**Informe final**

**Proyecto en tinkercad**

https://www.tinkercad.com/things/18t9aCXXPrk

**Algoritmos implementados**

Para el desarrollo de los patrones se implementan seis funciones distintas,  cuatro corresponden a la creación de los patrones, definidas con los nombres patron\_1, patron\_2,  patron\_3 y patron\_4; las dos funciones restantes, son destinadas a la reservación y liberación de memoria de la matriz base , están declaradas con los nombres crear\_matriz y liberar\_memoria respectivamente.

Dado que una matriz corresponde a un arreglo donde cada elemento corresponde a otro arreglo, la función crear\_matriz se define así:

* inicia con la reserva de memoria de una cantidad de elementos, esta cantidad representa el número de filas, y se almacena sobre una variable de tipo puntero doble booleano
  + utilizamos un for para iterar sobre cada fila
  + se hace una reservación de memoria en cada iteración, del tamaño de las columnas necesarias
  + agregamos un for que itere sobre cada columna creada
  + inicializamos cada elemento en 0,esto con el fin de asegurarnos de no almacenar basura en la matriz

La función liberar\_memoria sigue un proceso análogo a la reservación en crear\_matriz

* Con un ciclo for que recorre las filas, se elimina en cada iteración la memoria almacenada en las filas, es decir que eliminamos las columnas
* Una vez salimos del ciclo se elimina la memoria de todo el puntero, es decir, que eliminamos la memoria que almacena a las filas.

Para las funciones que crean de los patrones, se inicia de la misma forma

* se almacena el return de crear\_matriz en una variable puntero doble de tipo booleano,

En patron\_1

* con un ciclo while se controlan que el límite superior del intervalo no supere las dimensiones de la matriz
* recorremos la mitad de filas con un for
  + con otro for se recorren todas las columnas
  + se utiliza un condicional doble dentro de  if que verifica si el led en la columna actual se encuentra dentro del intervalo, el número de la columna debe ser menor o igual al límite superior del intervalo y a su vez debe ser mayor o igual al limite\_inferior
    - si cumple ambas condiciones, al valor en esa fila y en esa columna se le asigna 1.
* Una vez fuera del ciclo de las columnas, se actualiza los limites de intervalo, el límite superior disminuye uno y el límite inferior incrementa uno

En patron\_2

* se declara un ciclo for que recorre las columnas de la siguiente forma, al contador se le asigna el valor máximo que puede recorrer en las columnas, este contador se actualiza disminuyendo en una unidad y se repite el ciclo hasta que se llegue a la mitad de las columnas
  + la primera mitad del patrón corresponde a una igualdad entre columna y fila, por esto usando solo el contador de filas se hacer la asignación
  + con un if formamos la otra mitad, que evalúa sí el contador de las filas y las columnas al sumarse es igual al valor máximo que puede recorrer en las columnas.
    - Si cumple la condición se le asigna uno a esa posición

En patron\_3

* el patrón se basa en encender dos leds continuos y apagar el siguiente, lo único que diferencia a una fila de otra es desde donde se comienzan a encender, por esto se recorren las columnas con un for de la siguiente forma, el valor que se le asigna al contador del ciclo corresponde a una división entre 2 con el contador de las filas, dicho contador se actualiza aumentando en 3 unidades
  + se asigna uno al primer led de las parejas que encendemos
  + para no acceder a espacios de memoria no definidos, con un if  nos aseguramos de que el contador de columnas sea diferente al máximo espacio que puede recorrer
    - sí cumple la condición asignamos 1 al led continuo.

En patron\_4

* para cada fila del patrón se encienden una cantidad de leds que corresponden a la mitad de las columnas, la diferencia entre cada fila del patrón es la columna en que comienzan a encenderse, es por esto que la inicialización del contador de columnas está dado por una igualdad con las filas que itera hasta que el contador de columnas encienda los leds necesarios
  + se asigna a cada posición el 1.

La función verificación enciende y apaga los leds la cantidad de veces especificada y durante el tiempo especificado para esto hace uso de:

* un ciclo for que se ejecuta tantas veces como se le necesite encender y apagar los leds, dentro de este ciclo for se encuentran:
  + Un llamado a la función registros para que envíe 1 a través de todas las salidas del integrado 74hc595 que controla las filas y 0 a través de todas las salidas del integrado que controla las columnas.
  + Un delay del tiempo especificado.
  + Un llamado a la función registros para que envíe 0 a todos los puertos.
  + Otro delay del tiempo especificado.

El control de los leds está dado por la función imagen, esta muestra una matriz durante el tiempo especificado, para esto tiene:

* Un ciclo while que se detiene cuando el tiempo de ejecución es mayor que el tiempo de ejecución al momento de llamar la función más el tiempo especificado, dentro de el ciclo while se encuentra:
  + Un ciclo for que itera desde la primera hasta la última fila de la matriz que se le específico, dentro de dicho ciclo for se encuentra:
    - El llamado de la función notarray\_decimal para convertir cada fila en un número decimal y encontrar su inverso binario al restarle el número a 255 (el valor máximo que pueden tomar 8 bits).
    - El llamado a la función registros para poder enviar el número de fila al integrado 74hc595 encargado de manejar las filas y el valor inverso de la fila al integrado encargado de manejar las columnas.
* Un llamado a la función registros con 0 en las filas y 0 en las columnas para que apague todos los leds.

La matriz se enciende y apaga fila por fila lo suficientemente rápido como para que no se logre ver el cambio.

Para la manipulación de la matriz por parte del usuario contamos con la función publik, esta función cuenta con:

* Dos preguntas al usuario, una para saber qué quiere ver en la matriz y otra para saber cuánto tiempo desea verlo.
* Un switch que depende de la opción que haya elegido el usuario el cual tiene 3 opciones:
  + Si el usuario quiere verificar la matriz:
    - Le pregunta al usuario cuantas veces quiere que se prenda y se apague la matriz
    - Llama a la función verificación.
  + Si el usuario quiere ver un patrón:
    - Le pregunta al usuario qué patrón quiere ver.
    - Utiliza un switch que depende de el patrón elegido por el usuario para:
      * Invocar la matriz respectiva.
      * Llamar a imagen con dicha matriz.
      * Liberar la memoria utilizada para borrar dicha matriz.
  + Si el usuario quiere ver la sucesión de todos los patrones:
    - Utiliza un while que se ejecuta hasta que haya una nueva entrada por el serial  (esto es para que se repita el la sucesión hasta que el usuario lo decida), dentro del while se encuentran las siguientes instrucciones:
      * Crea el primer patrón.
      * Llama la función imagen para mostrarlo durante el tiempo especificado.
      * Elimina la matriz para liberar el espacio.
      * Crea el segundo patrón.
      * Llama la función imagen para mostrarlo durante el tiempo especificado.
      * Elimina la matriz para liberar el espacio.
      * Crea el tercer patrón.
      * Llama la función imagen para mostrarlo durante el tiempo especificado.
      * Elimina la matriz para liberar el espacio.
      * Crea el cuarto patrón.
      * Llama la función imagen para mostrarlo durante el tiempo especificado.
      * Elimina la matriz para liberar el espacio.

**Problemas de desarrollo**

Alrededor del 60% del proyecto tuvimos que cambiar la forma en la que se generaban las matrices, sin embargo esto implicaba varios cambios en el resto del código, el cual ya estaba en una fase bastante avanzada, sin embargo también fue una oportunidad para mejorar tanto la legibilidad como la subdivisión en funciones del código, si bien este cambio generó retrasos en forma de repetición de tareas ya realizadas en última instancia disminuye el tiempo total del proyecto debido a que la legibilidad y planteamiento mejorados aceleraron drásticamente la velocidad a la que se escribió el resto del código.

En la generación de las matrices se identificó un almacenamiento incorrecto en la memoria dinámica, esto resultó en un replanteamiento de los código y en el uso de la reserva de memoria, se re diseñaron las funciones con el objetivo de dar sentido al uso de matriz dinámica ya que su implementación da lugar a un posible reacondicionamiento de la matriz de leds, este cambio resultó en la implementación de parámetros en las funciones y en un cambio en la asignación de las variables.

**Evolución de la solución**

Durante el desarrollo del proyecto hubo tres evoluciones importantes de la solución, el paso de la información a los integrados 74hc595, los patrones residuales generados, y la forma de generar los patrones.

durante la primera versión del código se llamaba shiftout (la función para enviar información a los integrados) de formas distintas dependiendo del momento en el que se llamara, luego del cambio de los patrones también se estandarizó la forma en la que se debía enviar la información, permitiendo encapsular el proceso en una función.

Al momento de encender los leds algunos de ellos se encendían sin realmente ser parte de la imagen, estos leds siempre se encontraban alrededor del patrón original dándole una unidad de grosor extra al patrón, debido a esto la solución es tan simple como eliminar los leds en el borde del patrón para disminuir en una unidad el grosor de la imagen, esto es posible para todos los patrones excepto para el número dos, debido a que el grosor original del patrón es uno, por lo tanto no se puede disminuir su grosor sin dejar la matriz totalmente apagada.

Los códigos para los patrones por su parte, evolucionaron teniendo en cuenta el uso adecuado de la memoria dinámica, la legibilidad del código y su capacidad de ser reutilizado.

En la primera versión de las funciones, se toma como  base la mitad de la matriz original de 8\*8, es decir, se genera una matriz de memoria dinámica de 4\*8 sin inicializar dentro de cada función, posteriormente se crea el patrón, y finaliza con el retorno de su dirección de memoria, sin solicitar parámetros, sin embargo, esta propuesta no considera que crear solo la mitad generaba problemas con la matriz de leds en el arduino, dado que la otra parte del arreglo no existe en memoria, en un primer momento se considera la creación de una sola matriz de 8\*8 por fuera de las funciones y que en su lugar cada función recibe como parámetro su dirección de memoria, esta opción se descarta considerando que para generar cada patrón, necesito una matriz nula sobre la cual trabajar, en tal caso debería inicializar nuevamente toda la matriz en cero, por esto, se decide crear una función exclusiva para crear la matriz dentro de la función de cada patrón, finalmente se crea una función destinada a la liberación de la memoria reservada, la cual no se había tomado en cuenta en la primera versión y es necesaria para no causar contratiempos en el programa.