Parcial 1 Informática II:
Manejo matriz LED con Arduino

Yuliana Betancur Saldarriaga

Universidad de Antioquia

Facultad de ingeniería

Informática II

Medellín-Antioquia

2023

Propósito del documento

En el presente informe se detalla una solución para la gestión y manejo de matrices led construidas en un entorno de Tinkercad, con Arduino y C++. El planteamiento general consiste en dar manejo a una matriz 8x8 compuesta de LEDs interconectados que contaran con la capacidad de reproducir diferentes patrones operativos.

Descripción de la implementación

Para llevar a cabo los procedimientos se requiere hacer uso del software Tinkercad para la simulación del montaje de un circuito con Arduino uno, dicho circuito cuenta con 2 integrados 74HC595 usados para la gestión del procedimiento y la conexión de los LEDs.

Componentes del circuito

- Arduino uno.
- Protoboard o placa de pruebas.
- 2 integrados 74HC595.
- 64 LEDs.
- 8 resistencias.
- Cables conectores.

Conexiones

El circuito se encuentra conectado por medio de la protoboard, elemento que sirve para facilitar las diferentes conexiones, estas se encuentran definidas de la siguiente manera:

Arduino:

Conectividad con integrado #1 (conectado a columnas de la matriz):

- Pin digital #8: Pin ST_CP (Storage Register Clock Input) – Latch.
- Pin digital #11: Pin DS (Serial Data Input) – Data.
- Pin digital #12: Pin SH_CP (Shift Register Clock Input) – Clock.
- 5V: Pines VCC (Positive Supply Voltage) y MR (Master Reset Active Low).
- GND (Ground): Pines GND (Ground 0 V) y OE (Output Enable Active Low).

Conectividad con integrado #2 (conectado a filas de la matriz):

- 5V: Pines VCC (Positive Supply Voltage) y MR (Master Reset Active Low).
- GND (Ground): Pines GND (Ground 0 V) y OE (Output Enable Active Low).

Integrado #1

Conectividad a matriz LED

- Q0 – Q7: Conexión columnas matriz LED (ánodo).

Conectividad a integrado #2

- Q7': Pin DS (Serial Data Input).
- SH_CP: Pin SH_CP (Shift Register Clock Input) – Clock.
- ST_CP: Pin ST_CP (Storage Register Clock Input) – Latch

Conectividad a Arduino

- VCC – MR: Pin 5V
- GND – OE: Pin GND
- SH_CP: Pin digital #12
- DS: Pin digital #11
- ST_CP: Pin digital #8

Integrado #2

Conectividad a matriz LED

- Q0 – Q7: Conexión filas matriz LED (ánodo).
    - Resistencia de 500 Ω en cada fila.

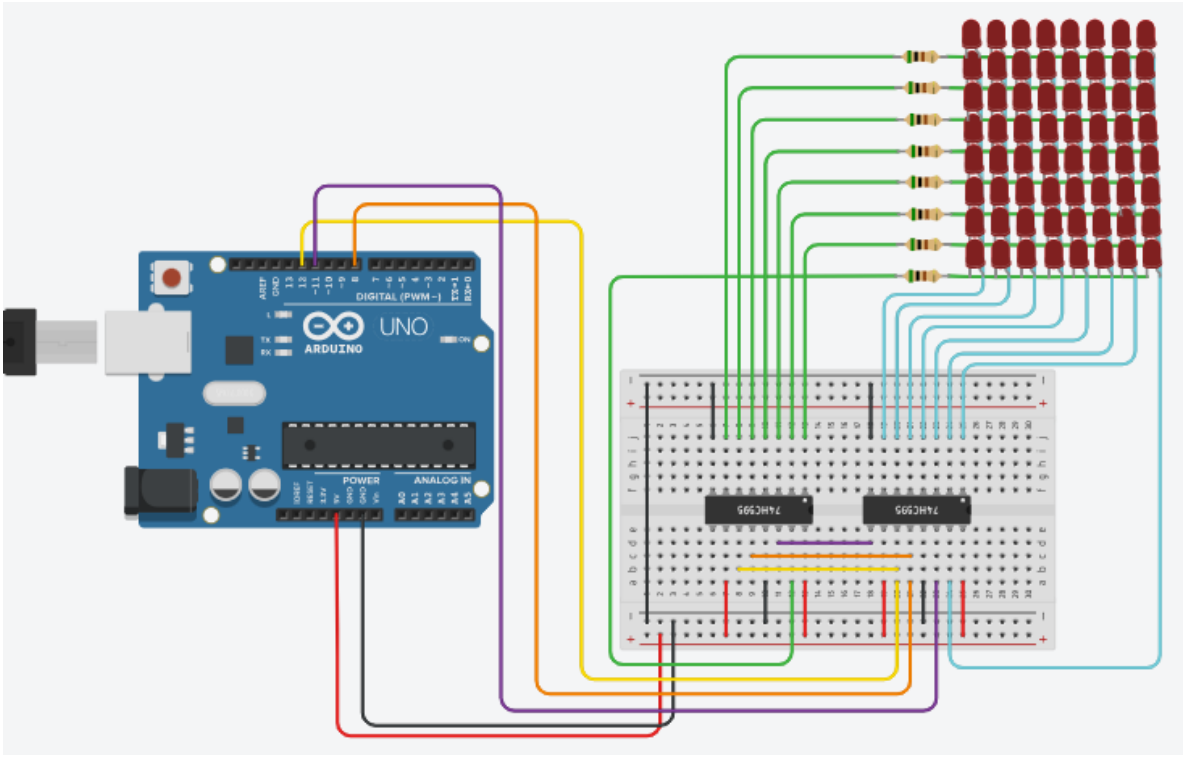
Conectividad a integrado #1

- DS: Pin Q7' (Serial Data Output).
- SH_CP: Pin SH_CP (Shift Register Clock Input) – Clock.
- ST_CP: Pin ST_CP (Storage Register Clock Input) – Latch

Conectividad a Arduino

- VCC – MR: Pin 5V
- GND – OE: Pin GND

Luego de realizar estas conexiones, se puede visualizar el circuito implementado de la siguiente manera:



Enlace circuito:
https://www.tinkercad.com/things/2q1gfj2subN?sharecode=GFboEZMuPIAaR3F56ewKGFEcRtGX9CpGMpFORL-Wd4E

Esta implementación brindaba funcionalidad para la necesidad planteada, ya que cada integrado 74HC595 cuenta con 8 pines de salida lo cual era de utilidad para poder contar con el manejo de la matriz ya que esta presentaba 8 filas y 8 columnas y este tipo de elementos podía ser controlado con cada integrado.

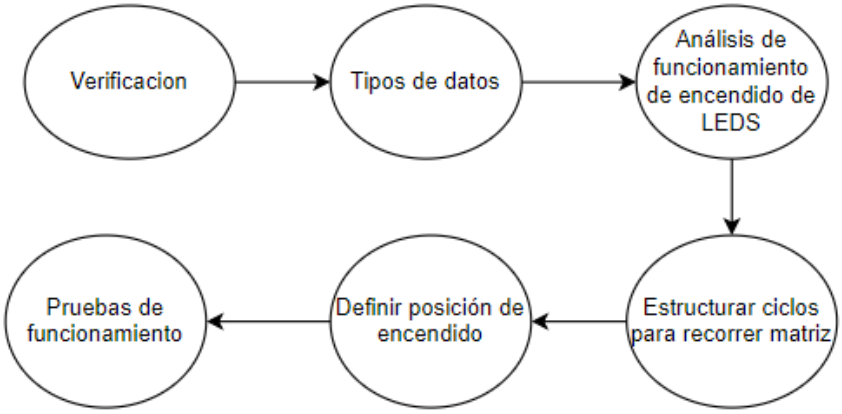
Desarrollo de la implementación

Tareas:

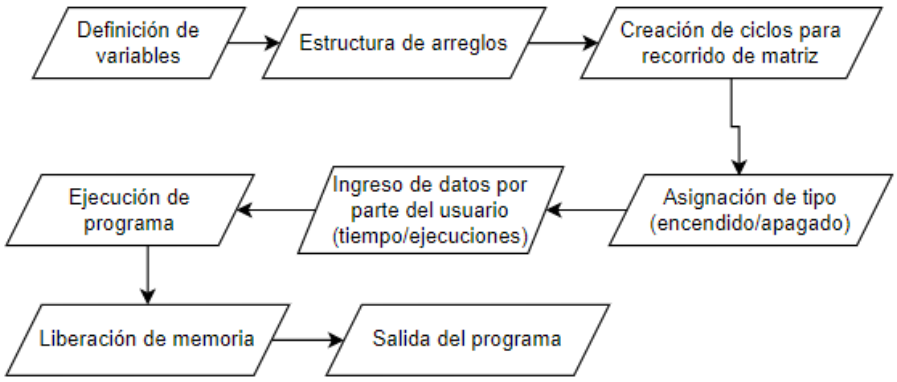
Definición de conexiones: Validación de esquemas de conectividad, entendimiento de conexión en cascada para integrados.

Manejo de Arduino uno: Revisión de documentación, opciones disponibles, manejo de puerto serial para interacciones del usuario final.

Estructuración de funciones: Las funciones requieren de ciertos pasos para su definición, dentro de los pasos principales se tuvieron en cuenta los siguientes:



En cuanto a la ejecución de variables se tiene un comportamiento general donde se busca hacer uso de ciclos para identificar cuáles son las posiciones a iluminar en cada interacción, lo que lleva a realizar un análisis general de las siguientes tareas, considerando información solicitada al usuario por medio del ingreso de información por el puerto serial.



Código

A continuación, se detalla el código implementado en las funciones. Aquella información explicada por línea previa se omitirá en el siguiente apartado de código.

Función verificación: Esta función es la encargada de verificar el funcionamiento de todos los LEDs, para ello fue llevado a cabo un análisis inicial donde se cubra la forma de realizar mediante un ciclo de recorrido de 8 filas y 8 columnas para posteriormente realizar el encendido de la misma, asi mismo se tuvo en cuenta el uso de arreglos y punteros al hacer el llamado de la función para validar la forma de indicar LEDs encendidos.

```
//Declaración de funcion verificacion que toma 4 argumentos
void verificacion(byte** leds, byte filas, byte columnas, unsigned long tiempo){
  Serial.println("Iniciando verificacion");  //Impresión de mensaje inicial
//Ciclo de encendido de LEDs
  for (int i = 0; i < filas; i++) { //Bucle de iteración sobre filas
    for (int j = 0; j < columnas; j++) { //Bucle de iteración sobre columnas
  digitalWrite(pinLatch, LOW);  //Poner el pin de latch en estado bajo
  shiftOut(pinData, pinClock, MSBFIRST, leds[i][j]); //Usa registro de desplazamiento para determinar que leds encender, tomará aquellos almacenados en leds[i][j]
  digitalWrite(pinLatch, HIGH); //Poner el pin de latch en estado alto para cargar el dato en el registro y poder activar los leds.
  delay(tiempo); //Tiempo de encendido de lo leds, se define por una variable relacionada a la función en el momento de la ejecución, será solicitada al usuario
    }
  }
//Ciclo de apagado de LEDs
  for (int i = 0; i < filas; i++) {//Bucle de iteración sobre filas
    for (int j = 0; j < columnas; j++) {  //Bucle de iteración sobre columnas
  digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, 0); //Indica el valor de 0 para apagar todos los LEDs en el ciclo.
  digitalWrite(pinLatch, HIGH);
    }
  }
  liberarMemoria(leds); // Invoca la función liberar memoria para liberar memoria asignada a una matriz dinámica generada en diversos puntos del código.
}
```

Los procedimientos para la implementación de esta función se complementan en el apartado de la función **publik()**

Función **imagen**: Función encargada de encender leds de acuerdo a la interacción con el usuario donde se indiquen cuáles son las posiciones, esta será tomada posterior a la definición de la matriz de operación.

```
void imagen(byte fila, byte columna){
  Serial.println("Patron ingresado");
  digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, ~(B00000001 << columna-1)); //Se invierten los bits (B11111101 – B00000010) y se hace uso del operador de desplazamiento a la izquierda para mover el bit 1 del binario a la izquierda buscando el led que corresponde
  shiftOut(pinData, pinClock, MSBFIRST, B10000000 >> fila-1); //128 decimal y desplazamiento a la derecho de acuerdo al valor obtenido de fila -1 en cada ciclo.
  digitalWrite(pinLatch, HIGH);
  delay(2000);
  digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, B00000000); //Patron de bits donde todos los leds se apagan
  digitalWrite(pinLatch, HIGH);
}
```

Los procedimientos para la implementación de esta función se complementan en el apartado de la función **publik().**

Función **patrones:** Función para ejecución de patrones de iluminación definidos. Esta se encuentra reducida para la explicación.

```
void patrones(unsigned long duracion) { //Parametro para determinar la duración de cada patrón
  byte** leds = new byte*[filas]; //Creación de matriz dinamica, reservando espacio de arreglos tipo
  for (byte i = 0; i < filas; i++) {     // bites para filas que a su vez reservan un espacio para los elementos de
    leds[i] = new byte[columnas]();}  //las columnas
  unsigned long inicio; //variable para llevar registro de tiempo
//Patron1
  inicio = millis(); //Almacenamiento de función para medir en tiempo en variable inicio
  while (millis() - inicio < duracion) { //Ejecución de patrón durante el tiempo definido en 'duracion'
  for (byte i = 0; i < filas; i++) {
    for (byte j = i; j < columnas - i; j++) {//Definición de ciclo i=filas, j=columnas de acuerdo a la necesidad
      encenderLED(i + 4, j);  asignacion de fila y columna resultante para funcion 'encenderLED'
      encenderLED(filas - i - 5, j); }… // asignacion de fila y columna resultante para funcion 'encenderLED'
//Patron2…
  for(byte i = 0; i < filas; i++) { //Definicion de ciclo mediante filas
    encenderLED(i , i);
    encenderLED(i, filas - i - 1); }…
//Patron3…
….
…//Patron4
  for (byte i = 0; i < 4; i++) { //Definición de ciclo para matriz superior
    for (byte j = 0; j < 4; j++) {
      encenderLED(i, i + j);
--
  for (byte i = 4; i < filas; i++) {//Definición de ciclo para matriz inferior
    for (byte j = 0; j < 4; j++) {
      encenderLED(i, (j - i + columnas - 1));
…
  for (byte i = 0; i < filas; i++) {
    for (byte j = 0; j < columnas; j++) {
      apagarLEDs(i, j); }//Recorrido en todas las filas y columnas para posteriormente realizar apagado
…
  // Liberar memoria haciendo uso de la función definida
  liberarMemoria(leds);
```

Los patrones se definen mediante ciclos que luego serán utilizados para definir cuáles serán los leds a encender.

Los procedimientos para la implementación de esta función se complementan en el apartado de la función **publik()**

Función **publik**: Diseñada para interacción con el usuario y ejecución de las funciones definidas previamente.

```
void publik(){ //Declaración de la función
  menu(); Llamado a la función menu
  while (Serial.available() == 0); //Espera hasta que haya datos disponibles por Puerto serie
  opc = Serial.parseInt(); //Leer numero entero desde Puerto serie y asignarlo a variable
  if(opc==1){ //Detecta si el numero ingresado es 1 y ejecuta los pasos siguientes
   byte rept, repa=0; Definicion de variables para control de interacciones
… tiempo = Serial.parseInt(); Define tiempo ingresado por el usuario y lo asigna a la variable
…rept = Serial.parseInt(); //Define cantidad de ciclos ingresados por el usuario para repetir
    while (repa<rept){ //Define ciclo, la variable repa si incrementa en cada ciclo
    byte** leds = new byte*[filas]; //Definición de arreglo y punteros para crear matriz
            for (byte i=0; i<filas; i++){
    leds[i] = new byte[columnas];
    for (byte j=0; j<columnas; j++) {
      leds[i][j] = 255;… //Asignación 255 para indicar encendido en la función verificacion
     verificacion(leds, filas, columnas, tiempo); //Llamado a la función verificacion
     repa++;} //Incremento de variable para duración de ciclos
            return;… //Salida del programa cuando finalice el ciclo
  else if(opc==2){
    byte fila, columna, rep, repi=0;
    Serial.println("Ingrese cantidad de posiciones que desea indicar: ");
    while (Serial.available() == 0);
    rep = Serial.parseInt(); //Ciclo de repetición para la cantidad de posiciones a ingresar
    while (repi < rep) {
    Serial.println("Ingrese posicion de los LEDS");
            Serial.println("Indique numero de fila y columna (1-8) separado por un espacio");….
    while (Serial.available() == 0);    columna = Serial.parseInt();    while (Serial.available() == 0);    fila =
Serial.parseInt();…
            if (fila >= 1 && fila <= 8 && columna >= 1 && columna <= 8) {//Condicional para búsqueda
de patron y tolerancia a errors se genera desde 1 a 8 para mayor facilidad en el manejo del usuario
final
    imagen(fila, columna); //llamado de la función imagen
    repi++;            }
            else {  Serial.println("Opcion invalida. Por favor ingresa valores entre 1 y 8."); //Mensaje
ilustrado en caso de no cumplir con el formato          }
            digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, B00000000);//Apagar todos los leds
    digitalWrite(pinLatch, HIGH);
  }   Serial.println("Ejecucion finalizada.");    return;}
else if(opc==3){
    Serial.println("Ingrese el tiempo de duración de ciclos en milisegundos:");
    while (Serial.available() == 0);
    unsigned long duracionc = Serial.parseInt();
    Serial.print("Mostrando patrones definidos");
    patrones(duracionc); //llamado de la función patrones
  }
```

Funciones adicionales:

Además de generar las funciones solicitadas en la necesidad inicial, con el fin de optimizar el uso del código evitando la generación de líneas repetitivas en tareas que se repiten, se crearon las siguientes funciones:

Función **LiberarMemoria**: Liberar memoria asignada a matriz dinámica generada.

```
void liberarMemoria(byte** leds) { //Toma argumento 'leds' que hace referencia a un puntero de un puntero de bytes, ya que la matriz generada 'leds' es una matriz de punteros a bytes.
  for (byte i = 0; i < filas; i++) { //Ciclo para recorrer las filas de la matriz
    delete[] leds[i]; //Liberar la memoria asignada a cada fila de leds, siendo leds[i] un puntero de un arreglo de bytes.
  }
  delete[] leds; //Liberar memoria asignada a la matriz 'leds' eliminando el puntero al arreglo de punteros.
}
```

Función **encenderLED**: Encender LED de una matriz de leds a partir de los índices de fila y columna.

```
void encenderLED(byte fila, byte columna) { //Toma de argumentos de fila y columna
  digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, ~(1 << (7 - fila))); //Uso de un patron de bits para invertir la fila indicada de acuerdo a las conversiones de binario a decimal, e invertimos los bits del número resultante (00010000 a 11101111). Esto se hace mediante el uso de desplazamiento de bits en los binarios, para este caso tomamos el número de la fila y lo desplazamos hacia la izquierda el número de veces determinadas por (7- fila).
  shiftOut(pinData, pinClock, MSBFIRST, B10000000 >> columna);  //128 decimal, en este caso desplazamos los bits hacia la derecha de acuerdo al valor de la variable columna en cada punto del ciclo.
  digitalWrite(pinLatch, HIGH);
}
```

Función **apagarLEDs:** Definida para apagar los leds basado en la lógica anterior

```
void apagarLEDs(byte fila, byte columna) {
  digitalWrite(pinLatch, LOW);
  shiftOut(pinData, pinClock, MSBFIRST, ~(1 << (7 - fila)));
  shiftOut(pinData, pinClock, MSBFIRST, B01111111 >> columna);  // Apagar led especifico encendido en la función anterior (B01111111 es el inverso de B10000000) lo que invierte la lógica del encendido anterior.
  digitalWrite(pinLatch, HIGH);
}
```

Función **menu**: Imprime un menú en el puerto serie brindando 3 opciones para ejecutar las funciones (patrones, verificación e imagen), este se usa como medio informativo para el manejo del usuario final.

```
void menu(){
  Serial.println("MENU PRINCIPAL");
  Serial.println("1. Encender todos los LEDs");
  Serial.println("2. Ingresar patron");
  Serial.println("3. Visualizar patrones");
  Serial.println("Ingrese una opcion: ");
}
```

Link código: https://github.com/YulianaBSu/Parcial1/blob/Parcial1YBS/parcial.cpp

Problemas encontrados

Fueron encontrados diversos problemas a lo largo de la solución, teniendo en cuenta el tipo de implementación.

Fue necesario realizar diversas pruebas en los siguientes aspectos:

- Tipo de resistencias para evitar colapso de elementos
- Conexión de integrados
- Manejo de encendido y apagado de LEDs
- Interpretación de binarios y representación en el programa

Para esto fue necesario hacer uso de la documentación y ver videos de referencia, esto llevo a encontrar una solución al planteamiento inicial.

Referencias

https://docs.arduino.cc/tutorials/communication/guide-to-shift-out

https://www.arduino.cc/reference/en/