Parcial 1

Juan David Martinez Bonilla
Emmanuel Garay Rivera
Sofia Marin Cacante

Departamento de Ingeniería Electrónica y
Telecomunicaciones
Universidad de Antioquia
Medellín
Abril de 2021

Índice

1. Problemas para resolver

A continuación se irán enumerando y documentando por fecha los problemas que van surgiendo al resolver el examen parcial con su respectivo análisis y solución/decisión tomada.

1.1. Problema 1.

Organización y disposición de los componentes electrónicos (19/04/21).

Análisis

Para tener un control organizado de la matriz 8x8 de leds se plantea la idea de usar dos circuitos integrados 74HC595 [1], uno para conectar los ánodos de las luces leds y otro para conectar los cátodos de dichas luces. La intención de esto es tener un sistema de referencia por "coordenadas" que permita ubicar cada led requerido de forma fácil y rapida.

Implementación

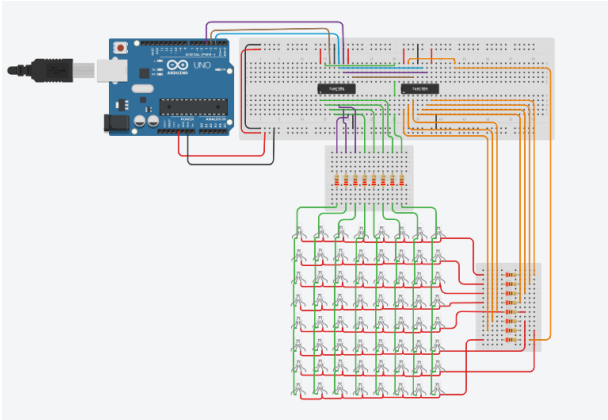

Figura 1: Primer circuito

1.2. Problema 2.

Conectando los componentes de la forma planteada en la solucion del problema 1 se detectó que, al prender todos los leds (de forma "manual") se quemaban los

circuitos integrados, no permitían un óptimo desarrollo del programa y se presentaban inconvenientes a la hora de realizar algunos patrones básicos (20/04/21).

Solución

Se rediseñó el circuito de tal manera que al seguir las compuertas lógicas de la siguiente tabla de verdad, los circuitos integrados no se quemaban, además de permitir (siguiendo la misma tabla de verdad) un diseño de algoritmo más eficiente y con posibilidad de ingresar cualquier patrón sin problema.

COLUMNA	FILA	SALIDA
0	0	0
0	1	1
1	0	0
1	1	0

Figura 2: Tabla de verdad

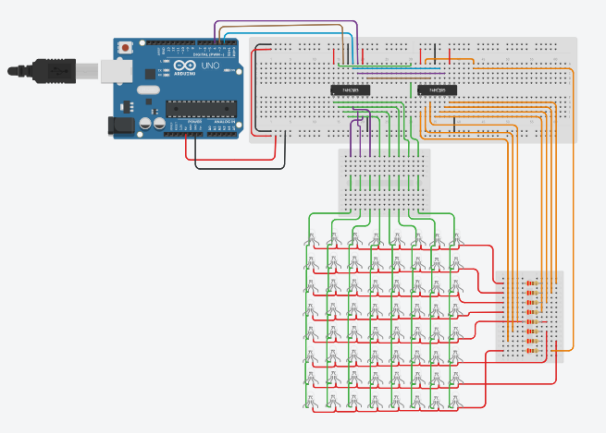

Figura 3: Segundo circuito

1.3. Problema 3.

¿Como recibir los patrones que el usuario quiera ingresar y traducir cada parte del patrón a la posición del led correspondiente? (19/04/21).

Análisis

Como idea principal se plantea nombrar e identificar cada posición de la matriz de leds, de forma que mediante una grafica mostrada en el manual de uso, el usuario pueda identificar que leds desea prender mediante el numero propio de cada led, además con el orden establecido de los números se puede dar la opción al usuario de prender varios leds al tiempo mediante un rango ingresado entre el 1 y el 64.

Ejemplo:

1 2 3 4 5 6 7 8
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
57 58 59 60 61 62 63 64

Implementación

```
36
37 void setup(){
38   // (Paso 1) Configuracion de puertos digitales como salida
39   pinMode(SER, OUTPUT);
40   pinMode(RCLK, OUTPUT);
41   pinMode(SRCLK, OUTPUT);
42
43   digitalWrite(SER, 0);
44   digitalWrite(RCLK, 0);
45   digitalWrite(SRCLK, 0);
46
47
48   digitalWrite(SER,1);  //Fila 1
49
50                         //Activar el reloj de la primera etapa
51   digitalWrite(SRCLK,0);
52   digitalWrite(SRCLK,1);
53   digitalWrite(SRCLK,0);
54
55   digitalWrite(SER,1);  //Columna 8
56   digitalWrite(SRCLK,0);
57   digitalWrite(SRCLK,1);
58   digitalWrite(SRCLK,0);
59
60                         //Activar el reloj de la segunda etapa
61   digitalWrite(RCLK,1);
62   digitalWrite(RCLK,1);
63   digitalWrite(RCLK,0);
64
```

Figura 4: Primer codigo

Después de inicializar las variables del pin de entrada (SER), reloj de desplaza-

miento (SRCLK) y reloj de registro de salida (RCLK), en modo OUTPUT, se realizó el desplazamiento de cada dato de forma manual, para analizar el funcionamiento e ir interiorizándolo, permitiendo crear patrones básicos.

1.4. Problema 4.

Al intentar realizar el desarrollo del problema 3 se concluyó que dicho método tenía poca optimización para implementar el programa al hacerlo de forma "manual" , además se detectaron problemas al querer desarrollar patrones complejos, dado que el sistema de filas y columnas anteriormente implementado, generaba un conflicto cuando se deseaba prender algunos patrones intermitentes (su forma no era continua/lineal) (21/04/21).

Solución

1. Se descartó la idea de enumerar cada posición de la matriz de muestra (del 1 al 64) junto con la idea de permitir que el usuario ingrese un rango de valores para encender ese número de leds siguiendo el orden numérico descartado anteriormente.

2. Se recurrió al uso de la función shiftOut [2] la cual recibe 4 parámetros:

-dataPin

-clockPin

-bitOrder

-value

Permitiéndonos así configurar la matriz de una forma más ordenada y sencilla, además de que con el método de entrada por bits (siguiendo la tabla de verdad de la figura 2) e implementando el método de desplazamiento por barrido [3] se puede lograr una ilusión de imagen solida logrando imprimir cualquier patrón sin ningún inconveniente.

Implementación

```
6
7  void ledWrite(int RLed, int GLed){
8
9     shiftOut(SER, SRCLK, LSBFIRST, GLed);
10    shiftOut(SER, SRCLK, LSBFIRST, RLed);
11
12    digitalWrite(RCLK, HIGH);
13    digitalWrite(RCLK, LOW);
14 }
15
16 void setup(){
17    pinMode(SER, OUTPUT);
18    pinMode(RCLK, OUTPUT);
19    pinMode(SRCLK, OUTPUT);
20 }
21
22 void loop(){
23    ledWrite(B00000000,B00000001);   // * leds fila 1
24    ledWrite(B00000000,B00000010);   // * leds fila 2
25    ledWrite(B11111000,B00000100);   // 3 primeros led fila 3
26    ledWrite(B10000000,B00001000);   // Todos menos el ultimo fila 4
27    ledWrite(B00000111,B00010000);   // Los tres ultimos fila 5
28    ledWrite(B00000000,B00100000);   // Fila 6
29    ledWrite(B00000000,B01000000);   // * leds Fila 7
30    ledWrite(B10000001,B10000000);   // *(1,8) Fila 8
31
32 }
33
```

Figura 5: Segundo codigo

```
ledWrite(0,1);         // * leds fila 1
ledWrite(0,2);     // * leds fila 2
ledWrite(31,4);    // 3 primeros led fila 3
ledWrite(1,8);     // Todos menos el ultimo fila 4
ledWrite(248,16);    // Los tres ultimos fila 5
ledWrite(56,32);       // Fila 6
ledWrite(0,64);        // * leds Fila 7
ledWrite(129,128);  // *(1,8) Fila 8
```

Figura 6: Segundo codigo (forma 2)

1.5. Problema 5.

Después de imprimir cualquier patrón se llegó a la conclusión de que los tiempos de ejecución eran infinitos, nunca terminaban, por esto surgió la necesidad de ejecutar ciclos en T tiempo (21/04/21).

Solución

después de analizar diferentes opciones como delay, o librerías como time.h se decidió implementar millis, ya que este sigue su conteo mientras van ejecutando los ciclos. Basándose del código extraído de **ArduWiki** [4].

Implementación

```
unsigned long t;            //Es muy importante respetar el tipo de dato
const int d = 10000;        //Duración en milisegundos del ciclo completo
void setup()
   pinMode(13, OUTPUT);
   Serial.begin(9600);
}
void loop() {

   while (1){
     t = millis();
     Serial.println(t);
     while (t%d < d/2){
        t = millis();
        Serial.println(t%d );
        Serial.println(d/2);
        digitalWrite(13,HIGH);       //Prende LED
        Serial.println(t);
     }
     digitalWrite(13, LOW);          //Apaga LED
     Serial.println(t);
     delay(1000);
     break;

   }
}
```

Figura 7: Tercer codigo

2. Diseño de funciones

2.1. Funcion verificación (full)

Esta función crea una matriz bidimensional cuyo contenido está en memoria dinámica, creando una columna de ceros y otra columna dependiente de la posición de la fila.
Ejemplo:

(0,1) == (B00000000,B00000001)
(0,2) == (B00000000,B00000010)
(0,4) == (B00000000,B00000100)
(0,8) == (B00000000,B00001000)
(0,16) == (B00000000,B00010000)
(0,32) == (B00000000,B00100000)
(0;64) == (B00000000,B01000000)
(0,128) == (B00000000,B10000000)

Codigo

```
45 void full(){
46
47    puntero_full = new int*[nFilas];
48
49    for(int i = 0;i<nFilas;i++){
50    puntero_full[i] = new int[nCol];
51    }
52
53    for(int i = 0;i<nFilas;i++,Cont = Cont*2){
54       *(*(puntero_full+i)+0) = 0;
55       *(*(puntero_full+i)+1) = Cont;
56    }
57 }
```

Figura 8: Cuarto codigo

2.2. Funcion clear

Se basa en la función verificación, con una modificación: en lugar de colocar un cero en la columna uno, coloca un 255, limpiando así toda la matriz.

(255,1) == (B11111111,B00000001)
(255,2) == (B11111111,B00000010)
(255,4) == (B11111111,B00000100)
(255,8) == (B11111111,B00001000)
(255,16) == (B11111111,B00010000)
(255,32) == (B11111111,B00100000)
(255;64) == (B11111111,B01000000)
(255,128) == (B11111111,B10000000)

Codigo

```
31 void clear(){
32
33    puntero_clear = new int*[nFilas];
34
35    for(int i = 0;i<nFilas;i++){
36    puntero_clear[i] = new int[nCol];
37    }
38
39    for(int i = 0;i<nFilas;i++,Cont = Cont*2){
40       *(*(puntero_clear+i)+0) = 255;
41       *(*(puntero_clear+i)+1) = Cont;
42    }
43 }
44
```

Figura 9: Quinto codigo

2.3. ledWrite

Esta función recibe como argumentos dos enteros (int) bien sea en notación binaria o decimal (0-255). Esta función se usa en conjunto con la función shiftOut, primero se envían los datos de la segunda posición de los argumentos ya que como se sabe los datos se van desplazando al enviarlos de forma serial. para finalizar se realiza un flanco de subida en el reloj de registro de salida.

Codigo

```
16 void ledWrite(int GLed, int RLed){  //Gled = Columnas; Rled = Filas
17
18    /*Los datos se envian de tal manera que los primeros que ingresan
19    son los ultimos en la posicion del integrado, si estan en */
20
21    shiftOut(SER, SRCLK, LSBFIRST, RLed);
22    shiftOut(SER, SRCLK, MSBFIRST, GLed);
23
24    /*Activar el flanco de subido del regristo de salida. */
25    digitalWrite(RCLK, LOW);
26    digitalWrite(RCLK, HIGH);
27    digitalWrite(RCLK, LOW);
28 }
```

Figura 10: Sexto codigo

2.4. print-Matriz

Imprime una matriz 2x8

Codigo

```
83 void print_Matriz(int **puntero, int nFilas, int nCol){
84  for(int i=0;i<nFilas;i++){
85       Serial.print(*(*(puntero+i)+0));
86       Serial.println(" ");
87       Serial.print( *(*(puntero+i)+1));
88       Serial.println(" ");
89  }
90 }
```

Figura 11: Septimo codigo

2.5. image

Le permite al usuario mostrar un patrón en la matriz de leds.

Codigo

```
59  void image(int matriz[]){
60    int *ptr_matriz;
61    ptr_matriz=matriz;
62
63    puntero_image = new int*[nFilas];
64    for(int i = 0;i<nFilas;i++){
65      puntero_image[i] = new int[nCol];
66    }
67
68
69    for(int i = 0;i<nFilas;i++,Cont = Cont*2){
70
71        *(*(puntero_image+i)+0) = *ptr_matriz++;
72        *(*(puntero_image+i)+1) = Cont;
73    }
74  }
75
```

Figura 12: Octavo codigo

```
159     case 3:
160       int imagen[8]={0,126,126,126,126,126,126,0};
161       int temp;
162       /*
163       for (int i=0;i<8;i++){
164         while (Serial.available()==0){}
165         temp = Serial.parseInt();
166         imagen[i]=temp;
167         Serial.print(temp);
168         Serial.print(",");
169       }
170       */
171       image(imagen);
172       ptrTime=&Time;
173       *ptrTime = millis();
174
175       while (Time%Duracion < Duracion/2){
176         *ptrTime = millis();
177         viewMatriz(puntero_image,nFilas,nCol);
178       }
179       clear();
180       viewMatriz(puntero_clear,nFilas,nCol);
181       Serial.println(" ");
182       Serial.println("       One moment, plis");
183       delay(3000);  //Esto es para que el programa no se muera
184       break;
185
186   }
```

Figura 13: Octavo codigo (forma 2)

La función image no se logró desarrollar por completo

3. Algoritmo implementado

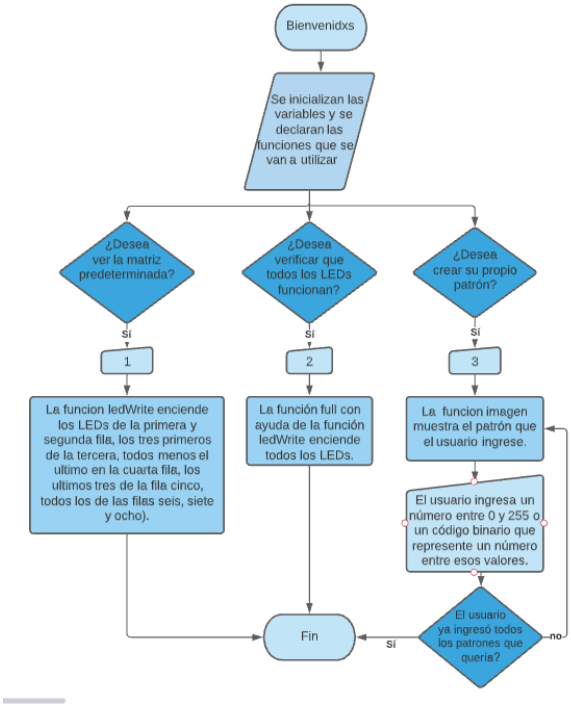

Figura 14: Algoritmo implementado

Referencias

[1] J. P. E. Aumentar pines arduino con shift register 74hc595. [Online]. Available: https://www.youtube.com/watch?v=-pvTlcFQ2Wgab$_channel = JohannPerezE$

[2] A. en español. shiftout(). [Online]. Available: http://manueldelgadocrespo.blogspot.com/p/shiftout$_1.html$

[3] Veritasium. How was video invented? [Online]. Available: https://www.youtube.com/watch?v=rjDX5ItsOnQab$_channel = Veritasium$

[4] ArduWiki. millis(). [Online]. Available: https://arduwiki.perut.org/wiki/millis()