

# **SISTEMA DE CONTROL DE ACCESO POR HUELLA DACTILAR**

**PROGRAMACIÓN DE SISTEMAS EMBEBIDOS EN RED**

**FACULTAD DE INFORMÁTICA  
UNIVERSIDAD DE MURCIA**



**PEDRO NICOLAS GOMARIZ**

**[pedro.nicolasg@um.es](mailto:pedro.nicolasg@um.es)**

**Enero 2021/2022**

## ÍNDICE

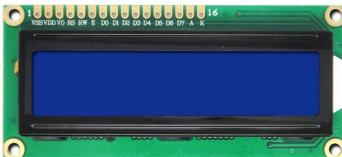
<b>1.- DESCRIPCIÓN DEL PROYECTO.</b>	<b>2</b>
<b>2.- MATERIALES.</b>	<b>2</b>
<b>3.- ARDUINO.</b>	<b>3</b>
3.1.- ESQUEMÁTICO.	4
3.2.- FINGERPRINT Y BIBLIOTECA Adafruit-Fingerprint.	5
3.3.- FUNCIONAMIENTO GENERAL DEL .ino.	7
<b>4.- INTERFAZ GRÁFICA EN JAVA.</b>	<b>9</b>
4.1.- DIAGRAMA DE CLASES DEL PROGRAMA.	11
4.2.- PERSISTENCIA DE USUARIOS.	12
<b>5.- COMUNICACIÓN SERIE ARDUINO-UI Java.</b>	<b>12</b>
<b>6.- CONCLUSIÓN.</b>	<b>15</b>

## 1.- DESCRIPCIÓN DEL PROYECTO.

El desarrollo del proyecto consiste en la implementación de un sistema de control de acceso basado en huella dactilar. Este tipo de sistemas podrían ser empleados a la entrada de ciertos recintos donde solo tienen acceso determinados usuarios autorizados, como podría ser una empresa, un gimnasio, una sala privada de estudio, etc... Además también permitirá llevar un registro de quiénes han accedido al recinto junto con la fecha y hora.

Para la implementación de dicho sistema utilizaremos Arduino, junto con otros componentes para recoger los datos de los clientes que intenten acceder al recinto, dichos datos serán posteriormente procesados por un programa Java que nos permitirá interpretarlos y mostrarlos de forma amigable mediante una UI. Todo esto lo analizaremos más adelante.

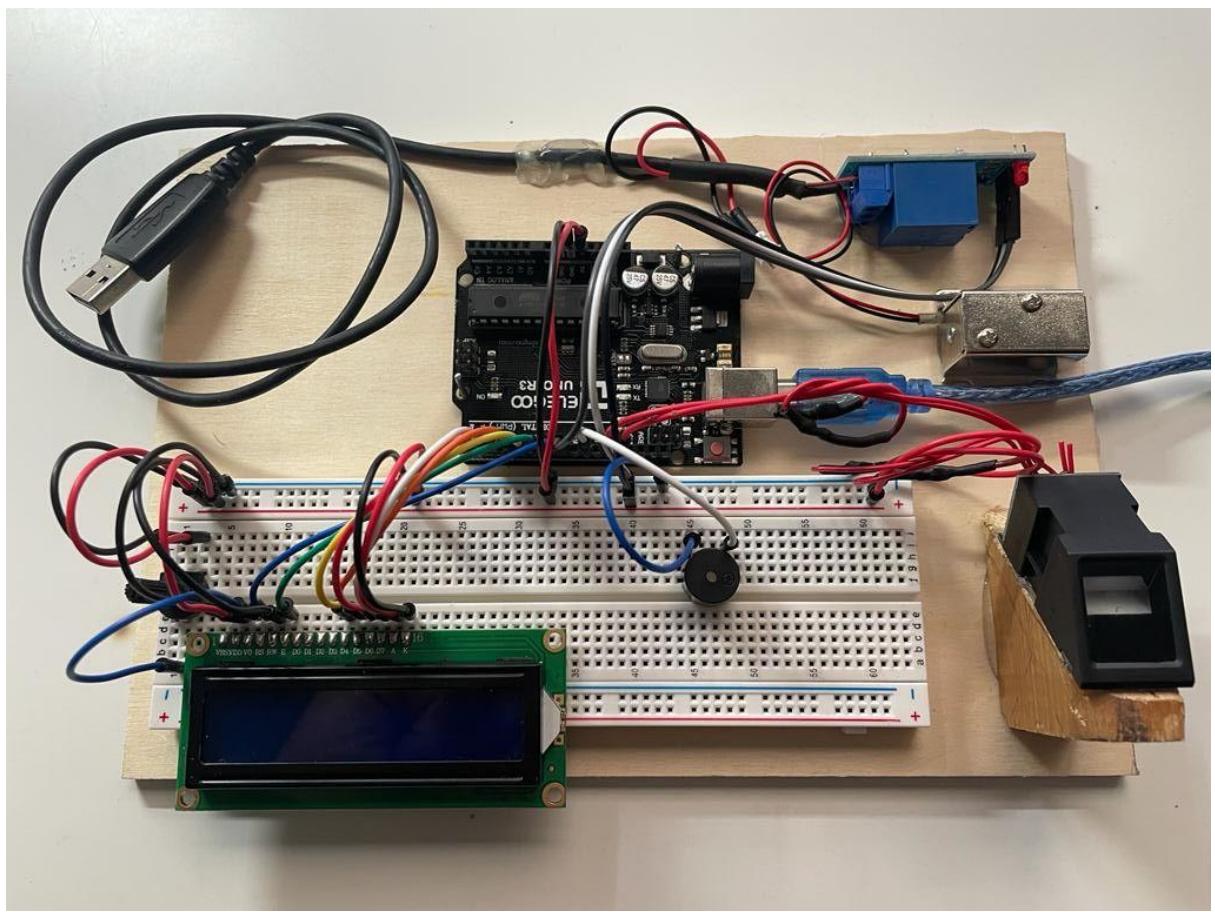
## 2.- MATERIALES.

	PLACA ARDUINO UNO	
	LCD 2X16	<a href="https://www.amazon.es/gp/product/B01MXGST4I/ref=ppx_yo_dt_b_asin_title_o05_s00?ie=UTF8&amp;psc=1">https://www.amazon.es/gp/product/B01MXGST4I/ref=ppx_yo_dt_b_asin_title_o05_s00?ie=UTF8&amp;psc=1</a>
	BUZZER ACTIVO	

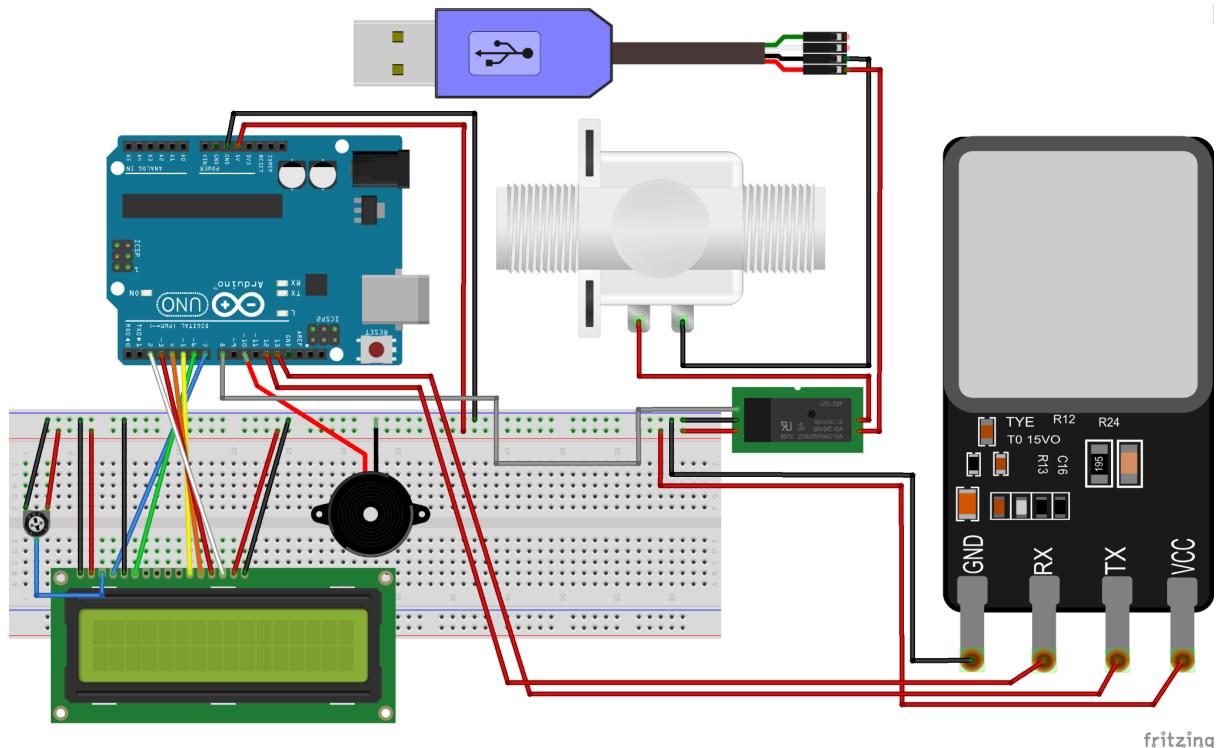
	<b>FINGERPRINT SENSOR</b>	<a href="https://es.aliexpress.com/item/32982364002.html?spm=a2g0o.9042311.0.0.ce4c63c0IUgl3I">https://es.aliexpress.com/item/32982364002.html?spm=a2g0o.9042311.0.0.ce4c63c0IUgl3I</a>
	<b>MÓDULO DE RELÉ 5V</b>	<a href="https://es.aliexpress.com/item/1005002604628994.html?spm=a2g0o.9042311.0.0.c4e4c63c0IUgl3I">https://es.aliexpress.com/item/1005002604628994.html?spm=a2g0o.9042311.0.0.c4e4c63c0IUgl3I</a>
	<b>ELECTROCERRADURA 5V</b>	<a href="https://es.aliexpress.com/item/32820530084.html?spm=a2g0o.9042311.0.0.ce4c63c0IUgl3I">https://es.aliexpress.com/item/32820530084.html?spm=a2g0o.9042311.0.0.ce4c63c0IUgl3I</a>

### **3.- ARDUINO.**

Para el desarrollo del proyecto se montan los componentes tal y como se puede ver en la siguiente imagen y esquemático. Por un lado tenemos el microcontrolador Arduino Uno, con el que estaremos alimentando y controlando el LCD para mostrar información relevante, por otro lado el fingerprint para poder reconocer las huellas de los usuarios, registrar nuevas o eliminar las que sean necesarias y además cuando acceda un usuario válido estaremos accionando el relé, el cual hará que la electrocerradura que está alimentada por una fuente de corriente externa de 5V, en mi caso por USB se accione, permitiendo el paso del cliente al recinto. Esta apertura de cerradura vendrá acompañada por un sonido que emitirá el zumbador para avisar al cliente de que puede realizar el acceso.



### 3.1.- ESQUEMÁTICO.



### 3.2.- FINGERPRINT Y BIBLIOTECA Adafruit-Fingerprint.

Este sensor nos permitirá asegurar nuestro recinto con datos biométricos, ya que es capaz de detectar y verificar huellas dactilares de una forma muy simple. Este está compuesto por un chip DSP que realiza el renderizado de imagen, el cálculo y la búsqueda de funciones e imágenes, esto permite que al conectarlo a casi cualquier microcontrolador sea posible enviar paquetes de datos para tomar fotos, detectar impresiones, hash y buscar, además de registrar nuevas huellas directamente. Además este posee una memoria flash integrada en la cual es capaz de almacenar hasta 127 huellas.

#### Principio de funcionamiento:

A través del principio de imagen óptica, las líneas creadas por las diferencias de la piel en el lado interno del dedo formarán varias imágenes de huellas dactilares. La textura de la piel es diferente en patrones, puntos de interrupción e intersecciones. Se les llama en procesamiento de información. "Puntos característicos", las características de cada dedo son diferentes, es decir, únicas, confiando en esta singularidad, podemos asociar a una persona con su huella digital, al almacenar previamente su huella digital comparando las huellas dactilares, puede verificar su verdadera identidad.

El sistema de identificación de huellas dactilares recopila, analiza y compara las huellas dactilares a través de un equipo de conversión fotoeléctrica especial y tecnología de procesamiento de imágenes, la identidad personal y puede identificar de forma automática, rápida y precisa. El sistema incluye principalmente procesos como la adquisición de imágenes de huellas dactilares, procesamiento de imágenes de huellas dactilares, extracción de características y comparación y coincidencia de valores de características.

Este sensor, ya nos proporciona toda la funcionalidad que necesitamos para implementar nuestro proyecto, ya que los requisitos que tenemos son los siguientes:

- Registrar una nueva huella en el sistema cuando llega un nuevo usuario.
- Eliminar una huella de un usuario dado para revocarle el acceso.
- Comprobar si un usuario puede acceder al recinto porque previamente ha sido registrado en el sistema utilizando su huella dactilar.

además, este sensor viene acompañado de una librería, *AdaFruit-Fingerprint* (<https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library>), la cual nos va a proveer de toda funcionalidad para que podamos utilizar este sensor con mucha facilidad, entre otras cosas, nos provee de una carpeta denominada “examples” donde tenemos a nuestra disposición distintos programas Arduino de ejemplo, entre los que podemos destacar:

- **enroll.ino:** Este programa de ejemplo pondrá el sensor en modo registro, esto nos permitirá registrar una nueva huella dactilar en el sensor, a la cual se le asignará un ID (identificador) para identificarla más adelante.

```

COM3

Adafruit Fingerprint sensor enrollment
Found fingerprint sensor!
Reading sensor parameters
Status: 0x0
Sys ID: 0x0
Capacity: 150
Security level: 3
Device address: FFFFFFFF
Packet len: 128
Baud rate: 57600
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
Enrolling ID #9
Waiting for valid finger to enroll as #9
.

.

Image taken
Image converted
Remove finger
ID 9
Place same finger again
.....Image taken
Image converted
Creating model for #9
Prints matched!
ID 9
Stored!
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
```

Autoscroll  Mostrar marca temporal Retorno de carro ▾

- **fingerprint.ino:** Este programa de ejemplo pondrá el sensor en modo escaneo, de forma que estará constantemente escaneando huellas dactilares, en caso de que esté registrada nos dirá cual es el ID (identificador) asociado a la misma, en caso contrario nos indicará que la huella no está registrada.

```

COM3

Adafruit finger detect test
Found fingerprint sensor!
Reading sensor parameters
Status: 0x0
Sys ID: 0x0
Capacity: 150
Security level: 3
Device address: FFFFFFFF
Packet len: 128
Baud rate: 57600
Waiting for valid finger...
Sensor contains 4 templates
No finger detected
No finger detected

No finger detected
No finger detected
Image taken
Image converted
Found a print match!
Found ID #9 with confidence of 270
No finger detected
No finger detected
```

```
No finger detected
No finger detected
Image taken
Image converted
Did not find a match
No finger detected
No finger detected
```

- **delete.ino:** Este programa de ejemplo pondrá el sensor en modo de eliminación, esto nos permitirá indicarle un ID, el cual está asociado a una huella, lo que provocará la eliminación de la huella asignada a dicho ID.



```
Delete Finger
Found fingerprint sensor!
Please type in the ID # (from 1 to 127) you want to delete...
Deleting ID #9
Deleted!
```

### 3.3.- FUNCIONAMIENTO GENERAL DEL .ino.

Puesto que nuestro Arduino necesita cambiar entre los tres comportamientos en tiempo de ejecución, he elaborado un programa en el cual he definido tres funciones, *enroll*, *eliminar* y *fingerprint*, en las cuales he añadido la funcionalidad implementada en cada uno de los ejemplos mencionados anteriormente. El comportamiento definido en el método principal del programa es que por defecto el sensor se encontrará en el modo *fingerprint*, pero en el caso de que nos llegue un 1 por el puerto serie, entonces el modo del sensor cambiará a *enroll*, en caso de llegar un 2 entonces el modo del sensor será *delete*. De esta forma en tiempo de ejecución podremos seleccionar el modo que deseemos.

```
// ----- LOOP -----
void loop()
{
    if (Serial.available() > 0) {
        int mode = Serial.read();
        switch (mode) {
            case '1':
                enroll();
                break;
            case '2':
                eliminar();
                break;
            default:
                Serial.println("Mode no disponible.");
        }
    }
    else {
        fingerprint();
    }
}
```

Además, también se añaden las distintas instrucciones para realizar la apertura de la electrocerradura y para que el zumbador produzca el sonido cuando se ha introducido una huella que es válida:

```

// found a match!
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);

// Abrimos la cerradura y activamos el zumbador:
digitalWrite(pin_ZUM, HIGH);
delay(70);
digitalWrite(pin_ZUM, LOW);
delay(70);
digitalWrite(pin_ZUM, HIGH);
delay(70);
digitalWrite(pin_ZUM, LOW);

digitalWrite(pin_RELÉ, LOW);
delay(2500);
digitalWrite(pin_RELÉ, HIGH);

```

En el caso de que se introduzca una huella no válida emitimos un sonido distinto para informar del intento de acceso:

```

} else if (p == FINGERPRINT_NOTFOUND) {

    Serial.println("Did not find a match");

    // Realizamos el sonido de denegación
    digitalWrite(pin_ZUM, HIGH);
    delay(250);
    digitalWrite(pin_ZUM, LOW);

```

De la misma forma también se añaden ciertos sonidos para que mientras se está produciendo el proceso de registro de una nueva huella, el usuario sepa cuando debe posicionar su huella en el sensor.

Además, cuando se cambia el modo en el que está operando el sensor de huellas, se añaden las instrucciones correspondientes para mediante el display informar.

```

void fingerPrint()                void enroll() {
{
    lcd.clear();                  lcd.clear();
    lcd.setCursor(0, 0);          lcd.setCursor(0, 0);
    lcd.print("APP Acceso");     lcd.print("APP Acceso");
    lcd.setCursor(0, 1);          lcd.setCursor(0, 1);
    lcd.print("MODE: Fing.Print"); lcd.print("MODE: Enroll");

void eliminar()
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("APP Acceso");
    lcd.setCursor(0, 1);
    lcd.print("MODE: Delete");
}

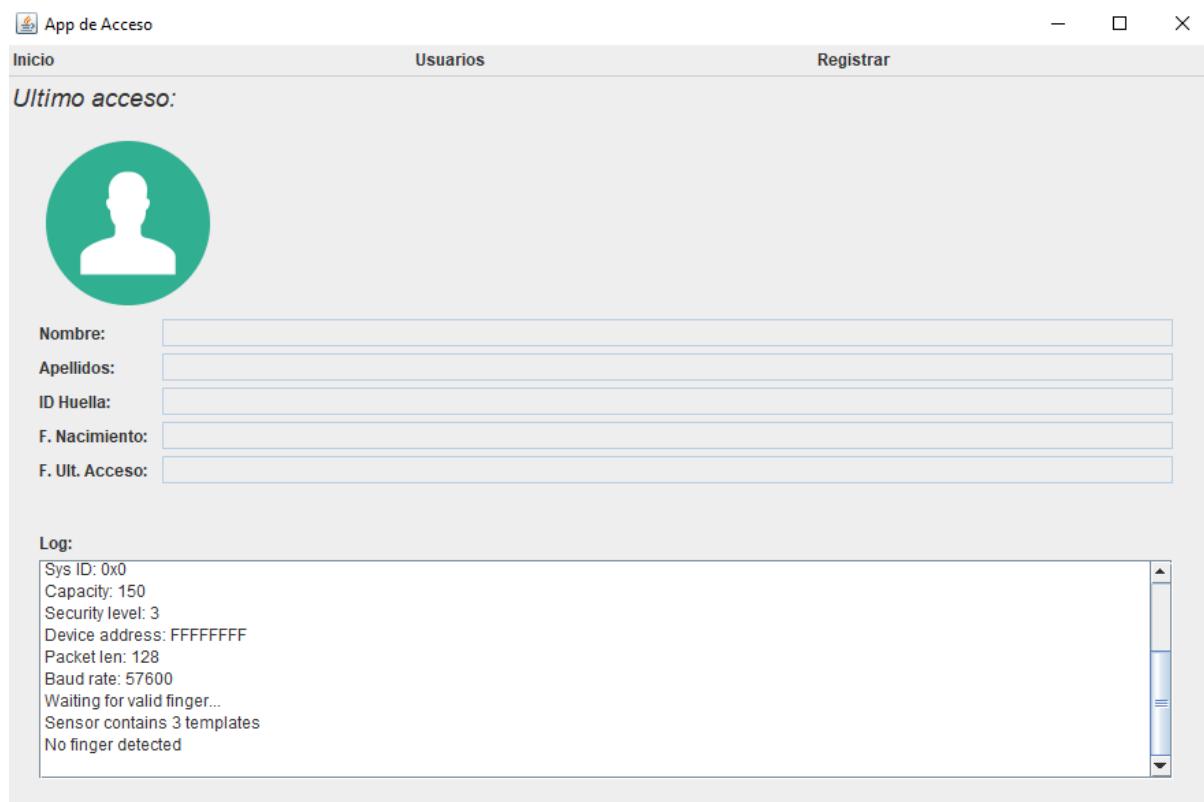
```

#### 4.- INTERFAZ GRÁFICA EN JAVA.

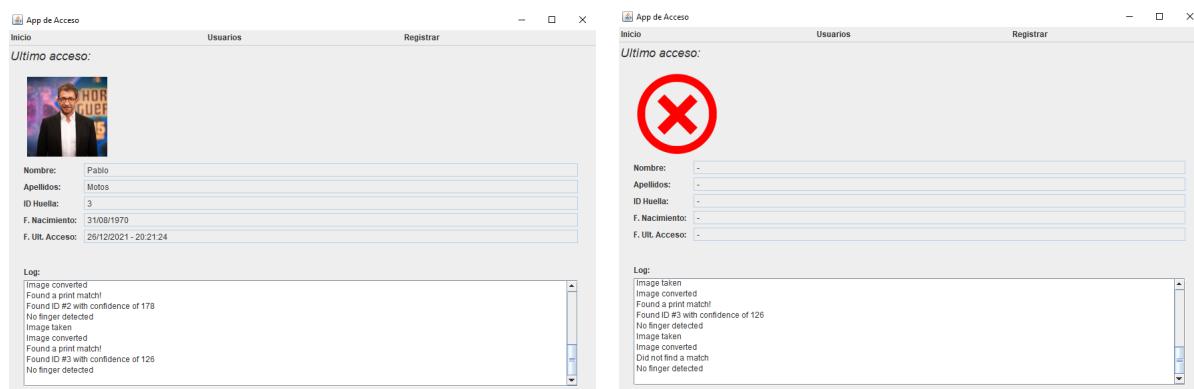
Se diseña una aplicación en Java, con interfaz de usuario desarrollada con Swing para representar de forma gráfica la información que nos llega del sensor de huellas a través de Arduino, y además poder controlar el programa Arduino comentado anteriormente de forma muy intuitiva a través de botones. También será utilizada para llevar el control y gestión de usuarios.

Para ello se definen tres ventanas:

- **Inicio:** El programa muestra los datos del último usuario que accedió al recinto, además podemos observar en el área de texto a modo de log todos los mensajes que nos van llegando del Arduino. En el caso de que intente acceder un usuario sin acceso aparecerá indicado mediante un ícono de aspa roja.



(hacer zoom)



- Usuarios:** En esta ventana podemos ver una tabla con todos los usuarios registrados en el sistema y sus datos, seleccionando uno de ellos podemos eliminarlo haciendo click en el botón. Esto provocará su eliminación de la BD y además se eliminará su huella del sensor poniendo el mismo en modo *delete*. A partir de este momento dejará de tener acceso al recinto.

App de Acceso

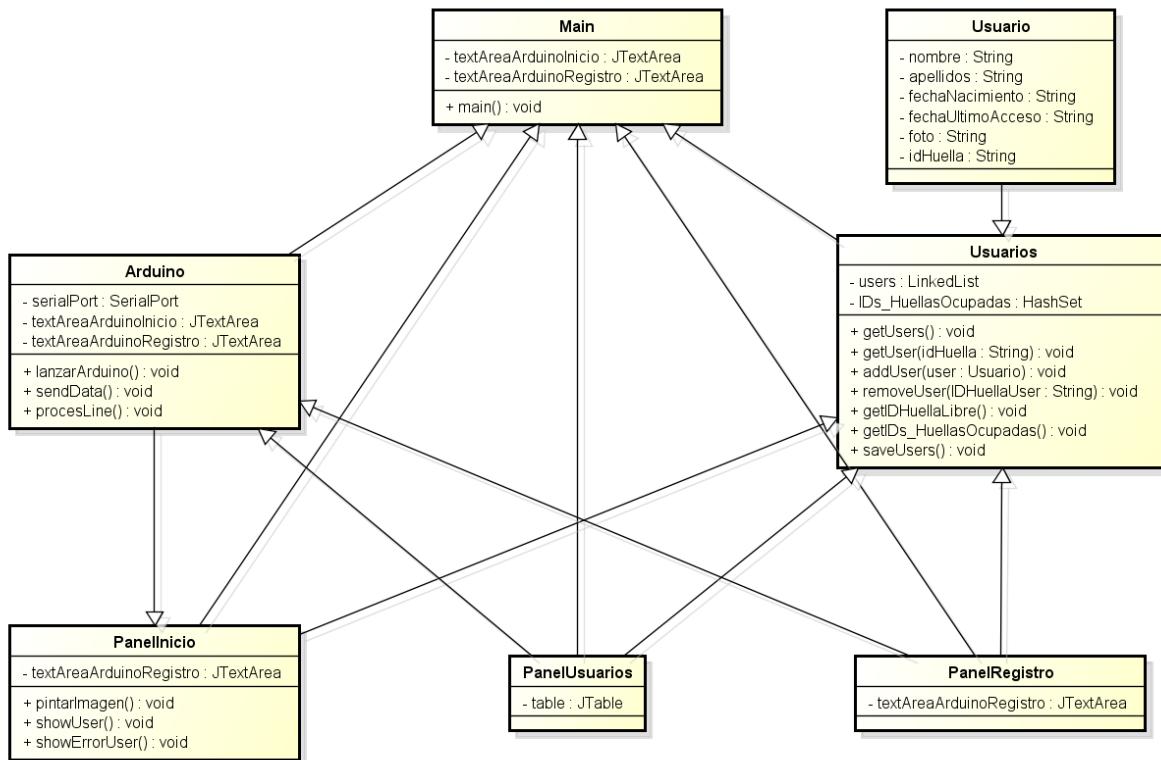
Inicio	Usuarios	Registrar		
<b>Usuarios:</b>				
Nombre	Apellidos	F. Nacimiento	ID Huella	F. Ult. Acceso
Pedro	Nicolas Gomariz	29/09/1999	1	26/12/2021 - 20:21:31
Elon	Musk	14/03/1965	2	26/12/2021 - 22:42:07
Pablo	Motos	31/08/1970	3	26/12/2021 - 22:42:12
<b>Eliminar</b>				

- Registrar:** En esta ventana podemos observar un formulario para recoger los datos de un nuevo usuario a registrar. Una vez rellenados haremos click en el botón “Registrar Huella”, entonces el sensor se pondrá en modo *enroll* y el cliente la introducirá para registrarla en el sensor, una vez registrada haciendo click en el botón “Seleccionar Foto” se abrirá un explorador de archivos que nos permita seleccionar una foto del nuevo cliente. Por último haremos clic en el botón de “Registrar” y a partir de este momento el usuario ya estará almacenado en la BD y su huella en el sensor, por lo que ya tendrá acceso al recinto.

App de Acceso

Inicio	Usuarios	Registrar
<b>Registrar nuevo usuario:</b>		
Nombre:	<input type="text"/>	
Apellidos:	<input type="text"/>	
F. Nacimiento:	<input type="text"/>	dd/mm/yyyy
ID Huella:	<input type="text"/>	
Foto:	<input type="file"/>	
<b>Registrar</b>		
<b>Log:</b>		
Sys ID: 0x0 Capacity: 150 Security level: 3 Device address: FFFFFFFF Packet len: 128 Baud rate: 57600 Waiting for valid finger... Sensor contains 3 templates No finger detected		

#### 4.1.- DIAGRAMA DE CLASES DEL PROGRAMA.



- **Main:** Clase principal de Swing, la cual define la aplicación, el menú de la misma, y carga el panel de inicio como predeterminado. También crea una instancia de la clase *Usuarios*, y otra de la clase *Arduino*, ejecutando además su método *lanzarArduino()*.
- **PanelInicio:** Clase de Swing que define todos los componentes del panel de inicio.
- **PanelUsuarios:** Clase de Swing que define todos los componentes del panel de usuarios.
- **PanelRegistro:** Clase de Swing que define todos los componentes del panel de registro.
- **Usuario:** Clase Java que define los atributos de un usuario del sistema e implementa sus métodos CRUD.
- **Usuarios:** Clase Java que hace de punto de conexión entre el frontend y backend, se encarga de gestionar el conjunto de usuarios, añadir, eliminar, obtener uno en concreto, obtener el conjunto, y además se encarga de gestionar la [persistencia](#) de usuarios como más tarde explicaremos.
- **Arduino:** Clase Java que se encarga de todo lo relacionado con la conexión y comunicación con el Arduino a través del puerto serie. Esto lo veremos más adelante en el [apartado 5](#).

## **4.2.- PERSISTENCIA DE USUARIOS.**

Para almacenar los usuarios del sistema junto a sus atributos utilizamos un fichero de texto plano, este fichero contiene por cada línea los datos de uno de los usuarios en formato JSON, si le echamos un vistazo:

```

1 {"nombre": "Pedro", "apellidos": "Nicolás Gomariz", "fechaNacimiento": "29/09/1999", "fechaUltimoAcceso": "27/12/2021 - 12:44:12", "foto": "C:\\Users\\Pedro Nicolás\\Documents\\eclipse-workspace-PSER\\fotos\\Pedro.jpg"}
2 {"nombre": "Elon", "apellidos": "Musk", "fechaNacimiento": "14/03/1965", "fechaUltimoAcceso": "27/12/2021 - 12:43:02", "foto": "C:\\Users\\Pedro Nicolás\\Documents\\eclipse-workspace-PSER\\fotos\\Elon.jpg"}
3 {"nombre": "Pablo", "apellidos": "Motos", "fechaNacimiento": "31/08/1970", "fechaUltimoAcceso": "27/12/2021 - 12:43:17", "foto": "C:\\Users\\Pedro Nicolás\\Documents\\eclipse-workspace-PSER\\fotos\\Pablo.jpg"} 
```

Para ello, cuando se arranca el programa, la clase *Usuarios*, se encarga de buscar este fichero, en caso de que exista entonces lo leerá y por cada línea creará un objeto de la clase *Usuario* con los atributos correspondientes y será añadido a la *LinkedList* *users* de la clase *Usuarios*. A partir de este momento el programa ya podrá hacer uso de los mismos, puesto que están en memoria será sencillo realizar operaciones para eliminar, editar o añadir usuarios.

En el momento que la aplicación se cierre, es decir finalice la ejecución, se ejecutará el método *saveUsers()* de la clase *Usuarios*, el cual se encarga de realizar el proceso contrario, crear o reescribir el fichero *user* con el nuevo registro de todos los usuarios y sus atributos correspondientes.

Esto está claro que en un sistema real no sería demasiado conveniente ya que no escalaría demasiado y sería poco seguro, al tener que tener todos los usuario del sistema cargados en memoria, pero debido a que se trata de una aplicación de prueba pequeña y este ámbito se sale de la práctica y asignatura he decidido utilizar esta solución ya que era la más sencilla.

## **5.- COMUNICACIÓN SERIE ARDUINO-UI Java.**

Para que nuestra aplicación de escritorio pueda interactuar con nuestro Arduino vamos a emplear la comunicación serie, es decir, a través del cable USB que conecta el Arduino al ordenador.

Para ello con nuestro programa Java trataremos de leer los datos que el arduino escribe en el puerto serie cuando ejecuta la operación *Serial.println()*; y con nuestro programa Java tendremos que escribir datos en el puerto serie que seran leidos cuando nuestro Arduino ejecute una instrucción del tipo *Serial.read()*:

Como ya hemos visto de todo esto se encarga la clase *Arduino* haciendo uso de la librería *jSerialComm* (<https://fazecast.github.io/jSerialComm/>) la cual nos proporcionara un objeto denominado *SerialPort* el cual nos permitirá implementar la funcionalidad mencionada.

De esta clase podemos destacar las siguientes funciones:

- **lanzarArduino():** Esta función se encarga de realizar la conexión serie con el Arduino empleando la librería *jSerialComm* anteriormente mencionada. Además define un nuevo hilo que se encarga de ir leyendo línea a línea los datos que van llegando del Arduino, cada una de estas líneas serán procesadas por la función *processLine()*. En el caso de que lleguen distintas líneas repetidas estas serían ignoradas.

```

public void lanzarArduino() {

    serialPort = SerialPort.getCommPort("COM3");
    serialPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER, 0, 0);
    if (!serialPort.openPort())
        System.out.println("No es posible conectarse al Arduino.");

    Thread thread = new Thread() {
        @Override
        public void run() {
            Scanner scanner = new Scanner(serialPort.getInputStream());
            String previousLine = "";
            while (scanner.hasNextLine()) {
                try {

                    String line = scanner.nextLine();
                    if (!line.equals(previousLine))
                        procesLine(line);
                    previousLine = line;

                } catch (Exception e) {
                }
            }
            scanner.close();
        }
    };
    thread.start();
}

```

- **processLine():** Función que se encarga de tomar las acciones oportunas en función de los datos que nos llegan del Arduino.

```

public void procesLine(String line) {

    textAreaArduinoInicio.append(" " + line + "\n");
    textAreaArduinoRegistro.append(" " + line + "\n");

    if (line.startsWith("Found ID")) {
        Pattern pat = Pattern.compile("#([0-9]*)");
        Matcher mat = pat.matcher(line);
        mat.find();
        panelInicio.showUser(mat.group(1));
    }

    if (line.startsWith("Did not find a match")) {
        panelInicio.showErrorUser();
    }
}

```

Lo primero que hace es añadir los mensajes que nos llegan en los *textArea* de log de los paneles de inicio y registro. Posteriormente, comprueba la línea, en caso que de que comience por “*Found ID*” significa que la huella introducida pertenece a un usuario cuya huella está registrada, por lo que extraemos de la línea el identificador de la huella y llamaremos el método *showUser* de la clase *PanelInicio* pasándole el

identificador de la huella. Este método se encarga de mostrar los datos de dicho usuario en el panel de inicio.

En el caso de que la línea sea “Did not find a match” significa que la huella introducida pertenece a un usuario que no está registrado en el sistema, por lo que en este caso se llamará al método `showErrorUser` de la clase `Panellnicio` el cual se encarga de mostrar el mensaje de acceso denegado en el panel de inicio.

Los ejemplos de esto los podemos ver en el [apartado 4](#).

- **`sendData()`:** Función que se encarga de enviar el String que se le pasa como parámetro a través del puerto serie empleando la librería `jSerialComm`. Esta función la utilizaremos para enviar distintos valores que nos permitirán cambiar el fingerprint de modo en tiempo de ejecución, tal y como veíamos en el [apartado 3.3](#). Además nos permite también enviar el identificador de la huella a eliminar cuando este se encuentre en el modo `delete`, o el identificador de una nueva huella a registrar cuando este se encuentre en el modo `enroll`.

```
public void sendData(String data) {
    serialPort.writeBytes(data.getBytes(), data.length());
}
```

Si observamos el uso de dicha función cuando se presiona el botón para eliminar un usuario:

```
JButton btnNewButton = new JButton("Eliminar");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow();
        if (row != -1) {
            String idUser = (String) table.getValueAt(row, 3);
            usuarios.removeUser(idUser);
            model.removeRow(row);
            arduino.sendData("2");
            arduino.sendData(idUser);
            JOptionPane.showMessageDialog(null, "Usuario eliminado correctamente.");
        } else
            JOptionPane.showMessageDialog(btnNewButton, "No has seleccionado el usuario a eliminar.", "ERROR",
                0);
    }
});
panel_sur.add(btnNewButton);
```

Podemos ver cómo en primer lugar se envía un 2 para poner el sensor en modo `delete`, y posteriormente se le envía el ID de la huella a eliminar.

Si observamos el uso de dicha función cuando se presiona el botón para registrar una nueva huella:

```

JButton btnNewButton = new JButton("Registrar Huella");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String idHuella = Integer.toString(usuarios.getIDHuellaLibre());
        textFieldHuella.setText(idHuella);
        arduino.sendData("1");
        arduino.sendData(idHuella);
    }
});
```

Podemos ver cómo en primer lugar se envía un 1 para poner el sensor en modo *enroll*, y posteriormente se le envía el ID al que se asociará la nueva huella registrada.

## **6.- CONCLUSIÓN.**

La realización de esta práctica me ha sido bastante satisfactoria, y aunque ha sido larga, ya que la he ido realizando prácticamente desde la mitad del cuatrimestre, me ha permitido poner en práctica y aprender sobre muchos aspectos. El primer desafío fué la realización de la interfaz gráfica en Swing, ya que nunca antes había realizado ninguna y no sabía cómo desarrollarlo, pero gracias a esto aprendí, y después pude ponerlo en práctica incluso en otra asignatura. Posteriormente, cuando recibí el sensor de huellas, tuve que investigar por mi cuenta como conectarlo al Arduino y cómo funcionaba para poder ponerlo en marcha, esto me llevó a descubrir la librería de *AdaFruit-Fingerprint* sobre la cual también tuve que investigar y aprender para poder utilizarla. Después cuando recibí la electrocerradura procedí a conectarla al Arduino y me di cuenta que no funcionaba, en un principio no tenía idea de porque sería, ya que la alimentación era de 5V, al igual que la del Arduino, esto me llevo varios días buscando por internet y aprendiendo algo de electronica basica, hasta que llegué a la conclusión de que lo que ocurría era que ambos tenían una tensión de 5V, pero el Arduino no era capaz de emitir la suficiente intensidad de corriente como para alimentar la bobina de la cerradura, por lo que tuve que comprar un relé y aprender a utilizarlo para poder alimentar la cerradura con una fuente de alimentación externa que si emitirse la tensión e intensidad que la cerradura requería. El último de los desafíos fue realizar la comunicación entre mi Arduino y mi programa Java a través del puerto serie, ya que tras investigar bastante por internet, encontré numerosas librerías como por ejemplo la de *PanamaHitek\_Arduino* ([http://panamahitek.com/libreria-panamahitek\\_arduino](http://panamahitek.com/libreria-panamahitek_arduino)) y otras más, las cuales me proveían de la funcionalidad necesaria para realizar esto, pero con varias de ellas tuve problemas y no conseguía hacer que funcionasen como mi programa lo requería, hasta que finalmente tras buscar e investigar mucho logré encontrar la librería *jSerialComm* que tenía todo lo que necesitaba y funcionaba correctamente.

Como conclusión final, esto me ha permitido no solo aprender sobre Arduino sino también, a ser capaz de conocer distintos sensores y librerías y hacerlos funcionar, también me ha servido para aprender a desarrollar interfaces gráficas en Java con Swing y también a mejorar mis habilidades de programación en dicho lenguaje.