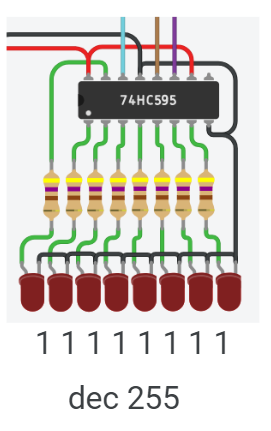
**Parcial 1**

MIGUEL ANGEL BOTERO GONZALEZ

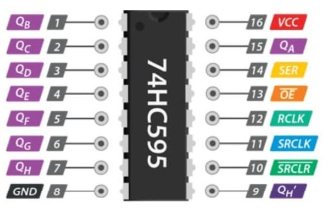
SEBASTIÁN OCHOA MOLINA

Informe

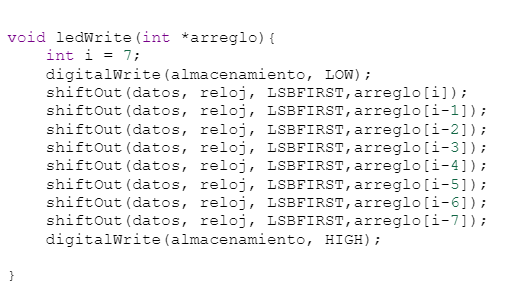
# Análisis y Solución Propuesta



El proyecto comenzó abordando la problemática de cómo controlar 64 diodos LED de manera sincronizada siguiendo un patrón específico por filas, empezando por la fila superior. Para esto, se decidió utilizar un arreglo unidimensional de números enteros, donde cada posición del arreglo representaba el comportamiento de una fila de LED, tomando valores de 0 a 255, que corresponde a la combinación máxima posible en un byte, y esto representaba las distintas combinaciones posibles para cada fila de LED.



Luego, se investigó y se buscó aprovechar el circuito integrado 74HC595, utilizando ocho de estos chips para almacenar los datos que debían ser representados en cada columna de LED. Cada salida de estos chips se configuró para tomar un valor binario de 1 o 0, indicando si el LED debía estar encendido o apagado.

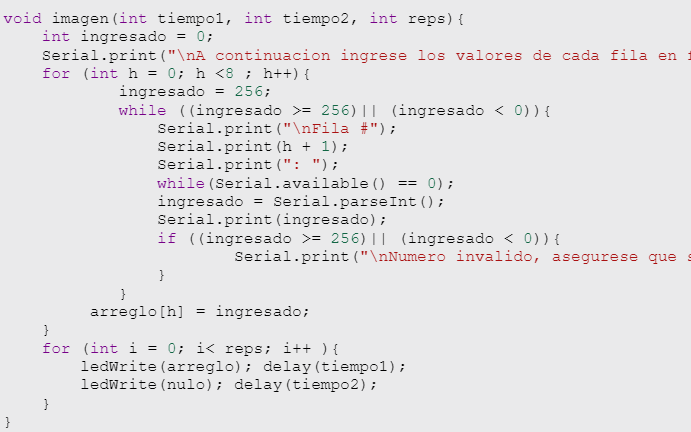
Función ledWrite

Una vez se estableció una base general para el circuito, se desarrolló una función llamada **"ledWrite"** para controlar cada LED. Esta función recibía un arreglo de 8 posiciones, donde la primera posición correspondía a la primera fila de LED y así sucesivamente. Se utilizó la función "shiftOut" para enviar los datos de forma inversa debido al desplazamiento de datos en los circuitos integrados.

Después de lograr un funcionamiento exitoso de los LEDs, se procedió a crear la función "verificación" para asegurarse de que cada LED se encendiera correctamente. Luego, se continuó con la función "imagen" para tener un control completo sobre cada posición de la matriz y extender este control a los patrones restantes.

En cuanto a los algoritmos, la función **"ledWrite"** se considera fundamental y base para todo el proyecto, ya que permitía controlar los LEDs sin importar los datos que contuviera. Además, se empleó un algoritmo propio en los patrones que hacía uso de la memoria dinámica y devolvía un puntero a una dirección de memoria donde se almacenaban temporalmente los datos del arreglo.

Patrón Imagen



El algoritmo "imagen" comienza por recibir tres parámetros: "tiempo1," "tiempo2," y "reps." Estos parámetros controlan la velocidad y la repetición de la visualización de la imagen generada.

A continuación, el algoritmo guía al usuario para ingresar los valores de cada fila en formato entero a través del puerto serial. Se utiliza un bucle for para iterar a través de las ocho filas de la matriz de LEDs. Dentro del bucle, se solicita al usuario que ingrese valores para cada fila, asegurándose de que estén en el rango válido de [0, 255].

Se muestra un mensaje indicando la fila actual.

Se espera a que el usuario ingrese un valor.

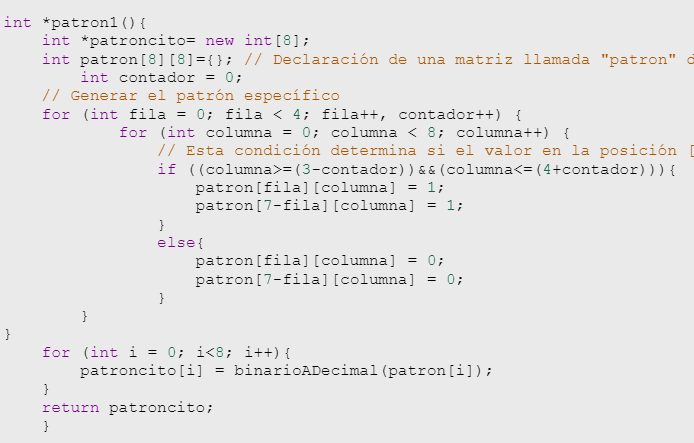
Se valida que el valor ingresado esté dentro del rango válido.

Los valores ingresados se almacenan en un arreglo llamado "arreglo."

Una vez que se han ingresado todos los valores de fila, se procede a generar la imagen dinámica utilizando la función "ledWrite." Se realiza una secuencia de encendido y apagado de la matriz de LEDs según los tiempos especificados en "tiempo1" y "tiempo2." Esta secuencia se repite según el valor de "reps," lo que permite visualizar la imagen dinámica varias veces.

# Algoritmos Implementados

Patrón 1

El algoritmo "patron1" se inicia mediante la declaración de un arreglo de enteros llamado "patroncito", que almacenará el patrón generado. Además, se declara una matriz bidimensional llamada "patron" con dimensiones 8x8 para representar la matriz de LEDs y definir su estado de iluminación.

El proceso de generación del patrón implica la iteración a través de la matriz "patron" utilizando dos bucles for anidados. Durante esta iteración, se determina si cada LED debe estar encendido (valor 1) o apagado (valor 0) en función de su posición en la matriz. La lógica detrás de esta decisión se basa en la creación de un patrón en forma de rombo.

Se utiliza un contador para controlar el número de iteraciones y la forma del rombo.

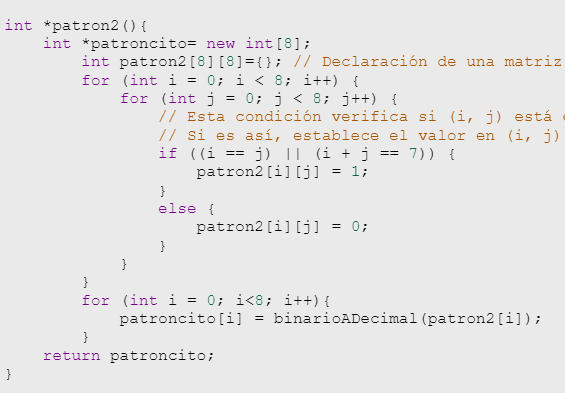
Los LEDs en las filas superior e inferior siempre están apagados.

Los LEDs en las filas intermedias se encienden de manera que creen una forma de rombo.

Los valores resultantes se almacenan en la matriz "patron".

Luego, se convierten los valores binarios de cada fila de la matriz "patron" en valores decimales y se almacenan en el arreglo "patroncito". Finalmente, el arreglo "patroncito" se devuelve como resultado del algoritmo.

Patrón 2

El algoritmo "patron2" comienza declarando un arreglo de enteros llamado "patroncito," que se utilizará para almacenar el patrón de iluminación generado. Además, se declara una matriz bidimensional llamada "patron2" con dimensiones 8x8, que representará la matriz de LEDs y contendrá información sobre su estado de iluminación.

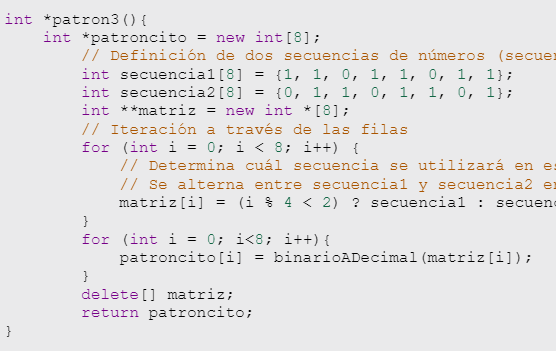
El proceso de generación del patrón se lleva a cabo mediante dos bucles for anidados que recorren las filas y columnas de la matriz "patron2." Durante esta iteración, se utiliza una condición para determinar si cada LED debe estar encendido (valor 1) o apagado (valor 0). La condición se basa en la posición del LED en la matriz y se enfoca en iluminar dos diagonales opuestas.

Si la posición (i, j) está en una de las dos diagonales de la matriz (ya sea la diagonal principal o la diagonal secundaria), se establece el valor en (i, j) como 1.

Si la posición (i, j) no está en ninguna de las diagonales mencionadas, se establece el valor como 0.

Una vez que se completa la generación del patrón en la matriz "patron2," se procede a convertir los valores binarios de cada fila de la matriz en valores decimales. Estos valores se almacenan en el arreglo "patroncito." Finalmente, el arreglo "patroncito" se devuelve como resultado del algoritmo, permitiendo que el patrón de diagonales se refleje en la matriz de LEDs.

Patrón 3

El algoritmo "patron3" se inicia declarando un arreglo de enteros llamado "patroncito," que se utilizará para almacenar el patrón de iluminación generado. Además, se definen dos secuencias de números llamadas "secuencia1" y "secuencia2," cada una con 8 elementos. Estas secuencias representan los valores de iluminación de los LEDs en dos estados diferentes.

Para generar el patrón alternante, se crea una matriz dinámica llamada "matriz," que consta de 8 filas y 8 columnas. Luego, se inicia una iteración a través de las filas de la matriz. Durante esta iteración, se determina cuál de las dos secuencias, "secuencia1" o "secuencia2," se utilizará para llenar cada fila de la matriz. La decisión se basa en el valor de la variable "i," que representa el número de fila actual.

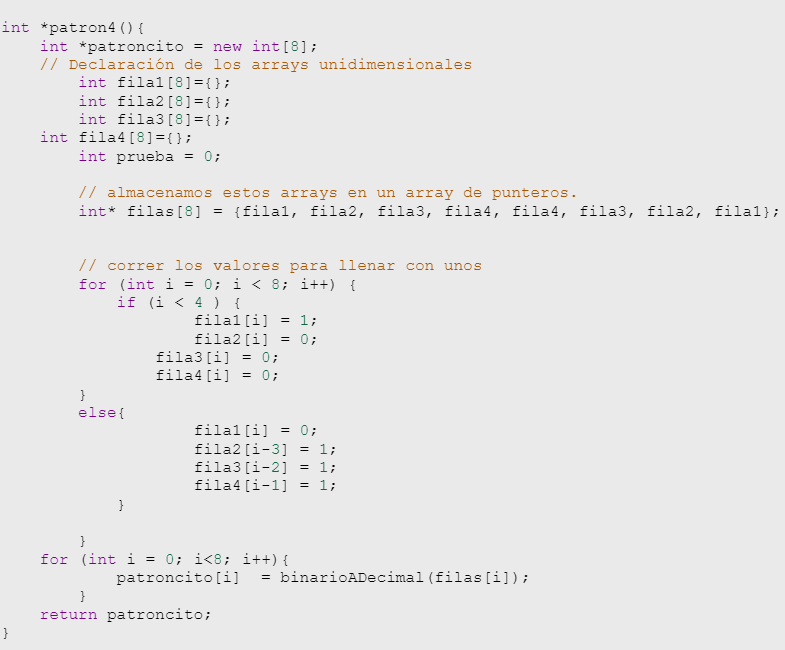
Si "i" es menor que 4, se utiliza "secuencia1" para llenar la fila actual.

Si "i" es igual o mayor que 4, se utiliza "secuencia2" para llenar la fila actual.

De esta manera, se crea una alternancia visual entre las dos secuencias de números en grupos de 4 filas. Cada fila de la matriz "matriz" representa un patrón alternante.

Luego, se convierten los valores binarios de cada fila de la matriz "matriz" en valores decimales y se almacenan en el arreglo "patroncito." Finalmente, se libera la memoria asignada a la matriz "matriz" y se devuelve el arreglo "patroncito" como resultado del algoritmo.

Patrón 4



El algoritmo "patron4" comienza declarando un arreglo de enteros llamado "patroncito," que se utilizará para almacenar el patrón de iluminación generado. Además, se declaran cuatro arrays unidimensionales llamados "fila1," "fila2," "fila3," y "fila4," cada uno con 8 elementos. Estos arrays representarán las filas de la matriz de LEDs.

Se crea un array de punteros llamado "filas," que almacena los punteros a los arrays "fila1" a "fila4." Esto permite acceder a las filas de manera eficiente durante la construcción del patrón.

El proceso de generación del patrón implica llenar los arrays unidimensionales "fila1" a "fila4" con valores 1 y 0 de manera alternada para crear el patrón deseado. Los bucles for se utilizan para realizar esta tarea

Las filas 1 y 2 se llenan con unos alternados (1, 0, 1, 0, ...).

Las filas 3 y 4 se llenan con unos alternados desplazados (0, 1, 0, 1, ...).

Una vez que se llenan las filas individuales, se utiliza un segundo bucle for para convertir los valores binarios de cada fila en valores decimales. Estos valores decimales se almacenan en el arreglo "patroncito."

Finalmente, se libera la memoria asignada a los arrays de punteros y se devuelve el arreglo "patroncito" como resultado del algoritmo.

# Dificultades

Uno de los desafíos iniciales fue la conexión del circuito y su adaptación a la estructura de datos. Sin embargo, una vez resuelto esto y superada la cuestión del comportamiento de los LEDs, el desarrollo del proyecto avanzó sin mayores dificultades.

Uno de los puntos críticos fue la conexión del circuito, ya que se tuvieron dos prototipos funcionales en la primera semana. Se eligió el prototipo de entrega debido a que el otro requería estructuras diferentes y no lograba mantener los LEDs encendidos de manera adecuada, ya que se apagaban después de encenderse para pasar a la siguiente fila.