Parcial 1

Desarrollo Parcial

Sergio Giraldo Salazar y Julio César Benavides Hernández

Departamento de Ingeniería Electrónica y Telecomunicaciones
Universidad de Antioquia
Medellín
Septiembre de 2023

Índice

1. Análisis de la Implementación:

- Matrices Dinámicas: La implementación utiliza matrices dinámicas (punteros a punteros) para representar las matrices en lugar de matrices estáticas de tamaño fijo. Esto permite crear matrices de tamaño variable en tiempo de ejecución y evita problemas de límites predefinidos, además hace parte de los requisitos mínimos impuestos en este parcial.

- Funciones Genéricas: Las funciones inicializarMatriz e imprimirMatriz son genéricas y se pueden reutilizar para trabajar con matrices dinámicas en otros programas. Esto promueve la reutilización de código y la modularidad.

- Manejo de Memoria Dinámica: Se utiliza new para asignar memoria dinámica a las matrices y delete para liberar la memoria al final de cada función. Esto es esencial para prevenir fugas de memoria y garantizar una gestión adecuada de los recursos.

- Algoritmos Específicos para Patrones: Cada función imprimirPatronX implementa un algoritmo específico para llenar la matriz con un patrón particular. Estos algoritmos son claros y se pueden modificar fácilmente para crear patrones diferentes.

- Llamadas de Funciones en main(): El programa principal (main()) llama a las funciones que imprimen los patrones en un orden específico. Esto facilita la organización y mantenimiento del código.

2. Propuesta de Solución:

La solución proporcionada es efectiva para imprimir lospatrones en una matriz dinámica. Algunos aspectos clave de esta solución son:

- Modularidad: El código está dividido en funciones, cada una con una tarea específica. Esto facilita la comprensión y el mantenimiento del código, así como la reutilización de funciones en otros programas.

- Memoria Dinámica: El manejo de la memoria dinámica se realiza adecuadamente. Se asigna memoria para las matrices cuando es necesario y se libera cuando ya no se necesita, lo que previene fugas de memoria.

- Flexibilidad: Las funciones imprimirPatronX son flexibles y se pueden modificar para crear diferentes patrones. Esto permite extender la funcionalidad del programa para imprimir patrones personalizados.

- Claridad y Legibilidad: El código está bien comentado y sigue una estructura lógica. Los nombres de las variables y funciones son descriptivos, lo que facilita la comprensión del código.

Mejoras posibles:

- Parámetros de Función: En lugar de pasar las dimensiones de la matriz como parámetros en cada función imprimirPatronX, podrías definirlas como constantes globales o variables globales para evitar repetir los mismos valores en varias partes del código.
- Optimización: Si necesitas imprimir matrices muy grandes o realizar operaciones más complejas, es posible que desees considerar optimizaciones de rendimiento, como evitar copias innecesarias de matrices.

En general, esta solución es sólida y cumple su propósito de imprimir patrones en matrices dinámicas de manera clara y eficiente.

3. Implementación del algoritmo

El algoritmo se centra en la generación e impresión de patrones en una matriz dinámica de caracteres. A continuación, se presenta una descripción paso a paso del algoritmo:

3.1. Objetivo:

Generar y mostrar patrones en una matriz de 8x8 caracteres en C++, luego implementar un circuito funcional de esto en TINKERCAD.

1. **Paso: Inicialización de la Matriz**

   Se inicia la función inicializarMatriz, que toma tres argumentos: una matriz dinámica (puntero a puntero de caracteres), el número de filas y el número de columnas (en este caso, 8x8).

   La función utiliza un bucle para crear un arreglo de punteros, donde cada puntero representa una fila de la matriz.

   Luego, se crea un bucle anidado para cada fila y se asigna memoria para un arreglo de caracteres (columnas) en cada fila.

   Todos los elementos de la matriz se inicializan con espacios en blanco ' '.

2. **Paso: Generación del Patrón**

   El algoritmo para generar el patrón específico se implementa en la función imprimirPatronX (donde X es 1, 2, 3 o 4).

   En el caso del Patrón 1, el algoritmo llena la matriz con asteriscos '*' siguiendo un patrón específico, aumentando y disminuyendo la cantidad de asteriscos en cada fila.

   Para el Patrón 2, se colocan asteriscos '*' en las diagonales de la matriz, creando un patrón simétrico.

El Patrón 3 llena la matriz con asteriscos '*' en dos mitades, una en la parte superior y otra en la parte inferior.

El Patrón 4 llena la matriz con asteriscos '*' en dos mitades, una en la parte superior y otra en la parte inferior

3. **Paso: Impresión de la Matriz**

   Una vez generada la matriz con el patrón, se llama a la función imprimirMatriz.

   La función imprimirMatriz recorre la matriz y muestra cada carácter en la consola, fila por fila, con saltos de línea entre las filas.

4. **Paso: Liberación de Memoria**

   Al final de cada función imprimirPatronX, se realiza la liberación de memoria dinámica. Se recorren las filas de la matriz y se libera la memoria asignada para cada fila, y luego se libera la memoria del arreglo de punteros de filas.

5. **Paso: Llamada desde 'main()'**

   En la función main(), se llama a las funciones imprimirPatronX en el orden deseado para imprimir los patrones en la consola.

A continuación, mostraremos un posible codigo de cómo mandar comandos por el puerto serial y utilizarlos para mostrar patrones y figuras. En este utilizaremos un programa de terminal de computadora para enviar comandos al Arduino a través de la comunicación serial.

4. Componentes

- Arduino Uno o cualquier placa Arduino compatible.
- Protoboard o placa de circuito impreso.
- Integrado 74HC595 (1 o más, dependiendo de la cantidad de columnas de la matriz).
- Resistencias de 470 ohmios.
- Cables jumper macho-macho y macho-hembra.
- LEDs (Matriz de LEDs 8x8).

5. Desarrollo del circuito

Para resolver el trabajo pedido empezamos organizando el circuito el cual permitirá dar solución a lo pedido y a su vez posibilitara que el software haga su función de manera correcta, por otro lado y paralelamente a lo anterior la implementación en c++ del código respectivo.

1. Primero organizamos la matriz de leds 8x8 y buscamos con que resistencias se podia trabajar para que los leds pudieran encender al máximo (**??**).

2. Luego organizamos los circuitos integrados conectandolos al arduino y también a la matriz de leds (ver Figura **??**).

3. Después de probar que todos los leds funcionaran correctamente, vimos la necesidad de agregar transistores junto con una resistencia de 1k ohm a las filas de la matriz de leds que corresponden a los catodos de los leds y a su vez las resitencias que se conectan a los anodos las intercambiamos por resistencias de 1k ohm. También decidimos cambiar los leds por leds de color rojo para mejorar la intensidad de brillo de estos (ver Figura **??**)).

5.1. Conexión de la matriz de LEDs y el 74HC595:

En la investigación previa del 74HC595 se hallo que este hacia un registro de desplazamiento: conviertiendo los datos en serie en salida paralela ademas, Con el 74HC595 ocupa 3 salidas digitales del Arduino pero, obtendremos 8 salidas digitas adicionales (ganamos 5 salidas) por lo que en tinkercad implementamos un diseño con 8 leds en serie para un total de 8 filas de leds (8*8).

Poniendo más chips conectados en serie se pueden obtener otras 8 salidas más por cada chip agregado, y la cantidad de pines ocupados en el Arduino sigue igual: solamente tres. Con tres chips tendremos 24 salidas, con 8 chips tendremos 64, y con 32 chips tendremos una ampliación de 256 nuevas salidas.

1. Conectaremos los terminales positivos (ánodos) de los LEDs de la matriz a través de las resistencias de 220 ohmios a las salidas Q0 a Q7 del 74HC595.

2. Conectaremos el terminal común de los cátodos de los LEDs de la matriz a tierra (GND).

3. Conectaremos el pin DS (Serial Data) del 74HC595 al pin digital 11 de Arduino.

4. Conectaremos el pin SH_CP (Shift Register Clock Pin) del 74HC595 al pin digital 12 de Arduino.

5. Conectaremos el pin ST_CP (Storage Register Clock Pin) del 74HC595 al pin digital 13 de Arduino.

6. Conectaremos el pin MR (Master Reset) del 74HC595 a 5V (VCC) para desactivar la función de reset.

asumiremos que enviaremos comandos desde una computadora al Arduino a través de la comunicación serial (USB) para dibujar patrones y figuras en una matriz de LEDs 8x8 o cualquier otro dispositivo con matrix led 8x8 de salida que se quiera controlar. Aquí una posible guía de cómo hacerlo:

Configuración del Hardware:

1. Conectaremos el Arduino al puerto USB de tu computadora.
2. Conectaremos la matriz de LEDs a los pines adecuados de tu Arduino. Asegúrarnos de tener la configuración hardware y las conexiones requeridas.

6. Código Arduino y Control de Patrones

En este proyecto, utilizaremos Arduino para controlar una matriz LED 8x8 mediante el uso del 74HC595. A continuación, se detallan los pasos necesarios:

- Utiliza la biblioteca 'shiftOut' de Arduino para enviar datos a los registros 74HC595. Puedes consultar la documentación de Arduino para aprender cómo usar esta biblioteca.
- Adapta el código anterior para que, en lugar de imprimir en la consola, envíe los datos necesarios a los registros 74HC595 para encender los LEDs correspondientes en la matriz LED. Debes tener en cuenta que, debido a la limitación de recursos de Arduino, es posible que debas dividir el proceso en varias etapas si la matriz es grande.
- Control de Patrones: Define los patrones que deseas mostrar en la matriz LED en el código de Arduino. Crea una lógica que controle cuándo y cómo se cambian los patrones en la matriz.
- Compilación y Carga: Compila el código de Arduino y cárgalo en tu placa Arduino utilizando el IDE de Arduino.
- Prueba: Verifica que la matriz LED muestre los patrones deseados de acuerdo con tu código.

Este proyecto combina hardware y programación para controlar una matriz LED 8x8 utilizando Arduino y el 74HC595. Puedes expandir este proyecto agregando botones u otros sensores para interactuar con la matriz LED y crear efectos visuales interesantes. Además, puedes utilizar bibliotecas específicas para matrices LED para simplificar el control y la animación de los patrones.

7. Conclusión

En conclusión, podemos afirmar que, a pesar de no ser la manera más óptima en términos de componentes, presupuesto, tiempo y desarrollo, con conocimientos básicos en programación y circuitos, como los impartidos en clase, es posible alcanzar soluciones que cumplan con los requisitos del usuario. Este caso particular se refiere al desarrollo de una matriz de 64 LEDs utilizando Arduino, el

74HC595 y la programación de memoria estática y dinámica, así como el uso de arrays.

A pesar de que podría haber enfoques más eficientes en términos de recursos y tiempo, este proyecto demuestra que incluso con recursos limitados y un conocimiento básico, es posible crear una solución funcional que cumpla con las necesidades del usuario. Esta matriz de LEDs para un anuncio publicitario es un ejemplo de cómo se pueden combinar habilidades de programación y electrónica para lograr un objetivo específico. A medida que se adquieren más conocimientos y experiencia, es posible refinar y optimizar aún más estos proyectos, pero esta implementación inicial es un valioso punto de partida.

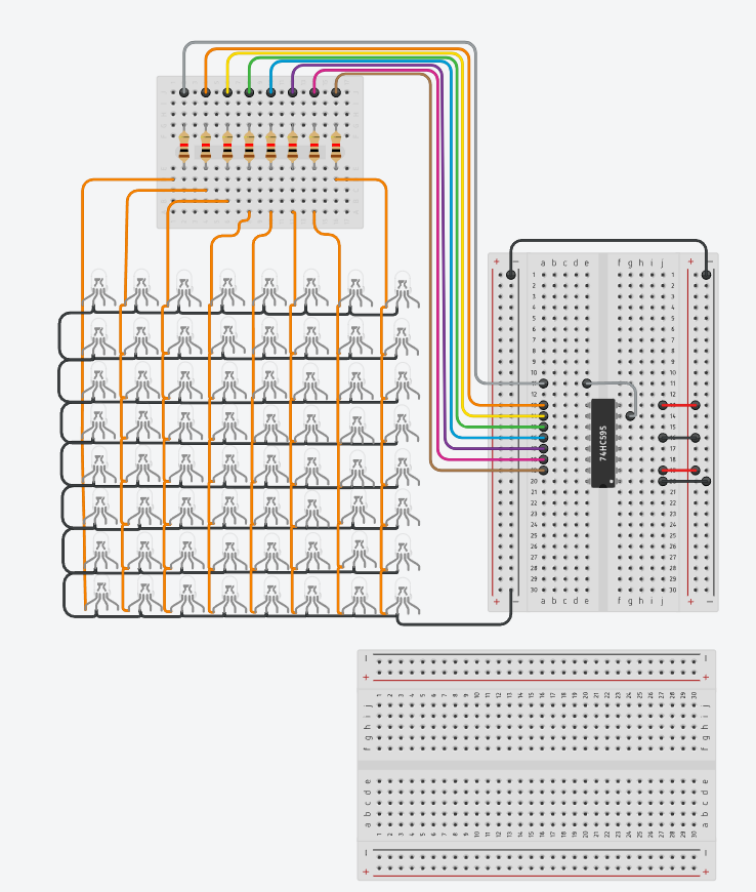

Figura 1: Matriz de LEDs

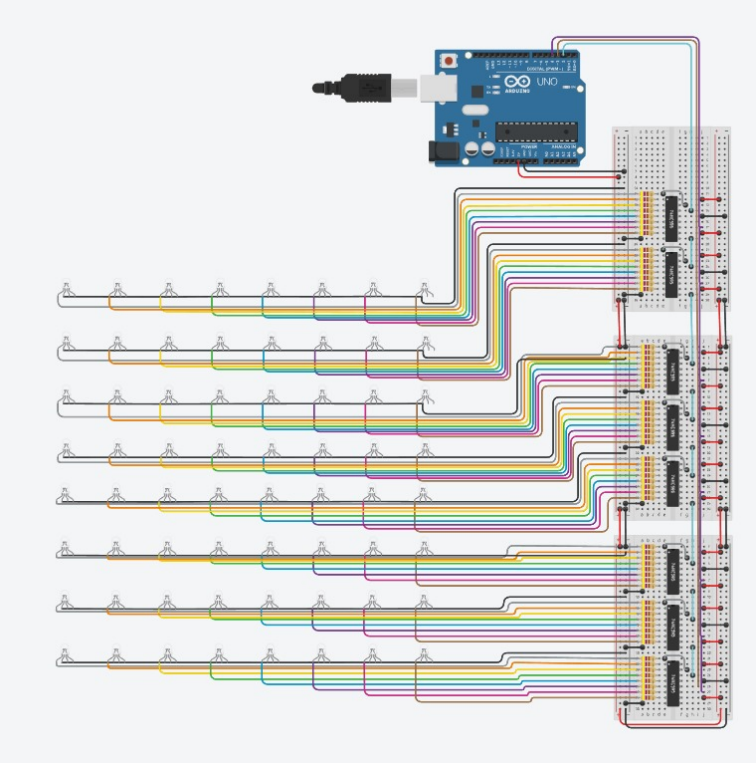

Figura 2: Circuito finalizado