UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Sistema de identificación de huella dactilares para el acceso a lugares restringidos y control de asistencia con protocolo de comunicación IEEE 802.11.

PRESENTADO POR:

FÉLIX ANTONIO PALACIOS ABARCA

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, MARZO 2016

UNIVERSIDAD DE EL SALVADOR

RECTOR INTERINO :

LIC. JOSÉ LUIS ARGUETA ANTILLÓN

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALETA DE AMAYA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

Sistema de identificación de huella dactilares para el acceso a lugares restringidos y control de asistencia con protocolo de comunicación IEEE 802.11.

Presentado por :

FÉLIX ANTONIO PALACIOS ABARCA

Trabajo de Graduación Aprobado por:

Docente Asesor :

Msc. e Ing. JOSÉ WILBER CALDERÓN URRUTIA

San Salvador, Marzo 2016

Trabajo de Graduación Aprobado por:

Docente Asesor :

Msc. e Ing. JOSÉ WILBER CALDERÓN URRUTIA

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, Miércoles 16 de marzo de 2016, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 10:00 a.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director

   Firma:

2. MSc. José Wilber Calderón Urrutia
   Secretario

   Firma: kWilber Calderón

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

Firma:

1- MSc. José Wilber Calderón Urrutia: kWilber Calderón

2- MSc. Salvador de Jesús German

3- Ing. Walter Leopoldo Zelaya Chicas

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

Sistema de identificación de huella dactilares para el acceso a lugares restringidos y control de asistencia con protocolo de comunicación IEEE 802.11.

A cargo del Bachiller:

Palacios Abarca Félix Antonio

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 6.41

(SEIS . CUATRO UNO)

Índice

Introducción *1*

Objetivos. *3*

Capítulo 1 : Marco teórico. *4*

1.1. Que es django *4*

1.2. Apache 2.0 ..... *7*

1.3. Bootstrap. *8*

1.4. Raspberry Pi 2.0. ..... *10*

1.5. Arduino Nano *11*

1.6. Sensor biométrico GT511C3. ..... *12*

Capítulo 2 : Configuración del sistema biométrico. *14*

2.1. Generalidades. *14*

2.2. Configuración inicial de la Raspberry Pi 2.0. ..... *15*

2.3. Instalación de django. *19*

2.4. Creación del proyecto Sistema en django. *21*

2.5. Creación de la base de datos en MySQL *25*

2.6. Contenido de las tablas creadas en MySQL *27*

2.7. Configuración de la red inalambrica *29*

2.8. Configuración del puerto Serial. *30*

2.9. Configuración del servidor Apache 2.0. ..... *33*

2.10. Configuración de los permisos MySQL *35*

Capítulo 3 : Programas del sistema. *36*

3.1. Generalidades. *36*

3.2. Raspberry Pi 2.0 ..... *36*

3.3. Programas que contiene la Raspberry Pi 2.0. ..... *36*

3.4. Programa que contine el Arduino Nano. *46*

Capítulo 4 : Diseño del circuito nivelador de tensión.....50
4.1. Diseño del PCB.....50
4.2. Esquema de conexión Arduino-Sensor.....51
Capítulo 5 : Resultados de la ejecución del proyecto. 52
5.1. Login de la aplicación Web.....52
5.2. Interfaz de control de la aplicación. 53
5.3. Vista registrar huella. 54
5.4. Vista registro de personas.....55
Presupuesto.....57
Concluciones.....58
Referencia Bibliográficas.....59

Índice de figuras

Figura 1.1 : Estructura de archivos del proyecto. *6*
Figura 1.2 : Página de descarga de Bootstrap. *8*
Figura 1.3 : Archivo de descarga Bootstrap. *9*
Figura 1.4 : Archivos del directorio Bootstrap. *9*
Figura 1.5 : Raspberry Pi 2.0. ..... *10*
Figura 1.6 : Distribución de pines de la Raspberry Pi 2.0. ..... *11*
Figura 1.7 : Distribución de pines del Arduino Nano. *12*
Figura 1.8 : Distribución de pines del sensor biométrico GT511C3. ..... *13*
Figura 2.1 : Dispositivo Sd montado. *16*
Figura 2.2 : Pantalla de primer arranque del sistema operativo. *17*
Figura 2.3 : Muestra el servidor de desarrollo corriendo correctamente. *23*
Figura 2.4 : Tablas en sistema biométrico. *26*
Figura 2.5 : Tabla empleado. *27*
Figura 2.6 : Tabla cargo *28*
Figura 2.7 : Tabla tiempo accesos. *28*
Figura 2.8 : Tabla tiempo de contratación *29*
Figura 4.1 : Vista superior de la placa PCB. *50*
Figura 4.2 : Vista inferior de la placa PCB. *51*
Figura 4.3 : Divisor de tensión. *51*
Figura 5.1 : Vista del login de la aplicación. *52*
Figura 5.2 : Vista principal de la aplicación. *53*
Figura 5.3 : Vista donde se piden datos del usuario. *54*
Figura 5.4 : Vista registro de personas *55*
Figura 5.5 : Vista editar, eliminar registro *56*

Introducción

El presente trabajo de graduación, muestra la implementación de un sistema de identificación y registro de huellas dactilares para el acceso a un lugar restringido, utilizando el protocolo de comunicación IEEE 802.11n, el cual consta de un Sensor Biométrico GT511C3, un Arduino Nano, una Raspberry Pi 2.0, una pantalla táctil LCD 3.5" y un módulo USB Wifi, que es el que implementa la conexión inalámbrica a la red de internet. Cada uno de los dispositivos tiene una configuración en particular que sirve para integrar los componentes de los que consta el sistema, siendo la Raspberry Pi 2.0 el elemento principal, el cual posee un sistema operativo Raspbian instalado, que es la plataforma de trabajo donde se desarrolla todo el entorno de programación. El entorno de programación para el desarrollo del aplicativo Web, se hizo mediante Django, el cual es un ambiente de trabajo que está hecho sobre la base de Python, este estructura el proyecto creado en una serie de directorios bien definidos, los cuales le dan la funcionalidad requerida. Dentro del cuerpo de este trabajo se explicará sobre la implementación y edición de cada uno de los archivos y directorios creados por Django. Los lenguajes de programación utilizados para implementar el control del hardware y la interfaz Web, fueron lenguaje C y Django - Python, este código es el que ayuda a interrogar y hacer peticiones de control sobre el sensor biométrico por medio de la inserción de éste a la tarjeta Arduino Nano, además es el que presenta una vista sobre el cambas de un navegador Web.

La idea principal de este sistema es proporcionar acceso a personas en lugares restringidos por medio de un registro de huella guardado en una base de datos, la cual en este caso es la de MySQL, que esta enlazada por medio de Django – Python para interrogar los campos de las tablas que conforman la funcionalidad del proyecto creado.

El sistema tiene la capacidad de registrar huellas de personas, las cuales se les asigna un identificador entero de entre 0 a 199 y almacenarlo en una base de datos instalada como es MySQl, además tiene la capacidad de almacenar nombres asociados al identificador generado, en la base de datos con el cual se puede habilitar el acceso hacia algún lugar específico.

El sistema puede mostrar hora y fecha de ingreso así como también hora y fecha de salida del acceso a un lugar específico y además formar el historial de la entrada-salida generado por un usuario especifico dado de alta en la base de datos.

Este sistema devolverá un pulso digital a 5V en la placa del Arduino Nano lo cual se controlará un relé que activará una chapa electrónica y de ese modo permitir el acceso hacia algún lugar específico, que para el prototipo presentado en este proyecto, está simulado por medio del parpadeo de un diodo led.

Objetivos

Objetivo General:

Construir un sistema seguro, confiable y capaz de reconocer huellas dactilares, llevar el control de entrada y salida de personas a un local.

Objetivos Específicos:

1. Utilizar el protocolo IEEE 802.11 para la comunicación del sistema con el administrador.

2. Elaborar una base de datos con huellas dactilares utilizando software y hardware de libre distribución.

3. Crear reportes del control de asistencia según lo solicite el administrador del sistema.

4. Ofrecer un sistema confiable tanto en la detección de huellas dactilares que resista posibles ataques de malware.

CAPITULO 1: Marco teórico

Generalidades:

1.1 ¿Que es Django?

Es un ambiente de trabajo, el cual posee todas las herramientas necesarias para integrar el desarrollo de la programación Web, utilizando código HTML y CSS e incrustando código Python a la vez. Django estructura la creación de un proyecto en una serie de archivos y directorios bien definidos de tal manera que se puedan editar y configurar de acuerdo al diseño particular de cada uno de los programadores, Django crea una plantilla de tal manera que únicamente se escriba el código necesario que le dé la funcionalidad al proyecto. Por el hecho de poseer un servidor de desarrollo se puede ejecutar cada uno de los proyectos generados sin tener que instalar un servidor de producción que le de funcionalidad a éste, que para el caso particular se utilizó Apache 2.0.

Por la creación de plantillas que este programa realiza ya no es necesario escribir el código desde cero, sino únicamente integrar parte del código necesario que utilizará el proyecto para funcionar correctamente.

Django estructura la creación del proyecto mediante el esquema ***Modelo, Plantilla y Vista***, que es la que da la interacción sobre cada uno de los directorios creados por el proyecto y presentadas como producto final a peticiones hechas por aplicaciones cliente como lo son los navegadores web.

- ✓ ***Modelo:*** Este define los datos almacenados, en forma de clases de Python, en un proyecto es donde se define el diseño y creación de las tablas en una base de datos configurada previamente en el archivo settings.py.

- ***Plantilla:*** La plantilla es básicamente una página HTML, es la que se presenta cuando un aplicativo cliente como un navegador Web realiza peticiones al servidor, esta puede contener además etiquetas propias de Django así como también código Css, Xml y jabascrip.

- ***Vista:*** Esta posee las funciones de Python, es la que define que plantillas se van a ejecutar, además en este caso es la que proporciona acceso al puerto serial de la Raspberry Pi 2.0, y recoge los datos recibidos para enviárselos a un formulario.

La configuración de las rutas:

Django organiza el proyecto en una serie de archivos y directorios donde residen, tanto las librerías del proyecto como las aplicaciones de este, que están enlazados por medio de URL's para servir una vista predeterminada. Django posee el control del proyecto por medio de las configuraciones de las URL's tanto del proyecto en sí, como el de las aplicaciones creadas.

Los archivos predeterminados que genera Django son:

Una parte muy importante es entender los archivos predeterminados que se generan con Django al crear el primer proyecto, ya que este se estructura por defecto como un contenido de librerías necesarias para la funcionalidad del proyecto y un contenido para crear y desarrollar aplicaciones, esta estructura de archivos se generan al escribir el siguiente comando en el Shell de Raspbian:

```
~$ django-admin.py startproject mysite
```

El cual genera un archivo y un directorio que se muestran en la *figura 1. 1:*

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Fig.1.1 estructura de archivos del proyecto

Como se pudo observar en la figura 1.1 se crea un directorio principal con el nombre *mysite*, este contiene un archivo y un directorio donde se guardan las librerías del primer proyecto con el nombre de manage.py y *mysite* respectivamente, encontrándose en el directorio *mysite* interno los siguientes archivos:

- ✓ ***__init__.py :*** Es un archivo vacío que le dice a Python que debe considerar este directorio como un paquete de Python.
- ✓ ***manage.py :*** Es un archivo que permite a Django interactuar entre el aplicativo Web y el terminal del sistema.
- ✓ ***settings.py :*** Este archivo contiene todas las configuraciones del proyecto creado.
- ✓ ***urls.py :*** Contiene las rutas que están disponibles en el proyecto y las aplicaciones, este archivo redirecciona las vistas hacia un navegador Web.

Todos estos archivos son generados en la creación del primer proyecto con Django.

Los archivos de la aplicación creados por Django son:

En su estructura del proyecto, Django crea una serie de archivos para administrar la aplicación, la cual se genera mediante la ejecución del comando

```
~$ python manage.py startapp aplicación.
```

La ejecución de este comando crea en el directorio principal del proyecto, el directorio de nombre *aplicación* el cual contiene los siguientes archivos:

aplicación/

- ✓ ***__init__.py :*** Al igual que en el proyecto principal, este es una archivo vacío que le dice a Python, que debe considerar este directorio como un paquete python .
- ✓ ***models.py :*** En este archivo se declaran las clases del modelo, es donde se crean las tablas en la base de datos.
- ✓ ***views.py :*** En este archivo se declaran las funciones de las vistas.

Estos archivos son generados en el momento que se crea la primera aplicación dentro del proyecto.

1.2 Apache 2.0:

Apache es un servidor Web de distribución libre, el cual aloja en su interior archivos HTML y Css para servirlos a programas cliente los cuales hacen peticiones por medio de una URL que indica la ruta donde residen los archivos, este responde mediante el envió de una vista presentada en el cambas de un navegador Web, por medio de la configuración del puerto 80 u 8080.

Este servidor tiene la particularidad de trabajar con código Python incrustado en páginas creadas en lenguaje HTML, mediante la instalación y configuración de los siguientes módulos:

- ➢ ***mod_wsgi*** - Permite al servidor interpretar y ejecutar código Python.
- ➢ ***mod_python*** - Páginas dinámicas en Python.
- ➢ ***mod_security*** - Filtrado a nivel de aplicación, para seguridad.

Puede servir, ya sea páginas Web de contenido estático o dinámico y además, puede configurarse para trabajar con bases de datos como MySQL.

1.3 Bootstrap:

Es un ambiente de desarrollo creado para facilitar el diseño de páginas Web, mediante la utilización de plantillas previamente establecidas, este utiliza las tecnologías HTML y Css para darle forma a las vistas presentadas en el cambas de un navegador Web, es soportado por la mayoría de navegadores Web, pero los que oficialmente lo están son ***Firefox*** y ***Opera.*** Existen varias formas de diseñar sitios Web con Bootstrap las cuales son:

1. Mediante la descarga de los códigos Css y JabaScrip compilados.
2. Mediante la descarga del código fuente.
3. Descargando el código fuente en formato Sass.

Siendo el más fácil de usar y el recomendado para iniciadores, el diseño mediante la descarga de los códigos ***Css y JabaScrip*** compilados, los cuales se descargan mediante dar click al botón Download Bootstrap de la siguiente URL http://getbootstrap.com/ el cual mostramos en la *figura 1.2.*

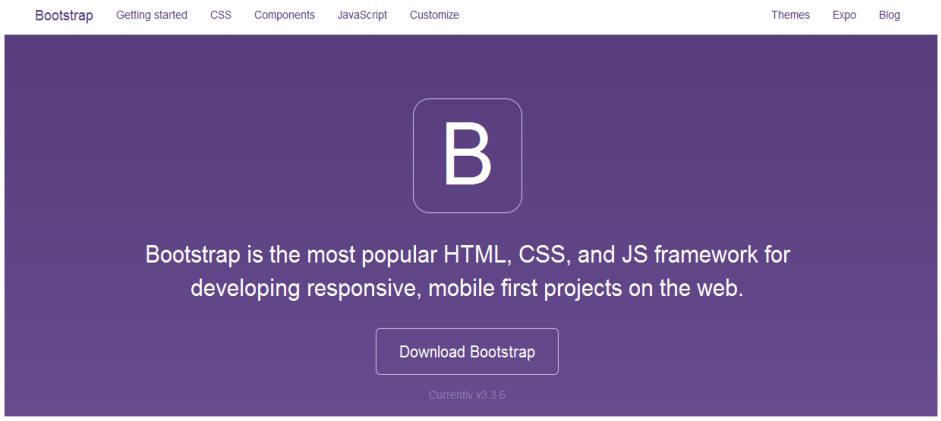


Fig.1.2 página de descarga de Bootstrap

Este link descarga un directorio de nombre bootstrap-3.3.6-dist.zip, el cual se muestra en la *figura 1.3.*

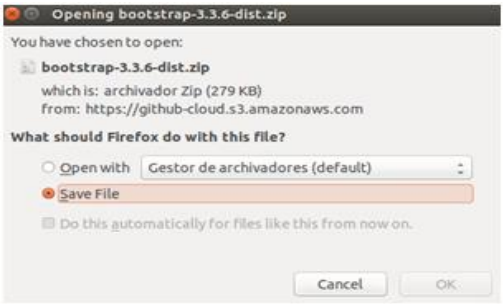


Fig.1.3 archivo de descarga Bootstrap

Este tiene una distribución de archivos los cuales se muestran en la figura 1.4 junto a sus contenidos.

Fig.1.4 archivos del directorio Bootstrap

Es este el archivo compilado que soportará las plantillas a utilizar para crear las vistas proporcionadas por el sitio oficial de Bootstrap, dichas plantillas se encuentran en la dirección http://getbootstrap.com/getting-started/#examples.

1.4 Raspberry Pi 2.0:

Este es un ordenador de bajo costo, que posee 4 puertos USB, un puerto de red rj-45, una salida de video HDMI, 40 pines GPIO y una ranura para memoria SD. Las características principales de su hardware son:

- Un procesador 900MHz quad-core ARM Cortex-A7 CPU.
- Memoria RAM de 1GB.
- Una memoria SD máxima de 32 GB, sin que se corrompa el sistema.

Una imagen de la placa se muestra en la figura 1.5.

Fig.1.5 Raspberry pi 2.0

Esta placa puede trabajar perfectamente con una fuente de alimentacion de 5V DC a 2 amp, a una temperatura de operación promedio de 45.5 °C, la distribucion de sus pines se muestra en la figura 1.6

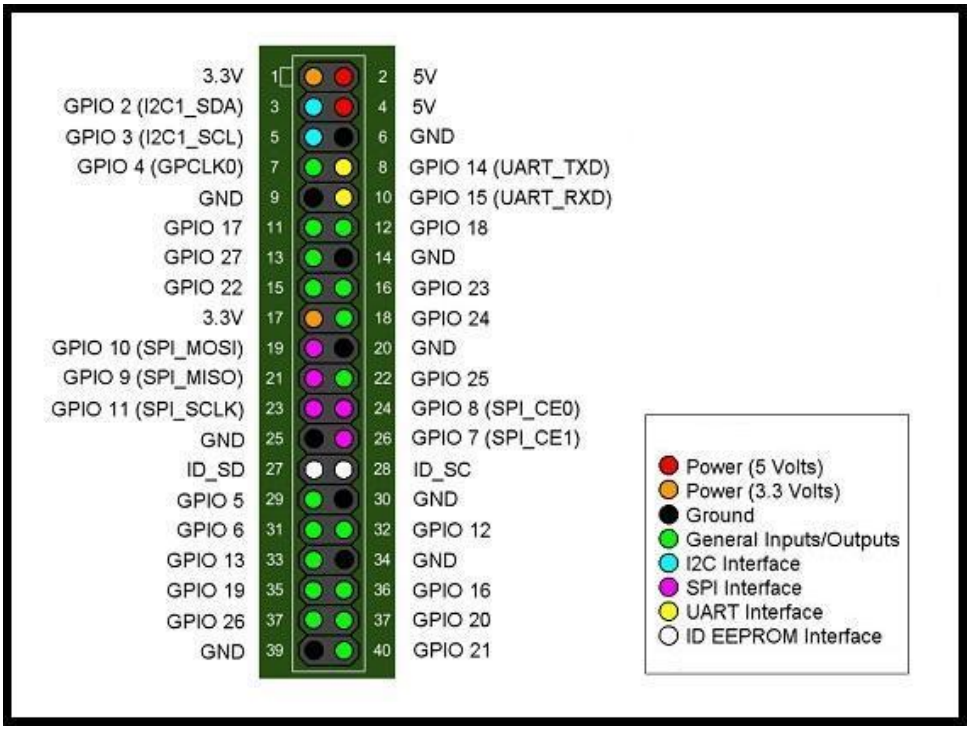


Fig.1.6 Distribución de pines de la Raspberry Pi 2.0

1.5 Arduino Nano:

Este es un dispositivo microcontrolador programable que posee pines de entrada salida digitales y pines de entrada salida analógicos a un nivel de tensión de entre 7 – 12 VDC y una corriente máxima por cada pin de 40 mA, pose dos pines de comunicación serie Rx, Tx colocados por defecto en los pines cero y uno respectivamente, el corazón de esta placa es el micro controlador ATmega 328P , que trabaja a una frecuencia de reloj de 16 MHz, posee una memoria flash de 32 KB de los cuales 2KB son utilizados por el arrancador de la memoria del Arduino, es decir es la memoria de instrucciones interna que posee para procesar los comandos de control sobre el dispositivo, una imagen de la distribución de pines se muestra en la figura 1.7

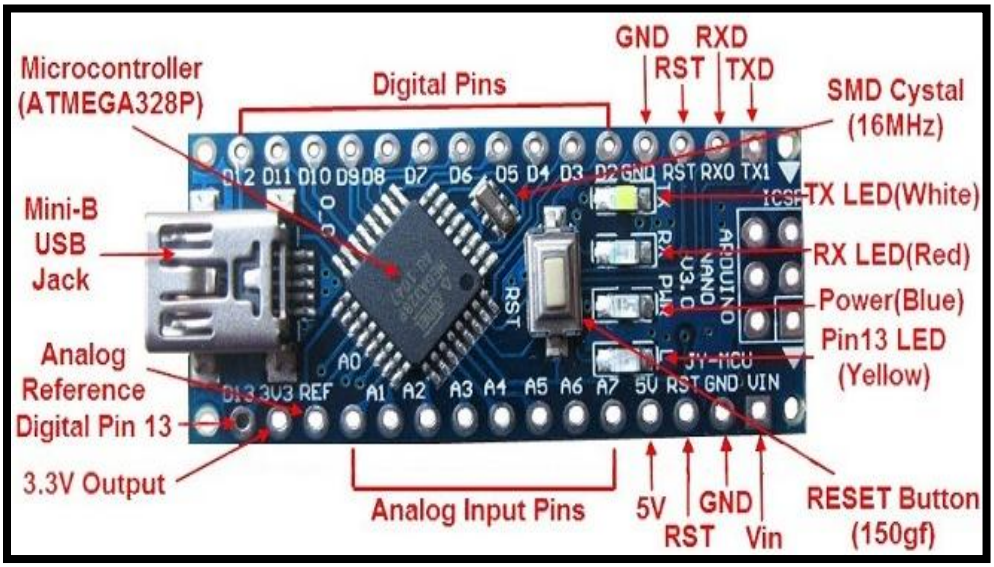


Fig.1.7 Distribucion de pines del Arduino Nano

1.6 Sensor biométrico GT511C3:

Este es un sensor de huellas dactilares con un procesador ARM cortex M3 Core (Holtek HT32F2755), un sensor óptico y una memoria interna con la capacidad de guardar hasta 200 huellas, el cual tiene una memoria de instrucciones interna almacenada, que se utiliza para que programas accedan a las funcionalidades que posee.

La velocidad de comunicación en transferencia de datos está configurada por defecto para trabajar a 9600 bps , y además posee un nivel de tensiones de operación de entre 3.3~6 VDC y una corriente de operación inferior a 130 mA, el tiempo de captación de huella está estimado a 3 segundos o inferior por defecto, la imagen del sensor y su correspondiente distribución de pines se muestra en la figura 1.8.

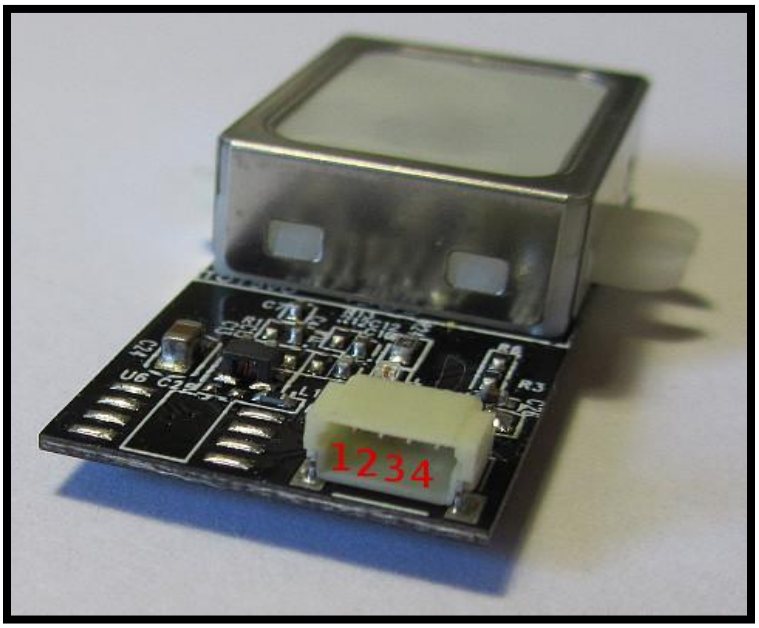


Fig.1.8 Distribución de pines del Sensor Biométrico GT511C3

Distribución de pines del sensor:

1- *UART Transmit (3.3 V)*
2- *UART Receive (3.3 V)*
3- *GND*
4- *Vin (5V)*

CAPITULO 2: Configuración del sistema biométrico

2.1 Generalidades:

El sistema de identificación de personas por medio de un sensor biométrico consta de cinco dispositivos con los cuales se logra la correcta funcionalidad del mismo, estos dispositivos son los siguientes:

- **Raspberry Pi 2.0** : Este dispositivo es el que se utiliza como un servidor Web, es el que guarda el programa desarrollado en Django, y que además tiene el control global sobre el sistema, por medio de la conexión serial entre el dispositivo de interfaz de comunicación, Arduino Nano.

- **Arduino Nano** : Sirve como interfaz de comunicación entre el sensor biométrico y la Raspberry Pi 2.0, es el que interpreta la información que se genera por el sensor biométrico por medio de una conexión serial entre ambos, ya que tiene los comandos que interrogan y hacen peticiones de control sobre el sensor biométrico, por medio de un archivo hexadecimal incrustado.

- **Sensor Biométrico GT511C3:** Este es un sensor de huella dactilar, siendo su principal función capturar la huella de una persona y asignarle un ID que lo identifique como un usuario único. Este sensor de huella dactilar contiene en su interior una base de datos, capaz de guardar el registro hasta 200 personas, iniciando por el ID = 0 hasta un ID = 199, este contiene en su interior librerías con las cuales interpreta las instrucciones con las que le hacen las peticiones.

- **Pantalla LCD de 3.5":** Este dispositivo servirá para visualizar los mensajes o peticiones que el sensor biométrico genera, ya sea para el registro de una persona en la base de datos del sistema o para la validación de una huella ya existente en el sistema.

- **Adaptador USB Wifi 802.11n:** Este dispositivo es el que permitirá efectuar una comunicación inalámbrica con la red de internet.

2.2 Configuración Inicial de la Raspberry Pi 2.0:

Para iniciar con la configuración de la Raspberry Pi 2.0 es de vital importancia conseguir una memoria SD tipo clase 10 ya que es un requisito primordial para instalar el sistema operativo. Hay que tomar en cuenta que la capacidad mínima que debe poseer es por lo menos 8 GB de almacenamiento.

El sistema operativo que se utiliza como plataforma de trabajo es el Raspbian que es un sistema operativo Linux especialmente diseñado para dispositivos Raspberry Pi , el cual se puede descargar de la siguiente dirección *https://www.raspberrypi.org/downloads/.*

En este caso se utilizó la imagen del sistema operativo Raspbian que proporciona el proveedor de la pantalla LCD ya que en su sistema operativo incluye los archivos necesarios de configuración, de esta forma con la instalación del sistema operativo queda funcionando en automático la pantalla LCD, esta imagen se puede conseguir en la siguiente dirección *http://www.waveshare.com/*

Lo primero que debe hacerse es descomprimir la imagen del sistema operativo en la micro SD, los pasos a utilizar son los siguientes y estos se realizaron utilizando la plataforma de trabajo Linux Ubuntu:

- ✓ Averiguar la dirección física del dispositivo SD montado, esto se hace por medio del comando **`df –h`**, el resultado de la ejecución de este comando se muestra en la figura 3.1

- ✓ Se descomprime la imagen del sistema operativo en la SD utilizando el comando **`dd bs=4M if=nombre.img of=/dev/direcciónfísicadelaSD.`** Donde *nombre.img* es la ruta donde se encuentra la imagen a descomprimir.

- ✓ Para asegurar que el sistema operativo se instaló con éxito en la SD se utiliza el comando **`PKILL –USR1 –n –x dd`** el cual muestra el porcentaje de avance de la instalación en la SD, mostrando un mensaje de **`DONE`**, cuando se instala con éxito.

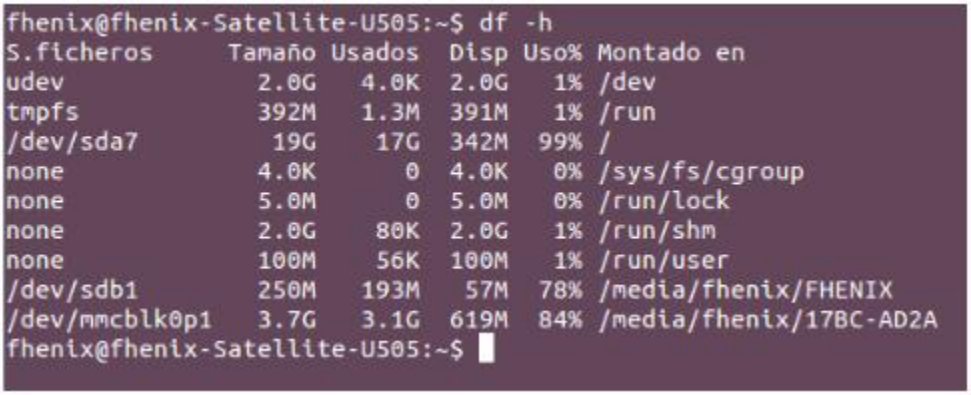


Fig.2.1 dispositivo SD montado

La figura 2.1 muestra el dispositivo SD montado en este caso `/dev/mmcblk0` que es el nombre del dispositivo completo sin partición alguna, es esta la dirección que se utilizará a la hora de instalar el sistema operativo en la memoria SD.

En el primer arranque del sistema operativo se mostrará una pantalla de configuración ver figura 2.2

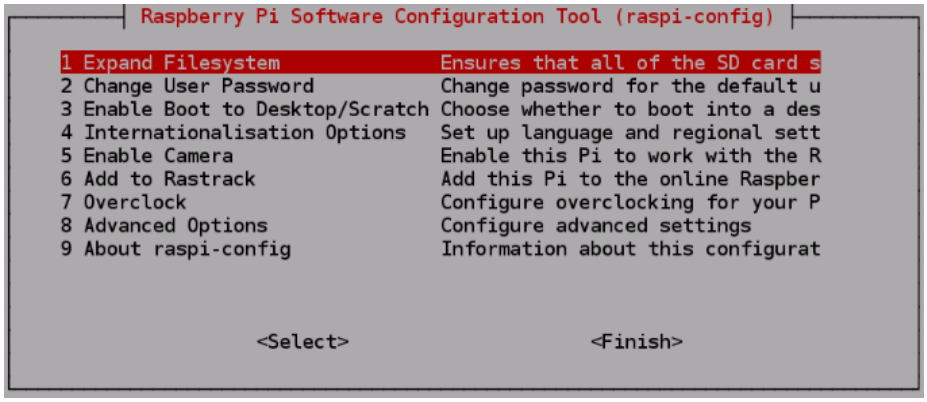


Fig.2.2 pantalla de primer arranque del sistema operativo

Expand Filesystem, es la primera opción seleccionada, con la finalidad de utilizar la capacidad total del dispositivo SD al que se le instaló el sistema operativo.

En su defecto también podemos efectuar esta configuración en un momento posterior cuando el sistema haya sido instalado mediante el comando siguiente:

```
~$ sudo raspi-config.
```

Esta configuración se puede realizar utilizando una conexión remota de la Raspberry Pi 2.0 con cualquier ordenador que corra un sistema operativo y un servidor VNC o directamente conectando teclado y monitor por medio de un cable HDMI a la Raspberry Pi 2.0, en este caso se utiliza una conexión remota con el servidor ***tightvncserver*** instalado en la Raspberry Pi 2.0.

En Raspbian el comando utilizado para instalar el servidor ***tightvncserver*** es el siguiente:

```
sudo apt-get install tightvncserver
```

Con esto el servidor está listo para realizar la conexión remota entre la Raspberry Pi 2.0 y para el caso particular Linux-Ubuntu.

El primer paso que se debe realizar para efectuar la conexión remota es, conocer la dirección IP del dispositivo que se quiere conectar remotamente, esto se logra mediante el comando **`nmap –sp direcciónIP/24`** que mostrará los dispositivos que están conectados a la misma red. Una vez que se conoce la dirección IP del dispositivo que se quiere conectar remotamente se utiliza el comando `ssh` siguiente:

```
sudo ssh usuario@direccionIP
```

Esto permitirá acceder de forma remota a la Raspberry Pi, desde el sistema operativo Linux Ubuntu, la conexión que se realizará utilizando el comando `ssh` es vía línea de comando, con esto ya es posible instalar el servidor VNC al interior del sistema operativo de la Raspberry Pi 2.0 mediante el comando:

```
sudo apt-get install tightvncserver
```

Una vez instalado el servidor VNC dentro de la Raspberry Pi 2.0 se corre el siguiente comando:

```
tightvncserver:2 –depth 16 –pixelformat rgb565
```

Es este comando quien levanta el servidor VNC para realizar la conexión remota al sistema operativo Linux-Ubuntu.

Salir de la conexión `ssh` y ejecutar el siguiente comando en la plataforma de Linux Ubuntu:

```
xtightvncviewer direcciónIP de la RaspberryPI:2
```

Y listo desde este momento ya se tiene una conexión remota con la Raspberry Pi 2.0, y de esta forma ya se puede trabajar en el propio sistema de la Raspberry Pi 2.0 desde otro ordenador, esto se puede hacer mediante el uso de un cable Ethernet conectado a la red LAN o vía red inalámbrica.

Desde este momento se pueden instalar los programas y paquetes que se necesiten para controlar las funcionalidades del sistema de captación de huellas biométrico, esto se hace mediante la instalación de ***Django*** que es un entorno de desarrollo orientado a la Web.

2.3 Instalación de Django:

Antes de instalar Django se debe asegurar de tener instalado como mínimo el lenguaje de programación Python y sus librerías, para ello se teclea el siguiente comando:

```
~ $ python --version
```

Si el programa está instalado devolverá en el terminal del sistema operativo la versión del programa instalado. La versión de Phyton utilizado es la versión 2.7.3. Luego de tener Python instalado en el sistema operativo, necesita tener instalado las dependencias o librerías necesarias para que Django funcione correctamente, una de ellas es Python Package Index la cual se instala por medio del siguiente comando:

```
~ $ sudo apt-get install Python-pip
```

Este comando instala una dependencia con la cual se permite instalar un programa utilizando el comando **pip**, de esta manera para instalar Django se usa el siguiente comando:

```
~ $ pip install Django
```

Este comando instalará Djando directamente en el sistema operativo. Los comandos necesarios para configurar correctamente el proyecto con Django, son los siguientes:

```
~ $ sudo apt-get install Python-django
```

- ✓ Instalación de MySQL-SERVER

```
~ $ sudo apt-get update
~ $ sudo apt-get upgrade
~ $ sudo apt-get install mysql-server
```

- ✓ Instalación de MySQLdb, conexión con la base de datos

```
~ $ sudo apt-get update
~ $ sudo apt-get install python-dev
~ $ sudo apt-get install python-MySQLdb
```

- ✓ Instalación de la librería para el control del Puerto serie desde Python

```
~ $sudo apt-get install Python-serial
```

- ✓ Instalación del navegador google chrome

```
~ $sudo apt-get install chromium
```

Estos son los comandos necesarios para que el proyecto en Django funcione de una manera correcta. Dado que se tiene una base de datos instalada como es ***mysql*** se procede a la creación de una base de datos la cual contendrá el listado de las personas registradas por el sensor biométrico.

- ✓ Creamos una base de datos por medio del siguiente comando dentro de ***mysql***

```
~ $ mysql –u root –p    //esto nos pedirá pasword root

~$password: root

>>> créate database sistemabiometrico
```

La ejecución del comando anterior crea la base de datos con la cual se enlazará el proyecto sistemabiométrico.

2.4 Creación del proyecto *sistema* en Django:

Para crear un proyecto en Django se teclea el siguiente comando:

```
~ $django-admin.py startproject Sistema
```

Esto creará un directorio con el nombre sistema que contendrá en su interior los siguientes archivos:

- ✓ manage.py //archivo con extensión .py
- ✓ sistema //directorio que contiene las librerías del proyecto

Siendo el contenido del directorio sistema el siguiente:

- ✓ setting.py
- ✓ urls.py
- ✓ wsgi.py
- ✓ __init__.py

Como se puede observar al crear el nuevo proyecto por defecto se tinen los archivos ***manage.py*** y el directorio ***sitema,*** y dentro del directorio ***sistema*** se generan los archivos mensionados anteriomente.

El proyecto contendrá la administración del sistema de captación de huellas biométrico.

Como siguiente paso es necesario crear la aplicación del proyecto sistema, esta contendrá los archivos necesarios de configuración de los modelos y las vistas de la aplicación global. Para crear las aplicaciones del proyecto se escribe el siguiente comando en el Shell de Raspbian:

```
~ $python manage.py startapp app
```

Este comando crea un nuevo directorio que contiene la aplicación del proyecto, con el nombre ***app*** , cuyo contenido es el siguiente:

- ✓ admin.py
- ✓ apps.py
- ✓ models.py
- ✓ __init__.py

- ✓ Test.py
- ✓ Views.py

Son estos los directorios que Django crea por defecto al establecer un nuevo proyecto y cuyo contenido ya tiene una estructura definida por el programa, son estos los archivos que se modificarán para administrar la funcionalidad global del sistema de captación de huellas por medio de un sensor biométrico.

Una vez el proyecto haya sido creado, verificar si se hizo de forma correcta, tecleando en la terminal del sistema operativo el siguiente comando:

```
//esto levanta el servidor interno de Django

~ $python manage.py runserver
```

De este modo si todo está correcto debería aparecer en un navegador Web al poner la siguiente dirección IP 127.0.0.1 el resultado se muestra en la figura 2.3.

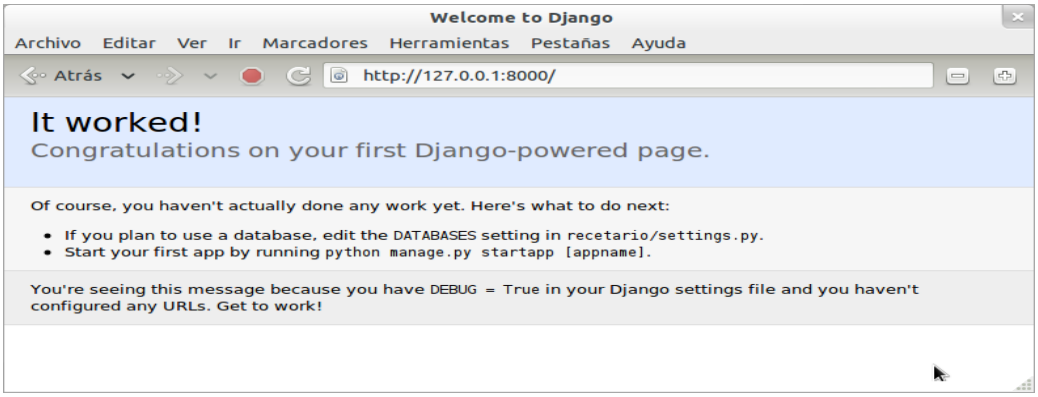


Fig.2.3 muestra el servidor de desarrollo corriendo correctamente

La cual muestra que el levantamiento del proyecto por medio del servidor interno de Django funciona satisfactoriamente.

Para empezar a trabajar en el proyecto y darle funcionalidad, se debe de crear una aplicación que gestione la información y las vistas que va tener el sitio Web. Para crear esta aplicación se procede de la siguiente manera:

```
~$python-manage.py startapp app
```

Esto creará dentro del proyecto un directorio con el nombre antes mencionado cuyo contenido es el siguiente:

- ✓ ***views.py***
- ✓ ***models.py***
- ✓ ***admin.py***
- ✓ ***test.py***

Para implementar la primera vista en Django, crear un directorio con el nombre de ***templates***, dentro del directorio ***sensorapp.***

Es en este directorio ***template*** donde se genera el archivo HTML, el cual contendrá la primera vista del sitio Web.

Se recomienda configurar y crear la base de datos que se va a utilizar, antes de levantar el servidor de desarrollo, de lo contrario el programa creará una base de datos por defecto en ***Sqlite3,*** la configuración de la base de datos utilizada por este proyecto se muestra en el capítulo 4.

2.5 Creación de la base de datos en MySQL:

El siguiente paso después de implementar el proyecto, es crear una base de datos en MySQL, que contendrá las tablas que se declararon dentro del archivo models.py que creo por defecto Django.

Para crear una base de datos en MySQL , ingresar a su entorno de trabajo, escribiendo el siguiente comando en el Shell de Raspbian:

```
//password:root
~$mysql –u root –p
```

La ejecución de este comando pedirá un password el cual se escribe como ***root.***

Después de esto ya se accedió al entorno de trabajo y se puede crear la base de datos que contendrá los registros que necesita el sistema de identificación de huellas para operar correctamente. La base de datos fue creada con el nombre de ***sistemabiométrico,*** la cual contiene las tablas mostradas en la figura 2.4, al escribir el siguiente comando:

```
//este comando muestra las tablas generadas
mysql>show tables;
```

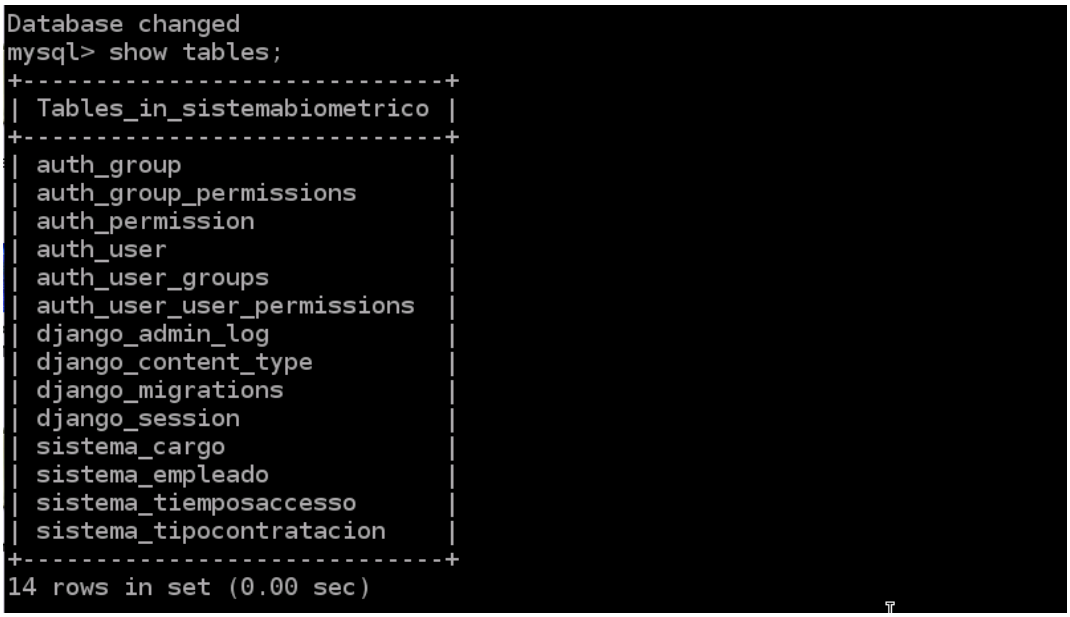


Fig.2.4 tablas en sistemabiometrico

Esta base de datos muestra tanto las tablas que fueron declaradas dentro del archivo models.py así como las tablas que Django necesita para levantar el proyecto y que fueron creadas después de hacer la migración por medio del siguiente comando:

```
~$ python manage.py Makemigrations

~$ python manage.py migrate
```

Las tablas que fueron creadas por el usuario para la administración del sistema son las siguientes:

- ✓ ***sistema_cargo***
- ✓ ***sistema_empleado***
- ✓ ***sistema_tiemposaccesso***
- ✓ ***sistema_tipocontratación***

Estas son las tablas que se necesitan para que el sistema funcione de una forma correcta en el registro de huellas asociadas a una persona en particular.

Después de las migraciones se instalan los siguientes comandos:

```
~$ sudo pip install django-widget-tweaks

~$ python manage.py createsuperuser
```

2.6 Contenido de las tablas creadas en MySQL:

Cada una de las tablas creadas en la base de datos juega un papel importante dentro del proyecto sistema biométrico, en la figura 2.5 se muestra la tabla sistema_empleado.

```
mysql> describe sistema_empleado;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| id                   | int(11)      | NO   | PRI | NULL    | auto_increment |
| id_biometrico        | int(11)      | NO   |     | NULL    |                |
| dui                  | varchar(10)  | NO   |     | NULL    |                |
| nit                  | varchar(17)  | NO   |     | NULL    |                |
| nup                  | varchar(10)  | NO   |     | NULL    |                |
| isss                 | varchar(10)  | NO   |     | NULL    |                |
| nombre               | varchar(50)  | NO   |     | NULL    |                |
| telefono             | varchar(10)  | NO   |     | NULL    |                |
| escuela              | varchar(100) | NO   |     | NULL    |                |
| fecha                | date         | NO   |     | NULL    |                |
| cargo_id             | int(11)      | NO   | MUL | NULL    |                |
| tipo_contratacion_id | int(11)      | NO   | MUL | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
12 rows in set (0.01 sec)
```

Fig.2.5 tablas empleado

Esta tabla contiene cada una de las personas registradas en la base de datos por medio del sensor biométrico y esta contiene los campos mostrados en dicha figura 2.5.

La tabla sistema_cargo, se muestra en la figura 2.6 es la que contiene el tipo de cargo que se le asigna a una persona registrada en la base de datos, esta puede ser de entre *docente* o *director*, estos son los únicos campos que contiene la tabla que son utilizados a la hora de generar el formulario, en la vista de un usuario registrado.

```
mysql> describe sistema_cargo;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | int(11)     | NO   | PRI | NULL    | auto_increment |
| nombre_cargo | varchar(40) | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

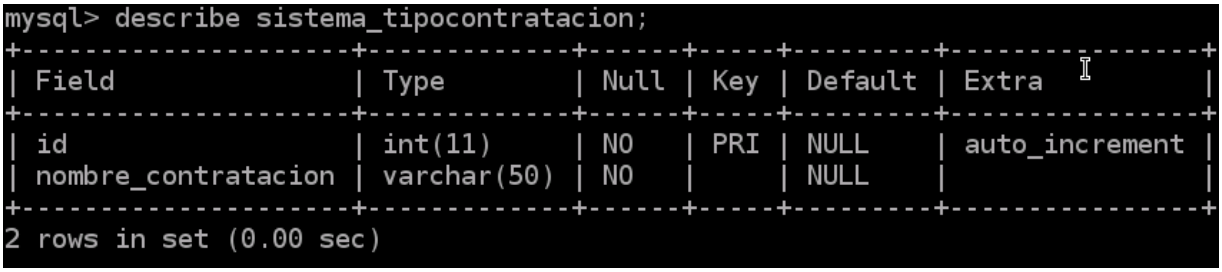
Fig.2.6 tabla cargo

La tabla sistema_tiemposaccesso, se muestra en la figura 2.7 es la que captura la hora y fecha del sistema, es decir si un usuario valida su huella, se genera una entrada a esta tabla, de tal manera que los campos entrada-salida, son las variables más importantes.

```
mysql> describe sistema_tiemposaccesso;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| id          | int(11)  | NO   | PRI | NULL    | auto_increment |
| entrada     | datetime | NO   |     | NULL    |                |
| salidad     | datetime | NO   |     | NULL    |                |
| is_open     | int(11)  | NO   |     | NULL    |                |
| empleado_id | int(11)  | NO   | MUL | NULL    |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Fig.2.7 tabla tiempo accesos

La tabla sitema_tipocontratación, la cual se muestra en la figura 2.8 en la que contiene el tipo de contratación que se le puede generar a un usuario en particular ya sea eventual o plaza fija, son estos los únicos nombres que contiene esta tabla, con el tipo de variables configuradas como se muestra en dicha figura.



```
mysql> describe sistema_tipocontratacion;
+--------------------+-------------+------+-----+---------+----------------+
| Field              | Type        | Null | Key | Default | Extra          |
+--------------------+-------------+------+-----+---------+----------------+
| id                 | int(11)     | NO   | PRI | NULL    | auto_increment |
| nombre_contratacion | varchar(50) | NO   |     | NULL    |                |
+--------------------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

Fig.2.8 tabla tipo de contratación

2.7 Configuración de la red inalámbrica:

Para establecer la comunicación inalámbrica de la Raspberry Pi 2.0 con la red de internet, se utilizó un dispositivo USB Wifi (ya que la Raspberry Pi 2.0 no posee una tarjeta de red interna) que usa el protocolo 802.11n, para lo cual se tuvo que configurar en archivo ***interfaces*** residente en la ruta `/etc/network/interfaces`, éste posee la siguiente información:

```
1 auto lo
2
3 iface lo inet loopback
4 iface eth0 inet dhcp
5
6 allow-hotplug wlan0
7 #iface wlan0 inet manual
8 auto wlan0
9 iface wlan0 inet dhcp
```

```
10          wpa-ssid "MOTOROLATIGO"
11          wpa-psk "palacios"
12 #wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
13 #iface default inet dhcp
```

De este código los campos que se tienen que cambiar cuando se desee una conexión a cualquier red en particular, son:

```
wpa-ssid "DNS"
wpa-psk "pasword"
```

Donde el DNS es el nombre de la red a la cual se va a conectar y el password, es la contraseña de dicha red.

2.8 Configuración del puerto serial:

Para poder conectar cualquier dispositivo al puerto serial de la Raspberry Pi 2.0, es necesario que éste tenga los puertos abiertos y tener permiso para poder escribir o leer en dicho puerto, para lo cual se deben modificar los archivos:

- ✓ ***cmdline.txt***
- ✓ ***inittab***

la ruta de dichos archivos se muestra a continuación con sus correspondientes líneas de código que se deberán modificar: **`sudo nano /boot/cmdline.txt`**.

El contenido de dicho archivo se muestra a continuación:

archivo cmdline.txt

```
dwc_otg.lpm_enable=0 console=ttyAMA0, 115200 kgdboc=ttyAMA0, 115200
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Este archivo se modifica para tener algo como esto:

```
dwc_otg.lpm_enable=0    console=tty1    root=/dev/mmcblk0p2    rootfstype=ext4
elevator=deadline rootwait
```

El siguiente archivo que se debe modificar se encuentra en la ruta:

sudo nano /etc/inittab

archivo inittab

En este archivo la única modificación que se debe realizar, es comentar la última línea de código o eliminarla.

```
 1 # /etc/inittab: init(8) configuration.
 2 # $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $
 3
 4 # The default runlevel.
 5 id:2:initdefault:
 6
 7 # Boot-time system configuration/initialization script.
 8 # This is run first except when booting in emergency (-b) mode.
 9 si::sysinit:/etc/init.d/rcS
10
11 # What to do in single-user mode.
12 ~~:S:wait:/sbin/sulogin
13
14 # /etc/init.d executes the S and K scripts upon change
15 # of runlevel.
16 #
17 # Runlevel 0 is halt.
18 # Runlevel 1 is single-user.
19 # Runlevels 2-5 are multi-user.
20 # Runlevel 6 is reboot.
21
22 l0:0:wait:/etc/init.d/rc 0
23 l1:1:wait:/etc/init.d/rc 1
24 l2:2:wait:/etc/init.d/rc 2
25 l3:3:wait:/etc/init.d/rc 3
26 l4:4:wait:/etc/init.d/rc 4
27 l5:5:wait:/etc/init.d/rc 5
28 l6:6:wait:/etc/init.d/rc 6
29 # Normally not reached, but fallthrough in case of emergency.
30 z6:6:respawn:/sbin/sulogin
31
32 # What to do when CTRL-ALT-DEL is pressed.
33 ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
34
```

```
35 # Action on special keypress (ALT-UpArrow).
36 #kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let
this work."
37
38 # What to do when the power fails/returns.
39 pf::powerwait:/etc/init.d/powerfail start
40 pn::powerfailnow:/etc/init.d/powerfail now
41 po::powerokwait:/etc/init.d/powerfail stop
42
43 # /sbin/getty invocations for the runlevels.
44 #
45 # The "id" field MUST be the same as the last
46 # characters of the device (after "tty").
47 #
48 # Format:
49 #  <id>:<runlevels>:<action>:<process>
50 #
51 # Note that on most Debian systems tty7 is used by the X Window System,
52 #so if you want to add more getty's go ahead but skip tty7 if you run X.
53 #
54 1:2345:respawn:/sbin/getty --noclear 38400 tty1
55 2:23:respawn:/sbin/getty 38400 tty2
56 3:23:respawn:/sbin/getty 38400 tty3
57 4:23:respawn:/sbin/getty 38400 tty4
58 5:23:respawn:/sbin/getty 38400 tty5
59 6:23:respawn:/sbin/getty 38400 tty6
60
61 # Example how to put a getty on a serial line (for a terminal)
62 #
63 #T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
64 #T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
65
66 # Example how to put a getty on a modem line.
67 #
68 #T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
69
70
71 #Spawn a getty on Raspberry Pi serial line
72 T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

La línea de código que se debe comentar o eliminar es la siguiente:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Después de comentar la línea de código anteponiéndole el símbolo de numeral #, se debe reiniciar el sistema operativo para que los cambios surjan efecto. Con esto ya se tiene liberado el puerto serial para su uso.

2.9 Configuración del servidor Apache 2.0:

Para iniciar el proceso de instalación se descargan todos los repositorios necesarios desde nuestro Raspbian, por medio del siguiente comando, en el Shell del sistema operativo:

```
~$ sudo apt-get update

~$ sudo apt-get install python-pip apache2 libapache2-mod-wsgi
```

Este comando instalará una dependencia Python con el servidor Apache2.0 este permitirá levantar aplicaciones o páginas Web escritas en python.

Para configurar la ruta WSGI se necesita editar el archivo virtual host por defecto.

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

A este archivo únicamente se le añadirán unas líneas conservando las directivas originales, para empezar configurar los archivos estáticos:

```
<VirtualHost *:80>

. . .

Alias /static /home/user/myproject/static
<Directory /home/user/myproject/static>
Require all granted

</Directory>

</VirtualHost>
```

Luego se le concede acceso al archivo wsgi.py, por medio de las siguientes líneas de código:

```
<VirtualHost *:80>

. . .

Alias /static /home/user/myproject/static
<Directory /home/user/myproject/static>
Require all granted
</Directory>
<Directory /home/user/myproject/myproject>
<Files wsgi.py>
Require all granted
</Files>
</Directory>
</VirtualHost>
```

Después de realizar la configuración anterior, se construye la porción del archivo que actualmente maneja la ruta WSGI.

```
<VirtualHost *:80>
. . .
Alias /static /home/user/myproject/static
<Directory /home/user/myproject/static>
Require all granted
</Directory>
<Directory /home/user/myproject/myproject>
<Files wsgi.py>
Require all granted
</Files>
</Directory>
WSGIDaemonProcess myproject python-
path=/home/user/myproject:/home/user/myproject
WSGIProcessGroup myproject
WSGIScriptAlias / /home/user/myproject/myproject/wsgi.py
</VirtualHost>
```

2.10 Configuración de los permisos de mysql:

Establecer los permisos para que Apache 2.0 realice procesos sobre la base de datos, se hace cambiando los permisos de los grupos de usuarios:

```
chmod 664 ~/myproject/mysql
```

Se necesita darle permiso al directorio donde reside Apache 2.0, `www-data`.

```
sudo chown :www-data ~/myproject/mysql
```

Se le da permiso a Apache2.0 de escribir en el directorio, mediante el siguiente comando.

```
sudo chown :www-data ~/myproject
```

Luego de haber ejecutado los pasos anteriores, se debe reiniciar el servidor para que se implementen los cambios realizados por medio, del siguiente comando .

```
sudo service apache2 restart
```

Desde este momento podemos acceder a la ruta donde se encuentra alojado el proyecto, desde el dominio o IP sin tener que especificar un puerto.

CAPITULO 3: Programas del sistema

3.1 Generalidades:

Este sistema en particular, consta de 3 dispositivos los cuales son usados para darle correcta funcionalidad al sistema global, estos dispositivos son la ***Raspberry Pi 2.0, Arduino Nano y el Sensor GT511C3,*** de los cuales solo dos se pueden programar para efectuar el control del sistema global.

3.2 Raspberry Pi 2.0:

En este proyecto, se utiliza como servidor Web es el que guarda el programa que contendrá el sitio Web que controlará las funcionalidades del sistema de identificación de huellas dactilares, además es el que recibe la información que manda el sensor por medio de la conexión de la interface de comunicación y la placa ***Arduino Nano,*** por medio del puerto serial.

Este dispositivo contiene un programa desarrollado en Django, el cual es un ***Framework*** de desarrollo ***Web*** que tiene la particularidad de integrar el sistema de programación en Python y un servidor de producción como lo es Apache 2.0, que es el que levantará la aplicación ***Web*** en la red de internet por medio de la configuración del archivo wsgi.py generado por Django al inicializar el proyecto, esto mediante la instalación y configuración del módulo mod_wsgi.

El proyecto está dividido en dos rutas creadas por Django, el directorio *aplicaciones* y el directorio del proyecto que sirve como librerías creadas para que el proyecto tenga funcionalidad.

3.3 Programas que contiene la Raspberry Pi 2.0:

Dentro de la Raspberry Pi 2.0 se encuentra un aplicativo Web creado en Django cuyo nombre es APP, residente en el directorio `/var/www/` dentro del servidor.

Dentro del directorio aplicación APP:

Settings.py

Dentro de este archivo, las líneas de código que se agregan y se editan se muestran a continuación:

```
31 # Application definition
32
33 INSTALLED_APPS = (
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'sistema',
41     'widget_tweaks',
42 )
```

Se agregaron las siguientes líneas:

- ✓ ***Sistema***
- ✓ ***Widget tweaks***

En el siguiente código, se muestra como se configura la base de datos que Django-MySQL utilizará en el proyecto.

```
86 # configuración de la base de datos
87
```

```
88 DATABASES = {
89     'default': {
90       'ENGINE': 'django.db.backends.mysql',
91       'NAME': 'sistemabiometrico',
92       'USER': 'root',
93       'PASSWORD': 'root',
94       'HOST': 'localhost',
95       'PORT': '3306'
96     }
97 }
```

Las líneas de código que se agregan para la configuración de la base de datos del sitema de captación de huellas dactilares, es esta base de datos que contendra la lista de personas que estan habilitadas para acceder hacia algún lugar específico.

```
99 # Internationalization
100 # https://docs.djangoproject.com/en/1.8/topics/i18n/
101
102 LANGUAGE_CODE = 'es-sv'
103
104 TIME_ZONE = 'UTC'
105
106 USE_I18N = True
107
108 USE_L10N = True
109
110 USE_TZ = False
```

El bloque anterior, muestra las líneas de código que se utilizan para configurar la zona horaria con que trabajará la Raspberry Pi 2.0, es decir hora y fecha de la región en particular donde recida el sistema de identificación de huella dactilar.

urls.py

```
1 from django.conf.urls import include, url
2 from django.contrib import admin
3
4
5 urlpatterns = [
6     url(r'^admin/', include(admin.site.urls)),
```

```
 7         url(r'^', include('sistema.urls', namespace='sistema-app',
app_name='sistema') ),
 8 ]
 9
```

El bloque de código anterior, muestra la configuración de las urls de la aplicación.

wsgi.py

```
1 import os
2 import sys
3
4 sys.path = ['/var/www/app'] + sys.path
5 os.environ['DJANGO_SETTINGS_MODULE'] = 'app.settings'
6
7 from django.core.wsgi import get_wsgi_application
8 application = get_wsgi_application()
```

El bolque de codigo anterior le dice al servidor la ruta de la aplicación, creada en Django.

admin.py

```
1 from django.contrib import admin
2 from .models import Cargo, TipoContratacion, Empleado, TiemposAccesso
3
4 admin.site.register(Cargo)
5 admin.site.register(Empleado)
6 admin.site.register(TiemposAccesso)
7 admin.site.register(TipoContratacion)
8
```

forms.py

```
1 from django.forms import ModelForm
2 from .models import Empleado, Cargo, TipoContratacion
3
4 class EmpleadoForm(ModelForm):
5         class Meta:
6                 model = Empleado
7                 exclude = ['fecha', 'id_biometrico']
```

models.py

```
 1 import datetime
 2 from django.db import models
 3
 4
 5 class Cargo(models.Model):
 6         nombre_cargo = models.CharField(max_length=40)
 7
 8         def __unicode__(self):
 9                 return self.nombre_cargo
10
11
12 class TipoContratacion(models.Model):
13         nombre_contratacion = models.CharField(max_length=50)
14
15         def __unicode__(self):
16                 return self.nombre_contratacion
17
18
19 class Empleado(models.Model):
20         nombre = models.CharField(max_length=50)
21         id_biometrico = models.IntegerField()
22         dui = models.CharField(max_length=10)
23         nit = models.CharField(max_length=17)
24         nup = models.CharField(max_length=10)
25         isss = models.CharField(max_length=10)
26         telefono = models.CharField(max_length=10)
27         escuela = models.CharField(max_length=100)
28         fecha = models.DateField(auto_now_add=True)
29         cargo = models.ForeignKey(Cargo)
30         tipo_contratacion = models.ForeignKey(TipoContratacion)
31
32         def __unicode__(self):
33                 return "%s - %s" % ( self.nombre, self.dui )
34
35
36 class TiemposAccesso(models.Model):
37         entrada = models.DateTimeField(auto_now_add=True)
38         salidad = models.DateTimeField(auto_now_add=True)
39         is_open = models.IntegerField(default='1')
40         empleado = models.ForeignKey(Empleado)
41         def save(self, *args, **kwargs):
42                 if self.id:
43                         self.salidad = datetime.datetime.today()
44                 return super(TiemposAccesso, self).save(*args, **kwargs)
```

Urls.py

```
 1 from django.conf.urls import url
 2
 3 from .views import *
 4
 5 urlpatterns = [
 6         url(r'^$', IndexTemplate.as_view(), name='index'),
 7         url(r'^registrar$', RegistrarTemplate.as_view(),
name='registrar'),
 8         url(r'^login/$', LoginTemplate.as_view(), name='login'),
 9         url(r'^logout/$', LogoutView.as_view(), name='logout'),
10         url(r'^search$', SearchView.as_view(), name='search'),
11         url(r'^edit/(?P<pk>\d+)/$', EditView.as_view(), name='editar'),
12         url(r'^eliminar/(?P<pk>\d+)/$', DeleteView.as_view(),
name='eliminar'),
13         url(r'^views/(?P<pk>[\w\-]+)/$', EmpleadoView.as_view(),
name='vista'),
14         url(r'^validar/$', ValidarHuella.as_view(), name='validar'),
15         url(r'^empleados/$', EmpleadoPage.as_view(), name='lista'),
16 ]
```

Views.py

```
  1 from django.contrib.auth.forms import AuthenticationForm
  2 from django.contrib.auth import login, logout
  3 from django.http import JsonResponse, HttpResponseRedirect
  4 from django.contrib.auth.decorators import login_required
  5 from django.views.generic import TemplateView, FormView, DetailView,
RedirectView, UpdateView, DeleteView, ListView
  6
  7 from .models import Empleado, TiemposAccesso
  8 from .forms import EmpleadoForm
  9
 10 # import para manejar el serial
 11 import serial
 12 import time
 13
 14 # para el logeo de los usuarios, estas son las funciones basica para
las paginas
 15 class LoginRequiredMixin(object):
 16         @classmethod
 17         def as_view(cls):
```

```
 18                 return login_required(super(LoginRequiredMixin,
cls).as_view())
 19
 20
 21 # pagina de inicio
 22 class IndexTemplate(LoginRequiredMixin, TemplateView):
 23         template_name = 'index.html'
 24
 25
 26 # pagina para salir del sistema
 27 class LogoutView(RedirectView):
 28         url = '/login'
 29
 30         def get(self, request, *args, **kwargs):
 31                 logout(request)
 32                 return super(LogoutView, self).get(request, *args,
**kwargs)
 33
 34
 35 # Pagina para logear al usuario al sistema
 36 class LoginTemplate(FormView):
 37         template_name = 'sistema/login.html'
 38         form_class = AuthenticationForm
 39         success_url = '/'
 40
 41         def form_valid(self, form):
 42                 login(self.request, form.user_cache)
 43                 return super(LoginTemplate, self).form_valid(form)
 44
 45
 46 # para buscar a un empleado en particular
 47 class SearchView(LoginRequiredMixin, TemplateView):
 48         def get(self, request, *args, **kwargs):
 49                 value = self.request.GET['query']
 50                 empleados = Empleado.objects.filter(id__contains=value)
 51                 data = [{'data':empleado.id, 'value':empleado.nombre}
for empleado in empleados]
 52
 53                 suggestions = {'suggestions':data}
 54                 return JsonResponse(suggestions, safe=False)
 55
 56
 57 # estas es la vista para visualizar al empleado
 58 class EmpleadoView(LoginRequiredMixin, DetailView):
 59         template_name = 'sistema/views.html'
 60         model = Empleado
 61
 62
 63 class ValidarHuella(LoginRequiredMixin, TemplateView):
 64         template_name = 'sistema/views.html'
```

```
 65
 66         def get_context_data(self, **kwargs):
 67                 context = super(ValidarHuella,
self).get_context_data(**kwargs)
 68                 options = "L\n"
 69                 output = ''
 70                 array = None
 71                 error = False
 72                 id_biometrico = -1
 73
 74                 arduinoPort = serial.Serial('/dev/ttyUSB0', 9600,
timeout=1)
 75                 if arduinoPort.isOpen():
 76                         arduinoPort.close()
 77
 78                 arduinoPort.open()
 79                 time.sleep(1.8)
 80                 arduinoPort.write(options)
 81
 82                 while True:
 83                         try:
 84                                 getSerialValue = arduinoPort.readline()
 85                                 array = getSerialValue.split(':')
 86                                 if len(array) == 2:
 87                                         if str(array[0]) == "success":
 88                                                 id_biometrico =
array[1].replace('\r\n', '')
 89                                         elif str(array[0]) == "error":
 90                                                 id_biometrico =
array[1].replace('\r\n', '')#
 91                                                 error = True
 92                                         break
 93                                 else:
 94                                         print(getSerialValue)
 95                         except Exception, e:
 96                                 print(e)
 97
 98                 arduinoPort.close()
 99
100                 if error:
101                         context['messagerror'] = array
102                 else:
103                         empleado =
Empleado.objects.get(id_biometrico=id_biometrico)
104                         tiempos =
TiemposAccesso.objects.filter(empleado_id=empleado, is_open=1)
105
106                         if len(tiempos) == 0:
107                                 tiempos =
TiemposAccesso(empleado=empleado)
108                                 tiempos.save()
109                         else:
```

```
110                                     tiempos = tiempos[0]
111                                     tiempos.is_open = 0
112                                     tiempos.save()
113
114                             context['id_biometrico'] = id_biometrico
115                             context['object'] = empleado
116                             context['tiempos'] = tiempos
117                     context['error'] = error
118                     return context
119
120 # Registrar al empleado al sistema
121 class RegistrarTemplate(LoginRequiredMixin, FormView):
122             form_class = EmpleadoForm
123             success_url = 'views'
124             template_name = 'sistema/registrar.html'
125
126             def get_context_data(self, **kwargs):
127                     context = super(RegistrarTemplate,
self).get_context_data(**kwargs)
128                     options = "H\n"
129
130                     output = ''
131                     array = None
132                     error = False
133                     id_biometrico = -1
134
135                     arduinoPort = serial.Serial('/dev/ttyUSB0', 9600,
timeout=1)
136                     if arduinoPort.isOpen():
137                             arduinoPort.close()
138
139                     arduinoPort.open()
140                     time.sleep(1.8)
141                     arduinoPort.write(options)
142
143                     while True:
144                             try:
145                                     getSerialValue = arduinoPort.readline()
146                                     array = getSerialValue.split(':')
147                                     if len(array) == 2:
148                                             if str(array[0]) == "success":
149                                                     id_biometrico =
array[1].replace('\r\n', '')
150                                             elif str(array[0]) == "error":
151                                                     id_biometrico =
array[1].replace('\r\n', '')#
152                                                     error = True
153                                             break
154                                     else:
155                                             if len(getSerialValue) > 0 :
156                                                     print(getSerialValue)
```

```
157                         except Exception, e:
158                                 print(e)
159                 arduinoPort.close()
160
161                 context['id_biometrico'] = id_biometrico
162                 context['error'] = error
163                 return context
164
165         def form_valid(self, form):
166                 id_biometrico = self.request.POST['id_biometrico']
167                 empleado = form.save(commit=False)
168                 empleado.id_biometrico = id_biometrico
169                 empleado.save()
170                 url_redirect = '%s/%s' % (self.get_success_url(),
empleado.id)
171                 return HttpResponseRedirect(url_redirect)
172
173
174 class EditView(LoginRequiredMixin, UpdateView):
175         model = Empleado
176         success_url = '/view'
177         template_name = 'sistema/editar.html'
178         fields = ['nombre', 'nup','isss', 'telefono','escuela',
'cargo','tipo_contratacion']
179
180         def form_valid(self, form):
181                 self.success_url = "/views/%s" % (self.get_object().id)
182                 return super(EditView, self).form_valid(form)
183
184
185 class DeleteView(LoginRequiredMixin, DeleteView):
186         model = Empleado
187         template_name = 'sistema/confirm.delete.html'
188         success_url = '/'
189
190         def delete(self, request, *args, **kwargs):
191                 array = None
192                 options = "D" + str(self.get_object().id_biometrico) +
"\n"
193                 arduinoPort = serial.Serial('/dev/ttyUSB0', 9600,
timeout=1)
194
195                 if arduinoPort.isOpen():
196                         arduinoPort.close()
197
198                 arduinoPort.open()
199                 time.sleep(1.8)
200                 arduinoPort.write(options)
201
202                 while True:
203                         getSerialValue = arduinoPort.readline()
204                         print(getSerialValue)
```

```
205                         array = getSerialValue.split(":")
206                         if len(array) == 2:
207                                 break
208
209                 ids = array[1].replace('\r\n', '')
210                 if( ids == '1' ):
211                         return super(DeleteView, self).delete(request,
*args, **kwargs)
212                 else:
213                         return HttpResponseRedirect('/')
214
215 # Vista para paginar los empleados
216 class EmpleadoPage(LoginRequiredMixin, ListView):
217         model = Empleado
218         template_name = 'sistema/paginator_empleado.html'
219         context_object_name = 'empleado_list'
220         paginate_by = 10
```

3.4 Programa que contiene el Arduino Nano:

Dentro de la memoria del Arduino Nano, corre un pequeño programa que le da funcionalidad al sensor biométrico ya que este interroga a dicho sensor de tal manera que le hace peticiones para que devuelva algo al Arduino por medio del puerto serial de este.

Además es el que sirve de interfaz de comunicación entre el sensor biométrico y la Raspberry Pi 2.0, este contiene un programa escrito en C, el cual fue proporcionado por el proveedor del sensor y modificado a conveniencia para realizar las peticiones requeridas para la correcta funcionalidad del sistema biométrico.

```
1//Mandamos a llamar la librería que interpreta los comandos del sensor
2 #include "FPS_GT511C3.h"
3//Utilizamos la librería que manipula el puerto serial
4 #include "SoftwareSerial.h"
5
6//Configuramos los pines 4 y 5 como Rx, Tx
7 FPS_GT511C3 fps(4, 5);
8
9
```

```
10//Declaramos una variable vacía
11 String inputString = "";
12
13//Cabecera principal del programa
14 void setup (){
15//Definimos la velocidad de transferencia de los bits a 9600 baudios
16   Serial.begin(9600);
17
18//Definimos un retardo de 100 ms
19   delay(100);
20//Abrimos la conexión con el sensor biométrico
21   fps.Open();
22//Actibamos el led interno del sensor biométrico
23   fps.SetLED(true);
24//Con este comando borramos la base de datos del sensor biométrico
25//fps.DeleteAll();
26 }
27
28 void loop (){
29   while (Serial.available() > 0){
30     char recibido = Serial.read();
31     inputString += recibido;
32     if( recibido == '\n' ){
33       char code = inputString[0];
34//Elije la funcion que se va a ejecutar
35       switch(code){
36         case 'H': OneFunction();break;
37         case 'L': TwoFunction();break;
38         case 'D': ThreeFunction(inputString);break;
39         case 'E': FourFunction(); break;
40       }
41       inputString = "";
42     }
43   }
44 }
45
46 void FourFunction(){
47//Esta funcion borra los registros existentes en el sensor
48   int result = fps.DeleteAll();
49   Serial.print("success:");
50   Serial.println(result);
51 }
52
53 void OneFunction(){
54//Esta funcion registra una huella en la base de datos
55       int enrollid = 0;
56         bool usedid = true;
57         while (usedid == true)
58         {
59                 usedid = fps.CheckEnrolled(enrollid);
60                 if (usedid==true) enrollid++;
```

```
61          }
62          fps.EnrollStart(enrollid);
63          Serial.print("Presione su dedo para registrar #");
64          Serial.println(enrollid);
65          while(fps.IsPressFinger() == false) delay(100);
66          bool bret = fps.CaptureFinger(true);
67          int iret = 0;
68          if (bret != false)
69          {
70                  Serial.println("Remover dedo");
71                  fps.Enroll1();
72                  while(fps.IsPressFinger() == true) delay(100);
73                  Serial.println("Presione el mismo dedo");
74                  while(fps.IsPressFinger() == false) delay(100);
75                  bret = fps.CaptureFinger(true);
76                  if (bret != false)
77                  {
78                          Serial.println("Remover dedo");
79                          fps.Enroll2();
80                          while(fps.IsPressFinger() == true) delay(100);
81                          Serial.println("Presionar el mismo dedo");
82                          while(fps.IsPressFinger() == false) delay(100);
83                          bret = fps.CaptureFinger(true);
84                          if (bret != false)
85                          {
86                                  Serial.println("Remover dedo");
87                                  iret = fps.Enroll3();
88                                  if (iret == 0)
89                                  {
90                                          Serial.print("registrado:");
91                                          Serial.println(enrollid);
92                                  }
93                                  else
94                                  {
95                                          Serial.print("error:");
96                                          Serial.println(iret);
97                                  }
98                          }
99                          else Serial.println("error:Failed to capture
third finger");
100                 }
101                 else Serial.println("error:Failed to capture second
finger");
102         }
103         else Serial.println("error:Failed to capture first finger");
104 }
105
106 void TwoFunction(){
107//Esta funcion verifica las huellas registradas en el sistema
108   while(true){
109     if (fps.IsPressFinger()){
110     fps.CaptureFinger(false);
```

```
111     int id = fps.Identify1_N();
112       if (id < 200){
113         Serial.print("success:");
114         Serial.println(id);
115         break;
116       }else{
117         Serial.println("error:Finger not found");
118         break;
119       }
120      }else{
121       Serial.println("Please press finger");
122     }
123     delay(500);
124   }
125 }
126
127
128 void ThreeFunction(String response){
129//Esta funcion borra Ids especificos registrados en el sistema
130  String ids = response.substring(1);
131  int returns = fps.DeleteID(ids.toInt());
132  Serial.print("response:");
133  Serial.println(returns);
134 }
```

CAPITULO 4: Diseño del circuito nivelador de tensión

4.1 Diseño del PCB:

Para poder efectuar la conexión del sensor biométrico con el ***Arduino Nano*** se necesita un nivelador de tensión entre el pin Tx del Arduino y el pin Rx del sensor biométrico, ya que el nivel de tensión de salida en el pin Tx del Arduino está a un nivel alto de 5V y el sensor biométrico está diseñado para recibir una señal a 3.3 V en su pin Rx, por tal motivo se necesita de un nivelador de tensión que realice la conversión entre estos dos dispositivos. El PCB del nivelador de tensión junto con la placa de conexión del Arduino Nano se muestra en la figuras 4.1 y 4.2.

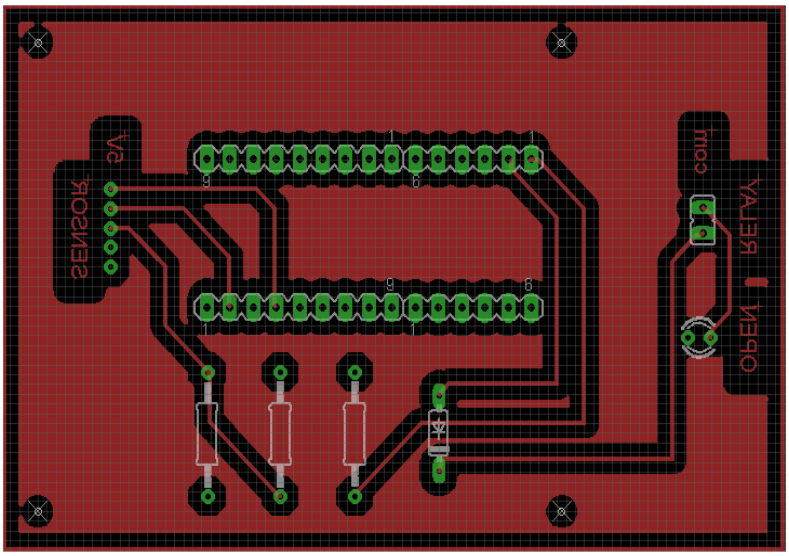


Fig.4.1 vista superios de la placa PCB

El diseño de este circuito, tiene las conexiones del sensor biométrico del Arduino Nano, la salida para un relay y un led indicador de activación de la salida de relay.

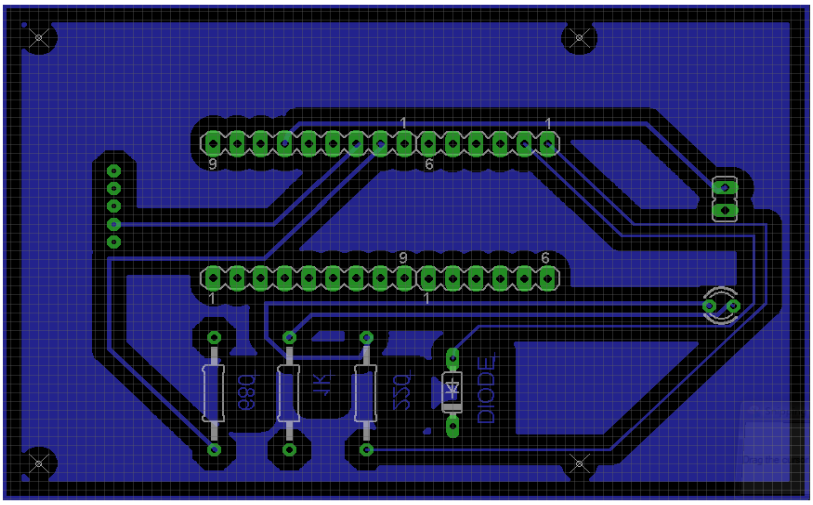


Fig.4.2 vista inferior de la placa PCB

El divisor de tensión esta formado por el punto de conexión común entre las resistencias de 680 Ω y 1k Ω, cuya conexión va directa al pin Rx del sensor biométrico, un esquema de la conexión se muestra en la figura 4.3

4.2 Esquema de conexión Arduino-Sensor:

La figura 4.3 muestra una interfaz de conexión entre el sensor biométrico GT511C3 y el Arduino Nano, en la cual se muestra la disposición de las resistencias para realizar una correcta comunicación entre los dispositivos.

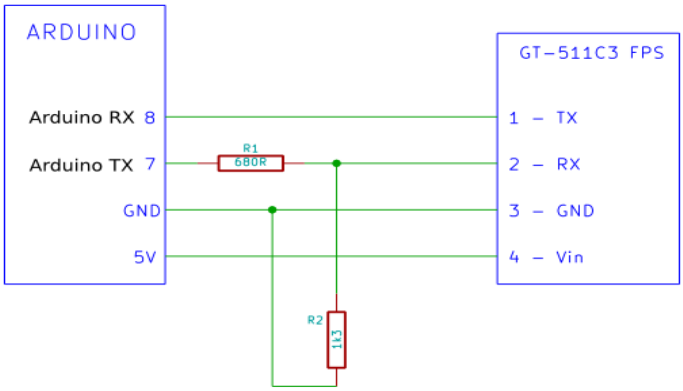


Fig.4.3 divisor de tensión

CAPITULO 5: Resultados de la ejecución del proyecto

5.1 Login de la aplicación Web:

Al ingresar al servidor que contiene la aplicación que controla el sistema de identificación de huellas, se pide un usuario y posword para ingresar. La figura 5.1 muestra la vista del login.

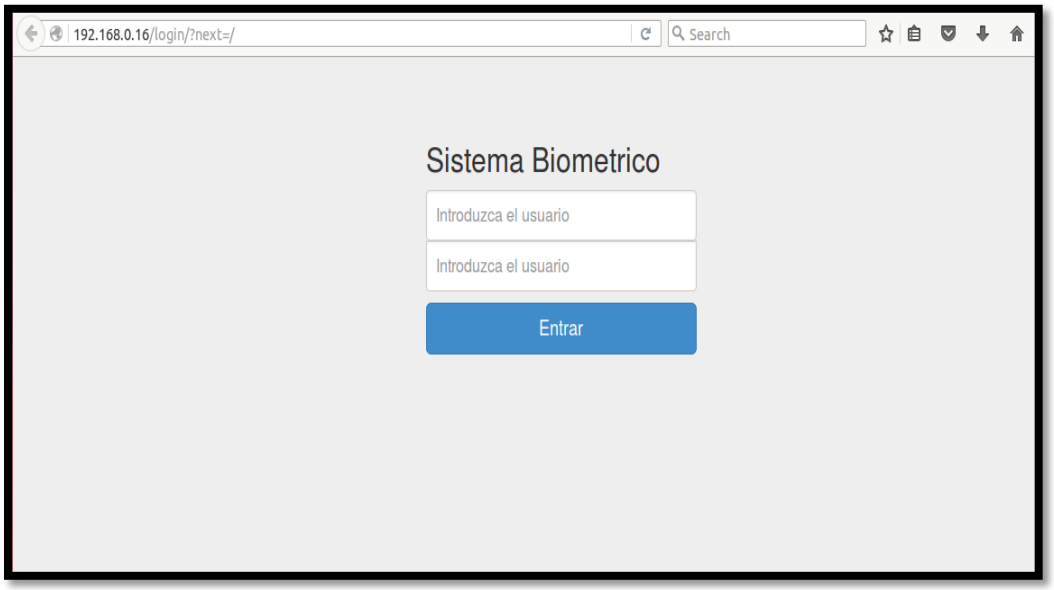


Fig.5.1 vista del login de la aplicación

En este caso tal como se muestra en la figura 5.1 se pide el usuario y pasword para ingresar a la administración de la aplicación siendo las credenciales requeridas las siguientes:

Usuari: Pi

Pasword:rasp

5.2 Interfaz de control de la aplicación:

Al ingresar usuario y pasword el sistema presenta la vista principal, es esta la que permite tener el control sobre el sistema, tales como validar huella de un usuario, registrar la huella de un usuario nuevo y ver las personas que están registradas en el sistema de identificación de huellas.

Fig.5.2 vista prncipal de la aplicación

Esta vista, realiza el control de las tareas antes mencionadas mediante la interrogación del sensor para realizar las tareas de registro y validación de huellas.

La vista hace las peticiones al sensor mediante el envío de comandos por medio del puerto serial de la Raspberry pi 2.0 que en este caso responde al puerto `/dev/ttyUSB0`, conectada a la interfaz de comunicación con el sensor, que en este caso es el Arduino Nano.

5.3 Vista registrar huella:

Al ejecutar el link Registrar Huella de la vista principal, se ejecuta la terminal shell de la Raspberry Pi 2.0 donde se pide que se presione el dedo a registrar sobre el sensor biométrico.

Si la huella es registrada con éxito, el aplicativo Web genera la vista mostrada en la figura 5.3

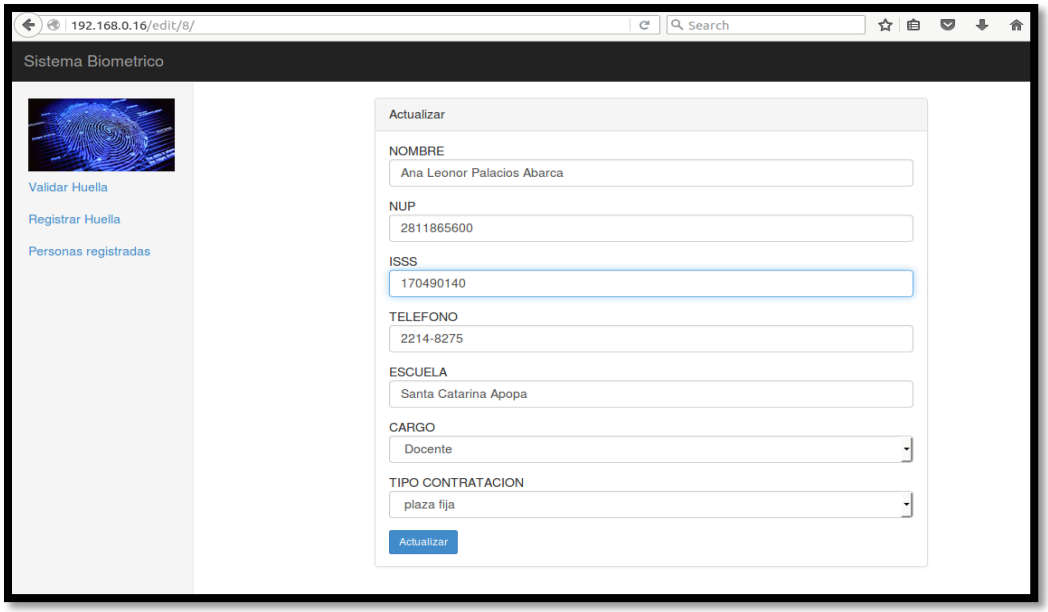


Fig.5.3 vista donde se piden datos del usuario

En la figura 5.3 se muestra una vista donde se piden los datos personales de un usuario a registrar en la base de datos del sistema.

Este formulario almacena la información en una base de datos en ***mysql*** cuyo nombre es ***sistema biométrico.***

5.4 Vista registro de personas:

En esta vista, se muestra un listado de personas que son o han sido registradas en la base de datos, son estas personas las que tiene el acceso hacia algún lugar restringido específico, mediante la validación y registro de huella.

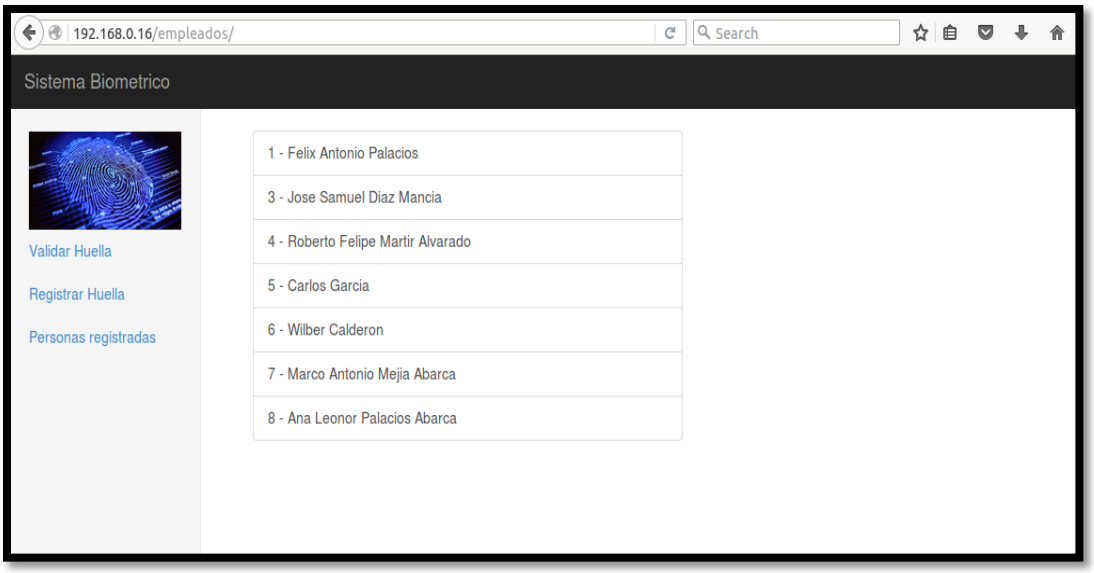


Fig.5.4 vista registro de personas

Cada una de estas personas registradas poseen una ficha la cual muestra los datos personales que fueron requeridos al momento de registrar una huella.

Al dar click sobre uno de los nombres registrados en la base de datos, el aplicativo web muestra una nueva vista, en la que aparecen los datos personales del usuario seleccionado, pero con la particularidad que se pueden editar los datos de este usuario registrado o dar de baja del sistema de validación de huellas.

La vista también muestra la hora y fecha de acceso hacia algún lugar restringido o la salida de esta persona del lugar, la figura 5.5 muestra la vista generada.

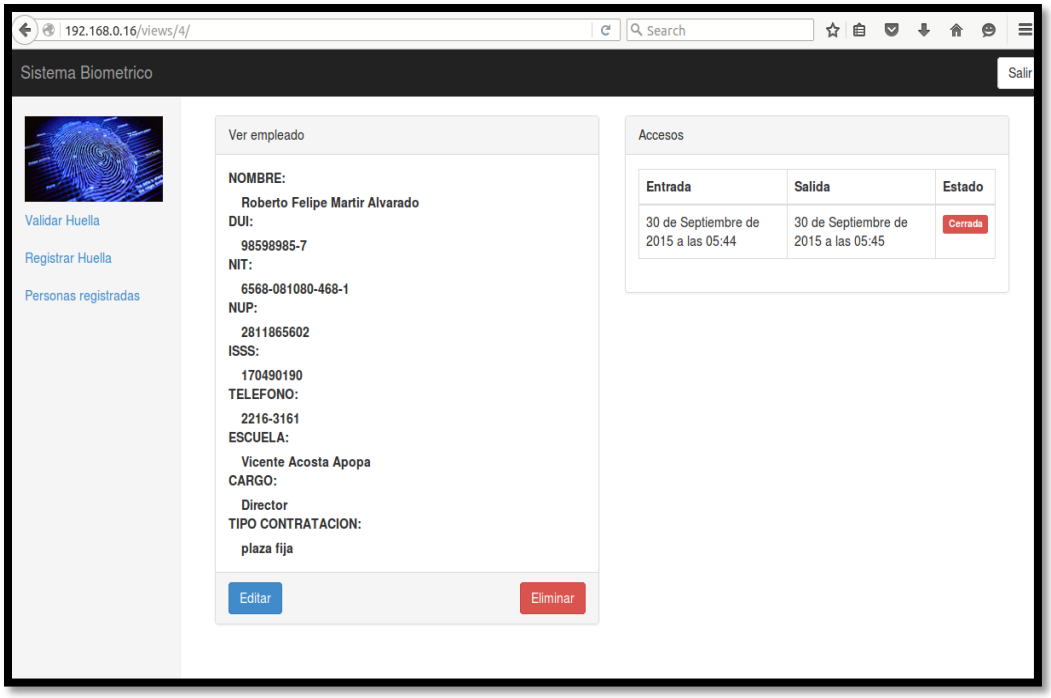


Fig.5.5 vista editar, eliminar registro

Presupuesto

Un estimado de cuanto cuesta este proyecto, se detalla a continuación, con los valores unitarios de cada uno de los componentes que conforman el sistema, además se consideran los costos de envío de los componentes que fueron adquiridos fuera de el país.

Presupuesto del Sistema				
Articulo	***Costo Unitario ($)***	***Número de Unidades***	***Costo de Envió ($)***	***Costo Total ($)***
Placa Raspberry Pi 2.0	51.49	1	14.95	66.44
Placa Arduino Nano	6.99	1	10.25	17.24
Sendor Biometrico GT511C3	49.95	1	11.75	61.7
Pantalla LCD 3.5 pulgadas	24.98	1	17.95	42.93
Chasis transparente 13 x 5.8 cm	12	1	0	12
Placa de cobre 20 x 10 cm	9.75	1	0	9.75
Diodo Led Azul	0.35	1	0	0.35
Diodo Led Amarillo	0.35	1	0	0.35
Usb wifi	7.99	1	10.35	18.34
Percloruro de sodio	2.75	1	0	2.75
			Total	231.85

Como se puede observar la implementación de este proyecto, tiene un costo de ***$231.85,*** al considerar unicamente la adquicisión de los componentes que lo comforman.

Conclusiones

Se logró implementar un proyecto utilizando el framework de django, el cual corre en un servidor de producción como lo es Apache 2.0. el cual esta alojado en el ordenador de bajo costo en la Raspberry Pi 2.0. este corre un aplicativo web que presenta una interfaz gráfica en el cambas de un navegador Web, la cual presenta tareas como el registro de la huella de una persona, la validación de huellas ya existentes en una base de datos, hacer manipulaciones con la base de datos tales como editar usuario o dar de baja del sistema de huellas dactilares a un usuario específico.

El sistema corre mediante la conexión a internet del dispositivo utilizado como servidor Web y se puede acceder de forma local por cualquier otro equipo que esté en la misma red del servidor instalado en la Raspberry Pi.

La base de datos del sensor biométrico tiene la capacidad de guardar en su interior un maximo de 200 ID, los cuales son consultados y comparados con el registro creado en la base de datos de mysql.

Efectivamente se utiliza un Arduino Nano como interfaz de comunicación entre el sensor biométrico y la Raspberry Pi 2.0 ya que fue esta configuración que arrojó los datos esperados y generados por el sensor biométrico.

Referencias Bibliográficas

Threespot, Anarevv:Django

https://www.djangoproject.com/

La guia definitiva de django

Adrian Holovaty y Jacob Kaplan Mosss

Copyright @ 2015 Saul Garcia M.

Python Web Development whit Django

Jeff Forcier, Paul Bissex

ISBN-13: 978-0-13-235613-8

Bootstrap

Jake Spurlock

O'REILLY

ISBN: 978-1-449-34391-0

Sergio Infante Montero:Curso de Django

http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/

Karlrunge: VNC/Server

http://www.penguintutor.com/linux/tightvnc

Foro de Python: Recoger parámetros de un formulario con Python

http://chuwiki.chuidiang.org/index.php?title=Recoger_par%C3%A1metros_de_un_formulario_con_Python

Downloads :Raspberry pi Foundation UK 1129409

https://www.raspberrypi.org/downloads

django para perfeccionistas con deadlines

Leo Soto M.

Imagemaker IT

Arananet-net:Como instalar una pantalla LCD Waveshare en Raspberry pi 2.0

http://arananet-net.kinja.com/como-instalar-una-pantalla-lcd-waveshare-en-nuestra-ras-1693591009