PARCIAL DE INFORMATICA 2

PRESENTADO POR:

Mario Estrada

Juan Ballesteros

PRESENTADO A:

Aníbal Guerra

UNIVERSIDAD DE ANTIOQUIA

CONTENIDO

1. Análisis del problema

1. Investigación
2. Planteamiento de Solución

1. Esquema de tareas

1. Algoritmos

1. Complicaciones encontradas

1. Solución final

1. Evidencias de Implementación

1. ANALISIS PROBLEMA

Se requiere un Programa que permita generar patrones en una matriz de LED’s de 8 x 8 mediante el uso de la placa Arduino UNO R3, esta placa puede ser programada mediante el leguaje C++, lenguaje que se esta usando en el curso de Informática II.

Los patrones deben ser mostrados en la matriz mediante el uso de funciones, donde el encendido de los LEDs debe ser mediante algoritmos no triviales, es decir primero se deben identificar patrones los cuales se van a usar en el momento de la programación en el software de desarrollo Qtcreator.

Para poder realizar una mejor comprensión del problema se procede a aplicar la técnica de “divide y vencerás “, por ende, dividiremos este problema en varios.

1 ¿Como conectar los 64 LEDs para luego poder manipularlos?

2 ¿Cómo energizar individualmente cada LED?

4.¿Que función genera la señal de entrada a shiftOut?

3 ¿Qué código se debe implementar en el IDE de Arduino?

1. INVESTIGACIÓN

Registro de desplazamiento 75HC595:

El chip shift registrer es un circuito integrado que hace parte de las librerías Arduino, el cual permite que a partir de 3 salidas digitales se consigue 8 salidas digitales y se pone en serie se puede conseguir 8\*n salidas digitales.

Su funcionamiento se realiza a través de algo llamado “Comunicación en seri síncrona” lo que significa que setea un pin en Alto o Bajo comunicando un BYTE (8 bits) al registro, donde luego se transmitirá a las 8 salidas de este. Para entender mejor su funcionamiento se muestra la siguiente figura 1 del datasheet, que indica su estructura esquemática.

Gráfico

Descripción generada automáticamente con confianza media

Fig. 1

Donde en la siguiente tabla se muestra la información de cada una de las salidas y entradas del circuito integrado.

|  |  |
| --- | --- |
| Q0 | Salida digital |
| Q1 | Salida digital |
| Q2 | Salida digital |
| Q3 | Salida digital |
| Q4 | Salida digital |
| Q5 | Salida digital |
| Q6 | Salida digital |
| Q7 | Salida digital |
| GND | Tierra |
| Vcc | Alimentación |
| DS | Entrada digital en serie de datos |
| OE | Habilitación de Salida |
| ST-CP | Activación de registro de almacenamiento |
| SH-CP | Pin de reloj de desplazamiento |
| MR | Reset de registro |
| Q’7’ | Salida para conexión en serie |

Tabla1

El funcionamiento del encapsulado es muy sencillo, se necesita una señal de 8 bits que será ingresada por la entrada digital DS, la lectura de cada bit se realiza con tiempo de reloj, es decir que es necesario crear un tren de pulsos y conectarlo a la entrada digital SH-CP, una vez todos los 8 bits han sido cargados es necesario habilitar las 8 salidas, esto se hace a través de la entrada digital ST-CP.

Como en este caso se usará la conexión en serie del dispositivo no solo se va a usar una entrada de 8 bits, será necesario crear una señal de 64 bits donde cada bloque de 8 será la señal que entrará a cada uno de los chips.

**Resistencia:**

Además la conexión al tratarse de LEDs es necesario determinar el valor de la resistencia a usar puesto que si supera la corriente nominal del Led o la del circuito integrado los podemos quemar, para esto se usara el valor de voltaje de trabajo del Arduino que son 5 Volts , la corriente máxima del LED  y la corriente máxima de  74HC9565  estos datos se pueden verificar en el datases como se muestra en la siguientes figuras.

Tabla

Descripción generada automáticamente

Fig 2

Tabla

Descripción generada automáticamente

Fig 3

Se observa que la corriente máxima que soporta el LED es de 155 mA y la del encapsulado es 35 mA  por tal motivo usaremos una resistencia que mantenga el nivel de corriente en 10mA.

Usando la ley de ohm se obtiene que

R=VI=50.010=500

1. PLANTEAMIENTO DE SOLUCIÓN

Para lograr resolver el reto propuesto y como antes se mencionó, se procedió por dividir el problema en general en problemas más específicos para lograr una generar una solución de una manera más sencilla.

1 ¿Como conectar los 64 LEDs para luego poder manipularlos?

Para lograr la conexión de los 64 LEDs se necesita hacer el uso como máximo de 7 pines digitales, ese decir se debe encontrar un método de conexión que con pocas salidas digitales se manipulen 64, basándonos en la investigación realizada anteriormente la respuesta a este problema es el uso de 74HC9565 nos permitirá acceder a 64 LEDs mediante 3 pines, la conexión deberá ser en serie, lo que nos lleva a usar los siguientes recursos.

* 64 LEDs color rojo
* 64 resistencias de 500 Ω
* 8 chips 74HC9565
* 1 placa Arduino UNO R3

Para ilustrar de una mejor manera como será la conexión de la matriz se ha optado por conectar en serie 2 chips, de la misma manera que se conectaran los 8 chips en el montaje final, esta implementación se puede observar en la siguiente figura.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

Fig 4

Los pines que se usaron fueron

pin 7= Entrada DS

pin 8= Entrada ST-CP

pin 9 = Entrada SH-CP

Como se puede ver en la anterior figura las salidas de cada uno de los chips están conectadas en serie con las resistencias y los LEDs, si se necesita observar de una mejor manera la simulación se puede ingresar al siguiente link.

<https://www.tinkercad.com/things/6WMndPlvGJw>

**2 ¿Cómo energizar individualmente cada LED?**

Para energizar cada LED se hace por medio de una señal de 64 bits donde la señal representa un número decimal, para su mayor comprensión se hace un modelo con 16 bits, es decir 2 chips 74HC595  para que en el momento de la implementación se extrapole a los 64 bits,

Imaginemos que tenemos 16 leds de la siguiente forma

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza baja

Fig 5

Donde cada LED tiene asociado un número decimal que corresponde a la base 2 elevado al exponente que corresponde a la posición del LED, sin embargo su energización se hace por medio de una salida digital donde cada chip es capaz de activar 8 salidas, por lo cual si se requiere encender el LED # 12, primero encontramos el número asociado su posición de la siguiente forma

Num= 212= 4096

binario = 0001000000000000

Que una vez sea ingresado a los registros del integrado se ubicaría de la siguiente manera.

Imagen que contiene reloj

Descripción generada automáticamente

Fig. 6

Como cada salida está conectada directamente a una resistencia y al LED se iluminará sólo aquel que su salida sea 1 (HIGH) en este caso el LED12.

**3. ¿Cómo detectar los LEDs prendidos en los patrones?**  
  
Con el motivo de facilitar la creación e implementación de los códigos necesarios para los patrones con las luces LED se decidió como paso inicial el hacer la lógica del programa en base al lenguaje estándar de c++ donde aquí se pensará en la posible solución y se comprobará que el método pensado funciona correctamente para así luego ser llevado a Arduino con sus respectivos cambios siguiendo la misma lógica

Patrón 1  
  
  
Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

Fig 7

 Lógica:

Creamos una función llamada patrón 1. Esta función toma un argumento filas, que representa el número de filas en el patrón. El patrón se divide en dos partes simétricas, la mitad superior y la mitad inferior. A continuación, creamos una variable llamada espacios Esta variable se utiliza para controlar la cantidad de espacios en blanco que se imprimen antes y después de los espacios rojos en cada fila. Comienza con el valor “filas – 1” y se reduce en cada iteración para crear el patrón en forma de pirámide.

Bucle para la mitad superior: En este bucle, se imprime la parte superior del patrón. Primero, se imprimen los espacios en blanco correspondientes a espacios, luego, se imprimen los espacios rojos, cuya cantidad es igual a filas \* 2 - 2 \* espacios. Esto crea la secuencia blanco y rojo en cada fila, finalmente, se imprimen más espacios en blanco, se repite este proceso hasta que los espacios llegan a 0.

Seguido a esto cuando se imprime la mitad superior se verifica si la fila actual es igual a “filas-i” ara así hacer una línea adicional de espacios rojos

Bucle para la mitad inferior: Se utiliza un bucle que es similar al bucle de la mitad superior, pero esta vez espacios se incrementa en lugar de disminuir. Esto crea la mitad inferior del patrón, que es simétrica a la mitad superior.

Patrón 2:

Un conjunto de letras negras en un fondo blanco

Descripción generada automáticamente con confianza media

Fig 8

Lógica:  
Para la solución de este problema se ideó usar dos bucles; uno tendrá que iterar las filas del patrón y el otro las columnas de cada fila. En cada celda, se verifica si la columna actual (j) es igual al índice de la fila actual (i) o si es igual al complemento del índice de la fila (filas - i - 1). Si alguna de estas condiciones es verdadera, será un espacio rojo, de lo contrario, será un espacio blanco. De esta manera, el código muestra espacios rojos en las posiciones diagonales desde la esquina superior izquierda hasta la esquina inferior derecha, y también desde la esquina superior derecha hasta la esquina inferior izquierda, creando el patrón en forma de cruz con espacios blancos en el resto de las posiciones.

Patrón 3:

Patrón de fondo

Descripción generada automáticamente

Fig 9

Lógica:

Utilizamos dos bucles anidados, uno para las filas (controlado por la variable i) y otro para las columnas (controlado por la variable j).

Utilizamos la condición ((i % 4 == 0 || i % 4 == 1) && (j % 4 == 0 || j % 4 == 1)) para verificar si estamos en las filas 0, 1, 4 o 5 y en las columnas 0, 1, 4 o 5. En estas posiciones, ira un espacio rojo luego tenemos la condición ((i % 4 == 2 || i % 4 == 3) && (j % 4 == 2 || j % 4 == 3)) verifica si estamos en las filas 2, 3, 6 u 7 y en las columnas 2 o 3. En estas posiciones, también irán espacios rojos.  
 por último usamos la condición (j == n - 1) para verificar si estamos en la última columna (columna 7) independientemente de la fila en la que estemos. En esta posición, ira espacio rojo para asegurarnos de que la última columna está compuesta por espacios rojos

 Patrón 4:

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

Fig 10

Lógica:

Nuevamente se ve que el patrón se refleja en la mitad así que haciendo la parte superior con un bucle y repetirlo con algún valor cambiado para que se impriman las filas en sentido opuesto vemos que es una solución bastante rápida.  
 Definimos dos variables, numFilas y longitudCadena, para controlar el número de filas y la longitud de cada cadena que queremos imprimir.  
 utilizamos dos bucles, el bucle exterior a través del número de filas especificado (numFilas), que en este caso es 4.

El segundo bucle está analizado (desde j = 0 hasta j < longitudCadena) se utiliza para recorrer cada posición en la fila. Dependiendo del valor de i (la fila actual) y j (la posición actual en la fila), el programa decide si será un espacio rojo o blanco. Con una condición if se verifica si j está dentro del rango de posiciones donde se debe mostrar un espacio rojo (en este caso, de i a i + 3). Si es así, se muestra un espacio rojo de lo contrario, será blanco

Luego se repite la lógica de estos dos bucles para replicar en sentido opuesto, el único cambio sería en el bucle exterior donde ahora la condición vendría siendo (desde i = numFilas - 1 hasta i >= 0)

**4 ¿Cómo generar la señal de salida que representan los patrones?**

Se tendrá una matriz de 8x8 con números enteros entre 0 y 63 donde cada posición de dicha matriz representa una posición de un led, cada fila tendrá un puntero para así poder diferenciar y poder actuar sobre ellas con facilidad.  
al momento de querer actuar dentro de cada fila se selecciona el puntero correspondiente a dicha fila y para luego saber sobre cual posición (columna) hay que habría que encontrar su valor en binario, lo cual podríamos hacer usando la siguiente operación

suma=suma+pow(2,fila{i})

Para hacerlo más entendible se hará un ejemplo de cómo sería el funcionamiento con el patrón 2

       Matriz       Patrón 2

Un conjunto de letras negras en un fondo blanco

Descripción generada automáticamente con confianza baja Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

          Fig 11         Fig 12

Primero identificamos el patrón, podemos ver que los espacios encendidos son (columna==fila) y (fila == columnas - 1 - columna).  
Como se mencionó anteriormente a cada fila para diferenciarlo le corresponde un puntero y dentro de este puntero están lo que vendrían siendo las columnas así que para encender los LEDs de la primera fila hay que usar su respectivo puntero para ingresar a su contenido, y ya que es la primer fila siguiendo el patrón que se mencionó antes el código debería de encender el primer y último led, acto similar a la segunda fila solo que ahora en lugar de ser el primer y último led los que se encienden estos vienen siendo el segundo y penúltimo, y así sucesivamente para cada fila de leds en la matriz

Por ejemplo, la operación para los leds encendidos en la primera fila sería

Led 1

suma=suma+pow(2,0)  
LEd 2

suma=suma+pow(2,7)

El 0 y 7 vienen de la posición a la que corresponde cada led y como se mencionó anteriormente los leds en esa posición son los que deberán de encenderse. En este caso suma valdría 129 (10000001) pues viene siendo la primera fila y da como resultado el valor binario usado para que el código sepa qué posición en la matriz queremos que encienda un led

**5 ¿Qué código se debe implementar en el IDE de Arduino?**

En el IDE de Arduino se hará uso de la función shitfOut la cual nos facilita el trabajo al ingresar la señal de entrada, puesto que es capaz de generar el ciclo de reloj y hacer la conversión a binario en la misma línea, esta función se declara de la siguiente manera.

shiftOut  ( Pin de Entrada, Pin Ciclo de reloj, (MSB o LSB), número en decimal)

Con esta función ya se tendría ingresada la señal de entrada a los chips, sin embargo al tratarse de una configuración en serie, es necesario ingresar los datos en bloques de 8 bits, lo cual conlleva a realizar una verificación del  tamaño de bits que tiene dicho número, para así generar un desplazamiento  para representar el número correctamente.

Basándonos en el caso de la figura 5 donde el LED a encender es el número 12 lo que significa que el número en la función shiftOut sería 4096 y este requiere más de 8 bits en su representación binaria, es necesario primero ejecutar un desplazamiento de la siguiente forma.

shiftOut(7,9,MSBFIRST,(LED12>>8));

shiftOut(7,9,LSBFIRST,LED12);

Análogamente se realizará este proceso para la conexión de los 64 LEDs, lo cual requerirá de 7 verificaciones.