Parcial I

Informática II

Juan Pablo Areiza Jiménez
Santiago Montoya Leal
Juan Diego Sánchez Ramos

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

Índice

1. Introducción

En el siguiente informe, se presenta en detalle el desarrollo y la solución propuesta del proyecto. El objetivo principal de este proyecto es demostrar las habilidades adquiridas en el análisis de problemas y el manejo del lenguaje C++ a lo largo del curso, realizando consideraciones para la solución del problema, así como la implementación de esquemas y algoritmos para conseguir otorgar una solución que cumpla con todos los requisitos del proyecto.

2. Materiales

- Cables jumpers
- Protoboard
- Arduino UNO
- Circuito integrado 74HC595 x8
- LED azul x 64
- Resistencia 560 ohm x 64

3. Contenido

En la sección de contenido, se detalla el análisis que se dio al problema, el montaje realizado para el cumplimiento de este, así como el código y su respectiva descripción.

3.1. Análisis y Desarrollo

El principal problema del proyecto era el uso de solo 7 pines digitales, por lo tanto, para conseguir controlar los 64 LED, se requería el uso del circuito integrado 74HC595, el cual es un registro de desplazamiento. Con este circuito integrado, se logra obtener 8 salidas digitales a partir de 3 pines digitales del Arduino.

Inicialmente, se planteó dar solución al proyecto a partir de la implementación de dos circuitos integrados en paralelo, sin embargo, este planteamiento fue descartado, pues con este, se notó una mayor dificultad para controlar cada LED de forma individual, por lo tanto, como grupo se pensó en una solución que permitiera controlar cada led de forma individual, así se llegó a la idea de implementar 8 circuitos integrados, de esta forma, se tendría un control de cada LED de forma individual, facilitando el planteamiento del problema.

Dicho análisis llevaba al montaje de 8 circuitos integrados en serie, empleando solo 3 pines digitales del Arduino. Una vez teniendo claro cómo estaría

dispuesto el montaje y teniendo claro el funcionamiento del circuito integrado, así como conociendo los 256 estados posibles (0 a 255) que proporciona los bits de salida.

El siguiente desafío surgió cuando se necesitaba realizar el desplazamiento de un byte (8 bits) desplazando un bit a la vez , para esto se implementó la función shifOut, la cual tiene como parámetros de entrada los datos del pin SER (entrada serial), el ciclo del reloj del registro de desplazamiento, una palabra reservada para determinar el orden del registro de desplazamiento, en este caso se utilizó LSBFIRST que hace un desplazamiento comenzando por el bit menos significativo y finalmente los datos que se desplazan a la salida, los cuales están en el rango de 0 a 255, según los patrones de los posibles bits de salida para cada circuito integrado.

El último parámetro de la función shifOut corresponde a la secuencia en binario que se desea representar, siendo 1 el estado alto (LED encendido) y 0 el estado bajo (LED apagado), por lo que si se deseaba representar todas las luces LED encendidas, se ingresa 11111111 y su correspondiente valor decimal es 255, de esta forma, se trabajó en la idea de permitir que el usuario ingrese una secuencia de unos y ceros, que representara las luces LED encendidas y apagadas para cada fila, es decir para cada registro de desplazamiento.

Además, a partir de este análisis de la secuencia binaria, se logró desarrollar la función verificación, asignando el último parámetro en 255, empleando la función shifOut 8 veces, es decir, para los 8 registros de desplazamiento.

Teniendo clara la idea de cómo serían ingresados los patrones que desea el usuario, se definió una función que captura los datos del puerto serial y los asigna a un arreglo, esta función se empleó tanto en la captura de las secuencias binarias como en la captura de datos como lo son la cantidad de patrones y el intervalo de tiempo entre estos. A su vez, esta función se empleó para ejecutar la verificación del funcionamiento de cada LED a partir de un comando ingresado por el usuario.

Para los datos correspondientes al número de patrones y al intervalo de tiempo entre cada patrón, se empleó la función atoi y atof, para convertir el arreglo donde se almacenaron estos datos a variables tipo entero y flotante, respectivamente.

En este punto, se lograba recibir los datos, sin embargo, se requirió implementar una función en la cual se lograra convertir el arreglo con la secuencia binaria a una variable tipo entero. Para esto, se implementó una función que tuviese como parámetro de entrada un arreglo y retornara una variable de tipo entero, el funcionamiento es muy simple, consiste en recorrer el arreglo por cada posición verificando que se encuentren unos y ceros, cuando se encuentra un uno este valor aumenta en una unidad y asigna el doble de este valor a una varia-

ble, si encuentra un 0, simplemente aumenta el doble a la variable mencionada anteriormente, como se está aumentando el doble antes de comprobar que se encontró un uno o un cero, al final, es necesario entregar la mitad de la variable, ese dato corresponde al valor decimal de la secuencia binaria.

Para la implementación de la función imagen, era necesario tener una secuencia binaria para cada fila de la matriz, esta es ingresada por el usuario, para cada uno de estos valores, cada secuencia es convertida en un decimal y este es el dato que se le asigna a la función imagen, como son 8 diferentes secuencias, la función imagen recibe 8 enteros diferentes y con ayuda de la función shiftOut se logra imprimir en los LED cada secuencia presentada para cada fila, obteniendo así un patrón.

Finalmente se necesitaba desarrollar la función publik, la cual tiene como parámetros de entrada la cantidad de patrones que se desean presentar y el tiempo entre estos, de esta forma, a partir del número de patrones que el usuario desee, para n patrones, el usuario deberá ingresar n*8 secuencias, una para cada fila y se le indica cuándo ha acabado de ingresar un patrón, a partir de estas secuencias y su respectiva representación decimal, se llena un arreglo y a partir de este y la función imagen, se imprimirán los patrones, el tiempo entre cada patrón ingresado por el usuario está dado en segundos, por lo tanto, es necesario multiplicar a este por 1000 y llevarlo a la función delay.

3.2. Algoritmo

A continuación, se presenta el código empleado en el proyecto:

```
/*
pinData ---> Entrada serial
pinLatch ---> Reloj de registro de salida
pinClock ---> Reloj de registro de desplazamiento
*/

int pinData  = 2;
int pinLatch = 3;
int pinClock = 4;

int patrones;
float tiempo;
int *ptrp=&patrones;
float *ptrt=&tiempo;

/*Funcion verificacion, empleada para iluminar todos
los LED.*/
void verificacion(){
```

```
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    shiftOut(pinData, pinClock, LSBFIRST, 255);
    digitalWrite(pinLatch, HIGH);
    digitalWrite(pinLatch, LOW);
}

/*Funcinn imagen, tiene como par metro de entrada un
valor entero de cada fila de LED, ese valor entero
corresponde al valor decimal de un patron binario.*/

void imagen(int aled,int bled,int cled,int dled,
        int eled,int fled,int gled,int hled){

    shiftOut(pinData, pinClock, LSBFIRST, hled);
    shiftOut(pinData, pinClock, LSBFIRST, gled);
    shiftOut(pinData, pinClock, LSBFIRST, fled);
    shiftOut(pinData, pinClock, LSBFIRST, eled);
    shiftOut(pinData, pinClock, LSBFIRST, dled);
    shiftOut(pinData, pinClock, LSBFIRST, cled);
    shiftOut(pinData, pinClock, LSBFIRST, bled);
    shiftOut(pinData, pinClock, LSBFIRST, aled);
    digitalWrite(pinLatch, HIGH);
    digitalWrite(pinLatch, LOW);
}

/*Funcion ingreso, es la empleada para recibir datos
por el puerto serial.*/
void ingreso(char ing[8]){
  int verif=0,posicion=0;
  Serial.println("Ingrese el dato: ");
  while (true){
    if (Serial.available()){
        posicion=0;
        while(Serial.available()>0)
        {
         delay(50);
             ing[posicion]=Serial.read();
         posicion++;
        }
        Serial.print("Ingresado: ");
```

```
                Serial.println(ing);
        /*Condicion que comprueba si un dato ingresado
        es la palabra 'test', y as  hacer la verificaci n*/
        if(ing[0]==116 && ing[1]==101 && ing[2]==115 &&
        ing[3]==116){
                verificacion();
                /*Se limpia el contenido del arreglo para
                seguir recibiendo el dato esperado.*/
                for(int i=0;i<=posicion;i++){
                  ing[i]=NULL;
                }
        }
        else{
             break;
             }
    }
  }
}

/*Funcion que permite convertir un arreglo que contiene
un numero binario a un entero decimal.*/
int BinToInt (char ing[8]){

  int i = 0;  //Variable para el contador
  int n = 0;  //Variable para calcular el resultado

  while ( ing[i] == '0' || ing[i] == '1' ) {
    if ( ing[i] == '0' )
      n <<= 1;
    else {
      n ^= 1;
      n <<= 1;
    }
  ++i;
  }
  n >>= 1;

  //Serial.print("Prueba: ");
  //Serial.println(n);

  return(n);

}

//Funcion empleada para pedir e imprimir patrones.
void publik(int patrones, float tiempo){
```

```
  int cantidad=0;
  char patron[9]={};
  int patronA;
  cantidad=(patrones*8)-1;
  int array[cantidad+1];

  int aux=0;
  int tiempoaux=int (tiempo*1000.0);

  for(int i=0;i<patrones*8;i++){
    ingreso(patron);
        //Serial.print("PATRON: ");
        //Serial.println(patron);
    patronA=BinToInt(patron);
        //Serial.println(patronA);
    array[aux]=patronA;
    aux=aux+1;
    if((i+1)%8==0){Serial.println("Ya ingresaste un patron.");}
  }
  /*Ciclo infinito que indica que los patrones
  ingresados por el usuario se presentar n hasta
  detener la simulaci n.*/
  while(true){
  for(int k=0;k<(patrones*8)-1;k=k+8){
    imagen(array[k],array[k+1],array[k+2],array[k+3],array[k+4],
    array[k+5],array[k+6],array[k+7]);
    delay(tiempoaux);
  }
  if(patrones==1){break;} //Si el usuario solo ingres  un patr n,
  el programa permitir  que siga ingresando patrones.
  }
}

void setup(){
   pinMode(pinData, OUTPUT);
   pinMode(pinLatch, OUTPUT);
   pinMode(pinClock, OUTPUT);

   digitalWrite(pinData, 0);
   digitalWrite(pinLatch, 0);
   digitalWrite(pinClock, 0);

   Serial.begin(9600);
   Serial.println("-El primer dato solicitado
    es el numero de patrones.");
   Serial.println("-Si la cantidad de patrones que ingreso es 1,
```

```
␣␣␣puede␣seguir␣ingresando␣patrones␣una␣vez␣que␣se␣presente␣este,
␣␣␣sin␣embargo,␣si␣presenta␣una␣");
    Serial.println("secuencia,␣el␣codigo␣se␣seguira
␣␣␣ejecutando␣con␣esta.␣");
    Serial.println("-Despues␣de␣brindar␣el␣numero␣de␣patrones,
␣␣␣ingrese␣el␣tiempo␣entre␣patrones␣en␣segundos.␣");
    Serial.println("-Acto␣seguido,␣se␣pide␣que␣ingrese␣8
␣␣␣patrones␣diferentes,␣1␣por␣fila.␣los␣unos␣representan␣un␣LED
␣␣␣encendido␣y␣los␣ceros␣un␣LED␣apagado.␣");
    Serial.println("-Por␣ejemplo;␣si␣desea␣que␣el␣led␣de␣la␣columna
␣␣␣2␣se␣ilumine␣y␣los␣demas␣esten␣apagados,␣ingrese:␣01000000␣");
    Serial.println("-Si␣desea␣hacer␣la␣verificacion␣en␣cualquier
␣␣␣momento,␣ingrese␣la␣palabra␣'test'␣y␣luego␣ingrese␣el␣dato
␣␣␣solicitado.");
}

void loop(){
  char *aux=new char[5];

  Serial.println("Ingrese␣cantidad␣de␣patrones:␣");
  ingreso(aux);
  *ptrp=atoi(aux);
  Serial.print("Cantidad␣de␣patrones:␣");
  Serial.println(patrones);

  delete [] aux;

  if(patrones!=1){
    Serial.println("Ingrese␣tiempo␣entre␣patrones:␣");
    ingreso(aux);
    *ptrt=atof(aux);
    Serial.print("Tiempo␣entre␣patrones␣:");
    Serial.println(tiempo);
    delete [] aux;
  }

  Serial.println("A␣continuacion,␣ingrese␣sus␣patrones." );
  publik(patrones,tiempo);

  Serial.println("El␣patron␣ya␣ha␣sido␣mostrado,␣si␣desea
␣␣volver␣a␣ingresar␣datos,␣empiece␣nuevamente␣por␣el␣#
␣␣de␣patrones.");
}
```

3.3. Problemas de desarrollo

El principal problema en la implementación del proyecto, fue la etapa del montaje, pues inicialmente se planteó una solución implementando dos circuitos integrados en paralelo, esto presentó una gran dificultad para el grupo, pues no se lograba tener un completo control de cada LED, sin embargo, esto ayudó a buscar otra solución que al final fue la que permitió desarrollar el proyecto cumpliendo todas las especificaciones.

Los demás problemas se debieron al programa tinkercad, pues para los arreglos, este alteraba el valor de la última posición de estos, esto se logró resolver agregando una posición extra a los arreglos.

3.4. Esquema de tareas

En la Figura (1), se presentan las tareas definidas durante la realización del proyecto, indicando los días en los que se trabajó en una tarea específica.

Figura 1: Esquema de tareas

3.5. Montaje

En la Figura (2), se presentan el montaje realizado a partir del uso de circuitos integrados 74HC595.

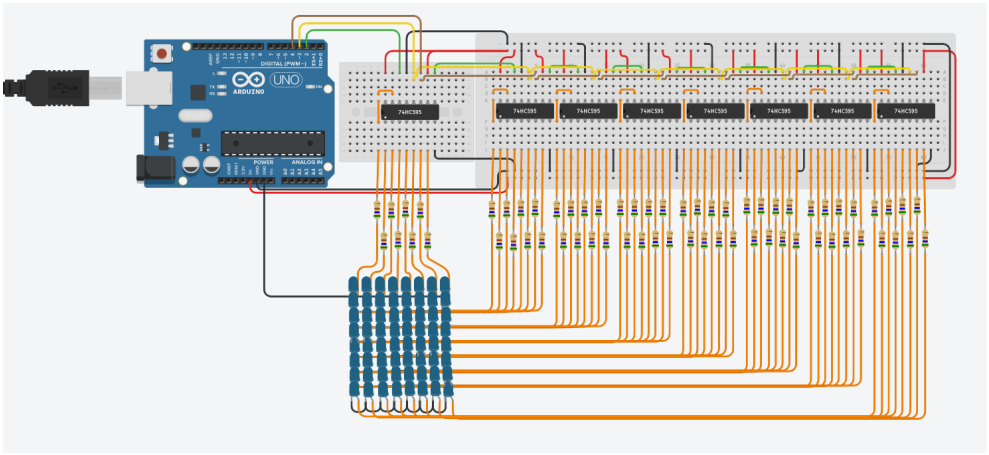

Figura 2: Montaje