ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Documentación Práctica 1

Grupo 8

Laboratorio Arquitectura de Computadoras y Ensambladores 1

Sección N

Integrantes:

Nombre	Carnet
Rodrigo Alejandro Hernández de León	201900042
José Manuel Solis Ortiz	201800517
Andrea María Cabrera Rosito	202010918
Ana Belén Contreras Orozco	201901604
Allen Giankarlo Román Vásquez	202004745
Carlos Roberto Quixtán Pérez	201901159
Pedro Antonio Castro Calderón	201900612

Introducción

Un microcontrolador es una computadora presente en un solo circuito integrado que se dedica a realizar una tarea y ejecuta una aplicación específica. Contiene memoria, periféricos de entrada / salida programables y un procesador. Los microcontroladores están diseñados principalmente para aplicaciones integradas, algunas de ellas son: manejo de sensores, juegos, calculadoras, agendas, avisos lumínicos, secuenciador de luces, cerrojos electrónicos, control de motores, relojes, alarmas, robots, etc.

Para aplicar los conocimientos sobre estos circuitos integrados, se realizó la práctica de laboratorio que consiste en una pantalla de leds que muestre cualquier texto ingresado por medio de una aplicación, utilizando una plataforma de simulación y programación orientada a objetos.

Desarrollo de la Práctica

Herramientas utilizadas:

- Proteus 8.10
- Arduino IDE 2.0.3
- Arduino Simulino Mega
- Apache Netbeans IDE 15
- Java Development Kit 17
- Repositorio de Github: https://github.com/rodrialeh01/ACE1_Practica1_G8

Código de Arduino

Dicho código es el que ayudó con el funcionamiento del Arduino MEGA 2590, se realizó de la siguiente manera:

```
#include "Abecedario.h"
int matrixled[][24] = {
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
};
int columnas[] = {22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37};
int filas[] = {38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 2, 4, 7, 8, 10, 11, 12, 13};
```

Inicialmente, se declararon las variables globales para un mejor control de las instrucciones; se incluyó "Abecedario.h", en este archivo se guardó cada una de las matrices de letras, siendo estas de dimensiones de 16x8. Puede visualizarse como se tiene la matriz principal de LEDs "matrixled" - la cual posee dimensiones de 16x24, en esta se logra mostrar el mensaje. Por último, se declararon los pines según su control, columnas o filas.

Arduinooo | Abecedario.h

```
int A[16][8] = {
  {1, 1, 1, 1, 1, 1, 1, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 1, 1, 1, 1, 1, 1, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 0, 0, 0, 0, 1},
};
```

Ejemplo de una matriz que representa el char "A"

```
void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:
  for (int i = 0; i < 24; i++) {
    pinMode(filas[i], OUTPUT);

  }
  for (int i = 0; i < 16; i++) {
    pinMode(columnas[i], OUTPUT);
  }
  LimpiarMatriz();
}
```

Luego, se encuentra el método de setup(), en este, se logra inicializar el puerto serial para su posterior comunicación con la aplicación Java, luego se encuentra la inicialización de cada uno de los pines como OUTPUT y, al final, la inicialización de la matriz por medio de un método "LimpiarMatriz()" del cuál se hablará más adelante.

Luego se tiene el método de loop(), en este se controlan de moto iterativo las instrucciones:

```
void loop() {
  char valor = "";
  if (Serial.available() > 0) {
    valor = Serial.read();
  }

  switch (valor) {
    case 'A':
      for (int col = 0; col < 16; col++) {
        for (int fila = 0; fila < 8; fila++) {
          matrixled[col][fila] = A[col][fila];
        }
      }

      break;
```

```
void loop() {
  char valor = "";
  if (Serial.available() > 0) {
    valor = Serial.read();
  }

  switch (valor) {
    case 'A':
      for (int col = 0; col < 16; col++) {
        for (int fila = 0; fila < 8; fila++) {
          matrixled[col][fila] = A[col][fila];
        }
      }

      break;
```

Puede verse en el código, que se realizan varias instrucciones, por lo que se secciona con colores para una mejor manera de entenderlo. En el cuadro amarillo, se verifica que el puerto se encuentre encendido, esto para que cuando se ingrese el mensaje por medio de la aplicación Java se logre obtener caracter por caracter y este ser impreso en la matriz. Puede visualizarse en la sección verde que se encuentra un switch, aquí se encuentran cada uno de los casos

donde los caracteres puedan venir. Al tener el caracter, se ingresa al cuadro naranja, donde se llena la principal con la matriz del caso correspondiente.

```
for (int paso = 0; paso <= 24; paso++) {
  for (int fila = 0; fila < 24; fila++) {
    for (int col = 0; col < 16; col++) {
      if (matrixled[col][fila] == 1) {
        digitalWrite(filas[23 - paso + fila], HIGH);
        digitalWrite(columnas[col], LOW);
        delay(1);
        digitalWrite(filas[23 - paso + fila], LOW);
        digitalWrite(columnas[col], HIGH);
      }
    }
  }
}

LimpiarMatriz();
```

Luego de cada caso, se logra por medio de iteraciones la visualización junto con el desplazamiento de izquierda a derecha la visualización del mensaje. Por último, se limpia la matriz.

```
void LimpiarMatriz() {
  for (int i = 0; i < 24; i++) {
    digitalWrite(filas[i], LOW);
    if (i < 16) {
      digitalWrite(columnas[i], HIGH);
    }
  }
}
```

El método de limpiar matriz, es aquel que, como su nombre indica, limpia la matriz principal - es decir, escribe 0s en ella para que no se muestre ninguna LED prendida.

Código de Java

Dicho código es el que ayudó con el funcionamiento de la aplicación de Java para poder enviar mensajes desde la interfaz a nuestro Arduino.

Dicha, aplicación cuenta con dos componentes principales, la primera que es nuestro main que es básicamente la Interfaz de la aplicación utilizando la herramienta de Drag and Drop en el que cuenta con tres botones, uno para “Conectar” al puerto serial necesitado, el segundo llamado “Enviar” que manda nuestra texto a nuestra aplicación de Arduino y por último el botón de “Desconectar” que termina la ejecución de nuestra aplicación.

```
private void btnConectarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int i= this.cboPuertos.getSelectedIndex();
    if (arduino.conectar(i)){
        JOptionPane.showMessageDialog(this,"CONECTADO");
        this.btnConectar.setBackground(Color.green);
    }else{
        JOptionPane.showMessageDialog(this,"ERROR");
        this.btnConectar.setBackground(Color.red);
    }
}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String texto = this.txt.getText();
    texto = texto.toUpperCase();
    if(arduino.estaConectado()){
        for (int i = 0; i < texto.length(); i++) {
            char letra = texto.charAt(i);
            arduino.enviarLetra(letra);
        }
    }else{
        JOptionPane.showMessageDialog(this,"ERROR");
    }
}
```

La función del botón “Conectar” se ejecuta con ayuda de la librería jSerialComm que básicamente llena con nuestros puertos el combobox y luego recibe el seleccionado, abre el puerto, setea

los parámetros para poder hacer la conexión serial.

La función de “Enviar” recibe el texto de la caja de texto y con un ciclo itera para poder volver en una variable de tipo char cada letra de la frase, para poder ser enviada al método de enviarLetra.

La segunda clase de Java hace todas las conexiones con el Arduino, entre sus funciones y métodos están: conectar, getPuertosSerie, estaConectado, desconectar, enviarLetra.

```
public boolean conectar(int indice){
    puertoSerie = SerialPort.getCommPorts()[indice];
    puertoSerie.openPort();
    puertoSerie.setComPortParameters(9600, 8, 1, 0);
    puertoSerie.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);
    return estaConectado();
}

public static SerialPort[] getPuertosSerie() {
    SerialPort puertos[] = SerialPort.getCommPorts();
    return puertos;
}


public boolean estaConectado() {
    return puertoSerie.isOpen();
}

public boolean desconectar() {
    return puertoSerie.closePort();
}


public boolean enviarLetra(char valor) {
    try {
        puertoSerie.getOutputStream().write(valor);
        puertoSerie.getOutputStream().flush();
        return true;
    } catch (IOException ex) {
        return false;
    }
}
```

“conectar” obtiene el puerto seleccionado y lo abre, seteando los parámetros para poder hacer la conexión. “getPuertosSerie” obtiene todos los puertos seriales que cuenta nuestra computadora. “estaConectado” valida que la conexión haya sido realizada. “desconectar” desconecta el puerto. Y por último “enviarLetra” recibe letra por letra y la envía a nuestro Arduino.

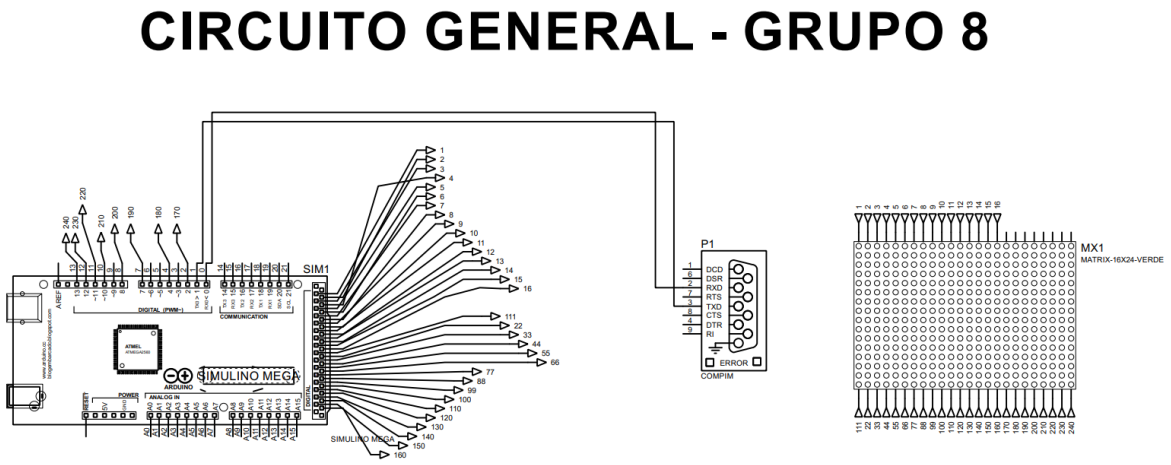
Diagramas de circuitos

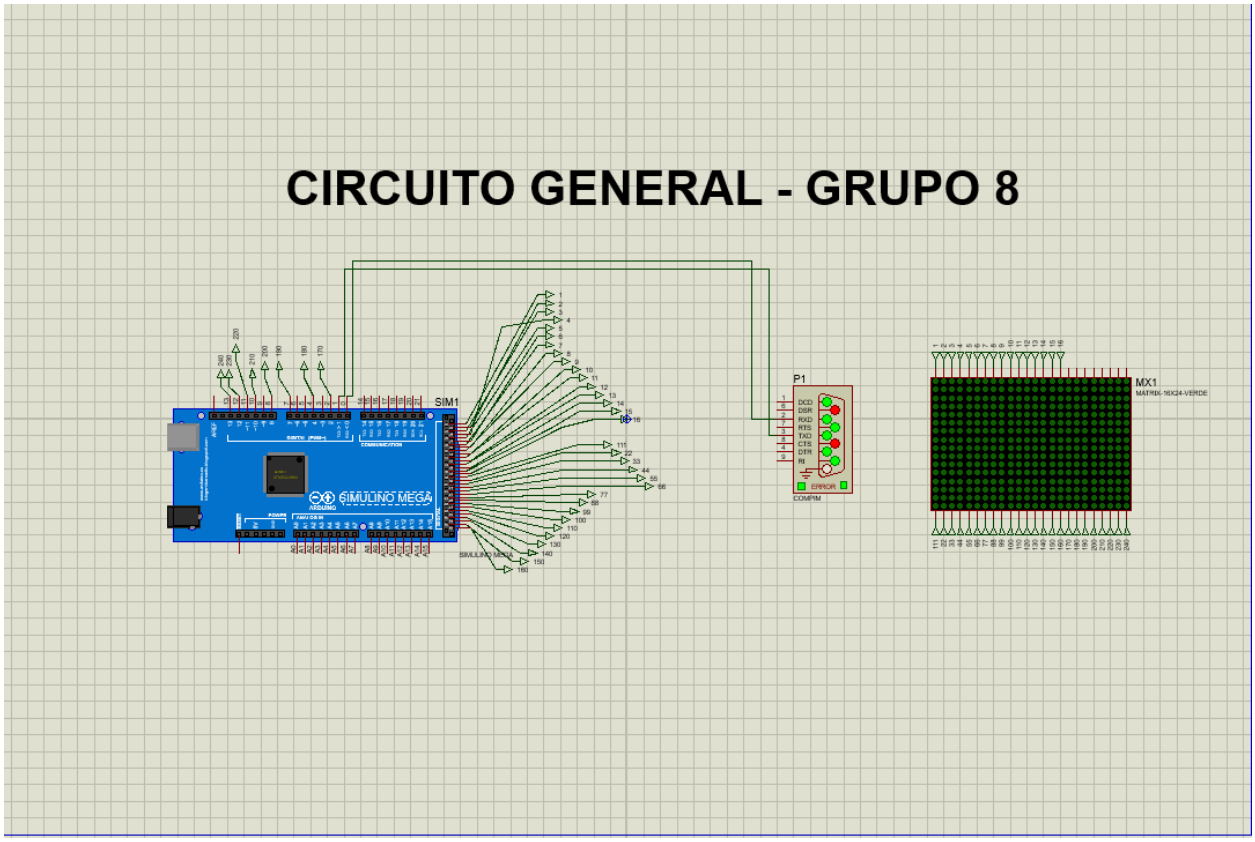
Para el desarrollo del circuito eléctrico y la simulación del arduino, se utilizó la plataforma de simulación Proteus en su versión 8.10.

Para la elaboración del circuito fueron necesarios los siguientes componentes:

- MATRIX-16X24-GREEN (Creada manualmente a partir del componente MATRIX-8X8-GREEN utilizando la herramienta descomponer y construir dispositivo, incluída en Proteus)
- SIMULINO MEGA
- COMPIM (Utilizada para la comunicación serial)

Diagrama general del circuito:



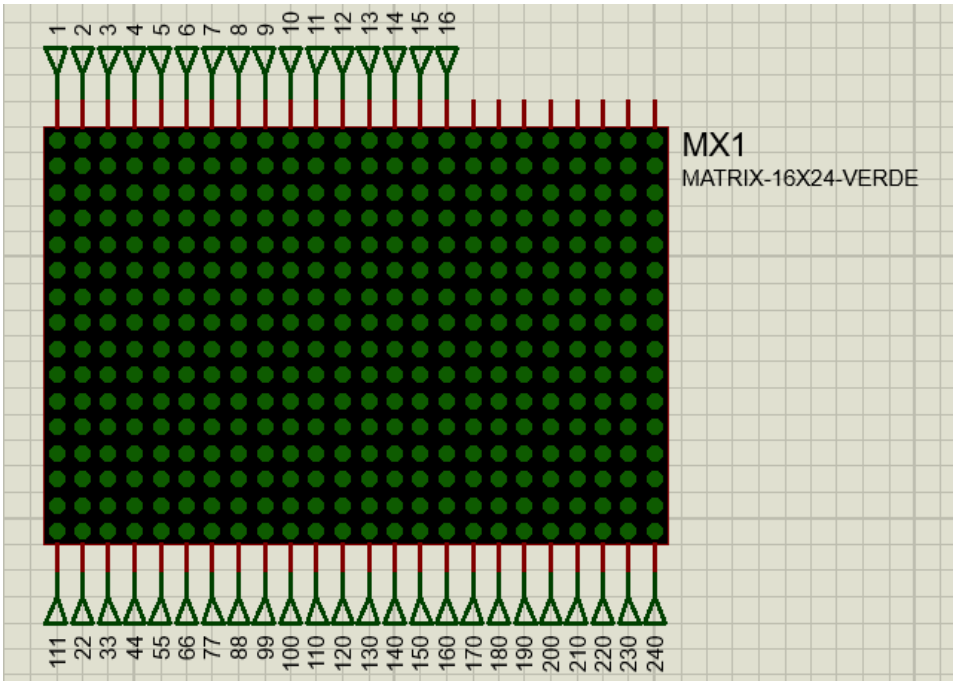


- Descripción de los pines:

Las representación de las columnas puestas en la matriz de leds son los pines
111,22,33,44,55,66,77,88,99,100,110,120,130,140,150,160,170,180,190,200, 210, 220, 230, 240.

La representación de las filas puestas en la matriz de leds son los siguientes pines:

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16



Repositorio en Github

https://github.com/rodrialeh01/ACE1_Practica1_G8