**Análisis del problema**

El problema consiste en desarrollar una solución que permita controlar una matriz de LEDs de 8x8 usando un Arduino. La solución debe cumplir con los siguientes requisitos:

**Alternativas de solución propuestas**

Para este proyecto decidimos utilizar el integrado 74HC595 que fue recomendado por el profesor. El integrado 74HC595 es un chip que expande las salidas digitales de un microcontrolador. Permite controlar múltiples dispositivos utilizando solo unos pocos pines. Los datos se ingresan de manera serial, se desplazan internamente y luego se almacenan para activar o desactivar salidas, como LEDs o displays, con facilidad. Es útil para ahorrar pines en proyectos de electrónica digital.

Este cuenta con las siguientes terminales:

SER (Serial Data Input): Aquí ingresas los datos de entrada uno a uno de manera serial.

SRCLK (Shift Register Clock Input): Este pin recibe pulsos de reloj para desplazar los datos desde la entrada serial al registro de desplazamiento.

RCLK (Latch Clock Input): Cuando aplicas un pulso de reloj aquí, los datos almacenados en el registro de desplazamiento se copian instantáneamente en el registro de salida para activar las salidas.

OE (Output Enable): Usado para habilitar o deshabilitar todas las salidas del chip. Cuando está en estado bajo (GND), las salidas están habilitadas. En estado alto (VCC), se deshabilitan.

QH' (Serial Data Output): Este es el pin de salida en cadena, que te permite conectar múltiples 74HC595 en serie para expandir aún más las salidas. Los datos se transmiten a otros chips conectados en cascada.

VCC y GND: Suministro de energía. VCC se conecta a la fuente de alimentación positiva (+5V típicamente), y GND a la tierra (0V).

La primera alternativa de solución propuesta fue usar un único integrado 74HC595 para controlar todos los LEDs de la matriz. Sin embargo, esta solución no era posible porque el proyecto nos limita a solo tiene 7 pines digitales disponibles en el Arduino. Con un solo integrado 74HC595 y 7 pines disponibles solo podemos controlar 7 filas ya que el integrado disponga de 8 pines de salida, uno de estos pines se utiliza para controlar el pin de datos, por lo tanto, nos quedan 7 pines de salida restantes, por lo cual no podríamos representar la matriz 8x8 ya que cada pin de salida representa una fila.

La segunda alternativa de solución propuesta fue usar dos integrados 74HC595, uno para controlar las filas y otro para controlar las columnas. Es decir, un integrado se utilizaría para controlar las filas, y el segundo para controlar el registro de desplazamiento

Las filas se controlan mediante los pines de la matriz de LEDs. El pin de fila se establece en alto para encender todos los LEDs de esa fila. Luego, se utilizan los pines del registro de desplazamiento para controlar el estado de cada LED de la fila.

Las columnas se controlan mediante los pines del registro de desplazamiento. Cada pin del registro de desplazamiento controla el estado de un LED de la matriz.

Ahora una tercera alternativa, seria usar 8 integrados y que cada uno controle una fila de la matriz de LEDs, esto es posible gracias al pin QH'. Esta opción nos parece la más fácil de entender por su simplicidad, pero es un poco engorroso ya que requiere de muchas conexiones donde una mala conexión puede ocasionar que nada funcione correctamente, pero como estamos usando Tinkercad nos facilita mucho el trabajo y la disminución de errores de conexión que se podrían presentar en la vida real, por la tanto usaremos esta opción para nuestro proyecto.

**Análisis de problemas propuestos.**

1) Para tratar de ensamblar la matriz de leds teníamos 2 opciones en el tinkercad, una tira de 8 led que representarían las 8 filas de mi matriz, y la segunda es usar 64 leds con su respectiva resistencia para cada una de las posiciones en las filas y columnas de la matriz

Decidimos usar 64 leds, ya que no entendíamos como funcionaban las conexiones internas en las tiras LED nos pareció mas factible no tomar ese camino ya que podría conllevar a confusiones.

Como vamos a utilizar 64 LEDs, en nuestro algoritmo vamos a utilizar un arreglo de tamaño 64 para representar cada LED.

2) Usando ciclos que enciendan cada led uno por uno y luego se apagaran, ya que solo necesitamos verificar que encienda, por lo tanto, no hay necesidad de que queden encendidos.

O también podrías simplemente pasar todos los estados de los LEDS a encendido y después de un tiempo que se apaguen.

3) Para este punto los problemas que presentamos fueron el cómo el usuario ingresara su patrón y el cómo ese patrón lo procesara el algoritmo en nuestro arreglo de tamaño 64, para que pueda mostrarse en la matriz de LEDs.

Para resolver este problema podríamos pedirle al usuario que ingrese su matriz de la siguiente manera:  
  
00000000 Los (0) representan los LEDs apagados y los (\*) respresentan los LEDs

\*\*\*\*\*\*\*\* encendidos.

00000000

\*\*\*\*\*\*\*\*

00000000

\*\*\*\*\*\*\*\*

00000000

\*\*\*\*\*\*\*\*

La idea sería crear un algoritmo que procese toda la matriz de 8x8, la ordene en nuestro arreglo de tamaño 64 dependiendo de que LEDs estarían encendidos en el patrón del usuario, y luego muestre el patrón deseado en la matriz de LEDs.

Otra forma seria que el usuario ingrese Fila por Fila su patrón deseado, esta fila se guardaría en un arreglo de tamaño 8, y luego con un ciclo este se iría almacenando en nuestro arreglo de tamaño 64, una vez que proceso ese arreglo, le pediría al usuario que ingrese la siguiente fila, hasta terminar con la fila 8, y así con el patrón deseado.

4) Para este punto pensamos en cómo se relacionan los patrones en sus filas y columnas, notamos que en los patrones algunas filas son las mismas y en general este problema tenía un parecido a prácticas pasadas, entonces se nos ocurrió que por medio de ciclos y contadores podías realizar las primeras 4 filas, luego realizando el proceso inverso, terminar las ultimas 4 filas, así teniendo el patrón completo, sin tener que recurrir a solución trivial.

5) en este punto debemos crear un menú con las siguientes opciones:  
  
a. Verificar funcionamiento de LEDs. Indicando cuanto tiempo entre encendido y apagado y cuantas secuencias.

b. Mostrar imagen de prueba. Ingresada por el puerto serial y prendiendo y apagando según el tiempo indicado por el usuario

c. Mostrar de forma alternada los patrones 1 al 4. Indicar el tiempo de retardo entre visualizaciones. Los patrones se deben generar a partir de secuencias como en el punto 4.

El funcionamiento del algoritmo del menú seria:

1. Que imprima las opciones disponibles, junto a las anteriores (a, b y c), que tenga una opción para cerrar el menú.
2. Que le pida al usuario que ingrese un numero para que pueda escoger la opción que corresponde a dicho número.
3. Dependiendo de la opción del usuario se invoquen las funciones correspondientes.
4. Una vez finalizada la opción ingresada por el usuario, le pregunte nuevamente si desea volver al menú inicial, si desea repetir esta opción, o si desea terminar el programa.

Ahora, empezando con las investigaciones para resolver los problemas planteados en este proyecto nos hicimos las siguientes preguntas:

**¿Cómo es el funcionamiento interno del integrado 74HC595?**

Con nuestras palabras, el integrado recibe datos en serie los cuales pueden ser 0 y 1, indicando así el estado LOW y HIGH para los LEDs, esos datos en serie al momento de soltarlos, lo hace en paralelo, esta es una ventaja ya que permite encender ciertos leds en específico que le indiquemos por los datos ingresados en cadenas de 8 bits como máximo. Principalmente se conectan tres pines digitales del Arduino al integrado los cuales serían los siguientes:

1) El pin de Data, que es por donde son ingresados los datos deseados.

2) El pin Clock de desplazamiento, que es el que desplaza los datos.

3) El pin Latch, que es el que los lanza a los pines de salida del integrado.

Ahora, el integrado funciona en tres bloques principalmente, el primero que es el registro de desplazamiento, este recibe los datos en serie y junto con los Clocks los va desplazando de uno en uno, después le sigue el registro de almacenamiento, que recibe la cadena de datos que están en la entrada del registro de almacenamiento, pasen y los almacena en la salida del registro de almacenamiento, en otras palabras, los posiciona en sus pines de salida correspondientes y por último con el pin Latch, se ejecuta la orden y se encienden los LEDs a los que se les indico 1 en la cadena de datos almacenados en cada pin de salida del integrado.

Después nos preguntamos **¿Cómo podíamos conectar varios integrados 74HC595 en serie?**

Para lograrlo, tenemos que hacer que la salida del integrado anterior, sea recibida por la entrada del integrado siguiente, es decir, el dato almacenado en el pin de salida de un integrado no se quedara allí, si no que viajara directamente al pin de entrada del siguiente integrado, lo que extiende la cadena de datos de 8 bits a 16 bits, y así sucesivamente.

Para el resto de las otras conexiones, se hacen de manera similar, pero esta vez siguiendo un orden regular, es decir, el registro de salida del anterior integrado, se conectará también al registro de salida del siguiente integrado y así con los demás.

Luego nos preguntamos, **¿Cómo el usuario puede interactuar con el circuito?**, para esto tenemos el monitor serial que es una función o ventana en el entorno de desarrollo Arduino (IDE) que permite la comunicación entre un Arduino y una computadora a través del puerto serie. Es una herramienta fundamental para depurar y visualizar información durante el desarrollo y la ejecución de programas en Arduino. Esto nos permitiría crear el menú con las funciones que usaríamos para solucionar los problemas planteados en este proyecto. También mientras investigamos encontramos que podríamos realizar un menú sin la necesidad del puerto serial, por medio pantallas LCD y botones para que el usuario pueda seleccionar la opción del menú que desee, aun que para este proyecto usaremos el puerto serial.

De lo anterior surgió otra pregunta, **¿cómo podemos hacer un menú utilizando el puerto serial?**, nos pareció que antes de intentar crear los algoritmos que resolverían los problemas planteados en este proyecto, primero teníamos que saber cómo crear un menú, para esto tenemos varias instrucciones que podemos utilizar.

Con **Serial.begin();** es una instrucción que se utiliza en Arduino para iniciar la comunicación serial entre el Arduino y un dispositivo cualquiera que se conecte al puerto serial del Arduino, y se usa comúnmente en el bloque **setup()** para configurar la comunicación serial antes de que el Arduino entre en su bucle principal de ejecución.

Para configurar la razón de datos o bits por segundo (baudios) que se utilizará en la comunicación serial, ponemos el número de baudios que deseamos usar, la más usual es 9600 que nos garantiza una comunicación confiable y sincronizada entre los dispositivos, por lo que nos queda la instrucción **Serial.begin(9600);**

Con **Serial.println(valor, formato);** esta instrucción imprime datos del Arduino al puerto serial para ser vistos en el monitor serial. **Valor** se refiere al dato a imprimir y el **formato** se refiere a expresar el valor en un formato especifico que pueden ser BIN, OCT, DEC, HEX. Si en la instrucción solo ponemos el **valor**, el **formato** será en ASCII, y esta instrucción salta a la siguiente línea una vez imprime el valor, a diferencia de la instrucción **Serial.print(valor, formato);** que cada vez que yo la utilice me seguirá imprimiendo en la misma línea. Con esta instrucción podremos imprimir cadenas de caracteres que representaran mensajes que el usuario usara para ver las opciones del menú.

Con **Serial.available()** es una función en Arduino que se utiliza para verificar si hay datos disponibles para ser leídos en el puerto serial. Devuelve el número de bytes disponibles para lectura en el bufer del puerto serial. Esta instrucción es útil cuando se desea leer datos del puerto serial de manera segura sin bloquear el programa mientras espera que lleguen datos, ya que esta instrucción permite determinar si hay datos pendientes antes de intentar leerlos, para esto utilizamos un if para evaluar la instrucción, que nos arrojara 0 cuando no hay datos en el puerto serial.