

UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA
CARRERA DE ESPECIALIZACIÓN EN SISTEMAS
EMBEBIDOS



MEMORIA DEL TRABAJO FINAL

Control de acceso biométrico

Autor:
Ing. Christian Yáñez

Director:
Esp.Ing. Ernesto Gigliotti

Jurados:
Esp. Ing. Diego Fernández (FIUBA)
Esp. Ing. Danilo Zecchin (FIUBA)
Esp. Ing. Carlos Mancón (FIUBA)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre mayo de 2018 y diciembre de 2018.

Resumen

En el siguiente documento se describe el desarrollo de un sistema de control de acceso biométrico implementado mediante la utilización de Linux sobre una plataforma de desarrollo embebida. Dicho sistema cuenta con el diseño de una interfaz gráfica sobre una pantalla con tecnología touch.

El sistema otorga a los usuarios permiso de acceso, mismo que se implementa mediante un mensaje sobre la interfaz gráfica y la activación temporal de una salida digital en caso de un reconocimiento efectivo de la huella digital.

El proyecto nace como un propósito personal del autor mediante el cual se aplican varias de las herramientas aprendidas durante la carrera de especialización entre las cuales se destacan: sistemas operativos de propósito general (kernel de linux, procesos y threads, gestión de procesos, Sockets), protocolos de comunicación (periférico UART), programación de micro controladores (Programación orientada a eventos, software modular, diseño de máquinas de estado) entre las mas importantes.

Agradecimientos

A mi madre Martha, por su valioso ejemplo y apoyo incondicional.

A mi director, por su guía, consejo y ayuda.

A mis profesores y compañeros, por su colaboración y paciencia.

Índice general

Resumen	III
1. Introducción General	1
1.1. Sistemas de control de acceso biométrico	1
1.2. Motivación	1
1.3. Objetivos y alcance	2
2. Introducción Específica	5
2.1. Visión general de los elementos constitutivos	5
2.1.1. Descripción funcional	5
2.2. Requerimientos	6
2.3. Planificación	7
3. Diseño e Implementación	9
3.1. Plataforma de desarrollo	9
3.1.1. Linux embebido	10
3.1.2. Pantalla táctil element 14	10
3.2. Desarrollo de interfaz gráfica	11
3.2.1. GTK+	11
3.2.2. Glade	12
3.3. Módulo sensor para adquisición de huellas	13
3.3.1. Lectores ópticos reflexivos	13
3.3.2. Módulo lector de huella Adafruit ZFM-20	14
3.4. Gestión de base de datos (SQLite)	16
3.5. Sistema General	16
3.6. Subsistema logicService	18
3.6.1. Módulo para el manejo de la interfaz gráfica	18
3.6.2. Módulo para el manejo del sensor lector de huella	20
3.6.3. Módulo GPIO	22
3.7. Subsistema dbService	24
3.7.1. Módulo para el manejo de la base de datos	24
3.7.2. Módulo para el manejo de archivo	26
3.8. Subsistema Servicio Web	26
4. Ensayos y Resultados	29
4.1. Pruebas funcionales del hardware	29
4.1.1. Pruebas sobre la plataforma de desarrollo	29
4.1.2. Pruebas sobre el conjunto pantalla, controlador, plataforma	30
4.1.3. Pruebas sobre el módulo sensor de huella dactilar	31
4.2. Pruebas funcionales de firmware	32
4.2.1. Pruebas de comunicación con el módulo sensor de huella .	32
4.2.2. Pruebas para la interfaz gráfica	33
4.2.3. Pruebas modelo cliente servidor	35

4.2.4. Pruebas sobre el módulo para gestión de base de datos	36
4.2.5. Pruebas sobre el servidor web	38
5. Conclusiones	39
5.1. Conclusiones generales	39
5.2. Próximos pasos	39

Índice de figuras

1.1.	Aplicación basada en biometría	2
1.2.	Disposición de hardware del sistema.	3
1.3.	Módulos funcionales del sistema.	3
2.1.	Secuencia funcional del sistema.	6
2.2.	Diagrama Activity on Node.	8
3.1.	Raspberry Pi 3 B+ módulos constitutivos.	10
3.2.	Interfaz Raspbian Lite.	11
3.3.	Pantalla touch 7 pulgadas element 14 y accesorios para conexión. .	12
3.4.	Entorno de desarrollo Glade.	13
3.5.	Procesamiento de huella dactilar.	14
3.6.	Formato para envío/recepción de paquetes de datos.	16
3.7.	Disposición del sistema general implementado.	17
3.8.	Disposición y dependencia de páginas de la interfaz diseñada. . . .	18
3.9.	Máquina de estado para validación de huella dactilar.	19
3.10.	Estructura de datos enviada desde el cliente TCP al generarse un evento.	19
3.11.	Máquina de estados para el enrolamiento de huellas.	22
3.12.	Distribución de pines Raspberry Pi 3.	24
3.13.	Esquema de servidor web implementado.	28
4.1.	Resultado de la lectura sobre la interfaz GPIO en raspberry Pi 3. . .	30
4.2.	Puertos para conexión controlador, pantalla, raspberry pi.	30
4.3.	Teclado virtual para raspberry pi y pantalla touch.	31
4.4.	Módulo lector de huellas.	32
4.5.	Resultado de envío y recepción de tramas pc-sensor.	32
4.6.	Página de inicio para la interfaz gráfica.	34
4.7.	Página para acceso mediante reconocimiento de huella.	34
4.8.	Página para acceso mediante ingreso de contraseña.	34
4.9.	Página para configuraciones.	35
4.10.	Estructura de archivos para los servicios cliente y servidor.	36
4.11.	Envío y recepción de mensajes cliente servidor.	36
4.12.	Lectura de la base de datos con cuatro usuarios registrados. . . .	37
4.13.	Lectura de la base de datos luego de borrar un usuario.	37
4.14.	Lectura de la base de datos a través de la herramienta DB Browser .	38
4.15.	Lectura de historia de accesos mediante el servidor web.	38

Índice de Tablas

3.1.	Trama para protocolo de comunicación	15
3.2.	Comandos módulo lector de huellas	15
3.3.	Denominación de pines	23
3.4.	Tabla implementada	25
4.1.	Resultado trama de envío	33
4.2.	Resultado de la trama recibida	33

Dedicado a Martha Flores

Capítulo 1

Introducción General

En este capítulo se explica que es un sistema de acceso biométrico y su importancia, además se presentan las necesidades a las que responde el proyecto junto con su objetivo.

1.1. Sistemas de control de acceso biométrico

El uso de los rasgos físicos intrínsecos de las personas como herramienta para su reconocimiento en sistemas de control es en la actualidad uno de los métodos más seguros y su implementación va en aumento.

Los sistemas biométricos evitan el tráfico de passwords o tarjetas de identificación que son métodos tradicionales y con gran deficiencia en cuanto a seguridad.

La utilización de la tecnología biométrica brinda no solo la posibilidad de implementar controles de acceso, son también utilizados para el reconocimiento en sistemas gubernamentales, transacciones bancarias, servicios de salud entre los más importantes.

Partiendo de la definición de biometría como la toma de medidas estandarizadas para el reconocimiento de personas basado en una o mas características físicas o conductuales, se establece entonces que los sistemas de control biométricos utilizan tales rasgos como fuente de información que luego de ser analizados servirán para determinar el comportamiento de determinado proceso, en el caso particular de este proyecto, establecen permisos de acceso para el usuario.

Existen dos tipos básicos de biometría; la de comportamiento, dentro de la cual se analizan características como la forma de caminar u ondas cerebrales y por otro lado la biometría fisiológica dentro de la cual se cubren los análisis de huella digital, rasgos faciales, forma de manos, retina, firma, vena, voz y análisis genético.

De los análisis mencionados, el más popular por su gran nivel de precisión en cuanto a reconocimiento es el análisis de huella digital.

La figura 1.1 ilustra las potencialidades de los sistemas de acceso biométricos.

1.2. Motivación

La principal motivación del presente trabajo se enfoca en el orden de lo académico, se desea mediante el mismo aplicar los conocimientos adquiridos durante

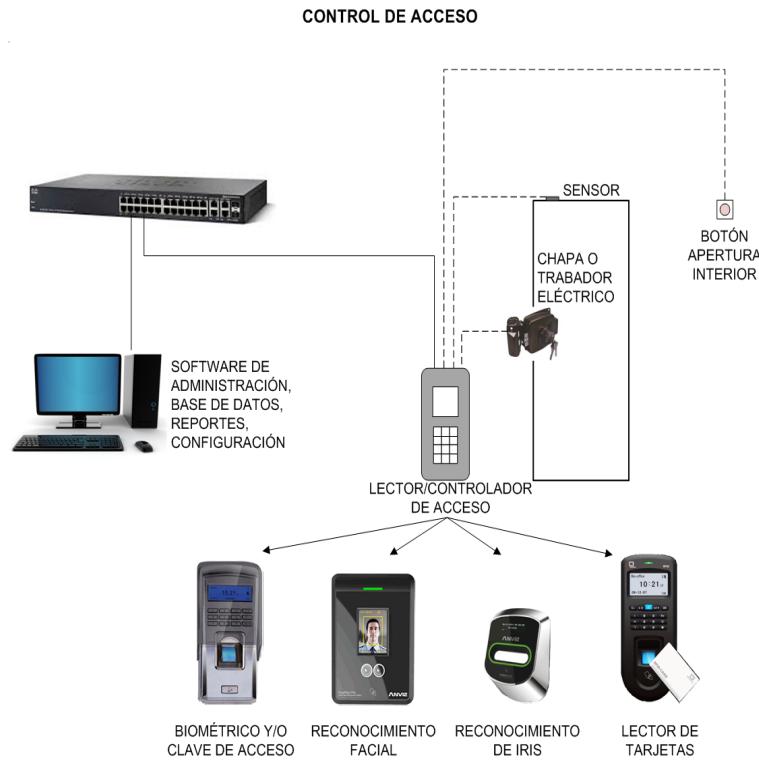


FIGURA 1.1: Aplicación basada en biometría

la carrera para desarrollar un producto tecnológico enfocado en cubrir requerimientos puntuales.

Se anhela finalizar exitosamente la especialización e involucrarse con los procesos de diseño y construcción de sistemas embebidos para un eventual emprendimiento personal.

Se pretende además, elaborar un producto final que a futuro pueda ser comercializado o acoplado a sistemas de seguridad más robustos y con aplicaciones específicas, por ejemplo, un prototipo base de identificación biométrica que pueda ser implementado en sistemas de vigilancia escolares o domiciliarios.

Se espera también implementar un sistema de alta tecnología que represente una opción frente a los módulos comerciales con el objetivo de acercar la tecnología a un mayor número de personas.

1.3. Objetivos y alcance

El desarrollo de este proyecto incluye la implementación de una interfaz gráfica mediante una pantalla que incorpora la tecnología touch, el manejo de un módulo lector óptico de huellas digitales, la incorporación del sistema de acceso alternativo mediante la utilización de password y la activación de una salida digital la cual a futuro podrá comandar el actuador para una cerradura eléctrica.

Se incluye además un sistema de monitoreo al que el usuario principal puede acceder y mediante el cual se informa el usuario registrado y la hora de acceso.

No se incluye la opción de acceso remoto ni comunicaciones con otros sistemas.

La figura 1.2 ilustra la disposición de hardware del sistema y la figura 1.3 muestra la estructura en módulos funcionales del mismo.

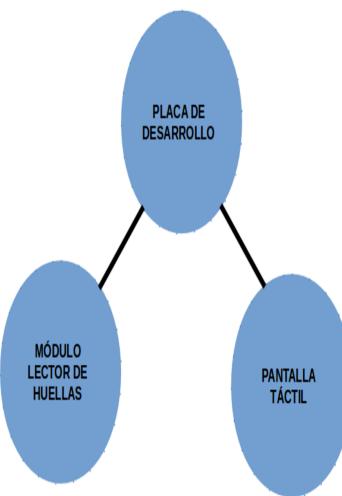


FIGURA 1.2: Disposición de hardware del sistema.

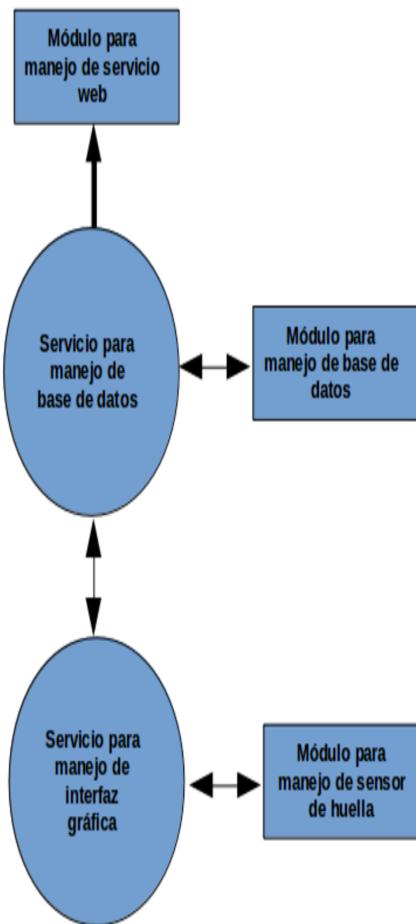


FIGURA 1.3: Módulos funcionales del sistema.

El proyecto esta basado en los siguientes lineamientos:

- Los componentes para el desarrollo son abiertos y de libre acceso.
- Interfaz amigable con el usuario final.
- Escalable.

El proyecto desarrollado incluyó:

- Selección de la plataforma de desarrollo y hardware asociado.
- Selección del módulo lector de huella.
- Selección de la plataforma para implementación de interfaz gráfica.
- Selección de la plataforma para base de datos.
- Implementación de mecanismos de comunicación con el módulo lector de huellas
- Selección del sistema operativo embebido.
- Documentación del proceso de desarrollo.

El proyecto desarrollado no incluyó:

- Diseño del circuito impreso.
- Selección de unidades de alimentación principal.
- Mecanismos de seguridad referentes al manejo de información.

Capítulo 2

Introducción Específica

En esta sección se presenta el contexto general en el cual se realizó el diseño, los elementos que conforman el sistema, la descripción de los requerimientos base y la presentación de las diferentes fases mediante las que se abordó la realización del proyecto.

Además se establece una perspectiva global de la ejecución mediante la planificación seguida durante el desarrollo.

2.1. Visión general de los elementos constitutivos

El sistema implementado se centra en una placa de desarrollo embebida, un módulo-sensor lector de huellas y una pantalla para interacción con el usuario.

La placa de desarrollo embebida puede ser definida como un sistema electrónico basado en microprocesador diseñado para realizar funciones dedicadas.

En el caso del módulo-sensor de adquisición de huella digital es un dispositivo capaz de capturar la imagen de la huella, procesarla y almacenarla para su posterior utilización.

Para una fácil manipulación del sistema por parte del usuario, se incorpora una interfaz gráfica desplegada sobre una pantalla táctil.

En el caso del software, se selecciona como base una distribución del sistema operativo Linux para sistemas embebidos.

2.1.1. Descripción funcional

La única vía para acceder al sistema por parte del usuario es a través de la interfaz gráfica la cual cuenta con un menú de tres partes principales:

- Menú para acceso biométrico (acceso por reconocimiento de huella).
- Menú para acceso mediante validación de clave numérica personal.
- Menú para configuraciones.

De esta forma el usuario puede acceder únicamente si su huella o clave esta registrada con anterioridad. Cabe mencionar que la asignación de claves y registro de usuarios es privilegio del usuario principal.

En caso de ser reconocida la huella o clave del usuario, se activa una salida digital de la placa de desarrollo (que en el futuro comandará un actuador final) y se despliega en pantalla un mensaje de notificación de bienvenida o caso contrario un mensaje de usuario no reconocido.

El usuario principal tiene además la posibilidad de acceder a una pagina web donde se guarda el registro de los últimos accesos. La figura 2.1 muestra la secuencia funcional del sistema.

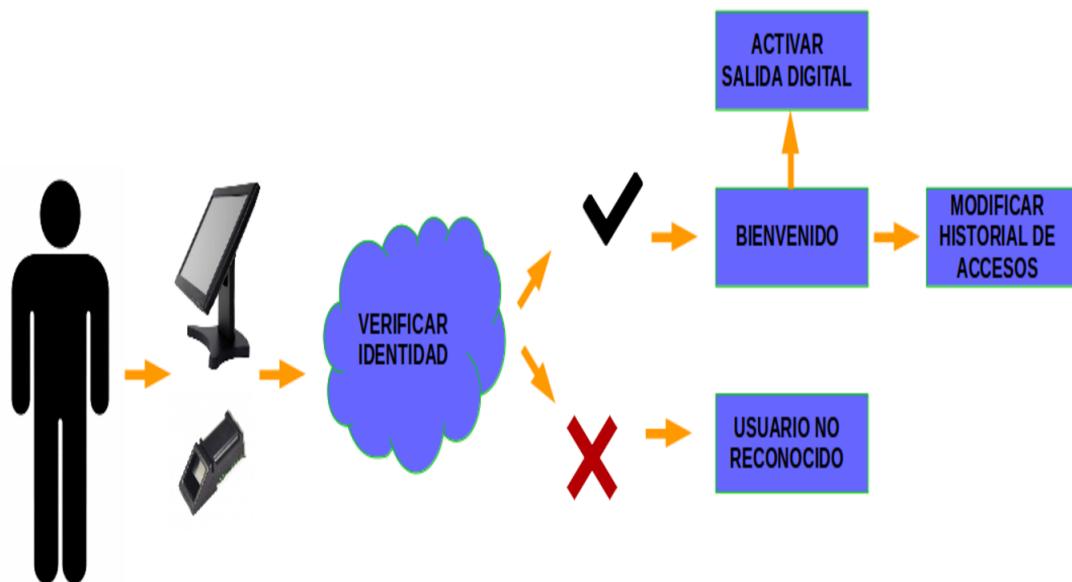


FIGURA 2.1: Secuencia funcional del sistema.

Para lograr este objetivo se diseñó un firmware con los siguientes módulos y sub módulos operativos.

- dbService: Servicio para manejo de base de datos y servidor TCP.
- logicService: Servicio para manejo de interfaz gráfica y cliente TCP.
- adafruit fingerprint: Sub módulo para manejo del sensor de huella.
- db lib: Sub módulo para lectura escritura de base de datos sobre SQLite.
- web: Sub módulo para desarrollo de la interfaz web.
- GPIO: Módulo para manejo de comunicación serial y salida digital.

Al desarrollar un sistema modular de este tipo se logra convertir la aplicación de un sensor de reconocimiento de huella en un posible sistema distribuido de seguridad, resultando en un sistema más completo. Resulta relativamente sencillo anexar mas clientes TCP que pueden a su vez manejar nuevas interfaces gráficas con nuevos sensores y con distintas funcionalidades.

2.2. Requerimientos

A continuación se presenta el detalle de los requerimientos.

1.- Requerimientos del sistema:

- 1.1. Interfaz de usuario simple.
- 1.2. El sistema posee dos modos de operación principales, biométrico y password.
- 1.3. En modo password se realiza la identificación del usuario mediante la validación de una clave numérica de cuatro dígitos.
- 1.4. En modo biométrico se realiza la identificación del usuario mediante la validación del patrón característico de la huella digital correspondiente al dedo indice de la mano derecha.
- 1.5. El sistema permite a un usuario identificado como principal, modificar la base de datos del sistema permitiendo crear nuevos usuarios o borrarlos.
- 1.6. El usuario identificado como principal tiene la opción de acceder a un registro de actividad que le informe que usuario y a que hora tuvo acceso.
- 1.7. El sistema implementa el modo de acceso permitido como la activación temporal de una salida digital de la placa de desarrollo y un mensaje en pantalla.

2.3. Planificación

A continuación se muestra el desglose de tareas del proyecto.

1. Planificación (35h).

- Realizar plan del proyecto.(15h)
- Realizar el análisis de factibilidad.(10h)
- Realizar la gestión de calidad.(10h)

2. Investigación Preliminar.(105 hs)

- Buscar info. sobre sistemas de control de acceso por parámetros biométricos.(10 hs)
- Buscar información sobre módulos de adquisición de huellas dactilares.(10 hs)
- Buscar información sobre módulos display con tecnología touch.(10 hs)
- Buscar información acerca de entornos para programación de GUIs.(15 hs)
- Buscar información sobre bibliotecas para el uso de pantallas touch.(20 hs)
- Buscar info. sobre bibliotecas para módulos de adquisición de huellas dactilares.(20 hs)
- Buscar información sobre aplicaciones web para monitoreo remoto.(20 hs)

3. Selección de módulos comerciales y plataformas para el proyecto.(130 hs)

- Selección y pruebas preliminares del módulo y display touch.(30 hs)
- Selección y pruebas preliminares del módulo de adquisición de huellas dactilares.(30 hs)

- Selección y pruebas preliminares del entorno de programación de GUIs.(45 hs)
 - Selección y pruebas para aplicación web para monitoreo remoto. (25hs)
4. Desarrollo de la interfaz de usuario (GUI).(40 hs) 5. Desarrollo de firmware.(200h)
- Implementar las funciones para la obtención de datos desde el módulo biométrico.(30 hs)
 - Implementar funciones para el desarrollo del módulo de monitoreo.
 - Corrección de errores.
6. Integración del sistema. (100h) 7. Procesos finales. (100h)

El diagrama Activity on Node final del proyecto se observa en la figura 2.2, en esta se puede apreciar el orden seguido para la ejecución de las tareas, las cuales en total suman 710 horas.

Puede notarse que el camino crítico requiere una inversión de 535 horas.

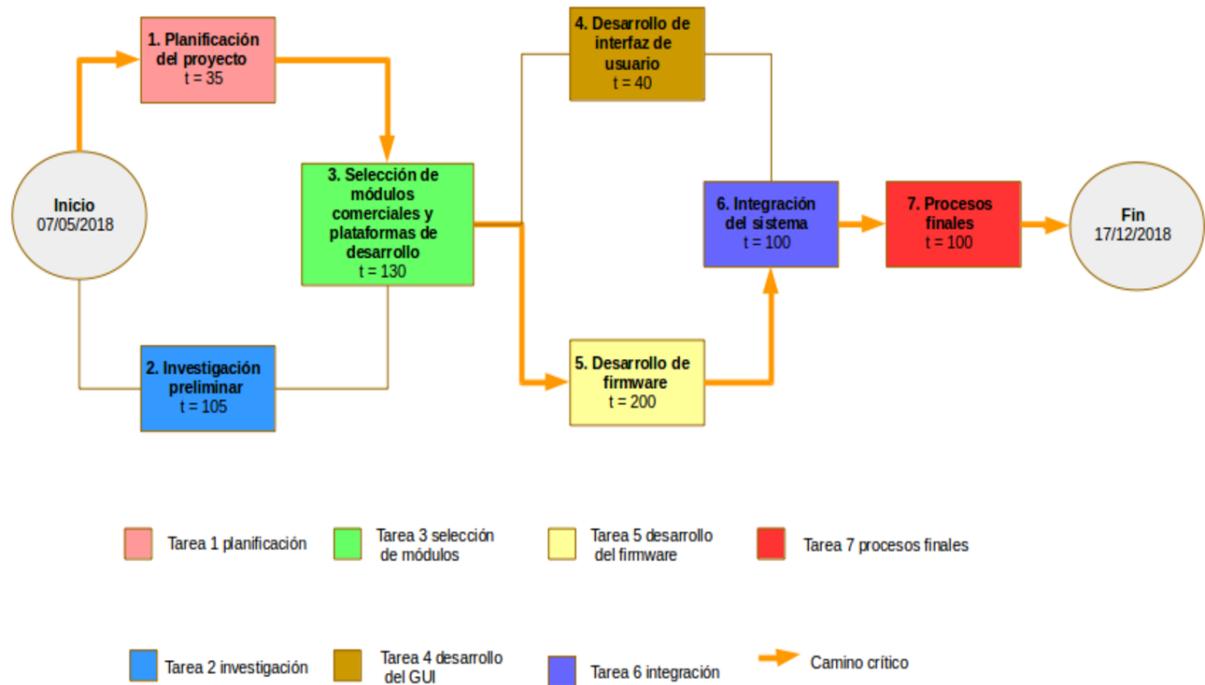


FIGURA 2.2: Diagrama Activity on Node.

Capítulo 3

Diseño e Implementación

En este capítulo se presenta las características del hardware utilizado y como se desarrollo e implementó el firmware dentro del entorno diseñado.

3.1. Plataforma de desarrollo

Uno de los aspectos importantes dentro de la ejecución del proyecto es la correcta selección de la plataforma de desarrollo ya que de esta dependen todas las funcionalidades a crear. Además, una correcta selección puede facilitar la implementación de los diferentes requerimientos.

Para el caso puntual de esta aplicación al considerar que los requerimientos demandan alta capacidad de almacenamiento, soporte para sistema operativo, interfaces de comunicación, conectividad y uso de periféricos externos, se establece como la mejor opción la placa Raspberry Pi 3 modelo B+ cuyas principales características se detallan a continuación.

Raspberry Pi es un ordenador de bajo costo pero altamente potente y de dimensiones muy pequeñas , fue desarrollada con la intención de facilitar la enseñanza de la informática y para que pueda ser utilizada por profesionales para grandes proyectos como por aficionados de la electrónica y computación.

Características técnicas:

- Procesador: Broadcom BCM2837BO Cortex-A53 (ARMv8) 64 bit
- Frecuencia: 1,4 GHz
- Memoria: 1 GB LPDDR2 SDRAM
- Conectividad inalámbrica: 2.4 GHz / 5 GHz IEEE 802.11b/g/n/ac Bluetooth 4.2 BLE
- Conectividad de red: Gigabit Ethernet over USB 2.0
- Puertos:
 - GPIO 40 pines
 - HDMI
 - 4xUSB 2.0
 - CSI (Cámara Raspberry)

- DSI (Pantalla táctil)
- Toma auriculares / video compuesto
- Micro SD
- Micro USB (Alimentación)

Otra de las ventajas del modelo seleccionado es el soporte wifi 802.11ac y del estándar Power-over-Ethernet (PoE).

La figura 3.1 muestra la distribución de los diferentes módulos constitutivos de la Raspberry.

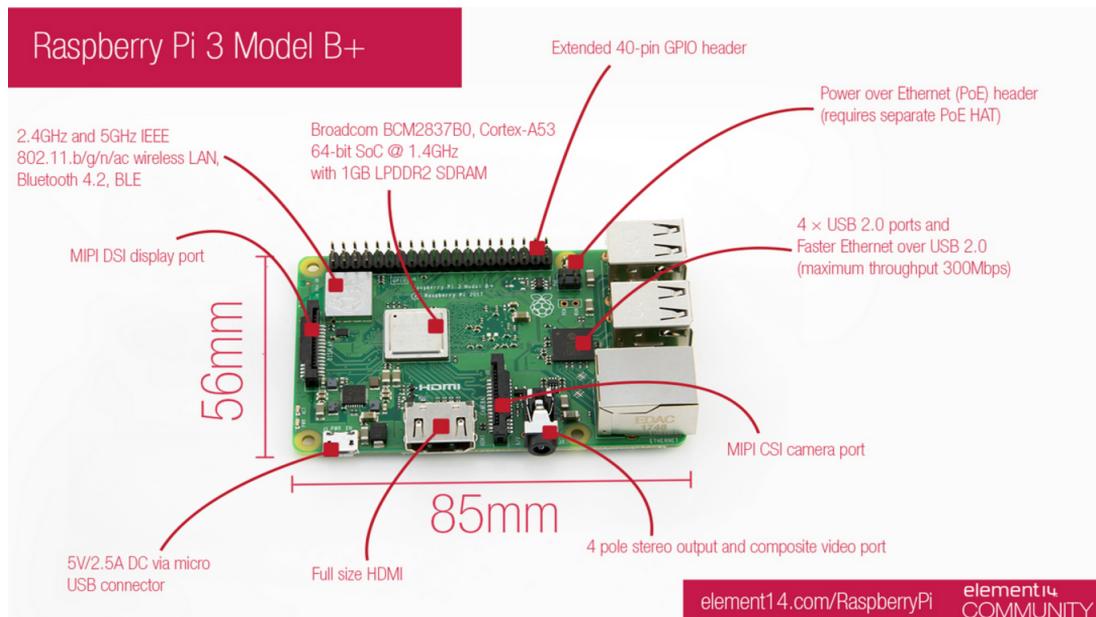


FIGURA 3.1: Raspberry Pi 3 B+ módulos constitutivos.

3.1.1. Linux embebido

El término Linux embebido se refiere al uso del núcleo Linux en un sistema embebido, en este caso la placa de desarrollo seleccionada. Dicho núcleo combinado con un conjunto de utilidades de software libre se ajustan dentro del hardware para el desarrollo del proyecto. Para tal propósito se ha seleccionado el sistema operativo Raspbian que es la distribución por excelencia para Raspberry.

Raspbian OS esta basado en la distro Debian Wheezy (Debian 7.0) optimizado para Raspberry.

La figura 3.2 muestra la versión Raspbian Lite, versión con interfaz de escritorio, ejecutándose sobre la placa embebida.

3.1.2. Pantalla táctil element 14

Pantalla de 7 pulgadas diseñada especialmente para la placa de desarrollo seleccionada y gran variedad de placas de la misma familia. Este accesorio ha sido

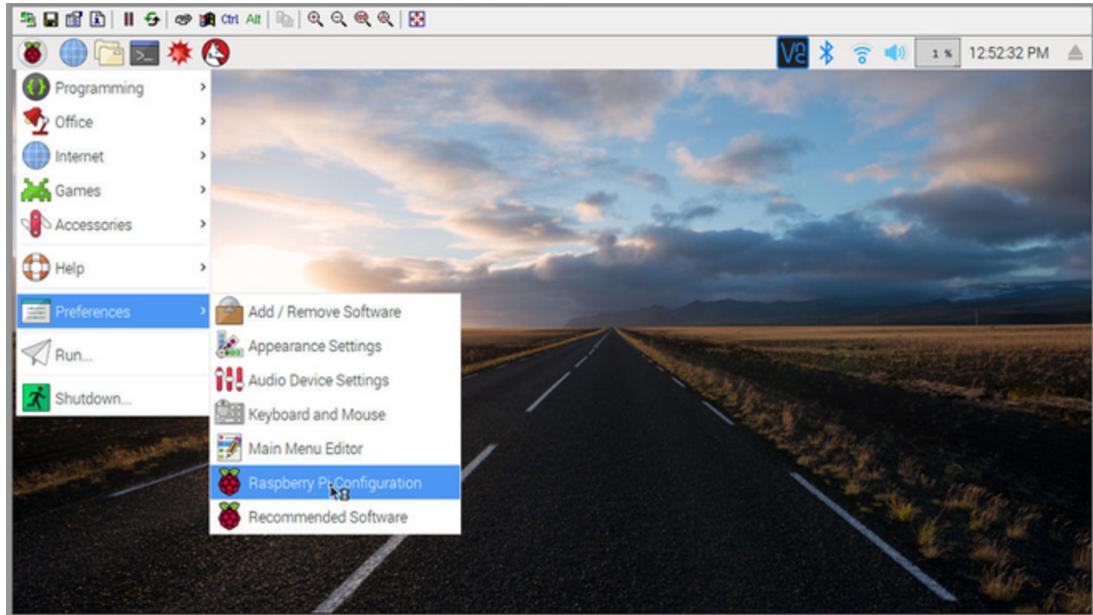


FIGURA 3.2: Interfaz Raspbian Lite.

integrado en gran cantidad de proyectos tanto de información y entretenimiento como de tecnología. Entre sus principales características se citan:

- Dimensiones : 194mm x 110mm x 20mm
- Resolución: 800 x 480 pixeles
- Tipo de detección: Capacitivo

Para su conexión con la Raspberry Pi utiliza una placa para adaptar señales de alimentación y señales lógicas, además, la comunicación se realiza mediante el puerto y protocolo DSI (Display Serial Interface).

Gracias a la implementación del sistema operativo, la pantalla funciona como un dispositivo “plug and play” por lo que no fue necesario el desarrollo de bibliotecas adicionales.

La figura 3.3 muestra los elementos constitutivos y necesarios para la conexión de la pantalla con la placa de desarrollo.

3.2. Desarrollo de interfaz gráfica

3.2.1. GTK+

GTK+ o GIMP Toolkit es una herramienta utilizada para crear interfaces gráficas de usuario, es una herramienta multiplataforma escrita en lenguaje C con soporte para el uso de otros lenguajes como Perl y Python.

GTK es software libre aunque puede ser licenciado permitiendo la creación tanto de software libre como propietario. La herramienta se basa en una serie de librerías entre las cuales destacan:

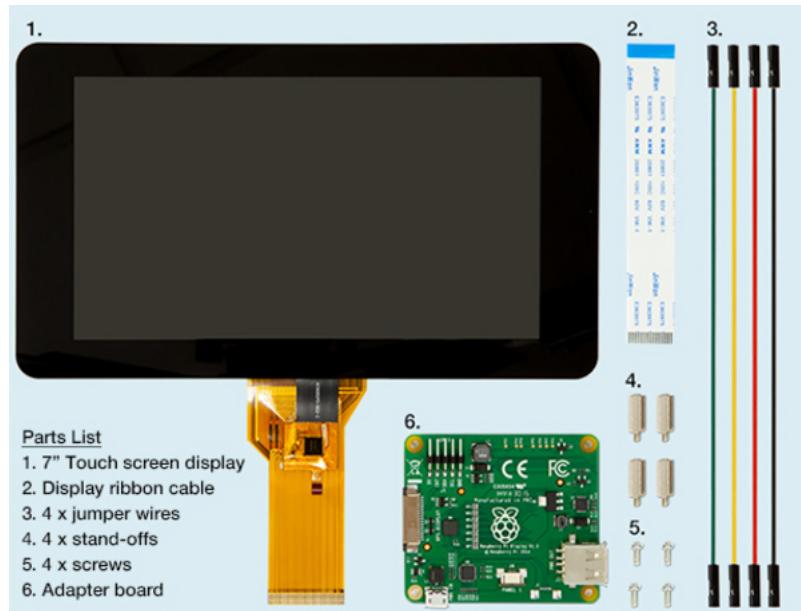


FIGURA 3.3: Pantalla touch 7 pulgadas element 14 y accesorios para conexión.

- Glib: Proporciona los bloques básicos para construir aplicaciones y bibliotecas escritas en C, proporciona la implementación del bucle principal y funciones para el uso de cadenas y estructuras de datos comunes.
- Gobject: Proporciona el sistema para manejo de objetos para el diseño y dibujo de texto internacional.
- GIO: Proporciona una abstracción del sistema de archivos permitiendo a las aplicaciones acceder a los mismos de forma local o remota consistentemente.
- GTK: Proporciona controles de interfaz de usuario y señales para los controles de usuario.

3.2.2. Glade

Glade es un entorno gráfico para el desarrollo de interfaces gráficas basadas en GTK+ el cual genera un archivo en formato XML y gracias a esta propiedad permite el desarrollo con soporte para diferentes lenguajes de programación. La figura 3.4 muestra el entorno de desarrollo de Glade.

El entorno se divide en tres columnas principales, a la izquierda la paleta de elementos con un conjunto de bloques como botones, contenedores y ventanas los cuales son posteriormente arrastrados hacia la columna central que ofrece la vista del proyecto. En esta locación se realiza la disposición de los componentes de nuestra interfaz según las necesidades del diseño.

Finalmente la columna de la derecha muestra el diagrama de árbol de la interfaz creada y también las diferentes propiedades de cada elemento como sus dimensiones, etiqueta, nombre o identificador y señales asociadas para su ejecución luego de ocurrido determinado evento.

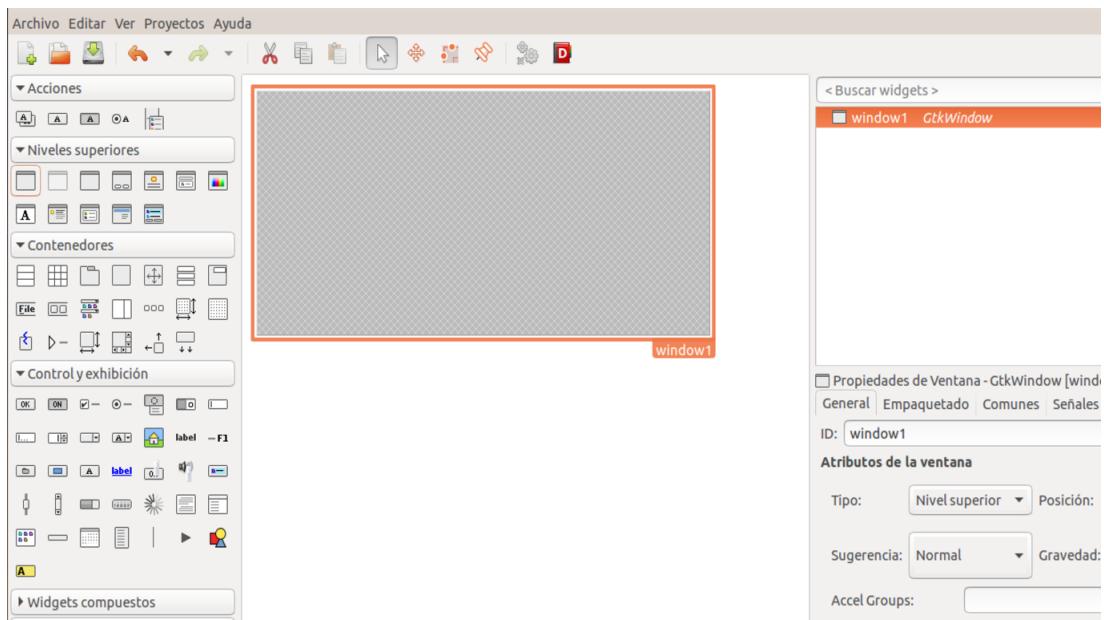


FIGURA 3.4: Entorno de desarrollo Glade.

Para el desarrollo de la presente memoria se efectuó el siguiente método de trabajo para cada una de las páginas desarrolladas.

En primer lugar se selecciona una ventana principal, sobre esta se distribuyen el resto de componentes; para lograr una distribución mas uniforme y estructurada se hace uso de contenedores, una vez con los elementos en su posición final, se procede a establecer identificadores adecuados para cada uno así como la asignación de señales a eventos.

La asignación de señales e identificadores es muy importante ya que estos se utilizan luego en el programa principal. Finalmente todo el diseño se guarda como un archivo con extensión (.glade) el cual se invocará desde la aplicación final.

3.3. Módulo sensor para adquisición de huellas

3.3.1. Lectores ópticos reflexivos

El funcionamiento del sensor de huella comienza luego de colocar el dedo sobre la superficie de cristal del mismo la cual esta iluminada mediante un diodo led. La luz incide sobre el dedo causando el efecto de reflexión, este efecto se manifiesta con la aparición de zonas mas oscuras en las crestas de la huella y zonas mas claras en los valles.

La imagen recogida es procesada en busca de características relevantes (Minucias), luego de esto, se genera una plantilla digitalizada la cual se guarda para ser consultada posteriormente. La figura 3.5 ilustra brevemente el proceso realizado por el sensor.

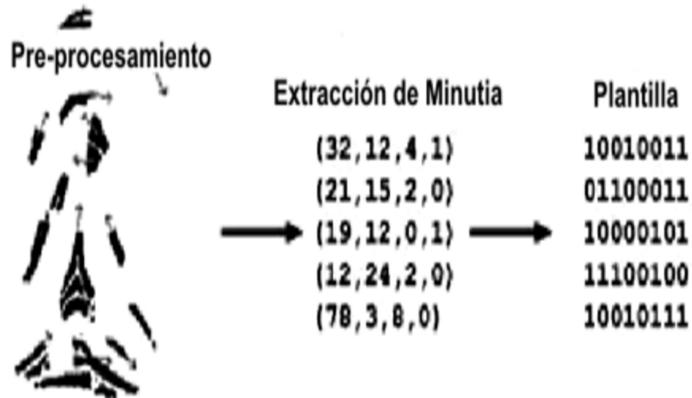


FIGURA 3.5: Procesamiento de huella dactilar.

3.3.2. Módulo lector de huella Adafruit ZFM-20

Módulo basado en el chip AS608 para el procesamiento digital el cual permite la representación, cálculo, extracción de características y cotejamiento de huellas. Entre los principales atributos se destacan:

- Fuente de alimentación: 3.6 – 6V.
- Corriente de operación: 120mA max.
- Tiempo de respuesta: <1s.
- Área activa del sensor: 14mm x 18mm.
- Capacidad de almacenamiento: hasta 162 patrones.
- Rangos de seguridad programables: 1 a 5.
- Taza de rechazo falso: <1
- Interfaz de comunicación: UART (TTL), USB.
- Velocidad de transmisión: 9600, 19200, 28800, 38400, 57600.

Otras características importantes de mencionar son la presencia de memoria RAM de 72Kb, un buffer para almacenamiento de imágenes y dos buffers de archivo para procesamiento los cuales pueden ser accedidos aleatoriamente mediante las instrucciones del sistema.

El módulo posee además una base de datos conocida también como librería en la cual se realiza el almacenamiento de patrones de huellas en un espacio de memoria flash.

Para la comunicación del sistema con el sensor se utiliza el interfaz UART y para tal propósito el dispositivo lector de huellas establece su propio protocolo el cual consta de comandos y datos enviados en paquetes con los elementos mostrados en la tabla 3.1.

Existen diversos comandos que permiten trabajar con todas las funcionalidades del módulo lector, la tabla 3.2 muestra un resumen de los principales.

Una vez configurados los parámetros para la comunicación que por defecto están establecidos con un baud rate de 57600, 8 bits de datos y 1 bit de parada; se envía

TABLA 3.1: Elementos que conforman el paquete de datos para el protocolo de comunicación

Nombre	Símbolo	Tamaño	Descripción
Inicio	START	2 bytes	Valor fijo 0xEF01.
Dirección	ADDER	4 bytes	Permite la identificación de varios sensores sobre un mismo sistema, por defecto su valor es 0xFFFFFFFF.
Id de paquete	ID	1 byte	0x01 indica envío de comando, 0x07 indica respuesta del sensor.
Longitud	LENGTH	2 bytes	Tamaño de la carga de datos a recibir
Datos	DATA	–	Pueden ser instrucciones, datos o parámetros.
Checksum	SUM	2 bytes	Utilizado para comprobar un envío correcto.

TABLA 3.2: Principales comandos para el módulo lector de huella

Tipo	Código	Descripción
Sistema	0x12	Cambiar contraseña.
Sistema	0x15	Cambio de dirección.
Comunicación	0x0e	Cambio de parámetros.
Imagen	0x01	Cargar imagen.
Imagen	0x02	Extracción de característica.
Imagen	0x03	Montaje de imágenes.
Proceso	0x04	Buscar modelo.
Proceso	0x06	Guardar modelo.
Proceso	0x0c	Borrar modelo.

la trama iniciando desde el bit más significativo respetando el orden mostrado en la tabla 3.1. En el caso de la recepción de la respuesta el procedimiento es el mismo.

La figura 3.6 muestra el ejemplo del formato de envío del código para adquirir la imagen de una huella y el formato de respuesta esperado.

En el ejemplo expuesto, la confirmación por parte del sensor podría tener tres posibles casos listados a continuación:

- Código de confirmación = 0x00 correspondiente a un proceso exitoso.
- Código de confirmación = 0x01 correspondiente a una recepción de datos fallida.
- Código de confirmación = 0x02 correspondiente a una adquisición vacía, es decir, no hay un dedo sobre el sensor.

Formato de envío de datos.

Inicio	Dirección	ID	Longitud	Datos/Instrucción	checksum
2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
0xef01	0xffffffff	0x01	0x0003	0x01	0x005

Formato de recepción de datos

Inicio	Dirección	ID	Longitud	Datos/Confirmación	checksum
2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
0xef01	0xffffffff	0x07	0x0003	--	0x005

FIGURA 3.6: Formato para envío/recepción de paquetes de datos.

3.4. Gestión de base de datos (SQLite)

En primera instancia no se planteó la posibilidad de trabajar con herramientas para la gestión de datos de usuarios, sin embargo, pensando en un desarrollo futuro, la portabilidad, la seguridad y un mejor desarrollo se implementa la herramienta SQLite.

SQLite es una biblioteca escrita en lenguaje C que permite implementar un motor de base de datos transaccional de SQL(Structured Query Language) auto contenido y sin servidor. Esta librería de código abierto a diferencia de otras bases de datos no tiene un proceso de servidor separado, esto le permite leer y escribir directamente en archivos de disco ordinarios, con esto se logra tener una base de datos SQL completa con múltiples tablas, índices, activadores y listas en un solo archivo de disco.

El formato de archivo de la base de datos obtenido con SQLite es multiplataforma lo que le permite trabajar libremente con bases de datos en sistemas de 32 y 64 bits o entre arquitecturas big-endian y little-endian.

La versión utilizada (SQLite3) permite crear bases de datos de hasta 2 Terabytes y contando con que es una librería que alcanza un promedio en tamaño de alrededor de 275 KiB, se establece que SQLite es ideal para ser implementada en sistemas embebidos.

3.5. Sistema General

Para la implementación del proyecto se seleccionó el modelo cliente-servidor para comunicación TCP entre procesos corriendo sobre el sistema operativo Linux.

Para tal propósito, el servidor llamado en adelante dbService, inicia el proceso de comunicación y queda a la espera de posibles clientes, puede soportar comunicación hasta con 100 clientes a los cuales asocia un canal propio y una rutina de ejecución propia.

A pesar que dbService tiene la capacidad de conectarse con muchos clientes, esa ventaja se utilizará en el futuro para lo será un sistema distribuido de seguridad.

DbService se basa en la teoría de un manejador de eventos, es así que luego de establecerse la comunicación, queda en espera de un evento que es proporcionado por el cliente, el cual es representado como un paquete de información que contiene el origen y los datos asociados a tal acontecimiento. La información es procesada mediante una máquina de estados la cual determinará la eventual salida del sistema.

Por otra parte, dbService se encarga de leer o escribir la información en una base de datos mediante una biblioteca desarrollada en base a SQLite siendo el único servicio con esta facultad.

Además, el servicio mencionado se encarga de generar y escribir sobre un archivo de texto que contiene el historial de últimos accesos; dicho archivo sirve posteriormente para ser mostrado en el servidor web.

Finalmente, el cliente llamado en adelante logicService, se encarga del envío de eventos al servidor los cuales son enviados luego de ser adquiridos por la interfaz gráfica o el sensor y luego de haber sido procesados.

LogicService se encarga del manejo de la interfaz gráfica y de la comunicación con el sensor lector de huella. La figura 3.7 ilustra la disposición del sistema general y los sub sistemas conformantes.

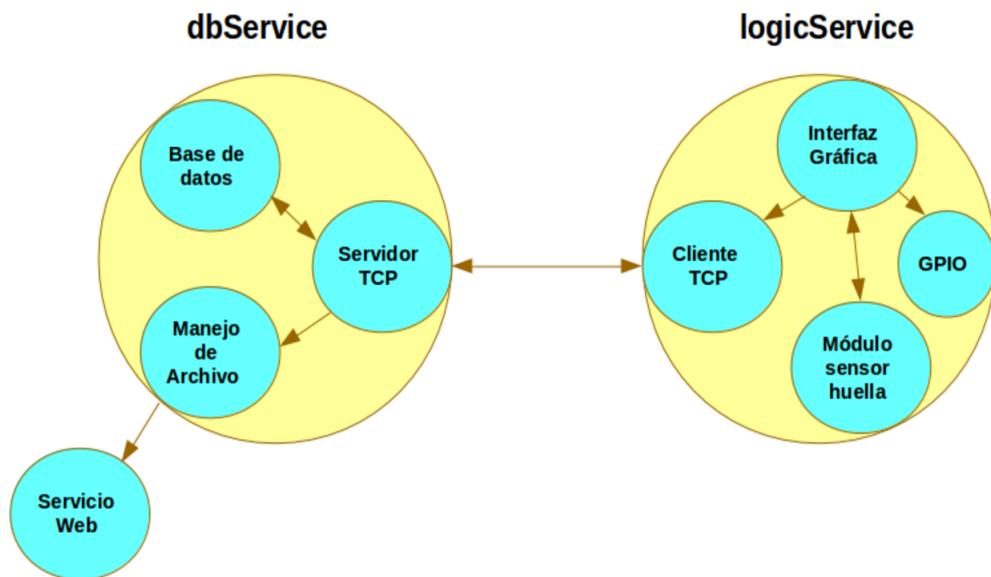


FIGURA 3.7: Disposición del sistema general implementado.

3.6. Subsistema logicService

A continuación se detallan los módulos que forman parte del sub sistema logicService y su funcionamiento dentro del entorno.

3.6.1. Módulo para el manejo de la interfaz gráfica

Como fue mencionado, la interfaz para el proyecto se desarrolló en base a la librería gtk+ bajo el entorno glade.

El módulo que maneja la interfaz se basa en la teoría de threads ya que la librería gtk+ da soporte para esta implementación, es así que cada página que forma parte de la interfaz puede ser manejada como un proceso paralelo e independiente.

La interfaz gráfica implementada consta de cuatro páginas.

La primera es la página de inicio en la que mediante tres botones se puede acceder a una página y una tarea asociada a cada botón.

Cada una de las ventanas invocadas por los botones son ventanas activas ya que dentro de su respectiva tarea realizan diferentes acciones que corren en segundo plano. Por tal razón fueron diseñadas para que su ejecución sea mutuamente excluyente, es decir, solo la página y por ende la tarea desplegada tiene acceso a los datos y vías de comunicación.

La figura 3.8 muestra la disposición de las páginas en la interfaz y la dependencia entre las mismas.

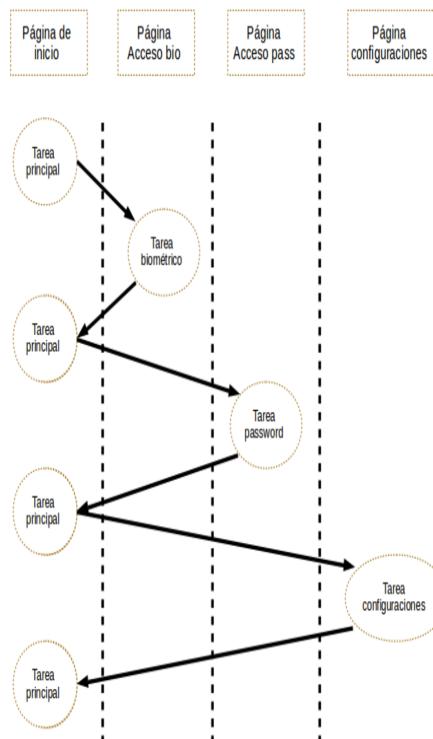


FIGURA 3.8: Disposición y dependencia de páginas de la interfaz diseñada.

La segunda página (Acceso bio) levanta los módulos sensor huella, en adelante llamado (fingerprint service), y el módulo GPIO. En esta se implementa el reconocimiento de la huella digital y para tal propósito se desarrolla una máquina de estados que funciona con la siguiente lógica:

El sistema se comunica con el sensor y en caso de haber la presencia de un dedo sobre el cristal del mismo, cambia de estado, recoge la imagen y la procesa, busca coincidencia con su base de datos interna y, si existe, levanta el módulo GPIO para activar una salida digital, además, despliega un mensaje en pantalla.

En caso de no existir similitud de huellas se envía un mensaje de notificación a pantalla; vuelve al estado inicial luego de remover el dedo del cristal.

La figura 3.9 muestra el diagrama para la máquina de estados diseñada.

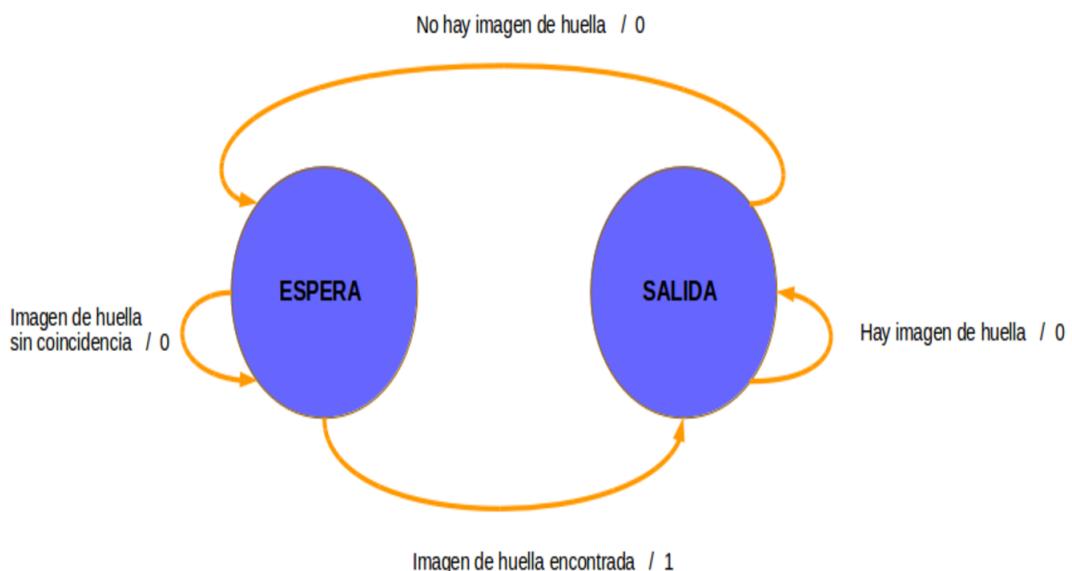


FIGURA 3.9: Máquina de estado para validación de huella dactilar.

Finalmente, en caso de existir una coincidencia con la huella ingresada, el sistema genera un evento enviando un paquete de datos tipo estructura con los campos mostrados en la figura 3.10.

El envío se realiza a través de la comunicación TCP donde serán recibidos por el servidor y posteriormente procesados.

ID Proceso
ID Usuario
Nombre
Contraseña

FIGURA 3.10: Estructura de datos enviada desde el cliente TCP al generarse un evento.

Los elementos de la estructura de datos son:

- ID Proceso corresponde al identificador de la página que genera el evento y se representa mediante un número entero.
- ID Usuario representa el número de la ubicación dentro de la librería del sensor en la cual se encuentra el modelo con la coincidencia de la huella, se representa también mediante un número entero.
- Nombre corresponde al campo donde se asigna al usuario un nombre, este campo se representa mediante un arreglo de caracteres.
- Contraseña es el campo donde se guarda la contraseña numérica del usuario y es representada mediante un arreglo de números enteros.

No todas las páginas llenan todos los campos luego de un evento, únicamente son completados los campos que correspondan a la aplicación que se ejecuta y el respectivo evento, los demás campos no se tocan; el servidor se basa en el ID Proceso para seleccionar que datos que debe analizar y descarta los otros.

La tercera página (Acceso pass) permite la validación de una clave numérica de cuatro dígitos.

El usuario introduce la clave mediante el teclado virtual desplegado en la pantalla la misma es enviada y validada por el servidor.

Finalmente la cuarta página corresponde a la interfaz para la configuración del sistema la cual se encuentra diseñada pero no está integrada operativamente por lo que su vinculación se realizará en el futuro, esta página permitirá modificar los diferentes campos de los usuarios registrados, generar nuevos o borrar los existentes y su uso será restringido.

3.6.2. Módulo para el manejo del sensor lector de huella

Este módulo es el encargado de interactuar con el sensor enviándole las tramas necesarias para la adquisición, procesamiento, almacenamiento y borrado de las huellas así como las posibles configuraciones del mismo.

Para tal propósito se desarrolló una biblioteca con diversas funciones entre las que se destacan:

- (void)AdafruitFingerprintpacket(int tipo,int tamaño,int* data): Encargada de formar la trama según el modelo visto en la figura 3.6.

Esta función es la base para interactuar con el sensor y está contenida dentro del resto de funciones.

Parámetros:

- tipo, corresponde a la identificación como comando o respuesta.
- tamaño, indica el tamaño del mensaje entre la instrucción, los datos y el checksum sin considerar el encabezado.
- data, es un puntero a un arreglo que contiene la instrucción a realizarse y los datos correspondientes a la misma.
- (int)GetStructuredPacket(struct* packet, int timeout): Función encargada de recibir la trama de respuesta desde el sensor. Recibe cada uno de los

elementos de la trama y los almacena en una estructura donde pueden ser consultados posteriormente.

Parámetros de entrada:

- packet, puntero a estructura donde se almacena la respuesta recibida.
- timeout, valor que establece el tiempo máximo de espera antes de generar un error de comunicación.

Parámetro de salida: Luego de comparar el valor del campo tamaño de los datos recibidos con el número de elementos contenidos en la estructura, devuelve un valor 0x00 en caso de coincidir o caso contrario un valor 0xfe.

- (void)WriteStructuredPacket(struct* packet): Encargada de enviar a través del puerto serial la trama formada por la instrucción AdafruitFingerprintPacket, también es la responsable de generar el checksum y enviarlo al final del paquete.

Parámetros: packet, puntero a la estructura que contiene la trama de envío.

- (int)Image2Tz(int slot): Se encarga de digitalizar la imagen contenida en uno de los dos buffers de archivo.

Parámetro de entrada: slot, indica el buffer de archivo en el cual esta cargada la imagen a digitalizar.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int)CreateModel(void): Permite crear el modelo de huella en base a las características encontradas en dos imágenes cargadas con anterioridad en los buffers de archivo; en caso que las imágenes sean muy diferentes notifica el error.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int)StoreModel(int location): Una vez creado el modelo de la huella, el modelo es guardado en los dos buffers de archivo. La función se encarga de guardar el modelo en el buffer de imagen alojada en la memoria flash.

Parámetro de entrada: location, indica el lugar dentro de la librería en el cual se guardará el modelo, son posibles de 1 a 162.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int>DeleteModel(int location): Permite borrar un modelo contenido en la librería del sensor.

Parámetro de entrada: location, indica el lugar dentro de la librería en el cual está el modelo a ser borrado.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int)EmptyDatabase(void): Permite borrar toda la librería del sensor.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int)GetImage(void): Permite obtener la imagen de la huella del dedo ubicado sobre el cristal del sensor.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

- (int)FingerFastSearch(void): Compara una imagen cargada en uno de los buffers de archivo, con los modelos de la librería del sensor y notifica en caso de haber coincidencia.

Parámetro de salida: valor de tipo entero con el código que indica un proceso exitoso o algún posible error.

La figura 3.11 muestra la máquina de estados desarrollada para la implementación del enrolamiento de huellas mediante el uso de la librería adafruit fingerprint.

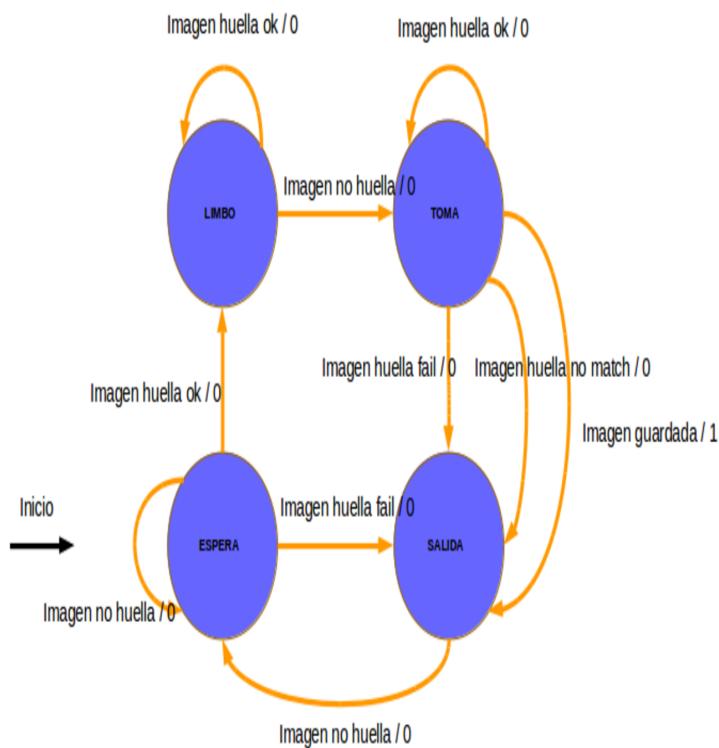


FIGURA 3.11: Máquina de estados para el enrolamiento de huellas.

3.6.3. Módulo GPIO

Para la implementación de la salida digital, así como el manejo de la comunicación serial por parte de la plataforma de desarrollo, se utiliza la librería WiringPi la cual esta escrita en lenguaje C y es compatible con los modelos de placas basadas en los procesadores BCM2835, BCM2836 y BCM2837.

Esta biblioteca esta diseñada especialmente para trabajar en Raspberry Pi que ejecuta Raspbian, el uso para otras condiciones no esta soportado.

Las funciones utilizadas para la comunicación serial son:

- int serialOpen (char *device, int baud); abre la comunicación del dispositivo.
- void serialClose (int fd) ; Cierra la comunicación con el dispositivo.
- void serialPutchar (int fd, unsigned char c) ; envía un byte único al dispositivo receptor.
- int serialDataAvail (int fd) ; devuelve el número de caracteres disponible para lectura.
- int serialGetchar (int fd) ; lee un carácter disponible en el dispositivo.
- void serialFlush (int fd) ; descarta los datos recibidos en espera.

Las funciones utilizadas para configurar la salida digital son:

- void pinMode(int pin, int mode): Sirve para especificar el modo de operación del pin seleccionado.
- void digitalWrite(int pin, int value): Se utiliza para poner un pin, que previamente ha sido configurado como salida un 1 lógico o 0 lógico.

La descripción en detalle de la biblioteca se encuentra en [[WIRINGPI](#)].

La tabla 3.3 muestra la relación entre las denominaciones de los pines para el procesador, la placa y la librería utilizados.

TABLA 3.3: Relación de denominaciones pin, procesador y librería

No Pin	GPIO	WiringPi	Función
8	14	15	TX
10	15	16	RX
16	23	4	Salida digital

La figura 3.12 muestra la distribución de pines de la placa de desarrollo.

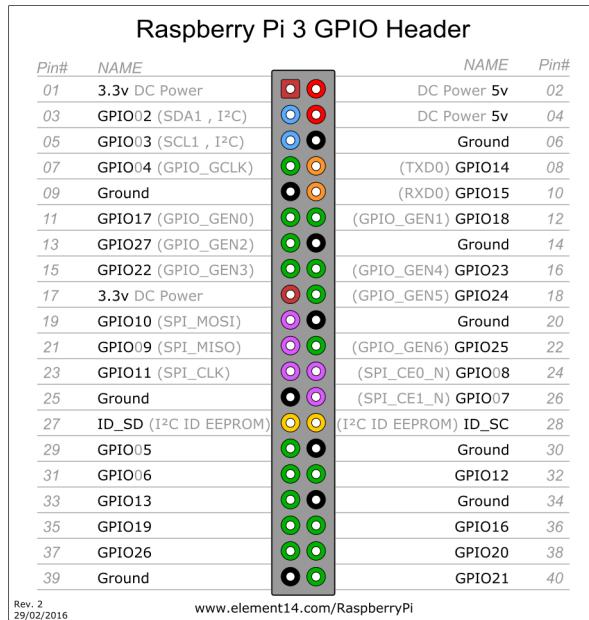


FIGURA 3.12: Distribución de pines Raspberry Pi 3.

3.7. Subsistema dbService

A continuación se detallan los módulos que forman parte del sub sistema dbService y su funcionamiento dentro del entorno.

3.7.1. Módulo para el manejo de la base de datos

Para acceder y recibir datos desde una base SQLite no es necesario trabajar con un estructura cliente – servidor ya que este tipo de bases se operan directamente desde el disco.

En consecuencia se trabaja directamente mediante el uso de comandos los cuales se los puede dividir en dos grandes grupos:

Los comandos META utilizados para definir el formato de salida para las tablas, examinar los datos y para operaciones administrativas.

Por otro lado están los comandos ESTÁNDAR, utilizados para operar sobre la base de datos. Dentro de esta rama se puede definir la siguiente clasificación:

- Comandos para definición de datos: Brindan la estructura y métodos de almacenamiento, ejemplo: CREATE, ALTER, DROP.
- Comandos para manipulación de campos: Permiten manejar los datos, ejemplo: INSERT, UPDATE, DELETE.
- Comandos para consulta: Utilizados para la recuperar los datos necesarios de la base, ejemplo: SELECT.

Existen gran variedad de instrucciones a parte de las mencionadas, sin embargo, no son tratados en su totalidad ya que superan los alcances del proyecto y únicamente se mencionan los relevantes para este trabajo.

Tipos de datos soportados:

Los datos almacenados en la base pueden ser implementados con alguno de los siguientes tipos:

- NULL: Valor nulo.
- INTEGER: Entero con signo entre 1 a 8 bytes.
- REAL: Número de punto flotante almacenado en 8 bytes.
- TEXT: Cadena de texto almacenada con las codificaciones UTF-8, UTF-16BE ó UTF-16-LE.
- BLOB: Datos en formato binario.

Formato de tabla:

La tabla 3.4 ilustra los campos contenidos en la base de datos implementada.

TABLA 3.4: Campos constitutivos de la base de datos

ID	NOMBRE	PASSWORD	IDSENSOR
1	Principal	1234	1
2	User1	1111	2
3	User2	2222	3

Diseño de la biblioteca para el manejo de la base de datos.

Existe una extensa librería en lenguaje C que permite trabajar con SQLite, la biblioteca desarrollada para este proyecto esta basada en las siguientes funciones pertenecientes a la librería estándar:

- sqlite3 open(const char *filename, sqlite3 **ppDb): Función que permite abrir un fichero de base de datos, especificando el nombre para el mismo. Filename define el nombre de la base de datos ppDB almacena el manipulador de la conexión con la base creada.
- sqlite3 close(sqlite3 *): cierra la conexión con la base, liberando todos los recursos asociados. Sqlite3 almacena el manipulador de la conexión a ser cerrada.
- sqlite3 exec(sqlite3*, const char *sql, int (*callback)(void*, int, char**,char**), void*, char** ermsg): Permite ejecutar múltiples sentencias SQL mediante un solo llamado. Sqlite3 almacena el manipulador de la conexión hacia la base de datos. Sql contiene la función SQL a ser ejecutada. Callback es la función de retro llamada que puede ser invocada tras la ejecución de la sentencia SQL. Void es el primer argumento enviado a la función de retro llamada. Ermsg corresponde al un mensaje de error.

Funciones implementadas.

En base a las funciones descritas anteriormente, se desarrollan las siguientes funciones que permiten las operaciones básicas con la base de datos.

- create db: Permite crear una nueva base de datos o abrir una conexión con una existente.

Parámetros de entrada: data, puntero al identificador o manipulador de la base; name, nombre de la base a ser manipulada o creada.

Parámetro de salida: Ninguno.

- `create table`: Permite crear una tabla con diferentes campos sobre una base de datos abierta.

Parámetros de entrada: data, puntero al identificador o manipulador de la base; sql dat, puntero a la función SQL a ejecutarse; name, nombre de la base sobre la que se creará la tabla.

Parámetro de salida: Ninguno.

- `insert user`, `print user` y `delete user` realizan las tareas de agregar un nuevo campo o usuario, obtener datos o borrarlos respectivamente. Poseen los mismos parámetros de entrada y salida que la función `create table`.

El acceso y manipulación de base de datos se realiza en el servicio dbService el cual se encarga de gestionar los datos de los usuarios dependiendo del evento que se genere.

3.7.2. Módulo para el manejo de archivo

Con el fin de establecer un medio de comunicación entre los servicios dbService y servicio web, se implementa un pequeño módulo para el manejo de ficheros bajo el lenguaje C.

El estandar de C contiene varias funciones para la edición de archivos a las cuales se puede acceder desde la librería stdio.h.

El sistema crea un archivo tipo texto sobre el cual se escribe la información del usuario que tuvo un reconocimiento previo sea por medio de del sensor biométrico o por clave de seguridad.

Sobre tal fichero se escriben los datos uno a continuación de otro generando de este modo un historial de accesos.

Luego de realizar la escritura correspondiente de los datos, el archivo puede ser recogido por el servicio web para ser mostrado en la pagina.

La manipulación del archivo esta a cargo de dbService quien coordina la lectura desde la base de datos y la escritura sobre el archivo.

3.8. Subsistema Servicio Web

Para lograr que el usuario pueda acceder eventualmente al historial de accesos registrados por el sistema, se implementó un servidor web elemental sobre la placa de desarrollo.

Para conseguir dicho objetivo existen en la actualidad diferentes herramientas combinadas que permiten lograr servidores con gran prestación de servicios. Este proyecto se basa en uno de los paquetes informáticos mas difundidos para tal trabajo, la arquitectura LAMP, acrónimo de Linux, Apache, MySQL y PHP.

Mediante el paquete mencionado se logra ejecutar páginas web HTML dinámicas; se aclara que la herramienta MySQL es reemplazada por SQLite para realizar el manejo de la base de datos.

A continuación se describe las características mas relevantes de cada uno de los paquetes informáticos utilizados.

- Linux: Sistema operativo embebido distribución Raspbian Lite.
- Apache: Se trata de el servidor web HTTP más utilizado, de código abierto y multiplataforma.

Permite procesar archivos escritos en diferentes lenguajes de programación como PHP, Python, Java entre otros.

Es altamente personalizable ya que posee una estructura en módulos permitiendo al administrador activar o desactivar funcionalidades.

Posee módulos de seguridad actualizados con frecuencia, almacenamiento de cache, reescritura de URL, verificación de contraseña entre los más importantes.

Permite configurar un hosting virtual basado en Ips o en nombres permitiendo alojar varios sitios web en el mismo equipo.

- SQLite: Sistema de gestión para el manejo de base de datos.
- PHP (PHP Personal Hypertext Preprocessor): Lenguaje de código abierto para generación de scripts para el desarrollo de páginas web el cual puede ser embebido en páginas HTML.

Con sintaxis recurrente a lenguaje C, permite una sencilla su implementación.

Incluye una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes.

Permite la conexión a diferentes tipos de servidores de bases de datos tanto SQL como NoSQL.

El código escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutarlo y enviar el resultado.

Basado en la arquitectura Linux, Apache, SQLite, PHP (LASP), el proyecto implementa un script para la lectura del archivo que contiene el historial de accesos y lo muestra mediante el navegador mediante una conexión local.

La figura 3.13 muestra el esquema final del servidor web implementado.

La página desarrollada es muy sencilla pero gracias a la estructura sobre la cual esta montada brinda grandes posibilidades de ampliación, dejando para trabajo futuro la posible implementación de formularios o gráficas que potencialicen al sistema.

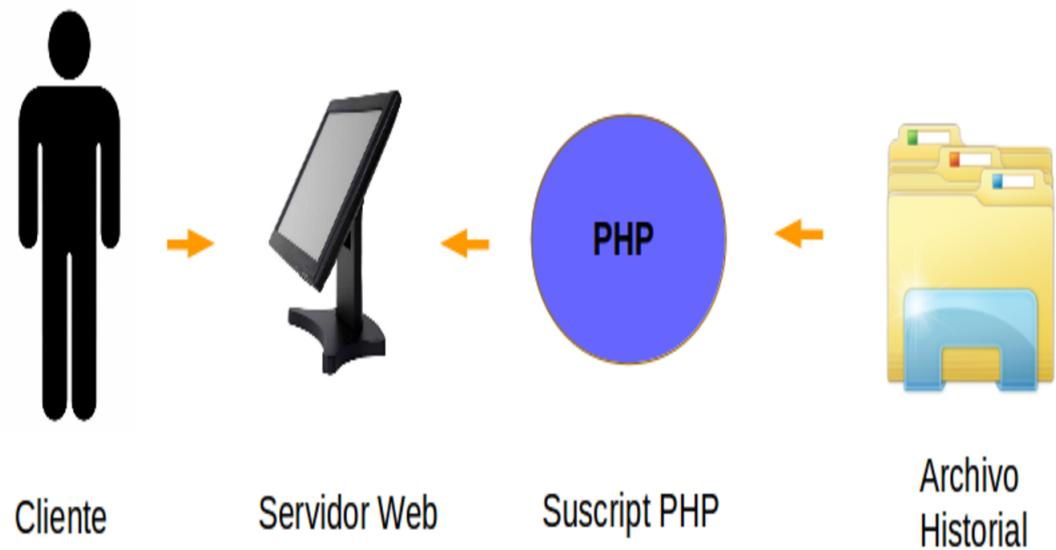


FIGURA 3.13: Esquema de servidor web implementado.

Capítulo 4

Ensayos y Resultados

En esta sección se detallan los ensayos realizados para probar el correcto funcionamiento del hardware y firmware.

4.1. Pruebas funcionales del hardware

Con respecto al hardware se realizaron ensayos individuales de cada componente.

4.1.1. Pruebas sobre la plataforma de desarrollo

La prueba de funcionamiento de la placa se realizó posteriormente a la instalación del sistema operativo. Para tal instalación es necesario cargar la imagen ISO del sistema en una memoria micro SD con capacidad de almacenamiento mayor o igual a 8GB.

Una vez con la imagen del sistema operativo en la memoria, esta es colocada en la ranura de la placa destinada para tal propósito.

Posterior a esto se energiza el dispositivo con una fuente idónea según las especificaciones y de manera automática arrancará el sistema operativo.

Para poder visualizar y realizar las primeras configuraciones es necesario conectar una pantalla, un mouse y un teclado.

Las primeras configuraciones a realizar son el cambio de contraseña, el idioma y configuraciones de red inalámbricas. Para tal propósito se puede utilizar el entorno gráfico o mediante comandos sobre la consola. Todas las configuraciones iniciales se realizaron sin mayor inconveniente demostrando la operatividad de la placa como del sistema operativo.

Finalmente se realiza la prueba de lectura de la interfaz GPIO para determinar la operatividad de la misma. Para esto se ejecuta desde la consola el comando gpio readall de la librería wiringPi previamente instalada, el resultado obtenido se muestra en la figura 4.1 .

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
Pi 3											
		3.3V			1 2			5v			
2	8	SDA.1	ALT0	1	3 4			5V			
3	9	SCL.1	ALT0	1	5 6			0v			
4	7	GPIO. 7	IN	1	7 8	1	ALT5	TxD	15	14	
		0v			9 10	1	ALT5	RxD	16	15	
17	0	GPIO. 0	IN	0	11 12	0	IN	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15 16	0	IN	GPIO. 4	4	23	
		3.3V			17 18	0	IN	GPIO. 5	5	24	
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39 40	0	IN	GPIO.29	29	21	
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	

FIGURA 4.1: Resultado de la lectura sobre la interfaz GPIO en raspberry Pi 3.

4.1.2. Pruebas sobre el conjunto pantalla, controlador, plataforma

Se realizaron pruebas de funcionalidad sobre el conjunto de componentes, para tal objetivo se cuenta previamente con la instalación del sistema operativo sobre la placa de desarrollo y las siguientes conexiones:

- Conexión pantalla controlador mediante la terminal panel(J4).
- Conexión controlador raspberry pi mediante terminal RPI display y puerto DSI respectivamente.
- Alimentación, se utilizan los pines 5v y GND de la placa de desarrollo para alimentar al conjunto controlador pantalla, la conexión se realiza entre las interfaces GPIO.

La figura 4.2 muestra los puertos de conexión sobre el controlador de la pantalla.

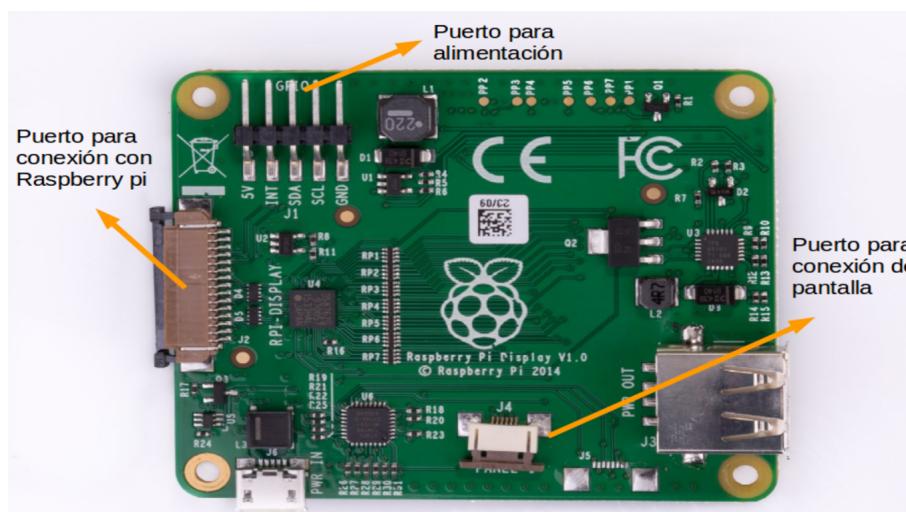


FIGURA 4.2: Puertos para conexión controlador, pantalla, raspberry pi.

Luego de realizadas las conexiones mencionadas, se energiza todo el sistema y se observa la ejecución del sistema operativo sin que sea necesaria ninguna instalación adicional.

En caso que no se pueda reconocer la pantalla, suele ser necesaria una actualización del sistema operativo.

La pantalla touch reemplaza el uso del mouse, pero, sigue dependiendo de un teclado externo. Por tal motivo y para dar mayor autonomía y sacarle mas provecho a la pantalla, se instala un teclado virtual mediante la ejecución del comando sudo apt-get install matchbox-keyboard a través del terminal.

La figura 4.3 muestra el funcionamiento del teclado virtual sobre el conjunto de elementos raspberry pi, controlador, pantalla touch.

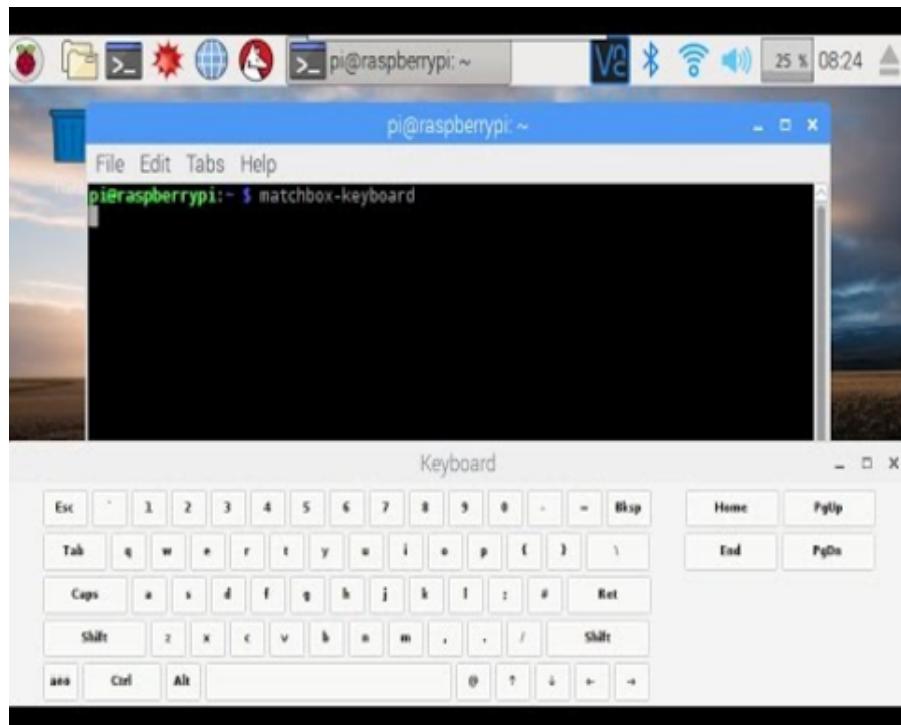


FIGURA 4.3: Teclado virtual para raspberry pi y pantalla touch.

4.1.3. Pruebas sobre el módulo sensor de huella dactilar

En cuestión de hardware el módulo sensor de huella es un solo bloque constitutivo y encapsulado, por tanto, para probar su funcionamiento se aplica voltaje, según especificaciones técnicas, a sus terminales de alimentación.

Luego de aplicar alimentación al sensor, la iluminación mediante luz led debe hacerse presente lo que indica en primera instancia que el módulo esta listo para recibir comandos.

Las pruebas mas rigurosas para este dispositivo se realizan mediante software.

La figura 4.4 muestra la respuesta del sensor luego de ser energizado.

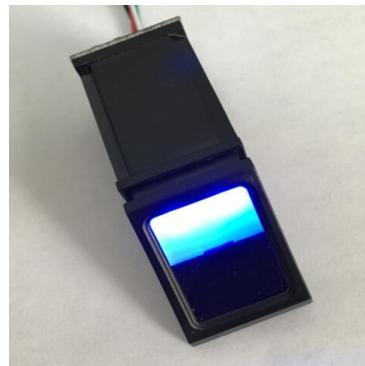


FIGURA 4.4: Módulo lector de huellas.

4.2. Pruebas funcionales de firmware

Con respecto al firmware se realizaron pruebas individuales de cada módulo.

4.2.1. Pruebas de comunicación con el módulo sensor de huella

El proceso de comunicación con el sensor consta de dos partes fundamentales, la construcción y envío de trama desde el ordenador y el recibimiento de trama de respuesta desde el sensor.

En base a los procesos mencionados se centra la construcción de la biblioteca que maneja el sensor bajo las diferentes modalidades del sistema.

La figura 4.5 muestra el resultado de envío y recepción para la instrucción “adquirir imagen” implementada con la función “getImage()” de la librería diseñada.



FIGURA 4.5: Resultado de envío y recepción de tramas pc-sensor.

Análisis de la trama de envío:

La tabla 4.1 muestra el significado de cada byte enviado.

TABLA 4.1: Trama enviada hacia el módulo sensor de huella

No	Contenido	Descripción
1	0xef	Primer byte para inicio de comunicación.
2	0x01	Segundo byte para inicio de comunicación.
3	0xff	Primer byte de dirección.
4	0xff	Segundo byte de dirección.
5	0xff	Tercer byte de dirección.
6	0xff	Cuarto byte de dirección.
7	0x01	Byte que indica que el contenido de la trama lleva una instrucción.
8	0x00	Primer byte para indicar el tamaño de los datos a enviar.
9	0x03	Segundo byte para indicar el tamaño de los datos a enviar.
10	0x01	Instrucción para que el sensor adquiera una imagen.
11	0x00	Primer byte de check sum.
12	0x05	Segundo byte de check sum.

Análisis de la trama recibida:

La tabla 4.2 muestra el significado de cada byte recibido.

TABLA 4.2: Trama recibida desde el módulo sensor de huella

No	Contenido	Descripción
1	0xef	Primer byte para inicio de comunicación.
2	0x01	Segundo byte para inicio de comunicación.
3	0xff	Primer byte de dirección.
4	0xff	Segundo byte de dirección.
5	0xff	Tercer byte de dirección.
6	0xff	Cuarto byte de dirección.
7	0x07	Byte que indica que el contenido de la trama lleva una respuesta.
8	0x00	Primer byte para indicar el tamaño de los datos a enviar.
9	0x03	Segundo byte para indicar el tamaño de los datos a enviar.
10	0x02	Respuesta del sensor indicando que no hay imagen de huella disponible.
11	0x00	Primer byte de check sum.
12	0x0c	Segundo byte de check sum.

4.2.2. Pruebas para la interfaz gráfica

A continuación se muestran los diseños finales de las páginas que conforman la interfaz desarrollada.

La figura 4.6 muestra el diseño de la página principal.

La figura 4.7 muestra el diseño de la segunda página del interfaz correspondiente al acceso mediante reconocimiento de huella.

La figura 4.8 muestra el diseño de la página correspondiente al acceso mediante contraseña.



FIGURA 4.6: Página de inicio para la interfaz gráfica.

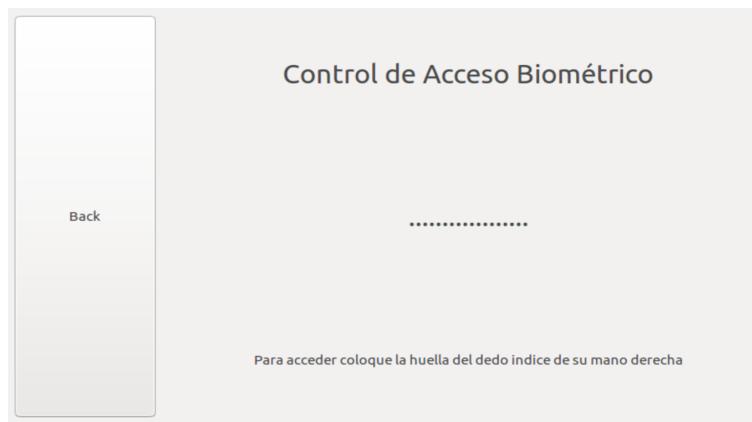


FIGURA 4.7: Página para acceso mediante reconocimiento de huella.



FIGURA 4.8: Página para acceso mediante ingreso de contraseña.

La última página corresponde al menú de configuraciones la cual esta subdividida en tres secciones, las que permitirán a futuro, ingresar un nuevo usuario, borrar un usuario existente y visualizar información importante respectivamente, la figura 4.9 muestra el diseño de las sub secciones.

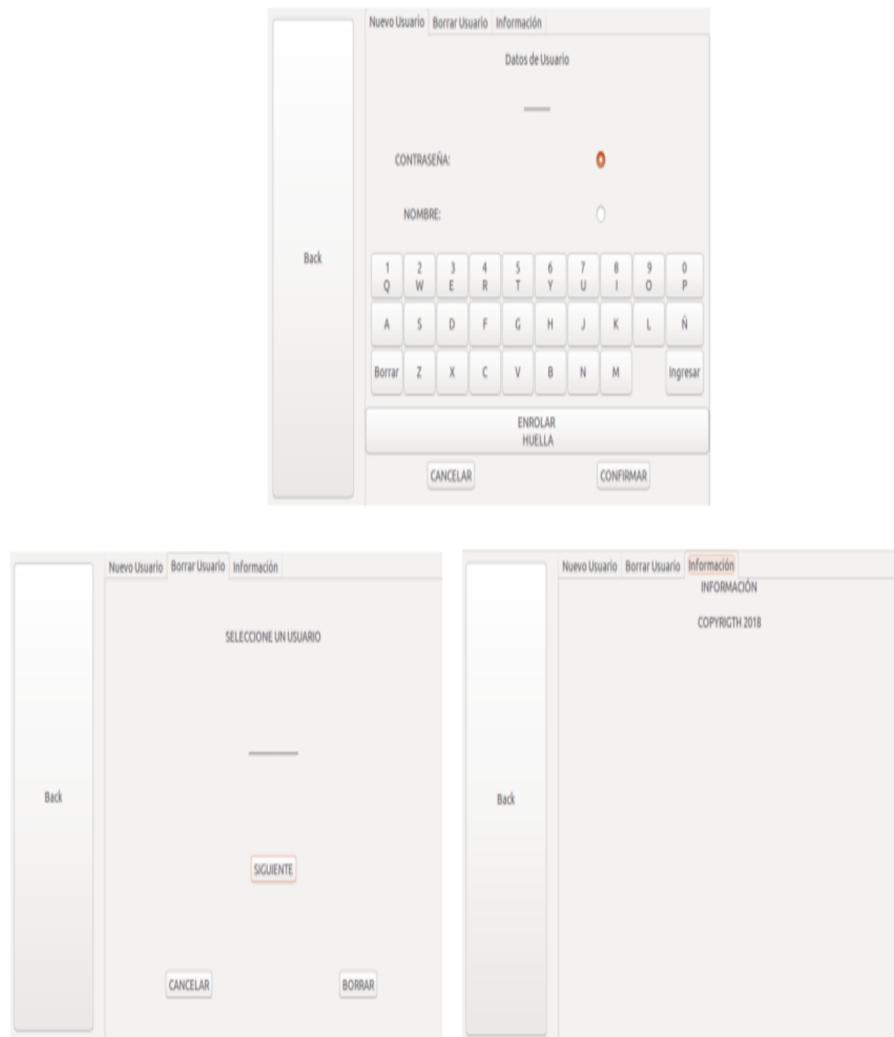


FIGURA 4.9: Página para configuraciones.

4.2.3. Pruebas modelo cliente servidor

El cliente y servidor son servicios separados con diferentes módulos constitutivos que interactúan en una comunicación local.

Cada servicio es compilado mediante la implementación de un archivo makefile y ejecutado mediante un ejecutable.

Para el caso del servidor, la carpeta src contiene los archivos.c y cabeceras.h de los módulos para manejo de base de datos y de ficheros; el archivo de compilación makefile y el ejecutable servidor.

Para el cliente, la carpeta src contiene los archivos.c y cabeceras.h para el manejo de los módulos de interfaz gráfica, sensor de huella y periféricos; la carpeta glade contiene los archivos.glade para la construcción de la interfaz; la carpeta res contiene los gráficos para la interfaz; el archivo de compilación makefile y el ejecutable cliente.

La figura 4.10 muestra la estructura de archivos implementada para los servicios.

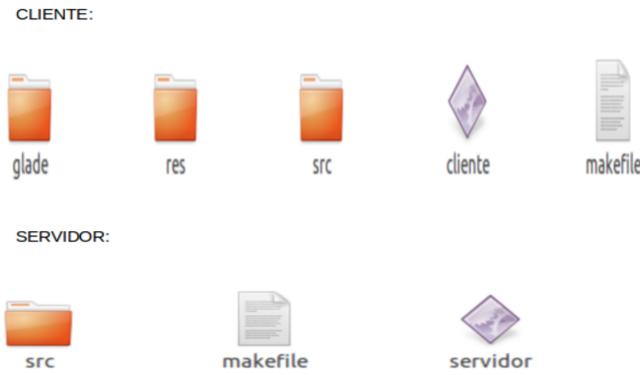


FIGURA 4.10: Estructura de archivos para los servicios cliente y servidor.

Para poner en funcionamiento el sistema se debe en primer lugar iniciar el servidor y posteriormente el cliente.

Tras la configuración respectiva para la comunicación, se realizan las pruebas enviando un paquete de datos entre ambos servicios, se especifica el número de bytes recibidos y enviados.

Los resultados obtenidos se muestran en la figura 4.11.

```
christian@home:~/Escritorio/s_c_file/servidor
christian@home:~/Escritorio/s_c_file/servidor$ ./servidor
server: conexión desde: 127.0.0.1
Recibi 20 bytes
llega evento
[]

christian@home:~/Escritorio/s_c_file/cliente
christian@home:~/Escritorio/s_c_file/cliente$ ./cliente
Conectado
enviando paquete
El tamaño es 20 bytes
Thread fingerprint completed...
```

FIGURA 4.11: Envío y recepción de mensajes cliente servidor.

4.2.4. Pruebas sobre el módulo para gestión de base de datos

Para probar la biblioteca diseñada se implementa un pequeño script con el cual se abre una base de datos con nombre base users.db. Dentro de esta se crea una tabla nombrada USER con los campos ID, NOMBRE, PASSWORD y DATA; se ingresan cuatro usuarios y al final se lee la base resultante.

La figura 4.12 muestra el resultado obtenido.

Otra prueba implementada consistió en borrar un usuario. De la base establecida, se borra el usuario con ID=3.

La figura 4.13 muestra el resultado obtenido al leer la base luego de borrado el usuario.

```

christian@home:~/Escritorio/s_c_file/db$ ./a.out
Opened database successfully
Records created successfully
Opened database successfully
Funcion llamada: ID = 1
NOMBRE = Mark
PASSWORD = 1234
DATA = 1

Funcion llamada: ID = 2
NOMBRE = Esteban
PASSWORD = 1111
DATA = 2

Funcion llamada: ID = 3
NOMBRE = Marcelo
PASSWORD = 2222
DATA = 3

Funcion llamada: ID = 4
NOMBRE = Juliana
PASSWORD = 3333
DATA = 4

Operation done successfully

```

FIGURA 4.12: Lectura de la base de datos con cuatro usuarios registrados.

```

christian@home:~/Escritorio/s_c_file/db$ gcc dbservice.c -l sqlite3
christian@home:~/Escritorio/s_c_file/db$ ./a.out
Opened database successfully
Funcion llamada: ID = 1
NOMBRE = Mark
PASSWORD = 1234
DATA = 1

Funcion llamada: ID = 2
NOMBRE = Esteban
PASSWORD = 1111
DATA = 2

Funcion llamada: ID = 4
NOMBRE = Juliana
PASSWORD = 3333
DATA = 4

Operation done successfully
christian@home:~/Escritorio/s_c_file/db$ 

```

FIGURA 4.13: Lectura de la base de datos luego de borrar un usuario.

Finalmente, para garantizar que la base de datos fue creada correctamente y podrá ser gestionada por otros sistemas, se utiliza la herramienta DB Browser for SQLite para visualizar el contenido de la misma.

La figura 4.14 muestra el resultado obtenido con la herramienta.

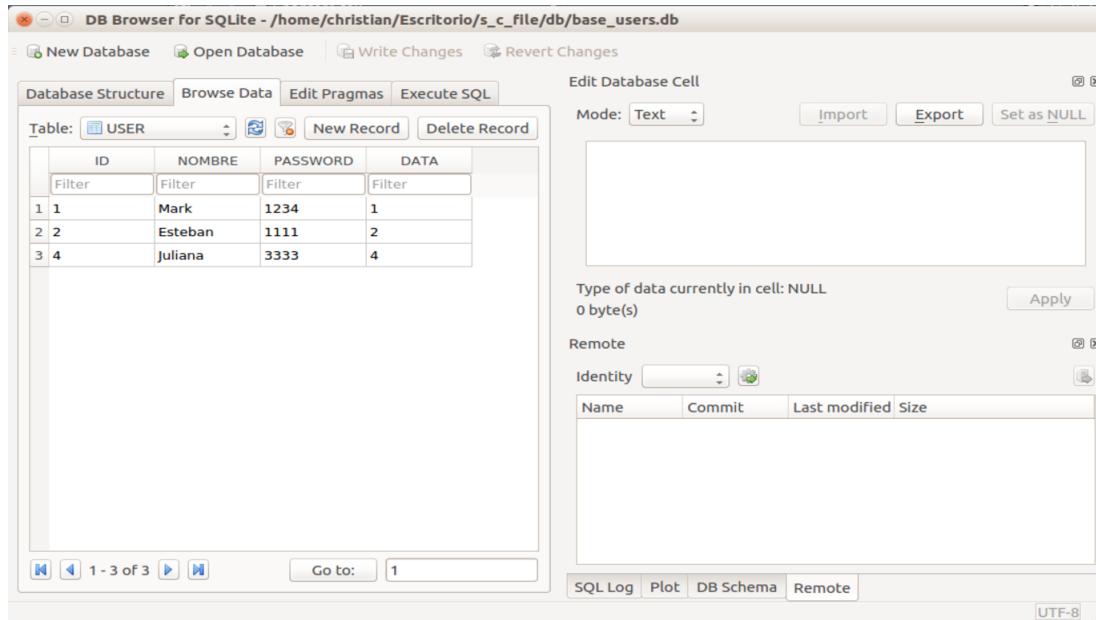


FIGURA 4.14: Lectura de la base de datos a través de la herramienta DB Browser .

4.2.5. Pruebas sobre el servidor web

Con el fin de probar el servidor web local, se diseña un script en PHP el cual lee un archivo de texto de su directorio y lo muestra sobre el navegador a través de la dirección de localhost.

El contenido del archivo de texto es un historial con el listado de usuarios y horas de registro.

El resultado obtenido se muestra en la figura 4.15.

The screenshot shows a web browser window with the URL 'localhost/info.php' in the address bar. The page content includes the logo of the Faculty of Engineering at the University of Buenos Aires, followed by the text 'FACULTAD DE INGENIERIA Universidad de Buenos Aires'. Below this, the heading 'Registro de acceso' is displayed. Underneath it, the text 'Usuarios registrados últimamente:' is shown, followed by a list of registered users with their last login times:

- Christian.....12:23:34
- Marcelo.....13:08:23
- Sebastian.....08:52:04
- Mario.....11:29:45
- Carolina.....05:15:01
- Rebeca.....22:20:12

FIGURA 4.15: Lectura de historia de accesos mediante el servidor web.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

El prototipo desarrollado corresponde a un sistema capaz de realizar el reconocimiento de usuarios previamente registrado, mediante la verificación del patrón característico de su huella digital o mediante la utilización de una clave numérica. El sistema permite crear una base de datos con los usuarios y visualizar un historial de reconocimientos.

Durante el desarrollo de este trabajo se aplicaron los conocimientos adquiridos a lo largo de la carrera logrando de esta forma alcanzar los siguientes objetivos:

- Desarrollar sistemas embebidos sobre plataformas con microcontroladores de 32 bits y el uso de periféricos.
- Utilizar buenas prácticas de programación sobre el lenguaje C.
- Gestionar proyectos mediante la elaboración de planes, calendarios, herramientas y metodologías de ingeniería.
- Utilizar Linux como sistema operativo para sistemas embebidos.
- Aplicar criterios sobre protocolos de comunicación para elaborar bibliotecas modulares para la abstracción de hardware.
- Iniciar en el desarrollo de sistemas embebidos con el uso de herramientas de software libre.
- Diseñar e implementar interfaces gráficas para usuarios.
- Iniciar en el desarrollo web para registro de acontecimientos sobre sistemas distribuidos.

5.2. Próximos pasos

Ciente de las potencialidades del sistema y para dar continuidad al esfuerzo realizado, se listan a continuación las principales líneas para un trabajo futuro apuntando al desarrollo de un producto comercialmente atractivo.

- Mejorar la interfaz web mediante el uso de formularios para mejorar la gestión de usuarios.

- Implementar toda la funcionalidad del interfaz gráfico en la sección de configuraciones y mejorar todo el conjunto.
- Ampliar la funcionalidad del sistema para cubrir un mayor número de sensores.
- Implementar todas las funcionalidades para la librería del sensor de huella.
- Aplicar el sistema en un proyecto de seguridad real.
- Implementar herramientas para testing de software.
- Implementar herramientas y métodos para seguridad de datos de usuario.
- Reemplazar el modelo cliente servidor local por el modelo global a fin que el cada parte del sistema pueda ser implementado bajo cualquier red y en cualquier lugar geográfico.