - [TIM1](#)
 - [TIM2](#)
 - [TIM3](#)
 - [TIM14](#)
 - [TIM16](#)
 - [TIM17](#)

- [Connectivity >](#)
- [Multimedia >](#)
- [Computing >](#)
- [Middleware and Software P... >](#)
- [Trace and Debug >](#)
- [Power and Thermal >](#)
- [Bsp >](#)
- [Other >](#)

RTC Mode and Configuration

Activate Clock Source

Activate Calendar

Alarm A Disable

Timestamp

Calibration Disable

Reference clock detection

GPIO_Input	PC11
PC12	
PC13	PC13
PC14..	
PC15..	
PF3	
VREF..	
VDDI..	
VSSI..	
RCC_OSC_IN	PF0..
RCC_OSC_OUT	PF1..
PF2..	
GPIO_Output	PC0
GPIO_Output	PC1
GPIO_Output	PC2
GPIO_Output	PC3

Configuration

Reset Configuration

Parameter Settings User Constants

Configure the below parameters :

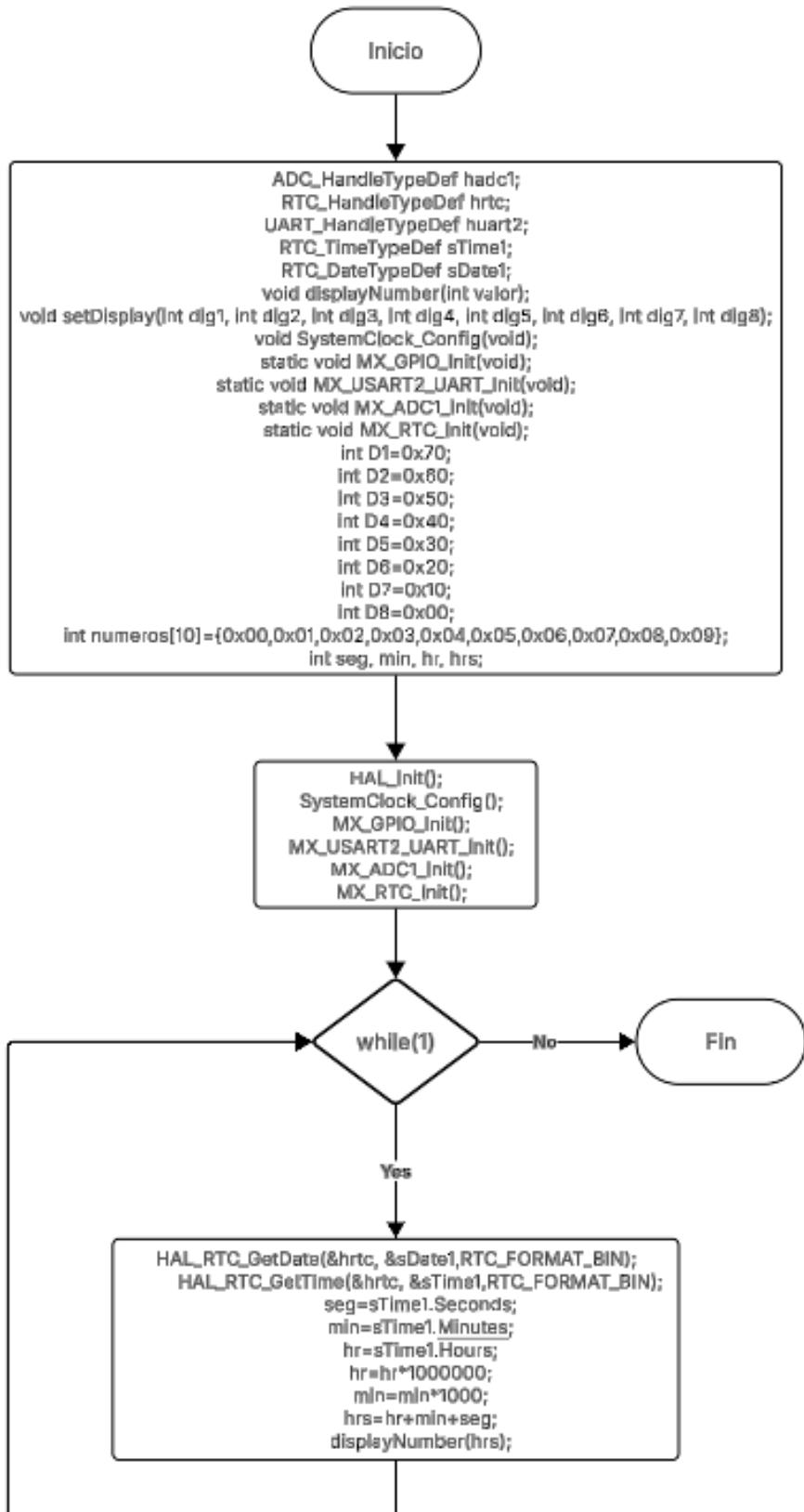
Search (Ctrl+F) (i)

General

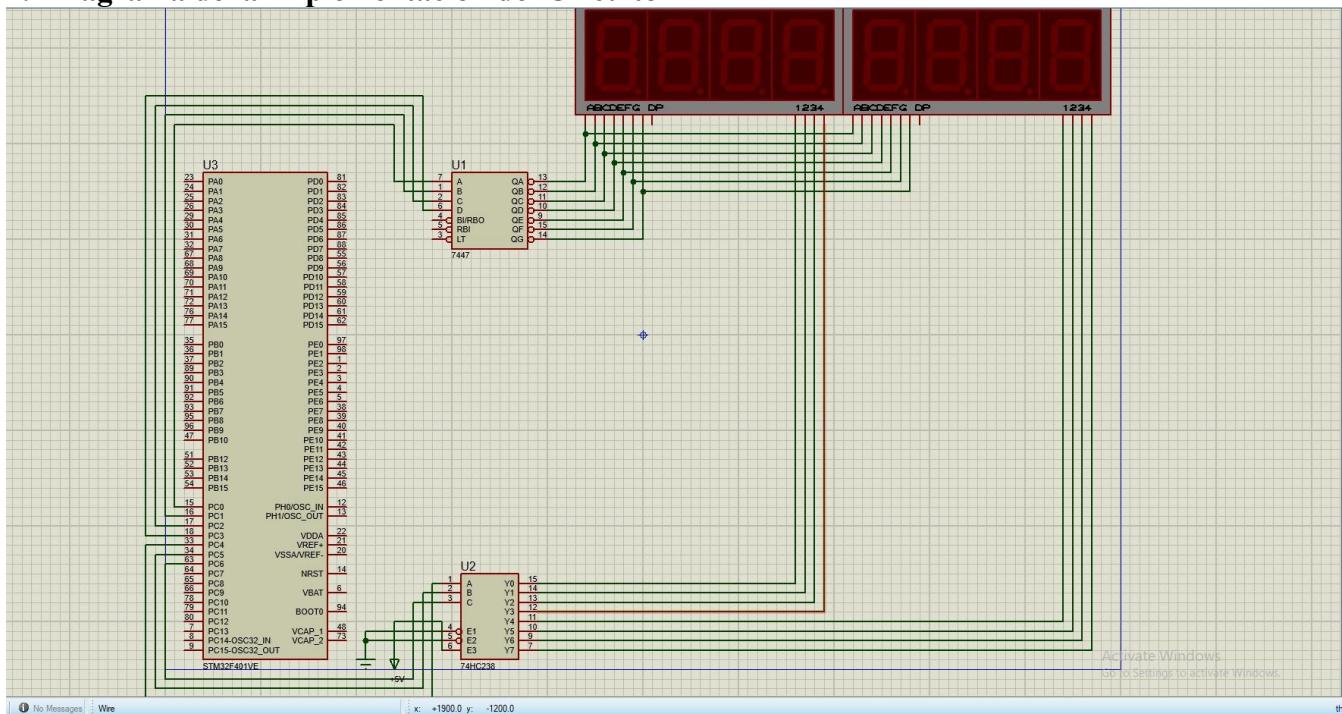
Hour Format	Hourformat 24
Asynchronous Pr...	127
Synchronous Pre...	255

Calendar Time

Data Format	BCD data format
Hours	6
Minutes	55
Seconds	00

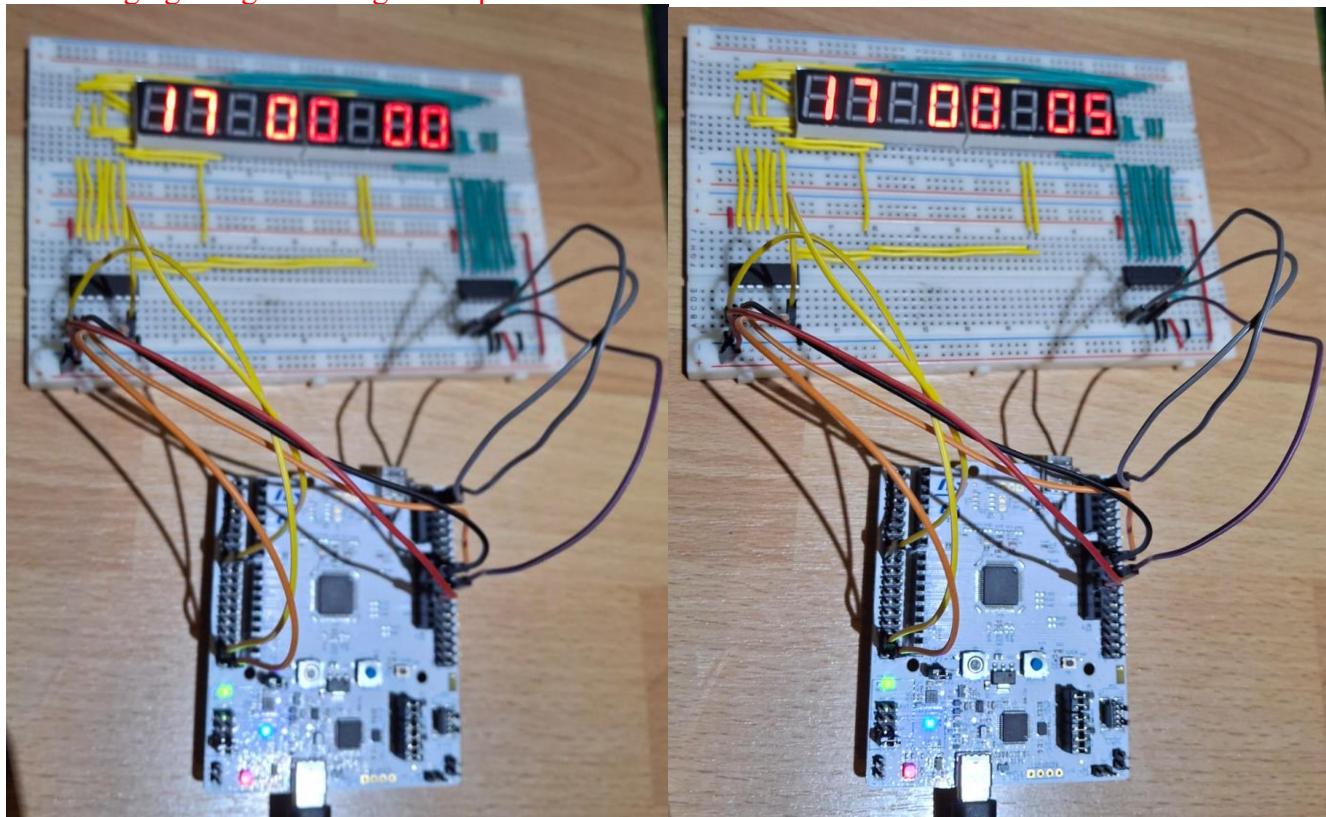


4.- Diagrama de la Implementación del Circuito



5.- Evidencia Grafica

Agregue algunas imágenes representativas del funcionamiento del circuito.



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 la función que muestra un número en el display de 8 dígitos. En la línea 4 se define el prototipo de la función que establece los valores individuales para cada dígito del display. En la línea 5 se declaran las variables enteras para almacenar segundos, minutos, horas y horas en formato de 12 horas. En la línea 6 se define la estructura para configurar y leer la hora del RTC. En la línea 7 se define la estructura para configurar y leer la fecha del RTC.

```
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);
5 int seg, min, hr, hrs;
6 RTC_TimeTypeDef sTime1;
7 RTC_DateTypeDef sDate1;
```

Llamadas utilizadas y su descripción.

En la línea 1 se da la orden de configurar la velocidad de reloj del sistema establecida en la función. En la línea 2 se da la orden de establecer la configuración de las terminales GPIO en función del código establecido en la función. En la línea 3.....

```
1 displayNumber(12345678);
2 setDisplay(dig1,dig2,dig3,dig4,dig5,dig6,dig7,dig8);
3 static void MX_RTC_Init(void);
4 Int GetTime (void);
5 MX_RTC_Init();
```

Llamadas a función

En la línea 1 tal cosa, en la línea 2 se ejecuta la función que configura las velocidades de reloj del sistema, en la linea 3....

Funcionamiento y estructura del loop principal

En la línea 3 ejecuta la llamada que obtiene la fecha actual del RTC. En la línea 4 ejecuta la llamada que obtiene la hora actual (horas, minutos y segundos) del RTC. En la línea 6 se asigna el valor de los segundos obtenidos a una variable. En la línea 7 se asigna el valor de los minutos obtenidos a una variable. En la línea 8 se asigna el valor de las horas obtenidas a una variable.

1	While(1)
2	{
3	HAL_RTC_GetDate(&hrtc, &sDate1,RTC_FORMAT_BIN);
4	HAL_RTC_GetTime(&hrtc, &sTime1,RTC_FORMAT_BIN);
5	
6	seg=sTime1.Seconds;
7	min=sTime1.Minutes;
8	hr=sTime1.Hours;
9	hr=hr*1000000;
10	min=min*1000;
11	hrs=hr+min+seg;
12	displayNumber(hrs);
13	}

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 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 módulo para separar la decena y la división para eliminar las unidades como se muestra en el ejemplo.

$$123456789 \% 100 = 78 \quad 78 / 10 = 7 \text{ dig7} = 7$$

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

.

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);