Parcial 1

Informa2 S.A.S.

Holman Londoño Restrepo

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

Índice

1. Análisis y consideraciones.

Inicialmente veía la implementación de este parcial como algo imposible de hacer, pero luego de leer e informarme más a fondo comencé a entender el funcionamiento de este circuito integrado (74HC595), desde un principio lo que más complicado me pareció fue encontrar la manera de conectar varios integrados paralelamente sin usar más de 7 pines digitales del Arduino y no fue hasta luego de buscar en internet información sobre este integrado que me enteré de la función del pin 9 (pin de salida invertida) y la manera en cómo podía enviar diferentes estados entre un integrado y otro, sin tener que usar más salidas digitales para el SER, posteriormente pensé en que la manera más optima de hacerlo era con dos integrados únicamente, usando el sistema de multiplexación, método en el cual está basado el funcionamiento del "7 segmentos", que consiste en ir iterando entre diferentes estados para cada fila y así dar la ilusión de que toda la matriz se encuentra encendida, en palabras menos formales, prende 'x' cantidad de leds en cada una de las filas en diferente tiempo, pero se hace de una manera tan rápida que para el ojo humano es imperceptible y da la ilusión de que todas la filas se encuentras prendidas al tiempo, la ventaja de este método es que era mucho más económico ya que solo se necesitarían 2 integrados a diferencia de los 8 que a primera vista parecía ser lo más viable, además de la bonificación que implicaba ahorrar el mayor número de pines en el Arduino, pero la desventaja es que necesitaba una codificación más compleja por el hecho de estar actualizando el estado de cada fila constantemente.

2. Desarrollo del algoritmo.

- Desarrollo de algoritmo.
  - Tareas planteadas.
    - Tarea 1. Enviar datos correctamente entre integrados.
    - Tarea 2. Hacer funciones para hacer ciclos de corrimiento correctamente.
    - Tarea 3. Definir forma en la que los usuarios ingresen patrones.
    - Tarea 4. Almacenar correctamente los patrones ingresados por el usuario.
    - Tarea 5. Mostrar correctamente los patrones usando multiplexación.

3. Algoritmo implementado.

```
int SER = 2;
int SRCLK = 3;
int RCLK = 4;

long long int FilaPrueba;

int FilasOn[8][8] = {}; //Arreglos.
int NumFila = 0;
int Contador =0;
int NEjec = 0;
int Opcion, Tam = 10;
char *S = new char[Tam]; //Memoria dinamica.
char W[10] = {'B','I','E','N','V','E','N','I','D','O'};
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(SER, OUTPUT);
  pinMode(SRCLK, OUTPUT);
  pinMode(RCLK, OUTPUT);

  Menu();
}

void loop()
{
  while(NEjec == 0){
    if(Serial.available() > 0){
      Opcion = Serial.parseInt();
      NEjec++;
    }
  }
  switch(Opcion){
    case 1:
    Verificacion();
    if(NEjec == 1){
      Serial.println("Verificando condicion de los leds.");
      NEjec++;
    }
    break;
    case 2:
    Imagen();
    case 3:
    Publik();
    break;
  }

}

void DisplaceAndShow()
{
  Corrido();
  Show();
}

void Corrido()
{
  digitalWrite(SRCLK, 0);
  digitalWrite(SRCLK, 1);
  digitalWrite(SRCLK, 0);
}

void Show()
{
  digitalWrite(RCLK, 0);
  digitalWrite(RCLK, 1);
```

```
  digitalWrite(RCLK, 0);
}

void Verificacion(){
  for(int i=0; i<16; i++){
    if(i<8){
      digitalWrite(SER, 1);
      Corrido();
    }
    else{
      digitalWrite(SER, 0);
      Corrido();
    }
  }
  Show();
}

void Imagen(){
  if(Contador == 0){
    Serial.println("Ingrese la configuracion de la primer linea de la matriz.");
    Serial.println("Recuerde que estan numeradas de arriba hacia abajo: ");
    while(!Serial.available() > 0){};
  }
  if(Serial.available() > 0){
    FilaPrueba = Serial.parseInt();
    for(int i=7; i>=0; i--){
      FilasOn[Contador][i] = FilaPrueba%10;
      FilaPrueba /= 10;
    }
    Contador++;
    Serial.println("\nSerial ingresado con exito.");
    if(Contador < 8){
      Serial.println("Ingrese la configuracion de la fila ");
      Serial.print(Contador+1);
    }
    if(Contador == 8){
      Serial.println("Mostrando patron.");
    }
  }

  if(Contador == 8){
    AlgoritmoDeOrdenamiento(NumFila);
        NumFila++;
        if(NumFila == 8){
      NumFila = 0;
        }
  }
}

void Publik(){

}
```

```
void Menu(){
  for(int i=0; i<Tam; i++){
    *(S+i) = *(W+i);
  }
  for(int i=0; i<Tam; i++){
    Serial.print(*(S+i));
  }
  Serial.print(".␣\n");
  Serial.println("\n1.␣Verificacion.");
  Serial.println("2.␣Mostrar␣patron␣ingresado␣por␣el␣usuario.(Imagen.)");
  Serial.println("3.␣Mostrar␣secuencia␣de␣patrones.(Publik.)");
  Serial.println("\nIngrese␣la␣opcion␣que␣desea␣ejecutar:");
}

void AlgoritmoDeOrdenamiento(int Fila)
{
  for(int i=7; i>=0; i--){
    digitalWrite(SER, *(*(FilasOn+((Fila-7)*-1))+i)); //Uso de punteros.
    Corrido();
  }
  for(int i=0; i<8; i++){
    if(i == Fila){
      digitalWrite(SER, 0);
    }
    else{
      digitalWrite(SER, 1);
    }
    Corrido();
  }
  Show();
}

void Imagen(){
  if(Contador == 0){
    Serial.println("Ingrese␣la␣configuracion␣de␣la␣primer␣linea␣de␣la␣matriz.");
    Serial.println("Recuerde␣que␣estan␣numeradas␣de␣arriba␣hacia␣abajo:␣");
    while(!Serial.available() > 0){};
  }
  if(Serial.available() > 0){
    FilaPrueba = Serial.parseInt();
    for(int i=7; i>=0; i--){
      FilasOn[Contador][i] = FilaPrueba%10;
      FilaPrueba /= 10;
    }
    Contador++;
    Serial.println("\nSerial␣ingresado␣con␣exito.");
    if(Contador < 8){
      Serial.println("Ingrese␣la␣configuracion␣de␣la␣fila␣");
      Serial.print(Contador+1);
    }
    if(Contador == 8){
      Serial.println("Mostrando␣patron.");
    }
```

```
  }

  if(Contador == 8){
    AlgoritmoDeOrdenamiento(NumFila);
        NumFila++;
        if(NumFila == 8){
      NumFila = 0;
        }
  }
}

void Publik(){

}

void Menu(){
  Serial.println("Bienvenido.");
  Serial.println("\n1. Verificacion.");
  Serial.println("2. Mostrar patron ingresado por el usuario.(Imagen.)");
  Serial.println("3. Mostrar secuencia de patrones.(Publik.)");
  Serial.println("\nIngrese la opcion que desea ejecutar:");
}

void AlgoritmoDeOrdenamiento(int Fila)
{
  for(int i=7; i>=0; i--){
    digitalWrite(SER, *(*(FilasOn+((Fila-7)*-1))+i)); //Uso de punteros.
    Corrido();
  }
  for(int i=0; i<8; i++){
    if(i == Fila){
      digitalWrite(SER, 0);
    }
    else{
      digitalWrite(SER, 1);
    }
    Corrido();
  }
  Show();
}
```

4. Problemas de desarrollo.

A lo hora de desarrollar este parcial me presente con varios problemas y a medida que iba avanzado aparecían más, como por ejemplo la poca estabilidad que proporciona la plataforma Tinkercad, ya que habían ocasiones en las que no se guardaban los cambios hechos en el código o el circuito y al momento de refrescar la página tenía que volver a realizar estos cambios efectuados anteriormente, para lidiar con este problema me fue de gran ayuda el repositorio en GitHub ya que cuando esto sucedía podía ir a ver los commits hechos y así recuperar los cambios que no se guardaban en Tinkercad; otro problema que presente fue aprender a controlar el tiempo de ejecución ya que esta vez estábamos programando hardware, el cual es más complicado debido a que se debe tener en cuenta que los códigos implementados en este se van a ejecutar indefinas veces hasta que el usuario detengo su ejecución, pero creo que el mayor

problema que presente a la hora de realizar el parcial fue en la codificación, ya que la multiplexación depende totalmente de una correcta programación, y de saber en qué estado poner la filas en un preciso instante para lograr mostrar el patrón ingresado por el usuario en la matriz de leds, a todo esto se le suma el tiempo para realizar este, ya que decidí hacerlo solo, entonces era mayor la carga pero considero que es lo mejor para mi aprendizaje.

5. Evolución del algoritmo.

En un principio el algoritmo era bastante engorroso ya que hacia los corrimientos con 3 líneas al igual que el pulso para mostrar los datos era con 3 líneas, así que solamente para hacer el corrimiento y darle salida a los datos usaba 6 líneas de código, pero luego implemente una función para esto y logre hacerlo con una sola línea, luego pensé en definir 26 patrones, uno por letra del alfabeto, para que el usuario ingresara la letra que quería mostrar en la matriz y el sistema la reconociera y dependiendo de esto mostrara la letra ingresada, pero el problema de esta idea es que el usuario estaba limitado a solo 26 diferentes patrones, además el profesor Augusto durante una clase dijo que el programa debería dejar ingresar al usuario el patrón que deseara, por lo tanto decidí guardar los estados de los leds en una matriz de 8x8 donde los leds prendidos se representan por '1' y los leds apagados por '0', de esta manera se podía controlar el estado de cada led de manera independiente, para la función Publik tenia pensado utilizar una matriz con memoria dinámica donde el numero de filas fuera la cantidad de patrones que quería mostrar el usuario y las columnas fueran por defecto 8 para que en cada fila se guardara un patrón diferente y las columnas representarían el estado de cada fila de la matriz led, para esto cada posición del arreglo de dos dimensiones tenía que almacenar un numero binario de 8 bits ya que cada fila de la matriz led tiene 8 leds, de esta manera en cada fila del arreglo habría 8 números binarios de 8 bits para un total de 64 números por fila del arreglo, cada uno haciendo referencia a un led en especifico de la matriz led, pero lastimosamente por falta de tiempo no pude implementar la función por completo.