INFORME DE PARCIAL I

Autores: *Cristian Flórez Grisales, Natalia Polo Peña*

Informática II

Departamento de Ingeniería Electrónica y de Telecomunicaciones

Universidad de Antioquia

Resumen

Una matriz de LEDs es un conjunto de leds colocados en filas y columnas que se pueden encender y apagar individualmente. Se utiliza para crear efectos de luz, mostrar imágenes o mensajes. Para controlar una matriz de LEDs, se necesita aplicar los valores HIGH y LOW a cada fila y columna según el led que se quiera activar.

Palabras clave: matriz, led, patrón, Arduino, registro.

Introducción

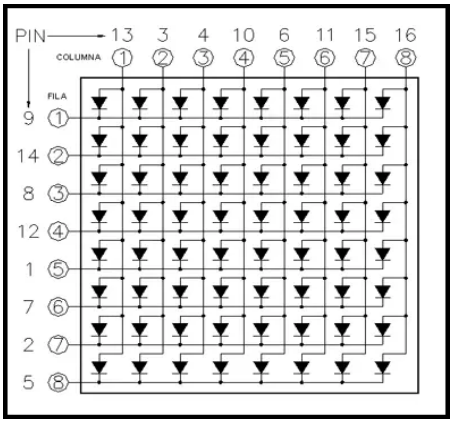
En este informe que corresponde al primer parcial de Informática II, se lleva a cabo la programación de una matriz de leds de 8x8, en la cual se imprimen cuatro patrones predeterminados, así como también patrones ingresados por el usuario.

El objetivo de realizar esta práctica es desarrollar la capacidad de solución de problemas de la vida cotidiana trabajando con Arduino e integrando la programación en C++.

Marco teórico

Matriz de leds

Las matrices de LEDs están conformadas por una serie de filas y columnas en donde hay un LED en cada una de las intersecciones. Para que un LED encienda se deberá recibir un «0» en la fila y un «1» en la columna simultáneamente.



LED

Diodo emisor de luz. En su interior hay un semiconductor que, al ser atravesado por una tensión continua, emite luz, lo que se conoce como electroluminiscencia.

Multiplexación

Es una técnica que permite usar pocos pines de entrada y salida del microcontrolador. Se utilizó dos tipos:

Latch: es una memoria que mantiene un valor en sus salidas hasta que se le indique. De esta forma se encienden los leds rápidamente y por turnos para formar una palabra o una secuencia.

Registro de desplazamiento: funciona de la misma manera que una fila. Por un extremo ingresan los datos 0,1 y del otro extremo van saliendo. Se necesitan solamente tres pines:

DATA: envía los datos al registro ya sea un <<0>> o un <<1>>.
CLOCK: avisa al registro que el dato ya está listo para ingresarse.
RESET: es el que vacía la fila al escribir un <<0>> en todas las salidas.

Análisis del problema

Analizando la matriz de Leds es importante saber cómo funcionan cada led interno, cada led funciona como un registro de desplazamiento (shift register) y mueve datos de la entrada a la salida y luego al siguiente led.

Para esto se consideró hacer uso del integrado 74HC595. Este registro se compone de una serie de biestables o flip-flops de tipo D comandados por una señal de reloj. Esos biestables son memorias que mantienen un valor anterior. Cada uno almacena un bit y, de su nombre también se puede deducir que, los puede desplazar. Al correr los bits de un lado a otro se pueden hacer operaciones digitales, en este caso, imprimir patrones de figuras.

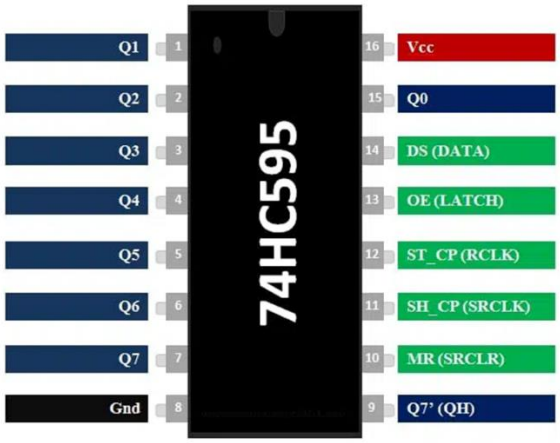
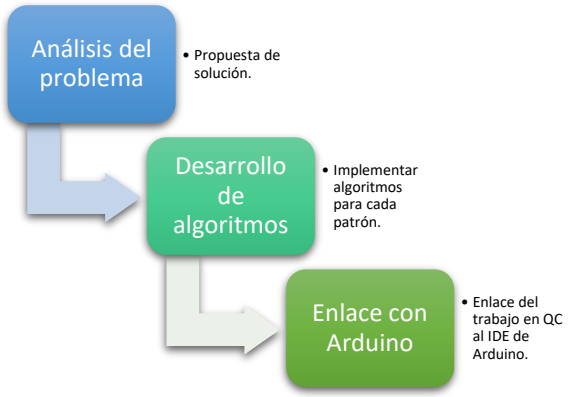


Figura 2. Registro de desplazamiento 74HC595.

En la figura anterior, se tiene el esquema del integrado, en el cual, se indican los pines correspondientes de entrada y salida de datos.

Esquema de tareas



Algoritmos implementados

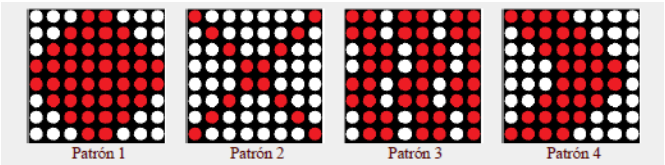


Figura 3. Patrones

Para imprimir los patrones no utilizamos la solución trivial que consiste en indicar los leds que deben encenderse y apagarse, sino que identificamos las secuencias que se generan en cada patrón, teniendo en cuenta que el patrón se visualiza en forma de matriz, por lo que, es posible realizar una relación de posiciones que se puedan recorrer para formar el patrón.

Para esto, se utilizaron punteros, que son variables que almacenan direcciones de memoria en lugar de valores directos, en conjunto a la memoria dinámica para la asignación y liberación de memoria en tiempo de ejecución. Por eso se hace uso de punteros, ya que se utilizan para acceder y gestionar la memoria dinámica.

Cuando se necesita crear un objeto o una estructura de datos en tiempo de ejecución, se asigna la memoria dinámica utilizando los operadores “new” en C++.

Para realizar la liberación de memoria o evitar pérdidas se utiliza el operador “delete”.

Problemas de desarrollo

1. Errores en la declaración de variables y funciones.
2. Errores en el manejo de arrays multidimensionales.
3. Complicaciones en las conexiones del circuito para conectar la matriz.
4. Problemas para proyectar los patrones en la matriz.

Evolución de la solución y consideraciones a tener en cuenta en la implementación

Durante el desarrollo se consideró estudiar con mas profundidad el funcionamiento del registro, ya que de este dependía la proyección de los patrones en la matriz. Por otro lado, se corrigieron los errores en cuanto a la estructura del programa, siendo totalmente funcional a nivel de codificación.

Bibliografía