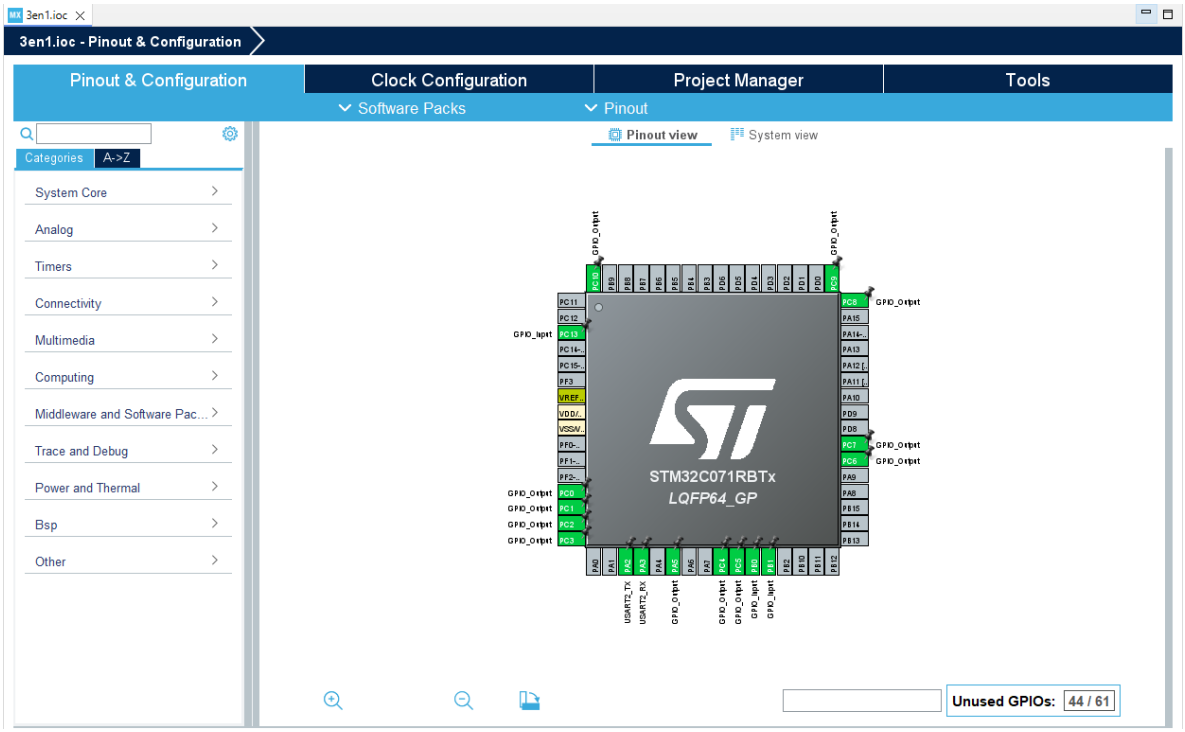
3 EN 1

CONFIGURACION INICIAL.



Actividad Fundamental 1.- 3 en 1

1.- Descripción del Problema

Utilizando un puerto GPIO configurado como entradas digitales, deberá seleccionar entre las siguientes funciones:

01 Contador en Binario

02 Auto Increíble

03 Contador RGB

Para ello deberá tomar en cuenta las siguientes consideraciones

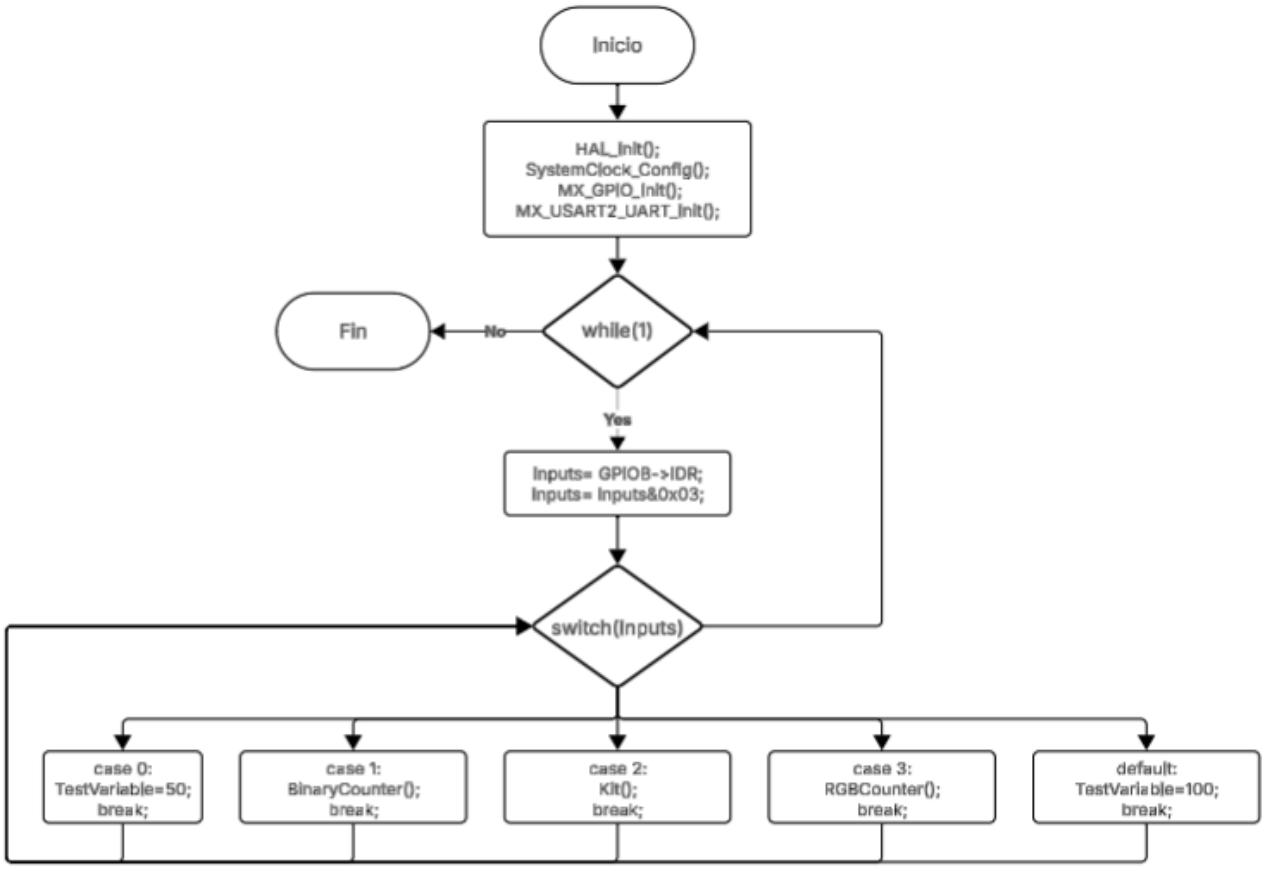
1. Visualizar en un grupo de 8 leds el efecto producido por una cuenta natural ascendente en código binario.
2. Lograr la visualización de una cuenta ascendente dentro de los rangos del 1 al 7 en un led RGB de Cátodo Común.

3. Se requiere establecer un Loop infinito de lectura entre un puerto paralelo de 2 bits configurado como entradas y la función a desarrollar.
4. Solo se deberá efectuar una operación a la vez, es decir si el contador ascendente es seleccionado se incrementara en una posición ejemplo de 0000 0001 a 00000010, y posteriormente regresara a leer las entradas, si es seleccionado el contador RGB el valor del contador binario deberá guardarse y continuar cuando sea seleccionado de nuevo. De la misma forma esto debe suceder para las otras opciones (un desplazamiento en el auto increíble o un incremento en la cuenta RGB).

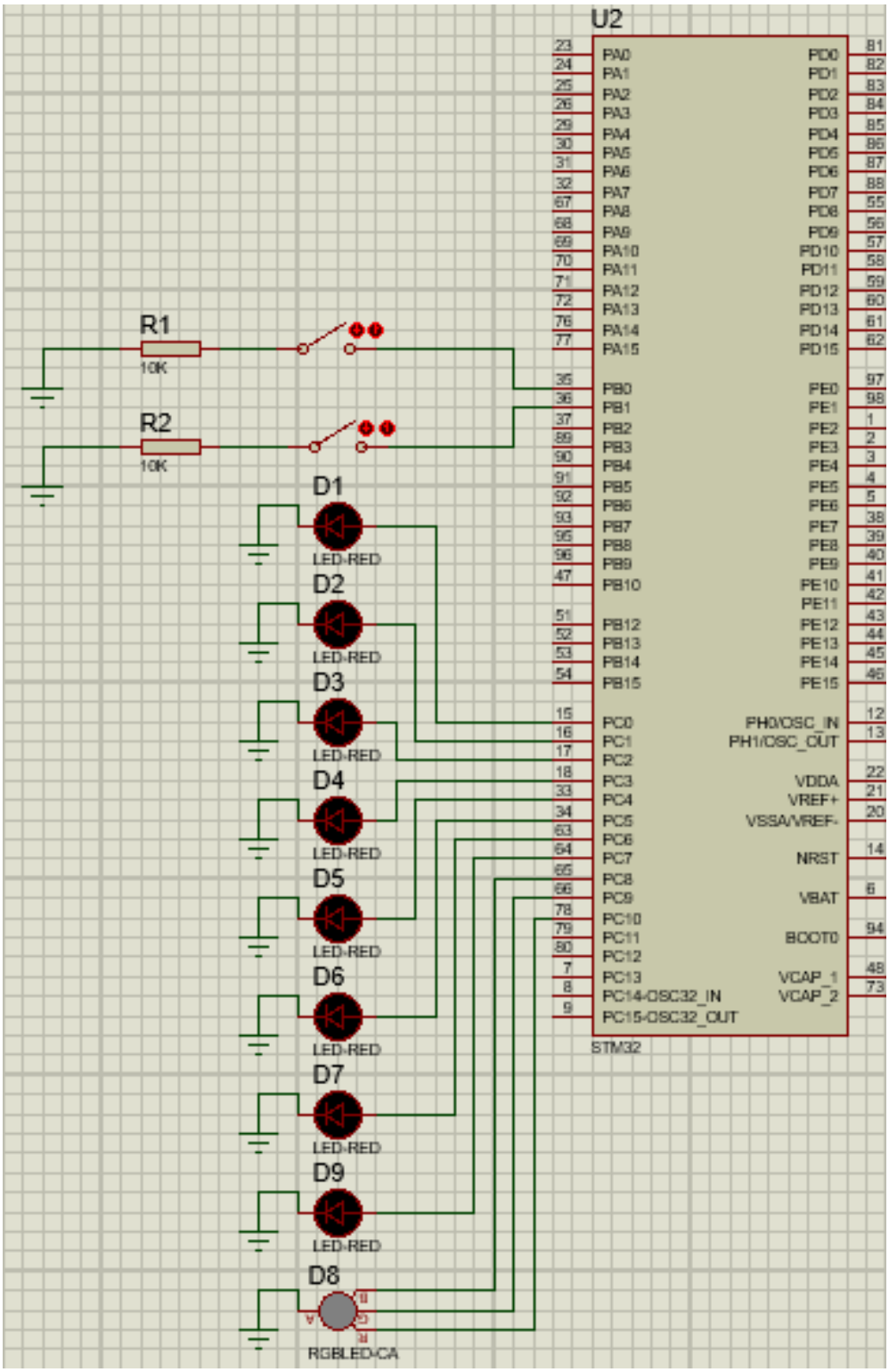
2.- Objetivo

1. Desarrollar un codigo que permita manipular el puerto para leer las entradas.

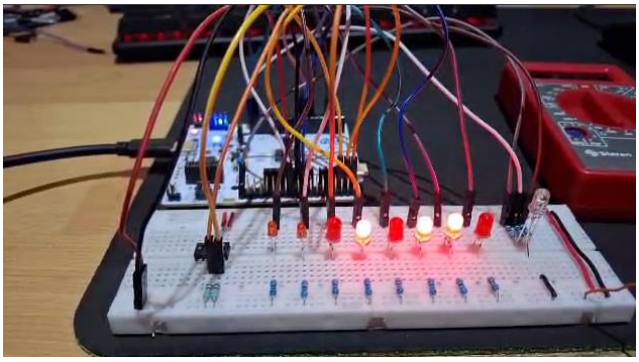
3.- Diagrama de Flujo Propuesto

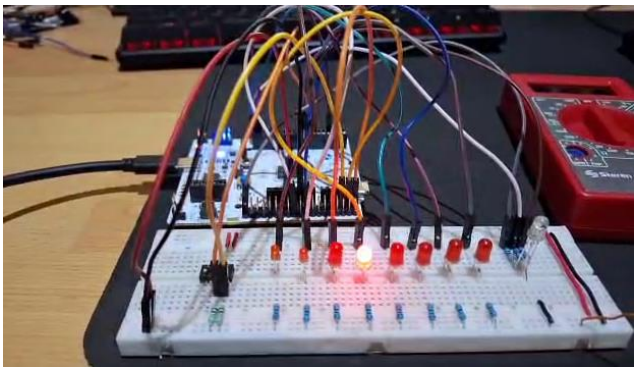


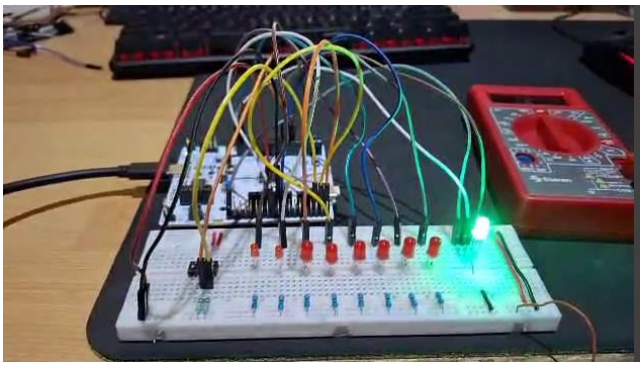
4.- Diagrama de la Implementación del Circuito



5.- Evidencia Grafica







6.- Prototipos de funciones principales

Prototipos utilizados y su descripcion

En la línea 1 esta 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……..

1	**void SystemClock_Config(void);**
2	**static void MX_GPIO_Init(void);**

```
3  void BinaryCounter(void);
4  void Kit (void);
5  void RGBCounter(void);
```

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 codigo establecido en la función. En la línea 3……

```
1  SystemClock_Config();
2  MX_GPIO_Init();
3  BinaryCounter();
4  Kit ();
5  RGBCounter();
```

Funcionamiento y estructura del loop de control principal.

En la línea 1 tenemos el while(1) que constituye el loop de control principal, el cual se ejecuta de forma continua e infinita. En la línea 3 se lee el registro de datos de entrada (IDR) del puerto B para obtener el estado de sus pines y se almacena en la variable Inputs. En la línea 4 se aplica un filtro (0x03) a la variable Inputs, con el fin de aislar y conservar únicamente el estado de los dos primeros pines (bit 0 y bit 1), eliminando cualquier valor basura o estado de los pines que no son de interés para este programa.

En la línea 6 se utiliza una sentencia switch para ejecutar diferentes bloques de código basándose en el valor ya filtrado de Inputs:

- En el caso 0 (líneas 7-8), se asigna el valor 50 a una variable de prueba (TestVariable).
- En el caso 1 (líneas 9-10), se llama a la función BinaryCounter().
- En el caso 2 (líneas 11-12), se llama a la función Kit().
- En el caso 3 (líneas 13-14), se llama a la función RGBCounter().
- El caso default (líneas 15-17), que captura cualquier valor no contemplado en los casos anteriores, asigna el valor 100 a la variable TestVariable.

El funcionamiento general de este loop es el de monitorear constantemente el estado de los dos primeros pines del puerto B (GPIOB) y, dependiendo de la combinación binaria que representen, ejecutar una rutina específica: un valor de prueba, un contador binario, un kit de luces con cambio de sentido o un contador RGB cíclico.

```
1  while (1)
2  {
3      Inputs= GPIOB->IDR;
4      Inputs= Inputs&0x03;
5      //Inputs=2;
```

```
6     switch(Inputs)
7     {
8         case 0:
9         TestVariable=50;
10        break;
11        case 1:
12        BinaryCounter();
13        break;
14        case 2:
15        Kit();
16        break;
17        case 3:
18         RGBCounter();
19        break;
20        default:
21        TestVariable=100;
22        break;
23    }
24  }
```

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

Contador en Binario

En la línea 1 esta el prototipo de la funcion del contador en binario. En la línea 3 se establecen a cero lógico las primeras 8 terminales del puerto C. En la línea 4 se establece el valor de la variable BinCounter en las primeras 8 terminales del puerto C. En la línea 5……..En la linea 6……

El funcionamiento general de esta funcion es el de incrementar el valor de una variable y colocar su valor en las 8 terminales del puerto C, estableciendo un contador en binario en dicho puerto.

```
1  void BinaryCounter(void)
2  {
3      HAL_GPIO_WritePin(GPIOC, 0xFF, GPIO_PIN_RESET);
4      HAL_GPIO_WritePin(GPIOC, BinCounter, GPIO_PIN_SET);
5      HAL_Delay(100);
6      BinCounter++;
7  }
```

Kit y funciones Izquierda y Derecha

En la línea 1 está el prototipo de la función Kit. En la línea 3 se verifica el valor de una variable global sentido. En la línea 5, si sentido es igual a 1, se ejecuta la función izquierda(). En la línea 6, se verifica si sentido es igual a 0. En la línea 8, si se cumple esa condición, se ejecuta la función derecha().

El funcionamiento general de esta función es el de actuar como un controlador o selector que, basado en el estado de la variable global sentido, decide y llama a la función adecuada para mover una secuencia de luces hacia la izquierda o hacia la derecha.

```
1  void Kit(void)
```

```
2   {
3       if(sentido==1)
4               {
5                 izquierda();
6               }else if(sentido==0)
7               {
8                derecha();
9               }
10  }
```

En la línea 1 está el prototipo de la función izquierda. En la línea 3 se verifica si el valor de la variable global Lights es menor que 0x80. En la línea 5 se enciende el LED o los LEDs correspondientes al patrón actual en Lights en el puerto C. En la línea 6 se introduce una pausa de 500 ms. En la línea 7 se apagan los LEDs que se habían encendido. En la línea 8 el valor de Lights se desplaza un bit a la izquierda, moviendo efectivamente la luz hacia el LED de mayor valor. En la línea 9, si Lights ya ha llegado al extremo izquierdo, se cambia la variable global sentido a 0 para que en la siguiente iteración el movimiento sea hacia la derecha.

El funcionamiento general de esta función es el de tomar un patrón de luz, encenderlo, apagarlo después de un breve tiempo y luego desplazarlo hacia la izquierda, creando un efecto de movimiento. Cuando la luz llega al extremo, cambia la dirección del flujo.

```
1   void izquierda (void)
2   {
3       if(Lights<0x80)
4       {
5       HAL_GPIO_WritePin(GPIOC, Lights, GPIO_PIN_SET);
6       HAL_Delay(500);
7       HAL_GPIO_WritePin(GPIOC, Lights, GPIO_PIN_RESET);
8        Lights<<=1;
9        }else{
10       sentido=0;
11       }
12  }
```

En la línea 1 está el prototipo de la función derecha. En la línea 3 se verifica si el valor de la variable global Lights es mayor que 0x01. En la línea 5 se enciende el LED o los LEDs correspondientes al patrón actual en Lights en el puerto C. En la línea 6 se introduce una pausa de 500 ms. En la línea 7 se apagan los LEDs que se habían encendido. En la línea 8 el valor de Lights se desplaza un bit a la derecha, moviendo efectivamente la luz hacia el LED de menor valor. En la línea 9, si Lights ya ha llegado al extremo derecho, se cambia la variable global sentido a 1 para que en la siguiente iteración el movimiento sea hacia la izquierda.

El funcionamiento general de esta función es el de tomar un patrón de luz, encenderlo, apagarlo después de un breve tiempo y luego desplazarlo hacia la derecha, creando un efecto de movimiento. Cuando la luz llega al extremo, cambia la dirección del flujo.

```
1  void derecha (void)
2  {
3      if(Lights>0x01)
4      {
5      HAL_GPIO_WritePin(GPIOC, Lights, GPIO_PIN_SET);
6      HAL_Delay(500);
7      HAL_GPIO_WritePin(GPIOC, Lights, GPIO_PIN_RESET);
8       Lights>>=1;
9        }else{
10       sentido=1;
11       }
12 }
```

Contador RGB

En la línea 1 está el prototipo de la función RGBCounter. En la línea 3 se enciende un pin del puerto C cuya definición está almacenada en un array llamado color en la posición indicada por el índice i. En la línea 4 se introduce un retardo de 1000 ms. En la línea 5 se apaga el pin que estaba encendido. En la línea 6 se incrementa el valor del índice i para pasar al siguiente color en la secuencia. En la línea 7 se verifica si el índice i ha excedido el valor 7. En la línea 9, si se cumple la condición, el índice i se reinicia a 0 para volver al primer color de la lista.

El funcionamiento general de esta función es el de recorrer de forma cíclica un array predefinido de colores (o pines), encendiendo cada uno durante un segundo antes de pasar al siguiente, creando un contador o secuenciador de colores.

```
1  void RGBCounter (void)
2  {
3      HAL_GPIO_WritePin(GPIOC, color[i], GPIO_PIN_SET);
4      HAL_Delay(1000);
5      HAL_GPIO_WritePin(GPIOC, color[i], GPIO_PIN_RESET);
6      i=i+1;//(i++)
7      if(i>7)
8      {
9        i=0;
10     }else
11     {
12      }
13 }
```