

Universidad San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Arquitectura de Computadores y Ensambladores

1



## Práctica 1 de Laboratorio

Brayan Alexander Mejia Barrientos – 201900576  
Estuardo Gabriel Son Mux – 202003894  
Fabian Esteban Reyna Juárez – 202003919  
Angel Eduardo Marroquín Canizales – 202003959  
Diego Andre Mazariegos Barrientos – 202003975  
Javier Alejandro Gutierrez de León – 202004765  
Wilber Steven Zúñiga Ruano – 202006629

**Arduino usado:** Arduino Mega 2560

Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación. (Arduino, 2022)

**Componentes Usados**

- Arduino Mega 2560
- 1 switch
- 1 Resistencia de 10k
- 6 matrices led 8x8
- 1 conector serial

## Aplicación de Java para ingresar el texto:

Se creó una aplicación en java en la cual consta de un menú con la finalidad de administrar la conexión, así como seleccionar el puerto y escribir el texto deseado para enviar.

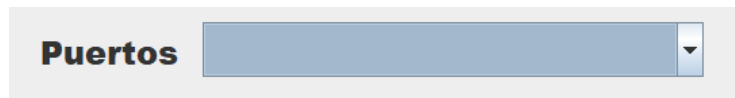
### Librerías

- jSerialComm: librería de java designada para proveer una forma independiente de acceder a los puertos seriales estándar sin requerir de librerías externas, código nativo, o cualquier otra herramienta. (Fazecast, Inc., 2022)

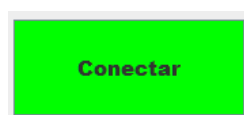
### Interfaz utilizada en la aplicación



**Puertos:** Permite seleccionar entre los puertos disponibles



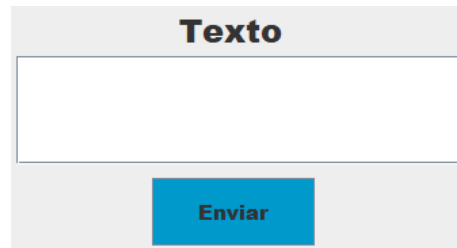
**Botón Conectar:** Establece la conexión



**Botón Desconectar:** Finaliza la Conexión



**Área para enviar texto:** Acá se escribe el mensaje deseado a mostrar en el display y se procede enviar.



**Archivos:**

**Ventana.Java**

Contiene la clase main y llama a la interfaz

**Inicio.Java**

Contiene el código del diseño utilizado

**Conexión.Java**

Contiene la clase Conexión donde por medio de la librería `javax.comm` establece conexión con los puertos y permite enviar la información referente a la cadena ingresada en la interfaz

**Funciones y métodos**

`getPuertos`: obtiene los puertos y los devuelve

```
public static SerialPort[] getPuertos() {  
    return SerialPort.getCommPorts();  
}
```

Conectar: Abre el puerto seleccionado

```
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 this.estaConectado();
}

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

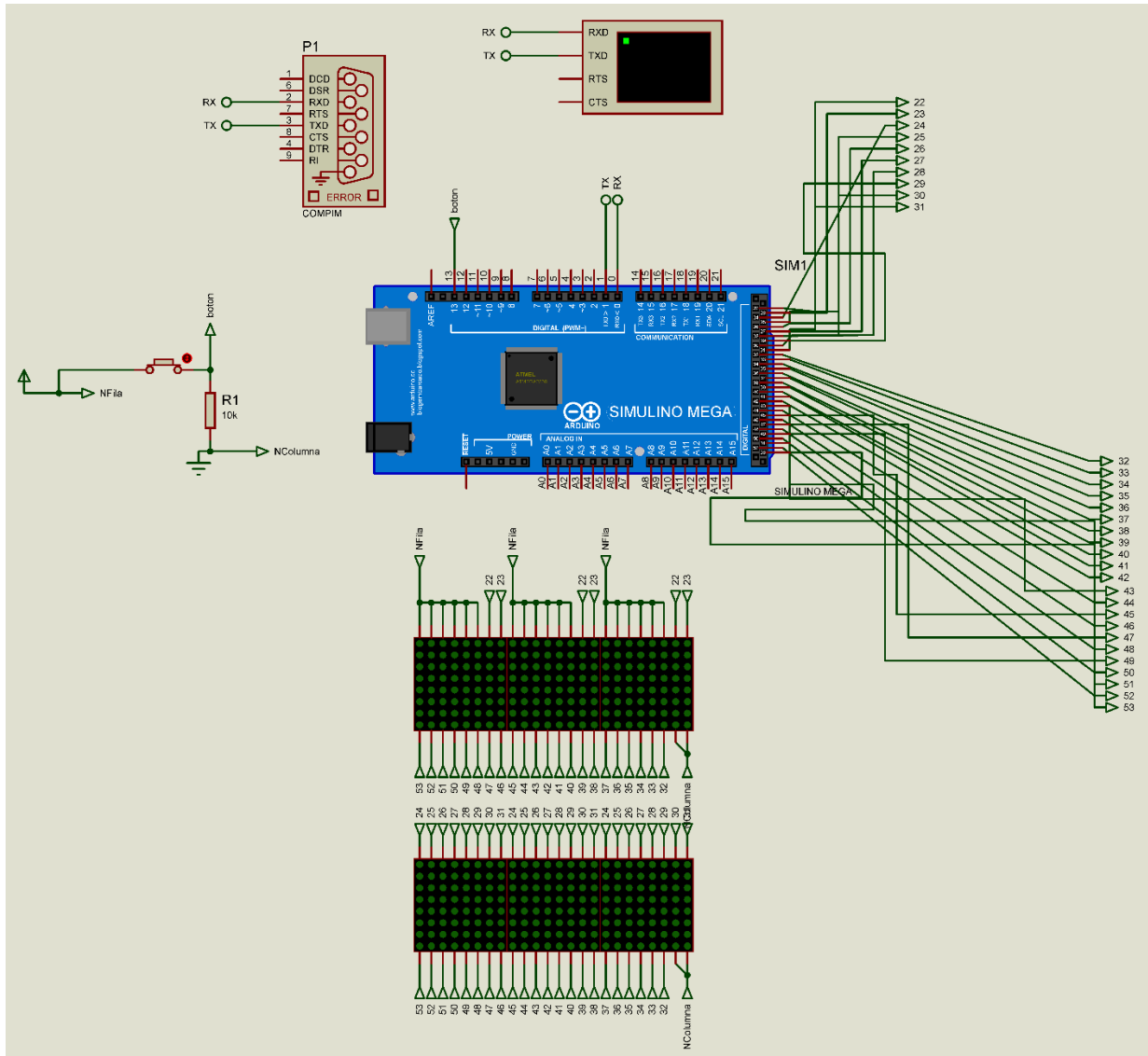
Desconectar: cierra el puerto

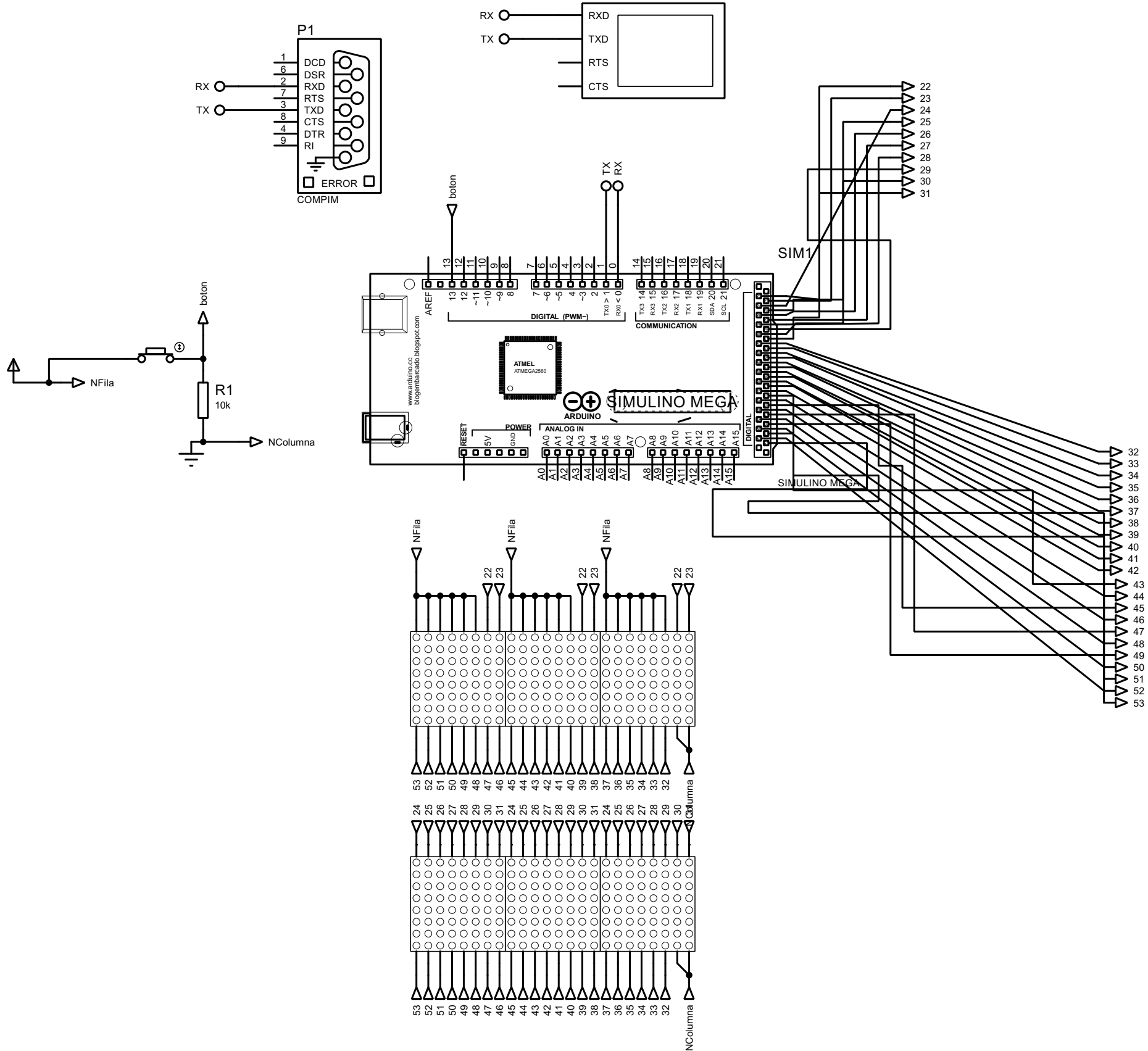
```
public boolean desconectar() {
    return puertoSerie.closePort();
}
```

enviarTexto: Solicita una cadena y luego envía los datos correspondientes por el puerto.

```
public boolean enviarTexto(String texto) {
    try {
        puertoSerie.getOutputStream().write(texto.getBytes());
        puertoSerie.getOutputStream().flush();
        return true;
    } catch (IOException ex) {
        return false;
    }
}
```

## Circuito:





## Código Arduino

Creación de variables globales

Y designación de que pines van a corresponder a filas y cual a columnas

```
int fila[] = {22,23,24,25,26,27,28,29,30,31};
int column[] = {32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53};
int cont = 0;
unsigned long tiempo;
String frase;
```

**Setup():** inicialización del puerto serial con una velocidad de 9600 baudios por segundo, del pin 13 como INPUT y dos ciclos for para recorrer los arrays de filas y columnas declarados anteriormente e inicializar sus respectivos pines

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(13, INPUT);
  for(int i = 0; i < 10 ; i++){
    pinMode(fila[i], OUTPUT);
    digitalWrite(fila[i], HIGH);
  }
  for(int i = 0; i < 22 ; i++){
    pinMode(column[i], OUTPUT);
    digitalWrite(column[i], LOW);
  }
}
```



**Loop():** método que se va a estar repitiendo en toda la ejecución

```
void loop() {  
    // put your main code here, to run repeatedly:  
    while(Serial.available() == 0){}  
    frase = Serial.readString();  
    frase.toUpperCase();  
    String cadena = generarCadena(frase);  
  
    while (digitalRead(13) == HIGH){  
        for (int c = 21; c >= 0 ; c--){  
            digitalWrite(column[c],HIGH);  
            String hexa = "";  
            hexa+=cadena[cont%cadena.length()];  
            hexa+=cadena[(cont+1)%cadena.length()];  
            hexa+=cadena[(cont+2)%cadena.length()];  
            String letraBinario = generarBinario(hexa);  
            letraBinario = letraBinario.substring(2, letraBinario.length());  
            for (int f = 0; f < 10 ; f++){  
                if(letraBinario[f] == '1'){  
                    digitalWrite(fila[f],LOW);  
                    //tiempo = millis();  
                    //while(millis() < 1+tiempo){}  
                    delay(1);  
                    digitalWrite(fila[f],HIGH);  
                }  
            }  
            cont = cont + 3;  
            digitalWrite(column[c],LOW);  
        }  
        cont = cont - 63;  
    }  
    cont = 0;  
}
```

se guarda en la variable **frase** los caracteres del buffer del serial gracias a Serial.readString(), luego se pasa a mayúsculas y se pasa **frase** como parámetro a la función generarCadena, de la cual se obtiene una nueva cadena, conformada por los respectivos códigos de cada letra, la cual se guarda en el String **cadena**.

```
while(Serial.available() == 0){}  
frase = Serial.readString();  
frase.toUpperCase();  
String cadena = generarCadena(frase);
```

El ciclo while más externo nos indica que mientras el pin 13 un estado HIGH, es decir este encendido, se repita las instrucciones, el cont – 63 regresa el contador a su posición inicial

Los for anidados controlan el encendido de las leds correspondiente así como su desplazamiento, el primer for, el más externo, decimos que se repita lo equivalente al largo de nuestra matriz, en donde colocaremos en estado HIGH al pin correspondiente del array de columnas, luego creamos un string **hexa** en el cual obtendremos de 3 en 3 un carácter de **cadena**, se llama a la función generarBinario con **hexa** como parámetro el cual nos devolverá el binario correspondiente al hexadecimal que le pasamos, posteriormente se remueve la parte que no nos interesa y se realiza el segundo for y se aumenta **cont** en tres que es la cantidad de caracteres que hemos tomado anteriormente.

El segundo for es para la filas por lo que se repite lo equivalente al alto de la matriz, y acá se analiza cada carácter del binario y si es 1 se hace el respectivo cambio de estado con el pin correspondiente del array de fila, se realiza un delay y se vuelve a cambiar.

```
while (digitalRead(13) == HIGH){
for (int c = 21; c >= 0 ; c--){
    digitalWrite(column[c],HIGH);
    String hexa = "";
    hexa+=cadena[cont%cadena.length()];
    hexa+=cadena[(cont+1)%cadena.length()];
    hexa+=cadena[(cont+2)%cadena.length()];
    String letraBinario = generarBinario(hexa);
    letraBinario = letraBinario.substring(2, letraBinario.length());
    for (int f = 0; f < 10 ; f++){
        if(letraBinario[f] == '1'){
            digitalWrite(fila[f],LOW);
            //tiempo = millis();
            //while(millis() < 1+tiempo){}
            delay(1);
            digitalWrite(fila[f],HIGH);
        }
    }
    cont = cont + 3;
    digitalWrite(column[c],LOW);
}
cont = cont - 63;
}
cont = 0;
```

**generarCadena(String frase):** se crea una variable **cadena** y se concatenan a partir de un ciclo for varios 0's, con otro ciclo for a partir de la frase que se obtiene como parámetro se lee carácter por carácter y se va concatenando a la **cadena** el código Hexadecimal correspondiente con la información de los leds a encender; finalmente se retorna la cadena.

```
String generarCadena(String frase)
{
    String cadena = "";
    for (int i = 0; i < 22; i++)
    {
        cadena += "000";
    }
    for (int i = 0; i < frase.length(); i++)
    {
        if (frase[i] == 'A')
        {
            cadena += "1FF2102102101FF";
        }
        else if (frase[i] == 'B')
        {
            cadena += "3FF2212212211DE";
        }
        else if (frase[i] == 'C')
        {
            cadena += "1FE201201201186";
        }
        ...

        else if (frase[i] == '^')
        {
            cadena += "0200C01000C0020";
        }
        else if (frase[i] == ' ')
        {
            cadena += "0000000000";
        }
        else
        {
            cadena += "0EC1122391120EC"; //sorpresa :v
        }
        cadena += "000000";
    }
    return cadena;
}
```

**generarBinario(String hexdec):** recibe un código hexadecimal con el cual genera y retorna su binario correspondiente.

```
String generarBinario(String hexdec) {
    String cadena = "";
    int i = 0;

    while (hexdec[i]) {

        switch (hexdec[i]) {
            case '0':
                cadena+="0000";
                break;
            case '1':
                cadena+="0001";
                break;
            case '2':
                cadena+="0010";
                break;
            case '3':
                cadena+="0011";
                break;
            ...

            case 'E':
            case 'e':
                cadena+="1110";
                break;
            case 'F':
            case 'f':
                cadena+="1111";
                break;
            default:
                break;
        }
        i++;
    }
    return cadena;
}
```

## **Bibliografía**

Arduino. (2022). *Arduino Mega*. Obtenido de Arduino.cl:  
<https://arduino.cl/producto/arduino-mega-2560/>

Fazecast, Inc. (2022). *What is jSerialComm?* Obtenido de jSerialComm:  
<https://fazecast.github.io/jSerialComm/>