SEP

SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MEXICO
Instituto Tecnológico de la Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Estructuración de un Servidor en Raspberry Pi como base de datos de Variables Ambientales"

POR

Ing. Rosendo De Avila Ortiz

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
Dr. Héctor Moreno Casillas

CODIRECTOR DE TESIS
Dr. Francisco Flores García

ISSN: 0188-9060

RIITEC: (14)-TMCIE-2017

Torreón, Coahuila. México
Junio 2017

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Laguna

Torreón, Coah., 21/Junio/2017
Dependencia: DEPI/CPCIE
Oficio: DEPI/CPCIE/069/2017
Asunto: Autorización de impresión de tesis.

C. De Ávila Ortiz Rosendo
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÈCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"Estructuración de un servidor Raspberry Pi con base de datos de variables ambientales"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (14)-TMCIE-2017**, para que proceda a la impresión del mismo.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO

DR. ARMANDO LONGORIA DE LA TORRE
Jefe de la División de Estudios de Posgrado e Investigación
del Instituto Tecnológico de la Laguna

ALT/JIHJ

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Laguna

Torreón, Coah., 23/Junio/2017

DR. ARMANDO LONGORIA DE LA TORRE
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"Estructuración de un servidor Raspberry Pi con base de datos de variables ambientales"

Desarrollado por el **C. De Ávila Ortiz Rosendo**, con número de control **M1513015** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Héctor A. Moreno Casillas
Asesor/Director de Tesis

Dr. Francisco G. Flores García
Co-director de Tesis

Dr. Enrique Cuan Durón
Comité Tutorial

M.C. Martín Vázquez Rueda
Comité Tutorial

Agradecimientos.

A mis asesores Dr. Héctor Moreno Casillas y Dr. Francisco Gerardo Flores García por el apoyo y orientación que me ofrecieron, gracias a sus conocimientos y enseñanzas ha sido posible la realización de este proyecto, y a mis compañeros y amigos Ángel Alberto Cabrera Lara y José Eduardo Mireles Méndez y mis padres por su apoyo incondicional.

Resumen

La contaminación del aire se produce cuando ciertos gases y partículas entran en contacto con la atmósfera, dependiendo de la concentración de contaminantes, de la prolongada exposición y de la sensibilidad, estos pueden perjudicar de forma seria la salud de personas, animales y plantas como lo advierten algunos estudios epidemiológicos. Esto puede producir efectos adversos como tos, irritaciones en ojos y garganta, problemas respiratorios, nerviosos y cardiovasculares; pudiendo llegando a causar cáncer en algunos casos, aumentando las visitas a urgencias, los ingresos hospitalarios y defunciones.

Este trabajo forma parte de un proyecto mayor. Una red de sensores con los cuales se pueden monitorear diferentes variables de la calidad del aire, que utiliza sensores comerciales y módulos de comunicación. Las mediciones tomadas por cada uno de los sensores serán almacenados en un servidor implementado en Raspberry Pi ya que cuenta con la tecnología necesaria para desarrollar el servidor con un bajo costo.

También se incursiona en la programación en lenguaje Python para la adecuación de la tarjeta como un servidor, además del diseño e implementación de la interfaz de

comunicación y visualización de los datos históricos.

Esto conlleva a la aplicación del protocolo TCP/IP que se refiere a un conjunto de protocolos para comunicaciones de datos, este conjunto toma su nombre de dos de sus protocolos más importantes, el protocolo **TCP** (*Transmision Control Protocol*) y el protocolo **IP** (*Internet Protocol*). También se empleara procesos de Cliente-Servidor el cual será desarrollado sobre el sistema operativo Linux.

Se logró la implementación de la Raspberry pi como servidor, utilizando la programación por hilos para identificar cada uno de los datos enviados.

Índice

Índice de figuras

Índice de Tablas

CAPITULO I

INTRODUCCION

En este trabajo de tesis se realiza la estructuración de un servidor en Raspberry Pi como base de datos de variables ambientales, con el propósito de generar un historial de la cantidad de CO_2 y CO en el aire, para proporcionar a las autoridades ambientales y de salud, investigadores, estudiantes, organismos de la sociedad civil y otros interesados, un panorama de las tendencias de la calidad del aire, con la finalidad de que cuenten con información robusta y confiable para diseñar y evaluar políticas públicas que permitan reducir los riesgos a la salud asociados con la exposición a los contaminantes atmosféricos.

Este trabajo de tesis está conformado por cuatro capítulos a continuación se describen de manera breve.

- En el capítulo uno se presentan antecedentes y objetivos que se buscan cumplir en este proyecto, así como el cronograma de actividades y presupuesto que se requirió.
- El capítulo dos contiene el marco teórico, en el que se da la descripción de lo que es el protocolo TCP/IP, programación por hilos, el sistema operativo Linux, el lenguaje de programación Python que es el utilizado para implementar el servidor y las características de la tarjeta Raspberry Pi.
- El capítulo tres contiene la metodología implementada para cumplir con el objetivo del proyecto.
- En el capítulo cuatro se presentan los resultados obtenidos del proyecto.
- Finalmente se comentan las conclusiones y recomendaciones.

1.1 Antecedentes.

En proyectos anteriores de medición de variables ambientales realizados en el Instituto Tecnológico de la Laguna, como el de "IMPLEMENTACIÓN DE UN MEDIDOR DE OZONO AMBIENTAL REMOTO USANDO COMUNICACIÓN GPRS" se utilizó una interfaz gráfica implementada con el software LabView de National Instruments. El cual permite, por medio del lenguaje de programación gráfico, el diseño de sistemas de adquisición de datos instrumentación y control, así como una gran compatibilidad para trabajar con equipo técnico, ya sean tarjetas de medición, adquisición y procesamiento de datos [1].

El proyecto se realizara en el Tecnológico Nacional de México Campus IT la Laguna, en el área de Posgrado; ubicado en Blvd. Revolución y Av. Tecnológico de la Laguna S/N, en la ciudad de Torreón Coahuila y en el periodo de enero 2015 a diciembre de 2016.

1.2 Planteamiento del Problema.

En el proyecto referido en la sección anterior, LabView fue seleccionado debido a que su área de mayor aplicación es en sistemas de medición, como monitoreo de procesos y aplicaciones de control. Además es compatible con muchas herramientas de desarrollo similares y puede trabajar con programas de otras áreas de aplicación, como por ejemplo Matlab. Esto a su vez es una desventaja importante para el desarrollo de proyectos de sistemas de monitoreo. La robustez del Software eleva el costo de cada una de las herramientas o dispositivos de LabView. Ya que para manejar una red más compleja de sensores se requieren diversas herramientas de LabView con un costo adicional [1].

Este proyecto trata de dar una solución económica, sencilla y de mayor cobertura a los medidores de variables y sistema de monitoreo ambiental.

El problema a resolver es Estructurar de un servidor en Raspberry Pi como base de datos de variables ambientales.

1.3 Objetivo general.

- Estructurar un servidor en Raspberry Pi como base de datos de variables ambientales.

1.4 Objetivos específicos.

- Implementar la programación en lenguaje Python para la adecuación de la tarjeta como un servidor, mediante el protocolo de comunicación TCP/IP desarrollado sobre el sistema operativo Linux.
- Diseño e implementación de página web para visualizar los datos obtenidos.
- Diseñar e implementar la interfaz de comunicación y visualización de los datos históricos.
- Almacenar datos históricos de las mediciones de concentración de CO_2 y CO en puntos estratégicos del Instituto Tecnológico de la Laguna.

1.5 Justificación.

Este proyecto permitirá: monitorear la calidad del aire, en cuanto a CO y CO_2 se refiere y llevar un registro del historial de la contaminación del aire en determinados puntos de la ciudad, lo cual podría permitir:

- Relacionar la concentración con la incidencia de enfermedades respiratorias.

- Difundir resultados.
- Proponer medidas para la reducción de la contaminación en la ciudad.
- Promover la aplicación de las tecnologías para el almacenamiento de datos históricos.

Se pretende desarrollar un sistema robusto, confiable y con un costo de implementación accesible. Además, la reducción de los impactos en la salud asociados con la contaminación del aire, redunda en beneficios económicos para toda la sociedad, no sólo por los ahorros en el tratamiento de enfermedades cardiovasculares y cardiopulmonares, diabetes y otros padecimientos, sino también porque se reducen las pérdidas en la productividad laboral, en el rendimiento de los cultivos agrícolas, los bosques y en la visibilidad.

1.5.1 Impacto social.

Se pretende que mediante el historial que se generara con la aplicación de este proyecto, se vean beneficiados los aproximadamente 700, 000 habitantes de Torreón, al poder contar con la información de la calidad del aire teniendo disponible una herramienta de fácil acceso y consulta, y como consecuencia concientizarlos de los valores de la contaminación ambiental o calidad del aire.

1.5.2 Impacto tecnológico.

El desarrollo del presente proyecto dejara antecedentes, para generar actualizaciones de aplicaciones o interfaces para el almacenamiento de datos y despliegue de los mismos, en plataformas de acceso global, de forma rápida y sencilla, que mediante cualquier PC o Smartphone el usuario pueda consultar.

1.5.3 Impacto económico.

A diferencia de estaciones de monitoreo ambiental que emplean sistemas robustos, este proyecto pretende disminuir los costos en un 50%, y tener una mayor cobertura de comunicación y visualización de los datos (un sistema más flexible a las tecnologías actuales), ya que como aplicación para los usuarios es gratuito.

1.5.4 Impacto ambiental.

El desarrollo y la implementación del proyecto generan un mínimo de contaminación ambiental, debido al servidor ya que este deberá estar en funcionamiento en todo momento, pero; no produce desechos electrónicos. Además el proyecto buscara presentar e implementar medidas preventivas de contaminación.

1.5.5 Viabilidad de la investigación.

El proyecto es viable porque se cuenta con: la asesoría por parte de especialistas, los conocimientos, las habilidades, las instalaciones y los recursos necesarios. Además el tiempo de desarrollo de la investigación es relativamente corto.

1.6 Metodología.

En la Figura 1.6.1, se presenta la secuencia de pasos que se realizará para cumplir con el objetivo del proyecto.

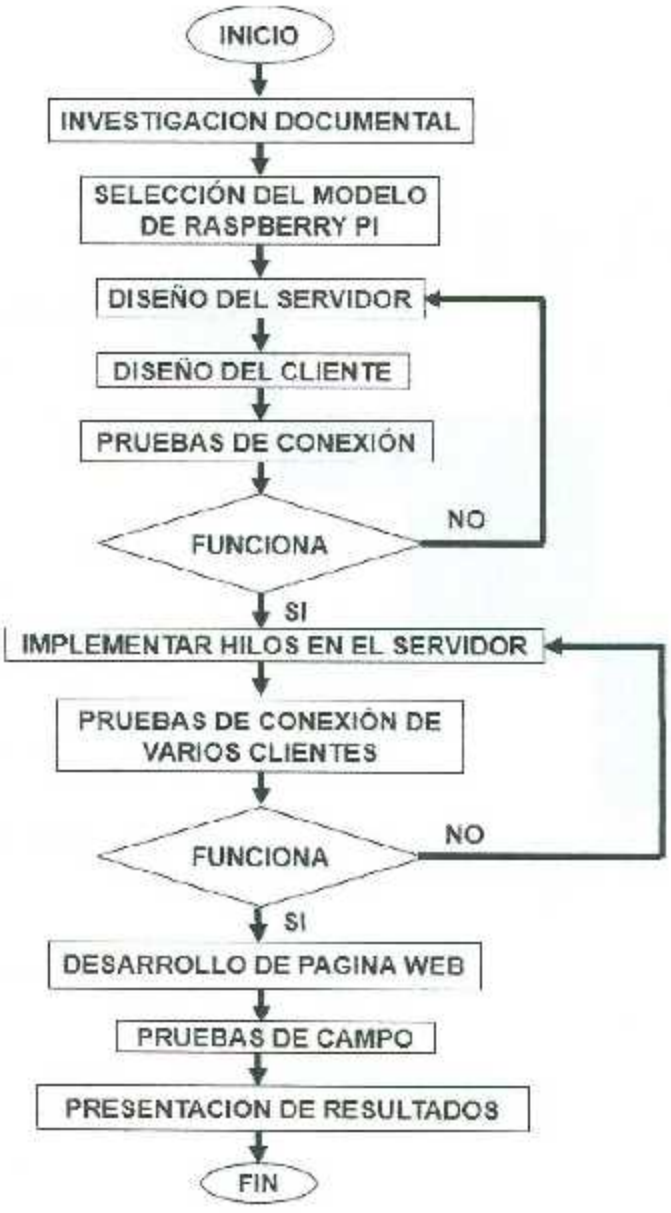


Figura 1.6.1. *Diagrama de flujo de la metodología.*

1.7 Cronograma.

El cronograma de la Tabla 1.61.1, muestra de forma organizada en meses el desarrollo de las actividades.

Table 1.61. *Cronograma de planeación.*

ACTIVIDAD	ENE.	FEB.	MAR	ABRIL	MAY	JUN	JUL	AGO	SEP.	OCT	NOV	DIC
INVESTIGACION DOCUMENTAL	■	■										
SELECCIÓN DE MODELO DE RASPBERRI PI		■										
DISEÑO DEL SERVIDOR			■	■	■	■	■					
DISEÑO DEL CLIENTE					■	■	■					
PRUEBAS DE CONEXIÓN					■	■	■					
IMPLEMENTAR HILOS EN EL SERVIDOR								■	■			
PRUEBAS DE CONEXIÓN DE VARIOS CLIENTES								■	■			
DESARROLLO DE PÁGINA WEB										■	■	
PRUEBAS DE CAMPO											■	
PRESENTACION DE ESULTADOS												■

1.8 Presupuesto.

El presupuesto requerido para este proyecto es un total de $ 2,500.00 MXN, que es el costo de la tarjeta Raspberry Pi.

CAPITULO II

MARCO TEORICO

2.1 Protocolo TCP/IP

Las siglas TCP/IP se refieren a un conjunto de protocolos para comunicaciones de datos. Este conjunto toma su nombre de dos de sus protocolos más importantes, el protocolo TCP (*Transmission Control Protocol*) y el protocolo IP (*Internet Protocol*).

La evolución del protocolo TCP/IP siempre ha estado muy ligada a la de Internet. En 1969 la agencia de proyectos de investigación avanzada, ARPA (*Advanced Research Projects Agency*) desarrolló un proyecto experimental de red conmutada de paquetes al que denominó ARPAnet [2].

ARPAnet comenzó a ser operativa en 1975, pasando entonces a ser administrada por el ejército de los EEUU. En estas circunstancias se desarrolla el primer conjunto básico de protocolos TCP/IP. Posteriormente, y ya entrados en la década de los ochenta, todos los equipos militares conectados a la red adoptan el protocolo TCP/IP y se comienza a implementar también en los sistemas Unix. Poco a poco ARPAnet deja de tener un uso exclusivamente militar, y se permite que contros de investigación, universidades y empresas se conecten a esta red [2]. Se habla cada vez con más fuerza de Internet y en 1990 ARPAnet deja de existir oficialmente.

En los años sucesivos y hasta nuestros días las redes troncales y los nodos de interconexión han aumentado de forma imparable. La red Internet parece expandirse sin límite, aunque manteniendo siempre una constante: el protocolo TCP/IP. En efecto, el

gran crecimiento de Internet ha logrado que el protocolo TCP/IP sea el estándar en todo tipo de aplicaciones telemáticas, incluidas las redes locales y corporativas. Y es precisamente en este ámbito, conocido como Intranet, donde TCP/IP adquiere cada día un mayor protagonismo [3]. La popularidad del protocolo TCP/IP no se debe tanto a Internet como a una serie de características que responden a las necesidades actuales de transmisión de datos en todo el mundo, entre las cuales destacan las siguientes:

• Los estándares del protocolo TCP/IP son abiertos y ampliamente soportados por todo tipo de sistemas, es decir, se puede disponer libremente de ellos y son desarrollados independientemente del hardware de los ordenadores o de los sistemas operativos.

• TCP/IP funciona prácticamente sobre cualquier tipo de medio, no importa si es una red Ethernet, una conexión ADSL o una fibra óptica.

• TCP/IP emplea un esquema de direccionamiento que asigna a cada equipo conectado una dirección única en toda la red, aunque la red sea tan extensa como Internet.

La naturaleza abierta del conjunto de protocolos TCP/IP requiere de estándares de referencia disponibles en documentos de acceso público. Actualmente todos los estándares descritos para los protocolos TCP/IP son publicados como RFC (*Requests for Comments*) que detallan lo relacionado con la tecnología de la que se sirve Internet: protocolos, recomendaciones, comunicaciones, etcétera [3].

2.2 Modelo Cliente-Servidor

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) (Figura 2.1). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras [4].

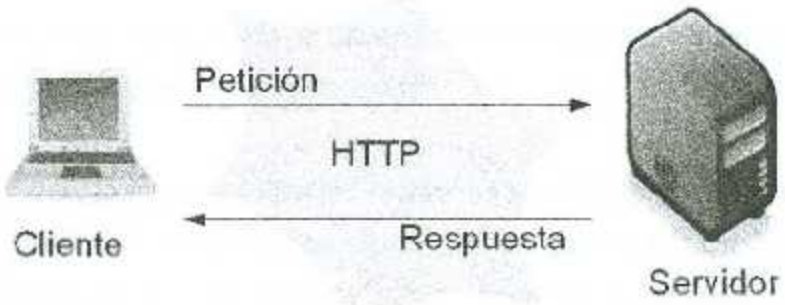


Figura 2.1. *Modelo Cliente/Servidor.*

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más adecuadas a sus características. Si esto se aplica tanto a clientes como servidores se entiende que la forma más estándar de aplicación y uso de sistemas Cliente/Servidor es mediante la explotación de las PC's a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales tipo mainframe. Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el o los procesos cliente sólo se ocupan de la interacción con el usuario (aunque esto puede variar) [5].

En otras palabras la arquitectura Cliente/Servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento.

Esta arquitectura permite distribuir físicamente los procesos y los datos en forma más eficiente lo que en computación distribuida afecta directamente el tráfico de la red, reduciéndolo grandemente.

2.3 Cliente.

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término *front-end*.

El Cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red [5].

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

2.4 Servidor.

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término *back-end* [6].

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.5 Características de la arquitectura Cliente/Servidor

Las características básicas de una arquitectura Cliente/Servidor son [6]:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco y input-output devices.

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.

- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.

- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.

- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.

- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

2.6 Sockets

Los sockets son un sistema de comunicación entre procesos de diferentes máquinas de una red. Más exactamente, un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.

Fueron popularizados por Berckley Software Distribution, de la universidad norteamericana de Berkley. Los sockets han de ser capaces de utilizar el protocolo de streams TCP (Transfer Control Protocol) y el de datagramas UDP (User Datagram Protocol) [7].

Utilizan una serie de primitivas para establecer el punto de comunicación, para conectarse a una máquina remota en un determinado puerto que esté disponible, para escuchar en él, para leer o escribir y publicar información en él, y finalmente para desconectarse (Figura 2.2).

Figura 2.2. Estructura de un socket

2.6.1 Funcionamiento genérico

Normalmente, un servidor se ejecuta sobre una computadora específica y tiene un socket que responde en un puerto específico. El servidor únicamente espera, escuchando a través del socket a que un cliente haga una petición.

En el lado del cliente: el cliente conoce el nombre de host de la máquina en la cual el servidor se encuentra ejecutando y el número de puerto en el cual el servidor está conectado. Para realizar una petición de conexión, el cliente intenta encontrar al servidor en la máquina servidora en el puerto especificado (**¡Error! No se encuentra el origen de la referencia.**) [7].

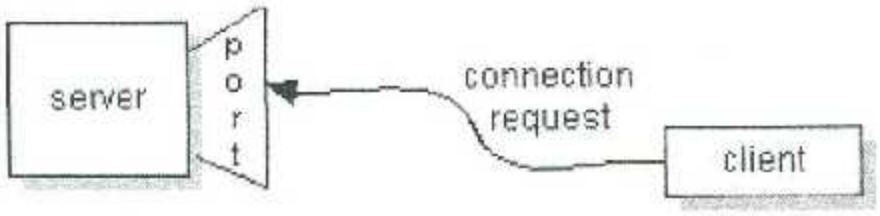


Figura 2.3. *Conexión del socket*

Si todo va bien, el servidor acepta la conexión. Además de aceptar, el servidor obtiene un nuevo *socket* sobre un puerto diferente. Esto se debe a que necesita un nuevo *socket* (y, en consecuencia, un numero de puerto diferente) para seguir atendiendo al *socket* original para peticiones de conexión mientras atiende las necesidades del cliente que se conectó (**¡Error! No se encuentra el origen de la referencia.**).

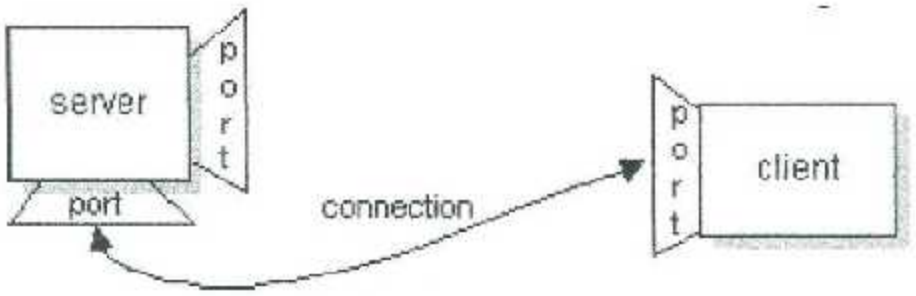


Figura 2.4. Nuevo socket creado

Por la parte del cliente, si la conexión es aceptada, un socket se crea de forma satisfactoria y puede usarlo para comunicarse con el servidor. Es importante darse cuenta que el socket en el cliente no está utilizando el número de puerto usado para realizar la petición al servidor. En lugar de éste, el cliente asigna un número de puerto local a la máquina en la cual está siendo ejecutado. Ahora el cliente y el servidor pueden comunicarse escribiendo o leyendo en o desde sus respectivos sockets [7].

2.7 Hilos (threads)

Las computadoras serían mucho menos útiles si no pudiéramos hacer más de una cosa a la vez. Si no pudiéramos, por ejemplo, escuchar música en nuestro reproductor de audio favorito mientras leemos un tutorial de Python en Mundo Geek.

Pero, ¿cómo se conseguía esto en computadoras antiguas con un solo núcleo / una sola CPU? Lo que ocurría, y lo que ocurre ahora, es que en realidad no estamos ejecutando varios procesos a la vez (se llama proceso a un programa en ejecución), sino que los procesos se van turnando y, dada la velocidad a la que ejecutan las instrucciones,

nosotros tenemos la impresión de que las tareas se ejecutan de forma paralela como si tuviéramos multitarea rea [8].

Cada vez que un proceso distinto pasa a ejecutarse es necesario realizar lo que se llama un cambio de contexto, durante el cual se salva el estado del programa que se estaba ejecutando a memoria y se carga el estado del programa que va a entrar a ejecutarse.
En Python podemos crear nuevos procesos mediante la función os.fork, que ejecuta la llamada al sistema fork, o mediante otras funciones más avanzadas como popen2.popen2, de forma que nuestro programa pueda realizar varias tareas de forma paralela.[2]

Sin embargo el cambio de contexto puede ser relativamente lento, y los recursos necesarios para mantener el estado demasiados, por lo que a menudo es mucho más eficaz utilizar lo que se conoce como threads, hilos de ejecución, o procesos ligeros.
Los threads son un concepto similar a los procesos: también se trata de código en ejecución. Sin embargo los threads se ejecutan dentro de un proceso, y los threads del proceso comparten recursos entre sí, como la memoria, por ejemplo [8].

El sistema operativo necesita menos recursos para crear y gestionar los threads, y al compartir recursos, el cambio de contexto es más rápido. Además, dado que los threads comparten el mismo espacio de memoria global, es sencillo compartir información entre ellos: cualquier variable global que tengamos en nuestro programa es vista por todos los threads.

2.8 Linux

Los antecedentes de Linux hay que buscarlos en el sistema operativo y los orígenes de UNIX se remontan a 1964. En este año, *Bell Telephone Laboratories* de AT&T, *General Electric Company* y el MIT (Instituto Tecnológico de Massachusetts) se plantearon desarrollar un nuevo sistema operativo en tiempo compartido para una máquina GE 645 (de General Electric) al que denominaron MULTICS. Los objetivos marcados inicialmente consistian en proporcionar a un conjunto amplio de usuarios una gran capacidad de computación y la posibilidad de almacenar y compartir enormes cantidades de datos si éstos lo deseaban [9].

Todos estos objetivos eran demasiado ambiciosos para la época, sobre todo por las limitaciones del hardware. Como consecuencia de ello los trabajos en el sistema operativo iban muy retrasados. Debido a eso, *Bell Laboratories* decidió dar por terminada su participación en el proyecto. A pesar del fracaso de MULTICS, las ideas empleadas para su diseño no cayeron en el olvido, sino que influyeron mucho en el desarrollo de UNIX y de otros sistemas operativos posteriores.

Ken Thomson, uno de los miembros del *Computing Scimce Reserch Center* de los *Laboratorios Bell*, encontró un computador DEC (*Digital Equipment Corporation*) PDP-7 inactivo y se puso a desarrollar en él un juego denominado *Space Travel.* El desarrollo de ese juego propició que Thompson adquiriese muchos conocimientos relacionados con la máquina en la que estaba trabajando. Con objeto de crear un entorno de trabajo agradable. Thompson, al que posterior mente se le unió Dennis Ritchie, se propuso la creación de un nuevo sistema operativo, al que denomino UNIX. Ritchie habia trabajado anteriormente en el proyecto MULTICS, de mucha influencia en el nuevo sistema operativo. Como ejemplos de esa influencia podemos citar la organización básica del sistema de archivos, la idea del intérprete de órdenes (shell) como proceso de usuario, e incluso el propio nombre UNIX deriva de MULTICS [9].

Aunque esta primera versión de UNIX prometía mucho, su potencial no pudo demostrarse hasta que se utilizó en un proyecto real. Asi pues, mientras se planeaban las pruebas para patentar el nuevo producto, este fue trasladado a un computador PDP-11 de Digital en una segunda versión. En 1973 el sistema operativo fue rescrito en lenguaje C en su mayor parte. C es el lenguaje de alto nivel (las versiones anteriores del sistema operativo habían sido escritas en ensamblador), lo que propicio que el sistema tuviera una gran aceptación por parte de los nuevos usuarios. El nuero de instalaciones en Bell laboratorios creció hasta 25, aproximadamente, y su uso también se difundió gradualmente a unas universidades con propósitos educacionales.

Tras la distribución de la Version 7, UNIX se convirtió en un producto y no solo en una herramienta de investigación o educacional, debido a que el UNIX *Support Group* (USG) asumió la responsabilidad y el control administrativo del *Research Grup* en la distribución de UNIX dentro de AT&T.

La Modularidad, la sencillez de diseño y el pequeño tamaño, hicieron que muchas entidades, tales como Rand, varias Universidades e incluso DEC, se pusieran a trabajar sobre él. La universidad de Berkeley California desarrollo una variante del sistema UNIX para maquinas VAX. Esta variante incorporaba varias características interesantes, tales como memoria virtual, paginación por demanda y sustitución de páginas, con lo cual se permitía la ejecución de programas mayores que la memoria física. (3BSD Berkeley *Software Distributions*). Todo el trabajo desarrollado en la universidad de Berkeley para crear BSD impulso a la *Defense Advanced Research Projects* Agency (DARPA) a financiar a Berkeley en el desarrollo de un sistema UNIX estándar de uso oficial (4BSD). Los trabajos en 4BSD para DARPA fueron dirigidos por expertos en redes y UNIX, DARPA Internet (TCP/IP). Este soporte se facilitó de un modo general En 4.2BSD es posible la comunicación uniforme entre los distintos dispositivos de la red, incluyendo redes locales (LAN), como *Ethernet y Token Ring*, y existen redes de ordenadores (WAN), como la Arpanet de DARPA.

Los sistemas UNIX actuales no se reducen a la Version 8, System V o BSD, sino que la mayoría de los fabricantes de micro y miniordenadores ofrecen su UNIX particular [9].

2.8.1 Aparición de Linux

Linux es un sistema operativo de distribución libre desarrollado inicialmente por Linus Torvalds en la Universidad de Helsinki (Finlandia). Una comunidad de programadores expertos en UNIX, han ayudado en el desarrollo, distribución y depuración de este sistema operativo. El núcleo de Linux no contiene código desarrollado por AT&T ni por ninguna otra fuente propietaria. La mayoría del software disponible en Linux ha sido desarrollado por el proyecto GNU de la *Free Software Foundation of Cambridge* (Massachusetts).

Con la aparición de ordenadores personales potentes aparece Linux. Linux se inspiró en Minix, un pequeño sistema UNIX desarrollado por Andrew S. Tanenbaum. Los primeros desarrollos de Linux tenían que ver con la conmutación de tareas en el microprocesador 80386 ejecutado en modo protegido, todo ello escrito en lenguaje ensamblador [10].

El 5 de octubre de 1991 Linux dio a conocer la primera versión oficial de Linux, ésta fue la versión 0.02, (generalmente al software sólo se le asigna como número de versión 1.0 cuando se supone que está en su mayoría libre de errores), esto ocurrió en marzo de 1992.

Actualmente Linux es un UNIX en toda regla, compatible POSIX, capaz de ejecutar X-Window, TCP/IP, Emacs, UUCP, correo electrónico, servicios de noticias, etc. La mayoría de los paquetes software de libre distribución han sido portados a Linux y cada vez son más las aplicaciones comerciales disponibles. Actualmente Linux soporta casi todo el hardware existente en el entorno PC y ha sido portado con éxito a otras plataformas como PowerPC de IBM, SPARC de Sun Microsystems o Macintosh. Su robustez y el hecho de

ser gratuito ha propiciado que Linux lo empleen como herramienta de desarrollo desde entidades de investigación como la NASA, hasta Dream Works, Pixar o Industrial Light and Magic [10].

En el campo de los servidores, Linux tiene en la actualidad una importante cuota de mercado y es en el ámbito de escritorio donde está teniendo un crecimiento importante. En resumen, Linux ha pasado en breve de ser un sistema operativo marginal a convertirse en una alternativa a sistemas comerciales como Windows o MacOS X.

2.8.2 Esquema de un sistema Linux

La configuración básica de un sistema Linux, de equipos se refiere, es la mostrada en la **¡Error! No se encuentra el origen de la referencia.**. A grandes rasgos, podemos distinguir las siguientes partes:

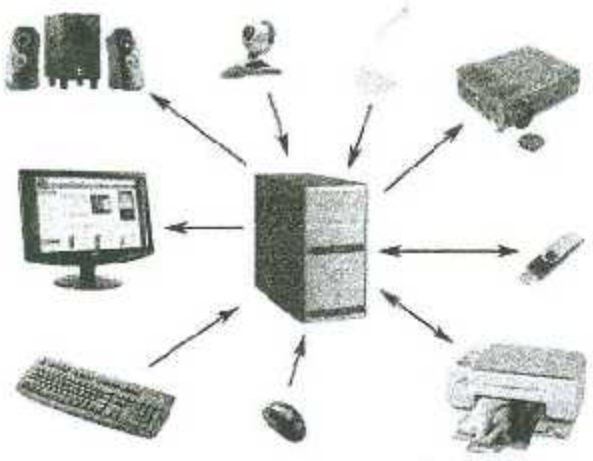


Figura 2.5. *Esquema básico de un sistema Linux.*

Unidad de proceso. La unidad de proceso es el verdadero corazón del sistema, puesto que en ella se ejecutan todos los programas, tanto los de los usuarios como los del propio sistema. El término unidad de proceso engloba elementos tales como la memoria, la unidad de manejo de memoria (UMM), los procesadores en coma flotante, los dispositivos de acceso a memoria (ADM), etc [10].

Dispositivos de almacenamiento secundario. Son los elementos en los que vamos a guardar toda la información de forma permanente.

Dispositivos periféricos. Son aquellos elementos que añadidos al sistema computador realizan, sobre todo, funciones de comunicación con las personas, y entre ellos podemos citar el ratón, la pantalla, el módem, la impresora, el lápiz USB, etc. Todos estos dispositivos están conectados a la central de proceso, la cual se encarga de manejarlos y planificarlos para que puedan ser compartidos sin problemas entre los usuarios.

2.9 Python

Básicamente, Python es un lenguaje de programación de alto nivel, *interpretado y multipropósito*. En los últimos años su utilización ha ido constantemente creciendo y en la actualidad es uno de los lenguajes de programación más empleados para el desarrollo de software.

Python puede ser utilizado en diversas plataformas y sistemas operativos, entre los que podemos destacar los más populares, como Windows, Mac OS X y Linux. Pero, además, Python también puede funcionar en *smartphones*. Nokia desarrolló un intérprete de este lenguaje para su sistema operativo Symbian [11].

Con este lenguaje podemos desarrollar software para aplicaciones científicas, para comunicaciones de red, para aplicaciones de escritorio con interfaz gráfica de usuario (GUI), para crear juegos, para *smartphones* y por supuesto, para aplicaiones web. Empresas y organizaciones del calibre de *Industrial Light & Magic, Walt Disney, la NASA,* Google,Yaoo!, Red Hat y Nokia hacen uso intensivo de este lenguaje para desarrollar sus productos y servicios [11].

Las principales caracteristicas por las que se distingue Python son por ser un lenguaje potente, flexible y con una sintaxis clara y concisa. Además, no requiere dedicar tiempo a su compilación debido a que es interpretado.

Python es *open source*, cualquiera puede contribuir a su desarrollo y divulgación. Además, no es necesario pagar ninguna licencia para distribuir software desarrollado con este lenguaje. Hasta su interprete se distribuye de forma gratuita para diferentes plataformas.

2.9.1 Historia de Python.

El origen del lenguaje de Python se remonta a principios de los noventa. Por este tiempo, un investigador llamado Guido Van Rossum,que trabajaba en el centro de investigación CWI (*Centrum Wiskunde & Informatica*) de Amsterdam, es asignado a un proyecto que consistia en el desarrollo de un sistema operativo distribuido llamado Amoeba [12].

La primera versión del lenguaje ve la luz en 1991, pero no es hasta tres años después cuando deside publicarse la versión 1.0. Inicialmente el CWI decidió liberar el intérprete del lenguaje bajo una licencia *open source* propia, pero en septiembre de 2000 y coincidiendo con la publicación de la versión 1.6, se toma la decisión de cambiar la licencia por una que se a compactible con la licencia GPL (GNU General Public Licence).

Esta nueva licencia se denominará *Python Software Foundaion Licence* y se diferencia de la GPL al er una licencia no *copyleft*. Este hecho implica que es posible modificar el código fuente y desarrollar código derivado sin la necesidad de hacerlo *open source*.

Hasta el momento solo se han liberado tres versiones principales, teniendo cada una de ellas diversas actualizaciones. En lo que respecta a la versión 2, la última en ser liberada fue la 2.7, en julio de 2010. En el momento de escribir estas líneas, la versión 3 cuenta con la actualización 3.2, liberada en febrero de 2011. Ambas versiones de 2 y 3, son mantenidas por separado [12].

Entre las características de las primeras versiones de Python cabe destacar el soporte de la orientación a objetos, el manejo de excepciones y el soporte de estructuras de datos de alto nivel, como, por ejemplo, las listas y los diccionarios.

El desarrollo y promoción de Pyhton se lleva a cabo a través de una organización, sin ánimo de lucro, llamada *Python Software Fundation*, que fue creada en marzo de 2001. Entre las actividades que realiza esta organización destacan el desarrollo y distribución oficial de Python, la gestión de la propiedad intelectual del código y documentos realizados, así como la organización de conferencias y eventos dedicados a poner en contacto a todas aquellas personas interesadas en este lenguaje de programación.

2.9.2 Principales características.

Dos de las principales características del lenguaje Python son, por un lado que es *interpretado* y, por otro lado, que es *multiplataforma*. Lo primero significa que no es necesario compilar el código para su ejecución, ya que existe un intérprete que se encarga de leer el fichero fuente y ejecutarlo. Gracias a este funcionamiento es posible ejecutar el código en distintas plataformas y sistemas operativos sin necesidad de cambiar el código fuente, bastará con tener instalado el intérprete [13].

Habitualmente, a los programas en Pyhton se les denomina *scripts*. En realidad, *script* es el término que se suele emplear para los ficheros de código fuente escritos en Python, pudiendo un programa contar con uno o más de estos *scripts*.

En lo que respecta a la sintaxis del lenguaje cabe destacar su simplicidad; es decir, gracias a la misma, es sencillo escribir código que sea fácil de leer. Este factor es muy importante, ya que, además de facilitar el aprendizaje del lenguaje, también nos ayuda a que nuestro código sea más fácil de mantener [13].

Además de soportar la orientación a objetos, Python también nos permite utilizar otros paradigmas de programación, como, por ejemplo, la programación funcional y la imperativa.

Con respecto a la sintaxis, una de las diferencias más destacables es el uso de la *identición* diferentes niveles de identación son utilizados para marcar las sentenciasd que corresponden al mismo bloque. Por ejemplo, todas las sentencias que deban ser ejecutadas dentro de un bloque *if* llevarán el mismo nivel de identación, mientras que el resto usara un nivel diferente, incluida la sentencia que contiene la condición o condiciones del moncionado *if*. Además cada sentencia no necesita un punto y coma (;), como sí ocurre en lenguajes como C/C++, PHP, y Java. En Python basta con que cada sentencia valla en una línea diferente. Por otro lado tampoco se hace uso de las llaves ({}) para indicar el principio y fin de bloque. Tampoco se emplean palabras clave como *begin* y *end*. Simplemente se utilizan dos puntos (:) para marcar el comienzo de bloque y el cambio de identación se encarga de marcar el final [14].

A diferencia de lenguajes compilados, como C++, en Python existe un recolector de basura (*garbage collector*). Esto significa que no es necesario pedir y liberar memoria, de forma explícita, para crear y destruir objetos. El intérprete lo hará automáticamente

cuando sea necesario y el recolector se encargará de gestionar la memoria para evitar los temidos *memory leaks*.

2.10 La Raspberry Pi.

La Raspberry Pi (**¡Error! No se encuentra el origen de la referencia.**) es una placa de pequeño tamaño (8.5 × 5.4 cm) que incluye todo un ordenador completo. A estos dispositivos se les conoce por las siglas SBC (**Single Board Computer**). En su diseño se utiliza un SoC (**System on a Chip**), que incluye en un solo chip el procesador o CPU (ARMv6/ARMv7), la memoria RAM (512 MB/1 GB) y la tarjet gráfica o GPU (VideoCore IV) [15].

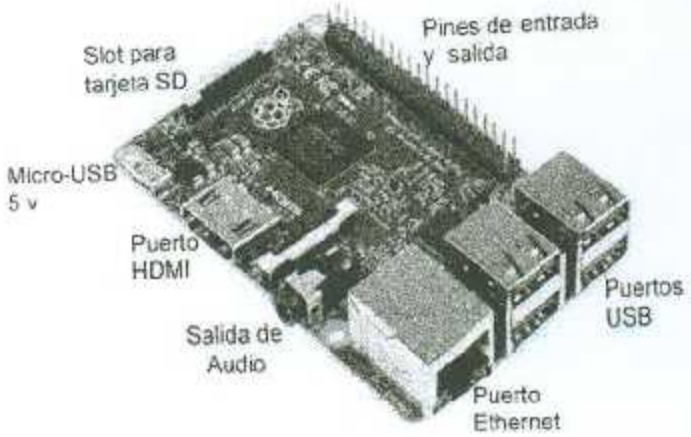


Figura 21.8.6. *Tarjeta Raspberry Pi*

Los primeros modelos que salieron al mercado consistían en dos tipos de placas: el modelo A, que estaba más bien destinado a desarrolladores y poseía menos prestaciones, y el modelo B, que es más usado por los usuarios domésticos. Este modelo incluye 512 MB o 1 GB de RAM, dos o cuatro puertos USB, un conector de red Ethernet 10/100, salida de vídeo HDMI y otra salida analógica de audio y vídeo. Como unidad de

almacenamiento utiliza una tarjeta SD y se alimenta a través de un conector mini USB, similar al de los cargadores para teléfonos móviles. Como sistema operativo puede usar una amplia variedad de distribuciones para Linux (entre ellas, Debian, Ubuntu, Fedora o Arch Linux) u otros sistemas como FreeBSD o RISC OS [15].

Los primeros diseños se realizaron en Reino Unido en el año 2006, pero no se empezó a comercializar hasta febrero de 2012 por parte de la Fundación Raspberry Pi, que es la que se encarga de su desarrollo.

En julio de 2014 se presenta una nueva placa, bajo el nombre de modelo B+, que es una actualización del anterior modelo B. Aparte del rediseño de algunos componentes electrónicos de la placa, las diferencias fundamentales entre ambas son la inclusión de 2 puertos USB más (lo que suma un total de 4), la sustitución de la tarjeta SD por una microSD y la fusión de los conectores analógicos (audio y vídeo) en un solo conector. Pero la parte fundamental, que es el SoC (CPU, GPU y RAM), es idéntica en los dos modelos de la clase B [15].

El 2 de febrero del 2015 salió al mercado una nueva placa: la Raspberry Pi2 modelo B. Visualmente es idéntica al modelo B+ anterior, pero añade dos importantes novedades: una CPU de cuatro núcleos (**quad core**) ARMv7 y 1 GB de memoria RAM. Todo lo demás se mantiene exactamente igual y, por ello, es totalmente compatible con el modelo B+.

Esta placa está destinada a desarrolladores y a personas a las que les gusta aprender, experimentar y trastear con electrónica [15]. Con ella pueden hacerse proyectos de todo tipo (robótica, domótica, etc.) pero la mayoría de los usuarios la utilizan para cosas sencillas y prácticas como las siguientes:

- Centro multimedia y servidor DLNA: se conecta a la TV por medio de la salida HDMI y permite reproducir vídeo y audio en alta definición; o servir contenido multimedia en streaming por DLNA a través de la red local.

- Servidor de almacenamiento e intercambio de ficheros (NAS y FTP), para guardar y/o compartir ficheros dentro de la red local o a través de internet.
- Nube personal: se puede instalar Apache y, si se desea, también PHP y MySQL para conseguir un sistema completo para webs y blogs, sin necesidad de contratar un hosting.

Debido a su bajo consumo (2.5 W), puede estar conectada las 24 horas del día, y como la CPU no se calienta apenas, no necesita ventiladores, por lo que es totalmente silenciosa.

2.11 Los anteriores modelos de la Raspberry Pi2

Cada evolución de la Raspberry Pi es una mejora respecto a la versión anterior. El modelo B consiguió actualizar la memoria RAM del modelo A para aumentar su velocidad; mientras que el B+ era una revisión de la totalidad, con puertos adicionales USB, un tipo diferente de tarjeta SD y más detalles. El modelo B+ presenta una diferente configuración y es mucho más grande que los otros modelos. La Pi2 mantiene el diseño de la B+ salvo que cambia el procesador y aumenta la memoria RAM. Sin embargo, cada uno de los modelos es aproximadamente del tamaño de una tarjeta de crédito. Definitivamente, con los modelos B+ y, sobre todo, la Pi2 notaremos importantes mejoras de velocidad con los programas de procesamiento de vídeo y gráficos intensivos [16].

2.11.1 El modelo A+

El modelo Raspberry Pi A+ únicamente tiene un puerto USB, no ofreciendo puerto Ethernet. Solo dispone de 256 MB de RAM. Consume alrededor de un tercio de la potencia del modelo B, así que es genial para modelos de baja potencia [16].

LA Raspberry Pi modelo A está diseñada específicamente para funcionar como equipo pequeño y portátil que no necesita acceso a internet. Las características de este modelo son las siguientes:

- **256 MB de RAM:** La pequeña cantidad de memoria RAM significa que este modelo no es tan potente como los otros, pero también significa que consume menos energía.

- **Broadcom BCM2835 SoC a 700 MHz:** El procesador fue diseñado originalmente para los teléfonos móviles, pero es lo suficientemente potente como para ejecutar un completo software de escritorio.

- Doble núcleo multimedia **VideoCore IV coprocesador:** Este procesador gráfico es el mismo para los 4 modelos. Está integrado en el SoC, pero es un pieza separada del hardware y permite a la Pi ejecutar muchos tipos de juegos y aplicaciones.

- **Ranura microSD:** Esto es para la tarjeta microSD. El Raspberry Pi no tiene un disco duro. En su lugar, se instalan los sistemas operativos en una tarjeta SD.

- **5 voltios** con ranura **microUSB:** Aquí es por donde se alimenta la Pi.

- **Conector USB**: Solo se puede conectar un dispositivo USB a la Raspberry Pi modelo A+. Esto significa que si deseamos utilizar un teclado y un ratón a la vez, necesitaremos un conector USB externo.

- **Puerto HDMI:** Salida de vídeo a través del puerto HDMI. La salida HDMI le da una imagen más clara de alta definición y funciona mejor con modernos monitores o televisores.

- **3.5 mm conector de audio:** Este conector de audio-plus-vídeo hace una doble función. Para el audio, se necesita un cable de audio de 3.5 mm, pero si se desea enviar vídeo compuesto, también necesitaremos un cable adaptador de 3.5 mm a RCA-3.

- **GPIO** (entrada de uso general/salida): Conector GPIO de 40 pines que le permite ser compatible con los otros modelos.

- **Conector de la cámara:** Esto le permite conectar la cámara oficial de Raspberry Pi.

- **Conector de pantalla:** El conector de la pantalla hace posible utilizar una pantalla en vez de usar el HDMI.

El modelo A+ no tiene un conector Ethernet propio, lo que significa que no puede conectarse a internet o redes de forma nativa. Podemos solventar el problema utilizando un adaptador wifi a través del conector USB.

2.11.2 El modelo B

El Raspberry Pi modelo B gasta un poco más de energía que el modelo A+ ya que proporciona más memoria RAM y más puertos [16].

- **512 MB de RAM:** El modelo B tiene el doble de memoria RAM que el modelo A+, por lo que puede ejecutar software más pesado. También significa que es mucho mejor ejecutando vídeo, incluyendo la codificación y emisión en HD.

- **Broadcom BCM2835 SoC a 700 MHz.**

- Doble núcleo multimedia **VideoCore IV coprocesador.**
- **SD, MMC, ranura para tarjeta SDIO:** El modelo B es el único modelo con una ranura para tarjetas SD de tamaño estándar.
- **5 voltios** con ranura micro USB: Aquí es por donde se alimenta la Pi.
- **Dos conectores USB:** Podemos conectar dos dispositivos USB diferentes.
- **10/100 Ethernet RJ45 jack:** Este conector le permite conectarse a Ethernet.
- **HDMI y salida RCA:** Son las mismas que en el modelo A+.
- **3.5 conector de audio:** El modelo B tiene una salida de audio tipo Jack de 3.5 mm.
- **GPIO** (conector de entradas/salidas): El modelo B tiene un conector de 26 pines.
- **Conector de cámara:** Este es el mismo para los cuatro modelos.
- **Conector de la pantalla:** Este es el mismo para los cuatro modelos.

2.11.3 El modelo B+

La Raspberry modelo B+ marca el primer gran avance en la línea de hardware. A diferencia del salto de las primeras Raspberry al modelo B, el modelo B+ cambia la arquitectura general y añade importantes mejoras. Con la excepción de una nueva ranura para la tarjeta SD, todas las mejoras en el modelo B+ se presentan en forma de nuevos puertos. Las especificaciones son básicamente las mismas que el modelo B [17].

- **512 MB de RAM:** El modelo B+ tiene la misma capacidad de memoria que el modelo B.

- **Broadcom BCM2835 SoC con 700 MHz:** El mismo procesador que los modelos anteriores.

- Doble núclo multimedia **VideoCore IV coprocesador:** El mismo que los modelos anteriores.

- **Ranura de tarjeta SDIO:** Ranura para la microSD.

- **5 voltios** con ranura microUSB: Aquí es por donde se alimenta la Pi.

El modelo B+ difiere de los modelos B y A+ en que tiene más conectores USB. Tanto el A+ y el B+ tienen más pines GPIO.

- **Cuatro conectores USB:** El modelo B+ duplica los conectores USB del modelo B; tiene un total de cuatro. Esto hace que sea mucho más fácil de usar como un ordenador con un teclado, ratón, adaptador wifi y cualquier otra cosa que necesitemos conectar.

- **10/100 Ethernet Rj45:** Es el mismo que el del modelo B.

- **HDMI:** El puerto es el mismo que el del modelo B.

- **3.5 mm conector de audio:** El modelo B tiene una salida de audio idéntica al modelo A+.

- **GPIO:** El modelo B+ y el modelo A+ tiene un conector de 40 pines.

- **Conector de la cámara:** Este es el mismo para los cuatro modelos.

- **Conector para la pantalla**: Este es el mismo para los cuatro modelos.

2.12 El hardware de la Raspberry Pi2

La nueva Raspberry es aproximadamente 6 veces más rápida que los modelos anteriores en las tareas más frecuentes. Atrás quedan también los modelos con 256 y 512 MB, ya que ahora la memoria RAM es de 1 GB. Como detalle curioso apuntar que la caja es más grande de todas las Raspberry Pi. Cuenta con un manual de inicio rápido y uso seguro en muchos idiomas. La placa base Raspberry Pi2 modelo B, es del tamaño de una tarjeta de crédito, cuenta con un chip Broadcom BCM2836 con cuatro núcleos a una frecuencia de 900 MHz con arquitectura de 32 bits, en lugar del chip de un solo núcleo de 700 MHz de las versiones anteriores. Sus características más relevantes en comparación con su modelo anterior se muestran en la Tabla 2.1 [17].

Tabla 2.1. Comparación de características técnicas de la Raspberry Pi2 con su modelo anterior.

	Raspberry Pi modelo B+	**Raspberry Pi 2 modelo B**
CPU	ARM11 ARMv6 700 MHz	ARM11 **ARMv7** ARM Cortex-A7 **4 núcleos** @ 900MHz
SoC	Broadcom BCM2835	Broadcom **BCM2836**
Overclocking	Si, Hasta 1000 MHz	Si, Hasta 1000 MHz
GPU	Broadcom VideoCore IV 250 MHz, OpenGL ES 2.0	Broadcom VideoCore IV 250 MHz, OpenGL ES 2.0
RAM	512 MB LPDDR SDRAM 400 MHz	**1 GB** LPDDR2 SDRAM 450 MHz
USB	4	4
Salidas de Vídeo	HDMI 1.4 @ 1920 × 1200 píxeles	HDMI 1.4 @ 1920 × 1200 píxeles
Almacenamiento	microSD	microSD
Ethernet	Si, 10/100 Mbps	Si, 10/100 Mbps
Consumo	5 v, 600 mA	5 v, **900 mA**, aunque depende de la carga de trabajo de los 4 cores

Como se puede comprobar en la tabla anterior y salvo algunas características tales como la GPU, conectividad Ethernet, tamaño, peso y precio en las que empatan, la Raspberry

Pi2 mejora bastante a su predecesora en cuanto a CPU, velocidad y cantidad de memoria. El nuevo procesador no es que sea ya más potente gracias a los cuatro núcleos, sino que abre las puertas a otro tipo de distribuciones y sistemas operativos

(Ubuntu Mate, Windows 10) e incluso a extensiones virtualizadas que basan su programación en las instrucciones más potentes del ARMv7 [18].

La memoria, mayor en cantidad y en velocidad, ayuda a dar ese empujón que necesitaba la Pi2 para proyectos más complejos. La diferencia de precio entre los modelos es insignificante si atendemos a las mejoras de la Pi2, por lo que es realmente aconsejable adquirir esta última.

La nueva Raspberry Pi2 ofrece mejores prestaciones, sí, pero no ha renunciado a ser una placa de desarrollo barata y orientada a la educación.

Antes de conectar nada, es recomendable inspeccionar visualmente la paca. También debemos familiarizarnos con las diferentes conexiones. En la siguiente figura se muestra el hardware de la Pi2 [19].

CAPITULO III

METODOLOGIA

3.1 Investigación documental.

En esta parte del proyecto se realizó el estudio sobre cómo manejar el sistema operativo Linux, características y sintaxis para programar en el lenguaje Python y HTML5, protocolos de comunicación TCP/IP y la teoría sobre el funcionamiento de emplear hilos para conectar más de un cliente, ya que estos temas son necesarios para cubrir y cumplir con los objetivos del proyecto. Así como la investigación de los distintos modelos de tarjetas Raspberry Pi para seleccionar la que mejor se adapte a las necesidades.

3.2 Selección de modelo de Raspberry Pi.

Se realizó un estudio preliminar de comparación de características técnicas entre las tarjetas Raspberry Pi modelo B+ y la Raspberry Pi2 modelo B, para evaluar características de funcionalidad y seleccionar la más adecuada para la implementación del servidor.

3.3 Diseño del servidor.

Se realizó el diseño del servidor, implementando el lenguaje de programación Python sobre el sistema operativo Linux.

3.4 Diseño del cliente.

Se diseñó un cliente sobre la misma tarjeta Raspberry Pi, para simular un sensor y mandar datos al servidor.

3.5 Pruebas de conexión.

En esta etapa los programas del Cliente y Servidor, son ejecutados para comprobar el correcto funcionamiento y conexión de ambos programas.

3.6 Implementar hilos en el servidor

Cuando la conexión de un Cliente con el servidor se realizó, se continuó a implementar el método de comunicación por hilos para poder conectar más clientes a la vez.

3.7 Pruebas de conexión de varios clientes

Después de implementar el método de hilos en el programa del servidor, se iniciaron las pruebas de conexión, ejecutando el programa del cliente varias veces al mismo tiempo en diferentes terminales de Linux y enviando datos de cada cliente al servidor diseñado. Una vez que el programa del servidor puede mantener varios clientes conectados a la vez y recibir los datos que cada uno envía se procede al desarrollo de la página web.

3.8 Desarrollo de página web.

Se desarrolló la página web que sirve como interfaz para mostrar la graficacion de datos obtenidos a partir de los sensores.

3.9 Pruebas de campo.

Para realizar las pruebas de campo el programa del cliente ya no es utilizado puesto que este solo simula un sensor para probar el funcionamiento del servidor en la etapa prueba. En esta etapa el programa del cliente es sustituido por el sensor físicamente y se probó la comunicación entre el servidor montado en la tarjeta Raspberry Pi y los módulos de sensores que miden CO_2 y CO como clientes.

3.10 Presentación de resultados.

Los datos obtenidos a partir de los sensores serán almacenados en el servidor para crear el historial y posteriormente graficarlos en la página web.

Los resultados se presentan en el capítulo IV, de este trabajo de titulación.

3.11 Desarrollo del servidor.

Esta es la etapa del proyecto donde se comienza a diseñar el programa para el servidor en la tarjeta Raspberry pi, para evitar conectar accesorios como el teclado, mouse y el monitor, se realizó el acceso a la tarjeta por medio del software Bitvise SSH Client, que es un sencillo y completo software cliente-servidor SSH.

El protocolo SSH es utilizado para conectarnos de forma remota y segura a ordenadores de una red. Gracias a este protocolo podemos conectarnos, controlar un ordenador y transferir archivos de forma segura y cifrada evitando que terceras personas puedan capturar y analizar el tráfico que generamos. Para poder establecer estas comunicaciones seguras se necesitan un cliente y un servidor (Figura 3.1).

Figura 3.1. *Bitvise SSH Client*

Una vez que el diseño del servidor se ha terminado, este se ejecutara en la terminal (Shell) de acceso remoto (Figura 3.2), al inicio de la ejecución pedirá el puerto por el cual los datos serán recibidos y se pondrá en espera de la conexión de un cliente.

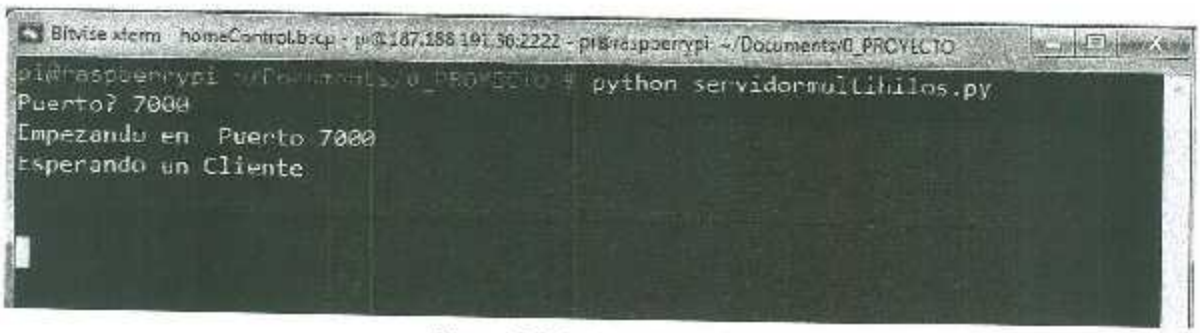


Figura 3.2. *Inicio del Servidor*

Para realizar pruebas de conexión y ajustes del servidor, se hace el uso de un software adicional llamado Hercules, este software crea un cliente y envía datos al servidor simulando datos que serán enviados a partir del sistema de monitoreo de variables ambientales que complementa este proyecto (Figura 3.3)

Figura 3.3. *Terminal Hercules como Cliente*

Cuando el cliente se conecta al servidor un hilo es creado para este cliente e indica su dirección IP y por el puerto que se está realizando la comunicación, el servidor seguirá esperando otro cliente en el hilo principal del programa y recibiendo datos, estos datos son obtenidos con la fecha y hora exacta de la medición (Figura 3.4).

Figura 3.4. *Conexión de un cliente*

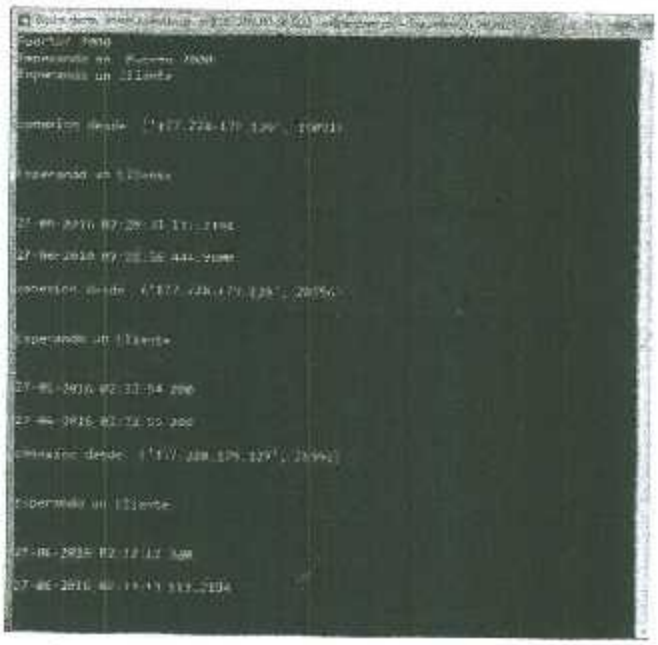
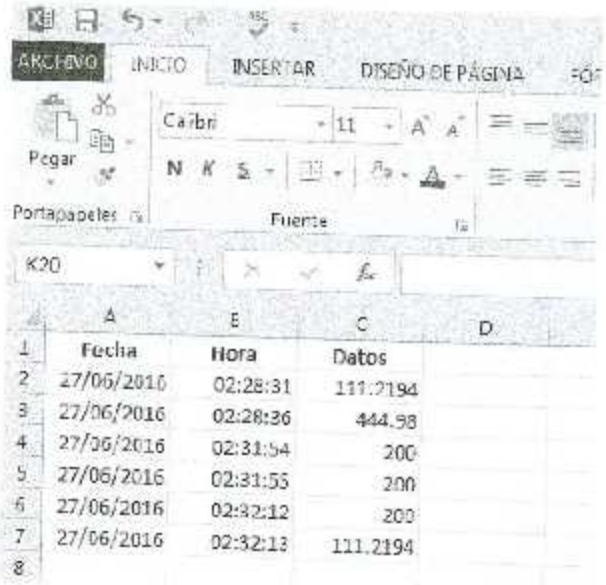


Figura 3.5. *Conexión de varios clientes al servidor*

Cada vez que un cliente se conecte al servidor se obtendrá su dirección IP y el puerto de conexión correspondiente, estos clientes podrán conectarse al mismo tiempo e enviar información puesto que cada uno se está ejecutando en un hilo diferente (Figura 3.5).

Los datos que son recibidos en el servidor son almacenados en un archivo csv (comma-separated values), esté es un tipo de documento en formato abierto para representar datos en forma de tabla y así ser analizados posteriormente (Figura 3.6).



ARCHIVO | INICIO | INSERTAR | DISEÑO DE PÁGINA

Pegar | Portapapeles | Calibri | 11 | Fuente

K20

	A	B	C	D
1	Fecha	Hora	Datos	
2	27/06/2016	02:28:31	111.2194	
3	27/06/2016	02:28:36	444.98	
4	27/06/2016	02:31:54	200	
5	27/06/2016	02:31:55	200	
6	27/06/2016	02:32:12	200	
7	27/06/2016	02:32:13	111.2194	
8				

Figura 3.6. *Datos obtenidos a partir de clientes*

3.12 Desarrollo de página web como interfaz.

Para el desarrollo de la página web donde se visualizarán los datos se empleó la programación en HTML5, ya que nos proporcionará una manera de hacer un código más limpio, más fácil de leer y escribir y la utilización de etiquetas (tags), hojas de estilo en cascada (CSS) y Java Script.

Etiquetas (tags)

El código HTML consiste de un archivo de texto delimitado por etiquetas o "tags" (palabra usada en inglés), dentro de estas etiquetas se coloca diversa información, como por ejemplo, qué texto se debería visualizar, en qué lugar deben aparecer los elementos, que imágenes se van a utilizar, entre muchas otras cosas.

Para entender el porqué del uso de este sistema de etiquetado se tendría que remontar a los días donde no existían mensajeros instantáneos y todos utilizaban listas de correo

con conexiones por módem dial-up, y donde gente bastante inteligente, utilizando estos recursos, se comunicaba para determinar qué etiqueta serviría para qué cosa y de qué manera debería escribirse.

Un ejemplo de etiqueta es, por ejemplo, la etiqueta <h2>, la cual sirve para darle a un texto características de título o cabecera. Utilizando esta etiqueta, el código HTML sería:

```
<h2>Hola, Bienvenido</h2>
```

Si se observa, el código se encontrará con los paréntesis angulares < > (los símbolos "menor que" y "mayor que", que enmarcan la etiqueta de apertura o inicio, y </ >, que acotan la etiqueta de cierre o final), éste es el caso para etiquetas que requieren apertura y cierre.

La etiqueta <img> permite colocar imágenes en sus páginas, el uso de esta etiqueta se vería así:

```
<img src="pelota.jpg" />
```

Esta etiqueta no utiliza cierre, es por eso que se le llama simple o vacía, e introduce una nueva característica en el uso de etiquetas, llamado atributo. Los atributos en una etiqueta indican propiedades o comportamientos que la etiqueta en cuestión debe tener, en este ejemplo se utilizó la propiedad src, la cual indica que la imagen que debe desplegarse corresponde al archivo pelota.jpg y no a cualquier otro.

Los atributos pueden ser opcionales u obligatorios, dependiendo de las especificaciones dadas a cada etiqueta.

Se utilizan etiquetas para dar formato a un texto, colocar un enlace, una imagen o algún otro tipo de elemento, el navegador lee e interpreta estas etiquetas y las coloca en una

forma "traducida" y visualmente amable para el usuario. Esto es básicamente el funcionamiento de las cosas.

Hojas de estilo.

CSS (Cascading Style Sheets: Hojas de Estilo en Cascada), es un lenguaje que no es propiamente parte del HTML en ninguna versión definida por la W3C, su objetivo es definir la presentación de un documento escrito en HTML o XML (y claro en XHTML) pero no es parte del código HTML del documento en sí.

Usualmente en un archivo separado, el programador usa CSS para especificar el formato para un tipo de etiqueta o para una etiqueta en particular. La idea principal detrás de todo esto es separar el código de estructura del código de presentación, de manera que cuando se modifica el archivo con código CSS, éste inmediatamente afecta a todas las páginas HTML que hacen referencia a él, evitando así editar cada página HTML de forma individual.

Java Script

JavaScript es otro lenguaje de programación que tiene una estrecha relación con HTML, JavaScript es esencial para HTML5, varias de las características más importantes de HTML5 requieren de JavaScript, como son la nueva etiqueta <canvas>, la geolocalización, el almacenamiento local de datos y otras.

En la Figura 3.7 se muestra una parte del código que se utilizó para crear la página web.

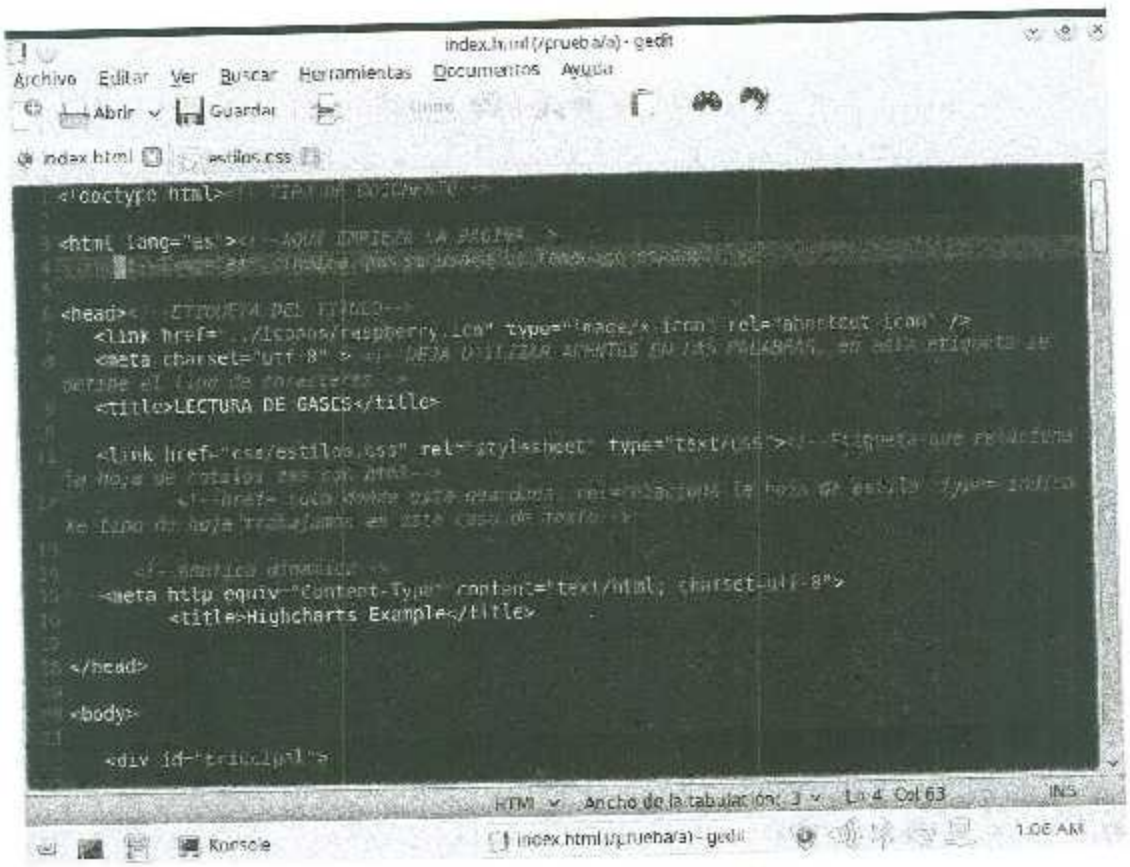


Figura 3.7. Algoritmo de página web.

CAPITULO IV

RESULTADOS.

4.1 Investigación documental

La recopilación de investigación documental ayudo a crear el marco teórico, obtener las herramientas y el conocimiento necesario para el desarrollo del proyecto.

4.2 Selección de modelo de Raspberry Pi.

De acuerdo a la comparación que se realizó de los dos modelos "Raspberry pi2 y Raspberry pi", la Raspberry pi2 cuenta con las características necesarias para configurar el servidor y minimizar fallas por variables como la temperatura o saturación de datos.

4.3 Diseño del servidor.

La flexibilidad a diferentes lenguajes de programación de la Raspberry pi2 permitió la implementación de la programación por hilos en Python, para identificar y comunicar los nodos de la red de sensores, ya que antes esto requería de un código robusto y complejo con los lenguajes de programación más utilizados (C, C#, etc.).

4.4 Implementar hilos en el servidor.

Se logró enlazar a diversos clientes en un solo servidor, ya que se comparte la memoria y los recursos del proceso al tener varios hilos de ejecución, mejorando el tiempo de respuesta, ya que el programa puede continuar ejecutándose, aunque parte de él esté bloqueado debido a que se encuentra atendiendo un cliente.

4.5 Desarrollo de página web.

Los sitios Web pueden ser visualizados o accedidos desde un amplio abanico de dispositivos con conexión a Internet, como computadoras personales, portátiles, PDA´s y Teléfonos móviles.

4.6 Pruebas de campo.

Un punto importante dentro del diseño de la página web, fue generar la gráfica con los datos recibidos de los sensores, esto implica generar un archivo tipo .csv, en el cual los datos fueron guardados para ser leídos y graficados, además este tipo de archivo es compatible con otros programas por ejemplo Excel, La información obtenida se divide en Fecha, hora y dato.

En la Figura 4.1, se muestra la interfaz del servidor desarrollada como página web, para mostrar los datos que se almacenaron en el servidor implementado en la tarjeta Raspberry pi2, de los sensores MQ-135.

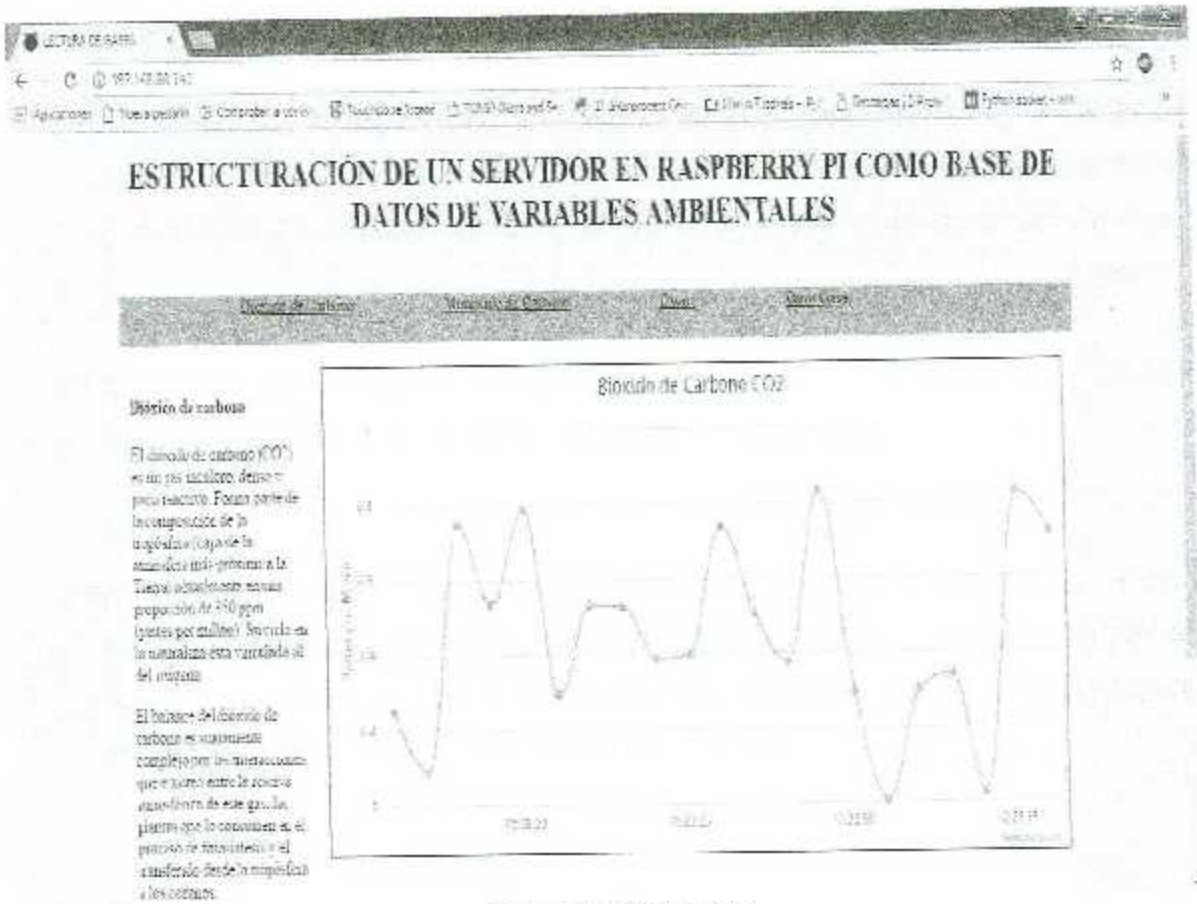


Figura 4.1. Página web

CONCLUSIONES Y RECOMENDACIONES

- Se cumplió con el objetivo de estructurar el servidor en Raspberry Pi como base de datos de variables ambientales.

Se cumplió con los objetivos específicos siguientes:

- Se implementó la programación en lenguaje Python para la adecuación de la tarjeta como un servidor, mediante el protocolo de comunicación TCP/IP desarrollado sobre el sistema operativo Linux.

- Se Diseñó e implementación de página web para visualizar los datos obtenidos.

- Se cumplió con el objetivo de la creación de la página web, como interfaz del proyecto para el despliegue de la información de los nodos, la cual permite tener o consultar el comportamiento de la red en la mayoría de las plataformas de internet, ya sea desde una computadora personal o mediante un Smartphone.

- Almacenar datos históricos de las mediciones de concentración de CO_2 y CO en puntos estratégicos del Instituto Tecnológico de la Laguna.

- En versiones anteriores se utilizaron protocolos de comunicación XBee. Estos módulos presentan la ventaja de ser fáciles en su configuración básica en comunicación punto a punto. Sin embargo; al utilizarlos como una red de sensores presentan una mayor complicación en su configuración,

- Se buscó una manera de manejar o procesar la información recibida por un número indefinido de nodos de la red, además de conseguir una mejor robustez, respecto al proyecto anterior.

- La programación por hilos da una forma sencilla eficaz y menos robusta al manejar paquetes de información proveniente de varios clientes (nodos).

- En una etapa posterior, se podría buscar diseñar o generar graficas mostrando el comportamiento de cada nodo sensor, comparando dicho comportamiento para resaltar el área donde hay mayor concentración de gas.

RECOMENDACIONES

- Que la Raspberry pi tenga acceso las 24 horas a la red, para evitar la pérdida de datos.

- La depuración mensual o quincenal de los archivos creados en la Raspberry PI2, para tener un trabajo óptimo de la misma.

FUENTES DE INFORMACION

[1] PROYECTO DE INVESTIGACIÓN: SENSOR DE EVAPORACIÓN ULTRASÓNICO APLICADO.
Autor: Ángel Alberto Cabrera Lara, Torreón Coahuila. Mayo 2013

[2] ARQUITECTURA DEL PROTOCOLO TCP/IP.
Protocolo TCP/IP,
https://arquitecturaprotocolos.wikispaces.com/MODELO+TCP+IP (Consultada el 25/ Agosto/ 2016)

[3] INFORMATION GENERAL TCP/IP. AT&T.
https://docs.oracle.com/cd/E19957-01/820-2981/ipov-10/ (Consultada el 6/sep/2016)

[4] EL PROTOCOLO TCP/IP, REDES, INGENIERA DE SISTEMAS Y AUTOMÁTICA, Universidad de Oviedo.
https://www.yumpu.com/es/document/view/23521400/el-protocolo-tcp-ip-area-de-ingenieria-de-sistemas-y-automatica-/3 (Consultada el (6/sep/2016))

[5] MODELO OSI,
Autor: Ing. William Marin Moreno.

[6] INTRODUCCIÓN A LA CONFIGURACIÓN DE ROUTERS CISCO, FACULTAD DE INGENIERA, UNIVERSIDAD ORT DE URUGUA.
http://www.ort.edu.uy/fi/pdf/configuracionroutersciscomatturro.pdf (Consultada el 22 / sep /2016)

[7] CURSO DE ADMINISTRADORES DE SERVIDORES /EXTRANET /INTRANET, REDES TCP/IP, LOS PROTOCOLOS TCP/IP.
http://www.juntadeandalucia.es/empleo/recursos/material_didactico/especialidades/materialdidactico_administrador_servidores/mod3.html (Consultada el 29 /sep/ 2016)

[8] BEGINNIG PYTHON FROM NOVICE TO PROFESSIONALS.
Autor: Magnus Lie Hetland. Apress.

[9] LINUX, MANUAL DE REFERENCIA,
Sexta edición, Autor: Richard Petersen, ED: McGRAW-HILL, ISBN: 978-0-07-149247-8.

[10] LINUX DESDE CERO, REDUSER,
Autor: Franco Rivero.

[11] APRENDER A PENSAR COMO UN PROGRAMADOR CON PYTHON,
Autor: Allen Downey-Jeffrey Elker-Chris Meyers. Green Tea Press

[12] INTRODUCCION A LA PROGRAMACION CON PYTHON.
Autor: Andres Marzal- Isabel Garcia.Universidad Juame.

[13] GUIA DE APRENDIZAJE DE PYTHON.
Autor: Guido Van Rossum-Fred L. Draker Jr.Python Software Foundation (docs@python.org).

[14] ARRANCAR CON HTML5, CURSO DE PROGRAMACIÓN.
Autor: Herrera Ríos Emmanuel,
Primera Edición, Alfaomega Grupo Editor, S.A. de C.V. México, ISBN: 978-607-707-331-4, Formato: 17 x 23 cm Páginas: 264

[15] HTML DINÁMICO, MODELOS DE OBJETOS Y JAVASCRIPT,
Autor: Marino Posadas y José Luis, Hevia ISBN 84-88457-22-7

[16] TUTORIAL ON NETWORK PROGRAMMING WITH PYTHON,
Autor: Norman Matloff, University of Califomia, Davis C 2003-2009, N. Matloff May 3, 2009.

[17] THINK PYTHON, HOW TO THINK LIKE A COMPUTER SCIENTIST,
Version 2.0.17, Autor: Allen Downey, Green Tea Press, Needham, Massachusetts.

[18] ART. A STUDY ON THE POSSIBILITY TO USE RASPBERRY PI AS A CONSOLE SERVER FOR REMOTE ACCESS TO DEVICES IN VIRTUAL LEARNING ENVIRONMENTS
Autor: Diyana Kyuchukova. Georgi Hristov, Plamen Zahariev and Svilen Borisov Department of Telecommunications University of Ruse "Angel Kanchev" Ruse, Bulgaria, dkyuchukova@uni-ruse.bg, ghristov@uni-ruse.bg. pzahariev@uni-ruse.bg, ODPleven@mail.bg

[19] "CHALLENGES AND SOLUTIONS FOR REAL-TIME REMOTE CONSOLE ACCESS TO TELECOMMUNICATION EQUIPMENT",
Autor: G. Hristov, P. Zahariev, D. Kyuchukova, International conference on ITHET, York, UK, 2014.