# Manual Técnico

Practica 1
Arquitectura de Computadores y Ensambladores 1

---

**Grupo 10**

---

Integrantes

| Carnet | Nombre |
|---|---|
| 202003381 | Luisa María Ortíz Romero |
| 202000560 | Marjorie Gissell Reyes Franco |
| 202001534 | Erwin Fernando Vásquez Peñate |
| 201612332 | Carlos Emilio Campos Morán |
| 202004812 | Fredy Samuel Quijada Ceballos |
| 201504464 | Herberth Abisai Avila Ruiz |
| 202010751 | Paulo Vladimir Argueta Ortega |

# Tabla de Contenido



# Moving message

Crear una pantalla de leds de dimensiones de 10*22 leds como mínimo, se deberá tener una aplicación Java en la cual se ingrese el texto que se quiera mostrar en la pantalla, el texto mostrado en la pantalla deberá presentarse por medio de un desplazamiento a la izquierda.

El circuito debe contar con un botón de inicio/fin para iniciar el funcionamiento del circuito, así como terminar su funcionamiento.

# Requerimientos

- Proteus 8.10 SP0 (Build 29203) (Para Simulacion)
- Arduino IDE
  - Versión: 1.8.15.0
- Microsoft Windows 10

### Componentes

- 1 Arduino Mega 2560
- 6 Driver Max7219
- 1 COMPIM (COM Port Physical Interface Model)
- 1 Boton
- 6 Resistencias de 10k
- 6 Matrices 8x8

### Librerias

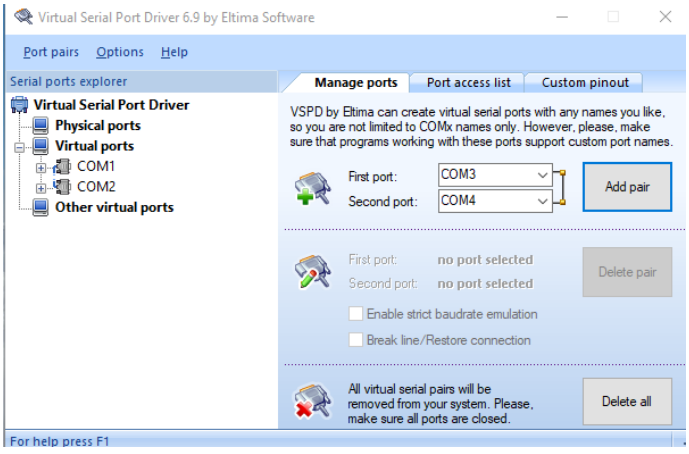
Es necesario el uso de las librerias LedControl.h (Luces led), panamahitek.Arduino (Comunicación).

```
// ARDUINO
#include <LedControl.h>

// JAVA
import panamahitek.Arduino.PanamaHitek_Arduino;
```

# Estructura del funcionamiento

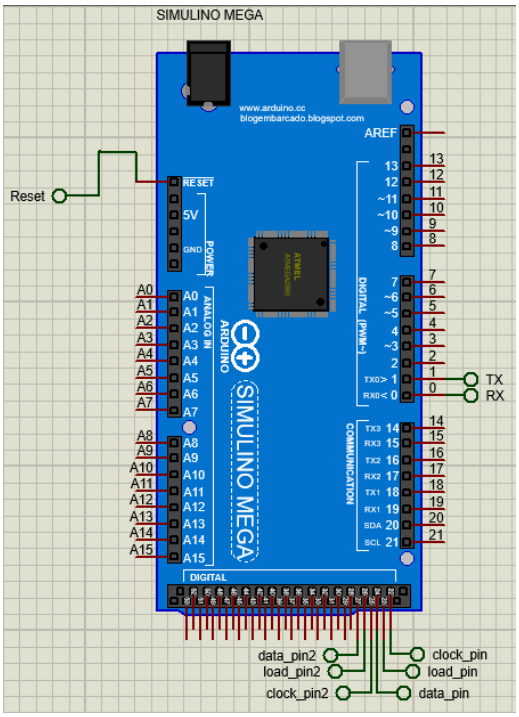
## Virtual Serial Port Driver

Puertos virtuales para la comunicacion entre el codigo Java y el Arduino en Proteus. Se crean dos puertos COM1 y COM2



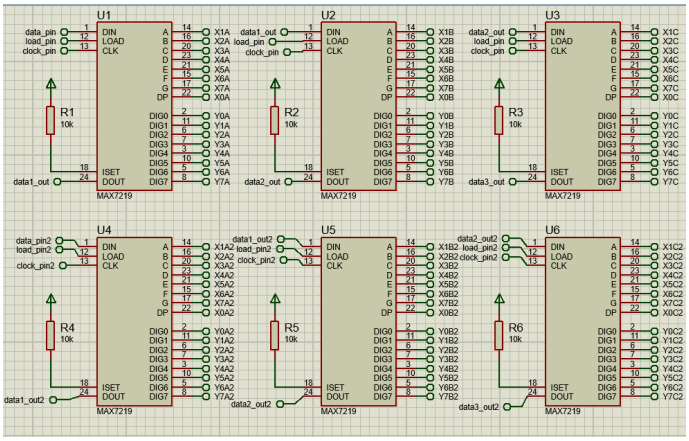
# Conexiones Proteus

## Arduino Mega



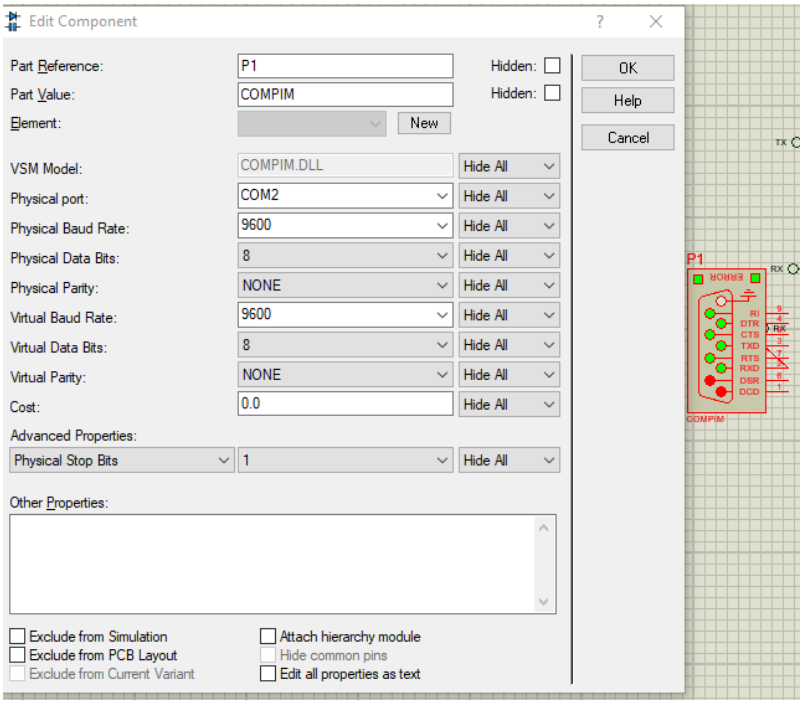

- Los pines digitales 22-24 son utilizados para el control de las matrices superiores con driver [OUTPUT].
- Los pines digitales 25-27 son utilizados para el control de las matrices inferiores con driver [OUTPUT].

## Driver MAX7219




El driver recibe como entradas los pines 24,23,22 (superior) y 27,26,25 (inferior) de arduino mega en sus entradas DIN, LOAD, CLK respectivamente.
Y como salidas tiene A-G y DP y DIG0-DIG7 como envio de datos a las matrices 8x8.
En cada driver se utiliza el DOUT del anterior como DIN del siguiente.

## COMPIM (COM Port Physical Interface Model)

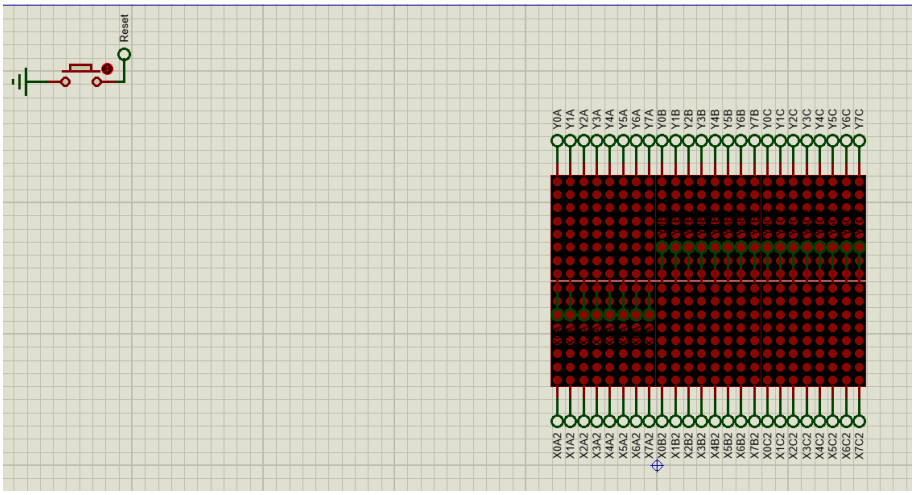



Configuración del COMPIM, establecido en el puerto COM2 para la comunciación con el arduino Mega.

### Control

El control se compone por:

- 6 Matrices
  - Cada matriz recibe las salidas del driver MAX7219 respectivo
- 1 boton
  - Se utiliza para resetear el arduino



## Comunicación con Java

### Instancias

Instacias del arreglo para almacenar las cadenas a mostrar y de la libreria que realiza la comunicación con el puerto.

```
static MessageList messageListHandler = new MessageList();
static PanamaHitek_Arduino arduinoHandler = new PanamaHitek_Arduino();
```

### Inicialización

Inicialización de la instancia del comunicador en el COM1 e inicialización de la aplicación.

```
public static void initApp() throws Exception {
    arduinoHandler.arduinoTX("COM1",9600);
    adminView();
}
```

### Envio del mensaje

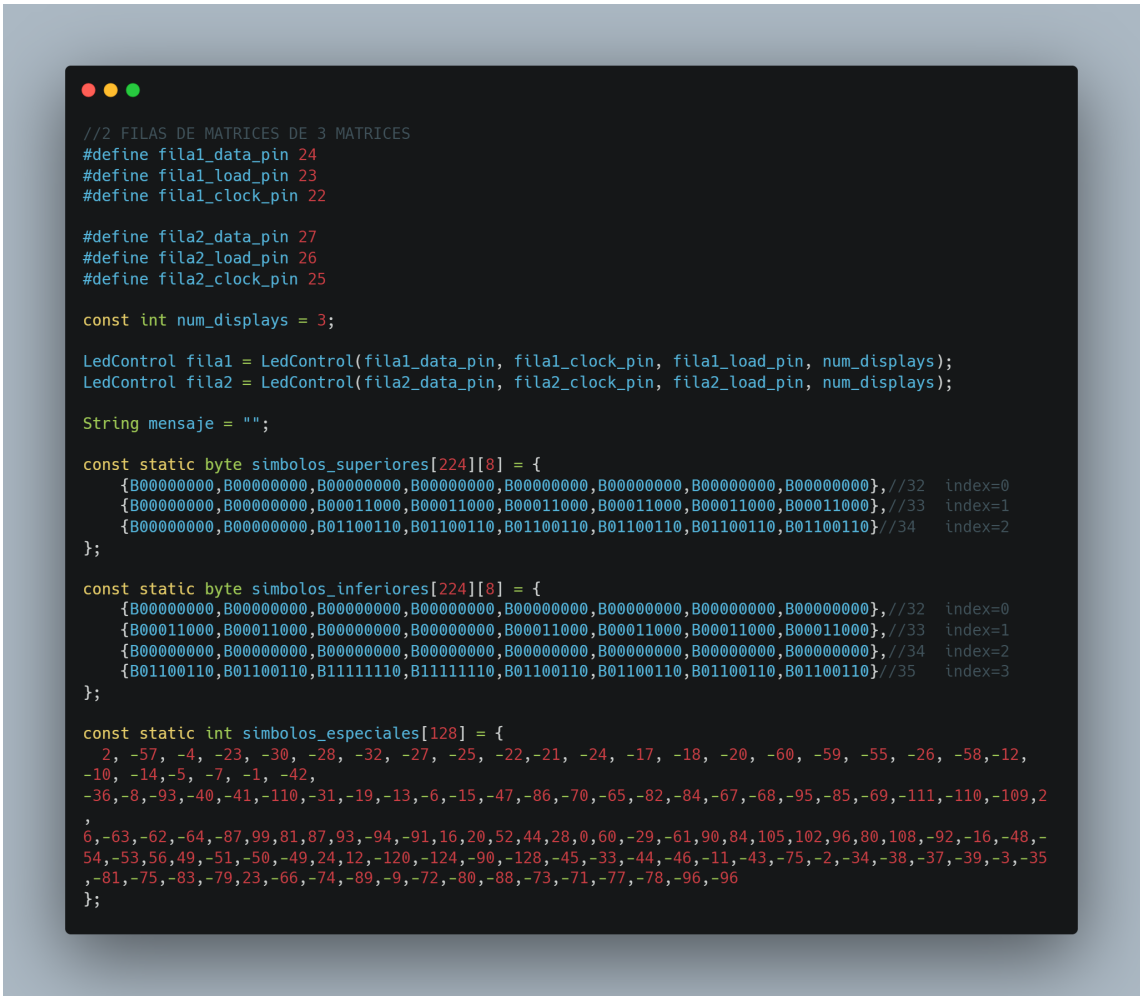
Evento que envia la cadena de caracteres que se quieren mostrar.

```
public static void adminView() throws Exception {
    // More code...

    //Send Message Button
    JButton sendMessageButton = new JButton("Send Message");
    sendMessageButton.setLayout(null);
    sendMessageButton.setVisible(true);
    sendMessageButton.setBounds(500, 630, 270, 60);
    sendMessageButton.setBackground(Color.yellow);
    sendMessageButton.setFont(font3);
    sendMessageButton.addMouseListener(new MouseAdapter(){
        public void mouseClicked(MouseEvent ecp){
            messageListHandler.finalInsert(messageTF.getText());
            adminView.dispose();
            try {
                for(int i=0;i<messageTF.getText().length();i++){
                    char c=messageTF.getText().charAt(i);
                    int charCode=c;
                    arduinoHandler.sendByte(charCode);
                }
                adminView();
            } catch (Exception e) {
                throw new RuntimeException(e);
            }

        }
    });
    adminView.add(sendMessageButton);
}
```

# Código Arduino

## Bloque de declaraciones

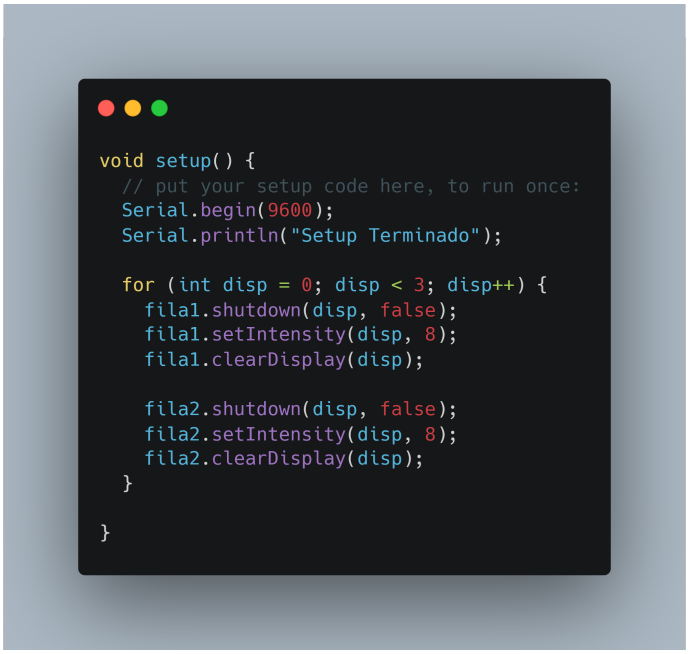
Este bloque contiene todas las variables, definiciones y constantes requeridas para el control de las matrices led y los pines definidos para el arduino.
Los arreglos static byte contiene los bytes que conforman cada uno de los caracteres a leer.



```
//2 FILAS DE MATRICES DE 3 MATRICES
#define fila1_data_pin 24
#define fila1_load_pin 23
#define fila1_clock_pin 22

#define fila2_data_pin 27
#define fila2_load_pin 26
#define fila2_clock_pin 25

const int num_displays = 3;

LedControl fila1 = LedControl(fila1_data_pin, fila1_clock_pin, fila1_load_pin, num_displays);
LedControl fila2 = LedControl(fila2_data_pin, fila2_clock_pin, fila2_load_pin, num_displays);

String mensaje = "";

const static byte simbolos_superiores[224][8] = {
    {B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000},//32  index=0
    {B00000000,B00000000,B00011000,B00011000,B00011000,B00011000,B00011000,B00011000},//33  index=1
    {B00000000,B00000000,B01100110,B01100110,B01100110,B01100110,B01100110,B01100110}//34   index=2
};

const static byte simbolos_inferiores[224][8] = {
    {B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000},//32  index=0
    {B00011000,B00011000,B00000000,B00000000,B00011000,B00011000,B00011000,B00011000},//33  index=1
    {B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000},//34  index=2
    {B01100110,B01100110,B11111110,B11111110,B01100110,B01100110,B01100110,B01100110}//35   index=3
};

const static int simbolos_especiales[128] = {
  2, -57, -4, -23, -30, -28, -32, -27, -25, -22,-21, -24, -17, -18, -20, -60, -59, -55, -26, -58,-12,
-10, -14,-5, -7, -1, -42,
-36,-8,-93,-40,-41,-110,-31,-19,-13,-6,-15,-47,-86,-70,-65,-82,-84,-67,-68,-95,-85,-69,-111,-110,-109,2
,
6,-63,-62,-64,-87,99,81,87,93,-94,-91,16,20,52,44,28,0,60,-29,-61,90,84,105,102,96,80,108,-92,-16,-48,-
54,-53,56,49,-51,-50,-49,24,12,-120,-124,-90,-128,-45,-33,-44,-46,-11,-43,-75,-2,-34,-38,-37,-39,-3,-35
,-81,-75,-83,-79,23,-66,-74,-89,-9,-72,-80,-88,-73,-71,-77,-78,-96,-96
};
```

## Bloque SETUP

Este bloque se encarga de definir el modo de los pines e inicializacion de objetos.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Setup Terminado");

  for (int disp = 0; disp < 3; disp++) {
    fila1.shutdown(disp, false);
    fila1.setIntensity(disp, 8);
    fila1.clearDisplay(disp);

    fila2.shutdown(disp, false);
    fila2.setIntensity(disp, 8);
    fila2.clearDisplay(disp);
  }

}
```

## Bloque LOOP

Este bloque contiene lo que se ejecuta ciclicamente.

```
void loop() {
  // put your main code here, to run repeatedly:
  checkIncommingData();

}
```

## Funciones y Metodos utilizados por los bloques anteriores

### checkIncommingData

Este metodo recabara los caracteres que estan siendo enviados desde la aplicacion y los almacenara de manera numerica en un arreglo.

```
void checkIncommingData() {

  if (Serial.available() > 0) {
    mensaje = "";
    while (Serial.available() > 0) {

      char dato = char(Serial.read());
      int dato_int = int(dato);

      mensaje += dato;
      Serial.println(dato_int);
    }
  }

  int arr[mensaje.length() + 1];
  for(int i = 0; i < mensaje.length(); i++){
    arr[i] = int(mensaje[i]);
  }

  scrollMsg(arr, 25, mensaje.length());
}
```

### scrollMsg

Este metodo ajustara el posicionamiento segun el tamaño de la frase.

```
void scrollMsg(const int* mensaje, unsigned long frecuencia, int size) {
  int inicio = 24;
  int pasos = size * 8;
  for (int i = inicio; i >= -pasos; i--) {

    escribirFrase(mensaje, i, size);
    if (i == 24) {
      limpiar();
    }
    // delay(frecuencia);
  }

}
```

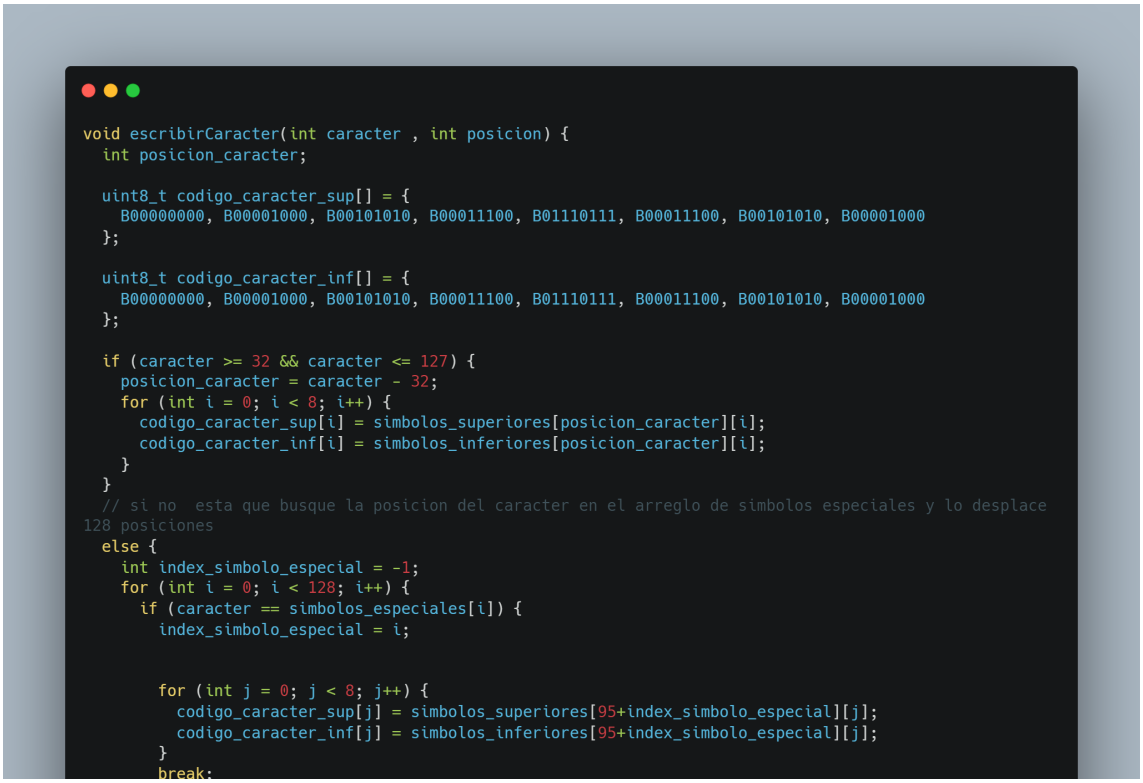
### escribirFrase

Este metodo desfragmenta la frase y envia caracter por caracter con una nueva posicion.
Si se envia otra frase mientras ya se esta ejecutando una, esta se interrumpe y se empieza a imprimir la nueva.

```
bool escribirFrase(const int* frase, int posicion, int size) {
  for (size_t i = 0; i < size; i++) {
    escribirCaracter(frase[i], (i * 8) + posicion);
    if (Serial.available() > 0) {
      //   limpiar();
      break;
    }


  }
}
```

### escribirCaracter

Este metodo contiene ya un caracter a mostrar por defecto por si no encuentra el caracter a buscar. La busqueda se realiza por el medio del numero ASCII del caracter, si lo encuentra se hace la busqueda en los arreglos destinados a contener los bytes que conforman la figura del caracter a imprimir.
Se hacen ajustes a la presentación de los bytes del caracter y se mandan a imprimir.



```
void escribirCaracter(int caracter , int posicion) {
  int posicion_caracter;

  uint8_t codigo_caracter_sup[] = {
    B00000000, B00001000, B00101010, B00011100, B01110111, B00011100, B00101010, B00001000
  };

  uint8_t codigo_caracter_inf[] = {
    B00000000, B00001000, B00101010, B00011100, B01110111, B00011100, B00101010, B00001000
  };

  if (caracter >= 32 && caracter <= 127) {
    posicion_caracter = caracter - 32;
    for (int i = 0; i < 8; i++) {
      codigo_caracter_sup[i] = simbolos_superiores[posicion_caracter][i];
      codigo_caracter_inf[i] = simbolos_inferiores[posicion_caracter][i];
    }
  }
  // si no  esta que busque la posicion del caracter en el arreglo de simbolos especiales y lo desplace
128 posiciones
  else {
    int index_simbolo_especial = -1;
    for (int i = 0; i < 128; i++) {
      if (caracter == simbolos_especiales[i]) {
        index_simbolo_especial = i;


        for (int j = 0; j < 8; j++) {
          codigo_caracter_sup[j] = simbolos_superiores[95+index_simbolo_especial][j];
          codigo_caracter_inf[j] = simbolos_inferiores[95+index_simbolo_especial][j];
        }
        break;
```

```
      }
    }

    if (index_simbolo_especial == -1) {

      //   si no lo encuentra
      for (int i = 0; i < 8; i++) {

        codigo_caracter_sup[i] = simbolos_superiores[31][i];
        codigo_caracter_inf[i] = simbolos_inferiores[31][i];
      }
    }
  }

  uint8_t temporal_sup[8] = {0, 0, 0, 0, 0, 0, 0, 0};
  for (int i = 0; i < 8; i++) {
    bitWrite(temporal_sup[7], i, bitRead(codigo_caracter_sup[i], 0));
    bitWrite(temporal_sup[6], i, bitRead(codigo_caracter_sup[i], 1));
    bitWrite(temporal_sup[5], i, bitRead(codigo_caracter_sup[i], 2));
    bitWrite(temporal_sup[4], i, bitRead(codigo_caracter_sup[i], 3));
    bitWrite(temporal_sup[3], i, bitRead(codigo_caracter_sup[i], 4));
    bitWrite(temporal_sup[2], i, bitRead(codigo_caracter_sup[i], 5));
    bitWrite(temporal_sup[1], i, bitRead(codigo_caracter_sup[i], 6));
    bitWrite(temporal_sup[0], i, bitRead(codigo_caracter_sup[i], 7));
  }
  memcpy(codigo_caracter_sup, temporal_sup, sizeof(temporal_sup[0]) * 8);

  uint8_t temporal_inf[8] = {0, 0, 0, 0, 0, 0, 0, 0};
  for (int i = 0; i < 8; i++) {
    bitWrite(temporal_inf[7], i, bitRead(codigo_caracter_inf[i], 0));
    bitWrite(temporal_inf[6], i, bitRead(codigo_caracter_inf[i], 1));
    bitWrite(temporal_inf[5], i, bitRead(codigo_caracter_inf[i], 2));
    bitWrite(temporal_inf[4], i, bitRead(codigo_caracter_inf[i], 3));
    bitWrite(temporal_inf[3], i, bitRead(codigo_caracter_inf[i], 4));
    bitWrite(temporal_inf[2], i, bitRead(codigo_caracter_inf[i], 5));
    bitWrite(temporal_inf[1], i, bitRead(codigo_caracter_inf[i], 6));
    bitWrite(temporal_inf[0], i, bitRead(codigo_caracter_inf[i], 7));
  }
  memcpy(codigo_caracter_inf, temporal_inf, sizeof(temporal_inf[0]) * 8);

  for (int i = 0; i < 8; i++) {
    int address = 0;
    int posendisplay = posicion + i;
    while (posendisplay > 7) {
      address++;
      posendisplay -= 8;
    }
    if (address > fila1.getDeviceCount() - 1 )
      return;


    fila1.setColumn(address, posendisplay, codigo_caracter_sup[i]);
    fila2.setColumn(address, posendisplay, codigo_caracter_inf[i]);


  }
}
```

### limpiar

Elimina todo lo contenido en las instancias del LedControl.

```
void limpiar() {
  for (int i = 0; i < fila1.getDeviceCount(); i++) {
    fila1.clearDisplay(i);
    fila2.clearDisplay(i);
  }
}
```