Universidad de Antioquia
Departamento de I. Electrónica y Telecomunicaciones
Correa Morales Robinson

Primer examen parcial

1 - Planteamiento del problema:

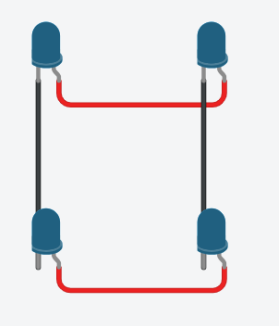
Se requiere construir una matriz de leds con dimensiones de 8 filas por 8 columnas que sea capaz de mostrar una serie de patrones. Para ello se debe hacer uso de un microcontrolador Arduino UNO y el lenguaje de programación C++ además de los conocimientos adquiridos en clase e investigación autónoma.

2 - Análisis del problema:

Al abordar el problema me he percatado de que tanto el hardware como el software están muy vinculados entre sí. Esto me ha llevado a tener presentes ambos frentes de tal manera que se pueda generar una solución adecuada, evitando perder tiempo en reestructurar gran parte del análisis porque las soluciones de hardware y software son incompatibles entre sí.

2.1 – Abordando el hardware:

Una matriz es un conjunto de elementos distribuidos en forma de filas y columnas, en este caso, leds. Para controlar los leds de manera individual se pueden organizar sus terminales eléctricas de modo que estas formen un plano ordenado, de otra forma se podría contar con un número muy alto de terminales eléctricas (64 terminales positivas). Para solucionar esto, pueden conectar todos los ánodos de una misma fila teniendo así una única terminal positiva por cada fila de leds y conectar los cátodos de una misma columna para tener una sola terminal negativa por columna. (figura 1).



(figura 1. Rojo: ánodo, negro: cátodo)

Con esta disposición se reduce de 64 terminales positivas a tan solo 16 terminales entre positivas y negativas, que deben de poder ser controladas de manera individual. Esto genera un nuevo dilema pes la plataforma Arduino UNO posee una cantidad de conectores limitados y para el desarrollo de este parcial solo pueden ser usadas un máximo de 7 salidas digitales.

Este nuevo problema se puede resolver utilizando el **registro de desplazamiento 74HC595,** el cual nos permite transformar una **entrada de ocho bits en serie** a ocho **salidas individuales en paralelo,** es decir, el IC (circuito integrado) recibe un byte y lo transforma en ocho bits individuales.

Esto supone una gran ventaja, como explica del Valle (s.f.) en "*Programafacil.com"*, nos permite transformar tres salidas digitales del Arduino UNO en ocho salidas

digitales, además de poder conectarse varios de estos en serie aumentando en 8n salidas digitales con solo 3 salidas digitales del Arduino UNO.

Para resolver el problema inicial, se hará uso de 2 registros de desplazamiento 74HC595, uno controlará los ánodos de los leds y el otro los cátodos.

2.2 – Abordando el software:

Para controlar la imagen se planea generar un conjunto de 6 arreglos tipo *char* (de un byte de longitud) con ocho caracteres de longitud cada uno. Cada arreglo representará una figura distinta, el sexto arreglo tendrá los valores = {128, 64, 32, 16, 8, 4, 2, 1} que representan las filas a encender en un momento determinado generando un efecto "barrido", esto se debe a que la solución de hardware no puede mostrar la imagen completa en un solo instante, requiriendo de varios ciclos para realizarlo.

Para evitar la *solución trivial*, estos arreglos serán generados en tiempo de ejecución:

- Se harán 5 funciones, una por cada patrón, diseñadas para generar el arreglo con la información de la columna requerida para reproducir en la matriz cada patrón según sea solicitado (se incluye la función *"verificación"* solicitada en la guía).

- Se hará una función llamada *dibujador*, encargada de recibir una tupla de bytes y dibujarla en la matriz.

- Se hará una función llamada *"imagen"* que se comunicará con la función *"publik"* y la función *"dibujador"*, esta función permitirá al usuario ingresar un patrón arbitrario 8 x 8 para luego mostrarlo en la matriz.

- Se hará un pequeño programa que permita seleccionar entre distintos modos (Verificar funcionamiento de los leds, mostrar un patrón arbitrario, mostrar los patrones de forma alternada), este programa será la función *"publik"*.

3 – Esquema

Para la solución se analizó primero la manera de mostrar uno de los patrones en la matriz. A continuación, se presenta un esquema donde se muestran los momentos más relevantes del desarrollo (figura 2).

Figura 2, esquema

4 – Documentaciones

4.1 – Registro de desplazamiento 74HC595:

Es un registro de desplazamiento de ocho bits de entrada serie y salida en paralelo (figura 3).

Los bits ingresan al circuito en sincronía con una señal de reloj, que le indica cuando debe leer un bit. Los bits se van desplazando en el registro (registro de desplazamiento), de modo en que el primer bit será el último y el último bit ingresado será el primero.

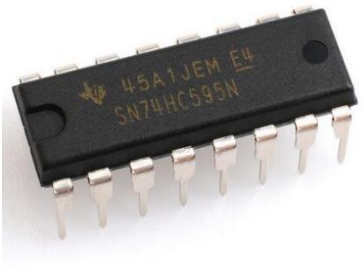


Figura 3 74HC595

Entradas y salidas:

1, 2, 3, 4, 5, 6, 7	Pines de salida (de Q1 a Q7)
8	tierra
9	(P7) Salida en serie
10	(MR) Restablecimiento maestro
11	(SH_CP) Reloj
12	(ST_CP) Pestillo
13	(OE) Activación de salida
14	(DS) Datos en serie
15	(P0) salida
16	Voltaje +5v

(Damian, 2021).

Para la solución, se hizo uso de la función shiftOut de la librería estándar de Arduino que permite simplificar las comunicaciones con el registro de desplazamiento. Esta función recibe como argumentos el pin de datos, el pin de reloj, el sentido del byte a transmitir, si desde el bit más significativo (MSBFIRST) o desde el menos significativo (LSBFIRST) y por último, el byte a transmitir.

Para activar la salida de los datos en paralelo, utilizamos un pulso HIGH en el pin 12 (pestillo) y luego un pulso LOW.

4.2 – Funciones:

4.2.1 Dibujar:

Entradas: Apuntador a char sin signo, apuntador a char sin signo.

Salidas: Ninguna (void).

Esta función llama a la función **shiftOut** para enviar los dos bytes necesarios para la impresión en la matriz utilizando multiplexación**.**

4.2.2 Funciones time y repetición:

Entradas: Ninguna (void).

Salidas: Entero sin signo.

Estas funciones son funciones de input, es decir reciben información del usuario, la validan y la transmiten como retorno.

4.2.3 Funciones patron0 a patron4:

Entradas: Ninguna(void).

Salidas: Apuntador a arreglo de char sin signo.

Estas funciones generan un arreglo dinámico de ocho posiciones que representan las columnas de cada una de las figuras. Los patrones se van formando a través de operaciones a nivel de bits, en concreto, las operaciones OR, desplazamiento y negación (NOT).

4.2.4 Función patronUser:

Entradas: Ninguna (void).

Salidas: Ninguna (void).

Esta función genera un arreglo dinámico de ocho posiciones, el cual se llena con los valores ingresados por el usuario (enteros positivos entre 0 y 255), los cuales representan una secuencia binaria que se puede utilizar para dibujar en la matriz cualquier imagen de dimensión 8 x 8.

4.2.5 Función borrarPatron:

Entradas: Apuntador a char sin signo.

Salidas: Ninguna (void).

Esta función libera la memoria reservada en el montón (heap) por las funciones patrón, incluyendo la función ***patronUser.***

4.2.6 Función dibujarEstatico:

Entradas: entero, entero**,** apuntador a char sin signo.

Salidas: Ninguna (void).

Esta función utiliza la medición del tiempo del microprocesador para determinar si debe o no ocurrir un evento. Esta función se encarga de dibujar los patrones estáticos, como el patron0 (verificación) o el patrón personalizado por el usuario.

4.2.7 Función dibujarSecuencia:

Entradas: entero, entero**.**

Salidas: Ninguna (void).

Esta función se encarga de mostrar en la matriz cada uno de los patrones predefinidos del 1 al 4**.** Se hace uso de la medición del tiempo para controlar cuando debe de cambiarse el patrón mostrado.

4.2.8 Función verificación:

Entradas: Ninguna (void).

Salidas: Ninguna (void).

Función que se encarga de encender los 64 leds dado un tiempo y unas repeticiones por el usuario.

4.2.9 Funcion patronArbitrario:

Entradas: Apuntador a entero sin signo, apuntador a char sin signo.

Salidas: Ninguna (void).

Esta función pasa los elementos de un arreglo de enteros a un arreglo de char con los bits negados.

4.2.10: función secuencia:

Entradas: Ninguna (void).

Salidas: Ninguna (void).

Esta función se encarga de invocar a las funciones necesarias para mostrar los patrones del 1 al 4.

4.2.11 Función publik:

Entradas: Ninguna (void).

Salidas: Ninguna (void).

Esta función se encarga de invocar las funciones principales y cuenta con un pequeño menú para elegir entre tres modos diferentes: Verificación, ingresar un patrón o mostrar los patrones en secuencia.

5 – Problemas encontrados

En la implementación se encontraron diversos problemas, entre ellos, el manejo del tiempo y el uso de la función **delay**. Puesto que se necesita que el Arduino esté procesando en todo momento, el uso de delay es completamente incorrecto ya que detiene la ejecución en el procesador durante un tiempo. Al contrario, se necesita que el código se ejecute lo más rápido posible para que la matriz funcione con suavidad. Se solucionó esto implementando la función **millis** y midiendo el tiempo antes de ejecutar una tarea, comprobar el tiempo transcurrido en un condicional y volverlo a actualizar si es necesario. Se debe tener presente que millis puede generar desbordamientos si los tiempos son muy grandes (aprox. 50 dias).

Se encontró en los primeros días de desarrollo cuando se estaba planeando la estrategia de solución, que el diseño de una matriz generada como habitualmente se generarían en un programa para otras aplicaciones (como la guía de laboratorio 2), donde la matriz es generada por dos iteradores; es sumamente ineficiente para un sistema tan limitado como Arduino por lo que se optó por cambiar el diseño a operaciones de bits.

También se encontraron problemas a nivel de hardware, poco brillo en los leds, patrones invertidos (los leds encendidos deberían estar apagados y viceversa) y al usar los registros de desplazamiento tales como “roturas por sobrecarga”. Estos problemas fueron solucionados a través de la investigación y asesorías con el monitor del curso y el profesor de laboratorio.

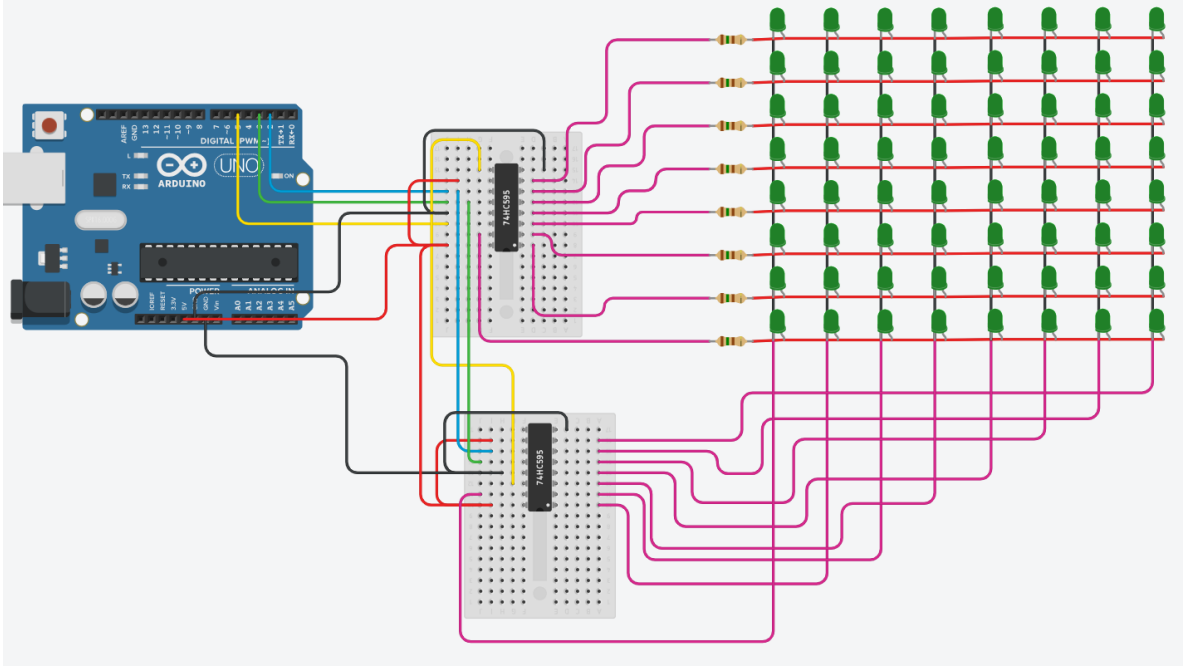
6 – Evolución

Una vez dado el enunciado por parte de los profesores, se idearon algunas soluciones que permitieran conectar la matriz al Arduino, puesto que no se podían usar más de 7 salidas digitales y porque es imposible conectar 64 leds de manera directa al Arduino UNO y controlarlos de manera independiente, se consultó sobre el registro de desplazamiento y como se pueden anidar en cascada para aumentar las salidas digitales.

Se consultó sobre la multiplexación, en concreto la multiplexación por división de tiempo. El Arduino envía la comunicación al último registro de desplazamiento (columnas), luego el Arduino envía los datos al segundo registro (filas) y después se muestra la información en las salidas de ambos registros. Esto solo enciende una sola fila, para encender la matriz, un ciclo cambia los datos a enviar a los registros en cada iteración pero ocurre tan rápido que genera la ilusión de una imagen estática.

La solución evolucionó sin mayores novedades hasta el momento de ensamblar las funciones que requerían el uso del manejo del tiempo, ya que se intentó realizarlo con la función delay, la cual no trajo resultados positivos, se consultó en internet a cerca de la función millis y se buscó asesoría sobre el tema con el profesor de laboratorio quien mostró en un ejemplo como usar millis de manera correcta.

A continuación, se presenta el montaje final (figura 4)



7 – Referencias

Damian, J. (2021). *Cómo funciona el 74HC595 Shift Register y su interfaz con Arduino*. Obtenido de https://www.electrogeekshop.com/como-funciona-el-74hc595-shift-register-y-su-interfaz-con-arduino/

Hernández, L. d. (s.f.). *74HC595 registro de desplazamiento con Arduino*. (Programarfacil.com, Productor) Obtenido de https://programarfacil.com/blog/74hc595-registro-de-desplazamiento-arduino/#Que_es_el_registro_de_desplazamiento_74HC595