UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE TECNOLOGIA

CARRERA ELECTRONICA Y TELECOMUNICACIONES

“DISEÑO DE UN PROTOTIPO BIOMÉTRICO CON RECONOCIMIENTO FACIAL Y DE HUELLA PARA CONTROL DEL PERSONAL DEL MINISTERIO DE HIDROCARBUROS”

PROYECTO DE GRADO PRESENTADO PARA OBTENER EL

GRADO DE LICENCIATURA

POR: XIMENA CHAMBI CONDORI

TUTOR: LIC. JULIA TORREZ SORIA

LA PAZ - BOLIVIA

GESTIÓN 2021

DEDICATORIA

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme el haber llegado hasta este momento tan importante de mi formación profesional. A mis padres, por ser el pilar más importante y por demostrarme siempre su cariño y apoyo incondicional sin importar nuestras diferencias de opiniones.

A mi hermana mayor Teo, a quien quiero como a una madre, por compartir momentos significativos conmigo y por siempre estar dispuesta a escucharme y ayudarme en cualquier momento.

A Rosy, porque te amo infinitamente hermanita. A Andrés porque sin el equipo que formamos, no hubiera logrado esta meta.

AGRADECIMIENTO

Me van a faltar páginas para agradecer a las personas que se han involucrado en la realización de este trabajo, sin embargo, merecen reconocimiento especial mi Madre y mi Padre que con su esfuerzo y dedicación me ayudaron a culminar mi carrera universitaria y me dieron el apoyo suficiente para no decaer cuando todo parecía complicado e imposible.

Asimismo, agradezco infinitamente a mis Hermanos as que con sus palabras me hacían sentir orgullosa de lo que soy, y de lo que les puedo enseñar.

De igual forma, agradezco a mi tutora Lic. Julia Torrez, que gracias a sus consejos y correcciones hoy puedo culminar este trabajo. A los docentes que me han visto crecer como persona, y gracias a sus conocimientos hoy puedo sentirme dichosa y contenta.

RESUMEN

El siguiente proyecto consiste en diseñar e implementar una solución a partir de un sistema biométrico utilizando software libre como hardware libre para el ingreso de personal del Ministerio de Hidrocarburos.

El reto se ha consistido en crear un sistema que, basándose en la nueva tecnología de programación y de hardware que pueda registrar el control de ingreso de todo el personal de la institución. El proyecto se divide en dos secciones: En una primera, se ponen en claro las necesidades que el sistema biométrico puede satisfacer y ofrecer con sus ventajas. Además, se estudia el mercado de este tipo de sistema y se especifica el tipo de sistema en el que se va a centrar el proyecto.

Posteriormente y como resultado de la fase inicial se implementa una solución técnica que complemente con las conclusiones obtenidas en el estudio inicial. Ésta sección consiste en el diseño puramente conceptual del sistema, teniendo claras las especificaciones que se debe cumplir para tomar las decisiones oportunas en la elección del protocolo, los componentes y los servicios que integrará el sistema biométrico.

La solución técnica consiste en un sistema biométrico que tiene como factor de lectura y detección para cada persona de forma dactilar y de forma facial, para el ingreso y salida dentro de la institución, también como beneficios tiene un sistema de administración en el cual utilizará el personal de RRHH para los descuentos y atrasos. Conjuntamente a las dos secciones principales ha sido precisa la inclusión de distintos anexos para que la solución técnica sea completa. En los anexos se encuentra un completo “tutorial” de configuración y programación del sistema biométrico y supervisión vía web. Gracias a este tutorial, el usuario podrá configurar cada uno de los dispositivos que integran el sistema.

INDICE

CAPITULO II

CAPITULO III

INDICE DE FIGURAS

CAPITULO I

CAPITULO II

CAPITULO III

INDICE DE TABLAS

CAPITULO IV

CAPÍTULO I

INTRODUCCIÓN

1.1 GENERALIDADES

En el Ministerio de Hidrocarburos existe un área muy vulnerable que se deja de lado, esta es el control de ingreso de personal, los cuales no tienen un sistema de control que se pueda modificar por la razón de que el sistema del equipo actual es cerrado, eso quiere decir que no podemos realizar ninguna modificación en su código fuente y en la parte de hardware. El presente proyecto plantea implementar un equipo que tenga dos entradas (biométrico, facial-dactilar) que marque el ingreso y salida de personal y tener un software propio de la institución en el cual se le pueda modificar y versionar los cambios que se realicen en el prototipo.

Es así que se implementara un equipo biométrico en un punto estratégico (puerta principal del ministerio) y como es un prototipo solo se realizarán una unidad de este equipo, también se realizara un software que tenga los atributos que el personal de RRHH necesarios y con los lenguajes y arquitectura de nuevos, es decir se realizara una página web para el control de asistencia de todo el personal y obtener los reportes necesarios que se guardará en una base de datos de un servidor, sobre la seguridad, de la parte del hardware se le piensa realizar una estructura metálica en el cual sea inaccesible de daños a la estructura en sí, en la parte de software y comunicación de los distintos micro servicios se realizarán con autenticación de token y middleware.

1.2 ANTECEDENTES

Una de las empresas conocida en Bolivia es Onda Sat que surge como un emprendimiento empresarial en el año 2011 con el objetivo de cubrir las necesidades de la población en el área de seguridad ciudadana.

En sus inicios, Onda Sat se enfocó al mercado de los equipos de seguridad personal, familiar y empresarial. Con el auge de las tecnologías, la empresa creció en la implementación de sistemas y productos de seguridad innovadores, que permitan al cliente sentirse seguro en cualquier lugar y a toda hora.

Actualmente, esta empresa busca la innovación y actualización constante en la adquisición y desarrollo de tecnologías, no solamente en equipos de seguridad; sino también soluciones empresariales en supervisión, redes y comunicación, sistemas de supervisión y monitoreo empresarial para una mejor optimización en los procesos productivos y mayor rendimiento laboral.

También tenemos como para resaltar instituciones públicas en el Estado de Bolivia que diseñan nuevas innovaciones como ser este dispositivo hecho en Bolivia, ése es el sello que distingue al sistema de registro biométrico que fue desarrollado en la Agencia de Gobierno Electrónico y Tecnologías de Información y Comunicación (Agetic). El dispositivo fue elaborado con base en software libre.

"Es el prototipo de diseño de registro biométrico que se ha desarrollado en el área de investigación, que pertenece a la unidad de innovación, investigación y desarrollo de la Agetic. En esta área se concentran en el plan de soberanía científica y tecnológica", informo el ingeniero electrónico Daniel Jiménez, quien es integrante del equipo que fabricó este dispositivo de seguridad.

El área de Innovación e Investigación de la Agetic estudia la factibilidad de nuevas alternativas de software y hardware libre. La unidad se formó en febrero y el sistema de registro biométrico se comenzó a producir en junio.

En la actualidad ya operan dos equipos en instalaciones de la Agetic, los cuales, en líneas generales, determinan los horarios del personal, calculan los minutos de atraso y/o compensación o faltas, entre otras funciones.

Figura 1.1 El sistema biométrico funciona en sus instalaciones
FUENTE:
[https://www.paginasiete.bo/u/fotografias/fotosnoticias/2016/11/19/135776.jpg]

*"**Hemos tratado de emular lo que se encuentra en el comercio, en el mercado. Hay equipos similares que reconocen rostro y huella, nuestro objetivo era emularlos, pero el inconveniente es que esos equipos no tenían compatibilidad con software libre, con Linux**",* explica Arturo Hernández.

El aparato está conformado por un pequeño ordenador de bajo costo, llamado Raspberry Pi. Éste detenta un procesador ARM, similar al de la mayoría de los celulares, pero está orientado a la enseñanza y el desarrollo.

Este microordenador -como toda una computadora- tiene puertos USB, salida de audio y video y un puerto de red que permite conectar el dispositivo a internet. El aparato utiliza el sistema operativo GNU/Linux, denominado Raspbian, sobre el cual se ha desarrollado un pequeño programa que procesa la información enviada por un sensor biométrico de huellas dactilares y una cámara de cinco megapíxeles.

La información generada se guarda en una base de datos en el mismo microordenador. El sistema tiene la capacidad de registrar hasta 235 huellas dactilares.

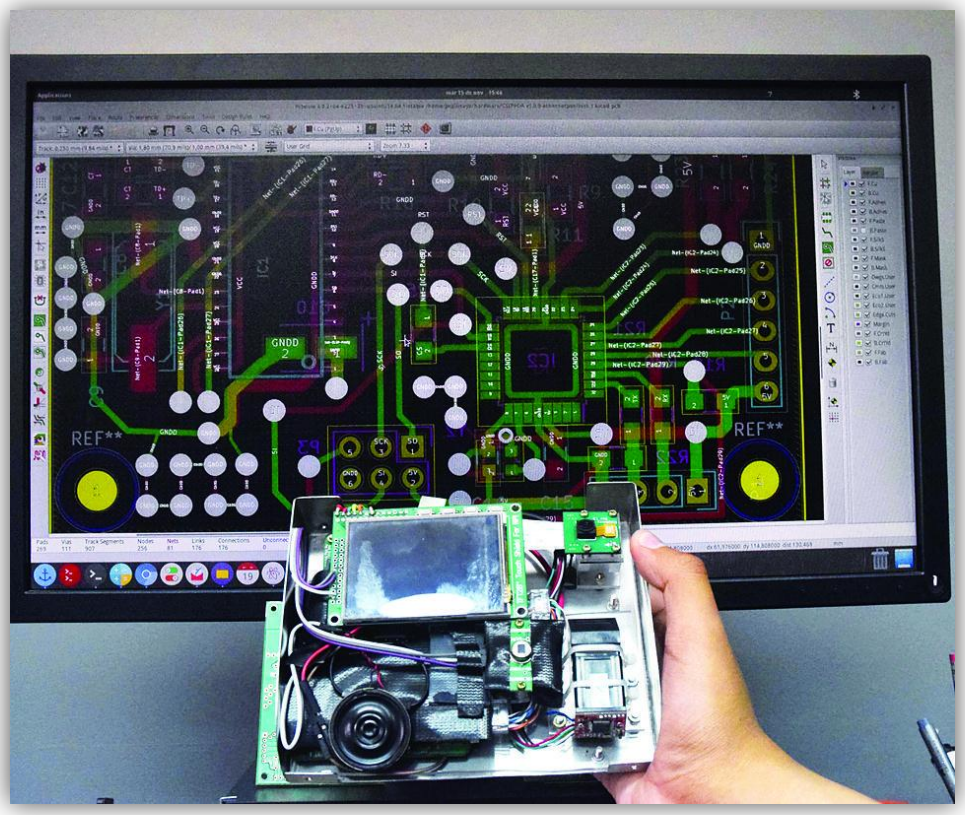


Figura. 1.2 Un prototipo del sistema de registro biométrico
FUENTE:
[https://www.paginasiete.bo/u/fotografias/fotosnoticias/2016/11/19/135777.jpg]

1.2.1. Hardware y Software Libre.

El equipo trabaja con software libre, que en palabras de Carlos Remuzzi resulta indispensable. ***"Fortifica la libertad del usuario en general, además alimenta la cultura del conocimiento; eso es importante y entra en el marco de la soberanía tecnológica del país",*** concluye.

El plan de soberanía científica y tecnológica guía la búsqueda del Estado de una independencia tecnológica. ***"Bolivia tiene que ser un país innovador y***

creativo, tenemos que dejar de tener dependencia tecnológica y a raíz de eso tratamos de construir prototipos diferentes; uno de éstos es el registro biométrico", manifiesta orgulloso Daniel Jiménez.

Además, este especialista expresa que a medida que se fabrique este tipo de sistemas en proporciones mayores, se abaratan los costos, con lo cual se pude dejar de importar este tipo de tecnología.

Ariel Condori menciona la premisa del director de la institución, Nicolás Laguna, quien tiene la mentalidad de realizar sus propios diseños electrónicos para salir de la dependencia electrónica.

1.3. PLANTEAMIENTO DEL PROBLEMA

1.3.1. Identificación del Problema

El problema consiste que dentro de la institución, actualmente se está utilizando un sistema biométrico dactilar antiguo que en la actualidad presenta fallos en el registro del ingreso y salida del personal, también siendo un registro dactilar, todo el personal de la institución tiene contacto físico con el biométrico el cual resulta peligroso para contagiarse del covid 19.

Por tal razón se está diseñando este dispositivo para la mejora de marcado de ingreso del personal del Ministerio de Hidrocarburos con los dispositivos ya mencionados, acceso a los datos de marcado de cada personal de forma transparente para que así no haya reclamo al momento de descuentos.

1.3.2. Formulación del Problema

Al momento de controlar la asistencia del personal la información no es confiable, y además es lenta, ya que el registro se realiza manualmente y a la hora

de contabilizar las horas se debe trasladar las mismas a una hoja de cálculo para efectuar el conteo de las mismas y esto causa aumento de trabajo para el encargado de Recursos Humanos.

También existe información voluminosa, se debe a que hay un excesivo uso de papel a la hora de registrar las horas de llegada y salida del personal, pero también a la hora de imprimir las hojas de cálculo realizadas para las planillas de sueldos, planillas impositivas y planillas patronales.

La información no es garantizada al momento de elaborar el registro de horas acumuladas del personal, ya que se efectúa manualmente en una hoja de cálculo y esto puede ocasionar equivocaciones, retrasos laborales, retrasos en los sueldos del personal, procesos morosos y tediosos e incremento de trabajo y documentación e información.

No existe una Base de Datos que registre y haga un seguimiento del personal, esto ocasiona que la información que se maneja no sea nada confiable ni oportuna, a la hora de realizar el control de asistencia. No existe un control de ingreso, salida, antigüedad y descuentos del personal automatizado, esto aumenta el trabajo tanto para coordinación general, como para el encargado de Recursos Humanos.

1.4. OBJETIVOS

1.4.1. Objetivo General

Diseñar un prototipo biométrico dactilar y facial para el control de persona del Ministerio de Hidrocarburos.

1.4.2. Objetivos Específicos

- Diseñar una herramienta automatizada que controle la fecha y hora que

ingreso el personal a trabajar en la Institución a través de su huella digital y/ o reconocimiento facial, de forma que la información sea segura y confiable.

- Garantizar los diferentes procesos y movimientos de la seguridad de la información ofreciendo un control adecuado, a través del manejo de información de forma sencilla, rápida, funcional y correcta para tener un buen control del personal.

- Controlar y elaborar planillas de sueldos y descuentos de manera sistematizada con el cálculo de todos los beneficios y descuentos de cada empleado, permitiendo emitir la misma como respaldo para el Ministerio de Trabajo.

- Diseñar e implementar una interfaz de usuario sencilla y atractiva, desarrollando un módulo de seguridad del sistema para que sólo el encargado de Recursos Humanos pueda manipular los datos.

- Diseñar el Hardware en base a dispositivos de placas y software libres.

1.5 JUSTIFICACIÓN

1.5.1. Justificación Técnica

Para el presente Proyecto de Grado se contempla utilizar la tecnología necesaria, que esté al alcance de las necesidades del Ministerio de Hidrocarburos, de esta manera poder agilizar los procesos de asistencia, planillas, boletas y controlar los permisos del personal, optimizando el manejo de información a través de un sistema desarrollado con la ayuda de lenguajes de programación orientada a objetos, que permitan la aplicación práctica de los principios de la metodología orientada a objetos, la cual permite desarrollar sistemas más fáciles de adaptar y mantener también con una placa de Raspberry Pi y dispositivos que nos ayudaran a reconocer al personal.

1.5.2. Justificación Económica

La justificación económica del proyecto de grado es la de proponer un prototipo de sistema para mejorar el manejo, papeleo y procedimientos morosos de la información en procesos confiables, precisos, seguros e inmediatos logrando reducir el tiempo de elaboración de las diferentes planillas y boletas del personal, además reducirá el trabajo del Departamento de Recursos Humanos y coordinación general como también de costos en el uso excesivo de papel para las diferentes planillas.

1.5.3. Justificación Ambiental

Con el fin de ayudar a esta prestigiosa Institución, se desea diseñar un prototipo de sistema que permita gestionar el control de asistencia, planillas, boletas del personal optimizando y automatizando el proceso y se pueda llevar de una forma ordenada, segura y efectiva para beneficio principalmente del Departamento de Recursos Humanos y coordinación general, mejorando el tiempo de pago de sueldos a los empleados y agilizando el tratamiento de la información

1.6 DELIMITACIÓN

1.6.1. Delimitación Temática

El proyecto de grado se enmarcará en el análisis y diseño de un prototipo de sistema biométrico de ingreso de personal, donde uno de los puntos principales a resolver es la buena administración de los datos de los registros guardados de cada personal. Eliminar todos los problemas que sufre la administración de los registros de ingreso del personal de la institución, para mejorar el funcionamiento, mejorar el flujo de trabajo para el encargado de en RRHH, mejorando en tiempo y eficiencia.

El proyecto se desarrollará en referencia a los fundamentos físicos y estructurales de la red, ofreciendo servicio control y seguridad del mismo. Consecuentemente se verá el procedimiento de realización de pruebas en construcción, activación y los parámetros físicos que afectan a la red.

1.6.2. Delimitación Temporal

En correspondencia con los objetivos planteados, se estima que el trabajo de diseño del proyecto se desarrollará en un lapso aproximado de 6 meses donde incluye implementación y pruebas.

1.6.3. Delimitación Espacial

El diseño del proyecto se desarrollará en el ingreso de la puerta principal a las oficinas del Ministerio de Hidrocarburos que está ubicada en el 12mo piso del edificio Centro de Comunicaciones La Paz.

1.7 METODOLOGIA DE INVESTIGACION

Para el desarrollo, y el estudio del proyecto, se realizará el análisis de los circuitos digitales, protocolos de comunicaciones, lenguajes de programación. Lo que nos induce a realizar diseños analíticos así como pruebas experimentales, buscando que la parte teórica se relacione con la práctica.

Inicialmente se realizará el planteamiento del problema para realizar la formulación del proyecto. Después se llevará a cabo una indagación sobre el estado de los servicios y seguridad actual.

El proyecto presenta diferentes etapas, por lo cual se utilizaran diferentes tipos de metodologías las cuales se detallan a continuación.

El método analítico será utilizado un análisis teórico de los protocolos de comunicación que serán utilizados como interfaz para la etapa de comunicación.
El método lógico deductivo se lo empleara en los circuitos de control y el software de visualización para la interfaz, puesto que los procesos se llevaran dentro de los mismos.

El método experimental es primordial en el presente proyecto porque el sistema propuesto deberá ser compatible con tecnologías existentes, cuya eficiencia y estabilidad será puesta a prueba de forma exhaustiva experimentando el correcto funcionamiento de los diversos subsistemas que la conforman.

CAPITULO II

FUNDAMENTACIÓN TEÓRICA

2.1 INTRODUCCIÓN

La biometría no se puso en práctica en las culturas occidentales hasta finales del siglo XIX, pero era utilizada en China desde al menos el siglo XIV. Un explorador y escritor que respondía al nombre de Joao de Barros escribió que los comerciantes estampaban las impresiones y las huellas de la palma de las manos de los niños en papel con tinta. Los comerciantes hacían esto como método para distinguir entre niños y jóvenes.

En Occidente, la identificación confiaba simplemente en la “memoria fotográfica” hasta que Alphonse Bertillon, Jefe del departamento fotográfico de la Policía de Paris, desarrolló el sistema antropométrico (también conocido más tarde como Bertillonage) en 1883.

Este era el primer sistema preciso, utilizado científicamente para identificar criminales y convirtió a la biometría en un campo de estudio. Funcionaba midiendo de forma precisa ciertas longitudes y anchuras de la cabeza y del cuerpo, así como registrando marcas individuales como tatuajes y cicatrices. El sistema de Bertillon fue adoptado extensamente en occidente hasta que aparezca defectos en el sistema, principalmente problemas con métodos distintos de medidas y cambios de medidas. Posteriormente las fuerzas policiales occidentales comenzaron a usar la huella dactilar, esencialmente el mismo sistema visto en china de años antes.

En estos últimos años la biométrica ha crecido desde usar simplemente la huella dactilar, a emplear muchos métodos distintos en cuenta varias medidas fiscas y de comportamiento. Las aplicaciones de la biometría también han aumentado desde solo la identificación hasta sistemas de seguridad. La idea para usar patrones

de iris como método de identificación fue propuesta en 1936 por el oftalmólogo Frank Burch. Para los 1980 la idea ya había aparecido en películas de James Bond, pero permanecía siendo ciencia y ficción. En 1985 los doctores Leonard Flom y Aran Safir retoman la idea, su investigación y documentación les concedió una patente en 1987.

En 1989 Flom y Safir recurrieron a John Daugman para crear algoritmos para el reconocimiento de iris. Estos algoritmos patentados por Daugman en 1994 y que son propiedad de Iridian Technologies, son la base para todos los productos de reconocimiento de iris.

El reconocimiento automático de huellas dactilares comenzó aproximadamente en los años 60. Desde entonces, los sistemas automáticos de identificación de huellas se utilizan en las instituciones policiales de todo el mundo. Se dice que el primer sistema de identificación de personas fue inventado por Juan vucetich, este invento se desarrolló y se patento en argentina. Donde también se usó por primera vez para esclarecer un crimen. Con este sistema se hacía identificación de los criminales y se mantenía un control de seguridad. A continuación, se muestra la manera en que se clasificaban las huellas dactilares.

Tipos de huella dactilar

Dactilograma Natural: Esta es la que se encuentra en la yema del dedo, formado por las crestas papilares de forma natural.

Dactilograma Artificial: es el dibujo que aparece como resultado al imprimir de cierta forma con ayuda de una tinta el dactilograma natural en una superficie.

Dactilograma latente: es una huella dejada por cualquier dactilograma natural al tocar un objeto o superficie. Este dactilograma queda marcado, pero no es visible. Para obtener la visibilidad de la huella hay que aplicar un reactivo adecuado.

2.2. DEFINICION DE SISTEMAS DE SEGURIDAD

Durante esta evolución, la comunicación y el intercambio de información entre sistemas fue pasado por alto, significando que la huella dactilar recogida con un sistema no podía ser buscada o almacenada por otro sistema que tuviera igual o características parecidas. Esta problemática conllevo a la necesidad y al desarrollo de estándares de huellas digitales.

Con ello se crearon diferentes técnicas y métodos de adquisición de huellas latentes, de identificación, de comparación de patrones y de clasificación. Se implementan en los sistemas de seguridad de manera que se hacían cada vez más rigurosos y exigentes con la identificación de patrones.

Pero para que fuera exitoso, se identificó e investigó tres desafíos principales:

- Adquisición de huellas digitales.
- Extracción de las características que componen las crestas.
- Concordancia de patrones de características de las crestas.

Luego a partir de los años 90, el desarrollo de nuevos dispositivos de captura de estado sólido permitió el desarrollo de algoritmos precisos y fiables, además de su bajo coste para poder acceder a ellos fácilmente e implementarlos. Estos han contribuido a la rápida expansión de los sistemas de reconocimiento de patrones biométricos basados en las huellas dactilares.

A través del tiempo han ocurrido diversos acontecimientos que han marcado la historia de la humanidad, sentando un precedente de vulnerabilidad e inseguridad. Es así que el hombre ha visto la necesidad de crear y desarrollar diferentes sistemas de seguridad los cuales han ido evolucionando desde el más simple has el más complejo, dependiendo de la tecnología existente en el momento.

Sin embargo, los sistemas de seguridad poseen vulnerabilidades, las mismas

que han sido aprovechadas inescrupulosamente para causar daño físico, material. Es por esto y debido a los hechos ocurridos en los últimos años (ataques, virus informáticos, spywares, etc.) que se están desarrollando sistemas que brinden mayor seguridad, confiabilidad y privacidad entre otras ventajas.

El término de sistemas de seguridad posee múltiples usos. A grandes rasgos, puede afirmarse que este concepto que proviene del latin securitas hace foco en la característica de seguro es decir, realza la propiedad de algo donde no se registran peligros, daños ni riesgos. Una cosa segura es algo firme, cierto e indubitable. Puede considerarse como una certeza.

2.3. BIOMETRIA APLICADA A LA SEGURIDAD

Desde el principio de los tiempos el hombre ha sido capaz de hacer un reconocimiento visual, auditivo, táctil, de personas y objetos por sus rasgos distintos para ser identificados como miembros de un grupo, sociedad o sistema. Es así que el hombre evoluciono el reconocimiento mediante características biométricas llegando a desarrollar dispositivos capaces de realizar algunas funciones del cerebro humano de manera similar y efectiva, a través de una serie de algoritmos matemáticos, pero años de investigación han demostrado que es una tarea difícil de realizar. Sin embargo, a pesar de las dificultades encontradas, hoy existen sistemas capaces de identificar a personas por su rostro, timbre de voz, iris del ojo, huellas dactilares. Con tal versatilidad 9 que se están utilizando para mejorar los sistemas de seguridad, ya que aportan una solución efectiva al problema de la identificación.

2.3.1. Biometría

Biometría es la ciencia y la tecnología dedicada a medir y analizar datos biológicos. En el terreno de la tecnología de la información, la biometría hace referencia a las tecnologías que miden y analizan las características del cuerpo humano, como el ADN, las huellas dactilares, la retina y el iris de los ojos, los patrones faciales o de la voz y las medidas de las manos a efectos de autenticación de identidades

Este concepto se extrapola a la tecnología de la información como la *“autenticación biométrica”,* que se trata de la aplicación de algoritmos matemáticos y estadísticos para automatizar el reconocimiento de los rasgos físicos o de conducta de un individuo para así poder autenticarlo, es decir, verificar su identidad.

Estos sistemas de reconocimiento se basan en indicadores biométricos, que son esas características de las que se hablaba y que permiten realizar reconocimiento biométrico. Cualquiera que sea ese indicador, debe cumplir los siguientes requisitos:

- **Universalidad:** la característica que se use para el reconocimiento debe ser común a todas las personas.
- **Unicidad:** la probabilidad de que haya dos personas con la misma característica biométrica es muy pequeña.
- **Permanencia:** la característica en cuestión debe mantenerse con el paso del tiempo.
- **Cuantificación:** la característica debe poder medirse de forma cuantitativa.

Es La medición biométrica se ha venido estudiando desde tiempo atrás y es considerada en la actualidad como el método ideal de identificación humana

2.3.1.1 Tipos de patrones

A continuación, se describirán los diferentes tipos de patrones que pueden estar presentes en un objeto.

- patrones vectoriales: se encargan de reconocer objetos por medio de la recopilación de sus características más importantes para ser comparados con una serie de grupos que contienen diferentes descripciones.

- patrones estructurados: un ejemplo claro de este tipo de patrones, son las

huellas dactilares ya que en esta basa en el reconocimiento de estas. básicamente los descriptores son codificados mediante relaciones entre los componentes del objeto, se puede decir que los patrones que hacen que una huella dactilar sea única son los puntos anormales en las crestas de la huella.

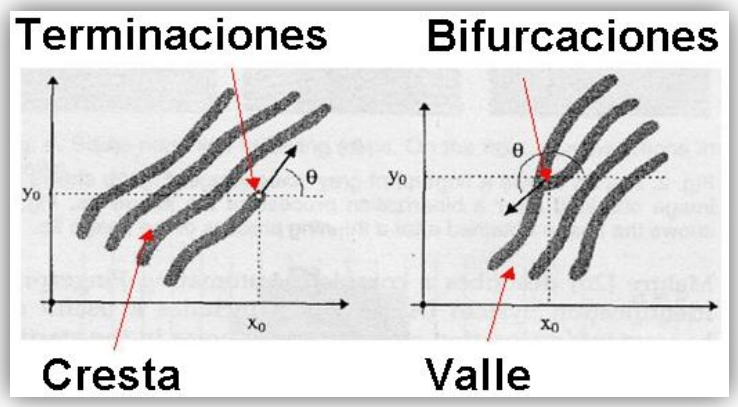


Figura. 2.1 Minucias.
FUENTE: [http://pds2006.galeon.com/Fig112.jpg]

Básicamente existen 8 puntos que caracterizan la composición de una huella digital.

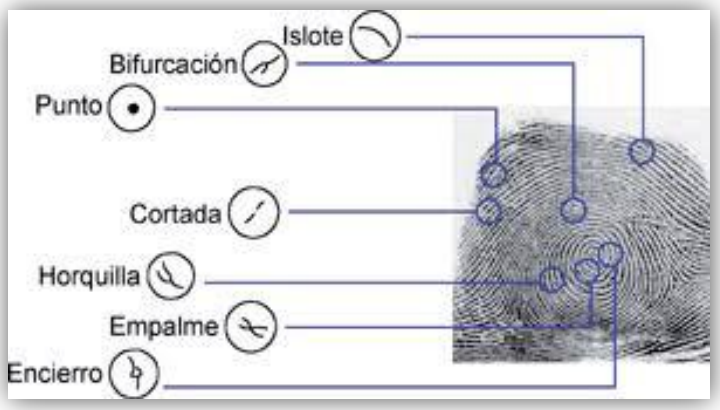


Figura. 2.2 Principales minucias.
FUENTE: [http://pds2006.galeon.com/Fig112.jpg]

Para obtener los patrones de una huella dactilar, se debe usar un sistema de reconocimiento automático de huellas dactilares, como el que se va a usar en este proyecto.

El procedimiento para la lectura de huellas dactilares es el siguiente:

Leer la huella por medio de un escáner de huellas, de acuerdo al escaneo el software de biometría crea un modelo de huella en dos dimensiones.
Luego la huella es codificada por el escáner, y es aquí donde se detectan las minucias.

Se crea una plantilla de la huella donde se identifica cada punto característico los cuales son utilizados para crear un conjunto de cadenas (como vectores), formando un mapa mediante la unión de estos puntos con rectas y esto genera un trazo de configuración única.
El software guarda y reconoce un conjunto de números. Estos son únicos para cada plantilla.

2.3.2 Clasificación de la biometría

- Estática: esta se encarga de las características físicas y/o biológicas que tiene un objeto o individuo para ser identificado.

- Dinámica: Esta se encarga de estudiar la conducta del individuo para determinar los comportamientos únicos que lo diferencian de otros individuos.

- El enfoque dentro de la biometría para este proyecto es la rama estática, ya que comprende la identificación de huellas dactilares.

2.3.3 Encriptación biométrica robusta

La autenticación robusta se basa en la combinación de al menos dos de tres principios de reconocimiento: algo que se sabe (contraseña, usuario); algo que se tiene (tarjeta, flash, dispositivo electrónico) o algo que se es (biometría).

La generación de la clave biométrica es simétrica, lo cual implica falencias en el caso de compartir la Have, ya que este hecho implicaría la posibilidad de hacerse

con la clave pública y suplantar la identidad encriptado con la clave biométrica.

Por otro lado, esto se puede subsanar mediante el principio de la autenticación robusta, que solicita que además de utilizar la clave, se aplique una contraseña el momento de la encriptación o des encriptación que solo el emisor conozca y que comparta solo con los receptores finales, los cuales la recibirán por otro medio.

Es así, que se tendrá una clave biométrica relativamente robusta frente al hecho de que se puede robar la clave, pero se requerirá a la persona como tal para desencriptarla. Entre los métodos biométricos sugeridos, se tienen la generación de clave mediante muestra de ADN digitalizado, huella dactilar y finalmente código de Iris.

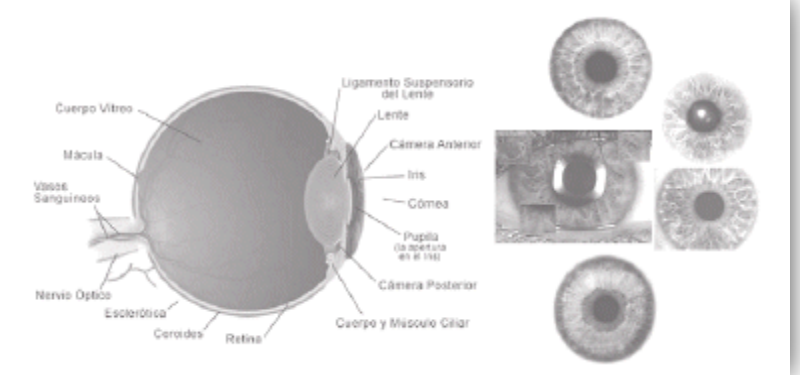


Figura. 2.3 La Iris como Elemento Biométrico
FUENTE: [https://n9.cl/k89t1]

El ADN es un depósito de información que se trasmite de generación en generación, conteniendo toda la información necesaria para construir y sostener el organismo en el que reside.

La función principal de la herencia es la especificación de las proteínas, siendo el ADN una especie de piano para las proteínas. La estructura del ADN es una pareja de largas cadenas de nucleótidos única para cada ser humano

Código de Iris: El proceso del Reconocimiento de Iris realizado por el dispositivo biométrico, consta de los pasos de captura de la imagen, localización del iris,

optimización de la imagen y clasificación de la imagen.

Huella Digital: Utilizan un escáner que hace rebotar rayos de luz en el dedo, donde un sistema procesa el patrón refractado. Esto permite al lector crear una imagen del dedo, que es trasmitida al software biométrico.

Según un prototipo, para combinar ambas técnicas y realizar así la arquitectura de un software que realiza la encriptación biométrica, para una autenticación robusta con clave pública en sistemas criptográficos avanzados como aquellos que aplican el algoritmo de encriptación AES o el 3DES. Donde: KPr: Clave privada generada por el algoritmo. KBio: Clave Biométrica generada por el dispositivo biométrico. KPu: Clave pública generada por el algoritmo:

- Generación Clave: Algoritmo de Encriptación (AES, TDES,etc.)

- KPuB: Clave pública encriptada por el dispositivo Biométrico.

Como se observa, es posible combinar técnicas de encriptación biométrica y de clave asimétrica; asimismo utilizar un algoritmo veloz de encriptación para el efecto, como lo es el algoritmo AES. La combinación de ambas técnicas de encriptación de mensajes se realizó según el diagrama en bloques de la figura anterior donde se muestra el proceso de la generación de claves, ya sean biométricas o la generación de claves pública-privada por otro medio. Es así que en el transmisor se crea la clave pública biométrica KPuB, que es generada al utilizar la clave pública KPu por un método tradicional, como el algoritmo AES; el emisor encriptara el mensaje a ser transmitido mediante la generación de la clave privada KPr. La clave pública biométrica KBio, sera difundida por otro medio y solo servirá para desencriptar la clave pública biométrica KPuB, no conocida por la entidad que produce la clave privada, ya que esta no tendría la clave biométrica.

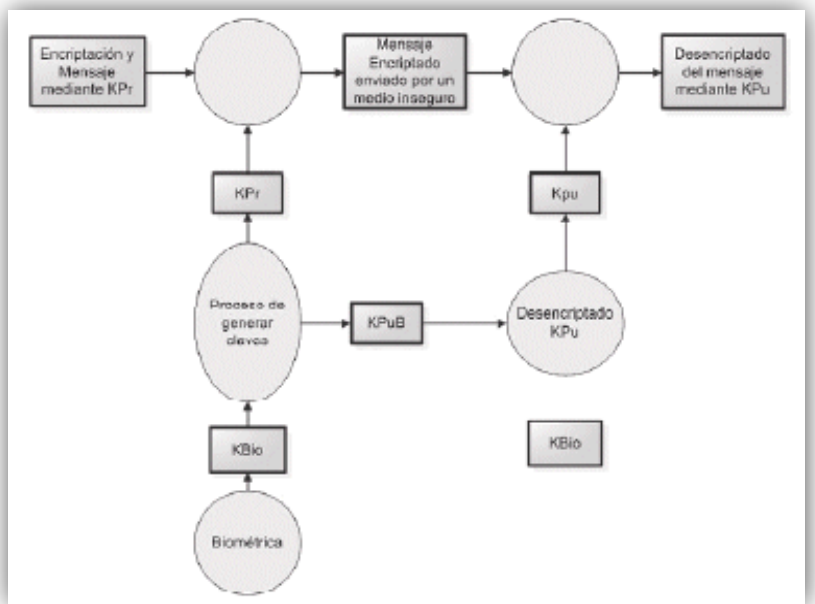


Figura. 2.4 Prototipo Encriptación Biométrica
FUENTES: [https://n9.cl/k89t1]

2.3.4 Reconocimiento de huellas dactilares

Es uno de los métodos más usados en el mundo por la facilidad que tiene para apoyar los sistemas de seguridad, ya que la autenticación de personas se puede obtener de manera eficaz. La ciencia que estudia los rasgos de las huellas es la dactiloscopia; esta se divide en cuatro grandes rasgos:

- Inmutabilidad: huellas que no son modificadas en el desarrollo físico de una persona.
- **Perennidad:** reconoce que las personas desde los seis medes tiene huellas dactilares.
- **Variedad:** huellas únicas en los individuos.
- **Clasificabilidad:** recopilación de información de bases de datos de aplicaciones con fines de control de acceso para la consulta de diferentes plantillas de huellas.

  Las crestas que conforman cada dedo es denominado dactilograma, de este se derivan tres tipos:

El primero es el natural que existe en la yema de los dedos, el segundo es el artificial que es el dibujo impreso por cada dedo y el latente es producido por el dedo al tocar una superficie.

Los dactilogramas son clasificados en diferentes tipos, pero antes de clasificarlos se debe hablar de las características de composición de las huellas dactilares.

Los dactilogramas están compuestos por tres zonas: marginal, nuclear y bacilar. Para poder identificar cada zona se debe tener en cuenta:

- Debe haber un delta en cada dactilograma.
- A partir del delta podemos identificar una directriz la cual encierra el núcleo.
- El núcleo es la parte más importante ya que en este se puede distinguir las características para la clasificación.

A continuación, se muestran los diferentes tipos de delta que pueden ser encontrados en las huellas dactilares.

Figura. 2.5 Deltas Negros
FUENTE: [http://1.bp.blogspot.com/s1600/Captura10.PNG]

2.3.5 Reconocimiento facial

El reconocimiento facial se ha convertido en los últimos años en un área de investigación activa que abarca diversas disciplinas, como procesado de imágenes, reconocimiento de patrones, visión por ordenador y redes neuronales. Involucra tanto a investigadores del área de informática como a neurocientíficos y psicólogos. Se podría considerar también dentro del campo de reconocimiento de objetos, donde la cara es un objeto tridimensional sujeto a variaciones de iluminación, pose, etc., y ha de ser identificada basada en su proyección 2D (excepto cuando se utilizan técnicas 3D).

El objetivo de un sistema de reconocimiento facial es, generalmente, el siguiente: dada una imagen de una cara "desconocida", o imagen de test, encontrar una imagen de la misma cara en un conjunto de imágenes "conocidas", o imágenes de entrenamiento. La gran dificultad añadida es la de conseguir que este proceso se pueda realizar en tiempo real. El sistema identificará las caras presentes en imágenes o videos automáticamente. Puede operar en dos modos:

Verificación o autenticación de caras: compara una imagen de la cara con otra imagen con la cara de la que queremos saber la identidad. El sistema confirmará o rechazará la identidad de la cara.
Identificación o reconocimiento de caras: compara la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en la base de datos para determinar su identidad.

Por su naturaleza amigable, este tipo de sistemas siguen siendo atractivos a pesar de la existencia de otros métodos muy fiables de identificación personal biométricos, como el análisis de huellas dactilares y el reconocimiento del iris.

Figura. 2.6 Lector de Reconocimiento Facial, lector de iris
FUENTE. [https://www.cucorent.com/blog/lector-biometrico-control-de-accesos/]

El siguiente paso dentro del lector biométrico, fue el desarrollo de la identificación por reconocimiento facial, un paso más en la tecnología, con la misma fiabilidad que el lector de huella dactilar. Tanto el lector reconocimiento facial como el lector de huellas digital reducen la posibilidad de error en el registro o de engaño a su mínima expresión.

2.3.5.1 Función de reconocimiento de rostro

El proceso consta de cuatro módulos principales:

Detección de la cara: detecta que hay una cara en la imagen, sin identificar. Si se trata de un video, también podemos hacer un seguimiento de la cara. Proporciona la localización y la escala a la que encontramos la cara.

Alineación de la cara: localiza las componentes de la cara y, mediante

transformaciones geométricas, la normaliza respecto propiedades geométricas, como el tamaño y la pose, y fotométricas, como la iluminación. Para normalizar las imágenes de caras, se pueden seguir diferentes reglas, como la distancia entre las pupilas, la posición de la nariz, o la distancia entre las comisuras de los labios. También se debe definir el tamaño de las imágenes y la gama de colores.

Normalmente, para disminuir la carga computacional del sistema, se acostumbra a utilizar imágenes pequeñas en escala de grises. A veces también se realiza una ecualización del histograma.

Extracción de características: proporciona información para distinguir entre las caras de diferentes personas según variaciones geométricas o fotométricas.
Reconocimiento: el vector de características extraído se compara con los vectores de características extraídos de las caras de la base de datos. Si encuentra uno con un porcentaje elevado de similitud, nos devuelve la identidad de la cara; si no, nos indica que es una cara desconocida.

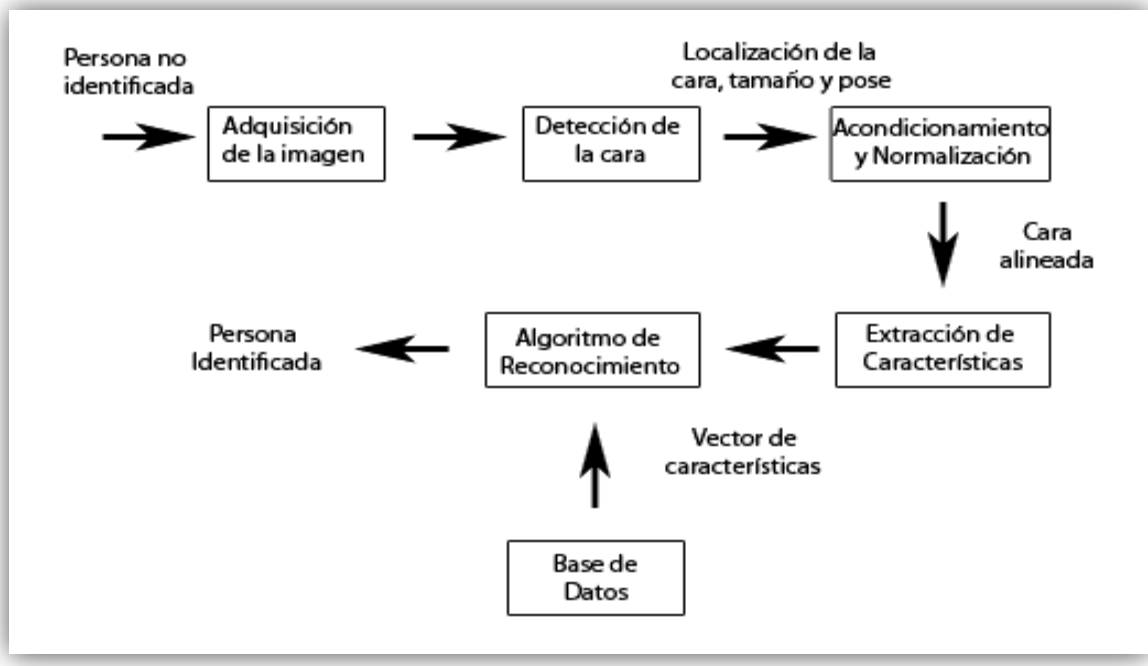


Figura. 2.7 Sistema de reconocimiento facial.
Fuente:
[https://upload.wikimedia.org/Face_Recognition_System_Workflow_castellano.gif]

Los resultados obtenidos dependen de las características extraídas para representar el patrón de la cara y de los métodos de clasificación utilizados para distinguir los rostros, pero para extraer estas características apropiadamente, hace falta localizar y normalizar la cara adecuadamente.

2.3.5.2 Técnicas y algoritmos

Reconocen según toda la imagen facial. Son métodos basados en correlación. El esquema de clasificación más simple, donde se utilizan modelos de comparación para el reconocimiento, es el template matching. El problema del template matching es que ha de comparar muchas características (para él, un pixel es una característica), y si tenemos en cuenta que en la base de datos encontramos M personas, con N imágenes por persona, observamos que este método no se puede implementar en tiempo real. Por lo tanto, se trabaja con otros métodos que de correlacionan las características entre sí para conseguir reducir el espacio facial en un número menor de coeficientes, que tengan un alto poder discriminatorio entre las personas. Es lo que se denomina subespacio facial. Ejemplos de métodos que trabajan a partir de subespacios son el Análisis de Componentes Principales (PCA - Principal Component Analysis) a partir de eigenfaces, el Análisis Linear Discriminant (LDA - Linear Discriminant Analysis) o el Discriminante Linear de Fisher (FLD - Fisher Linear Discriminant) a partir de fisherfaces.

La técnica PCA se considera una de las que proporciona un mayor rendimiento. Funciona proyectando las imágenes faciales sobre un espacio de facciones que engloba las variaciones significativas entre las imágenes faciales conocidas. Las facciones significativas se llaman eigenfaces, ya que son los eigenvectors, o componentes principales, del conjunto de caras. La proyección caracteriza la imagen facial de un individuo como la suma de los diferentes pesos de todas las facciones y, de la misma manera, para reconocer una imagen facial determinada sólo hará falta comparar estos pesos con aquellos de los individuos conocidos previamente. No tiene en cuenta la información de qué imágenes pertenecen a un mismo individuo. Es muy sensible a cambios en las condiciones de

iluminación en diferentes imágenes de una misma persona.

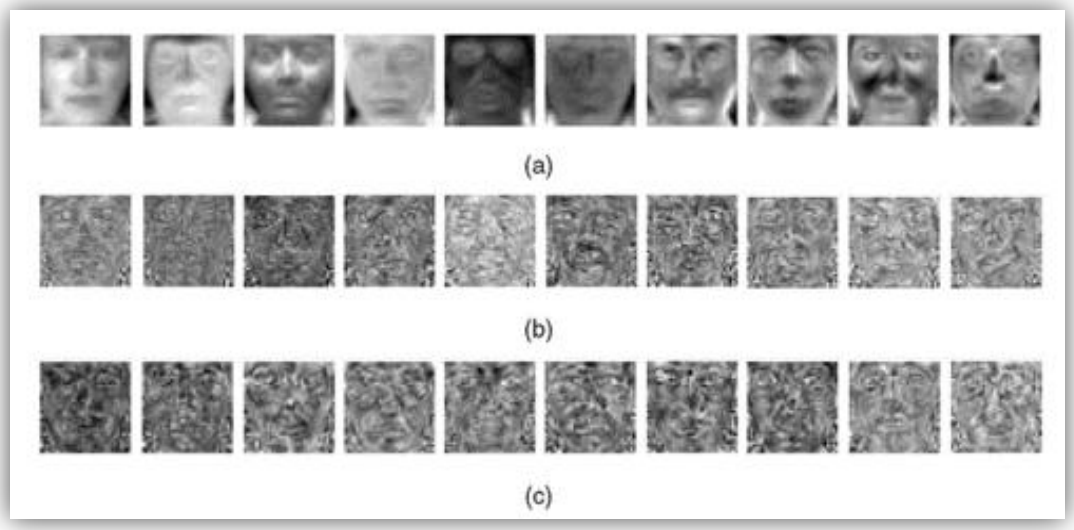


Figura. 2.8 Los primeros 10 a) Eigenfaces, b) Fisherfaces, y c) Laplacianfaces calculados a partir de imágenes de caras de la base de datos de YALE
FUENTE:
[https://upload.wikimedia.org/wikipedia/500p_eigenface_laplacianface.GIF]

El método LDA permite utilizar la información entre miembros de la misma clase (imágenes de la misma persona) para desarrollar un conjunto de vectores de características donde las variaciones entre las diferentes caras se enfatizan mientras que los cambios debidos a la iluminación, expresión facial y orientación de la cara no. Es decir, maximiza la variancia de las muestras entre clases, y la minimiza entre muestras de la misma clase.

La técnica FLD es equivalente al LDA. Los resultados obtenidos con FLD son bastante mejores que los que podemos obtener con PCA, sobre todo cuando las condiciones lumínicas varían entre el conjunto de imágenes de entrenamiento y de test, y también con cambios de expresión facial, dando más peso a zonas como los ojos, la nariz o las mejillas que a la boca, porque son zonas más invariables en las diferentes expresiones que puede tener una persona.

Otros métodos, en vez de utilizar subespacios faciales, siguen una clasificación

por redes neuronales y plantillas deformables, como EGM - Elastic graph matching.

2.3.6 Detección facial en python y opencv

En la actualidad la tecnología representa un constante crecimiento en las diversas áreas de nuestras vidas, prácticamente todo aquello que solíamos hacer mediante un grupo de personas o herramienta física, se está viendo reemplazada por un software o pieza de hardware cada vez más potente, que "facilita" de cierta forma las tareas cotidianas.

El software como bien saben es un ambiente bastante "volátil", hoy en día es más sencillo acceder a distintas herramientas que permiten "jugar y aprender" sobre temas que quizás en otro momento podrían parecer inaccesibles, y se debe a las diversas plataformas que difunden el "software libre"; estas permiten mejorar y crear nuevas herramientas para todo tipo de situaciones.

Hoy hablaremos de OpenCV, la cual, es una biblioteca libre de visión artificial que fue desarrollada en un principio por Intel y que gracias a los distintos contribuidores de código a nivel mundial, ha permitido el uso de muchas aplicaciones actuales. Desde sistemas de seguridad, hasta reconocimiento y detección de objetos. Lo más importante es que ha crecido a tal grado, que ahora contamos con esta fantástica librería en casi todas las plataformas de desarrollo posibles, además contiene más de 500 funciones nuevas que abarcan procesos de visión artificial, calibración de cámaras, visión estérea, visión robótica, entre otros.

2.3.6.1 La tecnología opencv

Primero hay que comprender que es "reconocimiento" y "detección", ambos conceptos son muy populares hoy en día cuando se habla de análisis de imágenes, pero a veces dista de lo que en realidad significa.

La "visión artificial" o "visión por computadora" es una disciplina dentro de la ciencia y la tecnología que se dedica a crear e implementar métodos para adquirir,

procesar, analizar y comprender las imágenes del mundo real (entorno análogo) con el objetivo de convertirlo en su semejante digital mediante información numérica o simbólica que las computadoras puedan comprender.

Así como el cuerpo humano ejecuta distintos procesos a nivel físicos, biológicos y químicos para comprender el mundo que nos rodea, la visión por computadora también trata de simular el mismo efecto para que estas puedan percibir y comprender imágenes, objetos, colores y profundidad.

En este tutorial vamos a abarcar de manera introductoria la "detección" facial, por lo tanto no debe confundirse con "reconocimiento". Si bien, es posible hacer un reconocimiento mediante OpenCV, el primer paso es crear un software o algoritmo que primero identifique lo que deseamos reconocer.

El reconocimiento de objetos, se diferencia porque además de haber pasado por una detección específica de la imagen, también es posible entregar información más precisa, como por ejemplo; si el rostro es de una mujer y si es así...de qué mujer se trata (un familiar, famoso, conocido, etc).

En la actualidad existen distintos algoritmos que pueden ser implementados para una detección facial, muchos de ellos un tanto complejos o específicos según el área en donde se desea emplear, estos pueden llegar a tratarse de redes neuronales, grafos, entre otros. Sin embargo vamos a comprender uno de los métodos más populares y eficientes que permiten detectar objetos en tiempo real.

2.3.6.2 Clasificador de cascada

Mejor conocido como algoritmo o clasificador Haar, fue el primer framework de detección de objetos propuesto por Paul Viola y Michael Jones en 2001 que permitía el análisis de imágenes en tiempo real, haciendo uso de una función matemática (Wavelet Haar) propuesta por Alfred Haar en 1909.

Los clasificadores haar, definen regiones rectangulares sobre una imagen en escala de grises (imagen integral) y al estar formada por un número finito de rectángulos, se puede obtener un valor escalar que consiste en sumar los pixeles de cada rectángulo, en base a una serie de clasificadores en cascada.

Cada clasificador determina si la subregión se trata del objeto buscado o no. A diferencia de otros algoritmos, este solo invierte capacidad de procesamiento a las subregiones que posiblemente representen un rostro.

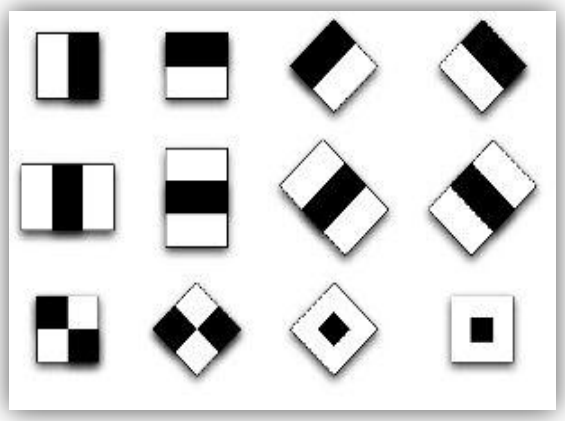


Figura. 2.9 Clasificadores HAAR

FUENTE: [http://ironsistem.com/media/images/1_fl2yzEM..width-800.jpg]

El algoritmo busca en la imagen combinaciones de estos patrones. Por ejemplo, si queremos detectar un rostro, como es en nuestro caso, el algoritmo buscará en la imagen la combinación de estos bloques que, si se juntan, se aproximan a un rostro:

Figura. 2.10 Cálculo de HAAR sobre Imagen
FUENTE: [http://ironsistem.com/media/images/2_Hst7DRc..width-800.jpg]

Este sistema tiene un porcentaje de aciertos bastante alto, aunque su éxito también dependerá del tipo de cámara utilizada, la iluminación de la sala, etc.

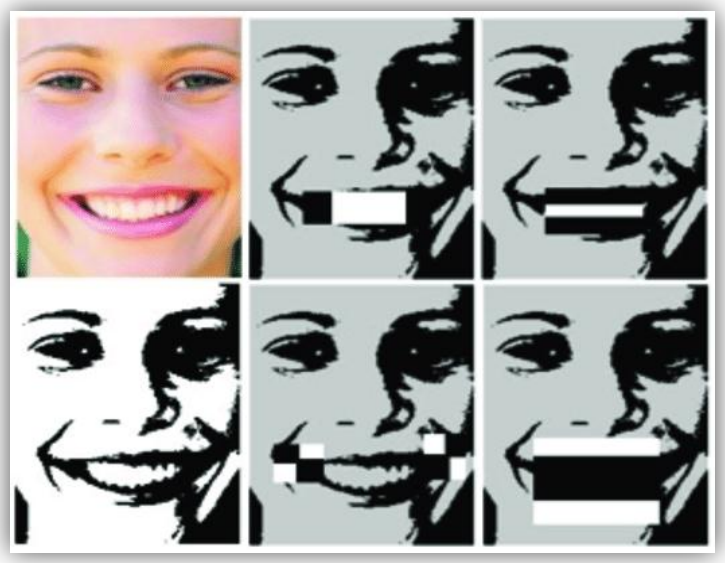


Figura.2.11 Características Haar que forman un clasificador fuerte para una sonrisa.
FUENTE: [https://www.researchgate.net/figure/Figura-2-Caracteristicas-individuales-de-Haar-que-forman-un-clasicador-fuerte-para-una_fig1_272784648]

2.4 RASPBERRY PI 2 B

2.4.1 Introducción a su historia

En 2006, los primeros diseños de Raspberry Pi se basaban en el microcontrolador Atmel ATmega644. Sus esquemas y el diseño del circuito impreso están disponibles para su descarga pública.

En mayo de 2009, la Fundación Raspberry Pi fue fundada en Caldecote, South Cambridgeshire, Reino Unido como una asociación caritativa que es regulada por la Comisión de Caridad de Inglaterra y Gales.

El administrador de la fundación, Eben Upton, se puso en contacto con un grupo de profesores, académicos y entusiastas de la informática para crear un ordenador con la intención de animar a los niños a aprender informática como lo hizo en 1981 el ordenador Acorn BBC Micro.1617. El primer prototipo basado en ARM se montó en un módulo del mismo tamaño que una memoria USB. Tenía un puerto USB en un extremo y un puerto HDMI en el otro.

Figura. 2.12 Eben Upton, fundador Raspberry Pi
FUENTE: [https://www.gov.uk/government/world-location-news/eben-upton-of-the-raspberry-pi-foundation-visits-japan]

El pre-lanzamiento en agosto de 2011, se fabricaron cincuenta placas Alpha, que tenían las mismas características que el modelo B, pero eran un poco más grandes para integrar bien unas interfaces para depuración. En algunas demostraciones se podía ver la placa ejecutando el escritorio LXDE en Debian, Quake3 a 1080p y vídeo Full HD H.264 a través de la salida HDMI.

En octubre de 2011, el logotipo se seleccionó entre varios diseños enviados por miembros de la comunidad. Durante el mismo mes, se trabajó en una versión de desarrollo de RISC OS y se hizo una demostración en público.

En diciembre de 2011, 25 placas Beta del modelo B fueron ensambladas y probadas de un total de 100 placas vacías. El diagrama de componentes de las placas finales sería el mismo que el de esas placas Beta. Durante las pruebas a las placas beta se encontró un error de diseño en los pines que suministraban alimentación a la CPU que sería arreglado en la versión final. Se hizo una demostración de la placa beta arrancando Linux, reproduciendo un tráiler de una película a 1080p y ejecutando el benchmark Rightware Samurai OpenGL ES.

Durante la primera semana de diciembre de 2011, se pusieron a subasta diez placas en eBay. Una de ellas fue comprada por una persona anónima y se donó al Centro para la Historia de la informática en Suffolk, Inglaterra. En total se consiguieron 21.269 $us. La última placa, con número de serie No. 01 se vendió por 4.557 $us.

Debido al anticipado anuncio de puesta a la venta a final de febrero de 2012, la fundación sufrió colapso en sus servidores web debido a las actualizaciones de páginas desde los navegadores de gente interesada en la compra de la placa. El primer lote de 10.000 placas se fabricó en Taiwán y China, en vez de Reino Unido. Esto fue en parte porque los impuestos de importación se pagan para los componentes individuales pero no para productos acabados, y porque los fabricantes chinos ofrecían un plazo de entrega de 4 semanas y en el Reino Unido de 12. Con este ahorro conseguido, la fundación podía invertir más dinero en investigación y

desarrollo.

Las primeras ventas comenzaron el 29 de febrero de 2012 a las 06:00 UTC; al mismo tiempo se anunció que el modelo A, que originalmente iba a tener 128 MB de RAM, tendría 256 MB. La página de la fundación también anunció que: "***Seis años después del origen del proyecto, estamos cerca de finalizar el primer arranque del proyecto - aunque esto es solo el principio de la historia de Raspberry Pi".***

Por otro lado las dos tiendas que vendían las placas, Premier Farnell y RS Components, tuvieron una gran carga en sus servidores inmediatamente después del lanzamiento. La cuenta oficial de Raspberry Pi en Twitter informó que Premier Farnell vendió toda su existencia de inventario a los pocos minutos del momento de lanzamiento, mientras que RS Components tuvo 100.000 peticiones de interés el primer día. En los seis meses siguientes llegarían a vender 500.000 unidades.

2.4.2 Hardware

Las ventas iniciales fueron del modelo B. El modelo A solo tiene un puerto USB, carece de controlador Ethernet y cuesta menos que el modelo B, el cual tiene dos puertos USB y controlador Ethernet 10/100.53 En 2014 se lanzó el modelo Raspberry Pi 2 B. El último modelo lanzado en 2015 es el Raspberry Pi Zero.

A pesar que el Modelo A no tiene un puerto RJ45, se puede conectar a una red usando un adaptador USB-Ethernet suministrado por el usuario. Por otro lado, a ambos modelos se puede conectar un adaptador Wi-Fi por USB, para tener acceso a redes inalámbricas o internet. El sistema cuenta con 256 MB de memoria RAM en su modelo A, y con 512 MB de memoria RAM en su modelo B. Como es típico en los ordenadores modernos, se pueden usar teclados y ratones con conexión USB compatible con Raspberry Pi.

El Raspberry Pi no viene con reloj en tiempo real, por lo que el sistema operativo debe usar un servidor de hora en red, o pedir al usuario la hora en el

momento de arrancar el ordenador. Sin embargo se podría añadir un reloj en tiempo real (como el DS1307) con una batería mediante el uso de la interfaz I^2C.

Los esquemas del modelo A y el modelo B fueron lanzados el 20 de abril de 2012 por la fundación. La aceleración por hardware para la codificación de vídeo (H.264) se hizo disponible el 24 de agosto de 2012, cuando se informó que la licencia permitiría su uso gratuitamente; se pensó en anunciarlo cuando se lanzara el módulo de cámara. También se puso a la venta la capacidad para poder usar el codificación-decodificación de MPEG-2 y Microsoft VC-1. Por otro lado, se hizo saber que el ordenador soportaría CEC, permitiendo que pudiera ser controlado mediante un mando a distancia de televisión.

El 5 de septiembre de 2012, se anunció una revisión 2.0 de la placa, que ofrecía un pequeño número de correcciones y mejoras, como unos agujeros de montaje, un circuito para hacer reset, soporte para depuración JTAG, etc.

El 15 de octubre de 2012, la fundación anunció que todos los Raspberry Pi Modelo B serían enviados a partir de ese momento con 512 MB de RAM en vez de 256 MB.

2.4.3 Sistemas operativos

A continuación, se muestra una lista de sistemas operativos que pueden alojarse en el RBP o están en proceso de ser lanzados:

- Sistemas operativos
  - AROS
  - Linux
    - Android99
    - Arch Linux ARM
    - Debian Whezzy Soft-Float, versión de Debian sin soporte para coma

flotante por hardware
    - DietPi, distribución ligera basada en Raspbian y de sencilla configuración mediante menús
    - Firefox OS
    - Gentoo Linux100
    - Google Chromium OS
    - Kali Linux
    - Open webOS101
    - PiBang Linux,102 distribución Linux derivada de Raspbian con diferente escritorio y aplicaciones
    - Pidora, versión Fedora Remix optimizada103
    - QtonPi, distribución linux con un framework de aplicaciones multiplataforma basado en Qt framework
    - Raspbian,104 versión de Debian Wheezy para ARMv6 con soporte para coma flotante por hardware
    - Slackware ARM, también conocida como ARMedslack
    - Ubuntu MATE
- Plan 9 from Bell Labs105 106
- RISC OS 52
- Unix
    - FreeBSD107
    - NetBSD108 109
- Windows 10
    - Windows CE

Distribuciones ligeras multipropósito:

- Minibian, distribución ligera basada en Raspbian.
- Moebius, distribución ligera ARM HF basada en Debian que usa el repositorio de Raspbian y que cabe en una tarjeta SD de 1GB, usa pocos servicios y está optimizada para usar poca memoria
- Squeezed Arm Puppy, una versión de Puppy Linux (Puppi) para ARMv6 (sap6) específicamente para Raspberry Pi110.

2.4.4 Pines GPIO

GPIO (General Purpose Input/Output) es, como su propio nombre indica, un sistema de E/S (Entrada/Salida) de propósito general, es decir, una serie de conexiones que se pueden usar como entradas o salidas para usos múltiples. Estos pines están incluidos en todos los modelos de Raspberry Pi, para que puedas realizar proyectos interesantes como lo harías con Arduino.

Los GPIO representan la interfaz entre la Raspberry Pi y el mundo exterior. Y con ello se aprovechará al máximo, desde hacer titilar un LED hasta otros mucho más sofisticados. Pero para se debe saber las características y como se debe programar. Lo primero variará en función de la revisión de placa que tengas o del modelo.

Todos los pines son de tipo “unbuffered”, es decir, no disponen de buffers de protección, así que hay que tener cuidado con las magnitudes (voltajes, intensidad…) cuando conectes componentes a ellos para no dañar la placa. Como podrás apreciar en las imágenes posteriores, no todos los pines tienen la misma función:

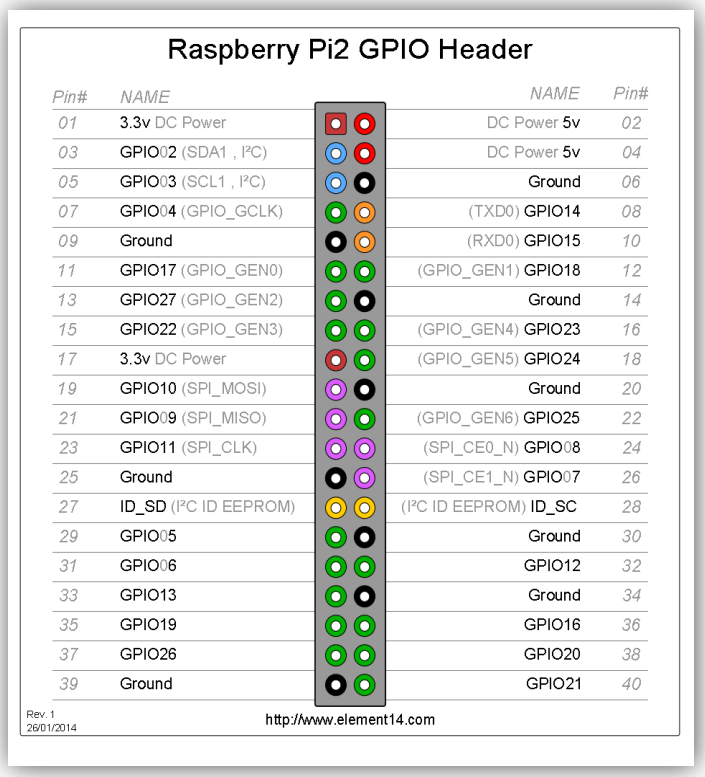


Figura. 2.13 Pines GPIO del Raspberry Pi 2
FUENTE: [https://es.pinout.xyz/]

- Pines de alimentación: se puede apreciar pines de 5v, 3v3 (limitados a 50mA) y tierra (GND o Ground), que aportan alimentación a estos voltajes para los circuitos. Pueden servir como una fuente de alimentación, aunque también puedes utilizar otras fuentes (pilas, fuentes de alimentación externas, etc). Recordemos que son unbuffered y hay que tener cuidado para no dañar la placa.
- DNC (Do Not Connect): son pines que por el momento no tienen función, pero en futuras implementaciones son utilizados para otros fines. Por eso solo se va a encontrar en modelos más primitivos de la Raspberry Pi. En las actuales placas han sido marcados como GND.

- GPIO normales: son conexiones configurables que se puede programar para el proyecto, tal como veremos más adelante.

- GPIO especiales: dentro de éstos se encuentran algunos pines destinados a una interfaz UART, con conexiones TXD y RXD que sirven para comunicaciones en serie, como por ejemplo, conectar con una placa Arduino. También podemos ver otros como SDA, SCL, MOSI, MISO, SCLK, CE0, CE1, etc.

2.4.5 Almacenamiento

La Raspberry Pi B+ no dispone de un disco duro, para ello trae un lector/ranura para memorias microSD, un sistema de almacenamiento en estado sólido. El arranque del sistema se hará desde la propia tarjeta microSD, con lo que debido a que tiene que albergar todo el sistema operativo, es necesario que la tarjeta sea de al menos 2 GB de capacidad para almacenar todos los archivos requeridos.

La placa carece de botones de encendido y apagado. Para su alimentación dispone de un conector micro USB tipo B que proporciona 5 V de tensión. La mayoría de los cargadores para smartphones (que suministren más de 700 mA) son compatibles para dar tensión a la Raspberry Pi.

2.4.6 Puerto Ethernet

Se dispone de un conector RJ-45 conectado al integrado LAN9514 de SMSC, nombrado en el apartado, que proporciona conectividad a 10/100 Mbps.
Es posible conectar la Raspberry directamente a un PC sin pasar por un router conectando ambos equipos de manera directa con un cable RJ45.

Los modelos actuales de la Raspberry Pi no cuentan con un componente para poder acceder a redes inalámbricas, pero es posible añadir soporte Wi-Fi a la Raspberry utilizando un adaptador USB para red inalámbrica.

2.4.6. Instalación del Sistema Operativo

Dado que se va a trabajar con Python; el sistema operativo que se va a instalar en este caso es Raspbian.

Para poder instalar la distribución elegida es necesario usar un ordenador con lector de tarjetas SD. Usando un ordenador con Windows, el primer paso que hay que realizar es obtener el sistema operativo desde el siguiente enlace:

https://www.raspberrypi.org/downloads/

Posteriormente, se necesita descargar el software gratuito "Win32 Disk Imager". En Image File se escoge el fichero que contenga la imagen del Sistema Operativo, en Device hay que asegurarse que se elige la unidad correcta asociada a la tarjeta SD y mediante el botón Write se pasa el sistema a la tarjeta.

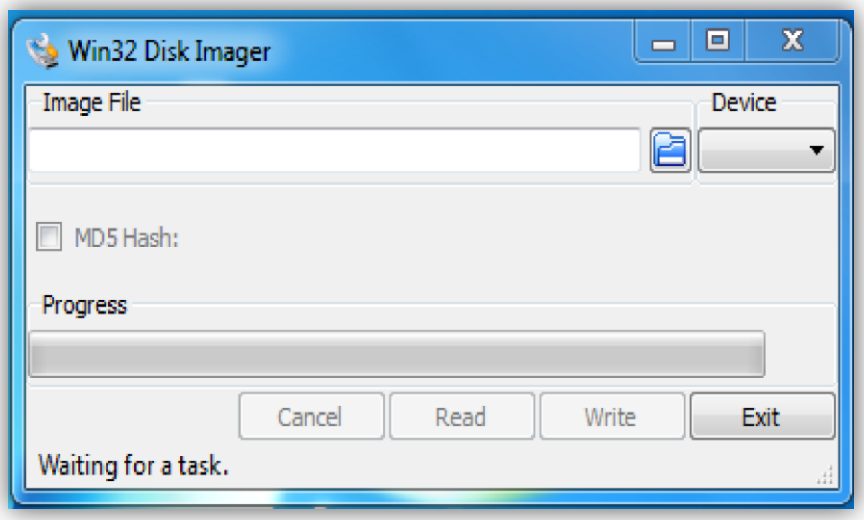


Figura. 2.14 Captura programa Win32 Disk Imager
Fuente: [Elaboración propia]

Una vez finalizado este proceso, ya se puede insertar la tarjeta en Raspberry Pi y comenzar a trabajar.

2.5 SENSOR DE HUELLA DACTILAR BIOMÉTRICO

2.5.1 Introducción a sensor huella dactilar

Un Sensor de huellas digitales (también conocido como *Sensor de huella dactilar*, *Lector de huella dactilar* o *Sensor biométrico*) Es un dispositivo que es capaz de leer, guardar e identificar las huellas dactilares (Generalmente del dedo pulgar, aunque la mayoría no tienen problemas en aceptar los demás dedos).

Todos los sensores biométricos cuentan mínimamente con una pieza que es sensible al tacto *(Que es el sensor en sí aunque luego hacen falta ciertas partes electrónicas)* Estos dispositivos se han hecho populares a raíz de que los últimos teléfonos inteligentes y tabletas han incorporado dicho sistema pues son los que mayor seguridad aportan. En la actualidad, las contraseñas proporcionan algo de protección, pero recordar y saber dónde están guardados los diferentes códigos de cada máquina es un problema en sí mismo.

Con las tarjetas inteligentes, sucede algo similar: si perdemos nuestra tarjeta no podremos hacer uso de las facilidades que brinda. Parecería lógico utilizar algún identificador que no se pudiese perder, cambiar o falsificar. Las técnicas de la biometría se aprovechan del hecho de que las características del cuerpo humano son únicas y fijas. Los rasgos faciales, el patrón del iris del ojo, los rasgos de la escritura, la huella dactilar, y otros muchos son los que se utilizan para estas funciones, incluyendo el ADN.

2.5.2 Lector de huellas para Arduino

La comunidad de Arduino no ha tardado en aprovechar esta técnica de identificación y por eso tenemos a nuestra disposición un módulo que tiene todo lo necesario para poder trabajar con ello. Este módulo de reconocimiento de huellas dactilares almacena las huellas en su memoria interna para compararlas y reconocerlas posteriormente.

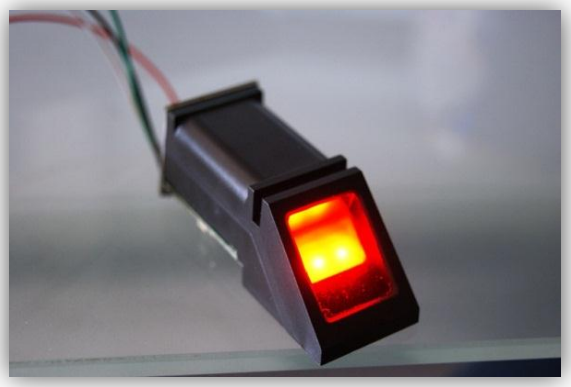


Figura. 2.15 Lector de huella biométrico
FUENTE: [https://hetpro-store.com/TUTORIALES/lector-de-huella-digital/]

2.5.2.1 Conexión y programación

El módulo es muy sencillo de conectar, dos cables para alimentación y dos para la comunicación serie. En uno de los extremos tienen un conector que nos facilita la conexión con el módulo lector, y que sólo encaja en una posición, así que no tiene perdida. Un pequeño inconveniente es que en el otro extremo no tienen ningún tipo de conector, así que os recomendamos que les soldéis unos pines, ya que sino no hacen bien contacto en los pines de Arduino ni en la protoboard.

Normalmente hay dos modelos, con 4 y 6 cables, aunque funcionan de la misma forma. En caso de que sea de 4, los cables rojo y negro se conectan a 5V y GND respectivamente para alimentar el módulo, y los otros dos son los encargados de la transmisión serie. El problema es que estos dos cables pueden tener varios colores y, por lo que hemos visto, también diferentes disposiciones.

Teniendo en cuenta que vamos a usar la librería Software Serial para no ocupar el puerto serie del UNO y dejarlo libre para la programación, esta sería la conexión con los colores más frecuentes.

También puede ser que el modelo tenga 6 cables, todos de color rojo. Esos dos cables extra nos dan otras funcionalidades que ya intentaremos explorar. Pero la

manera de conectarlo, igual que hemos descrito arriba es la siguiente:

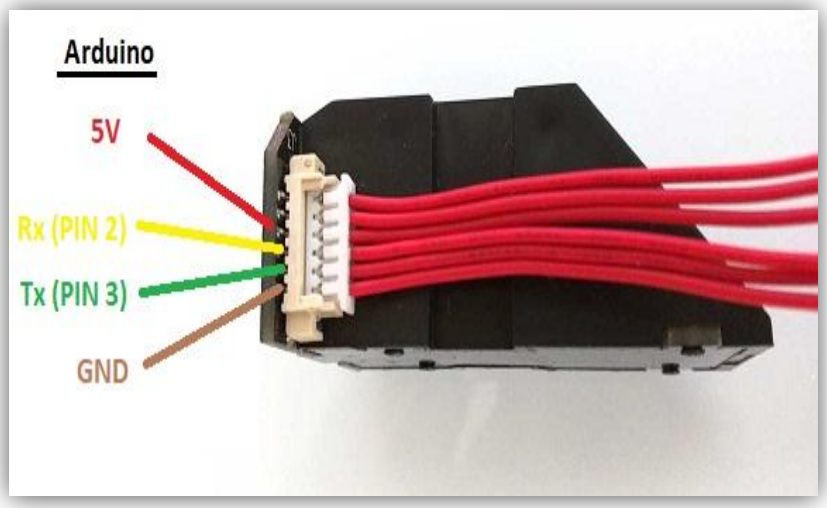


Figura. 2.16 Pineaje del Lector
Fuente: https://www.prometec.net/wp-content/uploads/2017/09/sensor-huellas-6-cables.jpg

La programación no tiene ningún misterio gracias a la librería que pone a nuestra disposición Adafruit ***https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library***, que hay que descargar e instalar como hemos hecho en otras ocasiones.

Una vez descargado abrimos el ejemplo “enroll” de la librería y lo cargamos en el Arduino. Este ejemplo sirve para escanear y guardar las que queramos reconocer más adelante. Depende del Arduino que se está utilizando puede ser que se tenga que cambiar los pines para la librería Software Serial en esta línea:

>> SoftwareSerial mySerial(2, 3);

Una vez cargado, abrimos el monitor serie y seleccionamos como velocidad de transmisión 9600 baudios. Si no lo encuentra el sensor les dará un mensaje como este. En ese caso lo más seguro es que tenga los cables de la transmisión serie al revés.

Si lo hemos conectado correctamente nos dirá que ha encontrado el sensor y nos pedirá un número de ID para guardar la huella que queramos. Sólo hay que darle un número, por ejemplo, el 0 y guardará la huella con esa ID. Podéis guardar hasta 162.

Cuando hayamos guardado las huellas que queramos, abrimos y cargamos otro de los programas de ejemplo de la librería, “fingerprint”. Este programa se encarga de escanear las huellas que pongamos en el sensor y compararlas con las que hemos guardado. Si hay alguna coincidencia nos informará por el puerto serie y nos dirá que número de ID le corresponde, además de otorgarles un nivel de confianza. SI no la hay, no mostrará nada.

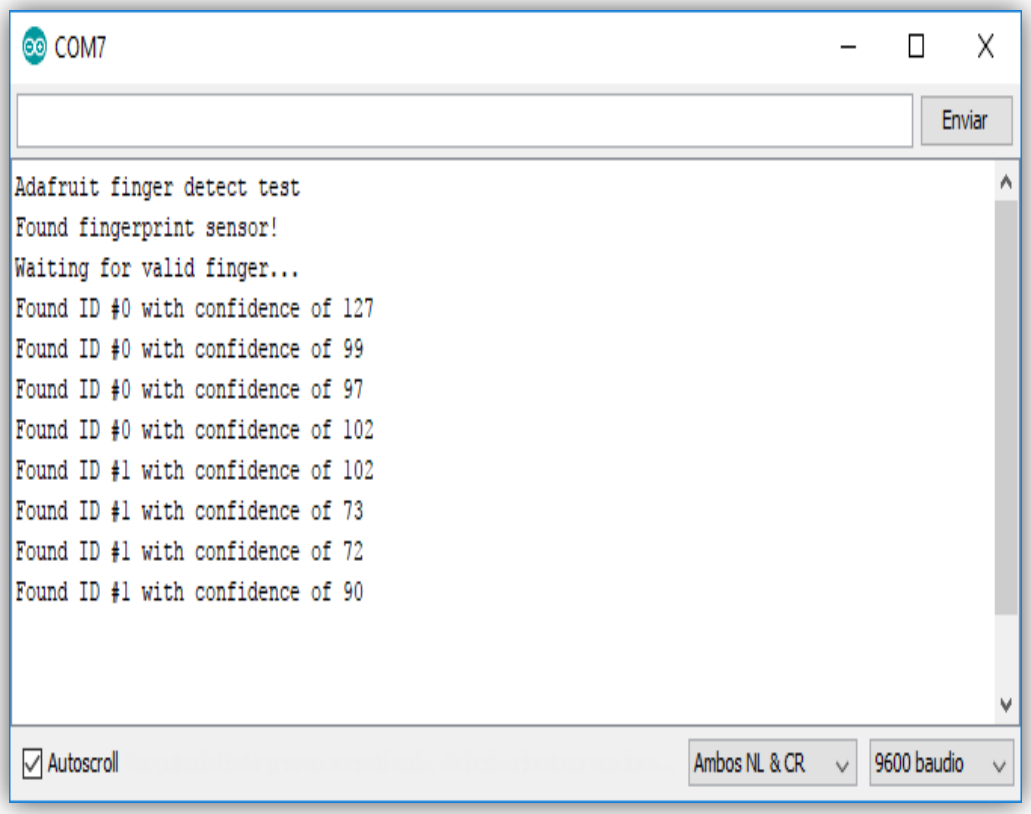


Figura. 2.17 Lecturas del Sensor

FUENTE: [https://www.prometec.net/modulo-huellas-dactilares-reconocer-huella.png]

2.6 CÁMARA RASPBERRY PI V2

Este módulo de cámara de 8 Mp es capaz de capturar video de 1080 p e imágenes; se puede conectar directamente a la Raspberry Pi. Compatible con el sistema operativo Raspbian listo para conectar y usar, muy adecuado para fotografiar por lapsos, grabar video o para usarlo en aplicaciones de seguridad y en detección de movimientos. Sólo hay que conectar el cable plano incluido al puerto CSI -Camera Serial Interface- de la Raspberry Pi.

La tarjeta es pequeña -mide 25 mm x 23 mm x 9 mm- y tiene una masa de 3 g, haciéndola perfecta para aplicaciones móviles u otras en donde el peso es un factor muy importante.

El sensor tiene una resolución neta de 8 megapíxeles y tiene un lente de enfoque. En cuanto a las imágenes, la cámara es capaz de tomar imágenes estáticas hasta de 3280 x 2464 pixeles y videos de 1830p30, 720p60 y 640x840p90.

2.6.1 Configurar la cámara en la Raspberry pi

El primer paso a seguir es el de actualizar correctamente el software de tu Raspberry Pi, en nuestro caso, hemos actualizado a la última versión de Raspbian en nuestra tarjeta microSD.

Luego debemos elegir el módulo de cámara que queremos utilizar. Actualmente hay dos en el mercado, el modelo Estándar y el modelo NoIR. La diferencia entre ambos la encontramos en el propio nombre, el segundo modelo no integra el filtro infrarrojo en su cámara de 5 MP.

Para conectar la cámara a nuestra Raspberry Pi debemos encontrar el puerto J3 que cuenta con una serigrafía en el PCB en la que se lee “CAMERA”. Aun así os dejamos una imagen de dónde se encuentra en la Raspberry Pi 3 model B+.

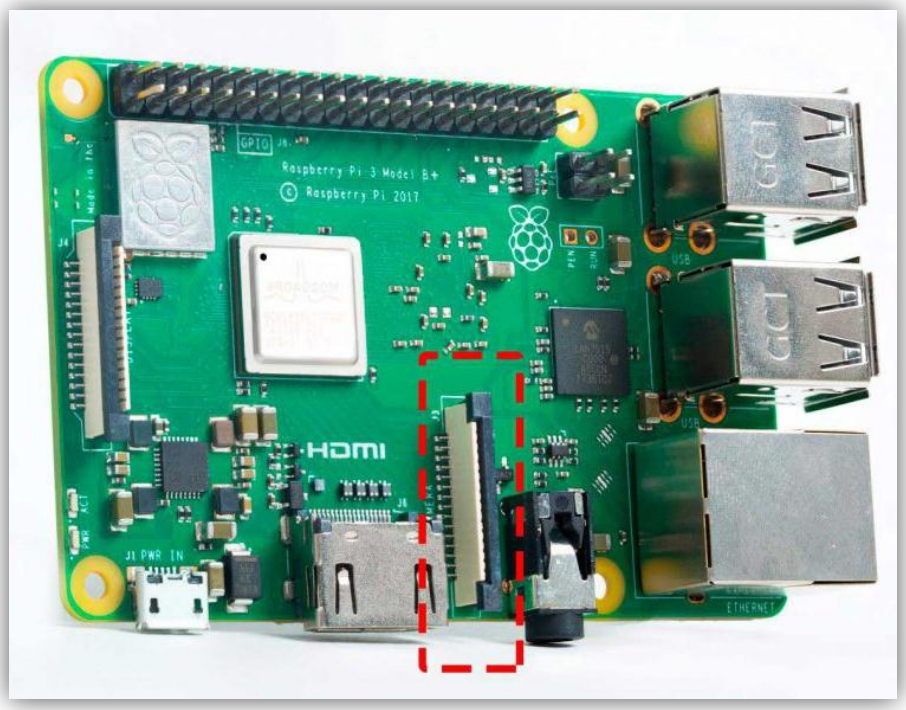


Figura. 2.18 Tarjeta y Slot de la Cámara
FUENTE: [https://hardzone.es/app/uploads/2018/08/raspberry-pi-3-model-b-localizacion-puerto-modulo-camara-1024x684.jpg]

Una vez hayas comprobado que todo se ha conectado correctamente, es el momento de encender la Raspberry Pi. Una vez se haya terminado de encender debemos dirigirnos a la pantalla de Configuración de la Raspberry Pi. En la interfaz de Raspbian se encuentra en el menú de Preferencias.

Otra forma de acceder es mediante el comando: ***sudo raspi-config***
Una vez hayas accedido a la configuración, dirígete a las Opciones de Interfaz y selecciona Cámara. En la ventana emergente selecciona Sí, ahora sólo debes esperar a que se reinicie el dispositivo.

2.6.1.1 Tomar fotos en la Raspberry pi

Una vez se haya finalizado el reinicio de la Raspberry Pi ya puedes tomar una foto. Para ello escribe en la ventana de comandos: raspistill -o image.jpg.

Esta foto de prueba se almacena en el directorio Imágenes en el escritorio de la Raspberry Pi. Este no es el único comando que podemos hacer, por ejemplo, podemos usar -vf o -hf para cambiar la rotación de la imagen. Si quieres conocer todos los comandos de la cámara utiliza raspistill en la ventana de comandos.

2.6.1.2 Grabar video en la Raspberry pi

Para grabar vídeos debemos aplicar este comando: raspivid -o video.h264
Si añades detrás de ese comando "-t X" siendo X el tiempo en milisegundos que quieres que grabe, el vídeo parará de forma automática al trascurrir el tiempo.

2.6.2 Conexiones y características

Un cable plano de 15 cm fijado a las ranuras del módulo directamente en el puerto de interfaz serie de su cámara Pi (CSI). Una vez conectado, puede acceder a la placa de cámara a través de la capa de abstracción multimedia (MMAL) o el vídeo para API de Linux (V4L). También hay bibliotecas como Picamera Python y muchas otras que puede encontrar en línea.

Características:

- Imágenes de alta calidad
- Alta capacidad de datos
- Enfoque fijo de 8 megapíxeles
- Compatible con 1080p, 720p60 y VGA90
- Sensor de imagen CMOS Sony IMX219PQ
- Cable plano de 15 contactos

2.7 PANTALLAS PARA RASPBERRY PI 3/2

La forma más sencilla de acceder a la señal de vídeo en un Raspberry Pi es utilizando una conexión HDMI. Otra opción es activar el soporte de escritorio remoto, aunque eso requiere un segundo ordenador, y no debemos olvidar la salida compuesta compatible con televisores antiguos. Sin embargo, tarde o temprano vamos a querer implementar una solución de vídeo más elegante en nuestro mini ordenador. Con esa idea en mente hemos decidido buscar cuatro pantallas para Raspberry Pi 3, y sus modelos anteriores.

Fue en octubre de 2016 que vimos funcionando a un Raspberry Pi conectado en un televisor del año 1980. A mediados del año pasado apareció RGB-Pi, un cable SCART personalizado que nos ayuda a reproducir con precisión los modos RGB sin diferencias entre el entorno emulado y los sistemas originales. Ambas opciones son maravillosas y garantizan una experiencia retro formidable, pero hay veces en las que solamente necesitamos reproducir texto, imágenes y vídeo en un formato mucho más compacto, y para ello no hay nada mejor que las pantallas dedicadas.

Figura. 2.19 Pantalla TFT para el Raspberry PI
FUENTE: [https://www.neoteo.com/wp-content/uploads/2018/05/logo-41-758x547.jpg]

2.7.1 Instalar pantalla en Raspberry pi

Conectar la pantalla TFT a la Raspberry antes de conectar está a la corriente y antes de empezar a modificar nada. Al encenderla la pantalla TFT se ha quedado en blanco. Mientras, tengo la Raspberry conectada a una TV a través del puerto HDMI para poder trabajar.

Antes de empezar vamos a actualizar la Raspberry Pi introduciendo estos comandos en la terminal:

1. **sudo apt-get update**
2. **sudo apt-get upgrade**

A continuación descargamos algunos archivos necesarios para la instalación del nuevo kernel mediante estos comandos:

1. **cd /home/pi**
2. **wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-bin-adafruit.deb**
3. **wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-dev-adafruit.deb**
4. **wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-doc-adafruit.deb**
5. **wget http://adafruit-download.s3.amazonaws.com/libraspberrypi0-adafruit.deb**
6. **wget http://adafruit-download.s3.amazonaws.com/raspberrypi-bootloader-adafruit-20140917-1.deb**

Al finalizar la descarga de los archivos podemos proceder a la actualización del kernel con este comando, este proceso durará un buen rato.

Guardamos los cambios como hemos hecho antes. La opción ***rotate=*** le indica

al driver de vídeo cuantos grados debe rotar la imagen *(0, 90, 180 o 270 grados)*, yo he usado 90 así la imagen queda como ***'landscape'***, mucho más grande que nos permitirá trabajar mejor. La opción ***frequency=*** le indica al driver de vídeo la velocidad de manejo de la pantalla, por ejemplo *32000000 (32Mhz)* nos sacará unos bonitos 20 FPS pero si la imagen falla podemos bajarlo a 16MHz (16000000).
Ahora podemos reiniciar la Raspberry a ver si todo funciona bien, ejecutamos este comando en la terminal:

1. **sudo reboot**

Si todo está bien nos debería aparecer algo como esto durante el booteo de Raspbian.

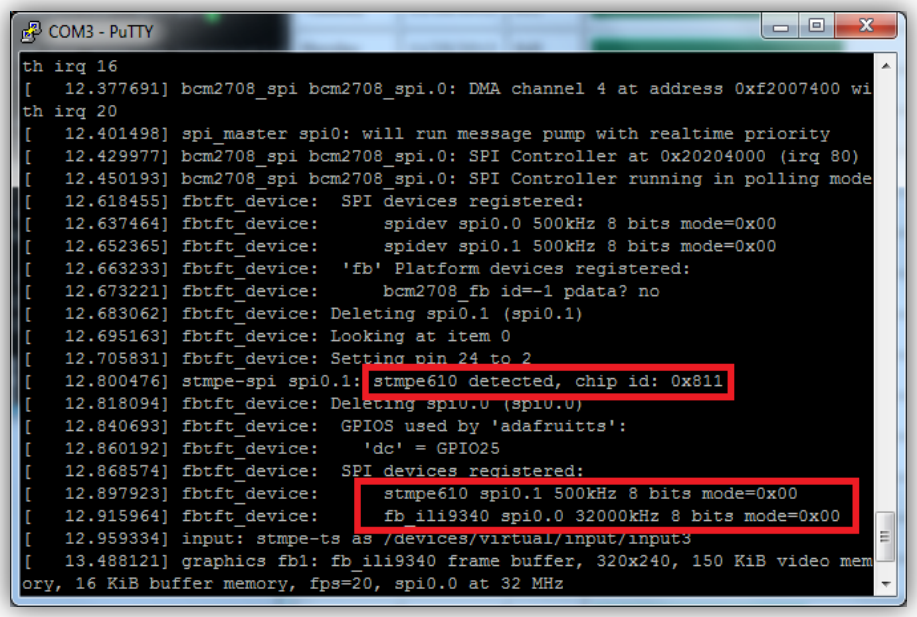


Figura. 2.20 Configuración de la Pantalla
FUENTE: [http://fpaez.com/wp-content/uploads/2014/10/raspberry_pi_tftpant.png]

Ahora después de reiniciar la tarjeta del Raspberry pi podemos, y antes de que inicie de nuevo desconectamos el cable HDMI que va a la TV y lo dejamos conectado la pantalla y veremos lo siguiente:

Figura. 2.21 Pantalla funcionando
FUENTE: [Imagen Propia]

2.8 INSTALACIÓN SOFTWARE REQUERIDO

2.8.1 Instalar Nginx en Raspbian jessie

Nginx es uno de los servidores web más populares del mundo y es usado por la mayoría de los sitios web con más tráfico de internet. Usa menos recursos que Apache en la mayoría de los casos, y puede ser usado como servidor web o proxy inverso. Lo primero es instalar Nginx en un servidor con Raspbian Jessie, donde puede ser tu propia máquina virtual.

Antes de comenzar, se debe tener en el Raspbian Jessie un usuario normal, un usuario que no sea root y que tenga configurado privilegios de sudo. El siguiente paso es darle al usuario privilegios de sudo. Para ello se va a utilizar el comando "**sudo**".

Ahora se procederá a instalar Nginx de forma fácil ya que el paquete se encuentra disponible en los repositorios oficiales de Raspbian. Como esta es la

primera vez que se va a usar el sistema de paquetes "***apt"*** en este, se debe actualizar el índice local de paquetes para tener la información más actualizada posible, empleando lo siguiente:

- *sudo apt-get update*
- *sudo apt-get install nginx*

Seguramente pedirá que se ingrese tu contraseña. Todo el software y sus dependencias serán descargados al servidor e instalado automáticamente.

En Raspbian Jessie, por defecto, Nginx arranca al ser instalado. Se puede acceder a la página por defecto de Nginx para confirmar que el servidor está funcionando de forma correcta visitando el dominio del servidor o IP pública (o local si es una máquina local). Deberías ver una página parecida a:

Figura. 2. 22 BIENVENIDA EN NGINX
FUENTE: [http://openwebinars.net/media/2014/05/nginx-defecto.png]

2.8.2 Instalar django 1.9 en Raspbian jessie

La plataforma donde se trabajará en la totalidad del sistema web Django, es un frameworks casi por excelencia para el desarrollo de web-app escrito en Python, se trata de una herramienta gratuita que permitirá mejorar el tiempo de desarrollo que empleamos en la creación del proyecto.

Django es un frameworks de desarrollo web de código abierto escrito en Python, que respeta el patrón de diseño conocida como Modelo – Vista – Controlador (MVC), fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Laurense Kansas que se liberó al público con una licencia BCD en Julio 2015.

Para instalar Django Framework y los demás paquetes que son necesarios debemos ingresar los siguientes comandos en nuestra terminal:

- *sudo su*
- *apt-get update*
- *apt-get install python3-pip*
- *pip3 install django==1.9 (en Repositorio se dará todos los paquetes instalado*

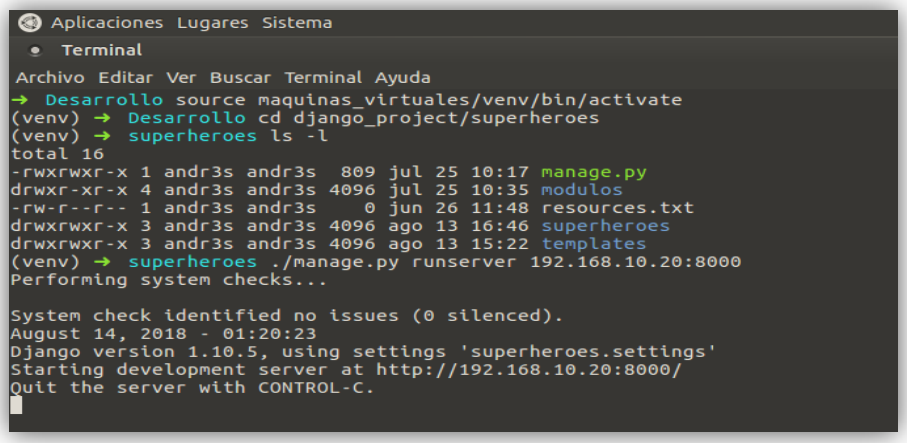


Figura. 2.23 Versión de Django1.9
FUENTE: [Elaboración Propia]

2.8.3 Instalar gulp-angular

Gulp es una herramienta de construcción utilizada por el equipo Angular y muchos desarrolladores más. La velocidad y la simplicidad se promocionan como beneficios sobre Grunt. Configurar un entorno de desarrollo es muy cómodo con estas herramientas, afortunadamente Gulp resuelve todas las tareas arduas que se realizaban antes.

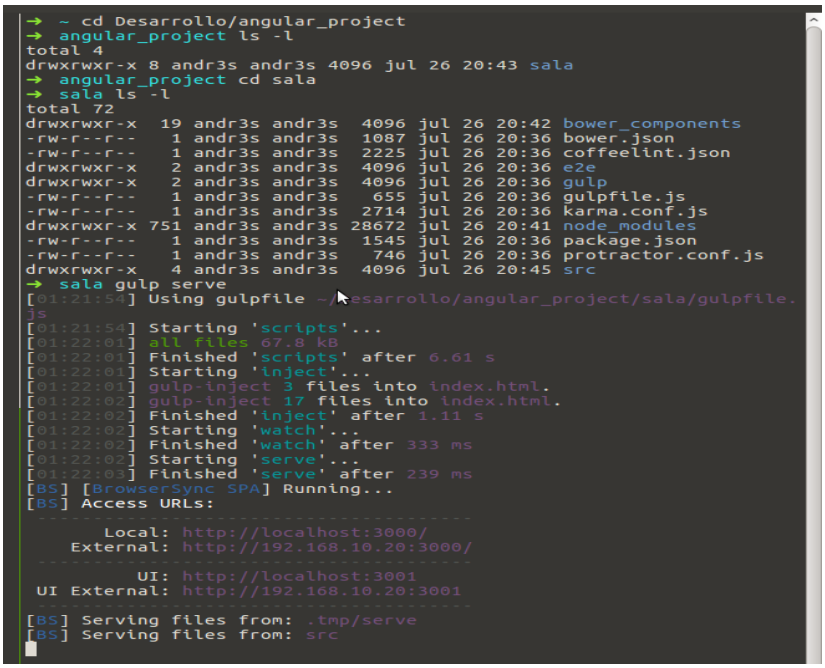


Figura. 2. 24 Correr el Entorno en Angular
FUENTE: [Elaboración propia]

2.8.4 Instalar PostgreSQL

PostgreSQL es un sistema de gestión de bases de dato relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyada por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PostgreSQL no tiene un gestor de defectos, haciendo muy difícil conocer el estado de sus defectos.

Figura. 2.25 Logo PostgreSQL
FUENTE:
[https://upload.wikimedia.org/wikipedia/commons/thumb/2/29/Postgresql_elephant.svg/200px-Postgresql_elephant.svg.png]

Usa los siguientes comandos en tu terminal para instalar el paquete de postgresql:

```
sudo apt-get install postgresql
```

Por ejemplo:

```
openerp@openerp-desktop:/$ sudo apt-get install postgresql
```

Para una interfaz gráfica de postgresql, usa el siguiente comando:

sudo apt-get install

Por ejemplo:

```
pgadmin3
openerp@openerp-desktop:/$ sudo apt-get install pgadmin3
```

Configura un usuario en PostgreSQL para OpenERP

Cuando la instalación del software requerido esté terminada, deberá crear un usuario de PostgreSQL. Este usuario deberá ser el mismo que tu sistema de usuario.

OpenERP usará este usuario para conectarse a PostgreSQL.

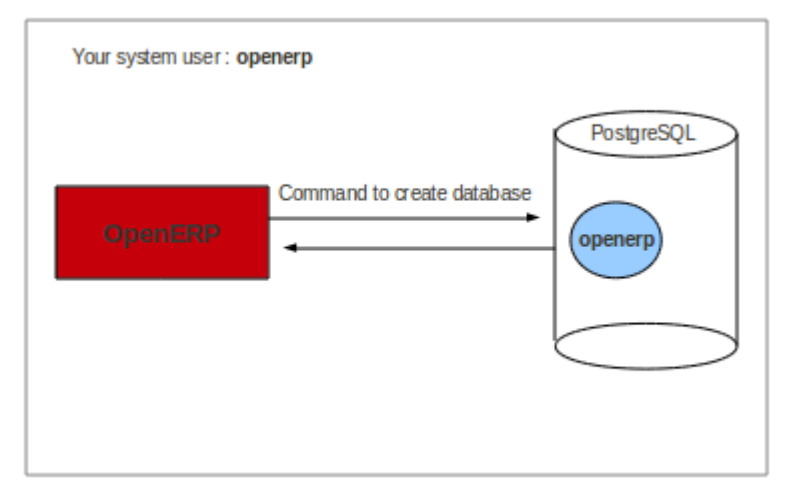


Figura. 2.26 OpenERP hace uso del usuario de PostgreSQL
FUENTE[https://doc.odoo.com/v6.0/es/install/linux/postgres/]

Base de datos

Como se explicó antes, sin crear y configurar usuario de PostgreSQL para OpenERP, no podrá crear una base de datos usando el cliente OpenERP.

Primer método

El súper usuario predeterminado de PostgreSQL se llama **Postgres**. Deberá ingresar con este usuario la primera vez.

```
openerp@openerp-desktop:/$ sudo su postgres

password: XXXXXXXXXX
```

Ahora crea el usuario de PostgreSQL **openerp** usando el siguiente comando:

```
postgres@openerp-desktop:/$ createuser openerp

Shall the new role be a superuser? (y/n) y
```

Deberá configurar las conexiones de la siguiente forma para poder tener acceso

vía **pgAdmin III**:

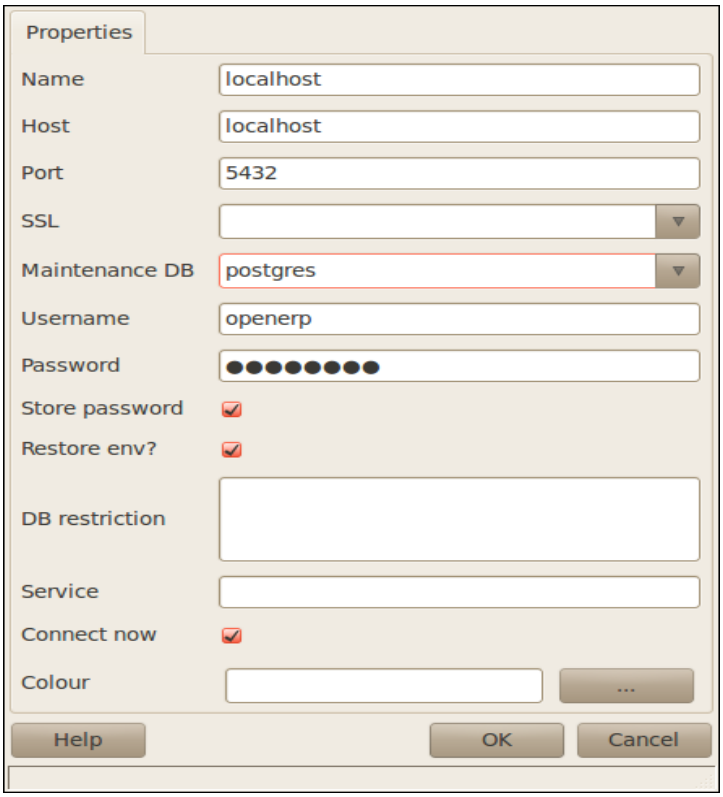


Figura. 2.27 Configuración de OpenERP
Fuente:[https://doc.odoo.com/v6.0/es/install/linux/postgres/]

CAPITULO III

INGENIERIA DE PROYECTO

3.1. INTRODUCCION

En esta parte del proyecto se desarrolla una descripción del diseño de un prototipo biométrico con reconocimiento facial y de huella para control del personal del Ministerio de Hidrocarburo.

3.2. DESCRIPCIÓN GENERAL DEL SISTEMA

Cumpliendo con el objetivo general del proyecto se realizó una aplicación web que interactúa con los procesos ligados a los Microcontroladores y Raspberry Pi por medio de conexión ethernet se puede conectar los distintos micro servicios que tiene alojado el RBP, como ser la parte de detección de rostro; que está basado en un servidor Flask de Python, otro es el sensor de huella que estará conectado a los GPIO del RBP, otro micro servició es el backend donde todo se centra que está basado en Django REST de Python y por ultimo necesitamos un frontend para la administración del personal y consulta de los marcados para ello se realizó una aplicación web en PHP sobre la base del Framework Codeigniter 3 que actúa como cliente.

3.2.1. Raspberry Pi 2

Raspberry Pi 2 es un ordenador de placa reducida, en el cual se alojará los diferentes micro servicios las cuales son: servicio de reconocimiento de rostro con Flask y OpenCV, servicio de backend el cual esta como un servicio API REST usando el Framework de Django REST de Python y por ultimo está el servicio de frontend que está en base a PHP con su framework Codigniter 3 que será un panel de administración de todos los usuarios, marcados, atrasos, reportes, etc.

3.2.2. Cámara Raspberry Pi v2

La cámara Raspberry Pi de alta definición (HD) se conectará al Raspberry Pi por un cable plano de 15 cm fijado a las ranuras del módulo directamente en el puerto de interfaz serie de su cámara Pi (CSI). Una vez conectado, puede acceder a la placa de cámara a través de la capa de abstracción multimedia (MMAL) o el vídeo para API de Linux (V4L). Con esta conexión podremos detectar los rostros del personal de la institución.

3.2.3. AS608 sensor óptico de huella

El escáner óptico de huellas digitales toma imágenes de huellas digitales y las convierte en identificadores únicos. El escáner de huellas digitales almacena hasta 128 huellas digitales diferentes y puede almacenarlas y escanear para encontrar la coincidencia correcta de huellas digitales, todo lo cual se realiza a través del escáner y la placa Arduino.

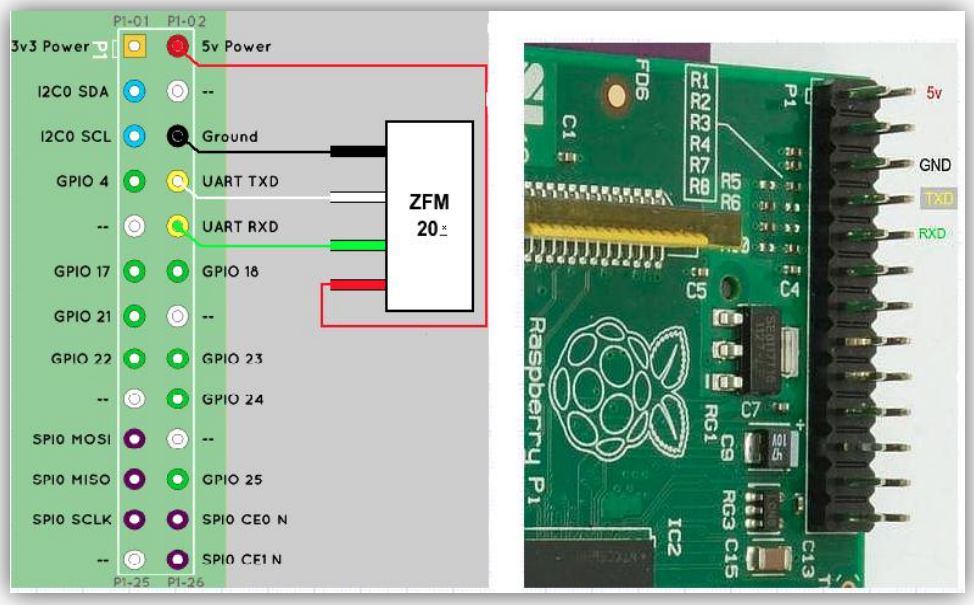


Figura. 3.1 Conexión Sensor de Huella con el RBP
FUENTE: [Elaboración Propia]

En la Figura 3.1 será la forma de conexión para que se puedan comunicar el modulo sensor y el Raspberry pi.

3.2.3. Pantalla TFT LCD

La pantalla para el Raspberry Pi permite a los usuarios poder visualizar la hora y el marcado de cada usuario, la conexión al Raspberry Pi es fácil de instalar, sólo dos de las conexiones de la RBP son necesarios para la conexión: el poder por el puerto GPIO y un cable plano que se conecta al puerto de ISD.

3.2.4. Fuente de alimentación

En la parte de la fuente de alimentación se contará con dos cargadores normales con entrada de 220voltios AC y con dos salidas de 5 voltios DC con una corriente de 5 A (amperios), se distribuirá uno para la tarjeta Arduino y el otro será para el pequeño servidor web Raspberry Pi.

Figura. 3.2 Cargador para la placa
FUENTE: [Elaboración propia]

3.3. APLICABILIDAD DEL SISTEMA A LA SALA DE SERVIDORES DEL MINISTERIO.

El diseño y prototipo del sistema se aplicará en la sala de Servidores del Ministerio de Hidrocarburos.

El Ministerio se encuentra ubicado en la Av. Mariscal Santa Cruz, esq. Calle Oruro, Edif. Centro de Comunicaciones N° 1260. El área que pertenece a los ambientes interiores del Ministerio, piso 12.

Figura. 3.3 Ambientes Exteriores e Interiores del Ministerio
FUENTE: [Elaboración Propia]

3.4. SOFTWARE DE CONFIGURACION DEL SISTEMA

Uno de los micro servicios más importantes es el backend el cual recibirá los datos de cada registro de biométrico hacia la base de datos y contiene los reportes, permisos, etc. En la configuración de este sistema se utiliza el framework Django REST, una herramienta de desarrollo web para hacer una aplicación basado en lenguaje Python.

El otro micro servicio es de igual importante, ya que este es el que monitorea los sensores de huella y reconocimiento facial está basado en el Framework Flask de Python junto al paquete Open CV y de igualmente para el sensor de huella todas estas herramientas estarán configuradas e instaladas en el Raspberry Pi2, con sistema operativo Raspbian de debian.

Para la parte del cliente y administración de usuarios y ver marcados se os puede realizar de cualquier lenguaje e incluso por aplicaciones móviles, siendo estos casos que solo consuman recursos y datos del backend. En este caso se lo realizara con el Framework CodeIgniter de PHP.

3.4.1. Framework Django REST como herramienta de desarrollo web

Django REST es un framework diseñado principalmente para crear aplicaciones web puramente diseñado en el lenguaje de Python que es lo mejor que puede tener Django, es un framework open-source, que quiere decir que su código raíz esta accesible para todos que nos servirá como un servidor REST.

3.4.2. Proceso de configuración y creación del backend

Para poder trabajar con Framework Django REST lo primero que se debe realizar es instalar la herramienta, para ello se tiene que abrir la terminal (CTR+ALT+t) para los usuarios GNU/Linux aparecerá una consola donde debemos de ingresar los siguientes comandos:

- *sudo apt-get update*
- *sudo apt-get install pip*
- *sudo apt-get install django*
- *sudo apt-get install unipath*
- *sudo apt-get install nginx*
- *sudo service restart nginx*

Con los anteriores comandos ya ingresados se prepara el entorno de trabajo. Lo siguiente que se hace es activar Django con los comandos, que se muestra en la figura 3.4, el cual se está activando el entorno virtual para que corra Django REST.

Figura.3.4 Activar Entorno Virtual Django1.9
FUENTE: [Elaboración Propia]

Ahora se creará una aplicación donde se direcciono, donde la dirección de la carpeta será: Documentos/proyectos_django:

- *cd Document/proyectos_django*
- *django-admin startproject primero*
- *cd primero*

Con el comando anterior ya ingresados se debe ver si se creó correctamente el proyecto en la figura 3.5:

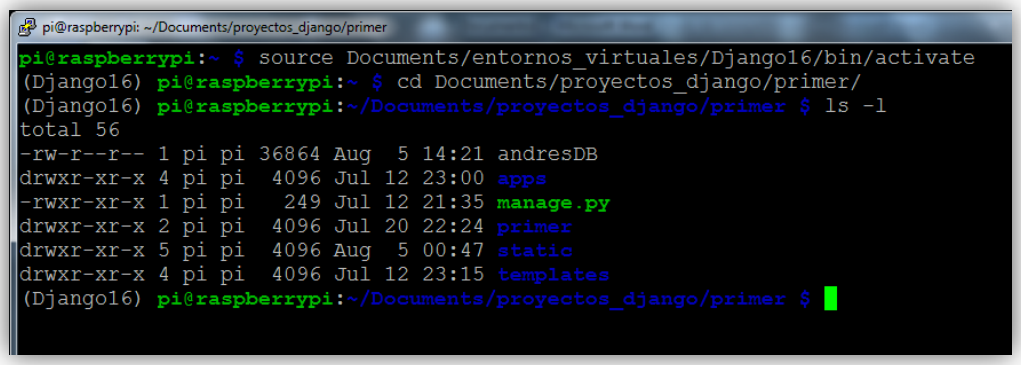


Figura.3.5 Comprobar el proyecto creado
FUENTE: [Elaboración Propia]

Si se observa en la figura 3.5, se notará que la herramienta de Django creo varias carpetas que utilizará el proyecto. El siguiente paso y el último es crear una aplicación dentro de la carpeta ***app***, ingresando los siguientes comandos:

- *cd app/*
- *django-admin startapp inicio*
- *cd ..*

Se debe probar si las instalaciones y creación del proyecto está correctamente hechas, se debe de ingresar a la carpeta principal e ingresar los comandos siguientes indicando el número IP del Raspberry y el puerto por donde se comunicará por el mundo exterior por defecto ya vienen en el puerto ***8000***, que se muestran en la figura 3.6.

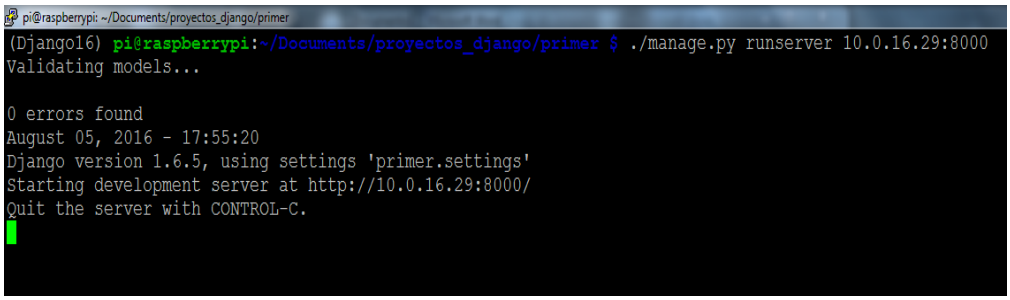


Figura.3.6 Arrancando el proyecto Entorno de desarrollo Django
FUENTE: [Elaboración Propia]

3.4. INSTALAR SOFTWARE REQUERIDO

La parte un poco morosa es la instalación de algunos paquetes de software que requiere Python y el sistema para que funcione el proyecto, los paquetes más importantes son las siguientes:

3.4.1 Instalar OpenCV

Como primer paso debemos de actualizar los paquetes del sistema, con los siguientes comandos:

~>$ sudo apt -y update
~>$ sudo apt -y upgrade

Luego de actualizar debemos de instalar los paquetes que son necesarios para Open CV, con el siguiente comando:

~>$ sudo apt-get -y install build-essential checkinstall cmake pkg-config yasm
~>$ sudo apt-get -y install git gfortran
~>$ sudo apt-get -y install libjpeg8-dev libjasper-dev libpng12-dev
~>$ sudo apt-get -y install libtiff5-dev
~>$ sudo apt-get -y install libtiff-dev
~>$ sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev
~>$ sudo apt-get -y install libxine2-dev libv4l-dev
~>$ cd /usr/include/linux
~>$ sudo ln -s -f ../libv4l1-videodev.h videodev.h
~>$ cd $cwd
~>$ sudo apt-get -y install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev
~>$ sudo apt-get -y install libgtk2.0-dev libtbb-dev qt5-default
~>$ sudo apt-get -y install libatlas-base-dev

~>$ sudo apt-get -y install libmp3lame-dev libtheora-dev

~>$ sudo apt-get -y install libvorbis-dev libxvidcore-dev libx264-dev

~>$ sudo apt-get -y install libopencore-amrnb-dev libopencore-amrwb-dev

~>$ sudo apt-get -y install libavresample-dev

~>$ sudo apt-get -y install x264 v4l-utils

Si todo funcionara perfectamente, podríamos clonar OpenCV desde git. Este paso también toma unos minutos.

~>$ git clone https://github.com/Itseez/opencv.git && cd opencv &&git checkout 3.0.0

Ahora podemos simplemente instalar vía pip NumPy. NumPy es una librería que hace muy fácil realizar operaciones de array en Python.

~>$ pip install numpy

Pero ahora vamos a compilar OpenCV. Para ello, se debe crear una carpeta de compilación en la que colocar los archivos compilados:

~>$ mkdir build && cd build
~>$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D INSTALL_PYTHON_EXAMPLES=ON -D INSTALL_C_EXAMPLES=ON -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules -D BUILD_EXAMPLES=ON ..

Ahora finalmente puedes compilar. Este paso lleva, dependiendo del modelo de Raspberry Pi, bastante tiempo, en una Pi 2 alrededor de una hora. Para usar los cuatro núcleos para compilar en la Raspberry Pi 2, escribe lo siguiente:

~>$ make -j4

Si la compilación ha funcionado sin problemas, podemos instalar OpenCV:

~>$ sudo make install && sudo ldconfig

Para comprobar si todo funcionó, puedes abrir la consola Python e importar la biblioteca: # import cv2

cv2.__version__

3.4.2 Instalar IMUTILS Y NUMPY

Estos paquetes son un poco más fáciles de instalar, ya teniendo PIP con gestor para instalar paquetes de Python lo podemos usar para instalar los paquetes que nos faltan, con el siguiente comando:

~>$ sudo pip3 install click==6.7
~>$ sudo pip3 install itsdangerous==0.24
~>$ sudo pip3 install Jinja2==2.9.6
~>$ sudo pip3 install MarkupSafe==1.0
~>$ sudo pip3 install Werkzeug==0.12.2
~>$ sudo pip3 install certifi==2019.6.16
~>$ sudo pip3 install chardet==3.0.4
~>$ sudo pip3 install Click==7.0
~>$ sudo pip3 install Django==2.2
~>$ sudo pip3 install django-cors-headers==2.4.0
~>$ sudo pip3 install django-qr-code==1.0.0
~>$ sudo pip3 install djangorestframework==3.9.1
~>$ sudo pip3 install djangorestframework-jwt==1.11.0
~>$ sudo pip3 install Flask==1.1.1
~>$ sudo pip3 install html5lib==1.0.1
~>$ sudo pip3 install idna==2.8
~>$ sudo pip3 install imutils==0.5.3
~>$ sudo pip3 install itsdangerous==1.1.0
~>$ sudo pip3 install MarkupSafe==1.1.1
~>$ sudo pip3 install Pillow==5.4.1
~>$ sudo pip3 install psycopg2==2.7.7

~>$ sudo pip3 install psycopg2-binary==2.7.7

~>$ sudo pip3 install PyJWT==1.7.1

~>$ sudo pip3 install PyPDF2==1.26.0

~>$ sudo pip3 install pytz==2019.1

~>$ sudo pip3 install qrcode==6.1

~>$ sudo pip3 install reportlab==3.5.17

~>$ sudo pip3 install requests==2.22.0

~>$ sudo pip3 install six==1.12.0

~>$ sudo pip3 install sqlparse==0.3.0

~>$ sudo pip3 install urllib3==1.25.3

~>$ sudo pip3 install webencodings==0.5.1

~>$ sudo pip3 install xhtml2pdf==0.2.3

3.5. HARDWARE DEL SISTEMA

El hardware del sistema contiene el Raspberry Pi 2, Sensor de reconocimiento de Huella, cámara del Picam, Pantalla TFT, salidas digitales, distintos sensores, fuente de alimentación del sistema (cargador).

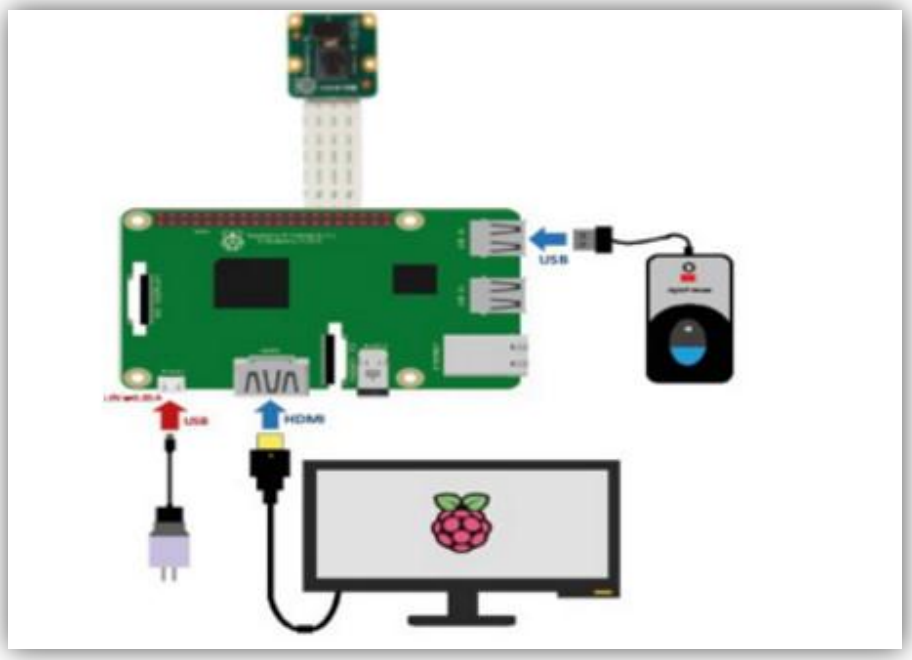


Figura. 3.7 Diagrama de Bloques del Hardware
FUENTE: [Elaboración Propia]

3.5.1. Conexión del Raspberry Pi 2 y Cámara

El primer paso a seguir es el de actualizar correctamente el software de tu Raspberry Pi, en nuestro caso, hemos actualizado a la última versión de Raspbian en nuestra tarjeta microSD.

Para conectar la cámara a nuestra Raspberry Pi debemos encontrar el puerto J3 que cuenta con una serigrafía en el PCB en la que se lee “CAMERA”. Aun así, dejamos una imagen de dónde se encuentra en la Raspberry Pi.

Figura. 3.8 Slot de la Cámara
FUENTE: [Elaboración Propia]

Una vez se haya comprobado que todo se ha conectado correctamente, es el momento de encender la Raspberry Pi. Una vez se haya terminado de encender debemos dirigirnos a la pantalla de Configuración de la Raspberry Pi. En la interfaz de Raspbian se encuentra en el menú de Preferencias.

Otra forma de acceder es mediante el comando: *sudo raspi-config*

Una vez hayas accedido a la configuración, dirígete a las Opciones de Interfaz y selecciona Cámara. En la ventana emergente selecciona Sí, ahora sólo debes esperar a que se reinicie el dispositivo.

El paquete de OpenCV, es una herramienta de adquisición y tratamiento de imágenes, esta librería consta básicamente de tres tipos de algoritmos: Eigenfaces, Fisherfaces y local Bynary Pattrerns Histograms). Se implementó el algoritmo de eigenfaces, el cual está basado en la técnica de análisis de componentes principales (PCA o ACP) la cual busca principalmente reducir la dimensionalidad de un grupo de datos buscando destacar y utilizar como punto de comparación las características no compartidas y que representan la mayor variación en un grupo de datos. Eigenfaces usa un gran conjunto de imágenes digitalizadas de rostros humano, tomados con condiciones de iluminación similares y las alinea a la altura de los ojos y boca, entonces son redimensionados a la misma resolución, y a partir de la aplicación de PCA se pueden extraer los suficientes datos de la imagen eigenfaces resultante, figura. 3.9, que sirvan como punto de comparación contra nuevos grupos de datos o imágenes.

Figura. 3.9 Ejemplo de Aplicación de Eigenfaces
FUENTE: [http://www.diegoguerrero.info/wp-content/uploads/2012/03/caras.png]

Se utilizó una base de datos la cual permite generar una imagen eigenfaces de características negativas para su comparación, figura. 3.10. En este caso se ha tomado la base de datos de rostros de AT & T desarrollada en Abril de 1992.

Figura. 3.10 Imagen Eigenfaces Negativa
FUENTE: [Elaboración Propia]

Se realizó en Python 3 a través de varios scripts que desarrollan tareas específicas, estos se invocan en un script central llamado camira_pi.py, las necesidades de la autentificación facial son:

1. captura y almacenamiento de nuevos rostros
2. Comparación y autentificación de rostros.

3.5.2. Conexión del Raspberry Pi 2 y sensor de huella

La autentificación de huella está compuesta por el módulo FPM10A, figura 26, que incluye un sensor de huella dactilar óptica, un chip DSP, y una UART de comunicación serial. Este también cuenta con dos buffers en los cuales se pueden leer y escribir, por medio de instrucciones, la información de estos buffers no se almacena después de que el modulo se apaga. Se escogió este módulo por su facilidad de conexión con cualquier micro controlador o sistema con TTL en serie, y su disponibilidad de adquirir en el mercado local.

Figura. 3.11 Lector de huella módulo FPM10A
FUENTE: [Elaboración Propia]

Se utilizó la librería “pyfingerprint” para Python, la cual permite la conexión, transmisión y recepción de los datos. El módulo FPM10A se comunica por el protocolo de comunicación RS232, maneja por defecto una velocidad de 57600 bps, con un formato de trama de 10 bits, un bit de inicio, datos de 8 bits y un bit de parada, sin paridad. La Raspberry pi solo le da instrucciones al módulo, y el modulo respuestas a la Raspberry, por medio de paquetes hexadecimales.

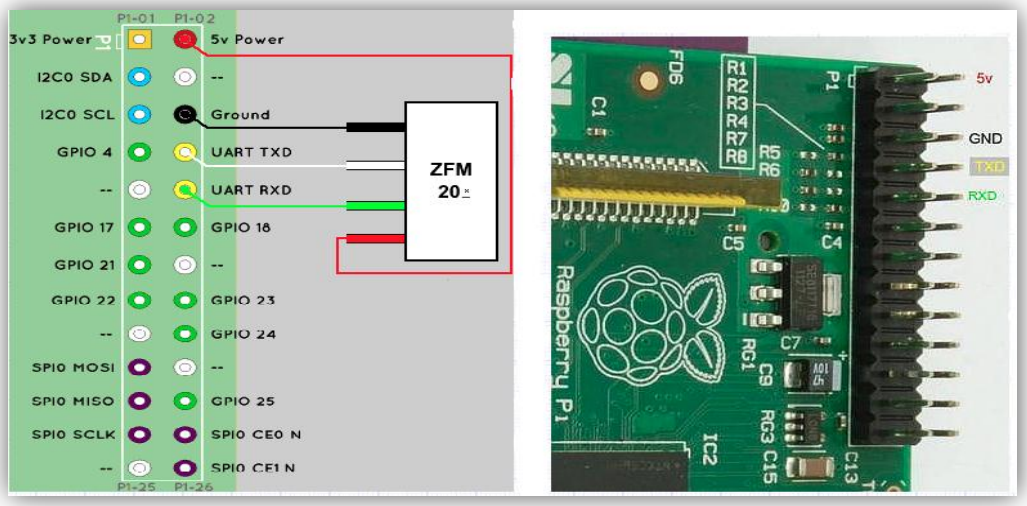


Figura. 3.12 Conexión Sensor de Huella con el RBP
FUENTE: [Elaboración Propia]

Para registrar la huella, primero se lee la huella del individuo, se le extraen las características y se guardan en el buffer 1 del módulo, se hace la verificación que esta no se encuentre registrada, si es así se procede otra vez a leer la huella del individuo, se extraen las características y se guardan en el buffer 2. Se comparan los resultados de los dos buffers, si estos son iguales, se procede a guardar la plantilla de la huella, en la memoria flash del módulo y retornar el número de serie con la que quedo almacenada (posición). Para la autentificación de la huella se realiza el mismo procedimiento anterior, con la diferencia que solo se toma una sola muestra de la huella, donde sus características se guardaran en el buffer 1, y estas se comparan con las que se encuentran guardadas en el módulo, dando un resultado positivo o negativo. En la eliminación de la huella solo se debe de ingresar la posición en donde quedo guardada la huella, el modulo retornara un dato booleano, si el numero ingresado corresponde a una posición (numero) valida.

3.7. DIAGRAMA COMPLETO DEL SISTEMA

En el diagrama se observa la parte central del circuito completo, en el cual se realiza una autentificación dactilar y facial del personal de la institución, el cual tiene que efectuar un registro previo donde se guarda su información correspondiente y realizar la carga de datos en el sistema, para que el usuario pueda realizar su validación en el sistema y marcado correspondiente. Al ejecutar una validación positiva y el usuario ingresa a la instalación se guardarán los datos de ingreso (hora y fecha) en una base de datos que es PostgreSQL, estos se podrán visualizar en una interfaz gráfica web. Por medio de comandos auditivos y visuales se le informa al personal su registro de ingreso/salida de la institución. En la figura 3.12 se observa el diagrama de bloques desarrollado para la elaboración del control de acceso por biometría. El control cuenta con 4 bloques fundamentales: Autentificación de huella dactilar, Autentificación facial, bloque de control, interfaz gráfica web, serán descritos a continuación.

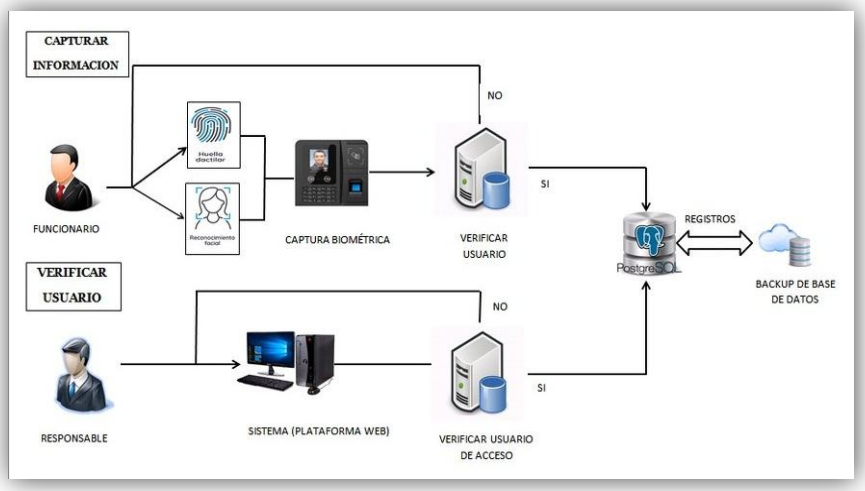


Figura. 3.13 Diagrama de bloques
FUENTE: [Elaboración Propia]

3.7.1. Circuito controlador

En el diagrama se observa la parte central del circuito completo. Las conexiones que tienen el Raspberry con los demás dispositivos, como ser: la cámara (para el reconocimiento facial), el sensor de detector de huellas, una pantalla TFT para la visualización de la hora, el rostro y la notificación y por ultimo tenemos un parlante de 8w para la notificación mediante sonido de un registro exitoso.

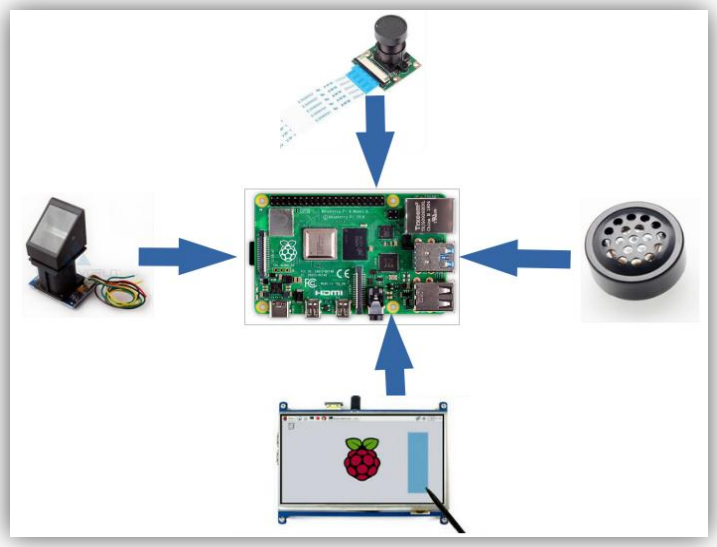


Figura. 3.14 Circuito Controlador
FUENTE: [Elaboración Propia]

3.7.2 Programa Simulador EasyADA

Se utilizó la herramienta gratuita EasyEDA una de las mejores herramientas software de simulación de circuitos y diseño de PCB que basada en la nube.

Una característica muy importante es que se podrá importar a EasyEDA otros diseños hechos con Altium, Eagle y KiCad, de esta forma no solo podremos guardar en la nube nuestros esquemas, sino también editarlos si fuera necesario.

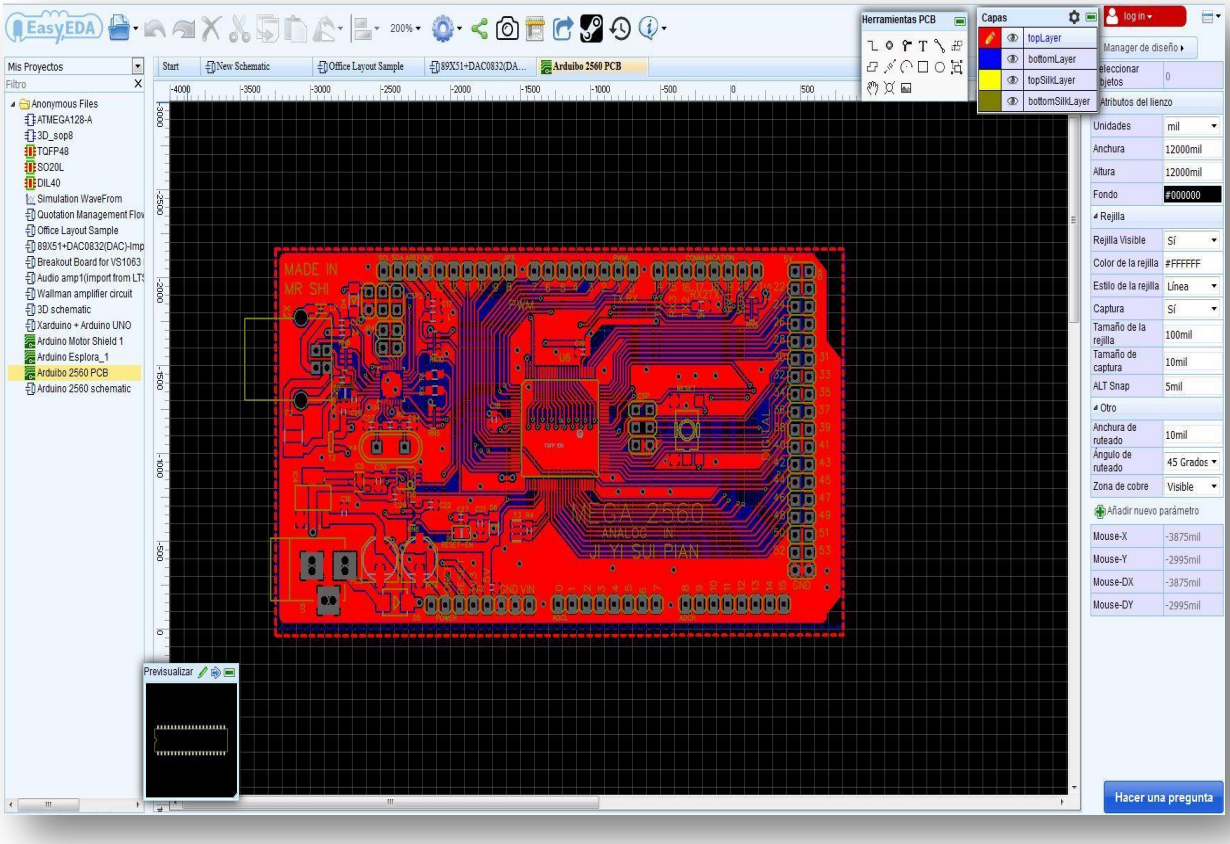


Figura.3.15 Programa easyADA
FUENTE: [Elaboración Propia]

3.7.3 Circuito Prototipo

Para el diseño del prototipo biométrico facial y dactilar se iniciará con el diseño del circuito, este diseño se realizó con el programa easyAda.

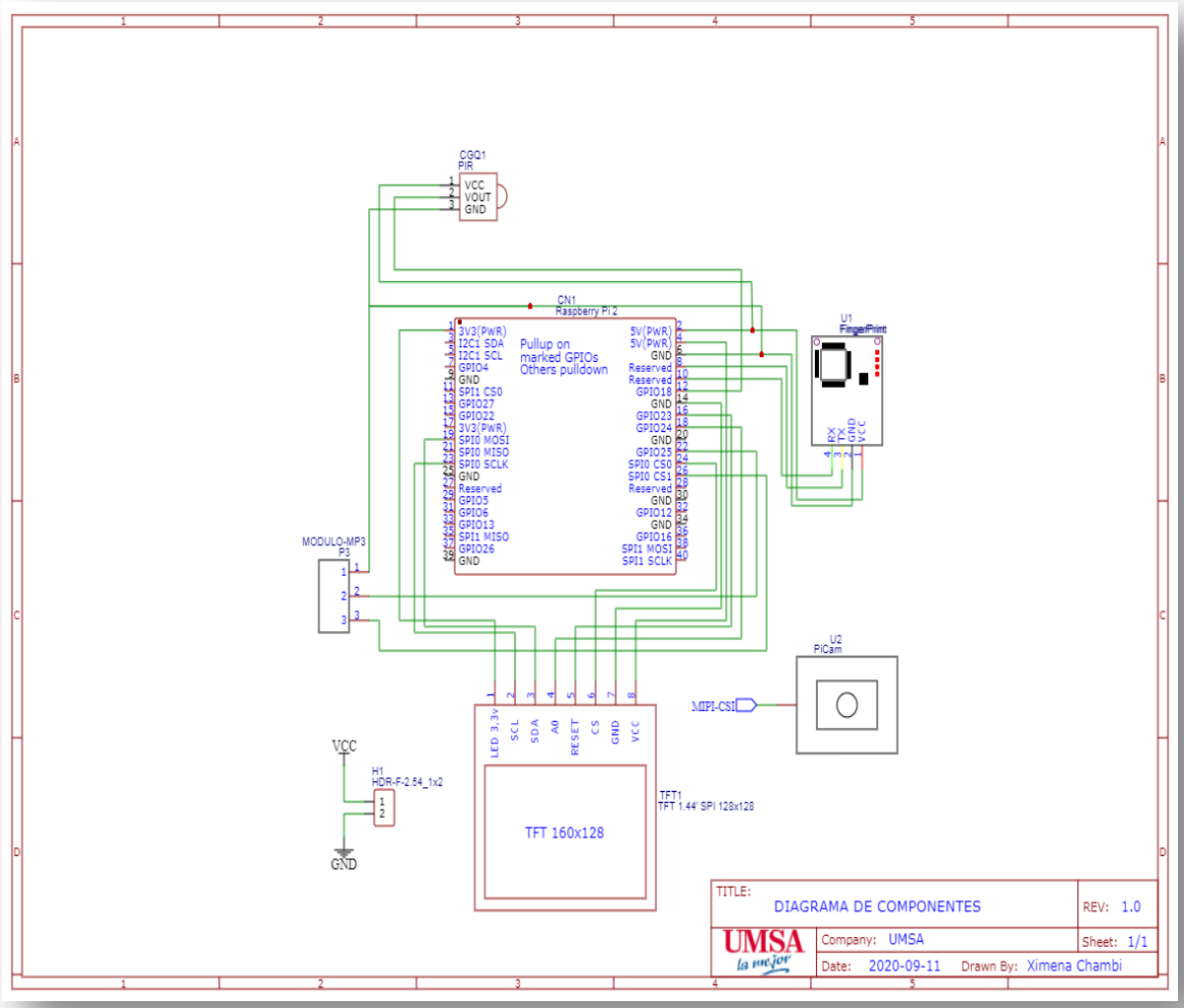


Figura.3.16. Simulación en easyADA del circuito completo
FUENTE: [Elaboración Propia]

Placa quemada

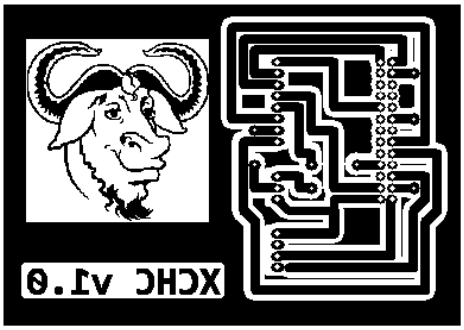


Figura.3.17. placa PCB
FUENTE: [Elaboración Propia]

3.8. FUNCIONAMIENTO DEL SOFTWARE

La Raspberry Pi, se encarga de hacer el control de 4 tareas: registro, autentificación, eliminación y actualización de datos del usuario, enviándole las funciones correspondientes a los bloques de autentificación de huella dactilar y facial. El usuario interactúa con el bloque de control por medio de una Pantalla TFT que visualiza el menú inicial y el paso a paso a seguir en cada bloque, para registrar necesitaremos un teclado, que le permite ingresar los datos correspondientes para poder acceder a cada tarea y una alerta de iluminación (verde= positiva y roja=negativa) según el acceso que se obtenga. En la figura 3.14 se puede apreciar el diagrama de flujo, que se tuvo en cuenta en la implementación del código en la Raspberry Pi.

El registro se encarga de guardar la huella en el módulo FPM10A, las 30 diferentes muestras del rostro (fotografías) del usuario, la posición en la que quedo guardada la correspondiente huella en el módulo, el ingreso del usuario y clave, las dos últimas para poder acceder a la interfaz gráfica de la página web, exceptuando la huella y los rostros, los demás datos se guardan en PostgreSQL, donde se puede modificar y obtener los datos desde Python o PHP.

La autentificación compara las huellas y rostros, con las que ya se han registrado posteriormente, si la comparación es positiva para cualquiera de los bloques, tendrá un acceso permitido, sino es así será un acceso negativo.
La eliminación se encarga de eliminar la huella, usuario y clave que fueron proporcionados en el registro.

La actualización de datos, básicamente esta función es para el administrador y aplicable para el reconocimiento facial, al realizar un registro de una nueva persona, los datos guardados (fotos) deben ser actualizados para la creación de una nueva imagen eigenfaces positiva, más o menos está tarda de 5 – 10 minutos, este tiempo depende del total de personas que se encuentren registradas.

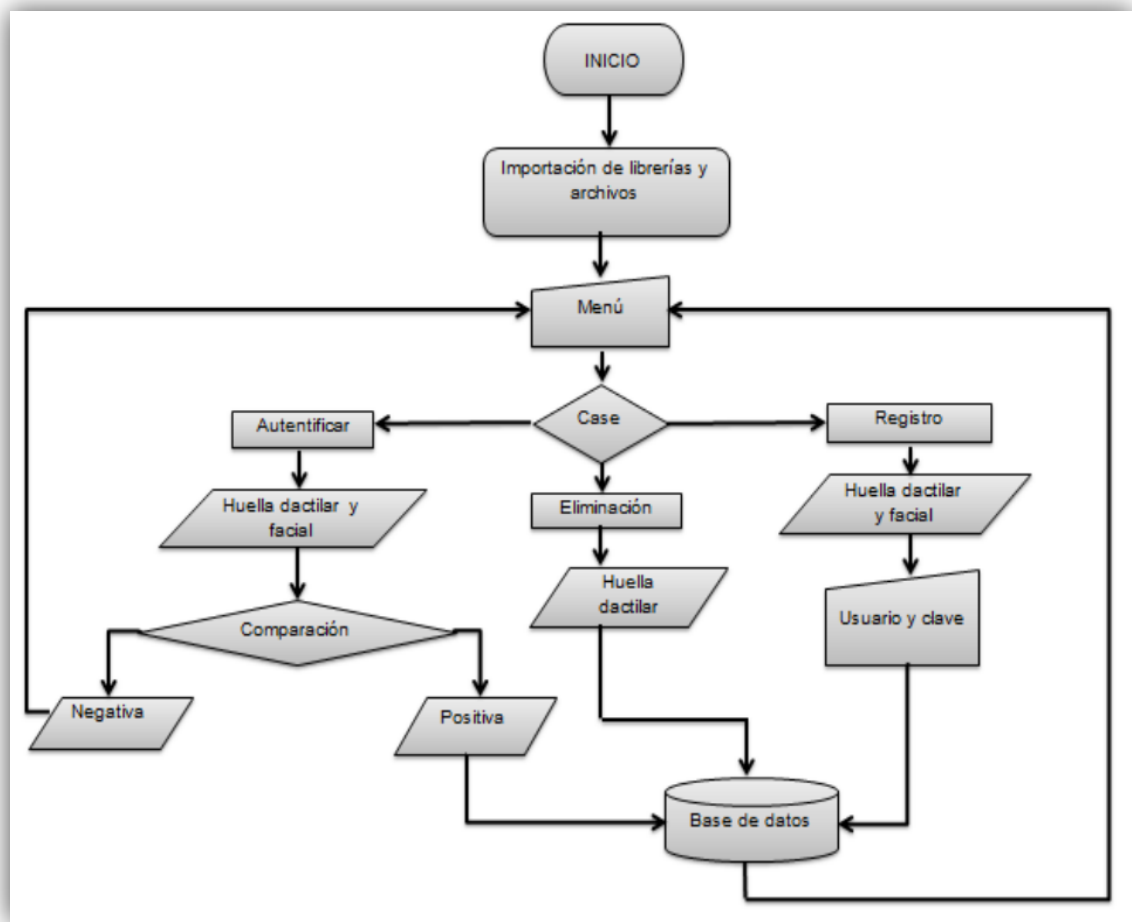


Figura. 3.18 Diagrama de flujo del bloque de control
FUENTE: [Elaboración Propia]

3.9. PRUEBAS Y RESULTADOS DE FUNCIONAMIENTO DEL SISTEMA

En esta sección se presentan las imágenes de la implementación física del sistema y algunos detalles relevantes al respecto. En el documento se encuentra el análisis de los resultados en base a pruebas de funcionamiento del sistema.

3.9.1. Implementación física del sistema

Se observa la implementación física del sistema y los módulos de control, también el sistema de alimentación y las salidas digitales del Raspberry Pi2 están

implementados dentro de una caja diseñada para que aloje todo el hardware que comprende el sistema biométrico.

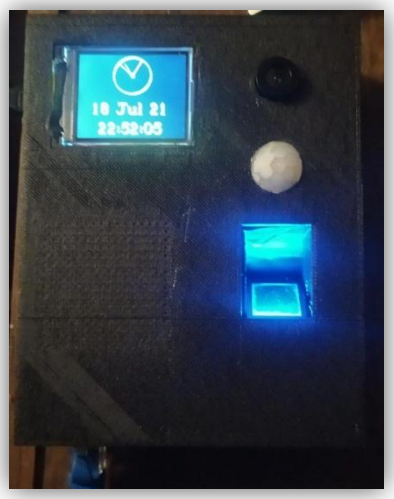


Figura. 3.19 Diseño del Case de las placas
FUENTE: [Elaboración Propia]

En la figura 3.16, se diseñó una caja de modo que proteja y oculte los cables que se colocaran entre el Raspberry Pi 2 y sus respectivos módulos.

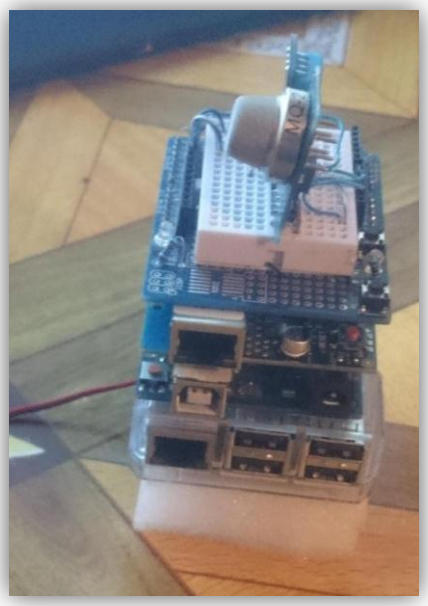


Figura. 3.20 Conexión de Raspberry Pi2 y el Arduino
FUENTE: [Elaboración Propia]

La integración de los dos dispositivos para que se comuniquen entre si junto con el módulo de detección de huella dactilar y la cámara, que vienen siendo los más importantes.

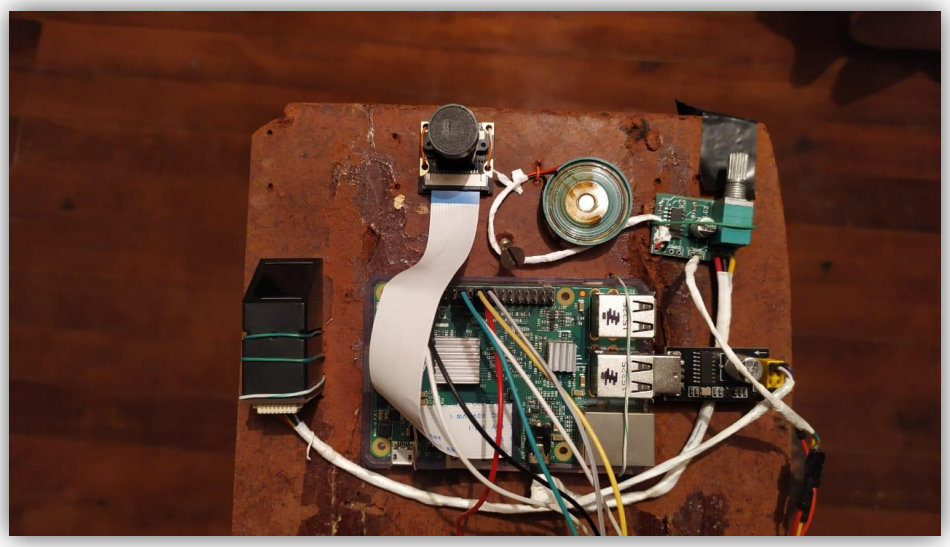


Figura. 3.21 Conexión completa de Prueba
FUENTE: [Elaboración Propia]

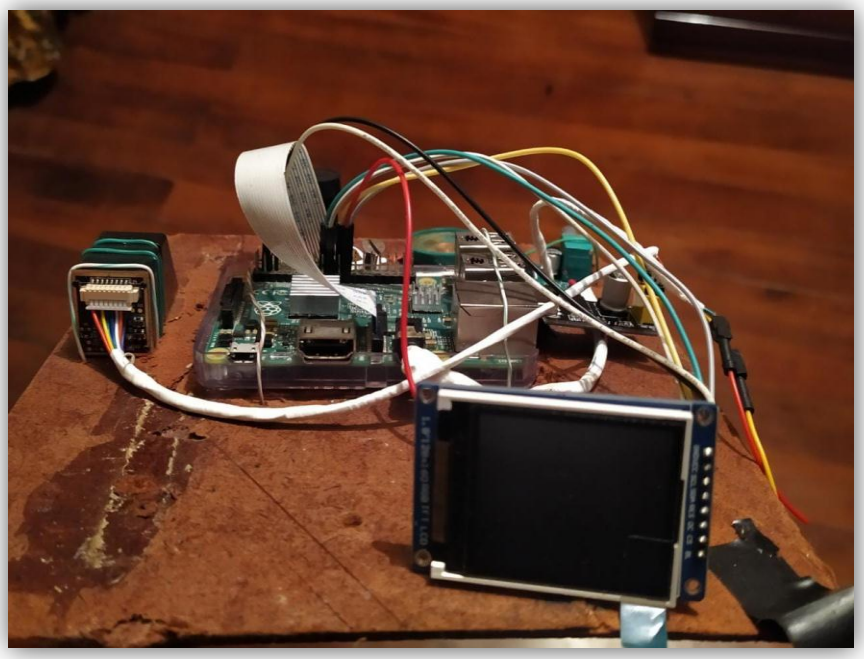


Figura. 3.22 Vista Frontal del Hardware
FUENTE: [Elaboración Propia]

Se aconseja que en la parte inferior del Raspberry Pi 2 se coloque un ventilador para que el dispositivo no caliente y así funciones correctamente, recordando que el Raspberry es una pequeña PC.

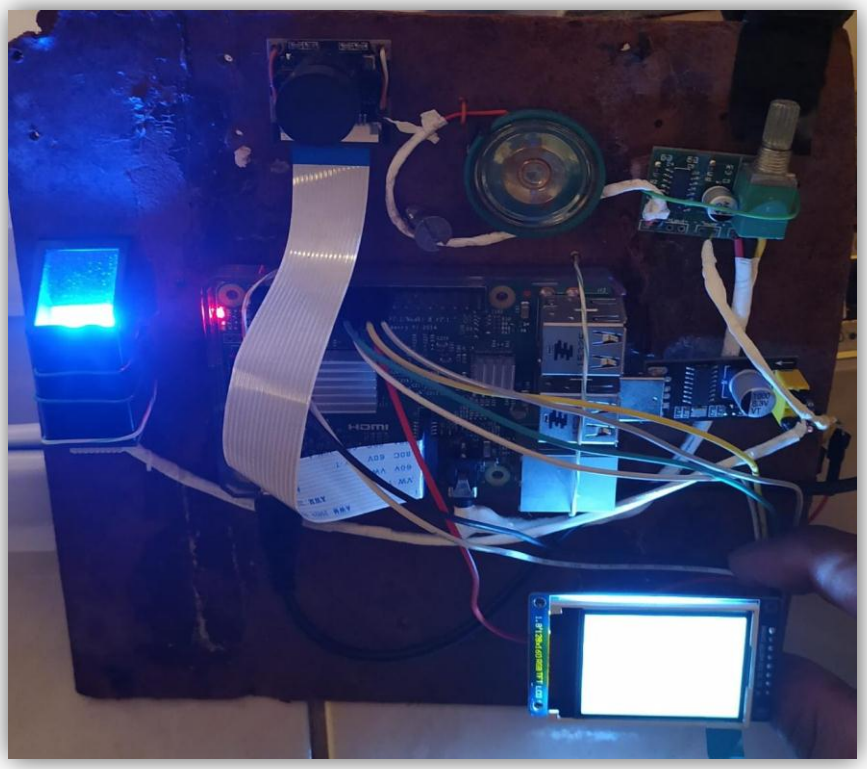


Figura. 3.23 Prueba de conexión del sistema biométrico
FUENTE: [Elaboración Propia]

La protección de la pantalla TFT del Raspberry pi debe ser protegida para prevenir que usuarios mal intencionados lo dañen o quieran manipular el sistema, por seguridad se debe colocar un vidrio templado u otro sistema de seguridad.

3.8.2. Prueba del sistema vía web

Se probará el producto final vía web, donde lo primero que se debe hacer es; dar un comando en nuestra Shell en Linux, para fines de prueba:

$ quasar dev

Luego de eso debemos de ingresar a al *url : [ip del raspberry]:8080,* este es la dirección del Raspberry PI:

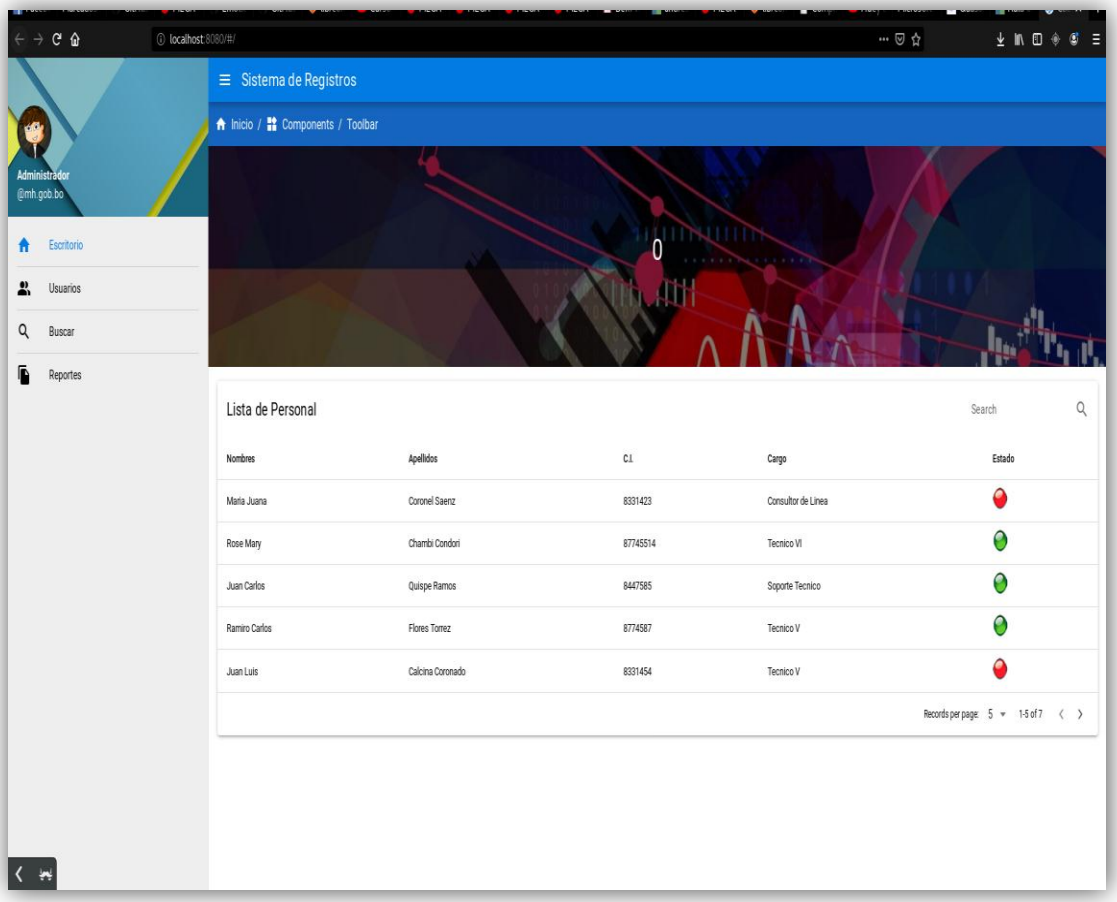


Figura. 3.24 Plantilla de Inicio del Sistema
FUENTE: [Elaboración Propia]

En la figura 3.21 podemos ver la página de inicio del sistema de consulta de marcados de todo el personal, donde además tienen varias funciones como ser añadido de nuevo personal, eliminación del nuevo personal, dar de baja, dar de alta, editar perfil, etc.

En la siguiente imagen vemos como es la vista de listado de personal de la institución:

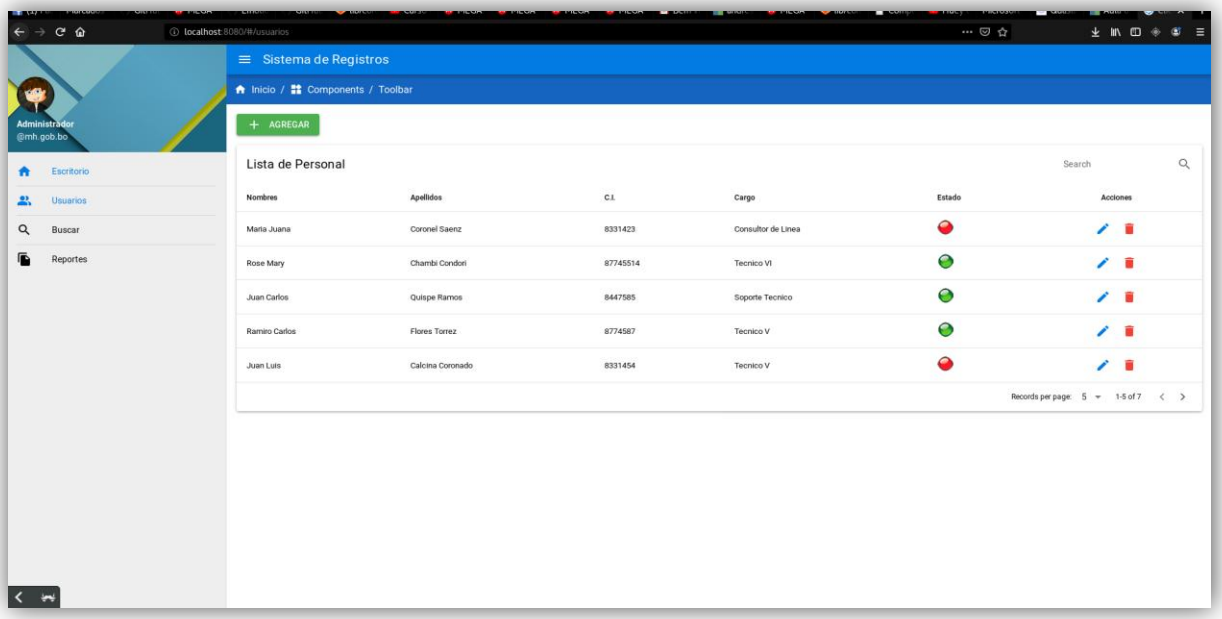


Figura. 3.25 Página de Listado de personal
FUENTE: [Elaboración Propia]

En la figura 3.22 para la buena administración y control del personal tenemos la vista para mostrar todo el personal y sus estados actuales.

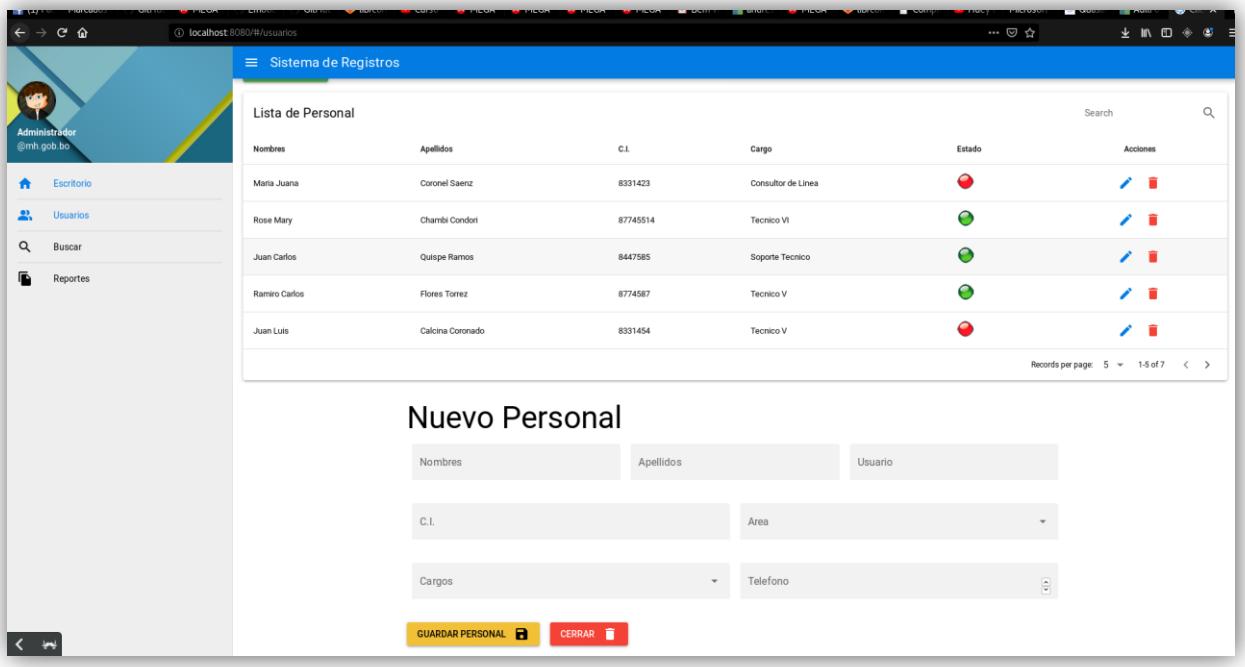


Figura. 3.26 Ingreso de Nuevo Personal
FUENTE: [Elaboración Propia]

Para el ingreso del nuevo personal tenemos un botón de color verde en la parte superior de la lista, cuando se da un clic nos muestra un formulario para el ingreso del nuevo personal.

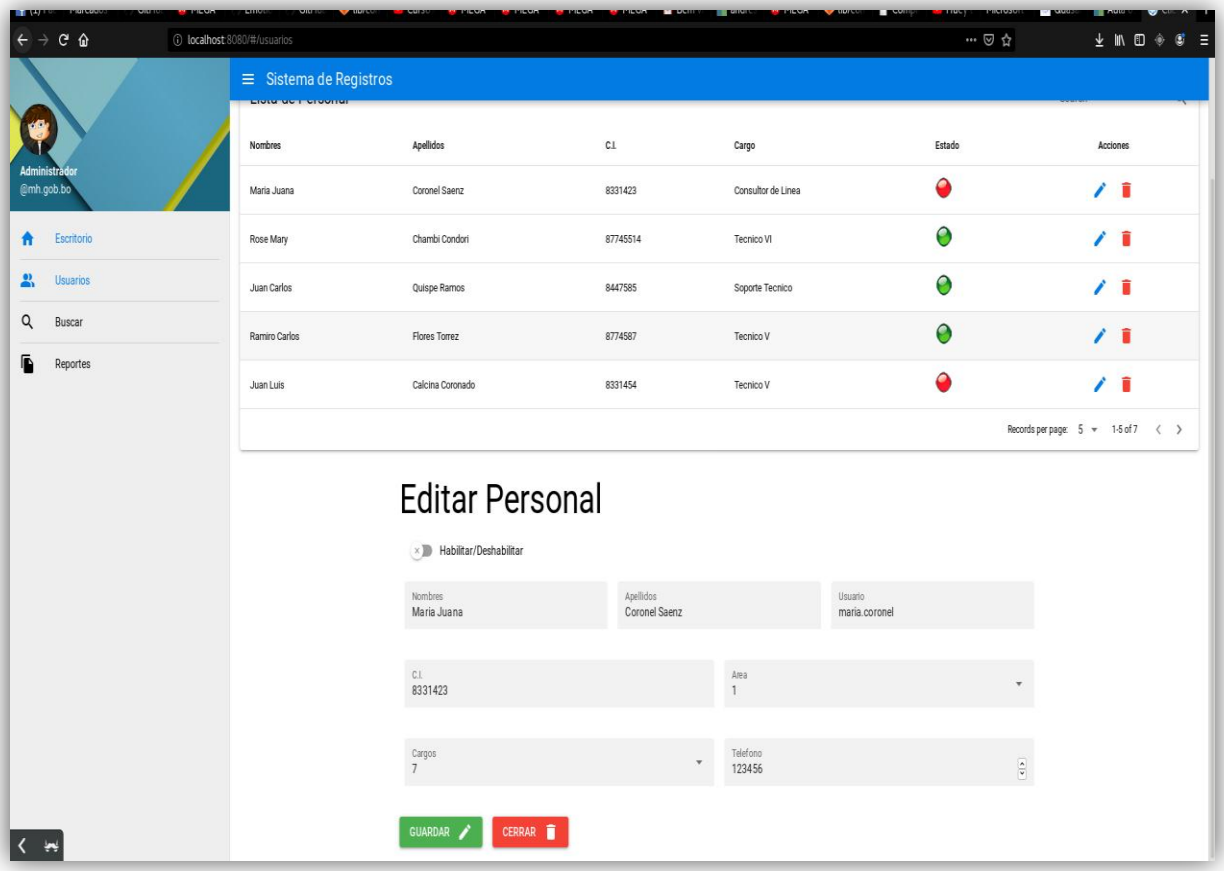


Figura. 3.27 Editar un Perfil
FUENTE: [Elaboración Propia]

En la figura 3.24, podemos ver que existe un formulario para editar el perfil actual de un usuario, este formulario aparecerá cuando le demos clic al icono de “pencil” de la tabla de usuarios.

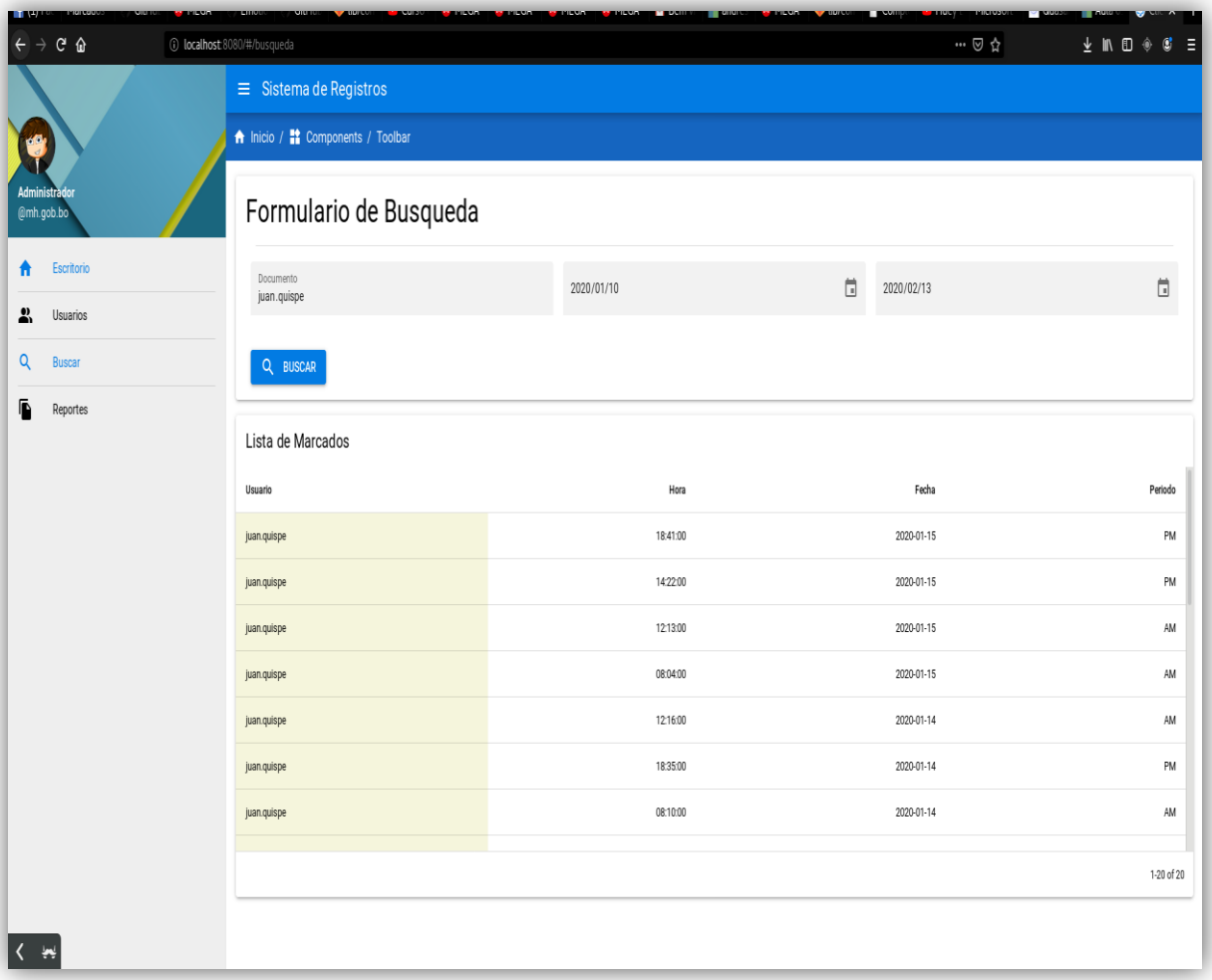


Figura. 3.28 Consulta de marcados
FUENTE: [Elaboración Propia]

En la figura 3.25, vemos una tabla donde nos muestra los marcados que realizo un usuario de una fecha de inicio a una fecha final (Esto se lo puede llenar en el formulario que se encuentra arriba de la tabla de marcados).

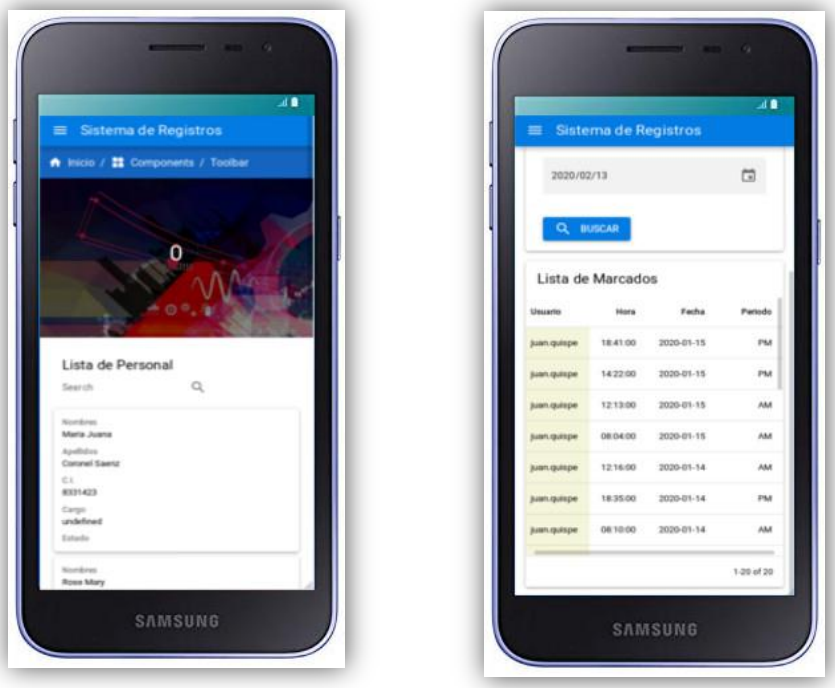


Figura. 3.29 Página Adaptada para Teléfonos Móviles
FUENTE: [Elaboración Propia]

El sistema también es responsivo ósea se adapta a un teléfono móvil, gracias a las tecnologías de frontend de Material Design junto a Quasar que tenemos la posibilidad de generar una APK para teléfonos móviles (Android y/o IOS) esto será en un futuro, por ahora solo tenemos la vista web SPA (Aplicación en una página simple).

CAPITULO IV

ANALISIS DE COSTOS

4.1 ANALISIS DE COSTOS

En esta sección se encuentra el análisis de costos del proyecto. Este se divide en el costo del desarrollo de software y el costo de construcción del hardware del prototipo.

4.1.1. Costo de Diseño Software

El análisis de costo de diseño del software se emplea el modelo constructivo de coste empírico. Según Boehm, quien es el desarrollador y está definido para tres tipos de proyectos: Orgánico, semi-acoplado y empotrado.
Las ecuaciones son las siguientes:

$$E=a_b * KLDC^{b_b} * FAE$$
$$D= C_b * E^{d_b}$$
$$N= E / D$$

Dónde:
E= Esfuerzo aplicado en personas-mes
KLDC=Número de líneas de código (en miles)
FAE=Factor de ajuste del esfuerzo
D=Tiempo de desarrollo (en meses)
N=Número de personas necesarias para el desarrollo del software
ab, bb , cb , db = Son coeficientes extraídos de la tabla 1

Proyecto de Software	ab	bb	cb	db
Orgánico	3,2	1,05	2,5	0,38
Semi-orgánico	3	1,12	2,5	0,35
Empotrado	2,8	1,2	2,5	0,32

Tabla. 1 coeficientes ab, bb , cb , db según el tipo de proyecto

El Factor de ajuste del esfuerzo FAE se determina multiplicando 15 factores extraídos de la tabla 2.

FAE = Fr * Tbd * Cp * Rt * Ra * Vmv * Tr * Ca * Ea * Cpr * Eso * El * Pm * Us * L

Conductores de coste	Valoración					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extr. Alto
Fiabilidad requerida del software(Fr)	0,75	0,88	1,00	1,15	1,40	-
Tamaño de la base de datos(Tbd)	-	0,94	1,00	1,08	1,16	-
Complejidad del producto (Cp)	0,70	0,85	1,00	1,15	1,30	1,65
Restricciones del tiempo de ejecución (Rt)	-	-	1,00	1,11	1,30	1,66
Restricciones del almacenamiento principal(Ra)	-	-	1,00	1,06	1,21	1,56
Volatilidad de la maquina virtual(Vmv)	-	0,87	1,00	1,15	1,30	-
Tiempo de respuesta del ordenador(Tr)	-	0,87	1,00	1,07	1,15	-
Capacidad del analista(Ca)	1,46	1,19	1,00	0,86	0,71	-
Experiencia en la aplicación(Ea)	1,29	1,13	1,00	0,91	0,82	-
Capacidad de los programadores(Cpr)	1,42	1,17	1,00	0,86	0,70	-
Experiencia en S.O. utilizado(Eso)	1,21	1,10	1,00	0,90	-	-
Experiencia en el lenguaje de programación(El)	1,14	1,07	1,00	0,95	-	-
Prácticas de programación modernas(Pm)	1,24	1,10	1,00	0,91	0,82	-
Utilización de herramientas software(Us)	1,24	1,10	1,00	0,91	0,83	-
Limitaciones de planificación del proyecto(L)	1,23	1,08	1,00	1,04	1,10	-

Tabla 2. Valoración de los conductores de coste para determinar la FAE.

4.1.1.1. Análisis de costo de diseño de Software

Se considera como un proyecto de software semi-acoplado para el análisis de costo.

FAE = 1 * 0.94 * 1 * 1 * 1 * 1 * 1.15 * 1 *0.91 * 1 * 1 * 1 * 1.1 * 1 * 1.08

FAE = 1.16

$$KLCD = \frac{420}{1000}$$

$$KLCD = 0.420$$

$$\mathrm{E} = 3 * 0.420^{1.12} * 1.16$$

$$\mathrm{E} = 1.31\ [personas - mes]$$

$$\mathrm{D} = 2.5 * 1.31^{0.35}$$

$$\mathrm{D} = 2.74\ [meses]$$

$$\mathrm{N} = \frac{1.31}{2.74} = 0.47 \approx 1[personas]$$

Para determinar el costo del software:

$$C_s = E * I_p$$

Dónde:

E=Proviene de las ecuaciones de Cocomo

Ip= Remuneración mensual de un profesional inicial en el área.

Cs = 1.31 * 2100 = 2751 Bs

4.1.2. Costo en la Construcción del Hardware

En las tablas se muestra la lista de componentes empleados, precios unitarios y cantidades, además de elementos adicionales en la construcción.

Nro.	**Componente**	Valor / Modelo	Precio unitario(Bs)	Cantidad	Precio Total(Bs)
1	**Raspberry Pi 2**	Pi 2	400	1	350
2	**Sensor Óptico de Huella digital UART**	AS608	150	1	150
3	**Cámara de vigilancia visión nocturna 5MP**	5MP	150	1	150
4	**Pantalla táctil de 3.5 pulgadas**	3.5 pulgadas	140	1	140
5	**Resistores**	330Ω / 1/4W	0.30	10	3
6	**Resistores**	1kΩ / 1/4W	0.30	5	1.50
7	**Resistores**	10kΩ / 1/4W	0.30	5	.50
8	**Cargadores**	5v, 2 A DC	30	2	60
9	**Otros**	Cables, pasta para soldar, estaño, parlante	40	1	40
TOTAL					896

Tabla 3. Costo total de los componentes del Proyecto

El costo total de los componentes y elementos que intervienen en la construcción del prototipo, más los módulos externos para realizar pruebas es de 896 Bs.

4.1.3. Costo de licencias de Programas empleados

En esta sección de costo de licencias empleadas, no se considera un costo por que todos los programas, frameworks, lenguajes, base de datos, sistema operativo y dispositivos son Open Source y no se paga licencia.

4.1.4. Costo Final del Proyecto

El costo final se determinará mediante la siguiente ecuación:
CFp = Costo total diseño del software + Costo total de los componentes + Costo total de licencias.
CFp = 2751 + 896 + 0
CFp = 3647 Bs

Es posible aclarar que el costo final del proyecto es para un prototipo con fines académicos.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Luego de realizar el diseño, construcción y pruebas de funcionamiento con el prototipo del sistema y el análisis económico, se destacan las siguientes conclusiones:

Es posible acceder a la tarjeta Raspberry Pi 2, mediante el protocolo de seguridad SCP para poder administrarlo remotamente.

La tarjeta Raspberry Pi 2, puede lograr una comunicación con la tarjeta Arduino y otros, utilizando los puertos digitales de los GPIOS del Raspberry Pi 2.

Si bien el costo de desarrollo del sistema como prototipo es elevado, la reproducción del mismo demanda un esfuerzo menor muy significativo y proporcional a la disminución del costo final planteado.

5.2 RECOMENDACIONES

Al momento de conectar la tarjeta Raspberry pi2 con otros dispositivos, se debe tener en cuenta que la tierra va conectada a ambas en común.

Se debe tomar en cuenta que la tarjeta del RBP trabaja con un mínimo de 5 (Voltios) y máximo 5.2 (Voltios), con una corriente mínima de 1(A) a medida que se coloca más dispositivos en sus puertos USB se debe de aumentar la fuente de corriente, por tal motivo se recomienda adquirir cargadores originales para este dispositivo.

Se recomienda colocar un pequeño ventilador a la placa Raspberry Pi 2, ya que con varios procesos que realizara dicha tarjeta los microprocesadores que tienen en su placa se calienta, de esta forma es muy recomendable colocar ventiladores o enfriadores a la tarjeta.

5.3 REFERENCIAS BIBLIOGRAFICA

[1] Tapiador M., Sigüenza J A. 2005, *"Tecnologías biométricas aplicadas a la seguridad"*

[2] Leytn, Q. E. (2007). *Diseño de una instalación domótica en un condominio para el control de seguridad e iluminación mediante la tecnología LonWOrks*.

[3] Schott Heriquez, M. R. (2005). *Investigación y desarrollo sistema prototipo de asistencia domótica para personas con movilidad limitada.*

[4] Martín Larrauri, J. (2013). *Sistema domótica para una casa inteligente*.

[5] Gómez Flores V. A. (2014). *Sistemas de control de Control de iluminación con protocolo de control domótica estandarizado*.

[6] Castillo, L. R. (2008), Diseño de infraestructura de telecomunicaciones para un data center.

[7] Evans, W. B. (2011), *Arduino Programming Notebook.*

[8] Lledo Sánchez, E. (2012). *Diseño de un sistema de control domótica basado en la plataforma de Arduino.*

[9] Carrillo Flores, R. A. (2015). *Diseño e implementación de un prototipo (DOMSYSTEM) de seguridad y control para mantener el resguardo de bienes y el confort mediante una red de sensores utilizando comunicación Wireless Bluethooth.*

5.3.1 WEBGRAFIA

[1] https://n9.cl/k89t1

[2] http://www.prometec.net/modulo-1/

[3] http://www.quiminet.com/articulos/que-es-la-automatizacion-27058.htm

[4] http://www.prometec.net/shield-ethernet/

[5] http://www.prometec.net/bt-hc06/

[6]https://www.beedigital.es/tendencias-digitales/historia-y-evolucion-del-reconocimiento-facial/

[7] http://www.dell.com/bo/empresas/p/servers

[8] http://www.dell.com/bo/empresas/p/enterprise-products

[9] http://www.lr21.com.uy/comunidad/1288818-antel-inauguro-el-data-center-mas-moderno-de-america-latina-donde-google-y-netflix-alojaran-sus-datos

[10] https://www.itsitio.com/ar/se-construye-datacenter-mas-grande-latinoamerica-colombia/

[11] https://www.monografias.com/trabajos76/automatizacion/automatizacion.shtml

[12] http://www.cienciaaldia.files.wordpress.com

[13] https://www.mobbeel.com/blog/historia-del-reconocimiento-facial/

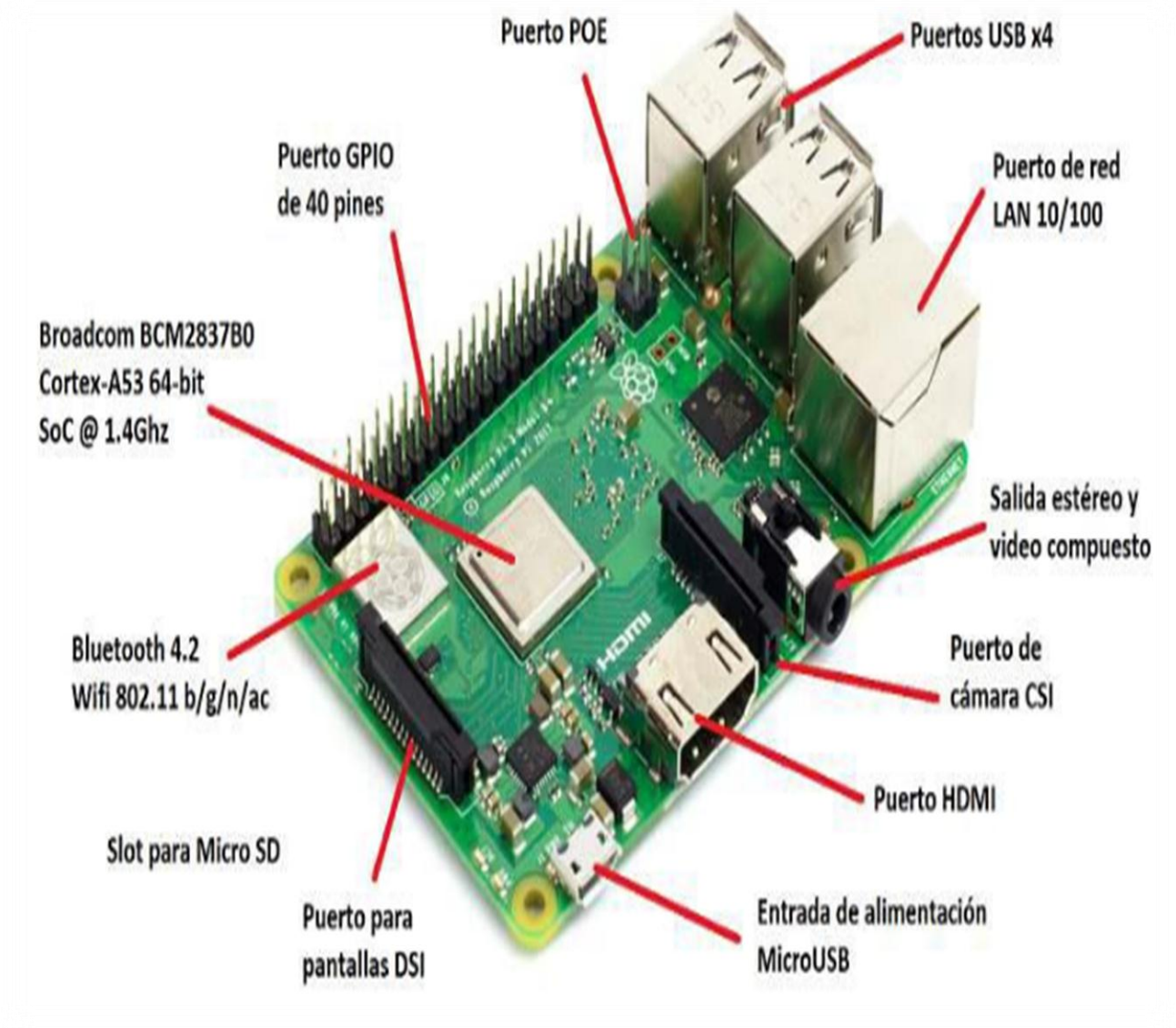
[14] https://serban.es/biometria-facial-el-sistema-de-reconocimiento-mediante-el-rostro/

[15] https://www.nanotech.com.mx/el-futuro-de-los-metodos-de-pagos-biometria-y-reconocimiento-facial/

ANEXO A

Especificaciones técnicas del Raspberry Pi 2

6.1 ESQUEMAS DE CONEXIONES



Especificaciones técnicas

Procesador	• Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memoria	• 1GB LPDDR2 SDRAM
Conectividad	• 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac Wireless LAN, Bluetooth 4.2, BLE • Gigabit Ethernet sobre USB 2.0 (rendimiento máximo de 300 Mbps) • 4 × Puertos USB 2.0
Acceso	• Cabecera GPIO de 40 pines
Video y Sonido	• 1 x HDMI • Puerto de pantalla DSI • MIPI CSI para módulo de cámara. • Salida de audio estéreo y puerto de video compuesto
Multimedia	• H.264, decodificación MPEG-4 (1080p30); Codificación H.264 (1080p30); OpenGL ES 1.1, gráficos 2.0
Soporte para tarjeta SD	• Formato MicroSD, para cargar el sistema operativo y el almacenamiento de datos.
Entrada de alimentación	• 5 V / 2A DC a través de conector micro USB • 5 V DC a través de cabecera GPIO • Alimentación a través de Ethernet (PoE): habilitada (requiere PoE HAT por separado)
Temperatura Operativa	• 0 – 50 º C

6.1.1 Definición de pines numeración de pines

En el firmware Particle, los pines están etiquetados de D0 a D15. Los números de pin de Broadcom, también conocidos como números de pin BCM o GPIO, también están disponibles de GPIO0 a GPIO27. Utilizado para E / S digital.

6.2 DIAGRAMA DE PIN OUT

Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
3.3V			1	X	X	2	5V		
I2C	GPIO2	SDA	3	X	X	4	5V		
	GPIO3	SCL	5	X	X	6	GND		
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
GND			9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14	GND		
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
3.3V			17	X	X	18	D11/A2	GPIO24	Digital I/O
SPI	GPIO10	MOSI	19	X	X	20	GND		
	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
GND			25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30	GND		
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34	GND		
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
GND			39	X	X	40	D16/A7	GPIO21	Digital I/O

Pin	Particle	Description
GPIO0		I2C data line used to identify Pi Hats (RESERVED FOR SYSTEM)
GPIO1		I2C clock line used to identify Pi Hats (RESERVED FOR SYSTEM)
GPIO2	SDA	I2C data line [2]
GPIO3	SCL	I2C clock line [2]
GPIO4	D0	Digital IO
GPIO5	D4	Digital IO
GPIO6	D5	Digital IO
GPIO7	CE1	SPI chip enable 1, digital IO
GPIO8	CE0	SPI chip enable 0, digital IO
GPIO9	MISO	SPI master-in slave-out [3]
GPIO10	MOSI	SPI master-out slave-in [3]
GPIO11	SCK	SPI clock [3]
GPIO12	D13/A4	Digital IO
GPIO13	D6	PWM-capable digital IO
GPIO14	TX	UART hardware serial transmit [1]
GPIO15	RX	UART hardware serial receive [1]
GPIO16	D14/A5	PWM-capable digital IO
GPIO17	D1	Digital IO
GPIO18	D9/A0	PWM-capable digital IO
GPIO19	D7	PWM-capable digital IO
GPIO20	D15/A6	Digital IO
GPIO21	D16/A7	Digital IO
GPIO22	D3	Digital IO
GPIO23	D10/A1	Digital IO
GPIO24	D11/A2	Digital IO
GPIO25	D12/A3	Digital IO
GPIO26	D8	Digital IO
GPIO27	D2	Digital IO

6.3 VISIÓN GENERAL

Esta página amplía las características técnicas de los pines GPIO disponibles en BCM2835 en general. Para ver ejemplos de uso, consulte Uso de GPIO. Al leer esta página, se debe hacer referencia a la hoja de datos de periféricos ARM BCM2835, sección 6.

Los pines GPIO se pueden configurar como entrada de propósito general, salida de propósito general o como una de hasta seis configuraciones alternativas especiales, cuyas funciones dependen de los pines.

Hay tres bancos GPIO en BCM2835.

Cada uno de los tres bancos tiene su propio pin de entrada VDD. En Raspberry Pi, todos los bancos GPIO se alimentan desde 3.3V. La conexión de un GPIO a un voltaje superior a 3.3V probablemente destruirá el bloque GPIO dentro del SoC.

Una selección de pines del Banco 0 está disponible en el encabezado P1 en Raspberry Pi.

6.3.1 Almohadillas GPIO

Las conexiones GPIO del paquete BCM2835 a veces se denominan en la hoja de datos de periféricos "pads", un término de diseño de semiconductores que significa "conexión de chip al mundo exterior".

Los pads son controladores de salida / búfer de entrada CMOS configurables push-pull. Los ajustes de control basados en registros están disponibles para:

Activar / desactivar pull-up / pull-down interno
Fuerza de transmisión de salida

Filtrado de disparador Schmitt de entrada

6.3.2 Estados de encendido

Todos los pines GPIO vuelven a las entradas de uso general en el reinicio de encendido. También se aplican los estados de extracción predeterminados, que se detallan en la tabla de funciones alternativas en la hoja de datos de periféricos ARM. La mayoría de los GPIO tienen aplicada una extracción predeterminada.

6.3.3 Interrupciones

Cada pin GPIO, cuando se configura como una entrada de propósito general, se puede configurar como una fuente de interrupción para el ARM. Se pueden configurar varias fuentes de generación de interrupciones:

Sensible al nivel (alto / bajo)
Borde ascendente / descendente
Flanco ascendente / descendente asincrónico
Las interrupciones de nivel mantienen el estado de interrupción hasta que el software del sistema borra el nivel (por ejemplo, al reparar el periférico adjunto que genera la interrupción).

La detección normal de flanco ascendente / descendente tiene una pequeña cantidad de sincronización incorporada en la detección. A la frecuencia del reloj del sistema, el pin se muestrea con el criterio para la generación de una interrupción siendo una transición estable dentro de una ventana de tres ciclos, es decir, un registro de '1 0 0' o '0 1 1'. La detección asíncrona evita esta sincronización para permitir la detección de eventos muy estrechos.

6.3.4 Funciones alternativas

Casi todos los pines GPIO tienen funciones alternativas. Se pueden

seleccionar bloques periféricos internos a BCM2835 para que aparezcan en uno o más de un conjunto de pines GPIO, por ejemplo, los buses I2C se pueden configurar en al menos 3 ubicaciones separadas. El control de almohadilla, como la fuerza de la unidad o el filtrado de Schmitt, todavía se aplica cuando el pin está configurado como una función alternativa.

6.3.7 Especificaciones de voltaje

La siguiente tabla proporciona las diversas especificaciones de voltaje para los pines GPIO.

6.3.4.1 Fuente de alimentación

Los requisitos de la fuente de alimentación varían según el modelo de Raspberry Pi. Todos los modelos requieren una alimentación de 5,1 V, pero la corriente suministrada generalmente aumenta según el modelo. Todos los modelos hasta el Raspberry Pi 3 requieren un conector de alimentación microUSB, mientras que el Raspberry Pi 4 usa un conector USB-C.

La cantidad exacta de corriente (mA) que requiere la Raspberry Pi depende de lo que le conecte. La siguiente tabla muestra varios requisitos actuales.

Producto	Capacidad actual de PSU recomendada	Consumo máximo de corriente periférico USB total	Consumo de corriente activo típico de placa desnuda
Raspberry Pi 2 Modelo B	1.8A	1.2A	350 mA

Raspberry Pi ha desarrollado sus propias fuentes de alimentación para usar con todos los modelos. Son fiables, utilizan cables de gran calibre y tienen un precio razonable.

Los requisitos de energía de la Raspberry Pi aumentan a medida que utiliza las diversas interfaces de la Raspberry Pi. Los pines GPIO pueden extraer 50 mA de forma segura, distribuidos entre todos los pines; un pin GPIO individual solo puede extraer 16 mA de forma segura. El puerto HDMI usa 50 mA, el módulo de la cámara requiere 250 mA, y los teclados y ratones pueden tomar tan solo 100 mA o más de 1000 mA. Verifique la potencia nominal de los dispositivos que planea conectar al Pi y compre una fuente de alimentación en consecuencia.

Si necesita conectar un dispositivo USB que cumpla con los requisitos de energía por encima de los valores especificados en la tabla anterior, entonces debe conectarlo a un concentrador USB con alimentación externa.

6.3.4.2 Advertencias sobre la fuente de alimentación

En todos los modelos de Raspberry Pi desde Raspberry Pi B + (2014) excepto el rango Zero, hay circuitos de detección de bajo voltaje que detectarán si el voltaje de suministro cae por debajo de 4.63V (+/- 5%). Esto resultará en un icono de advertencia que se mostrará en todas las pantallas adjuntas y se agregará una entrada al registro del kernel.

Si está viendo advertencias, debe mejorar la fuente de alimentación y / o el cable, ya que la baja potencia puede causar problemas con la corrupción de las tarjetas SD o un comportamiento errático de la propia Pi; por ejemplo, accidentes inexplicables.

Los voltajes pueden caer por una variedad de razones, por ejemplo, si la fuente de alimentación en sí es inadecuada, el cable de la fuente de alimentación está hecho de cables demasiado delgados o si ha conectado dispositivos USB de alta demanda.

6.3.5 Backpowering

La retroalimentación ocurre cuando los concentradores USB no proporcionan un diodo para evitar que el concentrador se encienda contra la computadora host. Otros

concentradores proporcionarán tanta potencia como desee en cada puerto. También tenga en cuenta que algunos concentradores retroalimentarán la Raspberry Pi. Esto significa que los concentradores alimentarán la Raspberry Pi a través de su cable de entrada de cable USB, sin la necesidad de un cable de alimentación micro-USB separado, y evitarán la protección de voltaje. Si está utilizando un concentrador que retroalimenta a la Raspberry Pi y el concentrador experimenta una subida de tensión, su Raspberry Pi podría dañarse.

6.3.6 USB

Los puertos USB permiten la conexión de periféricos como teclados, ratones, cámaras web que brindan al Pi una funcionalidad adicional.

xisten algunas diferencias entre el hardware USB de la Raspberry Pi y el hardware USB de las computadoras de escritorio o dispositivos portátiles / tabletas.

El puerto de host USB dentro del Pi es un host On-The-Go (OTG) ya que el procesador de aplicaciones que alimenta el Pi, BCM2835, originalmente estaba destinado a ser utilizado en el mercado móvil: es decir, como el único puerto USB en un teléfono para la conexión. a una PC oa un solo dispositivo. En esencia, el hardware OTG es más simple que el hardware equivalente en una PC.

En general, OTG admite la comunicación con todos los tipos de dispositivos USB, pero para proporcionar un nivel adecuado de funcionalidad para la mayoría de los dispositivos USB que se pueden conectar a un Pi, el software del sistema tiene que hacer más trabajo.

6.3.7 Dispositivos soportados

En general, todos los dispositivos compatibles con Linux se pueden usar con Pi, sujeto a algunas advertencias que se detallan más adelante. Linux tiene probablemente la base de datos de controladores más completa para hardware

heredado de cualquier sistema operativo (puede quedarse atrás para el soporte de dispositivos modernos, ya que requiere controladores de código abierto para que Linux reconozca el dispositivo de forma predeterminada).

Si tiene un dispositivo y desea usarlo con un Pi, conéctelo. Lo más probable es que "simplemente funcione". Si está ejecutando en una interfaz gráfica (como el entorno de escritorio LXDE en el sistema operativo Raspberry Pi), es probable que aparezca un icono o similar anunciando el nuevo dispositivo.

Si el dispositivo no parece funcionar, consulte la sección Solución de problemas a continuación.

6.3.8 Límites de potencia del puerto

Los dispositivos USB tienen requisitos de alimentación definidos, en unidades de 100 mA desde 100 mA hasta 500 mA. El dispositivo anuncia sus propios requisitos de energía al host USB cuando se conecta por primera vez. En teoría, la potencia real consumida por el dispositivo no debería exceder los requisitos establecidos.

Los puertos USB de una Raspberry Pi 1 tienen una carga de diseño de 100 mA cada uno, suficiente para controlar dispositivos de "bajo consumo", como ratones y teclados. Los dispositivos como adaptadores de LAN inalámbrica, discos duros USB, pendrives USB consumen mucha más corriente y deben alimentarse desde un concentrador externo con su propia fuente de alimentación. Si bien es posible conectar un dispositivo de 500 mA a una Raspberry Pi 1 y hacer que funcione con un suministro suficientemente potente, no se garantiza un funcionamiento confiable. Además, la conexión en caliente de dispositivos de alta potencia en los puertos USB de la Raspberry Pi 1 puede causar una caída de voltaje que puede hacer que la Pi se reinicie.

Desde la Raspberry Pi 2 en adelante, la potencia total suministrada a todos los puertos USB en total es de 1200 mA.

6.3.9 SPI

6.3.9.1 Visión general

La familia de dispositivos Raspberry Pi está equipada con varios buses SPI . SPI se puede utilizar para conectar una amplia variedad de periféricos: pantallas, controladores de red (Ethernet, bus CAN), UART, etc. Estos dispositivos se admiten mejor con controladores de dispositivos del kernel, pero la spidevAPI permite que los controladores del espacio de usuario se escriban en una amplia gama de idiomas.

6.3.9.2 Hardware

El núcleo BCM2835 común a todos los dispositivos Raspberry Pi tiene 3 controladores SPI:

SPI0, con dos selecciones de chip de hardware, está disponible en el encabezado de todos los Pis (aunque hay un mapeo alternativo que solo se puede usar en un módulo de cómputo.
SPI1, con tres selecciones de chip de hardware, está disponible en versiones de 40 pines de Pis.
SPI2, también con tres selecciones de chip de hardware, solo se puede usar en un módulo de cómputo porque los pines no se colocan en el encabezado de 40 pines.
BCM2711 agrega otros 4 buses SPI: SPI3 a SPI6, cada uno con 2 selecciones de chip de hardware. Todos están disponibles en el encabezado de 40 pines (siempre que nada más intente usar los mismos pines).

Asignaciones de pin / GPIO

SPI0 (disponible en encabezados J8 / P1 en todas las versiones de Rpi 2)

Función SPI	Pin de encabezado	Nombre de PIN de Broadcom	Función Pin Broadcom
HOLGAZANEAR	19	GPIO10	SPI0_MOSI
MISO	21	GPIO09	SPI0_MISO
SCLK	23	GPIO11	SPI0_SCLK
CE0	24	GPIO08	SPI0_CE0_N
CE1	26	GPIO07	SPI0_CE1_N

Modos maestros

Abreviaturas de nombres de señales

```
SCLK - Serial CLocK
CE   - Chip Enable (often called Chip Select)
MOSI - Master Out Slave In
MISO - Master In Slave Out
MOMI - Master Out Master In
```

Modo estándar

En el modo estándar SPI, el periférico implementa el protocolo serie estándar de 3 hilos (SCLK, MOSI y MISO).

Modo bidireccional

En el modo SPI bidireccional se implementa el mismo estándar SPI, excepto que se usa un solo cable para datos (MOMI) en lugar de los dos usados en el modo

estándar (MISO y MOSI). En este modo, el pin MOSI sirve como pin MOMI.

Modo LoSSI (interfaz serial de baja velocidad)

El estándar LoSSI permite la emisión de comandos a los periféricos (LCD) y la transferencia de datos hacia y desde ellos. Los comandos y parámetros de LoSSI tienen una longitud de 8 bits, pero se utiliza un bit adicional para indicar si el byte es un comando o un parámetro / dato. Este bit adicional se establece alto para un dato y bajo para un comando. El valor de 9 bits resultante se serializa en la salida. LoSSI se usa comúnmente con controladores LCD compatibles con MIPI DBI tipo C.

Tomar en cuenta:
Algunos comandos activan una lectura automática por parte del controlador SPI, por lo que este modo no se puede usar como un SPI multipropósito de 9 bits.

Modos de transferencia

- Encuestados
- Interrumpir
- DMA

6.3.9.3 Selección de chip

Los tiempos de configuración y retención relacionados con la afirmación y desactivación automática de las líneas CS cuando se opera en modo DMA son los siguientes:

- La línea CS se confirmará al menos 3 ciclos de reloj de núcleo antes del msb del primer byte de la transferencia.
- La línea CS se desactivará no antes de 1 ciclo de reloj central después del borde de salida del pulso de reloj final.

6.3.9.4 Software

Controlador de Linux

El controlador de Linux predeterminado es ahora el estándar spi-bcm2835.

SPI0 está deshabilitado de forma predeterminada. Para habilitarlo, use raspi-config , o asegúrese de que la línea dtparam=spi=onno esté comentada en /boot/config.txt. Por defecto usa 2 líneas de selección de chip, pero esto se puede reducir a 1 usando dtoverlay=spi0-1cs. dtoverlay=spi0-2cstambién existe, y sin ningún parámetro es equivalente dtparam=spi=on.

Para habilitar SPI1, puede utilizar 1, 2 o 3 líneas de selección de chip, agregando en cada caso:

```
dtoverlay = spi1-1cs # 1 selección de chip
dtoverlay = spi1-2cs # 2 selección de chip
dtoverlay = spi1-3cs # 3 selección de chip
```

Al archivo /boot/config.txt. Existen superposiciones similares para SPI2, SPI3, SPI4, SPI5 y SPI6.

El controlador no hace uso de las líneas de selección de chip de hardware debido a algunas limitaciones; en su lugar, puede usar un número arbitrario de GPIO como selecciones de chip de software / GPIO. Esto significa que puede elegir cualquier GPIO de repuesto como línea CS, y todas estas superposiciones SPI incluyen ese control; consulte los /boot/overlays/README detalles o ejecute (por ejemplo) dtoverlay -h spi0-2cs(dtoverlay -a | grep spipodría ser útil enumerarlos todos).

6.3.9.5 Velocidad

El controlador admite todas las velocidades, que son incluso divisores enteros

del reloj central, aunque, como se dijo anteriormente, no todas estas velocidades admitirán la transferencia de datos debido a los límites en los GPIO y en los dispositivos conectados. Como regla general, es poco probable que funcione cualquier valor superior a 50 MHz, pero su kilometraje puede variar.

6.3.9.6 Bits de modo admitidos

```
SPI_CPOL    - Clock polarity
SPI_CPHA    - Clock phase
SPI_CS_HIGH - Chip Select active high
SPI_NO_CS   - 1 device per bus, no Chip Select
SPI_3WIRE   - Bidirectional mode, data in and out pin shared
```

El módulo del kernel spi-bcm2835 admite el modo bidireccional o "3 hilos". Tenga en cuenta que en este modo, el campo tx o rx de la estructura spi_transfer debe ser un puntero NULL, ya que solo es posible la comunicación semidúplex. De lo contrario, la transferencia fallará. El código fuente de spidev_test.c no considera esto correctamente y, por lo tanto, no funciona en absoluto en el modo de 3 cables.

Bits por palabra admitidos

- 8 – Normal
- 9 - Esto es compatible con el modo LoSSI.

6.3.9.7 Modos de transferencia

El modo de interrupción es compatible con todos los buses SPI. SPI0 y SPI3-6 también admiten transferencias DMA.

DPI (interfaz de visualización paralela)

Una interfaz RGB paralela de hasta 24 bits está disponible en todas las placas

Raspberry Pi con el encabezado de 40 vías y los módulos de cómputo. Esta interfaz permite conectar pantallas RGB paralelas a la Raspberry Pi GPIO en RGB24 (8 bits para rojo, verde y azul) o RGB666 (6 bits por color) o RGB565 (5 bits rojo, 6 verde y 5 azul).

Esta interfaz está controlada por el firmware de la GPU y puede ser programada por un usuario a través de parámetros especiales config.txt y habilitando la superposición correcta del árbol de dispositivos de Linux.

GPIO pins

Una de las funciones alternativas que se pueden seleccionar en el banco 0 de Raspberry Pi GPIO es DPI (Display Parallel Interface), que es una interfaz paralela de reloj simple (hasta 8 bits de R, G y B; reloj, habilitación, hsync y vsync). Esta interfaz está disponible como función alternativa 2 (ALT2) en el banco GPIO 0:

GPIO	ALT Func2
GPIO0	PCLK
GPIO1	DE
GPIO2	LCD_VSYNC
GPIO3	LCD_HSYNC
GPIO4	DPI_D0
GPIO5	DPI_D1
GPIO6	DPI_D2
GPIO7	DPI_D3
GPIO8	DPI_D4
GPIO9	DPI_D5
GPIO10	DPI_D6
GPIO11	DPI_D7
GPIO12	DPI_D8
GPIO13	DPI_D9
GPIO14	DPI_D10
GPIO15	DPI_D11
GPIO16	DPI_D12
GPIO17	DPI_D13
GPIO18	DPI_D14
GPIO19	DPI_D15
GPIO20	DPI_D16
GPIO21	DPI_D17
GPIO22	DPI_D18
GPIO23	DPI_D19
GPIO24	DPI_D20
GPIO25	DPI_D21
GPIO26	DPI_D22
GPIO27	DPI_D23

Tenga en cuenta que hay varias formas en que los valores de color se pueden presentar en los pines de salida DPI en los modos 565, 666 o 24 bits (consulte la siguiente tabla y la output_formatparte del dpi_output_formatparámetro a continuación):

Mode	RGB bits	GPIO																							
		27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	565	-	-	-	-	-	-	-	-	7	6	5	4	3	7	6	5	4	3	2	7	6	5	4	3
3	565	-	-	-	7	6	5	4	3	-	-	7	6	5	4	3	2	-	-	-	7	6	5	4	3
4	565	-	-	7	6	5	4	3	-	-	-	7	6	5	4	3	2	-	-	7	6	5	4	3	-
5	666	-	-	-	-	-	-	7	6	5	4	3	2	7	6	5	4	3	2	7	6	5	4	3	2
6	666	-	-	7	6	5	4	3	2	-	-	7	6	5	4	3	2	-	-	7	6	5	4	3	2
7	888	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

6.3.10 Deshabilitar otros periféricos GPIO

Tenga en cuenta que todas las demás superposiciones de periféricos que utilizan pines GPIO en conflicto deben estar deshabilitadas. En config.txt, tenga cuidado de comentar o invertir cualquier dtparams que habilite I2C o SPI:

```
dtparam=spi=off

dtparam=i2c_arm=off
```

6.3.11 Controlar el formato de salida

El formato de salida (reloj, formato de color, polaridad de sincronización, habilitación) se puede controlar con un número mágico (entero sin signo o valor hexadecimal con el prefijo 0x) que se pasa al dpi_output_formatparámetro en config.txt creado a partir de los siguientes campos:

```
output_format          = (dpi_output_format >>  0) & 0xf;
rgb_order              = (dpi_output_format >>  4) & 0xf;

output_enable_mode     = (dpi_output_format >>  8) & 0x1;
invert_pixel_clock     = (dpi_output_format >>  9) & 0x1;

hsync_disable          = (dpi_output_format >> 12) & 0x1;
vsync_disable          = (dpi_output_format >> 13) & 0x1;
output_enable_disable  = (dpi_output_format >> 14) & 0x1;

hsync_polarity         = (dpi_output_format >> 16) & 0x1;
vsync_polarity         = (dpi_output_format >> 17) & 0x1;
output_enable_polarity = (dpi_output_format >> 18) & 0x1;

hsync_phase            = (dpi_output_format >> 20) & 0x1;
vsync_phase            = (dpi_output_format >> 21) & 0x1;
output_enable_phase    = (dpi_output_format >> 22) & 0x1;

output_format:
  1: DPI_OUTPUT_FORMAT_9BIT_666
  2: DPI_OUTPUT_FORMAT_16BIT_565_CFG1
  3: DPI_OUTPUT_FORMAT_16BIT_565_CFG2
  4: DPI_OUTPUT_FORMAT_16BIT_565_CFG3
  5: DPI_OUTPUT_FORMAT_18BIT_666_CFG1
  6: DPI_OUTPUT_FORMAT_18BIT_666_CFG2
  7: DPI_OUTPUT_FORMAT_24BIT_888

rgb_order:
  1: DPI_RGB_ORDER_RGB
  2: DPI_RGB_ORDER_BGR
  3: DPI_RGB_ORDER_GRB
  4: DPI_RGB_ORDER_BRG

output_enable_mode:
  0: DPI_OUTPUT_ENABLE_MODE_DATA_VALID
  1: DPI_OUTPUT_ENABLE_MODE_COMBINED_SYNCS
```

```
invert_pixel_clock:
  0: RGB Data changes on rising edge and is stable at falling edge
  1: RGB Data changes on falling edge and is stable at rising edge.

hsync/vsync/output_enable_polarity:
  0: default for HDMI mode
  1: inverted

hsync/vsync/oe phases:
  0: DPI_PHASE_POSEDGE
  1: DPI_PHASE_NEGEDGE
```

Nota: todos los campos de un solo bit actúan como un "comportamiento predeterminado de inversión".

6.3.12 Controlar tiempos y resoluciones

En el firmware con fecha de agosto de 2018 o posterior, la hdmi_timingsentrada config.txt que se usó anteriormente para configurar los tiempos de DPI ha sido reemplazada por un nuevo dpi_timingsparámetro. Si el dpi_timingsparámetro no está presente, el sistema volverá a utilizar el hdmi_timingsparámetro para garantizar la compatibilidad con versiones anteriores. Si ninguno de los dos está presente y se solicita un modo personalizado, se utiliza un conjunto de parámetros predeterminado para VGAp60.

Los parámetros dpi_groupy dpi_modeconfig.txt se utilizan para establecer modos predeterminados (modos DMT o CEA como los utiliza HDMI) o un usuario puede generar modos personalizados.

6.3.13 Superposiciones

Se utiliza una superposición de árbol de dispositivos de Linux para cambiar los

pines GPIO al modo correcto (función alternativa 2). Como se mencionó anteriormente, la GPU es responsable de impulsar la pantalla DPI. Por lo tanto, no existe un controlador de Linux; la superposición simplemente establece las funciones alt GPIO correctamente.

Se proporciona una superposición de DPI 'completa' (dpi24.dtb) que establece los 28 GPIO en modo ALT2, proporcionando los 24 bits completos del bus de color, así como la sincronización h y v, la habilitación y el reloj de píxeles. Tenga en cuenta que esto utiliza todos los pines GPIO del banco 0.

Se proporciona una segunda superposición (vga666.dtb) para controlar las señales del monitor VGA en el modo 666 que no necesita los pines de reloj y DE (GPIO 0 y 1) y solo requiere GPIO 4-21 para el color (usando el modo 5).

Estas superposiciones son bastante triviales y un usuario puede editarlas para crear una superposición personalizada para habilitar solo los pines necesarios para su caso de uso específico. Por ejemplo, si uno estaba usando una pantalla DPI usando vsync, hsync, pclk y de pero en modo RGB565 (modo 2), entonces la superposición dpi24.dtb podría editarse para que los GPIO 20-27 no se cambiaran al modo DPI y por tanto, podría utilizarse para otros fines.

CSI-2 (Interfaz serie de cámara 2) "Unicam"

Los SoC utilizados en la gama Raspberry Pi tienen dos interfaces de cámara que admiten fuentes CSI-2 D-PHY 1.1 o CCP2 (Compact Camera Port 2). Esta interfaz se conoce con el nombre en clave "Unicam". La primera instancia de Unicam admite 2 líneas de datos CSI-2, mientras que la segunda admite 4. Cada línea puede funcionar a hasta 1 Gbit / s (DDR, por lo que la frecuencia máxima de enlace es 500 MHz).

Sin embargo, las variantes normales de Raspberry Pi solo exponen la segunda instancia y enrutan solo 2 de los carriles de datos al conector de la cámara. La gama

del módulo de cómputo enruta todos los carriles de ambos periféricos.

6.4 INTERFACES DE SOFTWARE

Hay 3 interfaces de software independientes disponibles para comunicarse con el periférico Unicam:

6.4.1 Firmware

El firmware de la GPU de código cerrado tiene controladores para Unicam y tres sensores de cámara más un chip puente. Son la Cámara Raspberry Pi v1.3 (Omnivision OV5647), la Cámara Raspberry Pi v2.1 (Sony IMX219), la cámara Raspberry Pi HQ (Sony IMX477) y un controlador no compatible para el chip puente Toshiba TC358743 HDMI-> CSI2.

Este controlador integra el controlador de origen, Unicam, ISP y el control del sintonizador en una pila de cámaras completa que ofrece imágenes de salida procesadas. Se puede utilizar a través de MMAL, OpenMAX IL y V4L2 utilizando el módulo del kernel bcm2835-v4l2. Solo las cámaras Raspberry Pi son compatibles a través de esta interfaz.

6.4.2Componente rawcam MMAL

Esta era una opción provisional antes de que estuviera disponible el controlador V4L2. El componente MMAL vc.ril.rawcampermite recibir los datos CSI2 sin procesar de la misma manera que el controlador V4L2, pero toda la configuración de la fuente debe ser realizada por el área de usuario sobre cualquier interfaz que la fuente requiera. La aplicación raspiraw está disponible en github. Utiliza este componente y los conjuntos de registros I2C estándar para OV5647, IMX219 y ADV7282M para admitir la transmisión

6.4.3 V4L2

Hay un controlador de kernel de código abierto disponible para el bloque Unicam; este es un módulo del kernel llamado bcm2835-unicam. Esto interactúa con los controladores del subdispositivo V4L2 para que la fuente entregue las tramas sin procesar. Este controlador bcm2835-unicam controla el sensor y configura el receptor CSI-2 para que el periférico escriba las tramas sin procesar (después de Debayer) en SDRAM para que V4L2 las entregue a las aplicaciones. Excepto por esta capacidad para descomprimir los formatos CSI-2 Bayer a 16 bits / píxel, no hay procesamiento de imágenes entre la fuente de la imagen (por ejemplo, el sensor de la cámara) y bcm2835-unicam que coloca los datos de la imagen en SDRAM.

Mainline Linux tiene una variedad de controladores existentes. El árbol del kernel de Raspberry Pi tiene algunos controladores adicionales y superposiciones de árbol de dispositivos para configurarlos que han sido probados y confirmados para funcionar. Incluyen:

Dispositivo	Tipo	Notas
Omnivision OV5647	Cámara de 5MP	Cámara Raspberry Pi original
Sony IMX219	Cámara de 8MP	Revisión 2 de la cámara Raspberry Pi
Sony IMX477	Cámara de 12MP	Cámara Raspberry Pi HQ
Toshiba TC358743	Puente HDMI a CSI-2	
Dispositivos analógicos ADV728x-M	Video analógico al puente CSI-2	Sin soporte entrelazado
Infineon IRS1125	Sensor de profundidad de tiempo de vuelo	Apoyado por un tercero

Como el controlador del subdispositivo también es un controlador del kernel, con una API estandarizada, los terceros son libres de escribir el suyo propio para cualquier fuente que elijan.

Desarrollo de un controlador de terceros para bcm2835-unicam

Este es el enfoque recomendado para la interfaz a través de Unicam. Al desarrollar un controlador para un nuevo dispositivo destinado a ser utilizado con el módulo bcm2835-unicam, necesita el controlador y las superposiciones del árbol de dispositivos correspondientes. Idealmente, el controlador debe enviarse a la lista de correo de linux-media para la revisión del código y la fusión en la línea principal, luego trasladarse al árbol del kernel de Raspberry Pi , pero se pueden hacer excepciones para que el controlador sea revisado y combinado directamente con el kernel de Raspberry Pi.

Tenga en cuenta que todos los controladores del kernel tienen la licencia GPLv2, por lo que el código fuente DEBE estar disponible. El envío de módulos binarios únicamente constituye una violación de la licencia GPLv2 bajo la cual se licencia el kernel de Linux.

El bcm2835-unicam se ha escrito para tratar de acomodar todos los tipos de controladores fuente CSI-2 como se encuentran actualmente en el kernel principal de Linux. En términos generales, estos se pueden dividir en sensores de cámara y chips de puente. Los chips de puente permiten la conversión entre algún otro formato y CSI-2.

6.5 SENSORES DE CÁMARA

El controlador de sensor de un sensor de cámara es responsable de toda la configuración del dispositivo, generalmente a través de I2C o SPI. En lugar de escribir un controlador desde cero, a menudo es más fácil tomar un controlador existente como base y modificarlo según corresponda.

l controlador IMX219 es un buen punto de partida, la versión que se encuentra en el kernel 5.4 se puede encontrar aquí. Este controlador admite lectura Bayer de 8 bits y 10 bits, por lo que la enumeración de formatos y tamaños de cuadros es un poco más complicada.

Los sensores generalmente son compatibles con los controles de usuario V4L2 que se documentan aquí. No es necesario implementar todos estos controles en un controlador. El controlador IMX219 solo implementa un pequeño subconjunto, que se enumera a continuación, cuya implementación es manejada por la imx219_set_ctrlfunción.

En el caso del IMX219, muchos de estos controles se asignan directamente a las escrituras de registro del sensor mismo.

El árbol de dispositivos se utiliza para seleccionar el controlador del sensor y configurar parámetros como el número de carriles CSI-2, el funcionamiento continuo del carril del reloj y la frecuencia de enlace (a menudo, solo se admite uno). La superposición del árbol de dispositivos IMX219 para el kernel 5.4 se puede encontrar aquí

6.5.1 Chips de puente

Estos son dispositivos que convierten un flujo de video entrante, por ejemplo, HDMI o compuesto, en un flujo CSI-2 que puede ser aceptado por el receptor Raspberry Pi CSI-2.

El manejo de chips puente es más complicado, ya que, a diferencia de los sensores de la cámara, tienen que responder a la señal entrante e informar a la aplicación.

Los mecanismos para manejar chips de puente se pueden dividir ampliamente en analógicos o digitales.

Cuando se usa ioctlsen las secciones siguientes, una _S_en el ioctlnombre significa que es una función de conjunto, mientras que _G_es una función de obtención y _ENUMenumera un conjunto de valores permitidos.

6.5.2 Fuentes de video analógicas

Las fuentes de video analógicas utilizan el estándar ioctlspara detectar y establecer estándares de video. : VIDIOC_G_STD, VIDIOC_S_STD, VIDIOC_ENUMSTD, YVIDIOC_QUERYSTD

La selección del estándar incorrecto generalmente dará como resultado imágenes corruptas. Por lo general, establecer el estándar también establecerá la resolución en la cola CAPTURE de V4L2. No se puede configurar mediante VIDIOC_S_FMT. Generalmente, es una buena idea solicitar el estándar detectado a través de VIDIOC_QUERYSTDy luego configurarlo VIDIOC_S_STDantes de la transmisión.

6.5.3 Fuentes de video digital

Para fuentes de vídeo digitales, tales como HDMI, hay un conjunto alternativo de llamadas que permiten especificar de todos los parámetros de temporización digital (VIDIOC_G_DV_TIMINGS, VIDIOC_S_DV_TIMINGS, VIDIOC_ENUM_DV_TIMINGS, y VIDIOC_QUERY_DV_TIMINGS).

Al igual que con los puentes analógicos, los tiempos suelen fijar la resolución de la cola de CAPTURA de V4L2, y las llamadas VIDIOC_S_DV_TIMINGScon el resultado de VIDIOC_QUERY_DV_TIMINGSantes de la transmisión deben garantizar que el formato sea correcto.

Dependiendo del chip puente y del controlador, es posible que los cambios en la fuente de entrada se informen a la aplicación a través de VIDIOC_SUBSCRIBE_EVENTy V4L2_EVENT_SOURCE_CHANGE.

6.5.4 Dispositivos actualmente compatibles

Hay 2 chips de puente que son actualmente compatibles con el kernel de Linux Rasberry Pi, Analog Devices ADV728x-M para fuentes de video analógicas y Toshiba TC358743 para fuentes HDMI.

ANEXO B
CODIGO DEL SERVIDOR WEB EN EL RASPBERRY PI 2

DESARROLLO FRONTEND

1. package.json (con sus dependencias)

```
{
  "name": "client_registro",
  "version": "0.0.1",
  "description": "ESte es un proyecto para consumir los datos del backend del registro de asistencia y los maracos biometricos",
  "productName": "Cliente Registro App",
  "cordovaId": "org.cordova.quasar.app",
  "capacitorId": "",
  "author": "ximena.chambi <ximeemi89@gmail.com>",
  "private": true,
  "scripts": {
    "lint": "eslint --ext .js,.vue src",
    "test": "echo \"No test specified\" && exit 0"
  },
  "dependencies": {
    "@quasar/extras": "^1.0.0",
    "axios": "^0.18.1",
    "quasar": "^1.0.0"
  },
  "devDependencies": {
    "@quasar/app": "^1.0.0",
    "@vue/eslint-config-standard": "^4.0.0",
    "babel-eslint": "^10.0.1",
    "eslint": "^5.10.0",
    "eslint-loader": "^2.1.1",
    "eslint-plugin-vue": "^5.0.0"
  },
  "engines": {
    "node": ">= 10.18.1",
    "npm": ">= 6.13.4",
    "yarn": ">= 1.21.1"
  },
  "browserslist": [
    "last 1 version, not dead, ie >= 11"
  ]
}
```

2. Index.vue

```
<template>
  <div>
    <div class="col-6">
      <q-img  src="~assets/img/bg.jpg" style="height: 250px;">
        <div class="absolute-full text-subtitle2 flex flex-center">
          <p class="text-h4">{{count}}</p>
          <br />
        </div>
      </q-img>

      <div class="q-pa-md">
```

```
      <q-table
        :grid="$q.screen.xs"
        title="Lista de Personal"
        :data="lista"
        :columns="columns"
        row-key="name"
        :filter="filter"
      >
      <template v-slot:body="props">
        <q-tr :props="props">
          <q-td key="nombres" :props="props">{{ props.row.nombres }}</q-td>
          <q-td key="apellidos" :props="props">{{ props.row.apellidos }}</q-td>
          <q-td key="documento" :props="props">{{ props.row.documento }}</q-td>
          <q-td key="cargo" :props="props">{{ props.row.cargo.nombre }}</q-td>
          <q-td key="estado" :props="props">
            <img v-if="props.row.estado" src="~assets/img/verdes.png" alt="">
            <img v-if="!props.row.estado" src="~assets/img/rojo.png" alt="">
          </q-td>
        </q-tr>
      </template>
        <template v-slot:top-right>
          <q-input borderless dense debounce="300" v-model="filter" placeholder="Search">
            <template v-slot:append>
              <q-icon name="search" />
            </template>
          </q-input>
        </template>
      </q-table>
    </div>
  </div>
 </div>
</template>

<script>
import { mapState, mapMutations } from 'vuex'

export default {
  name: 'PageIndex',
  computed: {
    ...mapState(['count', 'lista'])
  },
  created () {
  },
  mounted () {
    this.$store.dispatch('loadCoins')
  },
  data () {
    return {
      filter: '',
      columns: [
        { name: 'nombres', label: 'Nombres', align: 'left', field: 'nombres', sortable: true },
        { name: 'apellidos', align: 'left', label: 'Apellidos', field: 'apellidos', sortable: true },
        { name: 'documento', align: 'left', label: 'C.I.', field: 'documento', sortable: true },
        { name: 'cargo', align: 'left', label: 'Cargo', field: 'cargo' },
        { name: 'estado', align: 'left', label: 'Estado', field: 'estado' }
      ]
    }
```

```
  },
  methods: {
    ...mapMutations(['increment'])
  }
}
</script>
```

3. Usuarios.vue

```
<template>
  <div>
    <div class="col-6">
      <div class="q-pa-md">
        <q-btn icon="add" label="Agregar" color="green" @click="formNuevo()" class="q-mb-md" />
        <q-table
          :grid="$q.screen.xs"
          title="Lista de Personal"
          :data="lista"
          :columns="columns"
          row-key="name"
          :filter="filter"
        >
        <template v-slot:body="props">
          <q-tr :props="props">
            <q-td key="nombres" :props="props">{{ props.row.nombres }}</q-td>
            <q-td key="apellidos" :props="props">{{ props.row.apellidos }}</q-td>
            <q-td key="documento" :props="props">{{ props.row.documento }}</q-td>
            <q-td key="cargo" :props="props">{{ props.row.cargo.nombre }}</q-td>
            <q-td key="estado" :props="props">
              <img v-if="props.row.estado" src="~assets/img/verdes.png" alt="">
              <img v-if="!props.row.estado" src="~assets/img/rojo.png" alt="">
            </q-td>
            <q-td key="acciones" :props="props">
              <q-btn flat round color="primary" icon="edit" @click="formEditar(props.row.id)" />
              <q-btn flat round color="red" icon="delete" @click="deleteUser(props.row.id)"/>
            </q-td>
          </q-tr>
        </template>

          <template v-slot:top-right>
            <q-input borderless dense debounce="300" v-model="filter" placeholder="Search">
              <template v-slot:append>
                <q-icon name="search" />
              </template>
            </q-input>
          </template>
        </q-table>
      </div>

      <div class="q-pa-md" v-if="nuevoForm">
        <FormularioNuevo />
      </div>
      <div class="q-pa-md" v-if="editarForm">
        <FormularioEditar />
      </div>
```

```
    </div>
  </div>
</template>

<script>
import FormularioNuevo from '../components/formularioNuevo'
import FormularioEditar from '../components/formularioEditar'
import { mapState } from 'vuex'

export default {
  name: 'Usuarios',
  computed: {
    ...mapState(['lista', 'nuevoForm', 'editarForm'])
  },
  components: {
    FormularioNuevo,
    FormularioEditar
  },
  data () {
    return {
      filter: '',
      nuevo: false,
      columns: [
        { name: 'nombres', label: 'Nombres', align: 'left', field: 'nombres', sortable: true },
        { name: 'apellidos', align: 'left', label: 'Apellidos', field: 'apellidos', sortable: true },
        { name: 'documento', align: 'left', label: 'C.I.', field: 'documento', sortable: true },
        { name: 'cargo', align: 'left', label: 'Cargo', field: 'cargo' },
        { name: 'estado', align: 'left', label: 'Estado', field: 'estado' },
        { name: 'acciones', align: 'center', label: 'Acciones', field: 'acciones' }
      ]
    }
  },
  methods: {
    formNuevo () {
      this.$store.dispatch('nuevoPersonal', true)
    },
    formEditar (id) {
      this.$store.dispatch('editarPersonal', { estado: true, id: id })
    },
    deleteUser (id) {
      this.$q.dialog({
        title: 'Eliminiar Personal?',
        message: 'Usted eliminara de nuestro sistema y definitivo al siguiente personal, esta deacuerdo?',
        cancel: true,
        persistent: true
      }).onOk(() => {
        this.$axios.delete('http://localhost:8000/api/usuarios2/' + id)
          .then((response) => {
            this.$q.notify({
              message: 'Se elimino correctamente el registro',
              textColor: 'black',
              color: 'warning',
              icon: 'notification_important',
              position: 'top-right'
            })
            this.$store.dispatch('loadCoins')
          })
```

```
        .catch((error) => {
          this.$q.notify({
            message: 'Ocurrio un error al realizar la operación',
            color: 'red',
            icon: 'security',
            position: 'top-right'
          })
          console.log(error)
        })
    }).onCancel(() => {
      console.log('>>>> Cancel')
    })
  }
 }
}
</script>
```

4. Buscar.vue

```
<template>
 <div class="q-pa-md">
  <q-card class="my-card">
   <q-card-section>
    <div class="row">
     <p class="text-h4">Formulario de Busqueda</p>
    </div>
    <q-separator inset />
    <q-form
     @submit="marcados(datos)"
     class="q-gutter-md"
    >
     <div class="row q-pt-md">
      <div class ="col-xs-12 col-sm-4 col-md-4 q-px-sm">
        <q-input v-model="datos.documento" filled outlined label="Documento"  :rules="[val => !!val ||
'Este campo es requerido.']"/>
      </div>
      <div class ="col-xs-12 col-sm-4 col-md-4 q-px-sm">
       <q-input filled v-model="datos.dateI" mask="date" :rules="['date']">
        <template v-slot:append>
         <q-icon name="event" class="cursor-pointer">
          <q-popup-proxy ref="qDateProxy" transition-show="scale" transition-hide="scale">
           <q-date v-model="datos.dateI" @input="() => $refs.qDateProxy.hide()" />
          </q-popup-proxy>
         </q-icon>
        </template>
       </q-input>
      </div>
      <div class ="col-xs-12 col-sm-4 col-md-4 q-px-sm">
       <q-input filled v-model="datos.dateF" mask="date" :rules="['date']">
        <template v-slot:append>
         <q-icon name="event" class="cursor-pointer">
          <q-popup-proxy ref="qDateProxy" transition-show="scale" transition-hide="scale">
           <q-date v-model="datos.dateF" @input="() => $refs.qDateProxy.hide()" />
          </q-popup-proxy>
         </q-icon>
        </template>
```

```
            </q-input>
          </div>
        </div>
        <div class="q-px-sm ">
          <q-btn color="primary" icon="search" label="Buscar" type="submit" />
        </div>
      </q-form>
    </q-card-section>
  </q-card>
  <div class="q-mt-md" v-if="sw">
    <marcadoList />
  </div>
 </div>
</template>

<script>
import marcadoList from '../components/marcadoList'

export default {
 name: 'PageBuscar',
 components: {
  marcadoList
 },
 created () {
 },
 data () {
  return {
   sw: false,
   filter: '',
   datos: {
    documento: '',
    dateI: '2020-01-01',
    dateF: '2020-01-01'
   }
  }
 },
 methods: {
  marcados (datos) {
   this.sw = true
   this.$store.dispatch('marcadosList', datos)
  }
 }
}
</script>
```

5. Route.js

```
const routes = [
 {
  path: '/',
  component: () => import('layouts/MyLayout.vue'),
  children: [
   { path: '', component: () => import('pages/Index.vue') }
  ]
```

```
  },
  {
    path: '/usuarios',
    component: () => import('layouts/MyLayout.vue'),
    children: [
      { path: '', component: () => import('pages/Usuarios.vue') }
    ]
  },
  {
    path: '/busqueda',
    component: () => import('layouts/MyLayout.vue'),
    children: [
      { path: '', component: () => import('pages/Buscar.vue') }
    ]
  }
]

// Always leave this as last one
if (process.env.MODE !== 'ssr') {
  routes.push({
    path: '*',
    component: () => import('pages/Error404.vue')
  })
}

export default routes
```

DESARROLLO DE UN MICROSERVICIO PARA EL RECONOCIEMIENTO FACIAL

1. requirements.txt (librerias que se usaron en el proyecto)

```
click==6.7
Flask==0.12.2
itsdangerous==0.24
Jinja2==2.9.6
MarkupSafe==1.0
Werkzeug==0.12.2

####################

aniso8601==8.0.0
astroid==2.2.5
certifi==2019.3.9
chardet==3.0.4
click==6.7
Django==2.2.5
django-cors-headers==3.1.0
django-mysql==3.2.0
djangorestframework==3.10.3
djangorestframework-jwt==1.11.0
Flask==0.12.2
Flask-JWT==0.3.2
Flask-RESTful==0.3.7
```

hashids==1.2.0
html5lib==1.0.1
idna==2.8
imutils==0.5.2
isort==4.3.16
itsdangerous==0.24
Jinja2==2.9.6
lazy-object-proxy==1.3.1
mccabe==0.6.1
mysql-connector==2.2.9
mysqlclient==1.4.4
numpy==1.16.3
opencv-contrib-python==4.1.0.25
Pillow==5.4.1
psycopg2==2.7.7
psycopg2-binary==2.7.7
pygame==1.9.6
PyJWT==1.4.2
pylint==2.3.1
PyPDF2==1.26.0
pyserial==3.4
pytz==2019.2
queuelib==1.5.0
reportlab==3.5.17
requests==2.21.0
six==1.12.0
sqlparse==0.3.0
typed-ast==1.3.1
urllib3==1.24.3
virtualenv==16.7.5
webencodings==0.5.1
Werkzeug==0.12.2
wrapt==1.11.1
xhtml2pdf==0.2.3

2. Camera_pi.py

```
import io
import time
import requests
from datetime import datetime
# import picamera
from base_camera import BaseCamera
import cv2
import numpy as np
import os
import pygame
import logging
import threading

#from urllib.parse import urlencode
#from urllib.request import Request, urlopen
class MyThread(threading.Thread):

    def run(self):
```

```
        vcf = threading.active_count()
        if vcf <= 3:
            pygame.mixer.init()
            pygame.mixer.music.load("./mp3/coin.mp3")
            pygame.mixer.music.play()
            while pygame.mixer.music.get_busy() == True:
                continue
            r = requests.get('http://worldtimeapi.org/api/timezone/America/La_Paz')
            json_response = r.json()['datetime']
            hora = json_response.split("T")[1]
            tiempo = hora.split("-")[0]
            periodo = ""
            ahora = datetime.now()
            fecha = ahora.strftime("%Y-%m-%d")
            if hora.split(":")[0]>="14" and hora.split(":")[0]<="24" :
                periodo = "PM"
            else:
                periodo = "AM"
            post_fields = {
                "usuario": self.getName(),
                "hora": tiempo,
                "fecha": fecha,
                "periodo": periodo
                }
            url = 'http://localhost:8000/api/marcados/'
            x = requests.post(url, data = post_fields)
            print(post_fields)

        time.sleep(2)
        print(self.getName())


class Camera(object):

    def __init__(self):
        self.recognizer = cv2.face.LBPHFaceRecognizer_create()
        self.recognizer.read('trainer/trainer.yml')
        self.cascadePath = "xml/haarcascade_frontalface_default.xml"
        self.faceCascade = cv2.CascadeClassifier(self.cascadePath);

        self.font = cv2.FONT_HERSHEY_SIMPLEX
        self.id = 0
        self.cont = 0
        self.names = ["Admin"]
        r = requests.get('http://localhost:8000/api/personal/')
        json_response = r.json()
        for k in range (len(json_response)):
            self.names.append(json_response[k]['usuario'])
        #self.names = ['None', 'Andres', 'Ximena', 'Ilza', 'Z', 'W']
        self.cam = cv2.VideoCapture(0)
        self.cam.set(3, 640)
        self.cam.set(4, 480)
        self.minW = 0.1*self.cam.get(3)
        self.minH = 0.1*self.cam.get(4)

    def get_frame(self):
        ret, img = self.cam.read()
```

```
        img = cv2.flip(img, 1) # Flip vertically
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        faces = self.faceCascade.detectMultiScale(
            gray,
            scaleFactor = 1.2,
            minNeighbors = 5,
            minSize = (int(self.minW), int(self.minH)),
        )

        for(x,y,w,h) in faces:
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
            id, confidence = self.recognizer.predict(gray[y:y+h,x:x+w])
            # Check if confidence is less them 100 ==> "0" is perfect match
            if (confidence < 100):
                id = self.names[id]
                sw = round(100 - confidence)
                confidence = "  {0}%".format(sw)
                if sw > 55:
                    mythread = MyThread(name = id)
                    mythread.start()
            else:
                id = "unknown"
                confidence = "  {0}%".format(round(100 - confidence))
            cv2.putText(img, str(id), (x+5,y-5), self.font, 1, (255,255,255), 2)
            cv2.putText(img, str(confidence), (x+5,y+h-5), self.font, 1, (255,255,0), 1)
            ret, jpeg = cv2.imencode('.jpg', img)
            return jpeg.tobytes()

        ret, jpeg = cv2.imencode('.jpg', img)
        return jpeg.tobytes()
```

3. app.py

```
#!/usr/bin/env python
from importlib import import_module
import os
from flask import Flask, render_template, Response

# import camera driver
# if os.environ.get('CAMERA'):
#    Camera = import_module('camera_' + os.environ['CAMERA']).Camera
# else:
#    from camera import Camera

# Raspberry Pi camera module (requires picamera package)
from camera_pi import Camera

app = Flask(__name__)


@app.route('/')
def index():
    """Video streaming home page."""
    return render_template('index.html')
```

```
def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/video_feed')
def video_feed():
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(host='0.0.0.0', threaded=True)
```

DESARROLLO BACKEND SERVIDOR API REST

1. requirements.txt (archivo de requerimiento)

```
aniso8601==8.0.0
astroid==2.2.5
certifi==2019.3.9
chardet==3.0.4
click==6.7
Django==2.2.5
django-cors-headers==3.1.0
django-mysql==3.2.0
djangorestframework==3.10.3
djangorestframework-jwt==1.11.0
Flask==0.12.2
Flask-JWT==0.3.2
Flask-RESTful==0.3.7
hashids==1.2.0
html5lib==1.0.1
idna==2.8
imutils==0.5.2
isort==4.3.16
itsdangerous==0.24
Jinja2==2.9.6
lazy-object-proxy==1.3.1
mccabe==0.6.1
mysql-connector==2.2.9
mysqlclient==1.4.4
numpy==1.16.3
opencv-contrib-python==4.1.0.25
Pillow==5.4.1
psycopg2==2.7.7
psycopg2-binary==2.7.7
pygame==1.9.6
PyJWT==1.4.2
pylint==2.3.1
```

PyPDF2==1.26.0
pyserial==3.4
pytz==2019.2
queuelib==1.5.0
reportlab==3.5.17
requests==2.21.0
six==1.12.0
sqlparse==0.3.0
typed-ast==1.3.1
urllib3==1.24.3
virtualenv==16.7.5
webencodings==0.5.1
Werkzeug==0.12.2
wrapt==1.11.1
xhtml2pdf==0.2.3

2. settings.py

```
import os
from datetime import datetime
import datetime

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SECRET_KEY = 'of1ugf8hvw3!wms_lya5-4(!f1g3@cbf2k9m%_axdr_him-h5)'

DEBUG = True

ALLOWED_HOSTS = ['*']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
    'rest_framework_jwt',
    'apps.registro',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
]
ROOT_URLCONF = 'asistencia_backend.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'asistencia_backend.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'NOMBRE DE LA BASE DE DATOS',
        'USER': 'NOMBRE DEL USUARIO DE LA DB',
        'PASSWORD': 'PASSWORD DEL USUARIO',
        'HOST': 'HOST',
        'PORT': 'PUERTO DE POSTGRES',
    }
}


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


CORS_ORIGIN_ALLOW_ALL = True

LANGUAGE_CODE = 'es-BO'

TIME_ZONE = 'America/La_Paz'
```

```
USE_I18N = True

USE_L10N = True

USE_TZ = False

STATIC_PATH = os.path.join(BASE_DIR,'static')

STATIC_URL = '/static/' # You may find this is already defined as such.

STATICFILES_DIRS = (
    STATIC_PATH,
)

# PRA REALIZAR LA CONFIGURACION DE L JWT
JWT_AUTH = {
    'JWT_VERIFY_EXPIRATION': False,
    'JWT_LEEWAY': 0,
    'JWT_EXPIRATION_DELTA': datetime.timedelta(seconds=3600),
    'JWT_AUDIENCE': None,
    'JWT_ISSUER': None,
}
```

3. urls.py (Principal)

```
from django.contrib import admin
from django.urls import path, include, re_path

urlpatterns = [
    path('admin/', admin.site.urls),
    re_path(r'^', include('apps.registro.urls'))
]
```

4. urls.py (aplicación)

```
from django.urls import path, include
from .views import AreaList, MarcadoList, UsuarioList, PersonalList, Usuarios2List,
CargoList,Usuario2Detail, MarcadosList

urlpatterns = [
    # ////////////////////////////////////////
    # URLS DE LA PARTE DEL CLIENTE
    # ////////////////////////////////////////
    path('api/areas/', AreaList.as_view()),
    path('api/cargos/', CargoList.as_view()),
    path('api/marcados/', MarcadoList.as_view()),
    path('api/usuarios/', UsuarioList.as_view()),
    path('api/personal/', PersonalList.as_view()),
    path('api/usuarios2/', Usuarios2List.as_view()),
    path('api/usuarios2/<int:pk>/', Usuario2Detail.as_view()),
    path('api/marcados/<str:pk1>/<str:pk2>/<str:pk3>/', MarcadosList.as_view()),
]
```

5. models.py

```
from django.db import models

class Area(models.Model):
    nombre = models.CharField(max_length=100)
    sigla = models.CharField(max_length=100)

    def __str__(self):
        return self.nombre


class Cargo(models.Model):
    nombre = models.CharField(max_length=100)
    sigla = models.CharField(max_length=100)

    def __str__(self):
        return self.nombre


class Usuario(models.Model):
    nombres = models.CharField(max_length=50)
    apellidos = models.CharField(max_length=50)
    usuario = models.CharField(max_length=50)
    estado = models.BooleanField(default=True)
    documento = models.CharField(max_length=50)
    area = models.ForeignKey(Area, on_delete=models.CASCADE)
    cargo = models.ForeignKey(Cargo, on_delete=models.CASCADE)
    telefono = models.IntegerField()

    def __str__(self):
        return self.nombres + ' ' + self.apellidos


class Marcado(models.Model):
    usuario = models.CharField(max_length=50)
    hora = models.TimeField(default=None, blank=True, null=True)
    fecha = models.DateField(default=None, blank=True, null=True)
    periodo = models.CharField(max_length=10)

    def __str__(self):
        return self.usuario
```

6. views.py

```
from rest_framework.response import Response
from rest_framework.views import APIView
from rest_framework import status
from django.http import Http404
from .models import Area, Cargo, Usuario, Marcado
from .serializers import AreaSerializer, CargoSerializer, UsuarioSerializer, MarcadoSerializer,
PersonalSerializer, Usuario2Serializer

# LISTA DE LAS AREAS
```

```
class AreaList(APIView):

    def get(self, request, format=None):
        area = Area.objects.all()
        serializer = AreaSerializer(area, many=True)
        return Response(serializer.data)


# LISTA DE LAS CARGOS
class CargoList(APIView):

    def get(self, request, format=None):
        cargo = Cargo.objects.all()
        serializer = CargoSerializer(cargo, many=True)
        return Response(serializer.data)


# LISTA DE LAS USUARIOS
class UsuarioList(APIView):

    def get(self, request, format=None):
        usuario = Usuario.objects.all()
        serializer = UsuarioSerializer(usuario, many=True)
        return Response(serializer.data)


# LISTA DE LAS USUARIOS
class PersonalList(APIView):

    def get(self, request, format=None):
        usuario = Usuario.objects.all()
        serializer = PersonalSerializer(usuario, many=True)
        return Response(serializer.data)


# LISTA DE LOS MARCADOS
class MarcadoList(APIView):

    def get(self, request, format=None):
        marcado = Marcado.objects.all()
        serializer = MarcadoSerializer(marcado, many=True)
        return Response(serializer.data)

    def post(self, request, format=None):
        serializer = MarcadoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


# LISTA NUEVOS USUARIOS
class Usuarios2List(APIView):

    def get(self, request, format=None):
        usuario = Usuario.objects.all()
        serializer = Usuario2Serializer(usuario, many=True)
```

```
        return Response(serializer.data)

    def post(self, request, format=None):
        serializer = Usuario2Serializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


class Usuario2Detail(APIView):

    def get_object(self, pk):
        try:
            return Usuario.objects.get(pk=pk)
        except Snippet.DoesNotExist:
            raise Http404

    def get(self, request, pk,format=None):
        snippet = self.get_object(pk)
        serializer = Usuario2Serializer(snippet)
        return Response(serializer.data)

    def put(self, request, pk, format=None):
        snippet = self.get_object(pk)
        serializer = Usuario2Serializer(snippet, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request,pk, format=None):
        snippet = self.get_object(pk)
        snippet.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)


class MarcadosList(APIView):

    def get(self, request, pk1, pk2, pk3):
        snippets = Marcado.objects.filter(usuario=pk1, fecha__gt=pk2, fecha__lt=pk3).order_by('-fecha')
        serializer = MarcadoSerializer(snippets, many=True)
        return Response(serializer.data)
```

UNIVERSIDAD MAYOR DE SAN ANDRES
FACULTAD DE TECNOLOGIA
CARRERA ELECTRONICA Y TELECOMUNICACIONES

Proyecto de Grado:

“Diseño de un prototipo biométrico con reconocimiento facial y de huella para control del personal del Ministerio de Hidrocarburos”

Presentado por: Ximena Chambi Condori

Para optar el grado académico de Licenciado en Electrónica y Telecomunicaciones.

Nota Numeral……

Nota Literal……...

Ha sido……..

M.Sc. Luis Richard Márquez Gonzales

Director de la Carrera de Electrónica y Telecomunicaciones

Tutor: Lic. Julia Torrez Soria
Tribunal: Ing. José Arturo Marín Thames
Tribunal: M.Sc. Ing. Ricardo Iván Gottret Rios