UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# Sistema de seguridad domótica para control de acceso mediante autenticación con huella dactilar

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: David Núñez Martínez

**Tutor**: Carlos Tavares Calafate

2020-2021

# Resumen

El proyecto propone, mediante el uso de la tecnología Arduino, implementar un sistema de autentificación y seguridad de la entrada de una vivienda o estancia privada. El sistema es capaz de almacenar las huellas dactilares de diferentes usuarios con el objetivo de poder administrar el acceso a la vivienda. Además, al proyecto se le suma un sistema de seguridad capaz de detectar si cualquier persona no autorizada intenta acceder al recurso protegido utilizando el acceso por huella dactilar; para ello detectará el perfil no registrado y hará saltar una alarma. Dicha alarma puede ser desactivada empleando un llavero o tarjeta, perteneciente al dueño de la estancia o autorizado y vinculada previamente al sistema, leído por un sensor RFID. Por último, el acceso control del sistema por parte del usuario es llevado a cabo mediante el uso de una aplicación móvil desarrollada en Android Studio que se comunicará con el sistema por medio de la tecnología Bluetooth.

**Palabras clave:** Arduino, seguridad, sensores, aplicación móvil, Bluetooth.

# Abstract

This project puts forward an authentication and security system using the Arduino technology to monitor the entrance of a private accommodation. The system's objective is mange the access of the accommodation through the storage of fingerprints of different authorized users. Furthermore, the project implements a security system able to detect if any outsider has accessed to the accommodation, trying to be dodging the fingerprint sensor and it will turn on a buzzer if any unauthorized access has been detected. This buzzer can be disabled by bringing a key ring or an access card, belonging to the owner of the accommodation or any authorized user and previously enrolled with the system, closer to the RFID sensor. Finally, user's access to system features is carried out through a mobile application developed in Android Studio that it uses Bluetooth technology to communicate with the system.

**Keywords:** Arduino, security, sensor, mobile application, Bluetooth.

# Índice de contenidos

# Índice de ilustraciones

# Índice de tablas

# 1. Introducción

## 1.1 Motivación

Desde siempre me he considerado un amante de la tecnología y soy muy aficionado a consumir contenido sobre las actualizaciones tecnológicas que sufren los productos de las grandes compañías actuales, tanto en los aspectos software como hardware.

Escogí la intensificación que he cursado sobre mis estudios (Ingeniería Informática, con intensificación en Ingeniería de Computadores) por el motivo planteado anteriormente, pero, a pesar de ello, no he tenido muy claro el futuro de los mismo. Decidí embarcarme en mi proyecto TFG como una fusión entre los conocimientos adquiridos en la intensificación en cuanto a la parte orientada a la parte física y funcionalidad del proyecto y los servicios proporcionados por la aplicación móvil, más centrada en el campo del desarrollo de aplicaciones y tecnologías de la información, de la cual me gustaría seguir formándome y especializándome en ella.

Otro motivo que me animó a plantearme este proyecto es que considero la domótica y el IoT aspectos de gran interés que vez van adquiriendo un papel más transcendente en nuestra sociedad a medida que las grandes empresas del sector tecnológico lanzan al mercado un amplio abanico de productos económicamente alcanzables y fáciles de emplear y configurar.

## 1.2 Objetivos

El objetivo principal del proyecto es la implementación de un sistema de seguridad que permita acceder mediante autentificación biométrica a un grupo de usuarios autorizados a la misma estancia donde el sistema esté instalado. Este sistema también hará sonar una alarma si se intenta acceder sin éxito. Por último, la alarma podrá ser desactivada por el usuario autorizado mediante un dispositivo acreditativo vinculado al sistema.

Al finalizar el proyecto, se pretenden cubrir los siguientes cinco objetivos:

1) Poder registrar, leer, editar y eliminar las huellas dactilares de diferentes usuarios mediante la interacción con la aplicación de móvil.
2) Poder acceder a la estancia únicamente con la huella dactilar.
3) Monitorizar los accesos a la estancia.
4) Activar la alarma únicamente al detectar varios intentos de acceso no autorizados.
5) Desactivar la alarma mediante la lectura del sensor RFID y el uso del llavero o tarjeta asociada al sistema.

## 1.3 Estructura de la memoria

En este apartado trataremos la estructura que va a seguir esta memoria y los aspectos que se van a abordar en cada capítulo.

- **Capítulo 1. Introducción:** Presentación del tema escogido para este proyecto y resumen de la funcionalidad del sistema. También se expondrán los motivos que llevan al desarrollo de este proyecto. Por último, se enumeran los objetivos que el sistema pretende alcanzar y la estructura de la mismo.
- **Capítulo 2. Contexto:** Descripción del uso de la tecnología Arduino y los sensores de huella digital aplicados actualmente al ámbito de la seguridad y comparación de diferentes sistemas ya implementados con la propuesta realizada.
- **Capítulo 3. Especificaciones del problema:** Síntesis sobre como los usuarios interaccionarán con el sistema y los requisitos funcionales y no funcionales que ha de cumplir el mismo, exponiendo así tanto las funcionalidades de la aplicación como las del sistema.
- **Capítulo 4. Diseño de la solución:** Exposición de las tecnologías empleadas en la propuesta, diseño de los esquemas de conexión de los diferentes componentes y de las herramientas y entorno de desarrollo que se han utilizado.
- **Capítulo 5. Desarrollo de la solución:** Descripción del proceso de evolución de la propuesta, y de las decisiones tomadas en cuanto a diseño y funcionalidad.
- **Capítulo 6. Pruebas:** Informe que recoge el comportamiento del sistema, en su versión final, a ciertas interacciones con el usuario con tal de comprobar si cumple las necesidades de este.
- **Capítulo 7. Conclusiones y trabajos futuros:** Evaluación final de la propuesta y verificación del cumplimiento de los objetivos marcados inicialmente. Además, se plantearán posibles propuestas con el fin de mejorar y completar el proyecto.

# 2. Contexto

En este apartado se tratarán la trayectoria de las placas programables de ámbito no profesional, en nuestro caso Arduino, y su gran popularización a lo largo de los años. Asimismo, se describirán tanto el concepto como la evolución de los lectores de huellas y sus aplicaciones actuales, incluyendo sus propósitos en el ámbito de la seguridad. Por último, contrastaremos diferentes propuestas de proyectos orientados a la seguridad y que emplean esta tecnología con la escogida para este proyecto.

## 2.1 Concepto de placa programable Arduino

Para antes poder hablar del concepto "Arduino" debemos hablar primero del concepto de microcontrolador. Un microcontrolador es un circuito integrado capaz de ser programado y almacenar ordenes en su memoria. Incluye las unidades funcionales propias de un computador: procesador, memoria RAM, memoria ROM y periféricos de entrada y salida (1).

Arduino es una plataforma de código abierto basada en el uso simplificado de software y hardware propios de la marca o compatibles. Centrándonos en el hardware, las placas Arduino implementan microcontroladores de la marca Atmel. Debido a que este tipo de placas se caracterizan por la integración de múltiples entradas digitales y analógicas, son compatibles con una gran diversidad de componentes y periféricos (2).

La misma marca ofrece distintos modelos de placa según las exigencias del proyecto en el que se van a implementar, donde varía el número de entradas, su tamaño, la conectividad o la memoria integrada (3).

Según la página de la marca, sus placas se encuentran clasificadas en diferentes categorías (3):

- **Gama de entrada**: Nos encontramos con el modelo más popular de toda la familia de placas Arduino; Arduino UNO. Este modelo fue lanzado junto a la primera versión del software Arduino IDE y en implementar la conectividad USB.
- **Funciones mejoradas**: En esta categoría destacamos el Arduino Nano 33 BLE el cual se recomienda para proyectos *wearable* y se caracteriza por ser el más pequeño de las placas Arduino, su procesador y su conectividad Bluetooth Low Energy, NFC y un sensor inercial de 9 ejes.
- **Internet of Things**: Esta categoría incluye productos orientados al mundo del Internet of Things, los cuales destacan por su conectividad inalámbrica. El Arduino UNO Wifi Rev.2 es el modelo que se empleará para esta propuesta y se caracteriza principalmente por su conexión BLE, WiFi y su procesador avanzado.

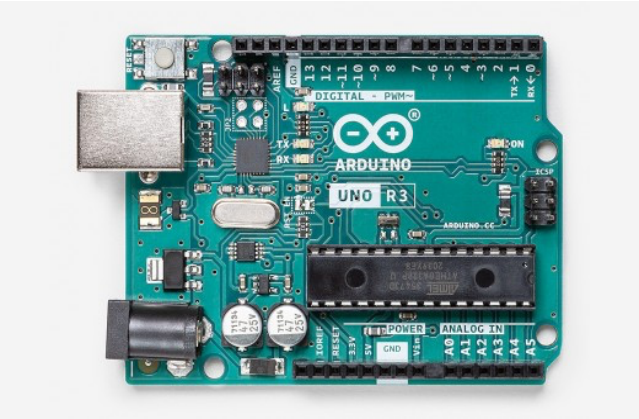



*Ilustración 1. Arduino UNO Rev.3*

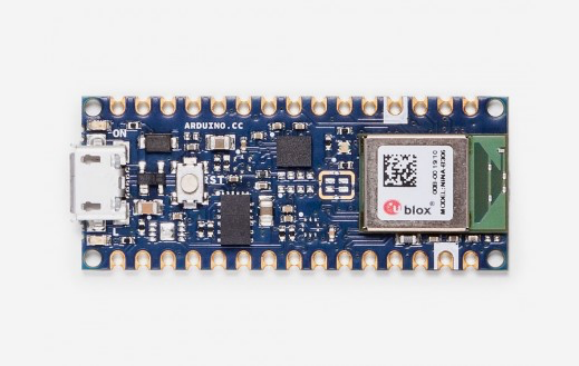



*Ilustración 2.Arduino Nano 33 BLE.*

*Ilustración 3. Arduino UNO Wifi Rev.2*

A continuación, mostramos una tabla comparativa con las diferencias principales entre estos modelos y otras propuestas más económicas de otras marcas:

*Tabla 1. Comparativa de diferentes PCB's basadas en Arduino.*

| Especificaciones | Arduino UNO (Rev.3) | Arduino Nano 33 BLE | Arduino UNO wifi Rev.2 | Elegoo UNO R3 | AZDelivery Modulo D1 R1 WiFi ESP8266MOD 12-F |
|---|---|---|---|---|---|
| Microcontrolador | ATmega 328p | nRF52840 | ATmega 4809 | ATmega 328p | ESP-8266EX. |
| Voltaje | 5V | 3.3V | 5v | 5V | 3.3V |
| Digital (I/O) Pins | 14 | 14 | 14 | 14 | 11 |
| PWM Digital (I/O) Pins | 6 | 14 | 5 | 6 | 11 |
| Input Pins Analógicos | 6 | 8 | 6 | 8 | 1 |
| Memoria Flash | 32 KB | 1MB | 48KB | 32KB | 4MB |
| SRAM | 2KB | 256Kb | 6,144 Bytes | 2KB | 50KB |
| EEPROM | 1 KB | NO | 256 Bytes | 1KB | Externa |
| Clock Speed | 16 MHz | 64 MHz | 16 MHz | 16Mhz | 80/160MHz |
| Bluetooth | NO | Low Energy | Low Energy | NO | NO |
| WiFi | NO | NO | SÍ | NO | SÍ |
| Precio | 19€ | 17’50€ | 38’90€ | 9’99€ | 8’82€ |

## 2.2 Arduino IDE

El entorno de desarrollo integrado de Arduino se caracteriza principalmente por permitir editar y crear diferentes programas o *Sketches*, como se denominan en el entorno, escritos en el propio lenguaje de la marca en el que se almacenan todas las funciones e instrucciones a realizar por nuestra placa Arduino. El mismo editor es capaz de compilar y subir posteriormente nuestros programas a la placa Arduino.

Posee un gestor de bibliotecas para poder interactuar con diferentes periféricos o manipular de forma especial los datos. Además, dichas bibliotecas ofertan programas de ejemplo para poder comprender y probar las funcionalidades de nuestra placa o periféricos.

El entorno es compatible con todas las placas de la marca e incluso con modelos de otros fabricantes, pero para poder utilizar ciertas herramientas hemos de indicar el modelo. A esta funcionalidad se le añade la de poder elegir el canal de

comunicación entre la placa y el PC mediante conexión USB o Bluetooth (solo para modelos compatibles).

Por último, el Monitor Serial es otro elemento destacable en el entorno, el cual nos permite interactuar con la placa en tiempo de ejecución, recibiendo y enviando instrucciones por la línea Serial.

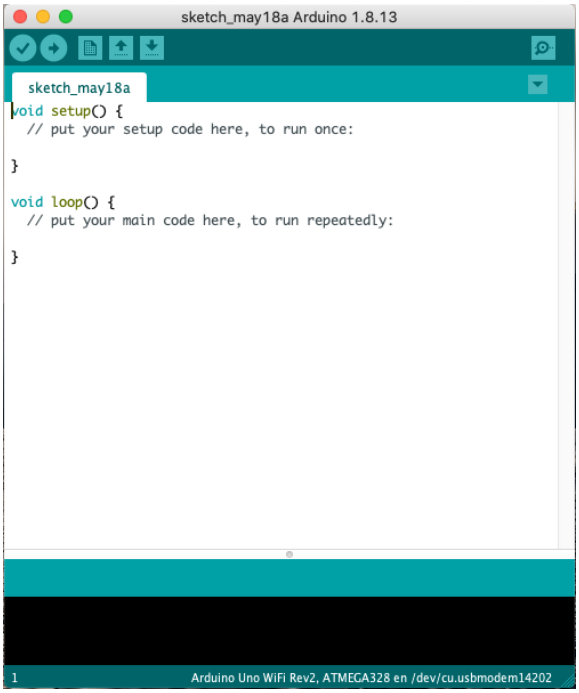



*Ilustración 4. Arduino IDE*

## 2.3 Origen de las placas programables Arduino

En el año 2003, un estudiante del *Integration Design Institute Ivrea (IDII)* en Milán, Hernando Barragán, presenta como proyecto para su tesis un entorno de programación junto a una placa PCB enfocada a la creación de proyectos para personas sin conocimientos avanzados sobre electrónica. Este proyecto se denominaba *Wiring* y planteaba, además, la solución a la problemática del encarecimiento de las placas programables en el ámbito académico.

Al proyecto que fue supervisado por Massimo Banzi y Casey Reass, se le unieron posteriormente David Mellis y David Cuartielles, en el año 2005, para abaratar aún más el coste de la placa cambiando el controlador y adaptando el entorno de desarrollo rebautizándolo así, con su actual nombre al proyecto; *Arduino*. El nombre del proyecto fue tomado en honor al nombre del bar donde los fundadores se reunían de forma frecuente. Hernando Barragán no continuó dentro del proyecto tras ser renombrado (4).

Actualmente Arduino se considera una marca que se caracteriza por la venta de sus placas de precio muy asequible, con infinidad de accesorios orientados al aprendizaje y la introducción a la programación y a la electrónica, con posibilidad de ser programadas en un entorno de desarrollo para todas las plataformas de S.O para computador y cuya licencia es de código abierto (4).

## 2.4 Uso de la tecnología Arduino actualmente en el sector de la seguridad

Dada la versatilidad y el alcance de esta tecnología, su uso en el ámbito de la seguridad está principalmente fundamentado en la implementación de proyectos hechos a mano desarrollados por usuarios no profesionales y sin ninguna finalidad lucrativa, comúnmente con un objetivo académico o lúdico.

## 2.5 Concepto de lector de huella digital

Un lector de huella digital se define, según la página web *Ayuda ley protección dato* (5), como un tipo de tecnología biométrica que se ocupa de identificar y registrar mediante combinación de técnicas software y hardware las huellas dactilares de diferentes individuos con tal de proporcionarles acceso a un sistema informático o una estancia física.

El funcionamiento de dicha tecnología se basa en dos procesos. El primero es la inscripción de la huella y consiste en decodificar y almacenar la imagen de su huella en una base de datos segura, para su futura verificación. El segundo es la verificación de las huellas almacenadas mediante un escaneo de la huella expuesta y comparación con la imagen previamente almacenada.

Los fundamentos de la inscripción de una huella se basan en capturar imágenes de las crestas y las líneas que componen la huella de un mismo dedo. Debido a ello, es muy complicado que dos sujetos presenten el mismo patrón.

Según el fabricante o la necesidad del sistema donde se va a implantar, se empleará un determinado algoritmo con tal de agilizar el reconocimiento. Las alternativas de algoritmos más comunes consisten en almacenar el final de una línea o cresta o su bifurcación. Estos puntos se denominan minucias.

La ventaja de almacenar dichas minucias reside en encontrar coincidencias con estas cada vez que se intenta identificar una huella, reduciendo así el tiempo de procesamiento considerablemente, y pudiendo acertar de manera exitosa si el dedo o el lector se encuentra ligeramente manchados, o sin necesidad de colocar en su totalidad la yema del dedo sobre el lector (5).

Existen diferentes tipos de lector de huella, los cuales se diferencian según la tecnología que empleen a la hora de almacenar los datos de nuestras huellas.

El más tradicional es el escáner óptico basado en la obtención de imágenes en dos dimensiones de la huella y, como hemos explicado anteriormente, capturando en detalle las minucias de la huella a almacenar. Actualmente no son los más empleados debido a que resultan demasiado voluminosos para ser implementados en tecnología móvil o dispositivos compactos. En términos de seguridad puede variar la precisión de lectura según la calidad del escáner e incluso en algunos casos pueden ser franqueados empleando modelos en dos dimensiones o imágenes de alta calidad (5) .

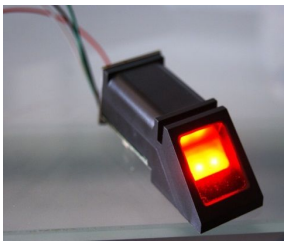


*Ilustración 5. Lector de huellas óptico*

Los lectores capacitivos adquieren más valor en términos de seguridad ya que la forma de capturar los detalles de la huella se realiza mediante la obtención de la señal eléctrica que emiten. Gracias a una serie de circuitos condensadores son capaces de almacenar la carga única que puede transmitir una huella al ser posicionada sobre una placa capacitiva (5).

*Ilustración 6. Lector de huellas capacitivo.*

Por último, el modelo de lector de huella ultrasónico. El mismo sensor es capaz de capturar y analizar una imagen tridimensional de la huella en cuestión y su funcionamiento se basa en analizar la frecuencia reflejada al emplear ultrasonidos de alta frecuencia sobre una huella que descanse sobre el mismo. Las principales ventajas de emplear este tipo de sensores es su gran fiabilidad y su fácil ubicación, llegando a ser posible colocarlo debajo de la pantalla de un Smartphone. En contrapartida cabe resaltar que la velocidad a la hora de verificar una huella es menor que con las otras dos alternativas citadas anteriormente (5).

*Ilustración 7. Lector de huellas ultrasónico.*

## 2.6 Origen y evolución de la identificación por huella dactilar

Desde el siglo XIV ya tenemos constancia, gracias a los escritos del explorador Joao de Barros, de que los comerciantes chinos estampaban impresiones de las huellas y las palmas de los niños jóvenes para poder distinguirlos.

No fue hasta el 1883 que el estudio de la biometría llegó a Europa de la mano del jefe del departamento de fotografía policial de París, Alphonse Bertillon, el cual comenzó a registrar rasgos de los criminales como medidas de las extremidades o marcas en el cuerpo para poder identificarlos posteriormente. Este método se fue expandiendo a lo largo de occidente, pero debido a diferencias en los métodos de

medida y cambio de unidades comenzaron a existir problemas. Esta problemática impulso al uso del registro de la huella dactilar como medida identificativa.

En 1891, Juan Vucetich, policía de la provincia de Buenos Aires, desarrolla las primeras fichas identificativas basadas en la huella dactilar. Además, su sistema identificativo se fundamentaba en la clasificación de las huellas en cuatro grandes grupos, que posteriormente se simplificaría en identificar cuatro rasgos principales: arcos, presillas internas, presillas externas y verticilos. Posteriormente en 1905, el sistema fue adoptado por la Policía de Buenos Aires y años más tarde, en 1907 fue reconocido por la academia de Ciencias de París como el método de reconocimiento de personas más preciso hasta la fecha (6).

En este punto, era una tarea muy engorrosa poder cotejar las huellas una a una manualmente y no fue hasta la década de 1980, con la expansión de los computadores, cuando la Agencia Nacional de la Policía Japonesa empezó a utilizar sistemas de Automatización de Huellas Dactilares (AFIS). Posteriormente, en la década de los 90, fue empleado por la División de Servicios de Información de Justicia Penal del FBI de los Estados Unidos, el cual ya permitía cotejar huellas de detenidos pertenecientes a diferentes estados (7).

Actualmente podemos encontrar diferentes propuestas de sistemas de control de acceso a diferentes recursos mediante reconocimiento de huella digital.

En el trabajo de Mahadik, del año 2009, el objetivo es diseñar un sistema biométrico basado en huellas dactilares que sea capaz de lograr una "identificación personal positiva" totalmente automática con un alto nivel de confianza. Para ellos identificaron y exploraron los siguientes problemas: (i) extracción de características (ii) mejora de la imagen, (iii) coincidencia de minucias. Los autores obtuvieron buenos resultados (~92% de precisión) utilizando la base de datos de huellas dactilares abierta (FVC2002). (8)

Por otra parte, Mitall, en el año 2015, ha propuesto dos aplicaciones de la biometría de huellas dactilares: un sistema de control de acceso (ACS) para el acceso de puertas específicas para personas, utilizando un dispositivo de huellas dactilares, y un sistema de gestión de asistencia en el aula (CAMS) que utiliza la huella digital como característica biométrica para la asistencia en el aula. Se espera que ambos sistemas mitiguen las deficiencias de los sistemas alternativos existentes y eliminen las posibilidades de suplantación de identidad o proxy. Estos sistemas almacenan huellas dactilares junto con el sello de fecha / hora de cada usuario. Las huellas dactilares se almacenan dinámicamente en una base de datos para calcular las diferentes estadísticas, por ejemplo, tendencias mensuales o semestrales en el caso de CAMS. El CAMS también puede proporcionar una solución al problema de las llegadas tardías. Se realizan experimentos para medir la precisión del reconocimiento, es decir, la coincidencia de huellas dactilares. Los resultados iniciales de precisión de reconocimiento al 87% para ACS y al 92% para CAMS. (9)

Más recientemente, en el 2017, Geralde ha propuesto una solución para modificar la forma convencional de abrir las aulas y laboratorios del edificio de Ingeniería del Colegio de San Juan de Letran, la cual tiene como objetivo disminuir el tiempo perdido esperando al personal a cargo en la apertura la habitación y la permanencia de las ofertas en una determinada clase. Para ello construyeron un prototipo enfocado a automatizar el acceso a la sala y el control de los servicios públicos como las luces y las unidades de aire acondicionado. las principales claves de acceso para la automatización fueron un sistema biométrico basado en huella dactilar, con un código de seguridad. El registro de asistencia y la información de

acceso se almacenan en una tarjeta SD. El proyecto de diseñó con un sistema de acceso biométrico seguro, y también permitió monitorizar la asistencia de los profesores por clase. (10)

## 2.7 Uso de los lectores de huellas digitales actualmente en el sector de la seguridad

Actualmente los lectores de huellas se han popularizado como medio de autentificación personal, y no tan orientado a un entorno policial o fiscal. Los usos más comunes de los lectores de huellas en el ámbito de la seguridad actualmente son:

- Desbloqueo de Smartphone, tabletas u ordenadores.
- Asistentes inteligentes.
- Sensores de huella dactilar para control de asistencia y acceso a determinados lugares (hoteles, centros de trabajo, etc.).
- Medios de pago.
- Identificación en los controles de fronteras.
- Procesos de firma digital.

## 2.8 Comparativa de diferentes soluciones

A continuación, compararemos diferentes proyectos comerciales con la propuesta expuesta en este trabajo.

### 2.8.1 Ekey home

Ekey es una compañía de origen austríaco, con más de 20 años de antigüedad en el sector de la biometría, cuyo principal producto es el suministro, instalación y mantenimiento de accesos basados en lectores de huella dactilar a viviendas y oficinas. Cuentan con diferentes paquetes según el entorno de instalación y el número de estancias que queramos controlar. Equiparándolo con la propuesta que este proyecto plantea en cuanto a funcionalidad lo compararemos con el modelo *Ekey home* (11).

*Ilustración 8. Ekey home, modelo integrado en pared.*

Las principales características que presenta el producto son:

- Se pueden controlar hasta 3 estancias con un solo escáner de dedo.
- Se pueden almacenar hasta 99 dedos
- Se pueden controlar de 1 a 3 funciones (por ejemplo, la puerta, el portón y el sistema de alarma)

- Manejo sencillo y administración central de usuarios directamente a través de la unidad de controla través de la aplicación ekey home (ekey home finger scanner integra Bluetooth o ekey finger scanner arte)
- Opcional: Acceso con tarjeta (RFID) posible (adicionalmente se pueden almacenar hasta 99 tarjetas).
-

*Ilustración 9. Diferentes modelos de marco de lector Ekey home.*

### 2.8.2 Kimaldi Flexy

Kimaldi es una empresa dedicada a la fabricación y distribución de hardware de alta calidad para la identificación automática. Poseen un centro de fabricación en Barcelona y tienen veinte años de experiencia en el sector. Por último, cabe destacar que ofrecen servicio de postventa y asistencia técnica ofrecida por el fabricante oficial.

*Ilustración 10. Kimaldi Flexy*

Las principales características de su sistema de control de acceso por medio de autentificación por huella dactilar Kimaldi Flexy Offline son: (12)

- Funcionamiento offline
- Soporta tarjeta RFID propia de la marca con almacenamiento (TOC – Template On Card).
- Lista blanca y lista negra de tarjetas perdidas.
- Protección IP65 contra polvo y agua
- Conecta todos los dispositivos de la marca a una misma red.
- Caja y Firmware personalizable
- Almacena los últimos 5.000 eventos (descarga de eventos mediante PC).
- Número máximo de usuarios solo con Biometría (1:N): 5.000 usuarios (2 huellas por usuario).
- Incorpora Dos puertos RS-232 (Conectividad puerto Ethernet).
- Incorpora un buzzer y led tricolor.

*Ilustración 11. Componentes internos Kimaldi Flexy*

### 2.8.3 Análisis comparativo

Seguidamente podemos añadir que las propuestas observadas presentan un grado de madurez del producto mayor en el que destacan el aspecto estético, el cual es configurable en ambos productos citados y están diseñados para ser instalados en exteriores (Resistencia IP65), mientras que nuestra propuesta está diseñada para interiores y no cuenta con carcasa protectora.

En cuanto a la configuración solo el primer producto es configurable mediante dispositivo móvil; en el segundo hemos de seleccionar una versión específica para poder adquirir esta funcionalidad. Cabe destacar que nuestra propuesta sí es configurable mediante una aplicación Android por medio de tecnología Bluetooth.

Con respecto al número de sistemas que es capaz de dar acceso, el primer producto es capaz de controlar hasta tres sistemas de acceso distintos como puertas, garajes o alarmas con un solo dispositivo, mientras que el segundo es capaz de conectar distintos dispositivos por red, pero es necesario un lector en cada acceso y nuestra propuesta solo proporciona acceso a una única estancia.

En términos de almacenamiento, el primer producto es capaz de almacenar noventa y nueve huellas distintas, el segundo hasta mil y nuestra propuesta un máximo de ciento veintisiete registros, ofrecidos por el lector implementado. Además, las tres propuestas son capaces de guardar el registro de accesos, pero el segundo producto solo hasta cinco mil accesos, y nuestra propuesta está limitada por la memoria de la placa Arduino (256KB).

Nuestra propuesta no proporciona acceso mediante lector de tarjetas RFID, mientras que los otros dos productos sí.

Para concluir la comparativa, la ventaja principal de nuestra propuesta reside en la modularidad del proyecto, siendo posible remplazar cualquier componente de manera autónoma y sencilla o incluso pudiendo ser actualizada con el paso del tiempo en caso de quedar obsoleta. Cabe destacar que nuestra propuesta es hasta cinco veces más económica que el primer producto, cuyo precio parte de los 500€ en su modelo más básico.

# 3. Especificaciones del sistema

En este capitulo se expondrá en profundidad el modo de interacción entre el usuario y el sistema. Asimismo, se definirán los requisitos que ha de cumplir el sistema y la aplicación móvil y se detallarán las funcionalidades que deberá proporcionar la propuesta realizada.

Nuestro sistema tiene como misión principal controlar el acceso en una estancia privada mediante el reconocimiento y verificación de la huella dactilar. Las funcionalidades del sistema se controlarán mediante una aplicación previamente instalada en nuestro dispositivo móvil, y por ello hemos de contar con el uso de la tecnología Bluetooth tanto en el sistema de seguridad como en el dispositivo móvil en el que esté instalada nuestra aplicación

El lector de huellas debe de estar ubicado junto a la entrada de la estancia, pero todos los demás componentes deberían estar en el interior de la habitación como medida de seguridad y con el fin de mantener la integridad del sistema. Cabe resaltar que el lector RFID es interesante ubicarlo al alcance de la persona encargada de la tarjeta o llavero con tal de poder desactivar la alarma y, según el entorno donde se instale nuestro sistema podrá variar su ubicación. Por último, dado que necesitamos que los datos de las huellas que registremos en el sistema y los accesos a la estancia persistan en el tiempo, es imprescindible instalar nuestro sistema junto a una fuente de alimentación dado que sin ella nunca funcionará.

Presentadas las siguientes especificaciones podremos concluir en que el sistema debe cumplir los siguientes requisitos:

## 3.1 Requisitos

En cuanto a los **requisitos funcionales** que el sistema debería cumplir podemos encontrar:

- El usuario será capaz de entrar en la estancia únicamente verificándose mediante huella dactilar.
- El usuario solo podrá controlar el sistema mediante su dispositivo móvil siempre y cuando haya vinculado la placa con su dispositivo mediante los ajustes Bluetooth.
- Si el usuario no se encuentra registrado y tiene acceso a la aplicación móvil vinculada al sistema, podrá registrar su huella tanto en el lector del sistema como en la aplicación móvil vinculada.
- El usuario podrá editar o eliminar cualquier huella registrada siempre y cuando tenga acceso a la aplicación móvil vinculada al sistema.
- Si el usuario posee acceso a la aplicación móvil vinculada al sistema podrá acceder al último registro de accesos.
- Si el usuario no está registrado e intenta acceder mediante su huella dactilar tres veces seguidas o más la alarma se activará.
- El usuario que posea la tarjeta o llavero vinculada al lector RFID del sistema es el único capaz de desactivar la alarma

**Los requisitos no funcionales** que el sistema deberá cumplir serán:

- La lógica del sistema estará implementada en lenguaje Arduino y sobre una placa Arduino o compatible.
- El dispositivo móvil vinculado ha de tener obligatoriamente sistema operativo Android.
- La comunicación entre el sistema y el dispositivo móvil será mediante tecnología Bluetooth.
- Al registrar una huella en el sistema, los datos de la imagen de la huella y su id se almacenarán en el lector, mientras que en el dispositivo móvil sólo se almacenarán su id y el nombre elegido por el usuario, todo simultáneamente y de forma transparente para el usuario.
- Al eliminar una huella se eliminará tanto de la memoria del lector como de la base de datos del sistema.

## 3.2 Casos de uso

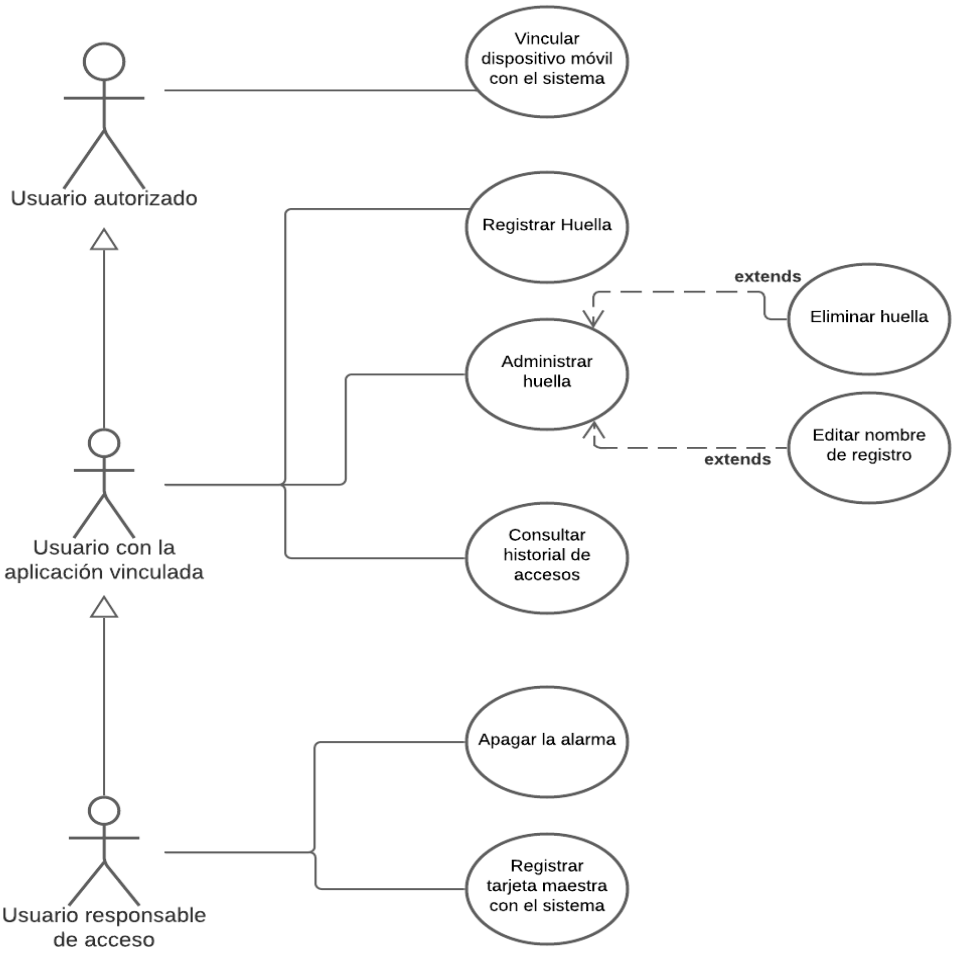



*Ilustración 12. Diagrama de casos de uso.*

*Tabla 2. Caso de uso de vincular dispositivo móvil con el sistema*

| **Caso de uso** | **Vincular dispositivo móvil con el sistema.** |
|---|---|
| **Actor** | Usuario Autorizado. |
| **Descripción** | El usuario vincula desde los ajustes de conexión Bluetooth de su dispositivo Android el módulo que ofrece la conexión al sistema. |
| **Pre-condición** | -El usuario está autorizado para llevar a cabo esta operación.<br>-El usuario ha de contar con un dispositivo móvil Bluetooth con el sistema operativo Android.<br>-El usuario conoce previamente el código PIN vinculado al módulo Bluetooth del sistema.<br>-El sistema debe de estar operativo para que el usuario pueda poder vincularse con él. |
| **Flujo general** | -El usuario abre los ajustes Bluetooth de su dispositivo móvil.<br>-Selecciona el dispositivo *FingerPass*.<br>-Introduce el código PIN asociado al dispositivo. |
| **Post-condición** | El dispositivo móvil queda vinculado al módulo Bluetooth |

*Tabla 3. Caso de uso de registrar huella.*

| **Caso de uso** | **Registrar Huella.** |
|---|---|
| **Actor** | Usuario con la aplicación vinculada. |
| **Descripción** | El usuario registra una huella de la cual ha de proporcionar un nombre y un ID. |
| **Pre-condición** | -El usuario ha de tener instalada la aplicación *FingerPass* en un dispositivo Android.<br>-El usuario ha de tener vinculado su dispositivo móvil con el sistema. |
| **Flujo general** | -El usuario abre la aplicación.<br>-Si no tiene activada la conexión Bluetooth del dispositivo, deberá activarla pulsando sobre la opción *conectar Bluetooth* para poder avanzar.<br>-El usuario selecciona de la lista de dispositivos vinculados el dispositivo *FingerPass*.<br>-El usuario selecciona la opción de *Registrar Huella*.<br>-El usuario introduce en el campo correspondiente el nombre y el ID del registro.<br>-El usuario pulsa el botón de guardar para guardar el registro.<br>-El usuario debe seguir las instrucciones que aparecen en pantalla referentes a la obtención de imágenes mediante el lector de huellas digitales. |
| **Post-condición** | La huella del usuario queda registrada en el lector de huellas. El ID y nombre de la entrada quedan registrados en la aplicación móvil. |

*Tabla 4. Caso de uso de administrar huella*

| Caso de uso | Administrar huella. |
|---|---|
| **Actor** | Usuario con la aplicación vinculada. |
| **Descripción** | El usuario visualiza un listado con las huellas dactilares registradas en el sistema. |
| **Pre-condición** | -El usuario ha de tener instalada la aplicación *FingerPass* en un dispositivo Android.<br>-El usuario ha de tener vinculado su dispositivo móvil con el sistema. |
| **Flujo general** | -El usuario abre la aplicación.<br>-Si no tiene activada la conexión Bluetooth del dispositivo, deberá activarla pulsando sobre la opción *conectar Bluetooth* para poder avanzar.<br>-El usuario selecciona de la lista de dispositivos vinculados el dispositivo *FingerPass*.<br>-El usuario selecciona la opción de *Administrar Huellas*. |
| **Post-condición** | Se visualizará el listado de huellas registradas un nombre, un ID y la posición que ocupa la entrada en el lector. |

*Tabla 5. Caso de uso de consultar control de accesos*

| Caso de uso | Consultar historial de accesos. |
|---|---|
| **Actor** | Usuario con la aplicación vinculada. |
| **Descripción** | El usuario visualiza un listado con los usuarios que han accedido a la estancia. |
| **Pre-condición** | -El usuario ha de tener instalada la aplicación *FingerPass* en un dispositivo Android.<br>-El usuario ha de tener vinculado su dispositivo móvil con el sistema. |
| **Flujo general** | -El usuario abre la aplicación.<br>-Si no tiene activada la conexión Bluetooth del dispositivo, deberá activarla pulsando sobre la opción *conectar Bluetooth* para poder avanzar.<br>-El usuario selecciona de la lista de dispositivos vinculados el dispositivo *FingerPass*.<br>-El usuario selecciona la opción de *Consultar historial de accesos*. |
| **Post-condición** | Se visualizará el listado con los nombres de los registros que han sido empleados para acceder, así como la hora a la que se accedido y si ha habido algún intento de acceso no autorizado o se ha activado la alarma. |

*Tabla 6. Caso de uso de eliminar huella.*

| Caso de uso | Eliminar huella. |
|---|---|
| Actor | Usuario con la aplicación vinculada. |
| Descripción | El usuario elimina una huella registrada en el sistema. |
| Pre-condición | -El usuario ha de tener instalada la aplicación *FingerPass* en un dispositivo Android.<br>-El usuario ha de tener vinculado su dispositivo móvil con el sistema. |
| Flujo general | -El usuario abre la aplicación.<br>-Si no tiene activada la conexión Bluetooth del dispositivo, deberá activarla pulsando sobre la opción *conectar Bluetooth* para poder avanzar.<br>-El usuario selecciona de la lista de dispositivos vinculados el dispositivo *FingerPass*.<br>-El usuario selecciona la opción de *Administrar Huella*.<br>-El usuario selecciona un registro del listado.<br>-El usuario pulsa el botón *Eliminar*. |
| Post-condición | El registro eliminado desaparece tanto de la memoria flash del lector como de la base de datos de la aplicación. |

*Tabla 7. Caso de uso de editar nombre de registro.*

| Caso de uso | Editar nombre de registro. |
|---|---|
| Actor | Usuario con la aplicación vinculada. |
| Descripción | El usuario edita el nombre asignado a una de las huellas del sistema. |
| Pre-condición | -El usuario ha de tener instalada la aplicación *FingerPass* en un dispositivo Android.<br>-El usuario ha de tener vinculado su dispositivo móvil con el sistema. |
| Flujo general | -El usuario abre la aplicación.<br>-Si no tiene activada la conexión Bluetooth del dispositivo, deberá activarla pulsando sobre la opción *conectar Bluetooth* para poder avanzar.<br>-El usuario selecciona la opción de *Administrar Huella*.<br>-El usuario selecciona un registro del listado.<br>-El usuario edita el nombre del registro.<br>-El usuario pulsa el botón *Guardar*. |
| Post-condición | El registro queda editado y guardado en la base de datos de la aplicación únicamente. |

*Tabla 8. Caso de uso de apagar la alarma.*

| Caso de uso | Apagar la alarma. |
|---|---|
| Actor | Usuario responsable de acceso. |
| Descripción | El usuario apaga la alarma una vez ésta se activa. |
| Pre-condición | -El usuario debe ser el responsable del acceso de la estancia.<br>-Debe de haber vinculada una tarjeta maestra en el sistema.<br>-El usuario debe poseer la tarjeta maestra previamente registrada en el sistema |
| Flujo general | -El usuario coloca la tarjeta maestra sobre el lector RFID. |
| Post-condición | -La alarma se desactiva. |

*Tabla 9. Caso de uso de vincular tarjeta maestra con el sistema.*

| Caso de uso | Vincular tarjeta maestra con el sistema. |
|---|---|
| Actor | Usuario responsable de acceso. |
| Descripción | El usuario vincula una tarjeta con el sistema. |
| Pre-condición | -El usuario debe ser el responsable del acceso de la estancia.<br>-El usuario debe poseer la tarjeta maestra. |
| Flujo general | -El usuario, mediante una conexión del sistema con un PC, ejecutará el programa para registrar la tarjeta como tarjeta maestra. |
| Post-condición | -La tarjeta queda registrada como maestra en la memoria EPROM del dispositivo. |

# 4. Diseño de la solución

En este capítulo trataremos las tecnologías escogidas para la implementación de la propuesta acorde con las especificaciones citadas anteriormente. Además, se expondrán los diferentes componentes del sistema, sus esquemas de conexión y la estructura de la lógica.

Como se citó en el capítulo anterior, partimos de la base de que la tecnología para la implementación de la lógica de nuestro proyecto tendrá que emplear una placa Arduino o placa compatible con el entorno de desarrollo y lenguaje de la marca. También asumimos que la tecnología escogida para el desarrollo de la aplicación móvil será Android Studio.

## 4.1 Tecnología del sistema

En consonancia con el preámbulo anterior, comenzaremos hablando de los componentes físicos que constituyen el sistema y exponiendo diferentes ilustraciones donde se muestra la conexión del los mismos.

### 4.1.2 Componentes físicos

El primer componente para tratar será la placa programable sobre la cual correrá la lógica del sistema. Para esta propuesta dadas sus características de conectividad y número de puertos, elegimos la placa *Arduino UNO Wifi rev.2*.

Cabe destacar que a pesar de su conexión *BLE (Bluetooth Low Energy),* no será empleada en este proyecto por ser más complicada de vincular con una aplicación Android Studio que la tecnología Bluetooth normal, pero puede ser interesante para futuros proyectos o actualizaciones de esta propuesta.

Como ya se indicó en el capítulo 2, es de mención su procesador *ATmega4809*, algo más avanzado que el modelo *Arduino UNO Rev.3,* ambos fabricados por la misma empresa y con arquitectura de instrucciones *RISC*.

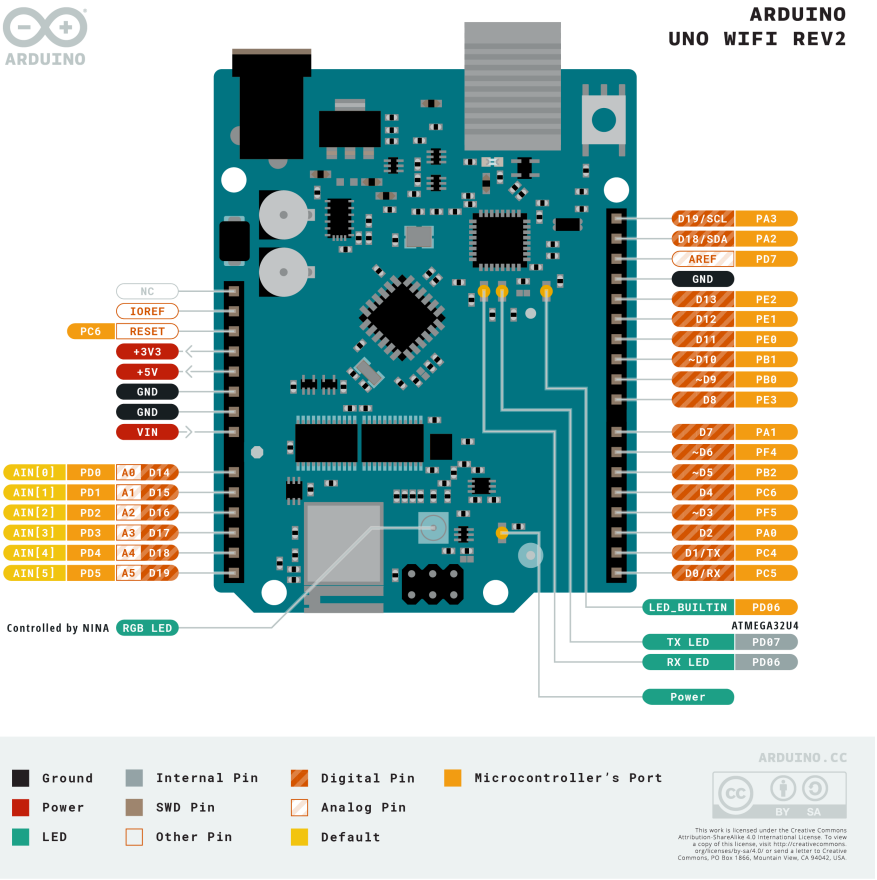



*Ilustración 13. Diagrama de conexiones de Arduino UNO Wifi Rev.2*

Otro componente indispensable para el funcionamiento del sistema será el lector de huellas dactilares. Para este proyecto se ha optado por la opción más común: un lector de huellas dactilares óptico compatible con la tecnología Arduino. En concreto estamos hablando del modelo que integra el módulo de procesamiento de huellas dactilares JM-101 con luz azul. Es capaz de almacenar ciento veintisiete registros de huella dactilar en su memoria *flash*. Por último, hemos de añadir que la conexión se realiza conectando los pines incorporados en su parte trasera con la conexión serial del Arduino (pines 0 y 1).

*Ilustración 14. Lector óptico integrado en el proyecto.*

La conexión entre la placa Arduino y el dispositivo móvil Android se llevará a cabo mediante tecnología Bluetooth, pero dado que la placa únicamente es capaz de emplear tecnología *BLE,* la comunicación se realizará mediante un módulo Bluetooth compatible con la placa, el cual lo dotará de esta tecnología. El modelo

escogido para esta propuesta será el HC_05, un módulo configurable en función del propósito al que va destinado que, además, incluye un botón físico para entrar en el modo de configuración de una manera sencilla.

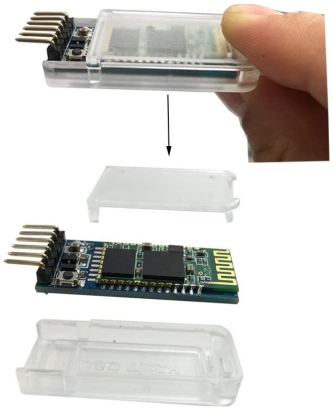


*Ilustración 15. Módulo HC-05.*

Para mantener la seguridad del sistema será imprescindible instalar un cerrojo o pestillo que mantendrá la estancia cerrada en caso querer restringir el acceso a usuarios que no estén autorizados para entrar. Dado que debe ser un dispositivo electrónico para poder ser controlado por la lógica del sistema, se ha decidido por implementar una cerradura magnética basada en un sistema de cierre activado por un solenoide. Es necesario acompañarla de una fuente de alimentación de 12 V. Es necesario resaltar que, debido a la simplicidad del proyecto y al presupuesto disponible, la fuerza de esta cerradura no es su punto más fuerte.

*Ilustración 16. Cerradura magnética.*

Para la implementación de la alarma, y para ofrecer una respuesta auditiva cuando se detecte una determinada interacción con el usuario, utilizaremos unos pequeños altavoces pasivos orientados para este tipo de proyectos. Hablaríamos del modelo KY-006, el cual permite reproducir tonos entre 1.5 kHz y 2.5 kHz. Estos dispositivos destacan por su fácil configuración y su precio económico.

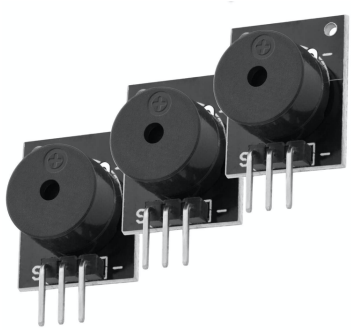


*Ilustración 17. Altavoces pasivos.*

Hemos hablado de la implementación de la alarma, pero su control se llevará a cabo por medio de un lector *RFID*, el cual será capaz de apagarla al acercar una tarjeta o llavero configurado previamente con el sistema. El lector escogido para la implementación es el modelo RC522, el cual cuenta con una frecuencia de lectura de 13,56 MHZ. La información de las tarjetas vinculadas al sistema se almacenará directamente en la memoria no volátil de la placa Arduino.

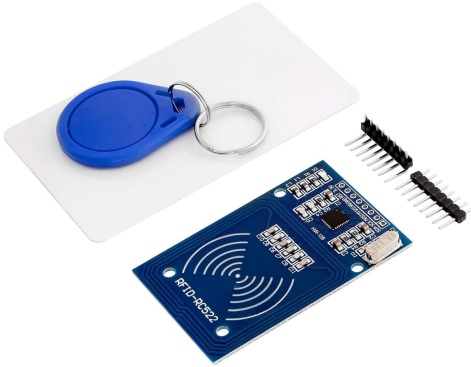



*Ilustración 18. Lector RFID utilizado.*

Por último, haremos uso de un reloj de tiempo real (RTC) para poder llevar a cabo el control de acceso de los usuarios al sistema. Dado que la placa no posee pila, en el supuesto caso en que el sistema se quedase sin suministro eléctrico, la fecha y hora del sistema se reiniciarían, teniendo así que configurarla mediante conexión a un PC, proceso que sería algo engorroso para un usuario sin conocimientos sobre la tecnología Arduino. El modelo de este reloj es el DS3231 e incorpora una pila con el fin de solventar el problema citado anteriormente.

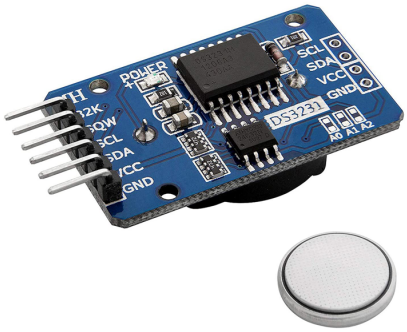



*Ilustración 19. Reloj de tiempo real utilizado.*

Los demás componentes empleados en el montaje del sistema no son demasiado relevantes para comentarlos individualmente, pero si se mencionarán posteriormente para conocer su función en el sistema.

### 4.1.3 Esquemas de conexiones

Comenzaremos mostrando un esquema general simplificado con todos los componentes instalados en el sistema. Cabe destacar que se han omitido ciertos componentes con tal de simplificar la imagen y que posteriormente se mostrarán diferentes esquemas que mostrarán las conexiones reales del sistema, pero desglosadas en diferentes módulos clasificados según la función que realicen.

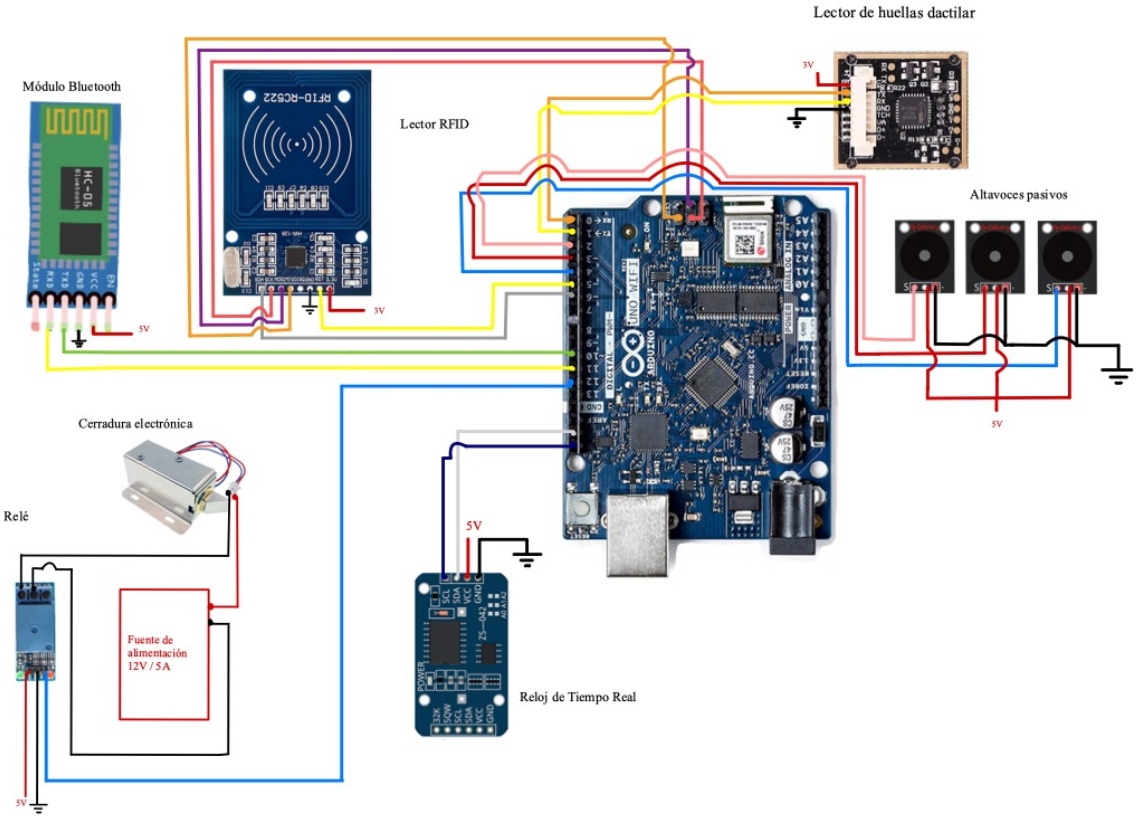



*Ilustración 20. Esquema de conexiones general.*

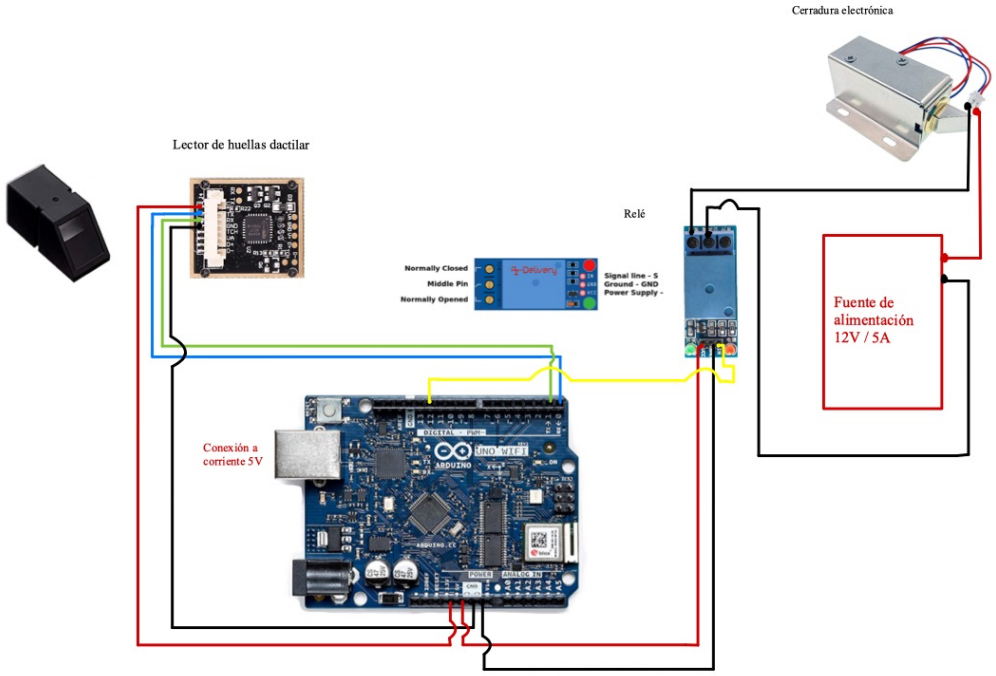



*Ilustración 21. Esquema de conexión del módulo de cerradura y huella dactilar.*

Este módulo sería suficiente para implementar un sistema de cerradura mediante autentificación con huella dactilar, pero la configuración de este se tendría que llevar a cabo mediante conexión a un PC y el entorno de desarrollo Arduino, algo complejo para usuarios no familiarizados con esta tecnología.

El funcionamiento del reconocimiento se basa en la conexión serial de los puertos de la placa Arduino con los pines del lector de huellas. En cuanto a la cerradura, está conectada a una fuente de alimentación de 12V similar a los típicos transformadores que encontramos en los cargadores de ordenadores portátiles. Al mismo tiempo, la alimentación de la cerradura depende de un relé conectado también a la placa, el cual será el encargado de permitir alimentar (o no) la cerradura en función de la señal que le llegue de la placa (si se detecta la huella como reconocida o no).

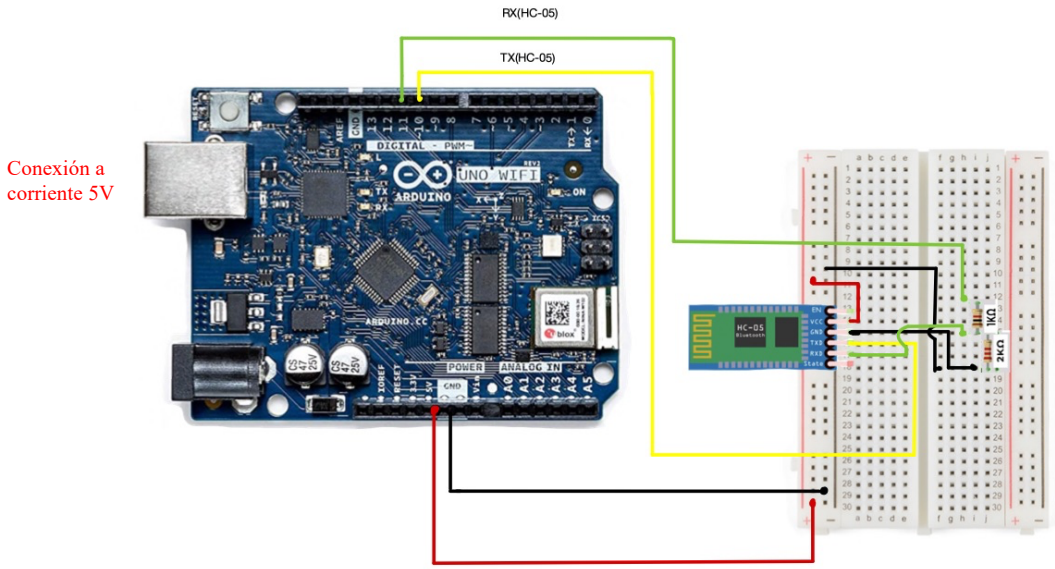



*Ilustración 22. Esquema de conexión del módulo Bluetooth.*

Este esquema muestra la conexión del módulo Bluetooth con la placa. Cabe resaltar que se ha empleado una tabla de conexiones o *breadboard para* la implementación del circuito. Esto es debido a que es recomendable por el fabricante aplicar un divisor de tensión en la recepción de datos del módulo.

Para implementar el divisor de tensión simplemente se instalarán dos resistencias de 1KΩ y 2KΩ en serie. Finalmente se conectan respectivamente las conexiones de recepción y envío de datos a la placa.

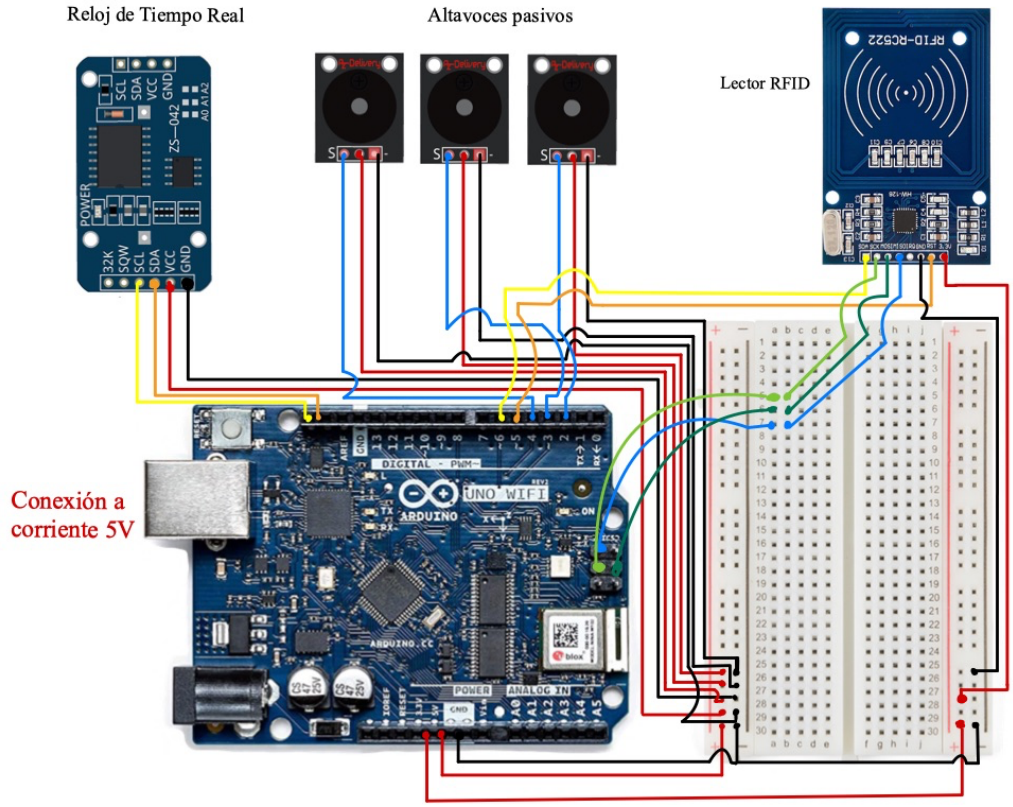



*Ilustración 23. Esquema de conexiones del módulo de alarma.*

En el siguiente esquema se representa la implementación del módulo capaz de otorgar al sistema la función de alarma y almacenar el control de accesos. Podemos observar que el sistema está compuesto por los tres altavoces pasivos, un reloj de tiempo real para poder almacenar la fecha y hora exacta y un lector de radio frecuencia para detectar la tarjeta capaz de desactivar la alarma.

## 4.2 Presupuesto del proyecto

Para finalizar el capítulo se mostrará una tabla con el coste de los componentes empleados en el sistema.

*Tabla 10. Presupuesto del proyecto.*

| Componente | Precio | Unidades | Subtotal |
|---|---|---|---|
| **Placa Arduino UNO Wifi Rev.2** | 47€ | 1 | 47€ |
| **Cable de conexión Arduino** | 6€ | 1 | 6€ |
| **Lector de huellas óptico** | 16€ | 1 | 16€ |
| **Cerradura magnética** | 10’19€ | 1 | 10’19€ |
| **Cables de conexión componentes** | 6€ | 1 | 6€ |
| **Fuente de alimentación 12V 5A** | 13’50€ | 1 | 13’50€ |
| **Tabla de conexiones** | 4’90€ | 1 | 4’90€ |
| **Módulo Bluetooth** | 10€ | 1 | 10€ |
| **Altavoces pasivos** | 2€ | 3 | 6€ |
| **Lector RFID** | 3’30€ | 1 | 3’30€ |
| **Resistencias** | 0’18€ | 2 | 0’36€ |
| **Reloj de tiempo real RTC** | 7’49 | 1 | 7’49€ |
| | | **Total :** | **130’74€** |

# 5. Desarrollo de la solución

En este capítulo detallaremos el proceso de desarrollo del proyecto de acuerdo con las especificaciones acordadas anteriormente, indicando las dificultades que han surgido, y como se han solventado.

El primer paso para el desarrollo del proyecto fue la compra de los componentes básicos del sistema de acceso, la placa, el lector de huellas dactilares, la cerradura electrónica y su conexión a la placa.

El programa instalado en la placa Arduino se ha escrito en el lenguaje propio de la marca, un lenguaje basado en C++ con características del lenguaje funcional como base.

Estos programas se caracterizan por constar de tres partes principales:

- **Implementación de librerías y declaración de variables globales**.
- **Setup**: Zona del código donde se inicializan ciertas variables necesarias para el funcionamiento posterior del programa, como pines de conexión o interfaces de comunicación.
- **Loop**: Zona del código en la cual se reproducirán repetidas veces las órdenes que aparezcan dentro de la misma.

En nuestro caso, al final de la zona del bucle definida como *loop*, se han implementado diversas funciones necesarias para el funcionamiento del sistema, las cuales se emplean en la zona de *loop*.

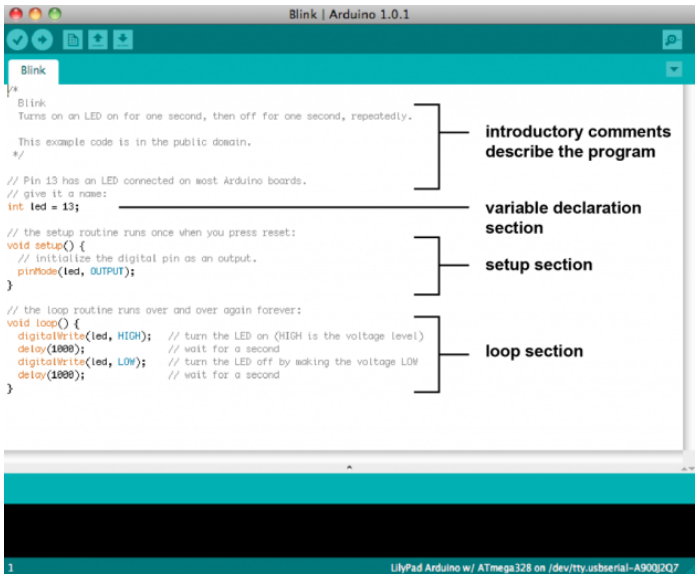



*Ilustración 24. Ejemplo de estructura de programa Arduino.*

La principal característica del programa instalado en nuestra placa, denominado de ahora en adelante como **MAIN_MENU**, es la implementación de un sistema de comandos, los cuales son reconocidos en el bucle principal (*loop*), y que será capaz de ejecutar las órdenes indicadas por el usuario mediante la

aplicación móvil *FingerPass,* que estará implementada en Android Studio. Además, MAIN_MENU nos permitirá reproducir el funcionamiento natural del sistema de manera automática, lo cual incluye la verificación de identidad mediante huellas dactilares, administración del acceso a la estancia y poder llevar el control de los accesos y eventos que se produzcan, así como la habilidad de poder activar la alarma en caso de intento de acceso no autorizado.

MAIN_MENU emplea diferentes funciones basadas en los programas ejemplo de las múltiples librerías suministradas por el entorno de desarrollo. Las más destacables son:

- **Adafruit FingerPrint Sensor Library**: Esta librería nos suministra funciones como *enroll* para poder registrar diferentes huellas, *delete* para poder eliminar registros, o *fingerprint* para poder verificar si la huella colocada sobre el lector está registrada o no.
- **MFRC522 y SPI**: Gracias a estas dos bibliotecas podemos hacer uso de las conexiones necesarias para el funcionamiento del lector de radio frecuencia (RFID).
- **Software serial**: Esta librería es necesaria para la creación del interfaz de recepción y envío de datos por Bluetooth.
- **RTC y TimeLib**: La implementación de ambas librerías nos permitirá obtener la hora actual por medio del reloj de tiempo real, así como trabajar con formatos de fecha.

Se comenzó implementando una versión donde mediante interacción con la consola del IDE de Arduino, podíamos añadir huellas dactilares al sensor haciendo uso del programa *enroll*, y mediante el programa *fingerprint* se reconocían las huellas registradas para abrir la cerradura.

Seguidamente se comenzó con el desarrollo de la aplicación Android Studio. Dado que nunca se había trabajado con este entorno ni se había desarrollado antes alguna aplicación similar, se optó por documentarse en el tema. Tras consultar fuentes diversas, ya sean tutoriales de *YouTube* como los de D2Mágicos sobre la implementación de una aplicación Android para controlar un coche teledirigido por conexión Bluetooth basado en una placa Arduino (13), el de Innova Domotics mostrando una aplicación para comunicar una aplicación Android con Arduino por medio del módulo Bluetooth (14), y leer varios capítulos de interés del libro *Desarrollo de aplicaciones para Android. Edición 2018* (15), se decidió continuar con el desarrollo, ahora ya sí con algo más de conocimiento sobre la materia.

Esta aplicación está escrita en Java, un lenguaje orientado a objetos más que conocido por los estudiantes de Ingeniería Informática de la ETSINF, dado que es con él que aprendemos a programar. Además, esta aplicación nos permitirá enviar las diferentes órdenes al sistema con el fin de configurarlo.

Para poder seguir hablando de nuestra aplicación, antes debemos hablar de Android Studio y de la estructura propia de los proyectos implementados en esta plataforma.

Android Studio es un entorno de desarrollo basado en *IntelliJ* en el cual se pueden desarrollar aplicaciones Android completas o, como se denominan dentro del mismo entorno, proyectos. Dentro de sus funcionalidades destacan la navegación entre los diferentes ficheros que componen el proyecto, el editor de archivos, el compilador de texto, la exportación del proyecto mediante *Graddle* o el

emulador de proyectos, capaz de lanzar a ejecución nuestra aplicación en un dispositivo móvil virtual.

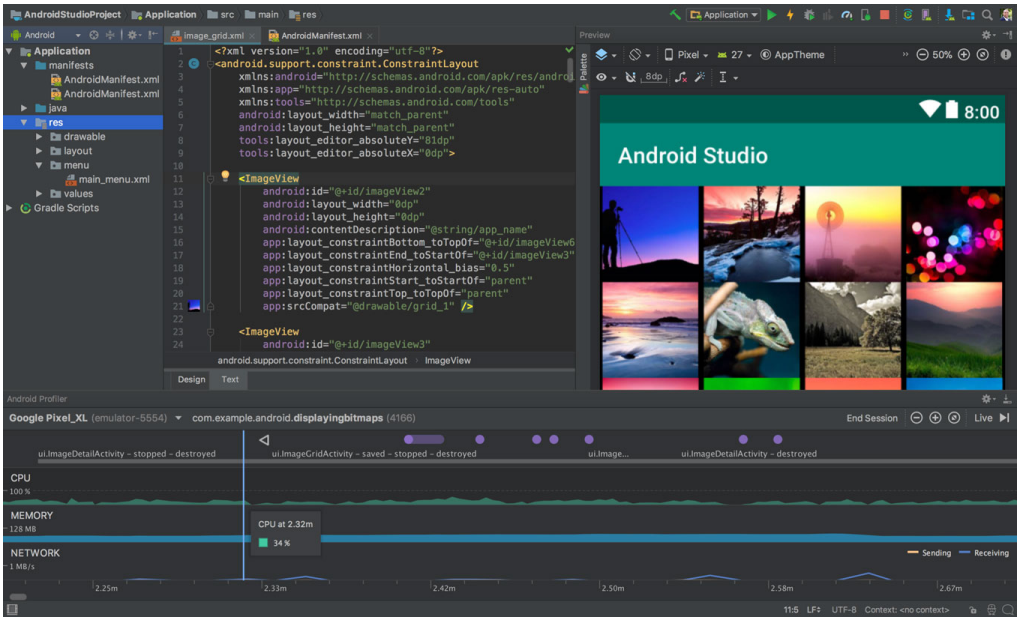



*Ilustración 25. Vista principal de un proyecto en Android Studio*

La estructura de un proyecto Android en lenguaje Java se basa principalmente en tres ficheros imprescindibles:

- **Java**: Fichero en el que se almacenan todos los archivos con las clases Java encargadas de proporcionar la lógica y funcionalidad a la aplicación.
- **Android Manifest**: Es el archivo donde se especifican ciertos aspectos como el icono o la vista principal de la aplicación, y donde se declaran todas las clases Java que formarán parte de nuestra aplicación.
- **res**: Este fichero hace referencia a *resources* o recursos de la aplicación, como pueden ser colores, textos o imágenes que vamos a emplear en ella, además del diseño de todas las vistas que estarán implementadas en archivos XML.

Por norma general, nuestras clases Java extenderán a la clase AppCompatActivity, lo cual nos lleva a hablar de los *activities* o actividades en Android Studio.

Una actividad proporciona la ventana en la que la aplicación dibuja su interfaz de usuario. Por lo general, esta ventana llena la pantalla, pero puede ser más pequeña y flotar sobre otras ventanas. Generalmente, una actividad implementa una pantalla en una aplicación. Una misma aplicación puede contener varias actividades y entre estas pueden existir diferentes dependencias como el orden en que son lanzadas. Para implementar las actividades en nuestro proyecto hemos de declararlas, como se ha especificado anteriormente en el archivo Android Manifest.XML (16).

A lo largo de su vida útil, una actividad pasa por varios estados. Para administrar las transiciones entre estados, debes usar una serie de devoluciones de llamadas (16).

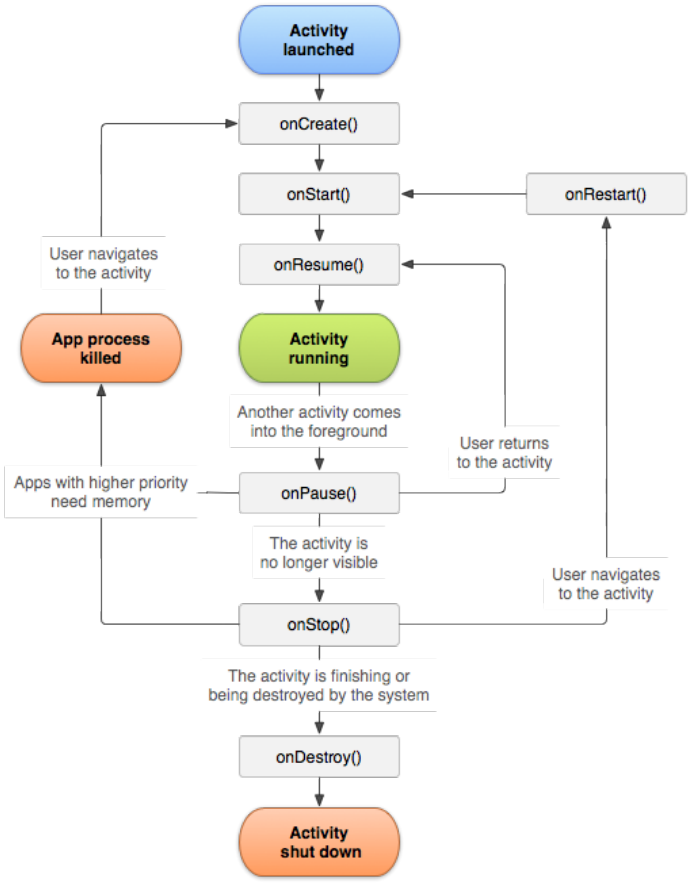



*Ilustración 26. Ciclo de vida de una actividad Android.*

A continuación, se tratarán las etapas más significativas y las llamadas que se han empleado para este proyecto:

- **onCreate()**: Es la primera etapa de vida de la actividad. En ella se inicializan los componentes elementales que posteriormente empleará la actividad. Además, se suelen adjuntar la vista de la interfaz por medio de la llamada al método setContenView().
- **onResume()**: Esta etapa se corresponde con el instante anterior a la interacción del usuario con el sistema. Es interesante incluir en esta etapa tareas como cargar los elementos de una lista o enviar un comando con tal de iniciar otra actividad.
- **onPause()**: Tras pulsar el botón de atrás o el de inicio y salir de la aplicación, la actividad entrará en esta etapa. Tras volver a la ventana vinculada con esta actividad se volverá a la etapa de onResume.

- **onDestroy()**: Cuando se elimina la actividad porque hemos cambiado a una nueva o se ha cerrado la aplicación se entra en la etapa onDestroy, que principalmente implementa métodos para la liberación de los recursos. (16)

Dicho esto, hablaremos de la estructura concreta de nuestro proyecto denominado **FingerPass**. FingerPass está formada por cuatro niveles de actividad distintos: actividad principal, menú de selección del dispositivo, menú de control y tres actividades que implementan las funciones necesarias para controlar el sistema.

La navegación a través del sistema se plantea de la siguiente manera: el usuario visualiza la ventana principal nada más abrir la aplicación. Esta le dirigirá, siempre y cuando tenga la conectividad Bluetooth activada, a la ventana de selección del dispositivo donde aparecerá, entre los dispositivos vinculados, el módulo Bluetooth conectado a la placa, siempre y cuando esté previamente vinculado con el dispositivo. Tras seleccionarlo, la aplicación mostrará el menú de control donde se mostrarán las diferentes funcionalidades que la aplicación implementa:

- Registrar huella dactilar.
- Administrar huellas.
- Consultar el control de accesos

Finalmente, según la opción que hayamos seleccionado, la aplicación nos mostrará la ventana referente a la funcionalidad seleccionada.

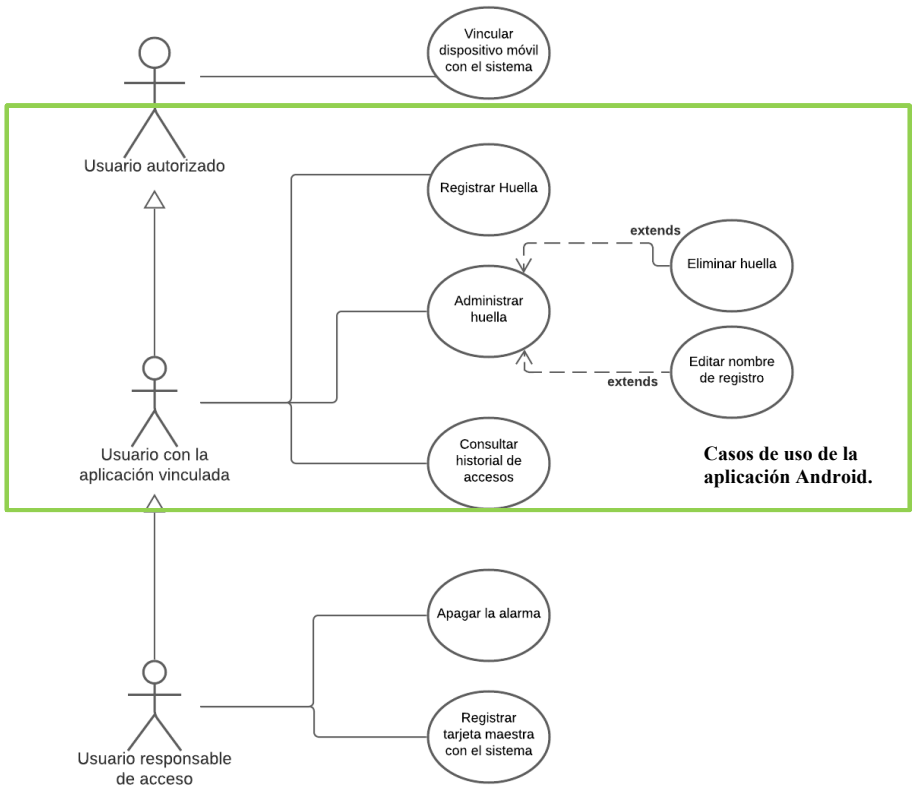



*Ilustración 27. Casos de uso de la aplicación Android Studio.*

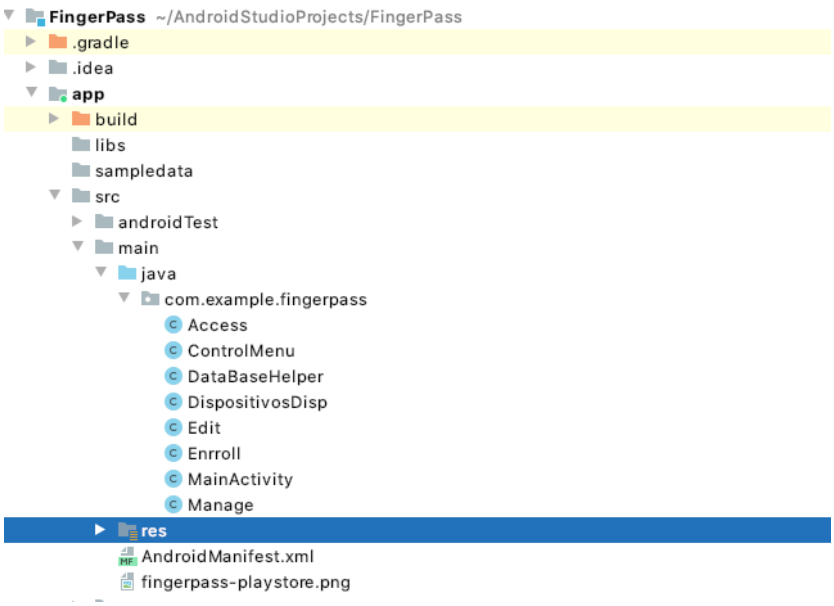



*Ilustración 28. Esquema organización de ficheros de la app Android Studio.*

Las primeras clases desarrolladas fueron *MainActivity* y *DispositivosDisp*. Respecto a la primera, implementa la lógica de la ventana principal donde se comprobará si la conexión Bluetooth está activada.

En cuanto a la segunda clase, implementa la lógica para obtener la lista de dispositivos Bluetooth vinculados y si detecta que la conexión no está activada, avisar al usuario con tal de que lo active para mostrar la lista con los dispositivos.

```
@Override
public void onResume() {
    super.onResume();

    // Comprobar la conexión del dispositivo
    comprobarConexion();

    //Adaptador de la lista de dispositivos vinculados
    ArrayAdapter<String> pairedDevicesArrayAdapter = new ArrayAdapter<>( context: this, R.layout.nombre_dispositivo);

    // Vinculamos nuestro adaptador a nuestra lista
    ListView pairedListView = findViewById(R.id.listDevice);
    pairedListView.setAdapter(pairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);


    Set<BluetoothDevice> pairedDevices = Collections.emptySet();

    try {
        //Obtenermos la lista de dispositivos BT sincronizados
        pairedDevices = btConexion.getBondedDevices();
    } catch (Exception e) {
        Toast.makeText(getBaseContext(), text: "Ha ocurrido un error al obtener la lista de dispositivos vinculados ", Toast.LENGTH_SHORT).show();
        finish();

    }
    if (pairedDevices.size() > 0) {

        //Barremos la lista de dispositivos sincronizados
        for (BluetoothDevice device : pairedDevices) {
            pairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        pairedDevicesArrayAdapter.add("No hay dispositivos disponibles");
    }
}
```

Declaración e inicialización de la lista de dispositivos vinculados.

Barrido de la lista para crear pares de nombre de dispositivo y dirección MAC.

*Ilustración 29. Lógica para la obtención de la lista de dispositivos vinculados.*

En la siguiente imagen podemos ver que inicialmente se comprueba la conexión Bluetooth por el método *comprobarConexion()*, implementado más abajo en el código, y que mediante el método *getBondedDevices()*, se obtienen todos los dispositivos vinculados. Por último, se recorre la lista y se van añadiendo elementos como pares de nombre de dispositivo/dirección MAC.

Inicialmente, se pretendía emplear la conexión Bluetooth Low Energy disponible en la placa Arduino, pero al encontrar tan pocas referencias y ejemplos sobre aplicaciones Android que empleasen esta conectividad, se optó por incorporar al sistema el módulo Bluetooth HC-05 con tal de otorgarle la tradicional conectividad Bluetooth. Una vez conectado a la placa Arduino ejecutaremos el programa de configuración del módulo y a través de la consola del IDE de Arduino por medio de los comandos AT.

Los comandos AT son un tipo de comandos para cambiar aspectos de la configuración del módulo Bluetooth, en nuestro caso hemos cambiado el nombre, el pin de seguridad para vincularlo con un dispositivo, la frecuencia de transmisión de datos y el modo de funcionamiento.

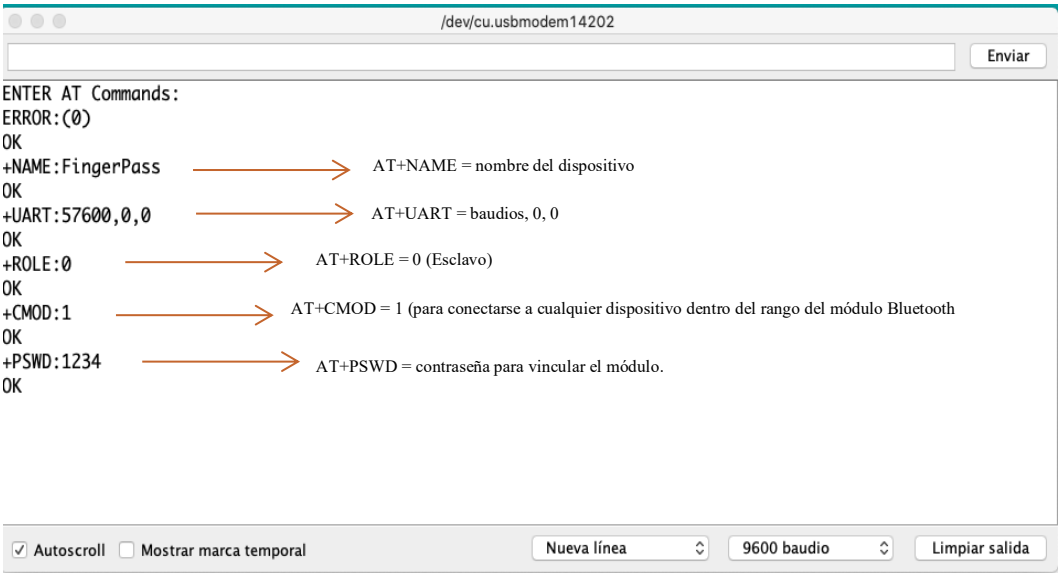



*Ilustración 30. Configuración del módulo Bluetooth.*

Volviendo sobre la aplicación FingerPass, el modelo de conexión que se ha optado por utilizar consiste en abrir una determinada conexión Bluetooth en cada actividad que haga uso de ella, y al pasar a otra, se cerrará la conexión de la actividad anterior, dejando paso a la nueva. Esto se consigue pasando entre actividad y actividad la dirección MAC del módulo Bluetooth obtenida en *DispositivosDisp*.

Una vez creada la actividad nos dispondremos a obtener el dispositivo Bluetooth a través de su dirección MAC. Por último, generaremos un *socketBluetooth* a partir de un *Identificador Único Uninversal* (UUID) aleatorio y el dispositivo obtenido anteriormente

```
private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException {
    //crea un conexion de salida segura para el dispositivo usando el servicio UUID
    return device.createRfcommSocketToServiceRecord(BTMODULEUUID);
}
```

*Ilustración 31. Declaración del método de creación del socket Bluetooth.*

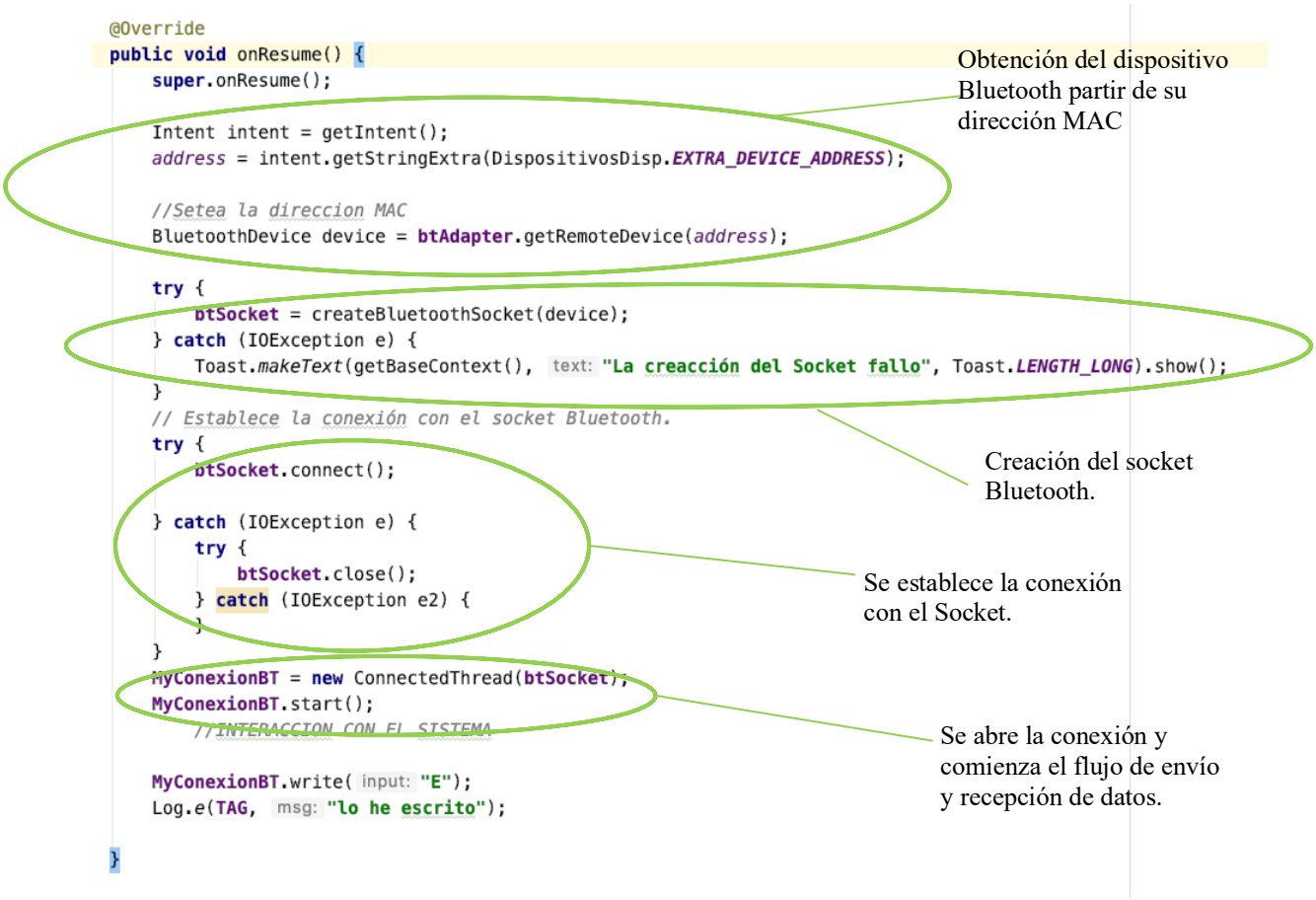
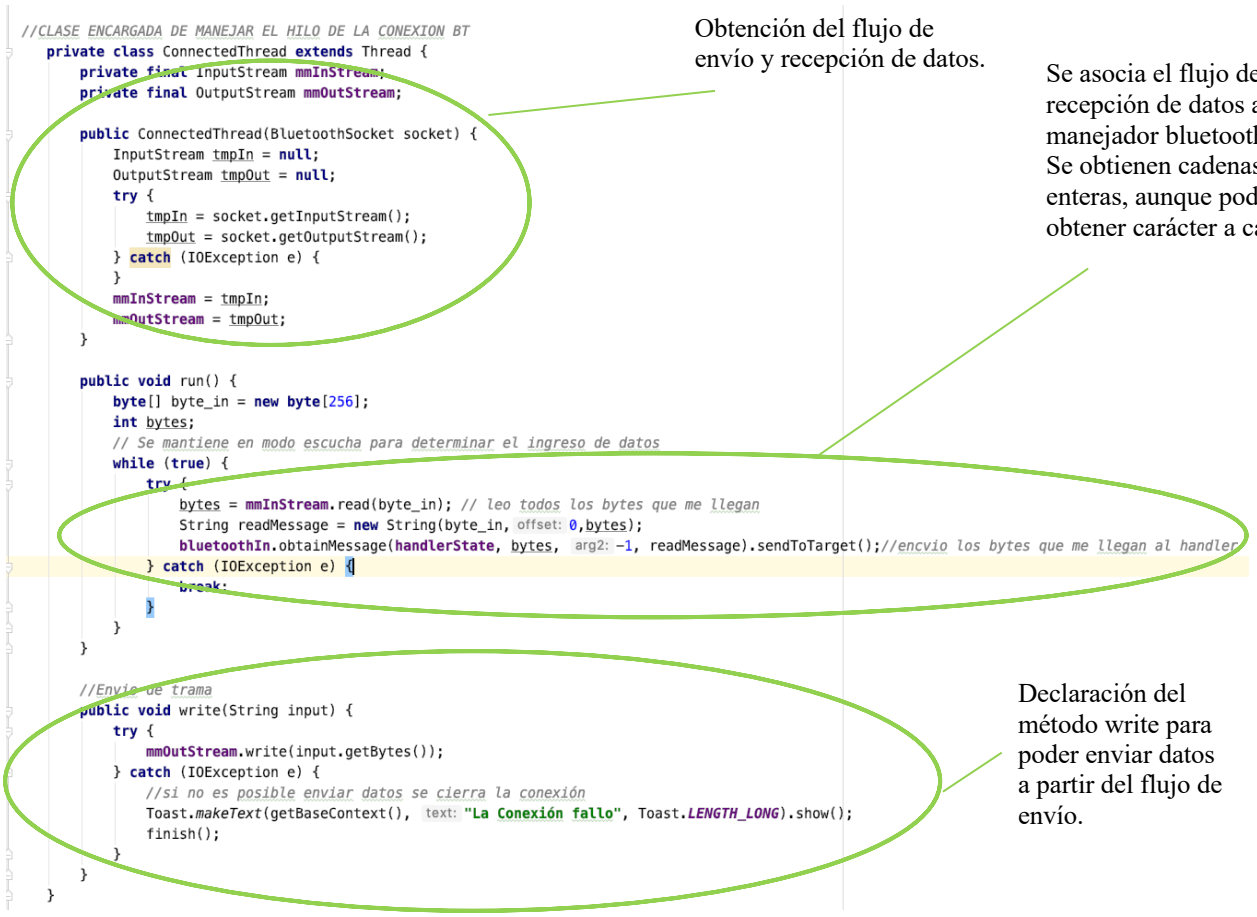



*Ilustración 32. Ejemplo de establecimiento de conexión Bluetooth.*

Se le añade a este modelo la implementación de la clase privada *ConnectedThread*, dentro de cada clase respectivamente, y haciendo uso de un manejador denominado *bluetoohIn* personalizado para cada actividad en función de que manera queramos recibir los datos y la interacción que deseamos tener con el sistema. Las actividades que emplean este modelo son *Enroll, Edit y Access*.




```
//CLASE ENCARGADA DE MANEJAR EL HILO DE LA CONEXION BT
    private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket) {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
            }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] byte_in = new byte[256];
            int bytes;
            // Se mantiene en modo escucha para determinar el ingreso de datos
            while (true) {
                try {
                    bytes = mmInStream.read(byte_in); // leo todos los bytes que me llegan
                    String readMessage = new String(byte_in, offset: 0,bytes);
                    bluetoothIn.obtainMessage(handlerState, bytes,  arg2: -1, readMessage).sendToTarget();//encvio los bytes que me llegan al handler
                } catch (IOException e) {
                    break;
                }
            }
        }

        //Envio de trama
        public void write(String input) {
            try {
                mmOutStream.write(input.getBytes());
            } catch (IOException e) {
                //si no es posible enviar datos se cierra la conexión
                Toast.makeText(getBaseContext(),  text: "La Conexión fallo", Toast.LENGTH_LONG).show();
                finish();
            }
        }
    }
```

*Ilustración 33. Clase ConnectedThread en Enroll.*

Toda línea enviada a través de nuestro sistema termina con el símbolo " ~ ".

```
bluetoothIn = (Handler) handleMessage(msg) → {
        if (msg.what == handlerState) {
            //SI ME LLEGA LA LISTA DE HUELLAS REGISTRADAS
            // SI ME LLEGA EL MENU DE REGISTRAR HUELLA
            String mensage = (String) msg.obj;
                recDataString.append(mensage);
                int endOfLineIndex = recDataString.indexOf("~");                    // se detecta el final de la línea.
                if (endOfLineIndex > 0) {
                    String dataInPrint = recDataString.substring(0, endOfLineIndex);  // se obtiene el texto recibido por el flujo de recepción.
                    Log.e(TAG, dataInPrint); // extract string
                    if(dataInPrint.equals("Z")){                                    // orden que anuncia el fin de la comunicación.
                       Intent back = new Intent( packageContext: Enroll.this, ControlMenu.class);
                       back.putExtra(EXTRA_DEVICE_ADDRESS, address);
                       startActivity(back);
                    }else{
                        Log.e(TAG, dataInPrint);
                    imprimir.setText(dataInPrint);                                  //se imprimen los datos recibidos a traves del flujo de recepción.
                    recDataString.delete(0, recDataString.length());
                    dataInPrint = " ";
                }

            }
```

*Ilustración 34. Manejador del flujo de recepción en Enroll.*

Terminaremos hablando de la comunicación Bluetooth sobre como, en el momento en que se pausa alguna de estas actividades, se cerrará la conexión del socket por dos motivos básicos: el consumo de recursos por parte de la aplicación y dejar paso a una nueva conexión.

```
@Override
public void onPause() {
    super.onPause();
    try { // Cuando se sale de la aplicación esta parte cierra el socket
        MyConexionBT.write( input: "Z");
        btSocket.close();
        Log.e(TAG,  msg: "se ha pausado");
    } catch (IOException e2) {
    }
}
```

*Ilustración 35. Cierre de la conexión al pausar la actividad.*

Para terminar de hablar sobre las especificaciones de FingerPass, es necesario mencionar la implementación de una pequeña base de datos empleando SQLite con tal de que persistan los datos de los registros almacenados.

En ella se almacenará únicamente una tabla, la cual almacenará los datos de un registro de huella en la aplicación:

- Un Identificador empleado para operaciones sobre la misma tabla con el roll de clave primaria.
- El nombre que el usuario da al registro.
- La posición que ocupará en la memoria *flash* del lector de huellas dactilares, la cual será única y no puede ser nula o igual a 0.

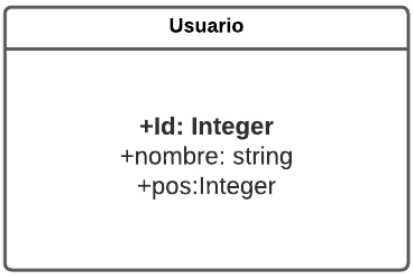



*Ilustración 36. Esquema de la base de Datos.*

Tras el desarrollo de la aplicación Android, se finalizó el programa MAIN_MENU en Arduino, sobre el cual destaca la funcionalidad de control de acceso.

El usuario solo recibirá los veinte primeros accesos que el sistema detecte tras ser reiniciado o encendido. Se llevará el registro de:

- La fecha y la hora del acceso.
- El registro que se empleado para acceder.
- Si el registro pertenece a un usuario no registrado.
- Si la alarma ha sido activada en algún momento.

```
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  //ABRIMOS LA CERRADURA
  digitalWrite(12, HIGH);
  //RESETEAMOS EL NÚMERO DE INTENTOS DE ACCESOS NO AUTORIZADOS
  fallo = 0;
  //SE REGISTRA EL ACCESO DEL USUARIO
  registrar_acceso(1, finger.fingerID);
  //SUENA EL ALTAVOZ
  aciertoBuzzer();
  //TIEMPO DE APERTURA DE LA PUERTA
  delay(3000);

  return finger.fingerID;
}
```

*Ilustración 37. Código que se ejecuta al detectar un usuario registrado.*

```
////////////////////////////////
// CONTROL DE ACCESO
////////////////////////////////
void registrar_acceso( int reg, int id) {
  if (entradas < MAX_ENTRADAS) {
    accesos[entradas][1] = rtc.now().year();
    accesos[entradas][2] = rtc.now().month();
    accesos[entradas][3] = rtc.now().day();
    accesos[entradas][4] = rtc.now().hour();
    accesos[entradas][5] = rtc.now().minute();
    accesos[entradas][6] = rtc.now().second();
    accesos[entradas][7] = reg;
    accesos[entradas][8] = id;
    entradas++;
  } else {
    Serial.println("ya no queda espacio");
  }
}
```

*Ilustración 38. Función encargada de registrar los accesos*

Como podemos observar en la función de arriba, los accesos se registran dentro de una matriz bidimensional donde cada fila corresponde a un acceso.

| Año | Mes | Día | Hora | Minutos | Segundos | Tipo registro | Id |
|---|---|---|---|---|---|---|---|
| • | | | | | | | |
| • | | | | | | | |
| • | | | | | | | |
| • | | | | | | | |

*Ilustración 39. Esquema de la matriz de control de accesos.*

```
if (DataBluetooth == 'A')
{
  for (int i = 0; i < MAX_ENTRADAS; i++) {
    // AÑO
    anyo = (String) accesos[i][1];
    //MES
    mes = (String) accesos[i][2];
    //DÍA
    dia = (String) accesos[i][3];
    //HORA
    hora = (String) accesos[i][4];
    //MINUTOS
    minutos = (String) accesos[i][5];
    //SEGUNDOS
    segundos = (String) accesos[i][6];
    //TIPO DE REGISTRO
    reg = (String) accesos[i][7];
    //ID DEL USUARIO
    usr = (String) accesos[i][8];
    String msg = anyo + '/' + mes  + '/'  + dia + ' ' +  hora +  ':' +  minutos +  ':' +  segundos  +  "~";
    // COMPROBAMOS QUE SEA UNA FILA CON UN REGISTRO
    if (accesos[i][1] != 0) {
      BT.println(msg);
      delay(2000);
      // COMPROBAMOS EL TIPO DE REGISTRO
      if (accesos[i][7] == 1) {
        BT.println( usr + "$" + "~");
        delay(2000);
      } else if (accesos[i][7] == 0) {
        BT.println("Usuario no autorizado$~");
        delay(2000);
        }else if (accesos[i][7] == 2){
          BT.println("La alarma fue activada$~");
        delay(2000);
        }
    }

  }
  BT.println("Z ~");
}
```

*Ilustración 40. Función encargada de enviar el registro de accesos al dispositivo móvil.*

Como podemos observar arriba, la función se ejecutará al recibir el comando A del dispositivo vinculado y cerrará la conexión al enviar el comando Z. Además, para enviar el control del acceso almacenado recorreremos toda la matriz, comprobando las filas que almacenen un acceso y comprobando el tipo de registro de acceso:

- Usuario no autorizado.
- Usuario registrado.
- Activación de la alarma.

Cerramos este capitulo especificando que el identificador de la tarjeta maestra, capaz desactivar la alarma incorporada en el sistema, se almacenará en la memoria principal de la placa base tras la primera configuración del sistema. Esto se debe a que, en caso de que el sistema se quede sin suministro eléctrico, el sistema recuerda cuál es la tarjeta vinculada. La tarjeta ya vendrá configurada de serie, lo cual se traduce en que el usuario no será capaz de cambiar esta configuración en esta versión del sistema debido a la especial dificultad de esta tarea.

# 6. Pruebas

En este capítulo se analizarán las características del sistema en su versión final en términos de velocidad y fiabilidad. También se tratarán ciertas imperfecciones observadas en el funcionamiento natural del dispositivo.

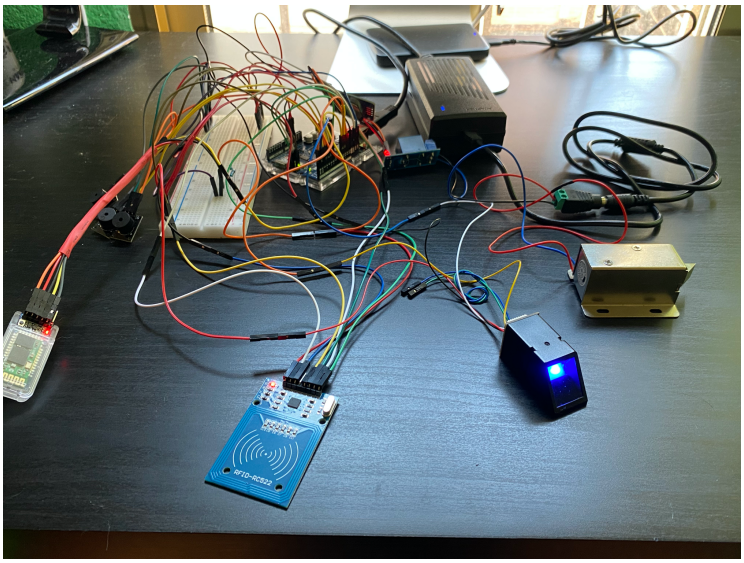


*Ilustración 41. Versión final del sistema.*

## 6.1 Pruebas de tiempo de reacción

Para validar el correcto funcionamiento de nuestra aplicación se ha considerado importante determinar los tiempos de respuesta correspondientes a diferentes eventos dentro de nuestro sistema. En particular, es especialmente importante que aquellas acciones que se van a repetir más frecuentemente, y que resultan de la interacción con los usuarios comunes, se caractericen por tiempos de respuesta bajos, a poder ser con tiempos inferiores a 1 segundo.

A continuación, se muestra una tabla con la información de los tiempos observados al llevar a cabo diferentes tareas propias del sistema. Estos tiempos se han medido en condiciones normales de uso y se ha optado por especificar el mayor de los tiempos medidos en las diferentes pruebas realizadas (peor caso), así como una media de los tiempos obtenidos durante las pruebas realizadas (cuatro veces por tarea, de manera consecutiva).

*Tabla 11. Tiempo de reacción del sistema.*

| Tarea | Tiempo máximo observado | Tiempo medio observado |
|---|---|---|
| Arranque del sistema | 2 segundos | 1, 7 segundos |
| Tiempo de apertura de la aplicación | 3,54 segundos | 3,07 segundos |
| Detección de huella registrada | 1,41 segundos | 1,05 segundos |
| Detección de una huella no registrada | 0,8 segundos | 0,67 segundos |
| Tiempo en abrir la cerradura | 1,02 segundos | 0,92 segundos |
| Tiempo entre dos lecturas simultaneas | 1'8 segundos | 1,53 segundos |
| Tiempo en activarse la alarma tras el tercer intento de acceso fallido | 0'8 segundos | 0,7 segundos |
| Tiempo en desactivar la alarma | 1'8 segundos | 1,4 segundos |
| Registrar una huella en el sistema | 33 segundos | 32 segundos |
| Eliminar una huella del sistema | 6'49 segundos | 5'72 segundos |
| Cambiar nombre a un registro | 9,30 segundos | 8 |
| Consultar el registro de accesos | 45 segundos | 45 segundos |

Se ha observado que todas las actividades relacionadas con el reconocimiento de la huella dactilar mediante el lector se ven afectadas por el estado de éste, es decir, si la superficie está más limpia la velocidad de lectura es mucho mayor, por ello es recomendable limpiarlo con cierta asiduidad.

Las funciones registrar huella, eliminar huella y consultar el control de accesos se han empezado a cronometrar desde el menú principal.

El tiempo de apertura de la cerradura ha sido configurado a tres segundos, y se puede cambiar editando el programa MAIN_MENU.

El tiempo en obtener el registro de accesos también depende del retraso entre mensajes que se ha especificado en el programa, y se ha fijado a dos segundos. También influye la cantidad de registros en el listado de accesos. El tiempo especificado en la tabla corresponde con aproximadamente la mitad de la capacidad de los accesos totales; once accesos.

En general se observa que los tiempos más críticos, que corresponden a detección de huella registrada/no registrada, se realizan en tan solo 0,67 segundos de media, lo cual para los usuarios es un tiempo más que aceptable. Igualmente, en el caso de lectura de huella registrada, la correspondiente apertura de puerta se da en 0,92 segundos de media después, lo cual también está dentro de los rangos que los usuarios considerarían normales. Destacar también que el tiempo entre dos lecturas consecutivas es de 1,53 segundos de media, lo cual es más que razonable hasta para entornos con un acceso masivo a determinado recurso físico.

## 6.2 Pruebas de seguridad contra la falsificación de huella

Se ha decidido realizar una prueba básica para medir la seguridad del sistema. Consiste en el popularmente conocido truco de poder burlar la seguridad de un lector óptico de huellas empleando un trozo de cinta adhesiva marcando la huella registrada y colocando la cinta en otro dedo no registrado.

Tras realizar diferentes pruebas con diferentes huellas registradas y diferentes tipos de cinta adhesiva, podemos concluir que el lector no reacciona al entrar en contacto con la cinta adhesiva, inclusive con el dedo encima de la cinta.

Podemos concluir finalmente que la seguridad del lector es bastante buena a pesar de su bajo coste.

## 6.3 Fallos observados durante las pruebas.

Durante las pruebas se han observado diferentes fallos tanto en el sistema como en la Aplicación Android. Comenzaremos hablando de los fallos encontrados en el sistema.

Dada la cantidad de cables conectados a la placa, muchos de ellos conectados a modo de alargadera entre un cable conectado a la placa y otro a un dispositivo o sensor, es muy difícil comprobar siempre que todas las conexiones estén en correcto estado:

- El lector de huellas a veces falla debido a posibles cables sueltos, provocando que la luz de éste sea más tenue y no identificando de manera correcta las huellas. Se ha conseguido solventar revisando las conexiones.
- La conexión Bluetooth puede fallar debido a alguna de las conexiones provocando anomalías en el diálogo entre el dispositivo móvil y el sistema. Se ha conseguido solventar revisando las conexiones
- Se ha observado que el primer fragmento de datos enviados desde la placa al sistema altera el texto enviado, cambiándolo por símbolos al

azar. Este fallo no se ha conseguido solventar a pesar de cambiar la configuración del módulo Bluetooth y la velocidad de transmisión.


*Ilustración 42. Fallo en la primera trama de datos enviada.*

En cuanto a la aplicación Android se ha observado que:

- Al entrar en la función de Administrar huella, si se hace diversas veces sobre un mismo registro, tarda en reaccionar, mostrando en primer lugar un mensaje de fallo de la conexión, y finalmente el registro a editar. La solución es ser un poco paciente, y hacer clic solo una única vez para observar el mismo registro; a pesar de ello, este fallo no compromete la funcionalidad del sistema.
- Al intentar obtener el registro de los accesos, la aplicación ha abortado y se ha cerrado. Este error está relacionado con el problema del envío de la primera trama de datos, ya que la aplicación detecta símbolos aleatorios en vez de una cadena como puede ser la fecha del acceso.

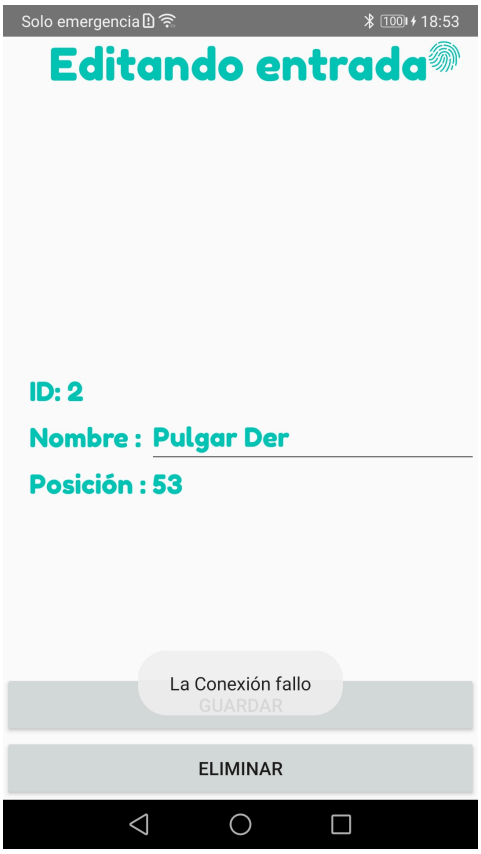



*Ilustración 43. Error al hacer clic dos veces seguidas sobre un mismo registro.*

# 7. Conclusiones y trabajos futuros

En este capítulo se tratarán las conclusiones que hemos sacado sobre los objetivos y requisitos alcanzados en este proyecto, como se relacionan los estudios cursados con la tecnología empleada en el proyecto, y, por último, como podríamos mejorar el sistema de cara a trabajos futuros.

## 7.1 Conclusiones

En cuanto a los resultados obtenidos podemos decir que se ha validado correctamente el sistema una vez observado que:

- Es posible abrir la cerradura simplemente empleando una huella dactilar registrada en el sistema.
- Es posible registrar diferentes huellas en el sistema.
- Es posible editar y eliminar los diferentes registros de las huellas registradas previamente en el sistema.
- La alarma se enciende al tercer intento fallido de acceso no autorizado.
- Es posible desactivar la alarma mediante la tarjeta previamente vinculada.

Podemos concluir en que los objetivos marcados inicialmente se han conseguido cumplir a pesar de algunos pequeños fallos.

Al principio, cuando la idea surgió no tenía claro el modo de interacción entre el usuario y el sistema. Tras investigar trabajos similares y consultarlo con mi tutor, se optó como mejor opción la de emplear una aplicación móvil que se conectase mediante Bluetooth a la placa desarrollada en Android Studio.

Lo más complicado y costoso de desarrollar para mi en este proyecto ha sido la aplicación y formarme en las bases del desarrollo con Android Studio. Sobre todo, investigar sobre la conectividad Bluetooth en Android, que en principio se iba a basar en BLE, pero, tras buscar durante varios días y no encontrar información relevante, se decidió continuar con la versión estándar de Bluetooth. Por último, es importante resaltar que cada funcionalidad del sistema (registrar, eliminar y consultar el registro de accesos) emplean el mismo modelo de conexión, pero tanto el formato de los datos recibidos como los enviados no son los mismos en ninguno de los casos, lo cual ha implicado tener que desarrollar un modelo de diálogo distinto para cada opción, lo cual ha resultado ser una tarea bastante costosa.

La parte del montaje y la selección de los componentes, a pesar de la falta de experiencia con la plataforma Arduino, así como el desarrollo del programa en sí, me ha resultado entretenido y ha sido bastante satisfactorio, ya que es el apartado más visual del proyecto. He de decir que la gran variedad de bibliotecas y programas de ejemplo disponibles a través del mismo entorno de desarrollo me han parecido de lo más interesante.

## 7.2 Relación de los estudios cursados con el proyecto

Este proyecto se encuentra bastante ligado a los conocimientos adquiridos en el grado de Ingeniería Informática ofertado por la Universidad Politécnica de Valencia. Concretamente, podemos decir que ha estado algo relacionado con la rama de Ingeniería de Computadores.

Todo el montaje y el diseño de los esquemas de conexión, así cómo la interpretación de los mismos evoca recuerdos referentes a las asignaturas de Fundamentos de Computadores y Tecnología de Computadores de primer curso donde se aprendieron los fundamentos básicos de los circuitos electrónicos y su interpretación.

La programación de la placa y análisis e interpretación de los pines del circuito impreso en la placa referencian a los conocimientos adquiridos en la asignatura de la rama de Ingeniería de Computadores, Diseño de Sistemas Digitales, donde también se programó una placa basada en tecnología FPGA, simulando diferentes circuitos como una salida de video o un procesador MIPS, aunque en un lenguaje distinto orientado a la simulación de componentes lógicos de más bajo nivel.

El programa Arduino, al estar escrito en un lenguaje similar a C, recuerda a la asignatura Fundamentos de sistemas operativos.

En cuanto a la aplicación Android, a pesar de no tener experiencia con esta tecnología, se han aprovechado todos los conocimientos sobre programación en Java como la estructura de clases, eficiencia, recursividad, herencia y uso de estructuras de datos. Estos conocimientos fueron más que asimilados en las asignaturas de Introducción a la informática y a la programación, Programación y Estructuras de datos y algoritmos.

Por último y no menos importante, a pesar de su simplicidad, la base de datos en SQLite ha sido de vital importancia en el funcionamiento del proyecto. Conocimientos como el uso de consultas o creación y definición de la base de datos adquiridos en la asignatura de Bases de Datos, han sido imprescindibles para su desarrollo.

## 7.3 Trabajos futuros

En cuanto a como se podría mejorar este proyecto, diferentes ideas han ido surgiendo a lo largo el desarrollo de este mismo, pero por falta de tiempo o extrema dificultad no se han llegado a implementar:

- **Mejora en la seguridad del sistema:** El acceso al sistema por medio de la aplicación está protegido por el pin de vinculación entre el módulo Bluetooth y el dispositivo móvil. Este mecanismo puede ser fácilmente esquivado si se accede a la placa o se sustituye el módulo, sin pensar en diferentes técnicas para la obtención del código del módulo. Por ello sería bastante más seguro establecer un sistema que

registre al usuario mediante un usuario y contraseña, lo cual nos llevaría a otra segunda posible mejora.

- **Mejora en la conectividad del sistema:** Si bien es cierto que la conectividad Bluetooth no funciona mal en distancias cortas o incluso medias, un gran avance sería conectar la placa mediante Wifi a un servicio web donde los usuarios pueden registrarse y acceder a las funcionalidades del sistema desde cualquier punto con conectividad.
- **Mejora en la precisión de detección de accesos:** Inicialmente se planteó un sistema de alarma que emplease unos sensores infrarrojos que detectasen accesos no autorizados. Si se plantea un buen protocolo de detección de accesos el sistema de alarma podría ir un paso más allá detectando intrusos incluso sin acceder por la entrada de la estancia.
- **Mejoras en el almacenamiento de accesos:** Dada las limitaciones de la memoria incluida en la placa solo podremos almacenar los veinte primeros registros de acceso que detecte el sistema tras ser reiniciado. Mediante el uso de un almacenamiento externo como una tarjeta de memoria o almacenamiento en el servicio web citado anteriormente, esta funcionalidad se podría mejorar sustancialmente en cuanto a la cantidad de accesos registrados y velocidad de obtención de los datos.
- **Adaptación del sistema a diferentes entornos:** Este sistema no está pensado para controlar el acceso a ningún tipo de estancia determinada, pero sería interesante orientarlo a:
    - Despachos en organizaciones u oficinas.
    - Cajas fuertes o de seguridad.
    - Estancias privadas como apartamentos, fincas o chalets.
    - Cajones y armarios.
    - Control de accesos en instalaciones deportivas y taquillas.

# 8. Bibliografía

1. Microcontroladores. *Wikipedia*. [En línea] [Citado el: 2 de Mayo de 2021.] https://es.m.wikipedia.org/wiki/Microcontrolador.

2. Historia. *Arduino: Tecnología para todos*. [En línea] [Citado el: 2 de Mayo de 2021.] https://arduinodhtics.weebly.com/historia.html.

3. Arduino Store. *Arduino* . [En línea] [Citado el: 2 de Mayo de 2021.]

4. Arduino. *Wikipedia*. [En línea] [Citado el: 2 de Mayo de 2021.] https://es.wikipedia.org/wiki/Arduino.

5. Sensor de Huellas Dactilar. *Ayuda ley protección datos*. [En línea] [Citado el: 2 de Mayo de 2021.] https://ayudaleyprotecciondatos.es/2020/05/18/sensor-huellas-dactilares/.

6. Historía de la biometría y la huella digital. *Maersa de México*. [En línea] [Citado el: 2 de Mayo de 2021.] https://www.maersa.com.mx/historia.html.

7. The History and Evolution of Fingerprint Identification. *North American Investigations*. [En línea] [Citado el: 2021 de Mayo de 3.] https://pvteyes.com/history-evolution-fingerprint-identification/.

8. Mahadik, Swapnali. Access Control System using fingerprint recognition. Proceedings of the International Conference on Advances in Computing, Communication and Control (ICAC3 '09), ACM, January 2009. DOI: https://doi.org/10.1145/1523103.1523166.

9. Y. Mittal, A. Varshney, P. Aggarwal, K. Matani and V. K. Mittal, "Fingerprint biometric based Access Control and Classroom Attendance Management System," 2015 Annual IEEE India Conference (INDICON), 2015, pp. 1-6, DOI: https://doi.org/10.1109/INDICON.2015.7443699.

10. D. D. Geralde, M. M. Manaloto, D. E. D. Loresca, J. D. Reynoso, E. T. Gabion and G. R. M. Geslani, "Microcontroller-based room access control system with professor attendance monitoring using fingerprint biometrics technology with backup keypad access system," IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2017, pp. 1-7, DOI: https://doi.org/10.1109/HNICEM.2017.8269432..

11. Ekey. *Home*. [En línea] [Citado el: 3 de Mayo de 2021.] https://www.ekey.net/es/.

12. Kimaldi Flexy Offline – Control de acceso por huella y tarjeta RFID. *Kimaldi*. [En línea] [Citado el: 3 de Mayo de 2021.] https://www.kimaldi.com/productos/sistemas_biometricos/kimaldi/kimaldi-flexy-offline-control-de-acceso-por-huella-y-tarjeta-rfid/.

13. D2Magicos. Android Studio-Aplicación Bluetooth. *Youtube*. [En línea] https://www.youtube.com/watch?v=RDH7S0UIoKc&list=PLTYm84ujubwJMTy8ekfLFt-3LwzFCH4M_.

14. Domotics, Innova. Android Studio 4.0 y Arduino - Comunicacion Bluetooth. *YouTube*. [En línea] Android Studio 4.0 y Arduino - Comunicacion Bluetooth.

15. Ribas Lequérica, Joan. *Desarrollo de aplicaciones para Android. Edición 2018*. s.l. : ANAYA, 2017. ISBN: 978-84-415-3892-4.

16. Itroduccion a las actividades. *Developer Android*. [En línea] [Citado el: 1 de Junio de 2021.] https://developer.android.com/guide/components/activities/intro-activities?hl=es.