INFORME FINAL

PARCIAL 1 - INFORMÁTICA 2
2023-2

INTEGRANTES:
Daniel Eduardo López (T.I. 1061654759)
Ana Isabel Salamanca (T.I. 1033258748)

DOCENTE:
Augusto Salazar

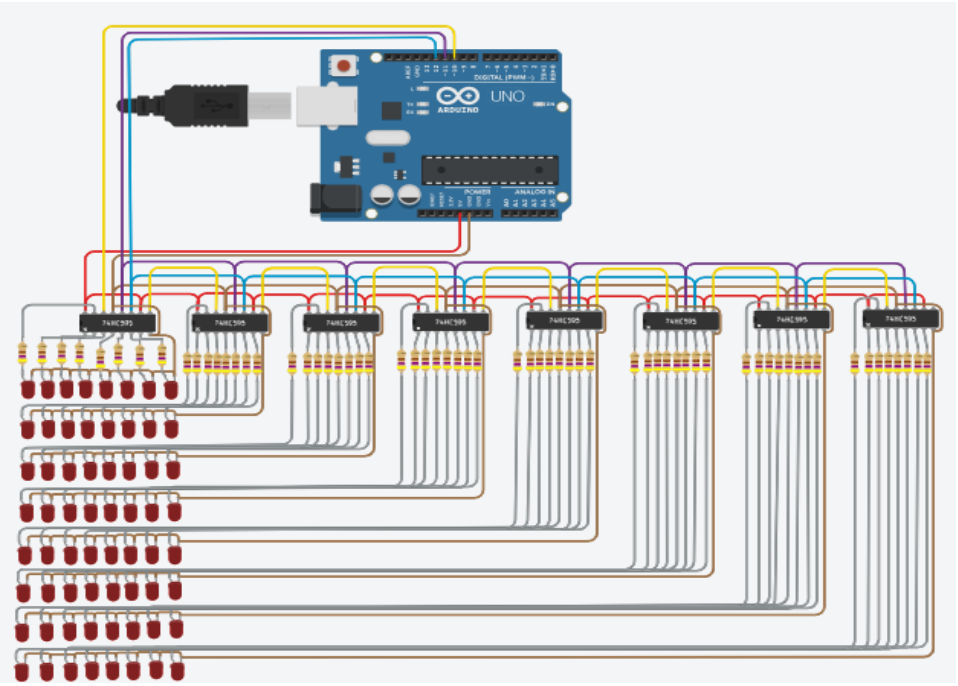
ENLACE DE LA SIMULACIÓN EN TINKERCAD:

https://www.tinkercad.com/things/jBmhGcoBAwH?sharecode=RsE5GaNY281TBmx_zjykPBtfEiTWWvERQFrDD5oo53Y

ANÁLISIS DEL PROBLEMA Y CONSIDERACIONES

Teniendo en cuenta los requerimientos para la resolución del parcial, nosotros optamos por resolver los problemas planteados multiplexando leds de manera que se conecten por filas; es decir, las primeras posiciones de cada columna, las segundas posiciones de cada columna y así sucesivamente.

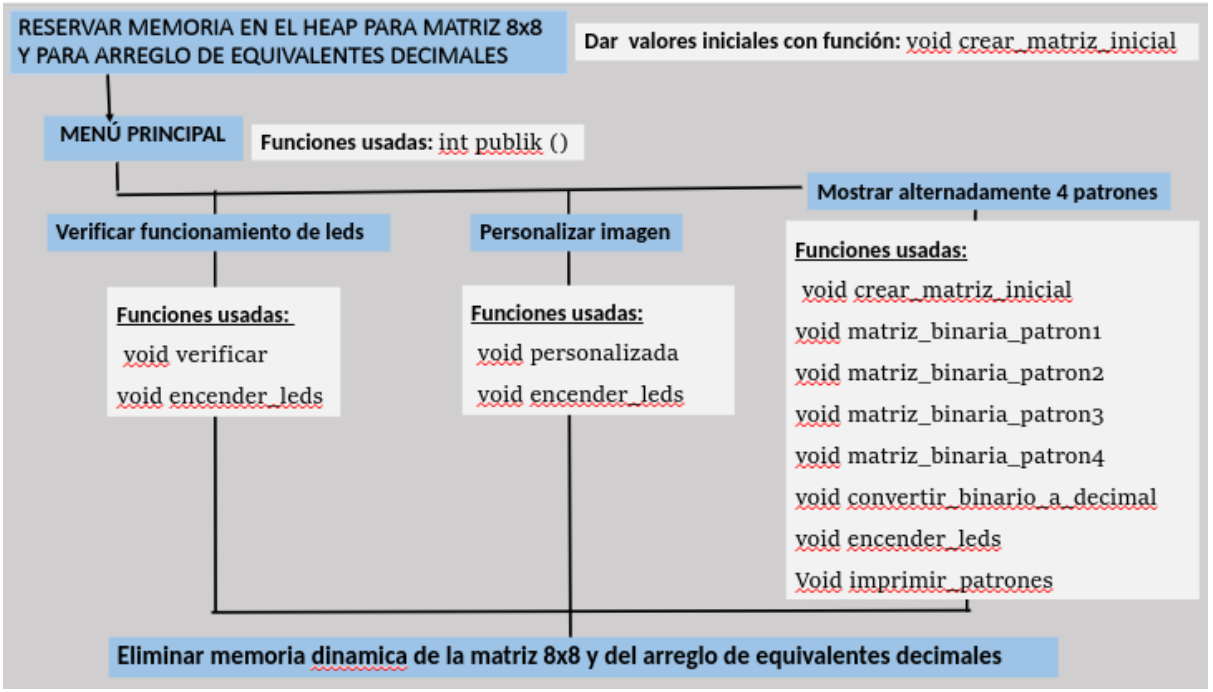
Usamos 8 sistemas de desplazamiento 74HC595 (1 por cada fila) ya que de esta manera se puede manejar con facilidad los estados de los 64 leds utilizando únicamente 3 pines digitales del arduino. En cada fila, los leds están conectados por los cátodos que son los que van a tierra, y los ánodos están conectados cada uno a un pin del sistema de desplazamiento.



Nos pareció pertinente usar 8 sistemas de desplazamiento ya que, si con uno se puede enviar señales de 8 bits y tener un “control” sobre ellas, con los 8 sistemas podríamos enviar las señales de los 64 bits requeridos, pero haciéndolo en conjuntos de a 8.

Es importante considerar varios aspectos sobre el funcionamiento del 74HC595 a la hora de hacer las conexiones e implementar el algoritmo, ya que si no se conectan de manera adecuada o no se tiene en cuenta se entiende la dinámica de cómo trabaja, la solución no será la esperada.

ESQUEMA



ALGORITMOS IMPLEMENTADOS

Primero, se reserva memoria dinámica para una matriz de enteros de 8 posiciones, en donde, cada posición, tiene a su vez, 8 posiciones; en la matriz irán los estados de los 64 leds (1 si está encendido o 0 si está apagado).

```
int **matriz= new int*[filas]; //declarar matriz 8x8
for (int i=0;i<filas;i++){    //reservar espacios para las columnas (arreglos internos)
      matriz[i]=new int[columnas];
}
```

Se reserva también memoria dinámica para un arreglo de enteros de 8 posiciones en donde irán los estados de los leds de cada fila pero en su equivalente decimal; por ejemplo, si se quiere que esté encendido únicamente el primer led de cada fila, el valor de cada posición del arreglo sería 128 ya que es el equivalente decimal de: 10000000.

```
int* valores_decimales=new int[8];
```

Para inicializar la matriz, utilizamos la función de tipo *void crear_matriz_inicial* que recibe como parámetros: el puntero doble de la matriz, el número entero de filas y el número entero de columnas. Esta función recorre las posiciones de la matriz y asigna un valor de 0 a cada una.

```
void crear_matriz_incial(int** matriz,int num_filas,int num_columnas)
{
    for(int filas=0;filas<num_filas;filas++){
        for(int columnas=0;columnas<num_columnas;columnas++){
            matriz[filas][columnas]=0;
        }
    }
}
```

Para encender los leds, según sea el caso, se usa la función de tipo *void encender_leds* la cual recibe como parámetros 8 números enteros, cada número es la equivalencia decimal de los estados de los leds de cada fila. Dentro de la función se implementa la función shiftOut para pasar los datos entre los sistemas de desplazamiento 74HC595 y luego, se encienden los leds.

```
void encender_leds(int fila1, int fila2, int fila3, int fila4, int fila5, int fila6, int fila7, int fila8)
{
   shiftOut(pinData, pinClock, LSBFIRST, fila8);
   shiftOut(pinData, pinClock, LSBFIRST, fila7);
   shiftOut(pinData, pinClock, LSBFIRST, fila6);
   shiftOut(pinData, pinClock, LSBFIRST, fila5);
   shiftOut(pinData, pinClock, LSBFIRST, fila4);
   shiftOut(pinData, pinClock, LSBFIRST, fila3);
   shiftOut(pinData, pinClock, LSBFIRST, fila2);
   shiftOut(pinData, pinClock, LSBFIRST, fila1);
   digitalWrite(pinLatch, HIGH);
   digitalWrite(pinLatch, LOW);
}
```

Para el menú de opciones principales, utilizamos la función *publik,* no tiene parámetros iniciales. Esta función se encarga de mostrar en el monitor serial las opciones de acciones y lee la opción que el usuario escribió. Retorna el número entero de la elección.

```
int publik()
{
        int eleccion;
        Serial.println("\n Ingrese el numero de la opcion que desea realizar:");
        Serial.println("1. Verificar el funcionamiento de los leds");
        Serial.println("2. Mostrar imagen personalizada");
        Serial.println("3. Mostrar alternadamente los 4 patrones predefinidos");

  while(Serial.available() == 0) {
  }
    eleccion = Serial.parseInt();
    Serial.print("escogiste: ");
        Serial.print(eleccion);

  return eleccion;
}
```

1) Para la opción de verificar el funcionamiento de los leds, se usa la función de tipo *void verificar.* Tiene como parámetros dos números enteros, el primero es el número de tiempo (en segundos) que el usuario desea que haya entre encendido y apagado de los leds y, el segundo es el número de veces que se va a repetir esta acción. Dentro de esta función se implementa a su vez la función *encender leds* en donde, cada parámetro será 255, indicando que el estado de los leds es 11111111 (todos encendidos).

```
void verificar(int tiempo, int veces){
  for(int i=0;i<veces;i++){
        encender_leds(255,255,255,255,255,255,255,255);
    delay(tiempo);
    encender_leds(0,0,0,0,0,0,0,0);
    delay(tiempo);
  }

}
```

2) Para la opción de ingresar una imagen personalizada, se usa la función de tipo *void personalizada.* Recibe como parámetros: un puntero simple a entero que apunta al arreglo de los valores decimales, un número entero que es el tiempo que el usuario desea que haya entre encendido y apagado y el número de veces que desea repetir la acción. Dentro de la función se le pide al usuario ingresar por el monitor serial los estados de los 8 leds por filas, es decir, si desea que la fila #1 tenga todos los leds encendidos, el usuario debe ingresar: 11111111. cada uno de este conjunto de bits, se hace la conversión pertinente decimal, se guarda cada equivalente en el arreglo de decimales, y se ejecuta la función *encender_leds.*
(El código de esta función es largo, entonces, preferimos no adjuntar la imagen).

3) Para la opción de encender los 4 patrones de manera alternada, utilizamos varias funciones.

   - Funciones que generan los patrones en la matriz 8x8: Implementamos 4 funciones, una por cada patrón, cada una recibe como parámetros: un puntero doble que apunta a la matriz 8x8, un número entero que hace referencia al número de filas y otro número entero que hace referencia al número de columnas. Cada función utiliza diferentes ciclos e iteraciones para cambiar los valores iniciales de la matriz poniendo “1” en las posiciones de los leds que van encendidos.

     Por ejemplo, para la función que gue genera el patrón #4:

     Función de tipo “void” llamada: “matriz_binaria_patron4”, que recibe 3 parámetros: “**matriz” que será la matriz que contendrá la forma del patrón, “num_filas” que será la dimensión de filas de la matriz y “num_columnas” que será la dimensión de columnas que tomará la matriz.

     En este patrón se muestran 4 Leds encendidos por fila, a medida que se pasaba de fila en fila los Leds que iban encendidos se movían una posición a la derecha hasta llegar a la mitad del tamaño de filas de matriz, a partir de este punto la posición de los Leds a medida que pasaba de fila en fila se movían una posición a la izquierda, mostrando una especie de flecha en la pantalla.

```
void matriz_binaria_patron4(int **matriz, int num_filas, int num_columnas){
    int tam = 8;
    int primera = 0;
    for(int i = 0; i < tam; i++){
        if(i < 4){
            for(int j = 0; j <= 3; j++){
                matriz[i][primera + j] = 1;
            }
            primera ++;
        }
        else{
            primera--;
            for(int j = 0; j <= 3; j++){
                matriz[i][primera + j] = 1;
            }
        }
    }
}
```

   - Función que convierte los estados binarios a su equivalente decimal: usamos la función *void convertir_binario_a_decimal*. Tiene como parámetros un puntero doble a la matriz de enteros y un puntero simple al arreglo de enteros que es donde se guardarán los equivalentes decimales. Esta función recorre

por filas y mediantes condicionales y ciclos, obtiene el equivalente decimal de los estados de los leds, el equivalente de cada fila lo almacena en una posición del arreglo.

```
void convertir_binario_a_decimal(int**matriz,int*arreglo_decimales){

    int tam=8;
    int cont_potencia=0;
    int suma_actual=0;
    int veces_multiplicadas=0;
    int potencia_actual=0;
    for(int i=0;i<tam;i++){
        cont_potencia=0;
        suma_actual=0;
        for(int j=7;j>=0;j--){
            if(matriz[i][j]==1){
                if(j==7){
                    suma_actual++;
                }
                else{
                    veces_multiplicadas=0;
                    potencia_actual=1;
                    while(veces_multiplicadas<cont_potencia){
                        potencia_actual=(potencia_actual*2);
                        veces_multiplicadas++;
                    }
                    suma_actual+=potencia_actual;
                }
            }
            else{
                arreglo_decimales[i]=0;
            }
        cont_potencia++;
        }
        arreglo_decimales[i]=suma_actual;
```

- Función que enciende los patrones alternadamente: Usamos la función *void imprimir_patrones.* Tiene como parámetros: un número entero que hace referencia al tiempo entre cada patrón, un número entero que hace referencia a las veces que se desea repetir la acción, un puntero doble que apunta a la matriz de enteros, un puntero simple que apunta al arreglo de los equivalentes decimales, un número entero que hace referencia al número de filas de la matriz y un número entero que hace referencia al número de columnas de la matriz.
  Esta función tiene el siguiente orden:

    - crea matriz inicial (todos los valores en 0)
    - genera patrón
    - obtiene los equivalentes decimales
    - enciende los leds con estos valores decimales
    - espera según el tiempo que se ingresó como parámetro

Esto lo hace por cada patrón, es necesario crear la matriz inicial cada vez que se quiere generar el siguiente patrón, ya que, si no se hace, se configurarán los valores teniendo como base la matriz del patrón anterior lo cual hará que no se obtenga el resultado esperado.
Este proceso se repite hasta que el contador llegue a las veces que se ingresó como parámetro.
(El código de esta función es largo, entonces, preferimos no adjuntar la imagen)

Al terminar la ejecución, se muestra en el monitor serial un mensaje que dice “ejecución finalizada” y se liberan los espacios de memoria del heap que habían sido reservados al inicio.

PROBLEMAS DE DESARROLLO QUE AFRONTAMOS

- Al comenzar a hacer pruebas encendiendo los leds, muchos de los sistemas de desplazamiento se quemaban, esto era ocasionado porque la máxima corriente que resisten es de 50 mAh, por lo cual, tuvimos que aumentar las resistencias de los leds.

- Al hacer la función en que el usuario desea mostrar una imagen personalizada (función “personalizada”), nosotros queríamos que se leyeran 8 datos (un arreglo) por el puerto serial para hacer más fácil la interacción con el usuario y muchas de las funciones para leer datos por el puerto serial no lo permitían, así que fue necesario hacer muchas pruebas con diferentes funciones de la clase Serial.

- Al ejecutar el patrón #4 en la simulación del tinkercad, uno de los leds de la primera fila encendía y según el patrón, no debía encender; al probar el código en qt y otros programas de compilación, funcionaba correctamente así que luego, observamos detenidamente y nos dimos cuenta de que la conexión de este led no era correcta ya que estaba conectado en el mismo pin de salida del led de su izquierda.

EVOLUCIÓN DE LA SOLUCIÓN

Al comenzar, debimos investigar sobre estos sistemas 74hc595 y su funcionamiento, además de como conectar y multiplexar leds entre sí, fue necesario también plantearnos cuál era la mejor manera de hacer las conexiones y cuantos sistemas de desplazamiento usar; luego de tomar la decisión, hicimos diversas pruebas y empezamos a desarrollar el código de los patrones siguiente su lógica y/o secuencia; fue importante hacer las consideraciones de memoria dinámica.

Luego, sabiendo la manera en que íbamos a encender los leds, desarrollamos funciones adicionales que serían necesarias en la ejecución. Subimos los códigos en la página de tinkercad y los organizamos correspondientemente para observar qué nos hacía falta y para poder hacer pruebas con el circuito. Por último, se organizó la ejecución del loop y se

hicieron modificaciones en el estilo y /o mensajes para hacer más agradable el proceso para el usuario.