ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Diseño e implementación de un prototipo de sistema electrónico para monitoreo y dosificación de medicamentos a adultos mayores"

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Electrónica y Telecomunicaciones

Presentado por:

Isaac Walter Guamán Torres

José Alejandro Ricaurte Pino

GUAYAQUIL - ECUADOR

Año: 2019

DEDICATORIA

Este proyecto está dedicado a mis padres, Antonio y Lorena, a mis hermanas, y a Martín, quien, aunque ya no está con nosotros, su recuerdo perdurará siempre. Mi padre siempre fue un gran maestro para mí, desde el principio de la carrera hasta el proyecto de graduación, y mi madre siempre apoyándome y dándome ánimos para cumplir mis metas.

También para las amistades de por vida que me dio la universidad, ya que gracias a ustedes la vida universitaria fue mucho mejor.

José Ricaurte Pino

El presente proyecto lo dedico a Dios, a la virgen María, por darme la sabiduría y la fortaleza para culminar esta etapa estudiantil.

A mis amados padres Margarita y Walter que, con su apoyo, cariño, esfuerzo y ejemplo, me motivan y me brindan las herramientas necesarias para seguir adelante y nunca rendirme.

A mis hermanos Ginger, Solange y Jacob, que esto sirva como un ejemplo de superación y perseverancia, para que tengan siempre en mente que las metas pueden ser alcanzadas con esfuerzo y dedicación.

A mi novia Karla, que con su cariño y su apoyo incondicional siempre me impulsa a seguir adelante y perseguir mis sueños.

Isaac Walter Guamán Torres

AGRADECIMIENTOS

Mi más sincero agradecimiento a mis maestros, que además de impartir conocimientos y compartir sus experiencias se preocuparon por brindarme una formación donde prevalece la honestidad, la ética y el profesionalismo.

De manera muy especial agradezco a la Ing. Marcia Garcés Mendoza y al Ing. Geovanny Alvarado Villa, quienes me brindaron su amistad y sabiduría, ya que con su ejemplo me enseñaron el valor del trabajo, el esfuerzo y la dedicación, los cuales influyeron en mí y en las futuras metas que deseo alcanzar como profesional.

A mi tutora de tesis, Ing. María de los Ángeles Santacruz, a mi maestro de la materia integradora, Ing. Francisco Novillo, y al Ing. César Yépez por su valiosas guías y asesoramientos en las diversas etapas del desarrollo del proyecto.

A mi amigo y compañero de tesis, José Ricaurte por su dedicación y compromiso en el desarrollo y culminación del proyecto integrador.

Isaac Walter Guamán Torres

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, me(nos) corresponde conforme al reglamento de propiedad intelectual de la institución; *Isaac Walter Guamán Torres* y *José Alejandro Ricaurte Pino* damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

José Ricaurte Pino

Isaac Guamán Torres

EVALUADORES

Ing. Francisco Novillo

PROFESOR DE LA MATERIA

Ing. María Angélica Santacruz

PROFESOR TUTOR

RESUMEN

El presente proyecto se desarrolló con la finalidad de realizar un aporte a la sociedad. Se tomó como lugar de aplicación el asilo “Hogar San Jose”. En este se hizo una investigación de los problemas existentes, los cuales eran la dosificación, control y administración de los medicamentos a los ancianos. Analizando esto, se llegó a la conclusión de que era necesario implementar un sistema que ayudara a resolver estos problemas, decidiendo así, crear un brazalete electrónico poco invasivo que pueda notificar a los cuidadores, y una base de datos para llevar control.

Para la realización del brazalete, el cual funciona a la par de una interfaz web, se utilizaron diferentes componentes tales como: microcontroladores para poder controlar los circuitos; un Arduino para poder programar los microcontroladores; una Raspberry que funcionó como servidor y un Router inalámbrico TP-LINK para la comunicación entre el dispositivo móvil(brazalete), y el servidor.

Al tener el prototipo en funcionamiento, se pudo apreciar que luego de pruebas que duraron varios días, el dispositivo funcionó de manera autónoma y sin ningún problema; así también se pudo comprobar que las actualizaciones remotas se realizaron de manera eficaz, y que la batería del dispositivo duraba al menos 10 horas ininterrumpidas.

Teniendo los puntos anteriores en cuenta, se puede concluir que el dispositivo notifica en tiempo real cuando es el momento de administrar un medicamento. Así mismo, la interfaz web sirve de gran ayuda a los cuidadores y a los médicos que realizan los chequeos, para que estos lleven una bitácora digital. También, y como punto más importante, se redujo significativamente los errores de administración de medicamentos, y el tiempo invertido en administrar los mismos.

Palabras clave: brazalete, interfaz, ancianos, médicos, microcontrolador.

ABSTRACT

This project was develop with the purpose of contributing to the society. The "Hogar San Jose" asylum was the application place. Here we made an investigation of the existing problems which were the dosage, control and administration of the medications to the elderly. Analyzing this, it was concluded that it was necessary to implement a system that would help solve these problems, thus deciding to create a non-invasive electronic bracelet that can notify caregivers, and a database to control.

For the realization of the bracelet, which works alongside a web interface, different components were use such as microcontrollers to control the circuits; an Arduino to be able to program the microcontrollers; a Raspberry that worked as a server and a TP-LINK wireless router for communication between the mobile device (bracelet), and the server.

Having an operative prototype, it could be seen that after tests that lasted several days, the device worked autonomously and without any problems, in which it was possible to verify that the remote updates were carried out efficiently, and that the battery of the device lasted at least 10 uninterrupted hours.

We can conclude that the device is helpful since it notifies in real time when it is time to administer a medication. Likewise, the web interface is of great help not only to caregivers, but also to doctors who perform weekly check-ups for the elderly, so that they carry a digital log. Also, and as the most important point, medication administration errors were significantly reduce, also as the time spent administering them.

Keywords: *bracelet, interface, elderly, doctors, microcontroller.*

ÍNDICE GENERAL

ABREVIATURAS

MIES	Ministerio de Inclusión Económica y Social
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
PHP	Hypertext Preprocessor
SBC	Single Board Computers
GPL	General Public License
WLAN	Wireless Local Area Network
I2C	Inter-Integrated Circuit
RTC	Real Time Clock
PCB	Printed Circuit Board
LAN	Local Area Network
ROM	Read only memory
SMD	Surface-Mount Device

SIMBOLOGÍA

A	Amperio
m	mili
V	Voltio
DC	Direct Current

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

A escala mundial, cada segundo dos personas cumplen 60 años y al momento existen 810 millones de personas mayores a esa edad. Según datos del Ministerio de Inclusión Económica y Social (MIES) en el Ecuador existen 1.049.824 personas mayores a 65 años, representando el 6.5% de la población total, de los cuales el 45% se encuentra en condiciones de pobreza y extrema pobreza por necesidades básicas insatisfechas y el 14.9% son víctimas de negligencia y abandono [1].

Los albergues o también denominados asilos de ancianos, han permitido que muchos ecuatorianos adultos mayores que se encuentran en situación de vulnerabilidad y/o sin vínculo familiar puedan acceder a un espacio para dormir, alimentarse y recibir los cuidados y atenciones necesarias para vivir dignamente, los cuales están amparados en la Constitución y Tratados Internacionales de Derechos Humanos [2].

Según los artículos. 35, 36, 37 y 38 de la Constitución de la República del Ecuador, el adulto mayor pertenece al grupo de personas de atención prioritaria por lo que recibirán atención especializada en los ámbitos público y privado en especial en los campos de inclusión social y económica, y protección contra la violencia, además el Estado proporcionará la atención en centros especializados que garanticen su nutrición, salud, educación y cuidado diario, en un marco de protección integral de derechos [3].

El asilo de ancianos “Hogar San José” ubicado en la ciudad de Guayaquil, tiene bajo su protección y atención a 98 adultos mayores, los cuales son cuidados por las monjas de la Congregación de las Hermanitas de los Ancianitos Desamparados, así como por profesionales de la salud, que cumpliendo con la misión de ofrecer un cuidado digno y eficaz realizan consultas médicas semanales

a cada uno de los residentes del asilo, sin embargo, según entrevistas realizadas a la Directora y personal colaborativo se requiere de un sistema que organice y gestione la dosificación de medicamentos a las personas de la tercera edad por parte de los médicos y cuidadores con la finalidad de asegurar la toma de los mismos en los horarios y dosis asignadas por el médico, ya que, al no disponerlo se presentan algunos inconvenientes entre los que podemos destacar: la alta confusión, inversión de tiempo excesivo y cronograma inadecuado en la administración de medicinas; así como la poca eficiencia en el control para la administración de medicamentos. Un análisis más detallado de estas y otras problemáticas se expone más adelante en este documento.

1.1 Descripción del problema

Basado en entrevistas realizadas a la directora: Sor. Elsa Rodríguez, del “Hogar San José”, al personal asistencial que labora en el asilo y al recorrido por las instalaciones del lugar se encontraron los siguientes problemas:

- Falta de recursos a la hora de agendar la administración de medicamentos.
- Cronograma inadecuado para la administración de medicamentos.
- Muchos adultos mayores no se enteran de la hora de comer debido a su dispersión en todo el lugar.
- Alta confusión en la administración de medicinas a los adultos mayores por parte de los cuidadores, debido a la gran cantidad de horarios y medicinas.
- Poca eficiencia en el control para la administración de medicamentos.
- Poca disponibilidad de tiempo para el cuidador en el desempeño de todas sus labores y tareas.
- Mucha carga de tareas asignadas a los cuidadores.
- Pocos cuidadores, los cuales tienen edad avanzada.
- Poca predisposición a la toma de medicinas por parte de los residentes.
- Falta de empatía por parte de los cuidadores.
- Mucho tiempo invertido buscando a los pacientes.

La mayoría de los inconvenientes expuestos son en torno a la administración de los medicamentos desde que son recetados por el médico hasta que finalmente son proporcionados a los residentes del asilo, este proceso se detalla a continuación.

El médico de cabecera se acerca al asilo dos veces por semana para atender a los residentes del mismo, los cuales tienen su historial clínico registrado de forma manual en carpetas y hojas con fichas médicas. El médico busca manualmente la carpeta expediente del paciente para verificar la medicación que dispone actualmente y así mantener o modificar la misma de acuerdo al diagnóstico establecido. Una vez que termina la consulta, la persona auxiliar de enfermería procede a buscar manualmente en su listado de medicinas y pacientes para verificar si ha cambiado de alguna manera la medicación para actualizarla. Posterior a este proceso, se da inicio a la administración de medicinas, ya que de igual manera se realiza una búsqueda manual de los horarios, nombres y medicaciones de cada uno de los residentes del asilo. Todo este proceso dispone de tareas redundantes, implicando degaste físico innecesario por parte del personal empleando más tiempo de lo necesario al realizar las búsquedas y agendamientos de forma manual, por otro lado, ya que las etapas entre el médico y el paciente en la administración de medicinas son realizadas por personas, el error humano aumenta ya que en ocasiones existen confusiones en los cronogramas de medicación, llegando a estar ausente alguna medicación a los residentes. Las medicaciones son de uso delicado por lo cual un error o confusión no es permitido en los adultos mayores ya que su consecuencia es la alteración violenta de su estado de salud o en el peor de los casos puede producir la muerte.

1.3 Justificación del problema

El agendamiento y administración de la medicación a cada uno de los pacientes residentes del asilo realizada de forma manual y mediante el uso de carpetas, papeles e incluso la memoria humana, implica que el proceso es más propenso a la presencia de errores debido a diversos factores, entre los cuales se tiene: errores

por parte del médico al visualizar una bitácora de medicación no actualizada por lo que provoca un diagnóstico no adecuado, pérdida o deterioro del historial clínico de medicación de los pacientes, errores debido a la actualización manual de la bitácora de medicación de todos los pacientes por parte del auxiliar de enfermería, omisión del suministro de medicación específica a los pacientes debido a la diversidad de medicaciones, pacientes y horarios de dosis, entre otros.

Con la finalidad de mejorar y reducir los errores producidos en el proceso anteriormente expuesto, el sistema electrónico que se propone se muestra como una solución robusta y eficaz, que goza de autonomía y que permitirá realizar el agendamiento, consulta y modificación de las historias clínicas de cada uno de los pacientes de manera digital por parte del médico, así como la actualización automática de la bitácora de medicación digital y el recordatorio efectivo y a tiempo mediante notificaciones personalizadas y distribuidas a cada uno de los miembros del personal de enfermería.

El respaldo, la actualización y la notificación oportuna de las medicaciones recetadas e ingresadas por parte del médico para cada uno de los pacientes es esencial para la correcta administración de los mismos lo que garantiza el cuidado, la buena salud y la vida plena de cada uno de los residentes en el asilo de ancianos.

La explicación de algunos conceptos básicos acerca del proceso de agendamiento, búsqueda, actualización, administración y notificación digital se realizarán en la siguiente sección.

1.4 Objetivos

1.4.1 Objetivo General

- Desarrollar un sistema electrónico para la dosificación de medicina, mediante el uso de microcontroladores y una interfaz web, para solucionar los problemas existentes al momento de administrar medicina a ancianos en el asilo “Hogar San Jose”.

1.4.2 Objetivos Específicos

- Realizar una revisión de los problemas encontrados.
- Realizar un esquema con la solución posible.
- Hacer el diseño para el prototipo del brazalete y de la interfaz web.
- Realizar un diseño lo mínimamente invasivo y de fácil uso para el médico y cuidadores (personal de enfermería).
- Realizar pruebas rigurosas al sistema.
- Solucionar los problemas puntuales que existen en el asilo “Hogar San José” acorde a la problemática expuesta.

CAPÍTULO 2

2. MARCO TEÓRICO.

2.1 Sistema electrónico para monitoreo y dosificación de medicamentos.

En el presente capítulo se presentan a detalle las herramientas y recursos utilizados para el diseño del sistema electrónico de monitoreo, los cuales son: Ordenador de placa recudida Raspberry, Raspbian como sistema operativo basado en LINUX, Apache como servidor web, MySQL para gestión de la base de datos, PHP como lenguaje de programación en el servidor y Python como lenguaje de programación para la comunicación entre el servidor y la pulsera inteligente.

Se definió el despliegue del sistema en la estación de enfermería y el respectivo personal del asilo de ancianos del “Hogar San José” en la ciudad de Guayaquil.

El diseño se basa en un sistema de monitoreo y notificaciones conformado por un servidor web, una base de datos y dispositivos terminales indicadores, tales como una pulsera inteligente y un monitor, implementados en conjunto sobre una red *Wireless Fidelity (WiFi)* con topología tipo estrella. Se optó por la utilización de un sistema inalámbrico para prevalecer la opción de movilidad de los dispositivos del sistema, así como la facilidad de portabilidad de los mismos a través del perímetro del asilo de ancianos.

El diseño posee características de alta escalabilidad mediante la posibilidad de agregar dispositivos terminales dependiendo de la demanda del personal de enfermería y la capacidad de soportar una gran cantidad de información de los pacientes. Así como la libre implementación ya que se utilizó software y hardware “abierto”. En la Figura 2.1, se presenta un diseño general del sistema electrónico propuesto.

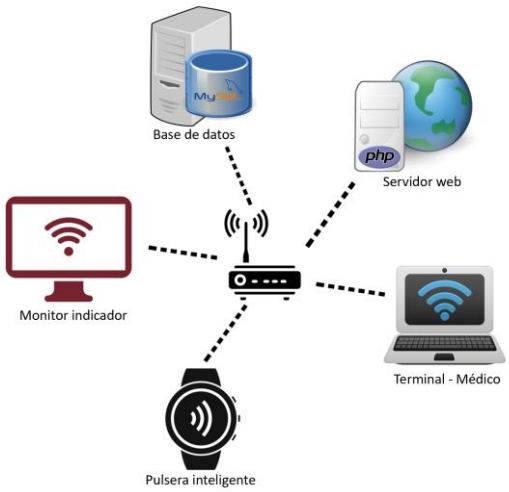


Figura 2.1 Diseño del sistema electrónico para el control de medicación en pacientes geriátricos.

2.1.1 Soluciones existentes en el mercado.

Existe en España una pulsera digital dirigida específicamente al sector oncológico. Esta pulsera muestra un código de barras, que mediante una pistola se puede ver el historial clínico del paciente, así como la medicina que está tomando en esos momentos [4].

2.2 Arquitectura referencia de base de datos para la web.

La base de datos que contiene los horarios, medicinas y pacientes ingresados por el médico se almacenan en el servidor de base de datos MySQL mediante una interfaz de *Lenguaje de Marcas de Hipertexto o HMTL* (viene del inglés, *HyperText Markup Language*) solicitada al servidor web Apache desde el navegador de internet del equipo terminal del médico utilizando una petición del *Protocolo de transferencia de hipertexto o HTTP* (viene del inglés: *Hypertext Transfer Protocol*) a la dirección de *Protocolo de Internet o IP* (viene del inglés, *Internet Protocol*) del servidor el cual mediante el motor del *Preprocesador de Hipertexto o PHP* (viene del inglés, *Hypertext Preprocessor*) realiza la creación,

consulta o modificación necesaria en el mencionado servidor de base de datos. En la Figura 2.2, se presenta la arquitectura utilizada en el desarrollo del sistema.

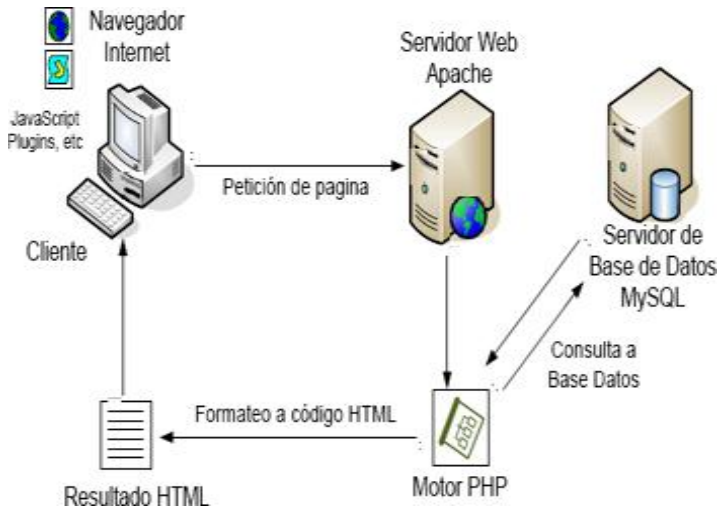


Figura 2.2 Arquitectura de base de datos para la web [5].

2.2.1 Computadoras de Placa Reducida o SBC (viene del inglés, Single Board Computers)

Las *SBC* se han vuelto de uso indispensable para diversos proyectos en los últimos años. Las SBC tienen la característica de ser eficientes, compactas y además disponen de bajo consumo energético. Al ser compactas solo disponen de los elementos más importantes de una computadora con la diferencia de que se encuentran en una sola tarjeta de circuito impreso, a excepción de otros componentes como monitor, disco duro, teclado, mouse, etc., los cuales se conectan a través de sockets a la tarjeta. Entre las más utilizadas se tiene a la Raspberry Pi3 Model B, el cual se describirá más a detalle en el capítulo 3 de este documento [5].

2.2.2 Sistema operativo Raspbian.

Para la implementación de los servidores mencionados anteriormente, se utilizó hardware libre el cual fue la placa microordenador denominado Raspberry Pi Model B, con el sistema operativo libre utilizado denominado Raspbian, el cual permitió la interoperabilidad entre los sistemas servidores y los diferentes elementos de hardware destinados para ser dispositivos indicadores ya que dispone de interfaz WiFi, un procesador Cortex-A53 (ARMv8) a 64 bits, así como la posibilidad de expansión de su *memoria de solo lectura o ROM* (viene del inglés, *read only memory*) mediante microSD, así como su bajo consumo teniendo como máximo 1.8W como una alimentación de 5VDC con 2.1A a pleno uso, tal como se puede apreciar en la tabla 2.1 [6].

Tabla 2.1 - Características Raspberry PI3 Model B+ [6].

Especificaciones	**Raspberry PI3 MODEL B+ (2018)**
Procesador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @1.4GHz
RAM	1GB
Conectividad	WiFi 802.11.b/g/n/ac de doble banda 2.4GHz y 5GHz. Bluetooth 4.2 Puerto Ethernet de hasta 300Mbps
Puertos	HDMI completo, 4 USB 2.0, MicroSD, CSI camera, DSI display.

Raspbian es el sistema intermediario entre el Software servidor Web, base de datos y de gestión de notificaciones utilizado y el hardware disponible en el microordenador Raspberry como su hardware de almacenamiento, procesamiento y Networking.

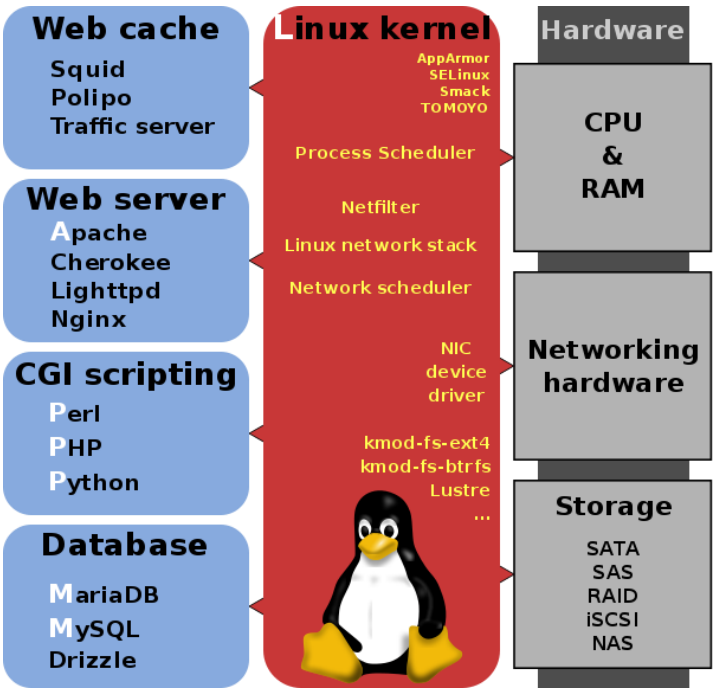


Figura 2.3 Raspbian como intermediario entre Software y Hardware del sistema [7].

2.2.3 Servidor de base de datos MySQL y PHP.

MySQL fue utilizada para la gestión de la base de datos de los pacientes debido a su alto rendimiento, su fiabilidad y a la facilidad de uso e integración con el sistema operativo Raspbian [7]. Al utilizarla con el lenguaje de programación PHP se realizaron aplicaciones cliente/servidor para la creación, edición y consulta de la base de datos con interfaz HTML versión 5 utilizando el protocolo HTTP mediante una transferencia de datos en la red [8].

2.2.4 Servidor Web Apache.

Al ser un software de código abierto y con *Licencia General Pública o GPL* (viene del inglés, *General Public License*), Apache fue utilizado como un componente de servidor Web junto con MySQL y el lenguaje de programación PHP para la base de datos y Python para el monitoreo y

gestión de la información alojada en la misma gracias al soporte proporcionado a diferentes lenguajes de programación [9].

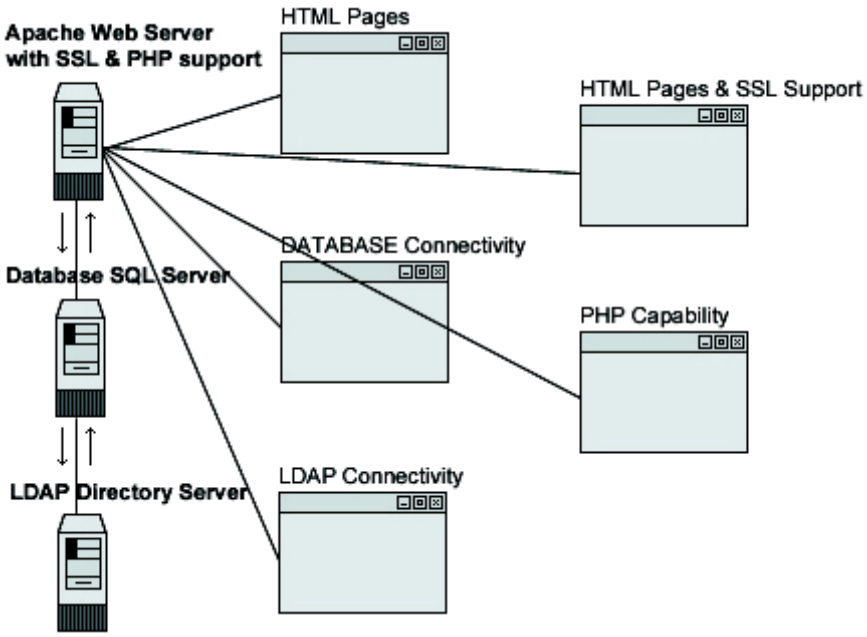


Figura 2.4 Esquema de funcionamiento de Apache [11].

2.3 Estándar 802.11b.

El estándar IEEE 802.11 define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en una *Red de Área Local Extendida o WLAN* (viene del inglés, *Wireless Local Area Network*) o Redes de área local inalámbricas. Hoy en día se usan generalmente sus versiones 802.11a, 802.11b y 802.11g para proporcionar conectividad en los hogares, oficinas y establecimientos comerciales [10].

2.4 Microcontrolador.

Un microcontrolador es un circuito integrado el cual en su interior posee una memoria, procesador y puertos de entrada y salida. Los microcontroladores

están presentes en la mayoría de circuitos electrónicos que están dentro de los aparatos de uso diario (computadoras, automóviles, refrigeradoras, etc.).

Un microcontrolador funciona básicamente como el cerebro de todo circuito electrónico. A este se le guardan las diferentes instrucciones que se desean que ejecute. La forma en que uno se puede comunicar con un microcontrolador, es mediante programación en lenguaje C (o C++) [11].

Para este caso, se hizo uso de microcontrolador ATmega328P de la marca ATMEL.

2.5 Arduino.

Arduino es una tarjeta electrónica la cual básicamente sirve para programar (pasar nuestro código desde la computadora, hacia el microcontrolador) el microcontrolador ATmega328P. Esto presenta una gran ventaja frente a otros microcontroladores ya que la forma de programarlos puede ser muy engorrosa.

2.6 Protocolos de comunicación.

Un protocolo de comunicación, es un método mediante el cual un microcontrolador se comunica con diferentes periféricos, ya sean sensores, pantallas, etc. Existen diferentes tipos de protocolos, y dependiendo del elegido, también variara la cantidad de pines usados. Para este proyecto se usó el protocolo de comunicación a través del *Circuito Inter-Integrado o i2c* (viene del inglés, *Inter-Integrated Circuit*), el cual hace uso de 4 pines: VCC, GND, SDA, SCL. Siendo VCC y GND la alimentación, SDA el pin de entrada y salida de datos, y SCL la señal de reloj mediante la cual va a trabajar.

El protocolo de comunicación serial, es aquel que envía datos un bit a la vez. Para su uso son necesarios 3 pines: TX, RX y GND. Siendo TX la transmisión de datos, RX la recepción de datos, y GND la tierra. En este proyecto se hizo uso de este protocolo para comunicar el módulo WiFi ESP8266 con el

microcontrolador, y de esta manera poder adquirir y procesar los datos receptados de manera remota.

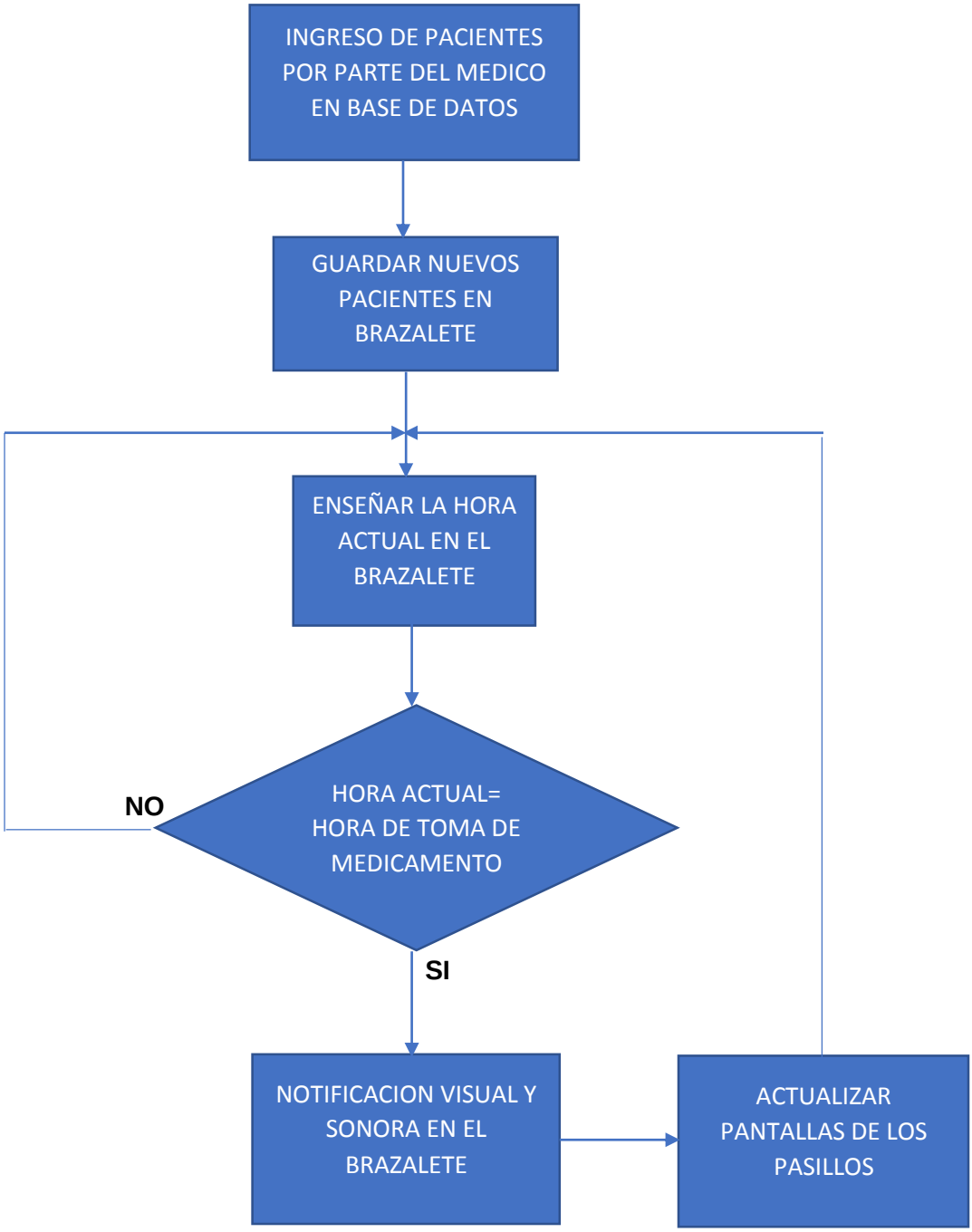
CAPÍTULO 3

3. DISEÑO Y MODELO DEL SISTEMA.

3.1 Escenario.

El lugar donde se utilizará el sistema electrónico a desarrollar es el asilo de ancianos “Hogar San José”, ubicado en la avenida Plaza Dañín. En dicho lugar habitan 98 ancianos, los cuales son cuidados por monjas de la Congregación de las Hermanitas de los Ancianitos Desamparados (algunas de las cuales también son de avanzada edad). Las cuidadoras y cuidadores reciben las recetas con las dosis de medicación proporcionadas por el médico tratante para posterior a ello administrar las medicinas a cada uno de los pacientes en los horarios previamente establecidos.

3.2 Modelo inicial planteado.



3.3 Modelo final propuesto.

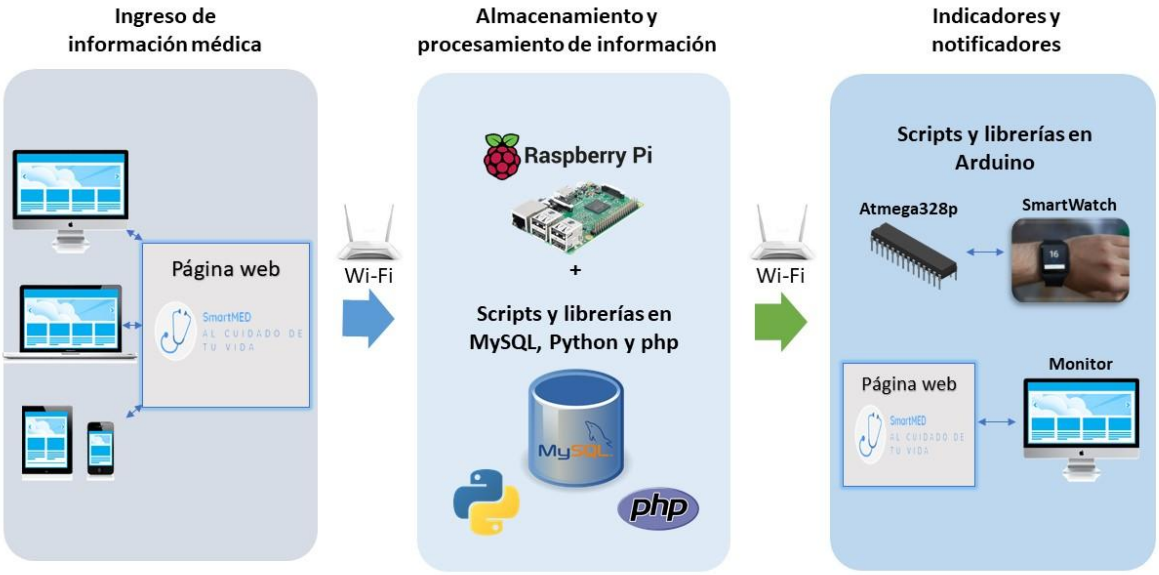


Figura 3.1 Modelo del sistema

Posterior a la realización de entrevistas y visitas al asilo de ancianos, se plantea el desarrollo de un modelo que consta de tres bloques, los cuales son: Ingreso de información médica, Almacenamiento y procesamiento de información e indicadores y notificadores. El modelo propuesto se ilustra en la Figura 3.1.

En el primer bloque se implementa el ingreso de la información médica a través de una página web, entre los cuales se tienen nombre del paciente, medicina recetada, dosis y horarios de administración de las mismas. El almacenamiento y procesamiento de la información, así como el manejo de la página web del primer bloque se describe en el segundo bloque. Finalmente, en el bloque de Indicadores y Notificadores se puede apreciar el sistema de notificación del brazalete electrónico, así como en un monitor con información desglosada de las alertas. Una descripción con más detalles de cada bloque mencionado anteriormente se muestra a continuación.

3.4 Bloque de Ingreso de Información Médica.

Previo al desarrollo del sistema propuesto se tomaron en cuenta diferentes aspectos de diseño, tales como: costos de herramientas de hardware (micro controladores, mini PC, pantallas, Access Points, etc.), las características de los microprocesadores, elección del software adecuado para una amigable e intuitiva interfaz y un eficaz procesamiento y administración de la información, recubrimientos y sistemas de autonomía energética en el SmartWatch, con la finalidad de que se cumplan a cabalidad los objetivos planteados y satisfacer los requerimientos del cliente ajustándonos a sus necesidades.

Para el ingreso de la información médica de cada paciente se ha optado por que sea a través de una página web basada en HTML versión 5 y Django, con la finalidad de ofrecer al usuario (médico) la posibilidad de acceder a ella a través de cualquier dispositivo portátil o computador de escritorio conectado a la red WiFi del asilo.

3.5 Indicadores y notificadores.

3.5.1 Brazalete electrónico.

Luego de que son ingresados los nuevos pacientes con sus medicinas y horarios, estos son ingresados automáticamente a la base de datos. Luego de esto, se produce una actualización en el sistema del brazalete, ingresando los nuevos horarios de toma de medicamentos (todo esto de manera inalámbrica).

Los componentes usados para la creación del brazalete electrónico fueron:

Arduino Uno: usado para realizar la programación del microcontrolador.

ATmega 328P: microcontrolador usado para que controle todo el sistema.

Módulo de *Reloj de tiempo real o RTC* (viene del inglés, *Real Time Clock*): módulo usado para mostrar la hora actual cada vez que se lo necesite. Lo especial de este módulo es que permite llevar un conteo actual de la hora y la fecha sin importar que se corte la alimentación principal.

Módulo WiFi ESP8266: módulo usado para conectarse a la red, y así recibir las actualizaciones de los horarios de medicación. En este se realizó la comunicación por protocolo serial.

Pantalla de *Diodo Orgánico de Emisión de Luz u Oled* (viene del inglés, *Organic Light-emitting diode*) Adafruit: pantalla usada para mostrar las notificaciones, así como hora y fecha.

Como componentes menores, se usaron capacitores, resistencias, y un buzzer para realizar las notificaciones sonoras (las mismas que suenan cada vez que exista una alerta de pastilla).

Se realizó la programación del microcontrolador, escribiendo código en lenguaje C en el ambiente de desarrollo de Arduino. (ver Anexos). La programación que se realizo fue básicamente para controlar la interacción entre la pantalla, módulo RTC y módulo WiFi.

Luego de haber realizado las pruebas, se realizó el diseño de una *Placa de circuito impreso o PCB* (viene del inglés, *Printed Circuit Board*), en la cual se procedió a soldar todos los componentes, y así mismo el diseño de una carcasa la cual fue impresa en 3D.

El funcionamiento básico del prototipo ya en su etapa final (véase Figura 3.2), constaba únicamente de un interruptor, un botón y la pantalla (únicos elementos que vería el usuario final). El interruptor serviría para apagar y prender el dispositivo, el botón para solicitar ver la hora y fecha actuales, y la pantalla para poder ver la hora, fecha y alertas cuando correspondan.

Figura 3.2 Prototipo en etapa final

3.6 Protocolos y Normativas utilizadas.

- ISO 8601, Normativa que asegura la compatibilidad de la información correspondiente a la representación de fecha y hora, intercambiada entre la base de datos del servidor, el software de control escrito en Python y los dispositivos terminales (brazalete inteligente).
- IEEE 802.11b/g/n, Estándar utilizado para la comunicación inalámbrica a través de la implementación de una WLAN, entre la Raspberry PI3 B+, computadores de escritorio o portátiles, teléfonos inteligentes y el brazalete inteligente.
- Protocolo de comunicación I2C: este permite comunicar por medio de 4 pines (VCC, GND, SCL, SDA) el microcontrolador con sensores.

NORMA USADA	USO
ISO 8601	COMUNICACIÓN CON EL SERVIDOR
IEEE 802.11B/G/N	COMUNICACIÓN INALAMBRICA
PROTOCOLO DE COMUNICACIÓN I2C	COMUNICACIÓN CON PANTALLA Y RELOJ DEL BRAZALETE

CAPÍTULO 4

4. RESULTADOS OBTENIDOS.

4.1 Pruebas de consumo eléctrico.

Al momento de desarrollar el prototipo, se realizaron pruebas para determinar cuánto sería el consumo del circuito, y así mismo también determinar la duración de la batería. Se utiliza una batería de 3.7V con capacidad de 1400mAh.

Se pudo determinar que el dispositivo consume una corriente de 100mA estando en modo reposo (con la pantalla apagada y sin recibir datos por el módulo WiFi).

Con la pantalla prendida (al solicitar ver la hora y fecha), se pudo apreciar un consumo de corriente de 250mA.

Sin embargo, al estar la pantalla encendida y el módulo WiFi receptando información, se pudo registrar un consumo de 500mA.

Como ese consumo no es habitual, y es solo por ciertas horas del día, la autonomía del dispositivo logra ser de 8 horas seguidas.

4.2 Pruebas de funcionamiento del brazalete.

Se realizaron diferentes tipos de pruebas para probar el funcionamiento del brazalete.

En primer lugar, se hicieron pruebas con una base de 50 pacientes con diferentes horarios en los cuales debían tomar medicina. Dicha prueba fue fructífera, ya que se pudo comprobar que las alarmas sonaban en el momento adecuado.

Luego de esto, se probó el sistema de recepción del brazalete, en el cual se pudo comprobar de la misma manera que se actualizaba su base de datos interna para implementar los nuevos horarios.

Como última prueba, se utilizó durante varios días el dispositivo para probar si existía algún problema debido al uso, o si alguna alarma dejaba de sonar. Se pudo comprobar así mismo que el dispositivo dispone de autonomía durante 8 horas, y que las alarmas programadas y las alarmas adquiridas mediante la actualización, suenan cuando deben de hacerlo.

4.3 Funcionamiento de la interfaz web.

La información presentada a continuación describe el funcionamiento de la página web en conjunto con la base de datos del sistema electrónico propuesto en el presente proyecto de materia integradora.

El acceso a las respectivas páginas web es realizado por personal dividido en dos roles, uno es el médico y el otro es el cuidador. El acceso a la web es local a través de los dispositivos conectados a la red WLAN.

4.3.1 Interfaz web/rol médico.

En el acceso realizado por el Rol del Médico es realizado desde cualquier dispositivo con conexión de la *Red de Área Local o LAN* (viene del inglés, *Local Area Network*) mencionada a través de un navegador compatible con HTML5.

Figura 4.1 Web de inicio - Rol de Médico

La Figura 4.1, la generó el servidor Web, en la cual se puede apreciar la página de inicio del sistema, en la cual el médico puede agregar un paciente nuevo en el caso que sea necesario o en su defecto realiza la búsqueda de un paciente ya ingresado previamente para realizar la consulta o modificación de la dosis asignada.

Figura 4.2 Ingreso de nuevo paciente

En la Figura 4.2, se puede apreciar la página web correspondiente al ingreso de un nuevo paciente por parte del médico en el cual se solicitan los nombres y apellidos completos del mismo con la finalidad de registrarlo y agregarlo a la base de datos de todos los pacientes.

Figura 4.3 Búsqueda de paciente.

En el caso que el médico desee realizar la búsqueda de un paciente específico para consultar su estado actual de medicación asignada puede realizarlo ingresando el nombre o apellido del paciente, al hacer “clic” en “buscar”, tal como se puede apreciar en la Figura 4.3.

Figura 4.4 Despliegue de pacientes existentes en la base de datos.

Por otro lado, en caso que se desee realizar la búsqueda o el despliegue del listado de todos los pacientes existentes en la base de datos del servidor se lo realiza al hacer “clic” en “Buscar todos”. Los resultados obtenidos se pueden apreciar en la Figura 4.4. En dicha página web se puede realizar la eliminación, modificación de una dosis asignada previamente o en su defecto se puede agregar una nueva medicación al paciente que el médico considere adecuado.

4.3.2 Interfaz web/rol cuidador.

Una vez ingresada la información respectiva a la base de datos de los pacientes por parte del médico, la información pasa a ser procesada y mostrada a través de una pantalla ubicada en un lugar estratégico de fácil visualización con el detalle de las medicaciones pendientes de entrega ordenadas cronológicamente, además en el caso de que exista una medicación pendiente de entrega, la misma titilará en un sombreado rojo

para alertar visualmente al cuidador como complemento al brazalete inteligente, tal como se lo puede observar en la Figura 4.5.

Figura 4.5 Alerta visual en pantalla para cuidador.

Una vez que el cuidador lee el detalle de medicación, se procede a realizar la administración de los medicamentos para posteriormente acercarse a un computador en donde a través de otra página web se podrá realizar la retroalimentación de entrega de medicamentos respectivos al sistema para finalizar con el proceso de medicación; esta retroalimentación es realizada al hacer “clic” en la fila correspondiente a la medicación administrada al paciente, tal como se muestra en la Figura 4.6, de esta manera la alerta visual de titileo desaparece así como la información de medicación de entrega pendiente de la pantalla que se pudo apreciar en la Figura 4.5, finalizando el proceso de entrega de medicación.

Figura 4.6 Web de retroalimentación al sistema de entrega de medicación.

4.4 Costos de elementos.

En la siguiente tabla se presentan los costos de nuestro dispositivo, en el cual se han utilizado herramientas de bajo costo ya que en su mayoría son de hardware libre, sin embargo, en el caso que se desee implementar el dispositivo a mayor escala y para mayor número de pacientes se deberán utilizar recursos de hardware más robustos, para lo cual el costo ascendería de acuerdo a los requerimientos y capacidad del sistema electrónico requerido.

Tabla 4.1 – Costos de elementos.

Elemento	Costo unitario
Raspberry PI 3B	$ 102.00
Adaptador HDMI – VGA	$ 15.00
Monitor 19 pulgadas	$ 130.00
Kit Alimentación 5VDC – 3ª	$ 15.00
Disipadores para Raspberry PI3B	$ 4.00
Kit Mouse + teclado	$ 17.00
Router WiFi 2 antenas	$ 24.00
Arduino UNO	$ 20.00
Módulo RTC	$ 5.00
Módulo WiFi	$ 7.00
Pantalla OLED	$ 6.00

Componentes electrónicos	$ 10.00
PCB	$ 10.00
Impresión 3D	$ 50.00
Total	**$ 415.00**

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES.

Basados en los requerimientos y necesidades del “Hogar San José” enfocados en el sector de la salud, se implementó un sistema electrónico utilizando microcontroladores para el brazalete electrónico y Raspberry PI3 modelo B+ para el servidor web, base de datos y servidor de notificaciones, además gracias a la flexibilidad y fácil integración del lenguaje de programación Python con protocolos de comunicación IPv4, gestión de base de datos MySQL, microcontroladores ATmega328P y recursos de fecha y hora del sistema operativo Raspbian actualizada desde la red LAN obtenida inalámbricamente, se logró diseñar el sistema capaz de notificar la alerta de administración de medicamentos establecida previamente por el médica a través de una comunicación inalámbrica

aprovechando la tarjeta de red inalámbrica de la Raspberry Pi3 y los módulos ESP8266 Wi-Fi que se integraron a los microcontroladores en los brazaletes con el respectivo uso de las librerías ESP8266 disponibles en los foros especializados en Arduino ya que la programación del microcontrolador fue realizada utilizando el Software Arduino.

En la etapa inicial del diseño del dispositivo electrónico se realizaron pruebas de cobertura con tecnología Bluetooth, sin embargo, su rango de cobertura era inferior al disponible en las redes Wi-Fi, por lo que se optó muy desarrollar el sistema utilizando comunicación Wi-Fi, además la administración de dispositivos es más sencilla ya que se realiza la identificación de los mismos a través de direcciones IP fijas.

Se puede apreciar que la interfaz web es una gran ayuda para los usuarios debido a su simplicidad en el manejo y la confiabilidad en el almacenamiento y gestión de la información clínica.

Los usuarios finales redujeron el tiempo invertido y la alta confusión en la administración de las medicinas a los pacientes debido al uso de SmartMED.

Se almacenaron los cronogramas de medicación de cada uno de los pacientes (adultos mayores), para el uso posterior por parte del médico y cuidadores.

Se recomienda utilizar equipos servidores dedicados basados en Linux, que incrementen en el desempeño en la gestión de base de datos y servicios web, es decir un equipo servidor para base de datos independiente al equipo servidor de los recursos web, para la implementación futura en centros médicos de mayor magnitud como hospitales o clínicas que disponen de un mayor número de usuarios pacientes y personal médico.

La electrónica utilizada puede ser reducida mediante el uso de placas electrónicas y componentes *SMD o Tecnología de Montaje Superficial* (viene del inglés, *surface-mount device*) ya que ocupa menos espacio y permite la utilización de diferentes

capas de pistas conductoras, esto logrará que el brazalete sea de menor volumen y por ende menos invasivo y más ergonómico para el usuario.

REFERENCIAS

[1] M. d. I. E. y. Social, «Ministerio de Inclusión Económica y Social,» 2015. [En línea]. Available: https://www.inclusion.gob.ec/direccion-poblacion-adulta-mayor/. [Último acceso: 05 Agosto 2019].

[2] E. Telégrafo, «El asilo, última opción para el adulto mayor,» *El asilo, última opción para el adulto mayor,* 14 Febrero 2015.

[3] R. d. Ecuador, «Defensoría pública del Ecuador,» 2015. [En línea]. Available: http://biblioteca.defensoria.gob.ec/bitstream/37000/823/1/Constituci%c3%b3n%20de%20la%20Rep%c3%bablica%20del%20Ecuador%202008.pdf. [Último acceso: 8 Agosto 2019].

[4] «Diario Enfermero,» [En línea]. Available: https://diarioenfermero.es/una-nueva-pulsera-digital-asegura-la-identificacion-administracion-medicacion-pacientes-quimioterapia/. [Último acceso: 2019 09 20].

[5] I. P. Nacional, «Boletín UPIITA,» Instituto Politécnico Nacional, 1 Julio 2018. [En línea]. Available: http://www.boletin.upiita.ipn.mx/index.php/ciencia/771-cyt-numero-67/1539-una-introduccion-a-las-single-board-computers. [Último acceso: 12 Agosto 2019].

[6] Raspbian, «Raspbian,» [En línea]. Available: https://www.raspbian.org/. [Último acceso: 13 Agosto 2019].

[7] MySQL, «MySQL,» [En línea]. Available: https://www.mysql.com/products/standard/. [Último acceso: 22 Mayo 2019].

[8] PHP.net, «PHP.net,» [En línea]. Available: https://www.php.net/manual/es/intro-whatis.php. [Último acceso: 23 Junio 2019].

[9] A. Foundation, «Apache,» Apache Foundation, 2019. [En línea]. Available: https://www.apache.org/foundation/. [Último acceso: 27 Mayo 2019].

[10] «EcuRed,» 2 07 2016. [En línea]. Available: https://www.ecured.cu/Est%C3%A1ndar_inal%C3%A1mbrico_802.11b. [Último acceso: 1 09 2019].

[11] «Internet of thing sagenda,» 21 03 2012. [En línea]. Available: internetofthingsagenda.techtarget.com. [Último acceso: 20 08 2019].

[12] T. Point, «Python 3 - MySQL Database Access,» [En línea]. Available: https://www.tutorialspoint.com/python3/python_database_access.htm. [Último acceso: 2 Mayo 2019].

[13]

[14] S. Networks, «Social Networks, Seguridad y privacidad en redes sociales,» Junio 2014. [En línea]. Available: http://ssocialnet.blogspot.com/2014/06/arquitectura-de-una-red-social.html. [Último acceso: 23 Julio 2019].

[15] S. C. O. Traian, «Wikipedia,» 30 Agosto 2019. [En línea]. Available: https://es.wikipedia.org/wiki/LAMP#/media/Archivo:LAMP_software_bundle.svg. [Último acceso: 12 Septiembre 2019].

[16] Culturación, Agosto 2019. [En línea]. Available: http://culturacion.com/que-es-apache/. [Último acceso: Septiembre 2019].

ANEXOS

Anexo 1:

Desarrollo del servidor en Raspberry y creación de red WLAN.

Credenciales_Red_Wifi

Administración:

User: admin

Password: integradora

Red Wireless:

Name: Integradora_SmartMED

Password: integradora

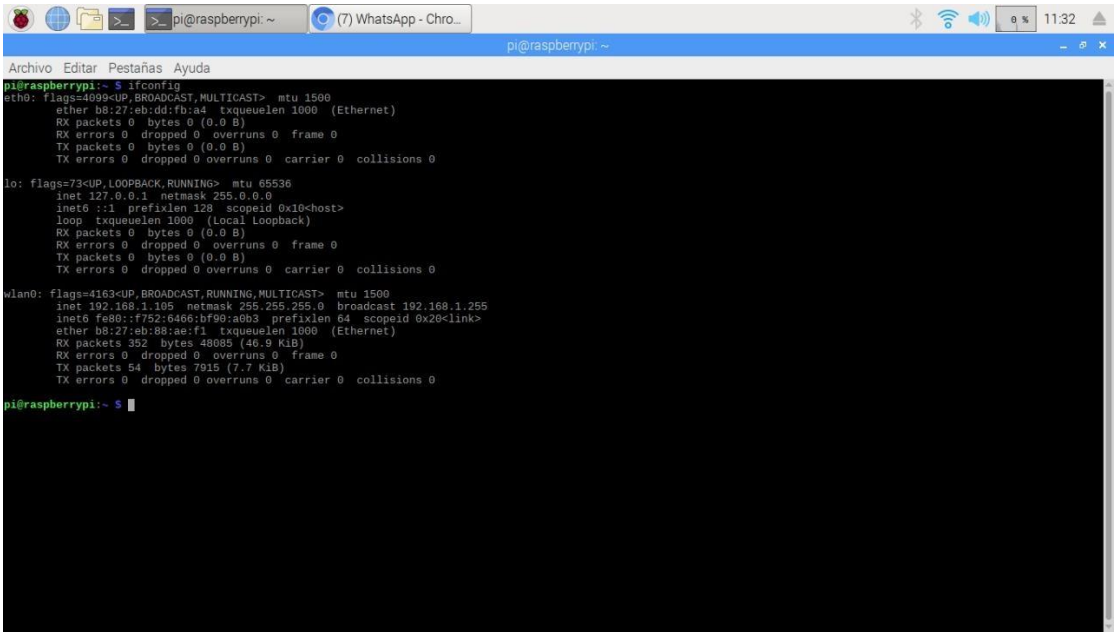
Servidor DHCP: Activo

Reservación de direcciones IP basado en MAC: Activo

MAC wlan0 Raspberry reservada: b8:27:eb:88:ae:f1

Dirección IP v4 asignada: 192.168.0.105.

Obtener la MAC de la Raspberry pi3: (actualizar IP)



Con el comando IFCONFIG se obtiene las direcciones IPv4 e IPv6, así como la dirección MAC de la interfaz de red ethernet o de la interfaz de red wlan, para este caso se trabajará con WLAN en la Raspberry.

Esto servirá para proporcionar una dirección IP estática a la Raspberry desde el Access Point usado.

Los recursos que se van a utilizar son los siguientes:

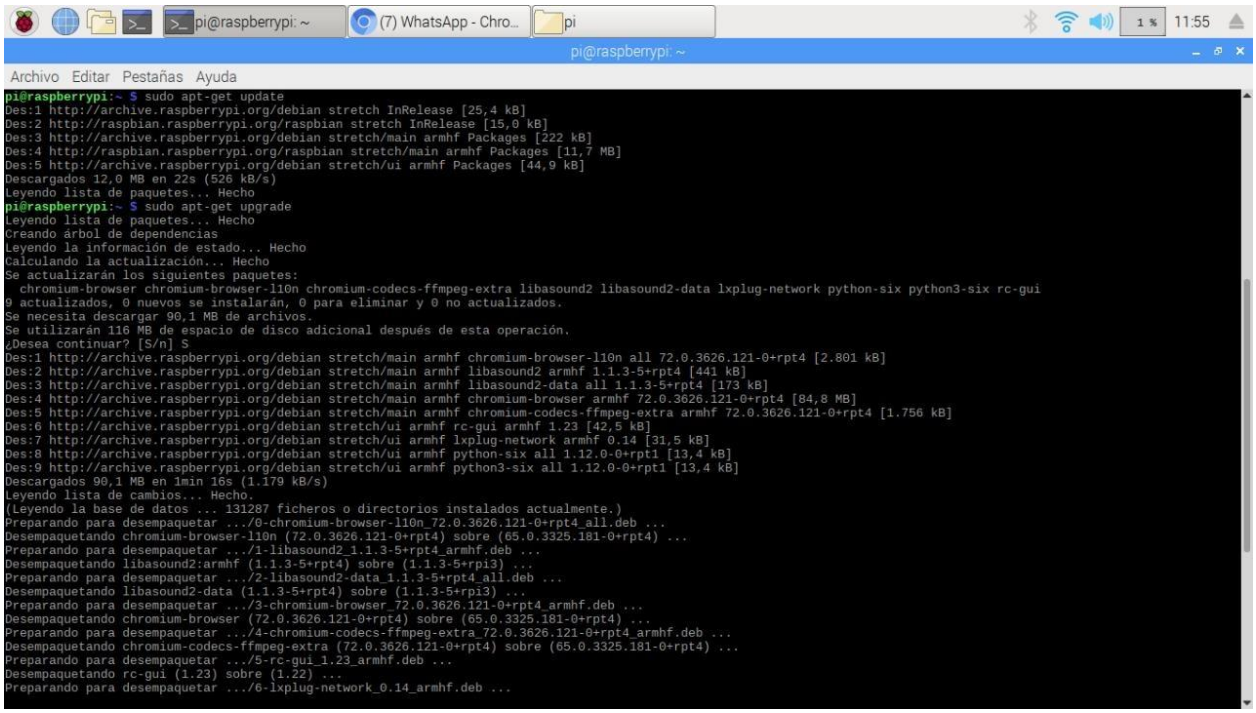
Raspbian como SO, basado en LINUX.

Apache como servidor web.

MySQL para gestión de la base de datos.

PHP como lenguaje de programación.

Actualización de recursos y software de la Raspbian.

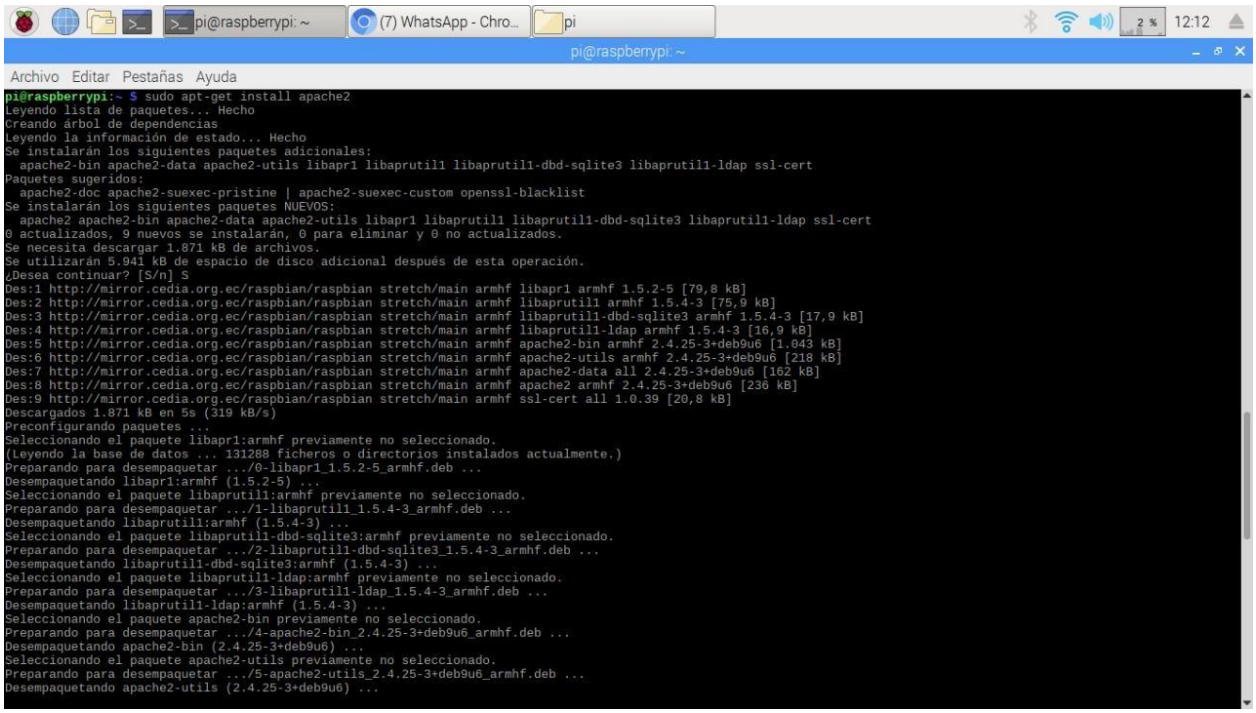


sudo apt-get update sudo

apt-get upgrade

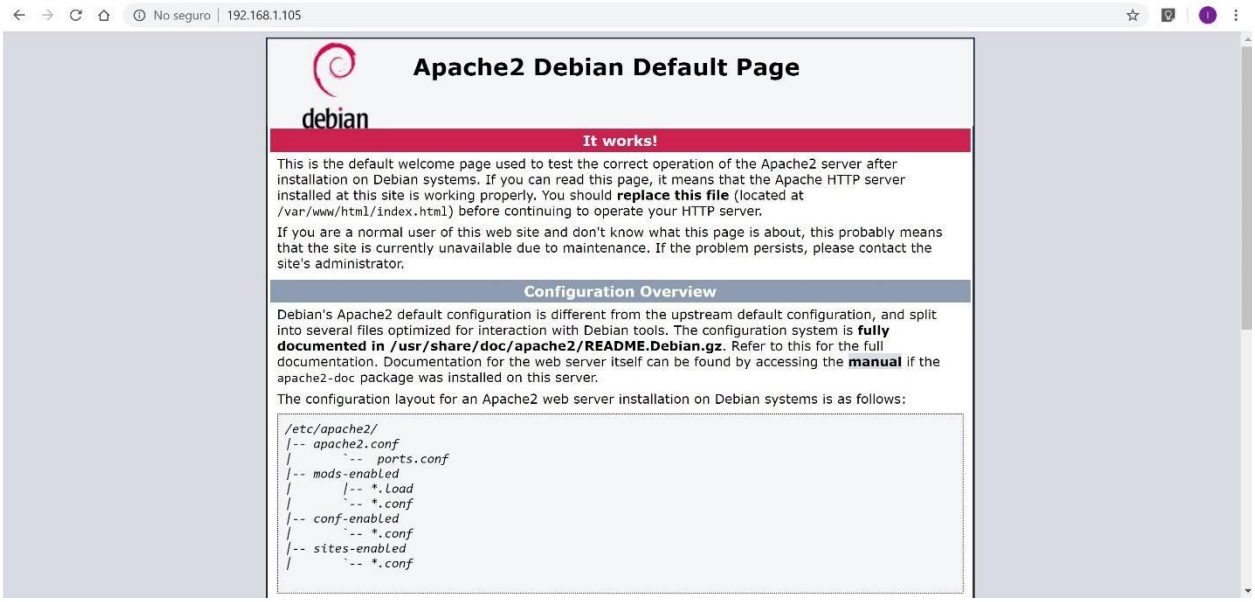
Instalación de Apache.

sudo apt-get install apache2



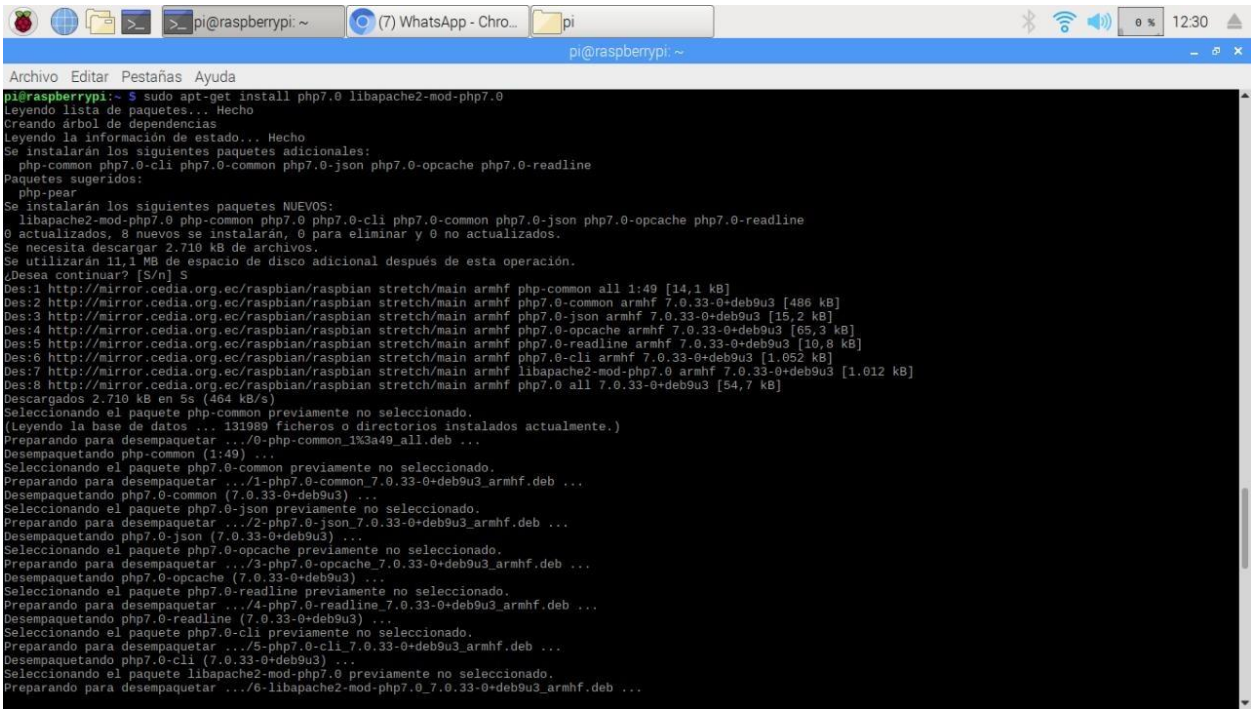
Verificación del servidor Apache desde otro ordenador utilizando la dirección IP de la Raspberry Server.

En este caso la dirección IP es: 192.168.0.105.



Instalación de PHP.

sudo apt-get install php7.0 libapache2-mod-php7.0



Reinicio de Raspbian para reiniciar el servidor.

sudo reboot

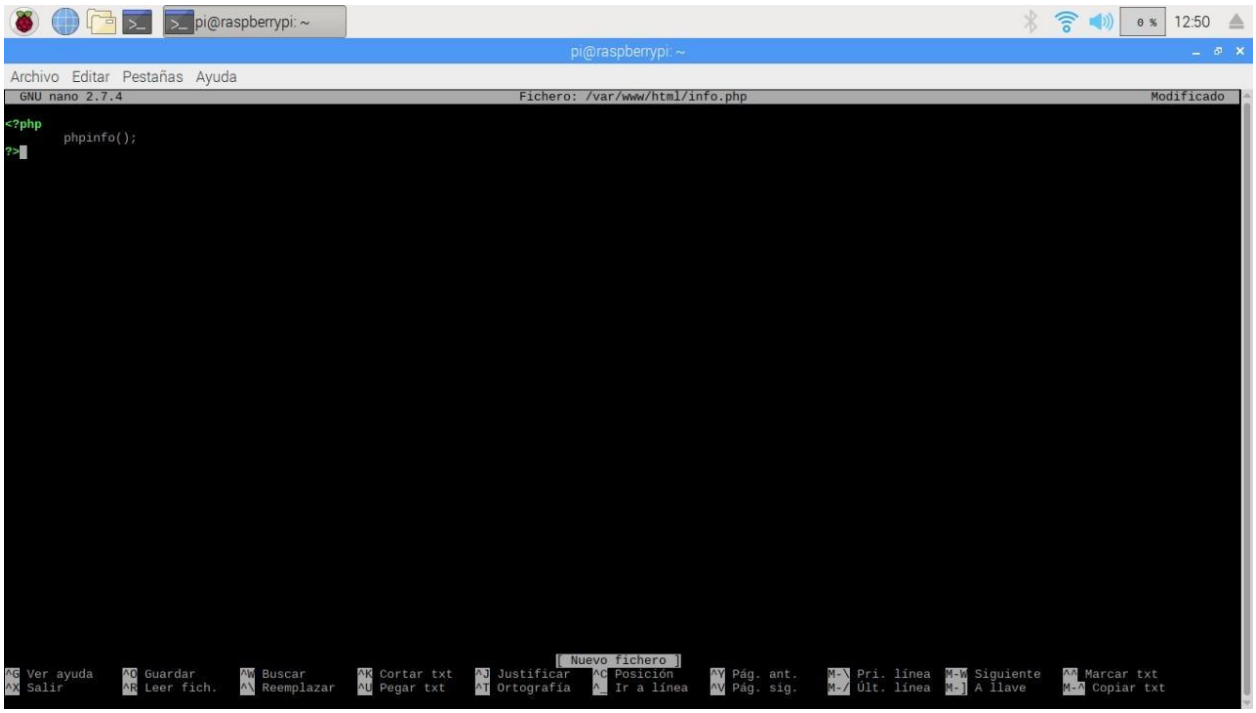
Creación de una página web como prueba Info.php sudo

nano /var/www/html/info.php

Al iniciar la creación de un nuevo archivo expuesto en la línea anterior, se escribe ese archivo con las siguientes líneas:

```
<?php
    phpinfo();
?>
```

Y se aplica guardar utilizando Ctrl+O.

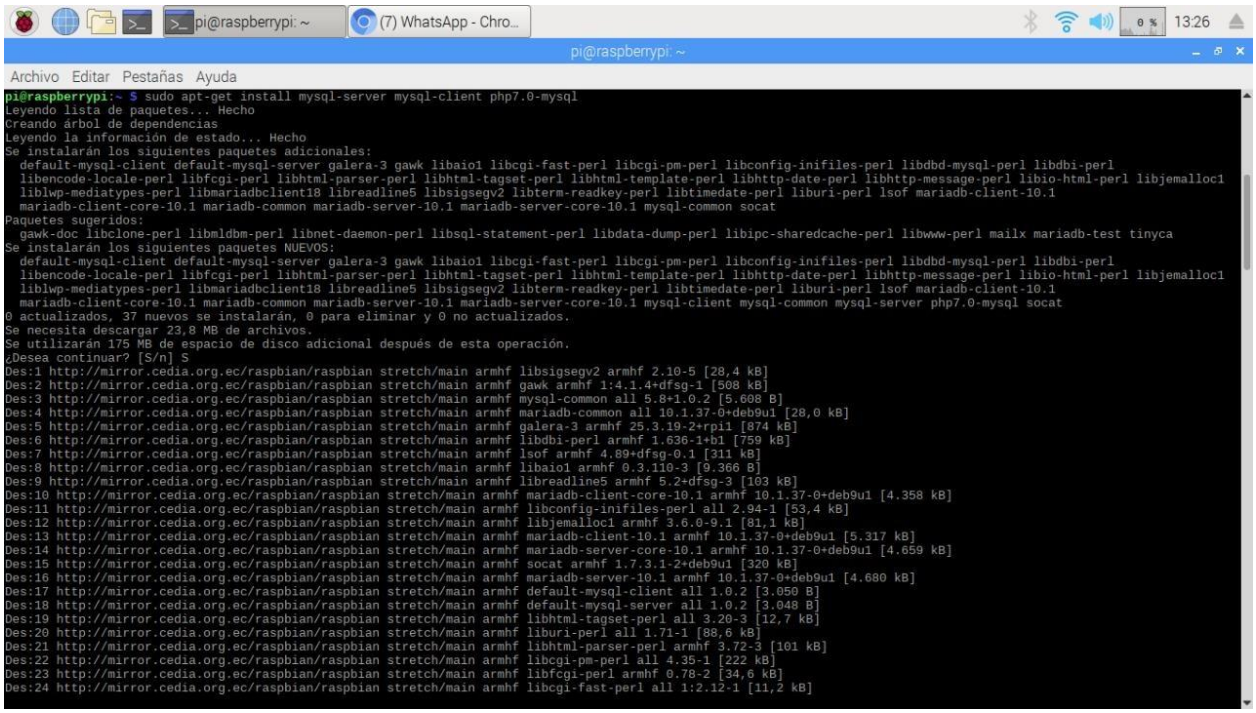


Verificación del archivo Info.php creado en el servidor Apache desde otro ordenador utilizando la dirección IP de la Raspberry Server.

En este caso la dirección IP es: 192.168.0.105/info.php

Instalación de MySQL.

sudo apt-get install mysql-server mysql-client php7.0-mysql

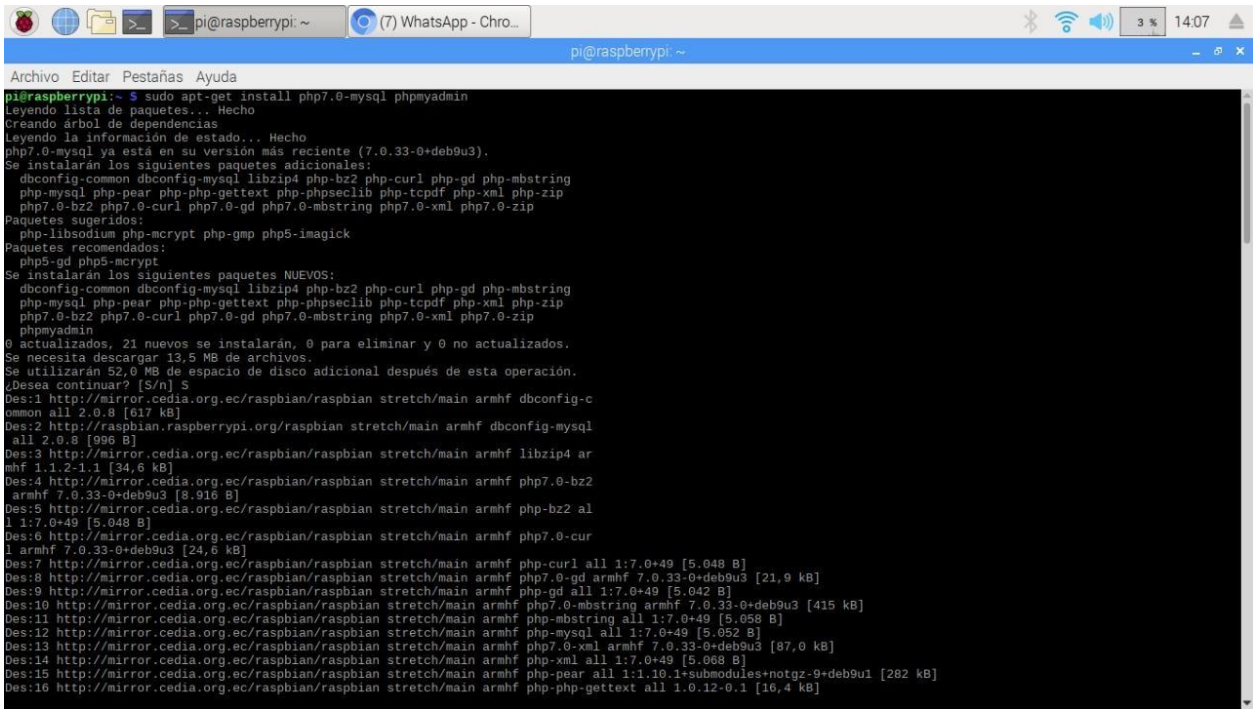


Al finalizar la Instalación, reiniciar.

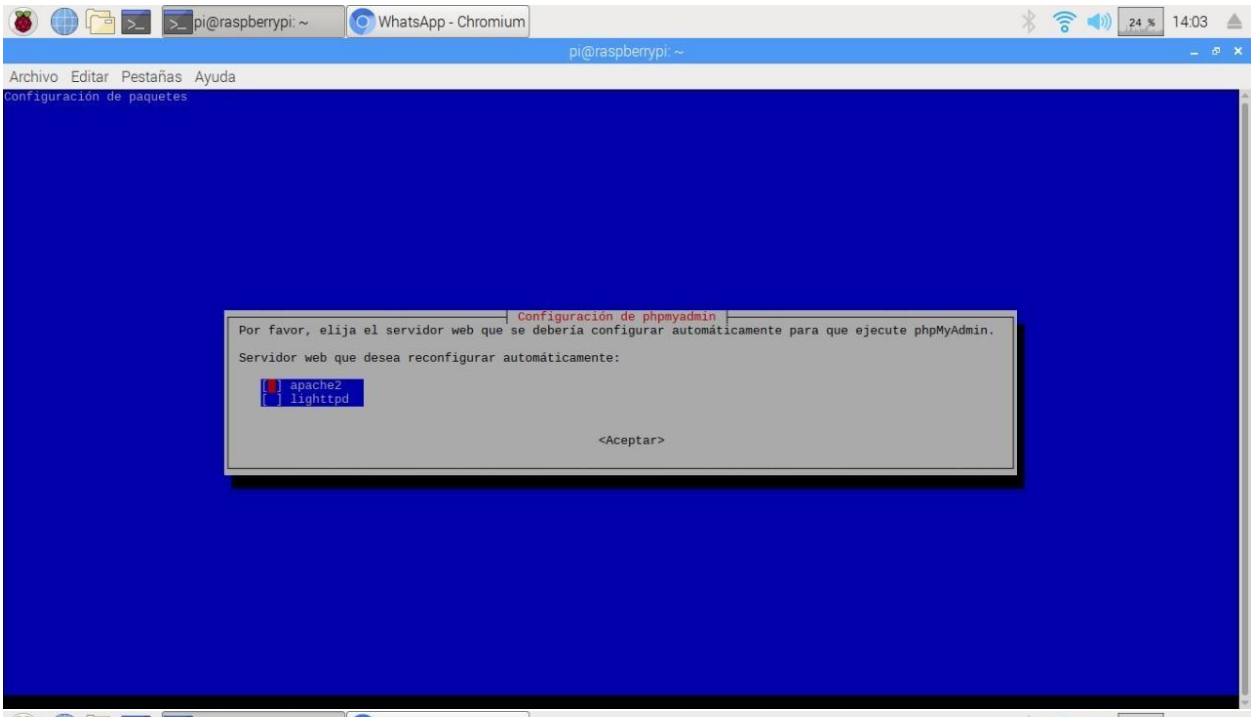
sudo reboot

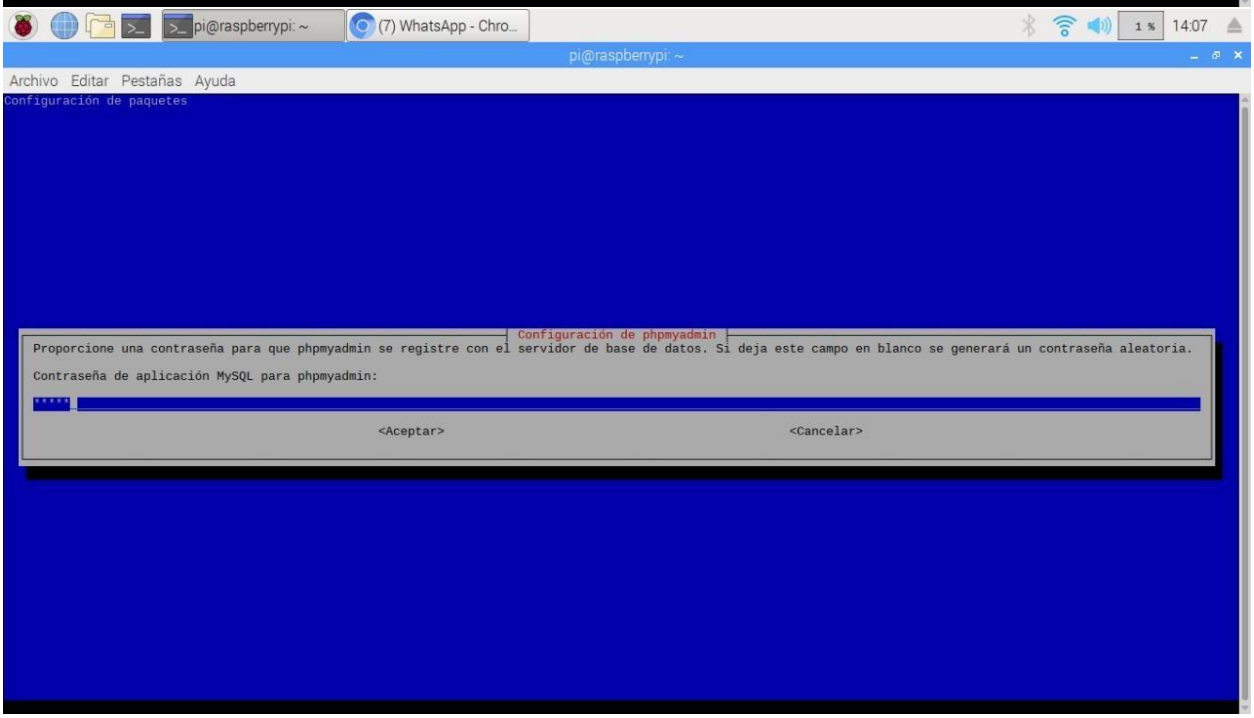
Instalación de PHPMyAdmin. para administrarla MySQL a través de internet.

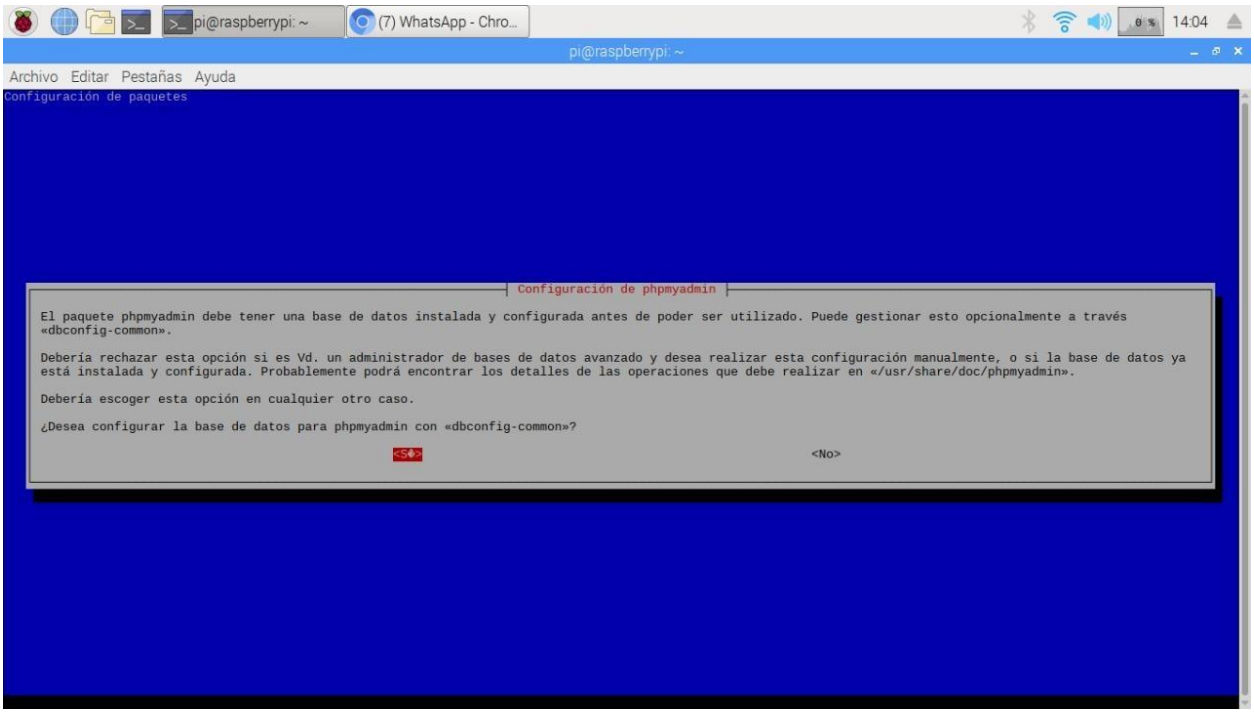
sudo apt-get install php7.0-mysql phpmyadmin



Durante la instalación, elegir el Servidor APACHE2 y configurar una contraseña, para este caso es “admin”.


pi@raspberrypi: ~
WhatsApp - Chromium
24 %
14:03
pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
Configuración de paquetes
Configuración de phpmyadmin
Por favor, elija el servidor web que se debería configurar automáticamente para que ejecute phpMyAdmin.
Servidor web que desea reconfigurar automáticamente:
apache2
lighttpd
<Aceptar>


pi@raspberrypi: ~
(7) WhatsApp - Chro...
1 %
14:07
pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
Configuración de paquetes
Configuración de phpmyadmin
Proporcione una contraseña para que phpmyadmin se registre con el servidor de base de datos. Si deja este campo en blanco se generará un contraseña aleatoria.
Contraseña de aplicación MySQL para phpmyadmin:
*****
<Aceptar>
<Cancelar>



Editar el archivo denominado apache2.conf desde terminal.

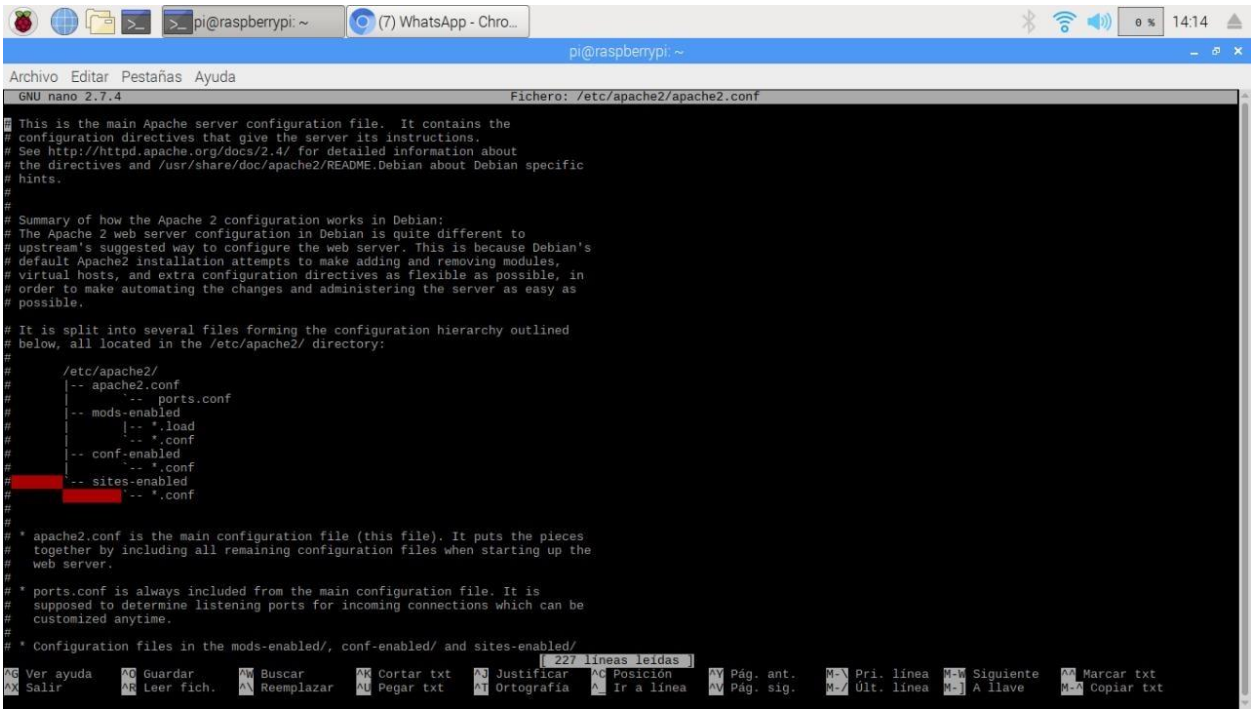
sudo nano /etc/apache2/apache2.conf

Al final de todo el texto del archivo mencionado anteriormente, escribir:

Include /etc/phpmyadmin/apache.conf

Luego, guardar con Ctrl+O

*Y Reiniciar la Raspberry (*sudo reboot*).*

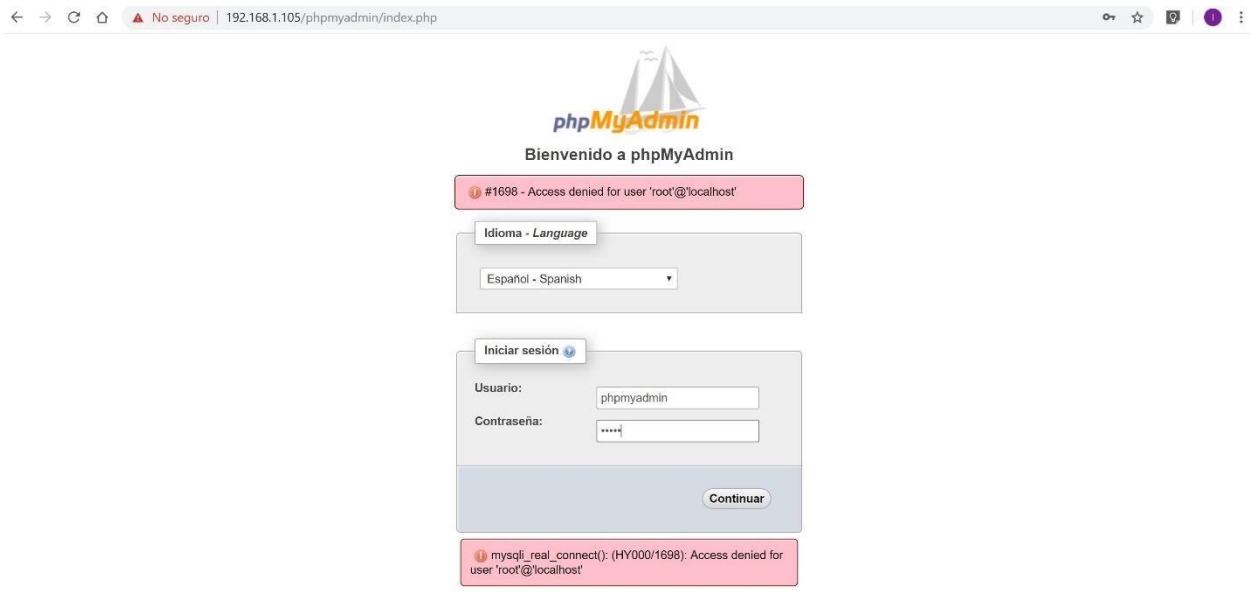


Verificación del servidor Apache usando MyPHPadmin desde otro ordenador utilizando la dirección IP de la Raspberry Server.

En este caso la dirección IP es: 192.168.0.105/phpmyadmin

Usuario: phpmyadmin

Password: admin



Creación de un nuevo usuario con todos los privilegios necesarios:

Acceder al terminal de la Raspberry, y escribir el siguiente comando:

sudo mysql -uroot mysql

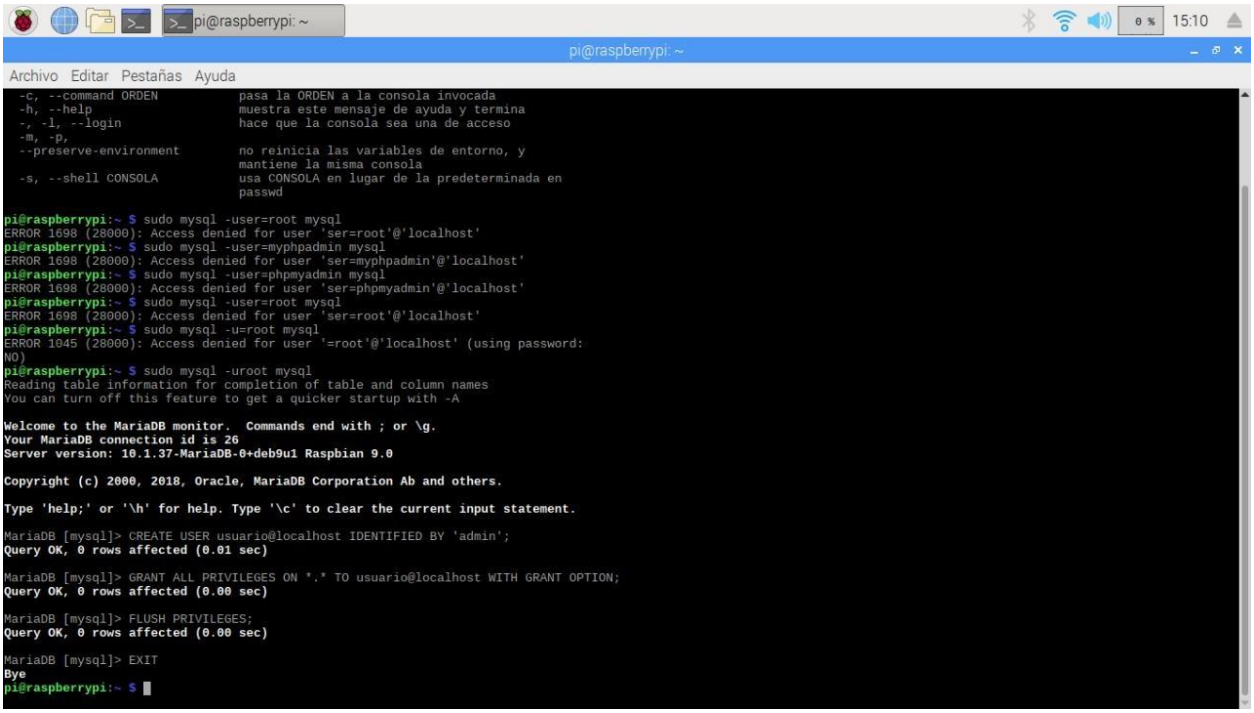
A continuación, escribir:

CREATE USER usuario@localhost IDENTIFIED BY ‘admin’;

GRANT ALL PRIVILEGES ON *.* TO usuario@localhost WITH GRANT OPTION; FLUSH PRIVILEGES;

Cabe recalcar que admin es la clave y usuario es el nombre del nuevo user creado.

*Reiniciar la Raspberry (*sudo reboot*).*



Creación de base de datos:

Iniciar sesión:

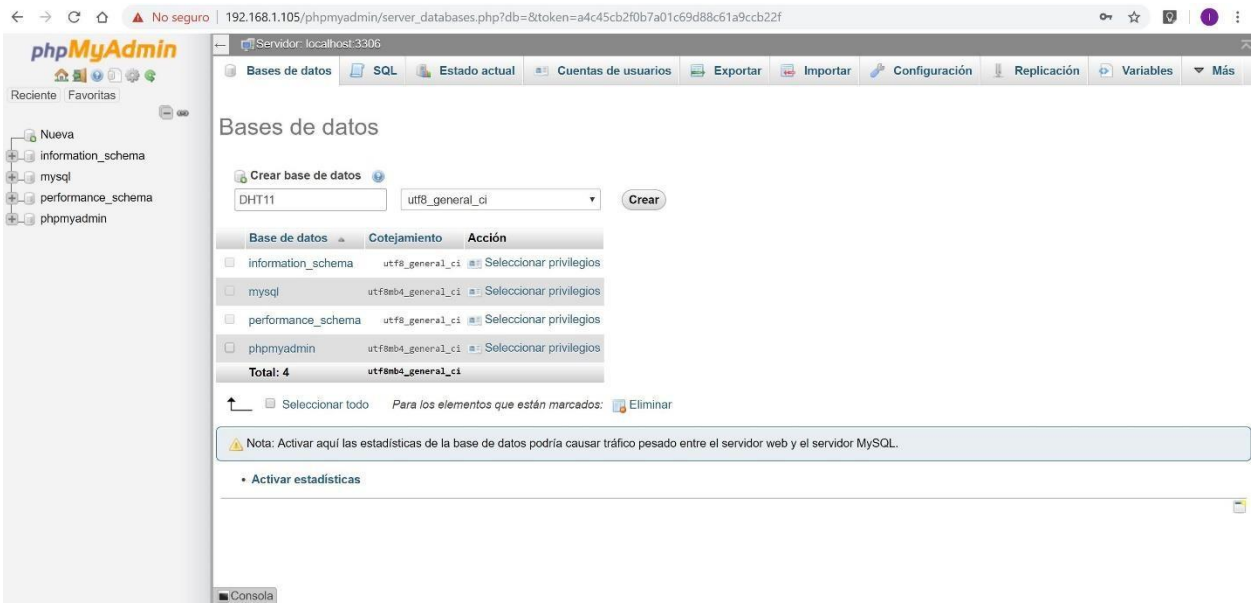
En este caso la dirección IP es: 192.168.0.105/phpmyadmin

Usuario: usuario

Password: admin

Clic en: Base de datos.

En el apartado base de datos escribir el nombre de la nueva base de datos, para nuestro caso es DHT11 y elegir en cotejamiento “utf8_general_ci” y clic en “Crear”.



Crear una tabla de datos, en este caso con 4 campos, denominada valores:

Llenar los 4 campos, como se detalla a continuación:

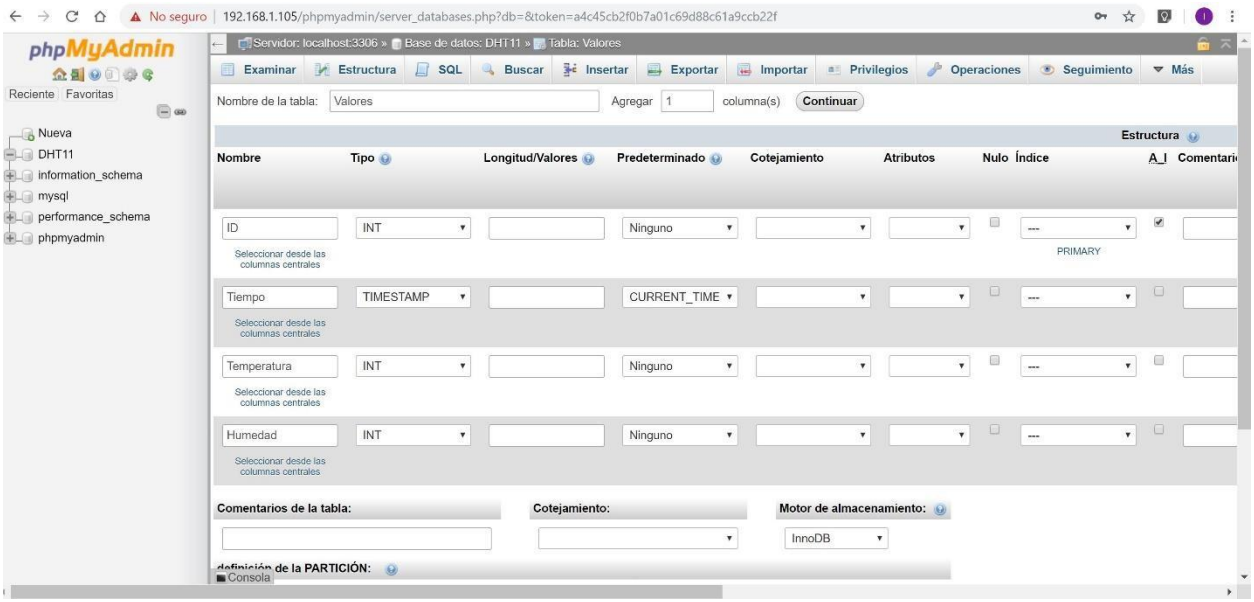
ID: identificador del registro. Será la clave primaria y marcaremos la casilla de auto incremento.

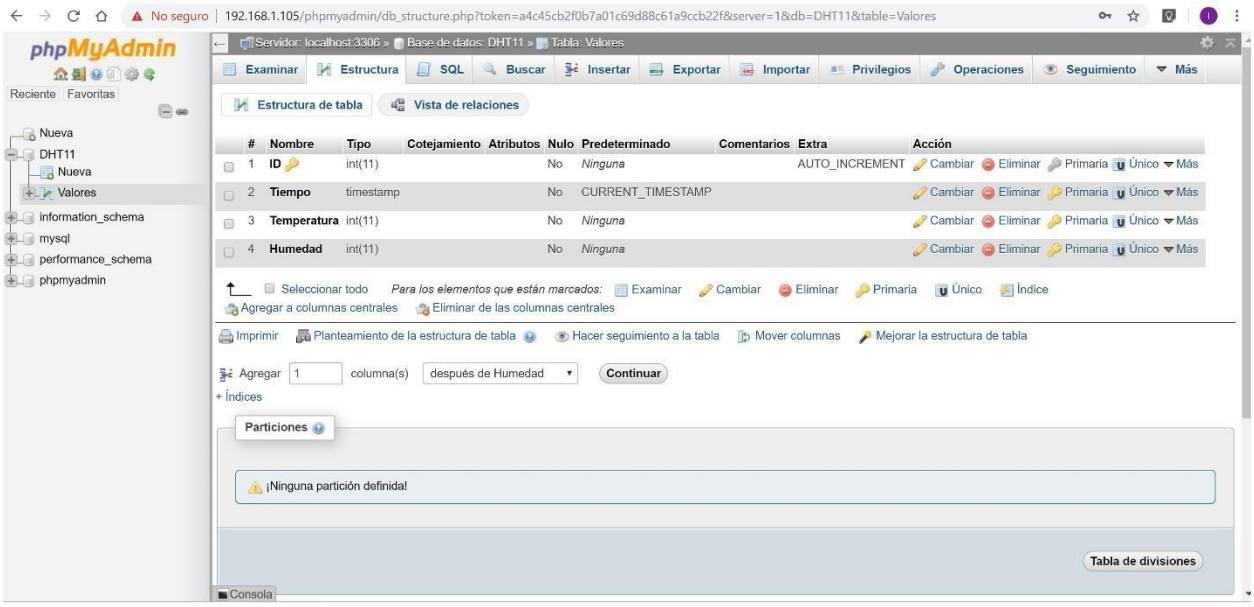
Tiempo: este campo almacenará la fecha y la hora en que se ha tomado la muestra. El tipo será TIMESTAMP y su valor por defecto CURRENT_TIMESTAMP.

Temperatura: en este campo se almacenará el valor de la temperatura del sensor. Será de tipo INT.

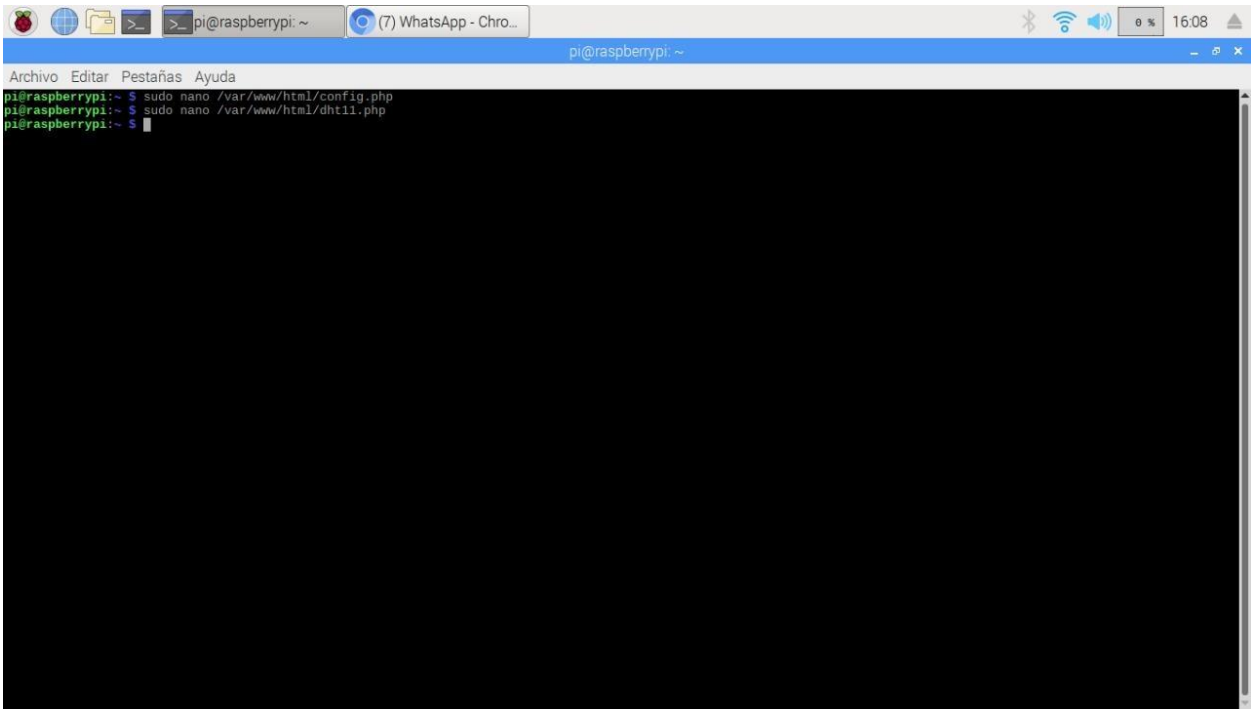
Humedad: este campo almacenará el valor de la humedad del sensor. Será de tipo INT.

Clic en guardar en la parte inferior derecha.





Creación de la página para subir valores a la base de datos creada.



Para poder subir datos a la base de datos creada es necesario crear una página en PHP, para este fin se crearán dos archivos en la Raspberry desde consola:

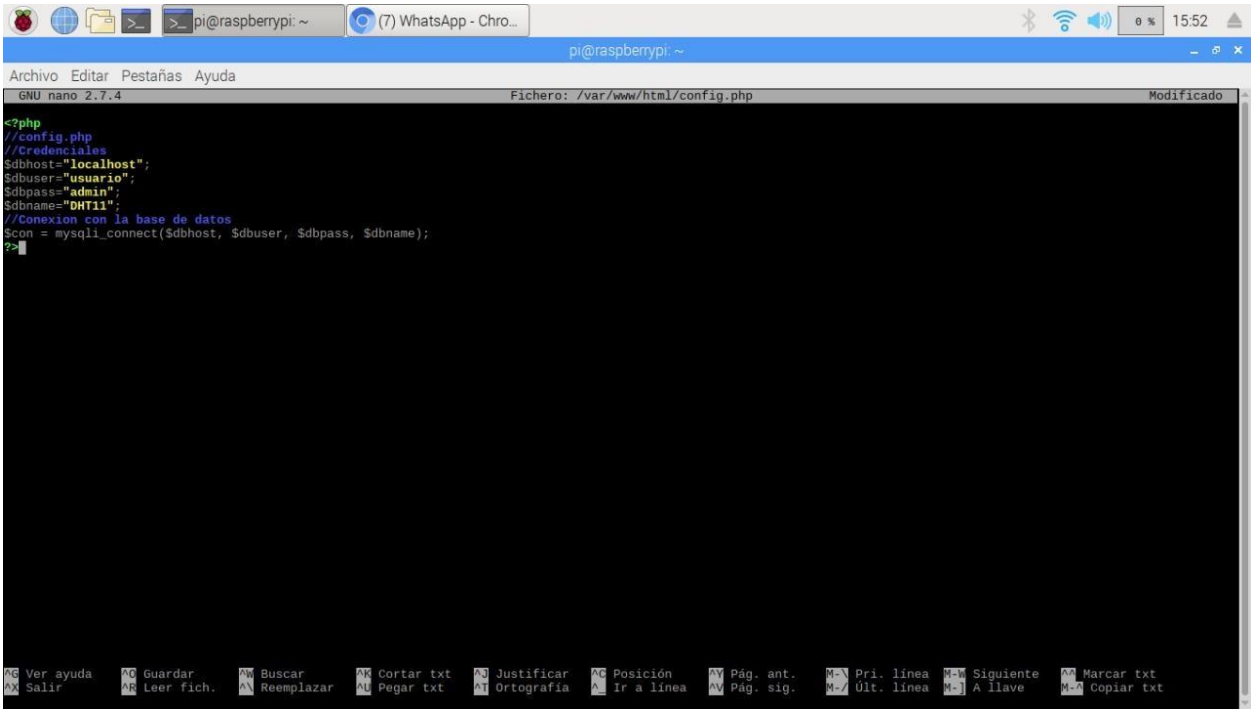
***Primer archivo**. (config.php) tendrá la información necesaria para establecer la conexión con la base de datos que hemos creado.*

sudo nano /var/www/html/config.php

Una vez abierto el archivo desde consola, escribir lo siguiente:

```
<?php
 // config.php
 // Credenciales
 $dbhost = "localhost";
 $dbuser = "usuario";
 $dbpass = "admin";
 $dbname = "DHT11";
 //Conexion con la base de datos
 $con = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
?>
```

Luego, guardar con Ctrl+O



***Segundo archivo**. (dht11.php) tendrá la página que usaremos para subir los datos*

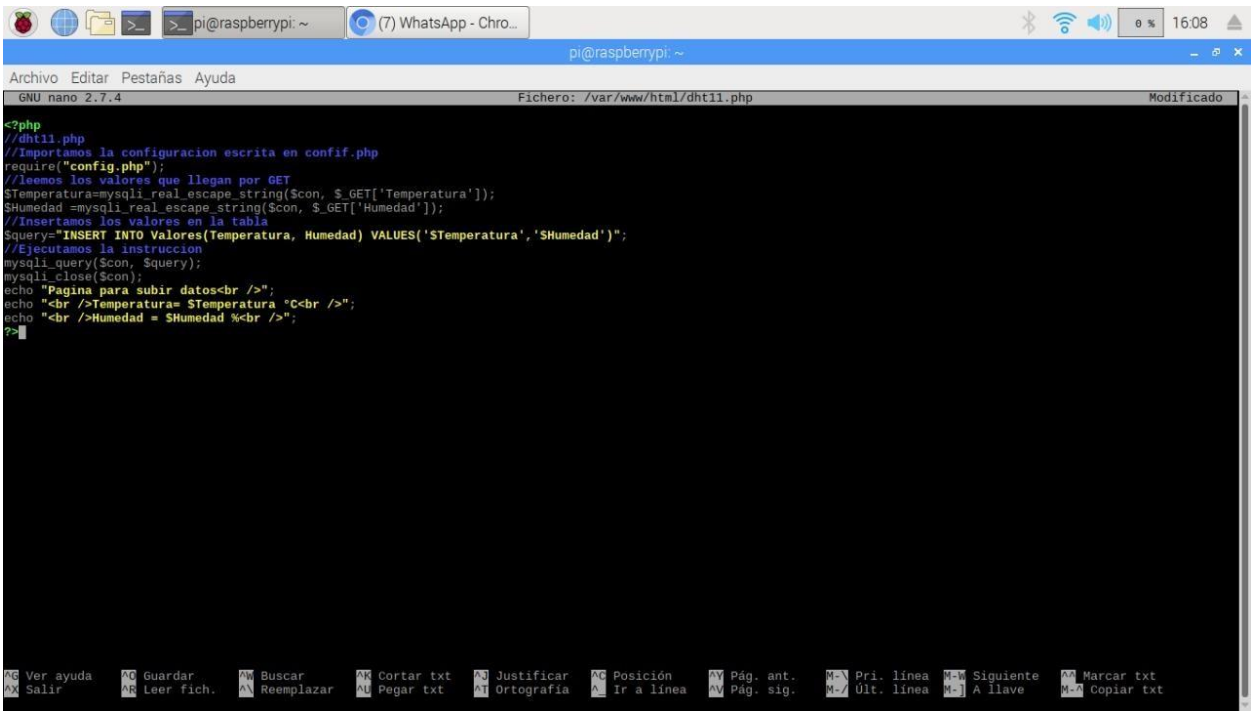
sudo nano /var/www/html/dht11.php

Una vez abierto el archivo desde consola, escribir lo siguiente:

```
<?php
 // dht11.php
 //Importamos la configuración
require("config.php");
 // Leemos los valores que nos llegan por GET
 $Temperatura = mysqli_real_escape_string($con, $_GET['Temperatura']);
 $Humedad = mysqli_real_escape_string($con, $_GET['Humedad']);
 // E Insertamos los valores en la tabla
 $query = "INSERT INTO Valores(Temperatura, Humedad) VALUES('$Temperatura','$Humedad')";
// Ejecutamos la instrucción
```

```
 mysqli_query($con, $query);  mysqli_close($con);
echo "Pagina para subir los datos<br />";  echo "<br
/>Temperatura = $Temperatura ºC<br />";  echo
"<br />Humedad = $Humedad %<br />";
?>
```

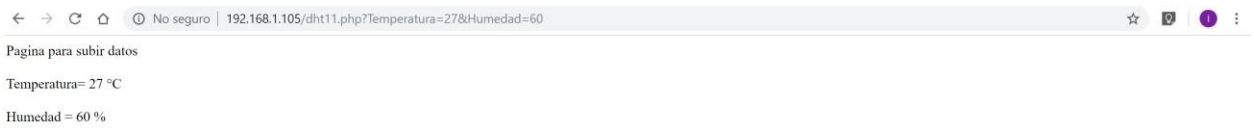
Luego, guardar con Ctrl+O



Escribir la IP de la Raspberry, seguido de /dht11.php?Temperatura=27&Humedad=60

En nuestro caso es:

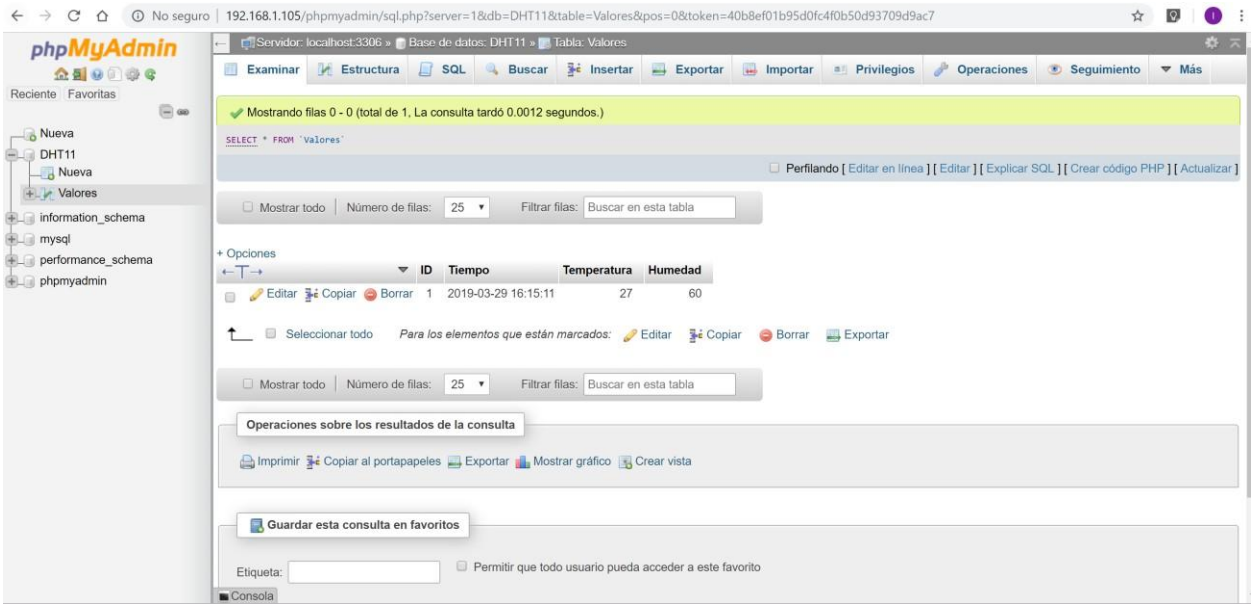
192.168.0.105/dht11.php?Temperatura=27&Humedad=60



Finalmente se puede apreciar que los datos han sido subidos sin problemas a la base de datos del servidor, ya que al entrar en:

192.168.1.105/phpmyadmin.php

Se observan los valores de temperatura y humedad.



Anexo 2:

Envío de datos al servidor desde Arduino UNO y configuración ESP8266.

Tomar en cuenta que el módulo wifi ESP8266 requiere una fuente de 3.3V con una corriente muy superior a 50mA en el arranque, se recomienda utilizar una fuente externa.

Envío de datos de sensores hacia el servidor en Raspberry Pi3.

Para las pruebas se utilizó un Arduino UNO montado con una Ethernet Shield, sin embargo, se desea implementar la comunicación vía WIFI, por lo cual se está usando la mini placa ESP8266.

Arduino usa los pines digitales 10, 11, 12, y 13 (SPI) para comunicarse con el W5100 en la ethernet Shield. Estos pines no pueden ser usados para e/s genéricas.

Los materiales a utilizarse son los siguientes:

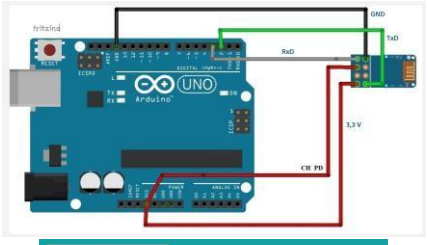
Arduino UNO con ethernet shield instalada (no se ha quitado intencionalmente).

Sensor DHT11.

Raspberry PI 3

Configuración del módulo ESP 8266, cambio de velocidad de transmisión serial.

La velocidad a la que opera el módulo por default es de 115200, pero debe cambiarse a 9600 para lograr una comunicación más estable con el microcontrolador del Arduino uno. Sin embargo, en la primera conexión se debe optar por la siguiente conexión, tomando en cuenta los puertos 2 y 3 como tx y rx en serial.



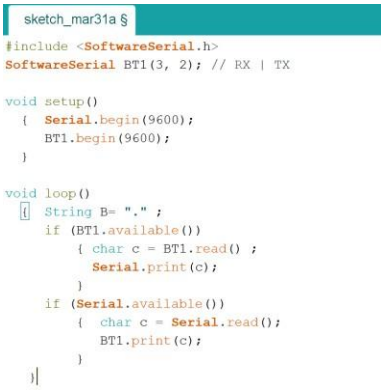
sketch_mar31a §

```
#include <SoftwareSerial.h>
SoftwareSerial BT1(3, 2); // RX | TX

void setup()
  {  Serial.begin(115200);
     BT1.begin(115200);
  }

void loop()
  {  String B= "." ;
     if (BT1.available())
         { char c = BT1.read() ;
           Serial.print(c);
         }
     if (Serial.available())
         {  char c = Serial.read();
            BT1.print(c);
         }
  }
```

De esta forma abriendo el monitor serial, se debe escribir el siguiente comando AT.

```
AT+UART_DEF=9600,8,1,0,0
```

De esta manera cambiará la velocidad de transmisión serial a 9600 baudios, posterior a esto, volver a cargar otro código en Arduino uno, cambiando la velocidad de 115200 a 9600 baudios.




```
#include <SoftwareSerial.h>

SoftwareSerial BT1(3, 2); // RX | TX



void setup()

 { Serial.begin(9600);

   BT1.begin(9600);

 }



void loop()

 { String B= "." ;

if (BT1.available())

      { char c = BT1.read() ;

        Serial.print(c);

      }

   if (Serial.available())

      { char c = Serial.read();

        BT1.print(c);

      }

 }
```

Configuración del módulo ESP 8266, cambio de modo de operación.

Se debe tener el módulo ESP 8266 en modo de operación 3, ya que esta proporciona el modo cliente+ servidor.

Mediante el monitor serial, enviamos el comando AT:

AT+CWMODE=3

Configuración del módulo ESP 8266, conexión a una red wifi.

Se visualizan todas las redes disponibles con el siguiente comando:

AT+CWLAP

Para conectarse a una red se aplica el siguiente comando, respetando los parámetros de ingreso.

AT+CWJAP=ssid,pwd

En nuestro caso es:

AT+CWJAP="Integradora_SmartMED","integradora"

La dirección estática asignada al módulo ESP8266 es la 192.168.0.106, y se la puede verificar mediante el comando AT:

AT+CIFSR

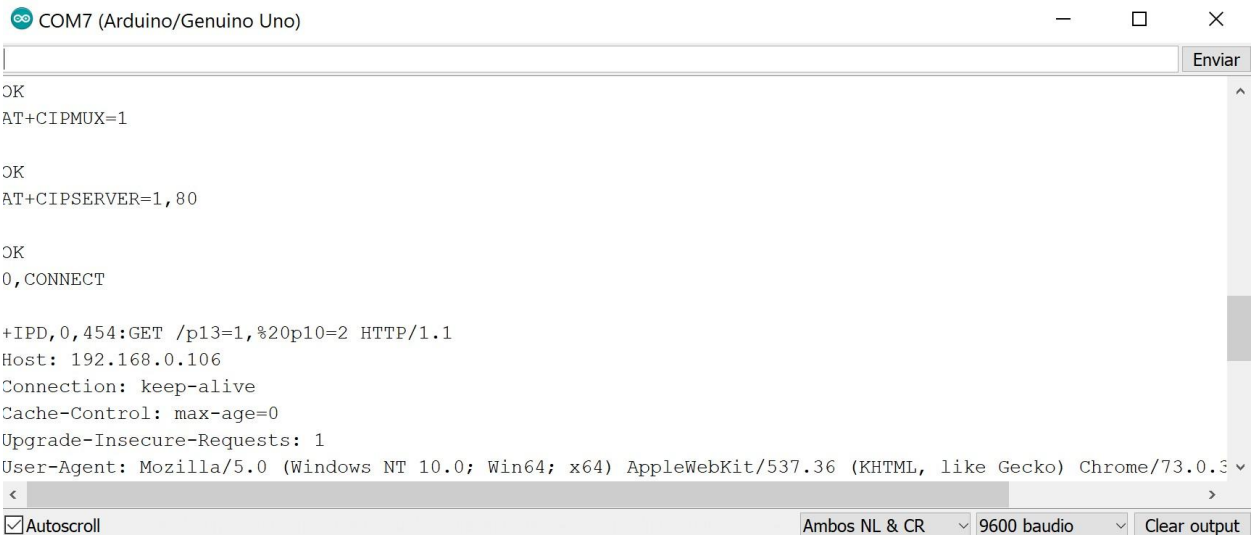
Configuración del módulo ESP 8266, activar el módulo para recepción de solicitudes http, simulando otro servidor web.

Para que el módulo ESP 8266 pueda recibir datos desde cualquier otro equipo o microcontrolador de manera inalámbrica, es necesario que se habiliten las conexiones simultáneas hacia el mismo; se activa con el siguiente comando:

AT+CIPMUX=1

Además, al recibir información mediante http (puerto 80) se comporta como un servidor, para aquello se debe activar esta función de la siguiente manera:

AT+CIPSERVER=1,80



Para realizar pruebas, se envió un string a la dirección del ESP 8266 utilizando su dirección IP.

Donde se pudo apreciar en el monitor serial del ESP8266 la recepción del string:

Se encontró que cada vez que se reseteaba el módulo esp8266 solo la configuración de conexión a la red wifi quedaba guardada, sin embargo, la configuración para recibir datos como servidor se deshabilitaba por ende se procedió a realizar la configuración mediante comandos AT a través de código en el microcontrolador Atmega.

Conexión automática y configuración del ESP8266 desde línea de código desde el microcontrolador.

Código utilizado: archivo: confmux.ino

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266(3, 2); // RX | TX
 void
setup()
  {         Serial.begin(9600);                          esp8266.begin(9600);
sendData("AT+RST\r\n",2000);                          //    resetear    módulo
sendData("AT+CIPMUX=1\r\n",1000);  //  configurar  para  multiples  conexiones
sendData("AT+CIPSERVER=1,80\r\n",1000);         // servidor en el puerto
80
  }  void loop()   {  String
B= "." ;      if
(esp8266.available())
         { char c = esp8266.read() ;
           Serial.print(c);
         }      if
(Serial.available())          {
char c = Serial.read();
esp8266.print(c);
         }
   }

//función sendData a través de serial void
sendData(String commando, const int timeout)
```

```
{  long int time = millis(); // medir el tiempo actual para verificar
timeout
   esp8266.print(comando); // enviar el comando al
ESP8266
   while( (time+timeout) > millis()) //mientras no haya
timeout
 {  while(esp8266.available()) //mientras haya datos por
leer
 {
 // Leer los datos disponibles
 char c = esp8266.read(); // leer el siguiente caracter
 Serial.print(c);
 }  }
return;
}
```

Se obtuvo el siguiente resultado por el monitor serial, donde se aprecia la correcta configuración y solución al problema encontrado con anterioridad:

Luego de esto se procederá a leer líneas de códigos (comandos) recibidos desde http hacia el ESP8266 hasta Arduino para ejecutar lo que necesitemos basando en los comandos recibidos por http.

Anexo 3:

Conectividad, envío de órdenes desde Python (Raspbian) a Arduino a través de HTTP.

Se instaló un editor de textos apropiado para Python con la siguiente línea de comando en la consola de la Raspberry:

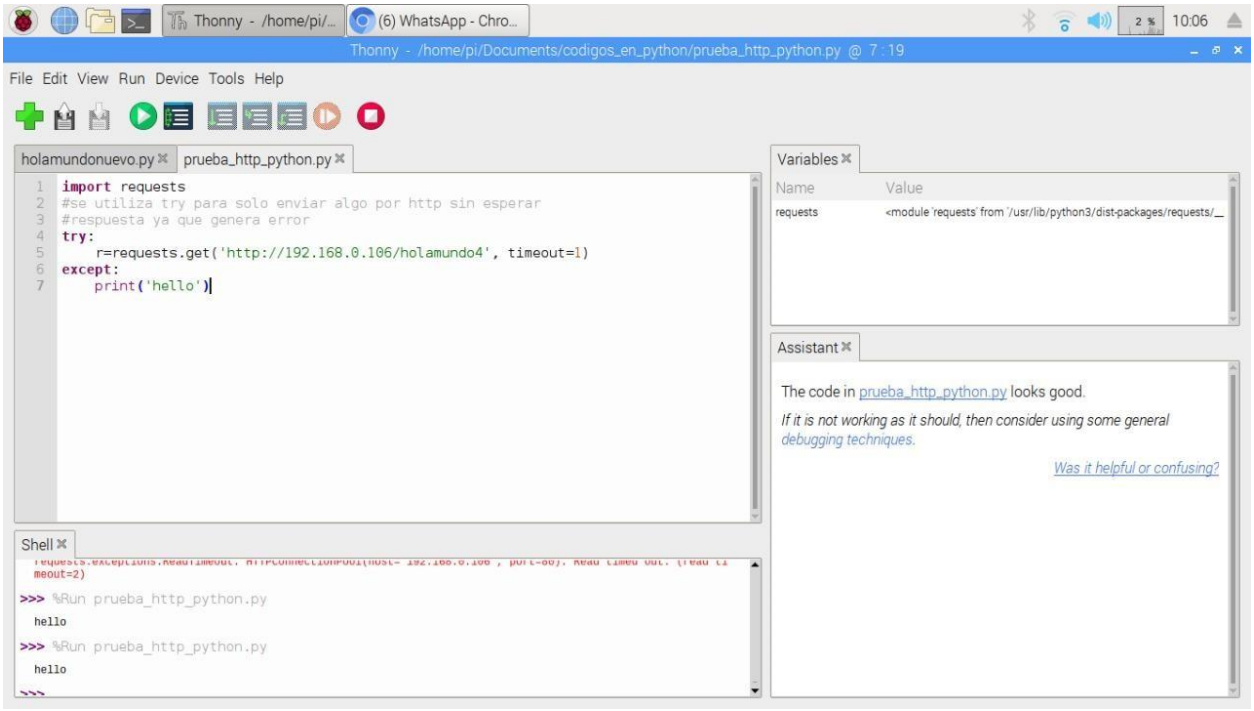
```
sudo apt-get install gedit
```

También se puede realizar mediante la aplicación nativa: Thonny Python IDE (de preferencia)

Prueba de conectividad entre Python y ESP8266, envío de texto.

Como prueba de conectividad entre Python de raspberry y el módulo ESP8266 se creó un código en Python y se lo ejecutó manualmente en Thonny Python IDE: (archivo: prueba_http_python.py) en raspberry

```
 import
requests
#se utiliza try para solo enviar algo por http sin
esperar #respuesta ya que genera error try:
    r=requests.get('http://192.168.0.106/holamundo4', timeout=1)
except:
    print('hello')
```



Se recibió en el módulo ESP8266:

Prueba de control entre Python y ESP8266: envío de comando desde Python para encendido de un led en placa de Arduino.

Al enviar desde Python la sentencia led=0 apaga el led y led=1 enciende el led. Archivo utilizado en Raspbian: prueba_http_enciendeled.py

```
import requests
#se utiliza try para solo enviar algo por http sin
esperar #respuesta ya que genera error try:
    r=requests.get('http://192.168.0.106/led=1', timeout=1)
except:
    print('hello')
```

Archivo utilizado en Arduino: Arduino_configuracion_encendido_led_http.ino

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266(3, 2); // RX | TX
 void
setup()
  {  Serial.begin(9600);
esp8266.begin(9600);
sendData("AT+RST\r\n",2000);      //
resetear módulo
```

```
sendData("AT+CIPMUX=1\r\n",1000); //
configurar para multiples conexiones
sendData("AT+CIPSERVER=1,80\r\n",1000);
// servidor en el puerto
80      pinMode(13,OUTPUT);     //inicializacion de led interno Arduino pin
13       digitalWrite(13,LOW);
  }  void
loop ()
{ if(esp8266.available())   // revisar si hay mensaje del
ESP8266
  {     delay(1500); // esperar que lleguen los datos hacia el buffer    int
conexionID = esp8266.read()-48; // obtener el ID de la conexión para poder
responder
   //Serial.print(String(conexionID));     esp8266.find("led="); // buscar el
texto "led="    int state = (esp8266.read()-48); // Obtener el estado del pin
a mostrar<  Al valor que lee del serial le restamos 48. Esto es debido a que
Arduino lee los caracteres en código ASCII, el cual podéis ver aquí, y en este
código el número cero equivale al valor 48    //Serial.print(String(state));
digitalWrite(13, state); // Cambiar estado del pin 13
   //while(esp8266.available())
    //{
    //char c = esp8266.read();
    //Serial.print(c);
    //}

 // comando para terminar conexión
    //String comandoCerrar = "AT+CIPCLOSE=";
    //comandoCerrar+=conexionID;
    //comandoCerrar+="\r\n";
    //sendData(comandoCerrar,3000);
 }
 }
}

//funcion sendData a través de serial void
sendData(String comando, const int timeout)
{  long int time = millis(); // medir el tiempo actual para verificar
timeout
   esp8266.print(comando); // enviar el comando al
ESP8266
   while( (time+timeout) > millis()) //mientras no haya
timeout
 {  while(esp8266.available()) //mientras haya datos por
leer
 {
 // Leer los datos disponibles
 char c = esp8266.read(); // leer el siguiente caracter
 Serial.print(c);
```

```
 }  }
return;
}
```

Prueba de control entre Python y ESP8266: envío de comando desde Python para encendido de varios leds en placa de Arduino.

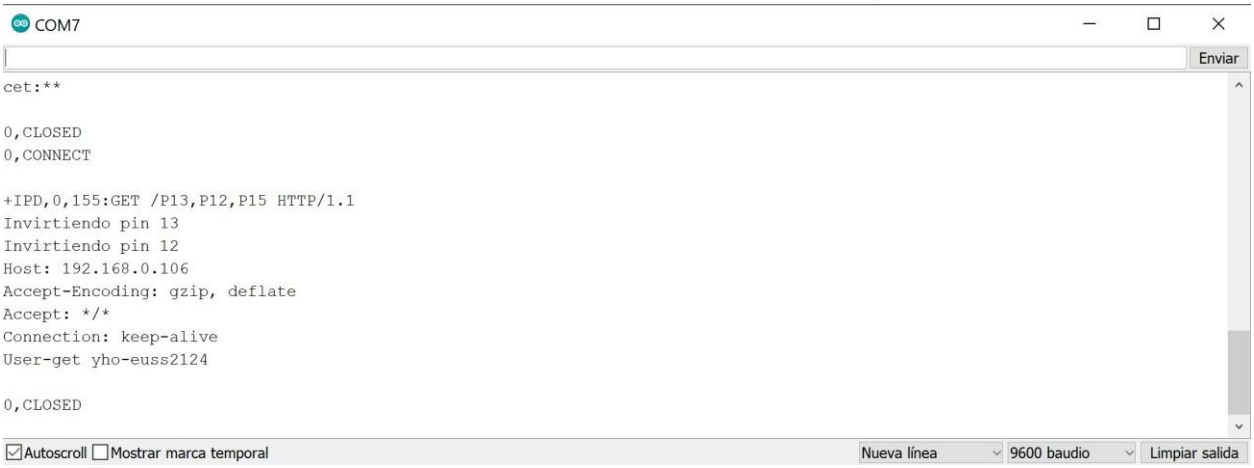
Al enviar desde Python la sentencia P13 y/o P12 invierte el estado del led correspondiente. Archivo utilizado en Raspbian: prueba_http_enciendevariosleds.py

```
import requests
#se utiliza try para solo enviar algo por http sin
esperar #respuesta ya que genera error try:
    r=requests.get('http://192.168.0.106/P13,P12', timeout=1)
except:
    print('hello')
```

Archivo utilizado en Arduino: Arduino_configuracion_cambiodeestados_http.ino

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266(3, 2); // RX |
TX String W =" "; char w ;  void
setup()
  {         Serial.begin(9600);                          esp8266.begin(9600);
sendData("AT+RST\r\n",2000);                         //    resetear    módulo
sendData("AT+CIPMUX=1\r\n",1000);  //  configurar  para  multiples  conexiones
sendData("AT+CIPSERVER=1,80\r\n",1000);         // servidor en el puerto
80      pinMode(13,OUTPUT);     //inicializacion de led interno Arduino pin
13       digitalWrite(13,LOW);
  }
 void
loop()
 {  if (esp8266.available())                // Lo que entra por WIFI à Serial
         { w = esp8266.read() ;
           Serial.print(w);
           W = W + w ;                     // Vamos montando un String con lo
que entra          }       if (Serial.available())             // Lo que entra
por Serial à WIFI
```

```
        {  char s = Serial.read();
esp8266.print(s);
        }       if ( w == '\n')                    // Sin han pulsado intro,
son los enters automáticos al final de cada línea recibida por wifi en serial
{ if ( W.indexOf("P13") > 0 )     // Comprobamos si P13 esta incluido en el
string enviado desde http desde python
              { digitalWrite( 13, !digitalRead(13)) ;
                Serial.println("Invirtiendo pin 13");
              }             if ( W.indexOf("P12") > 0 )    // Comprobamos si
P12 está incluido en el string enviado desde http desde Python
              { digitalWrite( 12, !digitalRead(12)) ;
                Serial.println("Invirtiendo pin 12");
              }
          W = "" ;  w = ' ' ;                    // Limpiamos las variables
        }
  }


//funcion sendData a través de serial void
sendData(String comando, const int timeout)
{  long int time = millis(); // medir el tiempo actual para verificar
timeout
   esp8266.print(comando); // enviar el comando al
ESP8266
   while( (time+timeout) > millis()) //mientras no haya
timeout
 {  while(esp8266.available()) //mientras haya datos por
leer
 {
 // Leer los datos disponibles
 char c = esp8266.read(); // leer el siguiente caracter
 Serial.print(c);
 }  }
return;
}
```


COM7
Enviar
cet:**
0,CLOSED
0,CONNECT
+IPD,0,155:GET /P13,P12,P15 HTTP/1.1
Invirtiendo pin 13
Invirtiendo pin 12
Host: 192.168.0.106
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
User-get yho-euss2124
0,CLOSED
Autoscroll
Mostrar marca temporal
Nueva línea
9600 baudio
Limpiar salida

Anexo 4:

Lectura de base de datos desde Python en localhost.

Recursos utilizados:

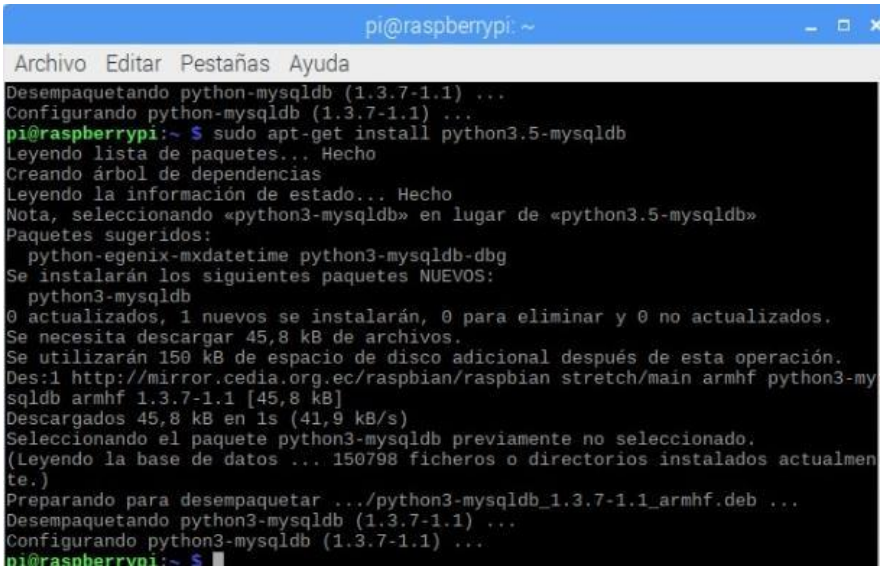
Thonny Python IDE 3.1.0 el cual tiene Python 3.5 instalado.

Desarrollo:

Instalación de la librería MySQLdb en Python.

La instalación se la realiza desde consola en las raspberry utilizando los siguientes comandos:

```
sudo apt-get install python3-mysqldb
 sudo apt-get install python3.5-
mysqldb
```



Acceso a la base de datos.

Para acceder a la base de datos se seguirá el siguiente procedimiento.

1.- Se abrirá la conexión.

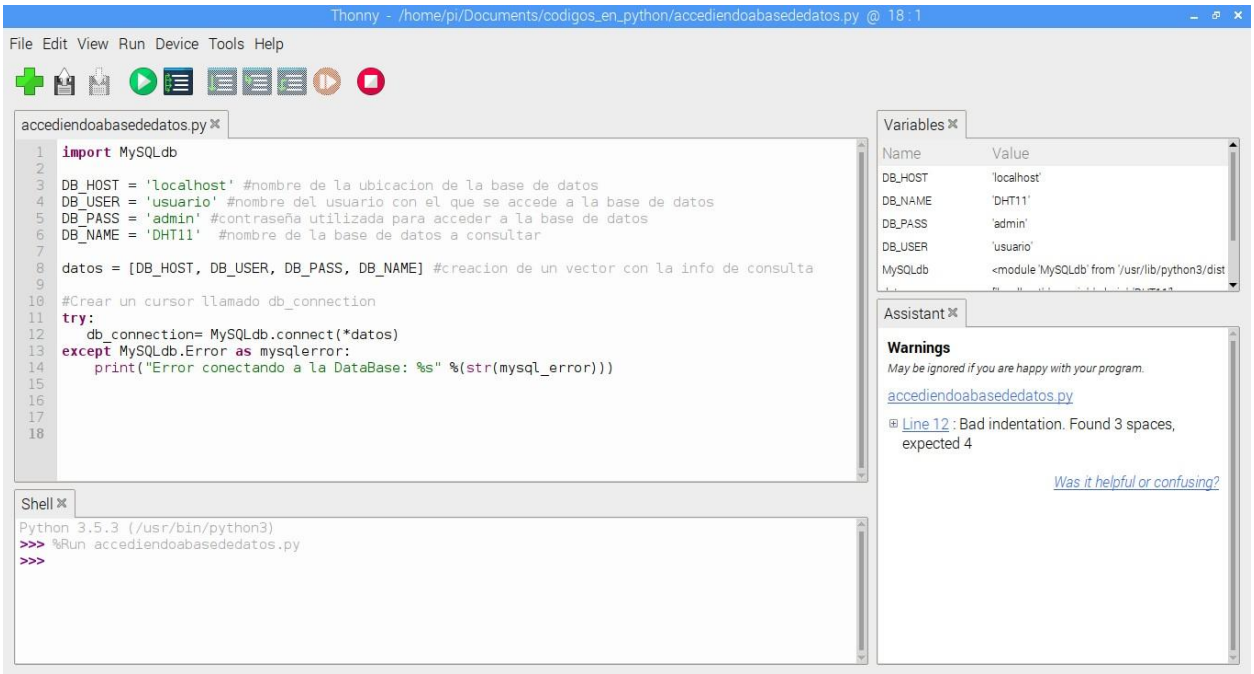
2.- Se creará un puntero el cual tendrá la información de la sección de la base de datos de interés.

3.- Se importarán los resultados desde la base de datos para lectura y consulta, y posterior a eso y de necesitarse se efectuará la escritura o sobreescritura correspondiente.

4.- Se cerrará el puntero y la conexión.

Prueba de conexión a la base de datos de prueba llamada DHT11

El archivo utilizado en Python fue el siguiente: accediendoabasededatos.py (alojado en la Raspberry)



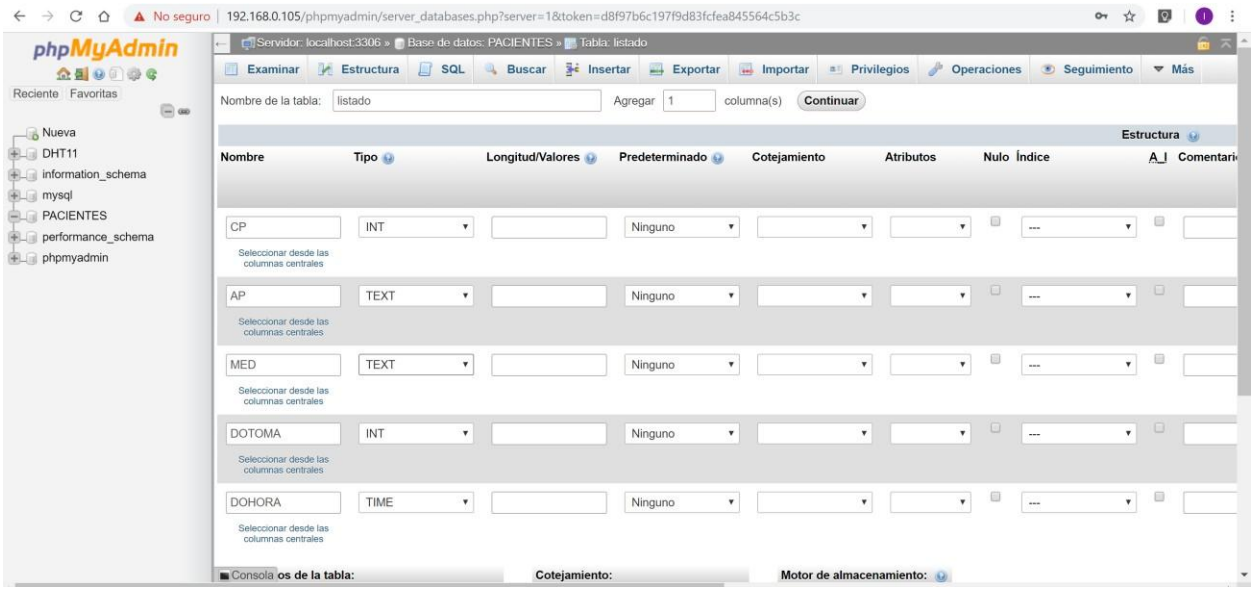
Creación de tabla en base de datos para los dos pacientes en prueba.

El modelo de la tabla de dosis a tomar de los pacientes es el siguiente:

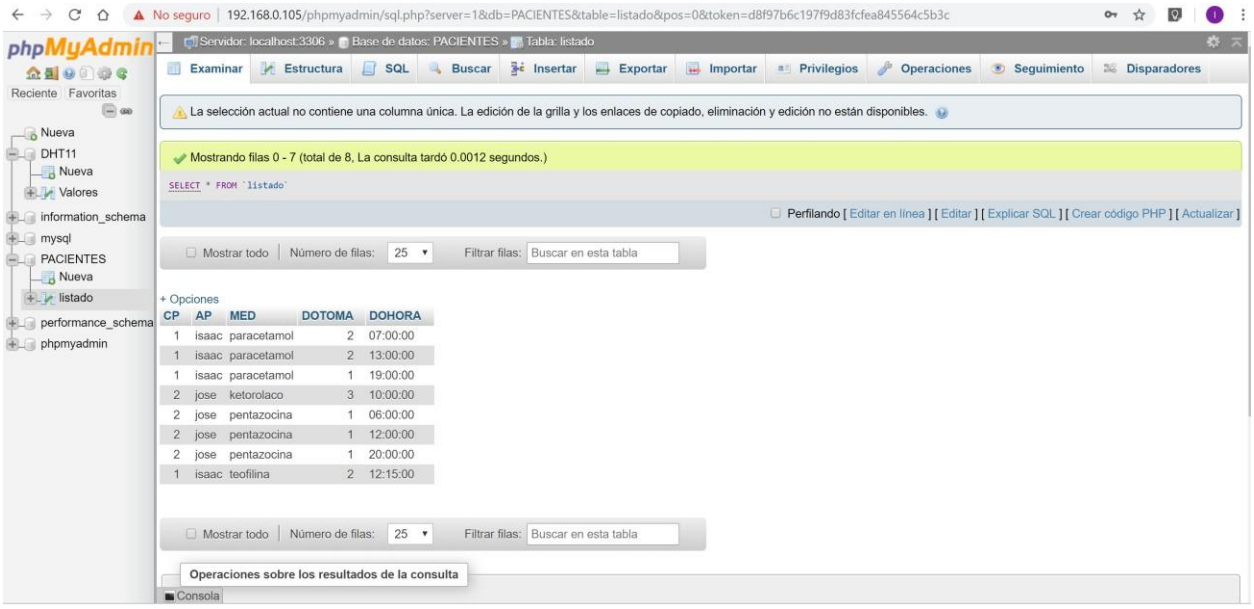
CP **Código de paciente**	**AP** **Alias paciente**	**MED** **Medicamento**	**DOTOMA** **Dosis por toma**	**DOHORA** **Dosis por hora**
1	isaac	paracetamol	2	07H00
1	isaac	paracetamol	2	13H00
1	isaac	paracetamol	1	19H00
2	jose	ketorolaco	3	10H00
2	jose	pentazocina	1	06H00
2	jose	pentazocina	1	12H00

2	jose	pentazocina	1	20H00
1	isaac	Teofilina	2	12H15

Creación de base de datos PACIENTES en tabla listado



Llenado de tabla manualmente.



Consulta de información guardada en la tabla PACIENTES en la tabla listado.

Código utilizado en Python.

```
import MySQLdb

DB_HOST = 'localhost' #nombre de la ubicación de la base de datos
DB_USER = 'usuario' #nombre del usuario con el que se accede a la base de
datos
DB_PASS = 'admin' #contraseña utilizada para acceder a la base de datos
DB_NAME = 'PACIENTES'  #nombre de la base de datos a consultar
 datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME] #creación de un vector con la
info de consulta

#Crear un cursor llamado db_connection
try:    db_connection=
MySQLdb.connect(*datos)
        except MySQLdb.Error as
mysqlerror:
    print("Error conectando a la DataBase: %s" %(str(mysql_error)))
 cursor =
db_connection.cursor()

#Selección y muestreo de todos los datos de la tabla listado
 sql_select_Query="SELECT * from
listado" cursor =
db_connection.cursor()
cursor.execute(sql_select_Query)
records=cursor.fetchall()
 print("El Número total de columnas en listado es - ",
cursor.rowcount) print("Imprimiendo cada valor de las columnas en
listado") for row in records:
    print("Código de paciente = ",row[0], )
print("Nombre de paciente = ",row[1], )
print("Medicamento = ",row[2], )
print("Dosis por toma = ",row[3], )
print("Hora de la dosis = ",row[4], "\n" )
cursor.close()
 db_connection.close()  #desconexión del
servidor print("MySQL connection finalizada")
```

Al ejecutar el código genera por pantalla los siguientes resultados:

```
>>> %Run accediendobasededatospacientes.py

  El Numero total de columnas en listado es -  8
  Imprimiendo cada valor de las columnas en listado
  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  2
  Hora de la dosis =  7:00:00

  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  2
  Hora de la dosis =  13:00:00

  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  1
  Hora de la dosis =  19:00:00

  Código de paciente =  2
  Nombre de paciente =  jose
  Medicamento =  ketorolaco
  Dosis por toma =  3
  Hora de la dosis =  10:00:00

  Código de paciente =  2
  Nombre de paciente =  jose
```

Código utilizado en Python organizado con funciones.

```
import MySQLdb

DB_HOST = 'localhost' #nombre de la ubicación de la base de datos
DB_USER = 'usuario' #nombre del usuario con el que se accede a la base de
datos
DB_PASS = 'admin' #contraseña utilizada para acceder a la base de datos
DB_NAME = 'PACIENTES'  #nombre de la base de datos a consultar
 datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME] #creación de un vector con la
info de consulta

#####################################################################
#DEFINICION DE FUNCIONES

#Funcion que se conecta a la base de datos y obtiene toda su info guardada
 def conectar_DB_all_data(): #genera la info en records y el
cursor
    #Crear un cursor llamado db_connection
try:        db_connection=
MySQLdb.connect(*datos)
                    except
MySQLdb.Error as mysqlerror:
        print("Error conectando a la DataBase: %s" %(str(mysql_error)))
     cursor =
db_connection.cursor()

#Selección y muestreo de todos los datos de la tabla listado
sql_select_Query="SELECT * from listado"     cursor = db_connection.cursor()
cursor.execute(sql_select_Query)     records=cursor.fetchall()
return(records,cursor)
```

```
#Imprime cada uno de los datos de la base de datos listado
def imprime_all_data(records,cursor):
         print("El Numero total de columnas en listado es - ",
cursor.rowcount)     print("Imprimiendo cada valor de las columnas en
listado")     for row in records:
        print("Código de paciente = ",row[0], )
print("Nombre de paciente = ",row[1], )
print("Medicamento = ",row[2], )
print("Dosis por toma = ",row[3], )
print("Hora de la dosis = ",row[4], "\n" )
cursor.close()

#print (records)
#db_connection.close()  #desconexión del servidor

#print("MySQL connection finalizada")


#########################################################################
#MAIN CODE
records=conectar_DB_all_data()[0]
cursor=conectar_DB_all_data()[1]
imprime_all_data(records,cursor)
```

Anexo 5:

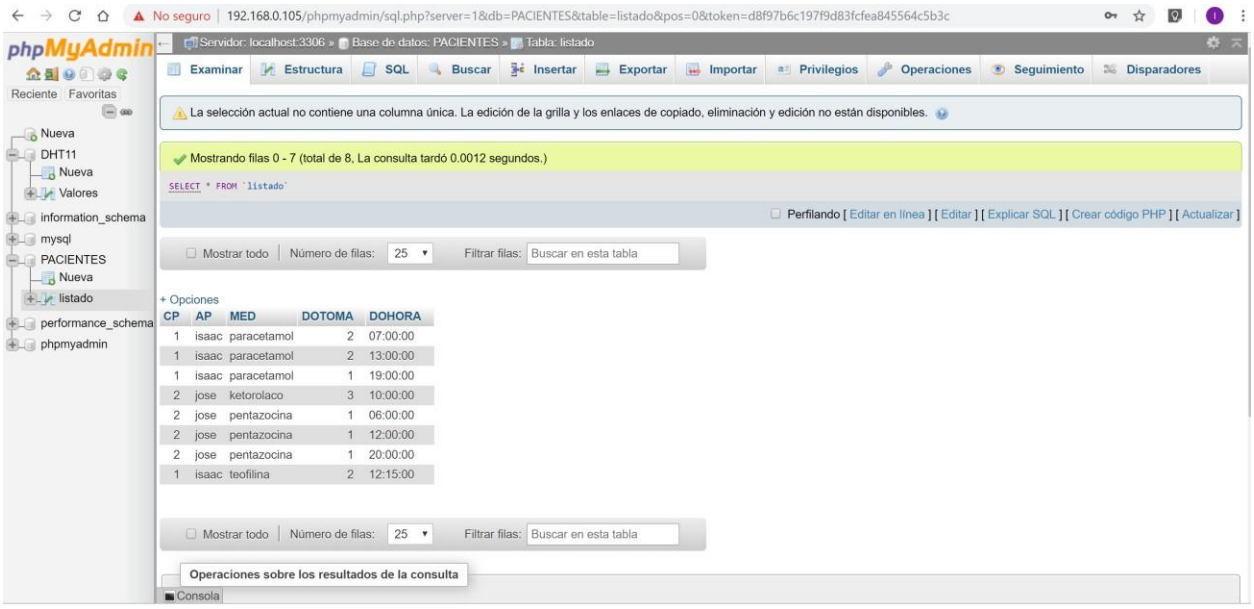
Envío de señales desde Python hacia Arduino según horarios de pastillas.

Desarrollo:

El modelo de la tabla de dosis a tomar de los pacientes es el siguiente:

CP Código de paciente	**AP Alias paciente**	**MED Medicamento**	**DOTOMA Dosis por toma**	**DOHORA Dosis por hora**
1	isaac	paracetamol	2	07H00

1	isaac	paracetamol	2	13H00
1	isaac	paracetamol	1	19H00
2	jose	ketorolaco	3	10H00
2	jose	pentazocina	1	06H00
2	jose	pentazocina	1	12H00
2	jose	pentazocina	1	20H00
1	isaac	Teofilina	2	12H15



Consulta de información guardada en la tabla PACIENTES en la tabla listado.

Código utilizado en Python (solo consulta).

Al ejecutar el código de consulta obtenido en la fase 4 genera por pantalla los siguientes resultados:

```
>>> %Run accediendobasededatospacientes.py

  El Numero total de columnas en listado es -  8
  Imprimiendo cada valor de las columnas en listado
  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  2
  Hora de la dosis =  7:00:00

  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  2
  Hora de la dosis =  13:00:00

  Código de paciente =  1
  Nombre de paciente =  isaac
  Medicamento =  paracetamol
  Dosis por toma =  1
  Hora de la dosis =  19:00:00

  Código de paciente =  2
  Nombre de paciente =  jose
  Medicamento =  ketorolaco
  Dosis por toma =  3
  Hora de la dosis =  10:00:00

  Código de paciente =  2
  Nombre de paciente =  jose
```

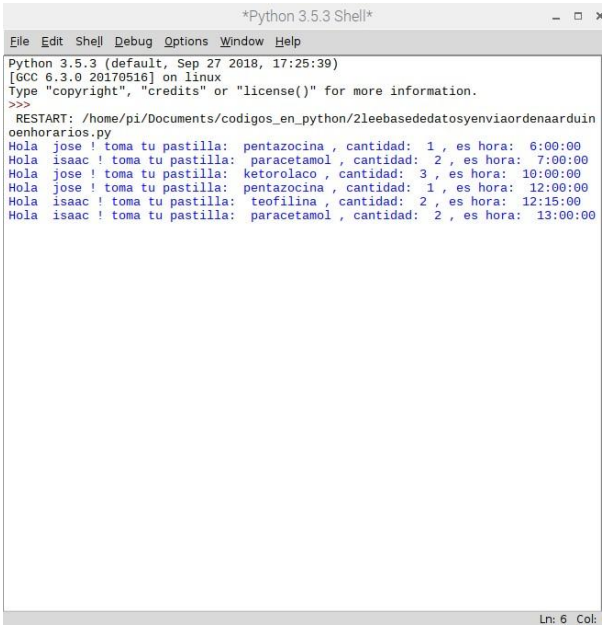
Código utilizado en Python para consulta en la base de datos, comprobación con hora actual y muestreo por pantalla las órdenes de toma de pastillas de acuerdo a la hora actual del sistema (RASPBIAN).

```
import        MySQLdb import       time

DB_HOST = 'localhost' #nombre de la ubicación de la base de datos
DB_USER = 'usuario' #nombre del usuario con el que se accede a la base de datos
DB_PASS = 'admin' #contraseña utilizada para acceder a la base de datos
DB_NAME = 'PACIENTES'  #nombre de la base de datos a consultar

datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME] #creación de un vector con la info de
consulta

#################################################################
#DEFINICION                                  DE                                  FUNCIONES

#Funcion que se conecta a la base de datos y obtiene toda su info guardada def
conectar_DB_all_data(): #genera la info en records y el cursor      #Crear un cursor
llamado db_connection     try:
       db_connection=                                          MySQLdb.connect(*datos)


    except MySQLdb.Error as mysqlerror:          print("Error conectando a la DataBase:
%s" %(str(mysql_error)))

    cursor                              =                              db_connection.cursor()

#Selección y muestreo de todos los datos de la tabla listado
```

```
    sql_select_Query="SELECT * from listado"        cursor = db_connection.cursor()
cursor.execute(sql_select_Query)                        records=cursor.fetchall()
return(records,cursor)

#Imprime cada uno de los datos de la base de datos listado def
imprime_all_data(records,cursor):

    print("El Numero total de columnas en listado es - ", cursor.rowcount)
print("Imprimiendo cada valor de las columnas en listado")
    for                         row                         in                          records:
        print("Código         de         paciente          =          ",row[0],          )
        print("Nombre         de         paciente          =          ",row[1],          )
        print("Medicamento                 =                ",row[2],                  )
        print("Dosis          por         toma          =          ",row[3],          )
        print("Hora               la       dosis       =       ",row[4],       "\n"       )
      de
cursor.close()
#print                                                                   (records)
#db_connection.close()                  #desconexión          del           servidor

#print("MySQL                          connection                          finalizada")


#Comparador de hora, solo compara hora y minuto, ingresa la hora a comparar, #la compara
con la hora del sistema y genera TRUE en caso de igualdad o FALSE caso contrario
#hora_DB     tiene  el     formato      HH:MM:SS def compara_hora(hora_DB):
igualdad_de_hora=False
    time_actual=time.strftime("%H:%M")
    #print(hora_DB)
    #print(hora_DB[1]==str(":")) #corregir el problema de agregación del 0     if
hora_DB[1]==str(":"):  #asigna un cero al inicio de la hora en el caso de ser de
un                                                                          digito
        hora_DB="0"+hora_DB
    if                                                  time_actual==hora_DB[0:5]:
        #print("Las                 horas                 son                  iguales")
igualdad_de_hora=True
    #else:
        #print("no                              son                             iguales")
    #print(time_actual)
    #print(hora_DB[0:5])
    return(igualdad_de_hora)

#########################################################################
#MAIN                                                                    CODE


records=conectar_DB_all_data()[0]
cursor=conectar_DB_all_data()[1]
#imprime_all_data(records,cursor)


#print(str(records[0][-1]))
#print                                                   (time.strftime("%H:%M"))
#print(compara_hora("21:20:12"))
#print(compara_hora(str(records[0][-1])))  #obtiene la primera fila del DB pero solo
la hora      (-1) hora_ultima_pastilla_alertada="52:90" while(True):
    records=conectar_DB_all_data()[0]       cursor=conectar_DB_all_data()[1]       for
elemento   in   records:                    if   compara_hora(str(elemento[-1]))   and
hora_ultima_pastilla_alertada!=time.strftime("%H:%M"):
```

```
            if      elemento[0]==1:           #paciente    1      en     este   caso
ISAAC                  #print("Pastillas       de:    ",elemento[1])
#print(elemento[1:3],     elemento[-1])                  print("Hola ",elemento[1],"!
toma tu pastilla: ",elemento[2],", cantidad:  ",elemento[3],",    es     hora:
",elemento[-1])             if elemento[0]==2:  #paciente 2 en este caso JOSE
#print("Pastillas de: ",elemento[1])                  #print(elemento[1:3], elemento[-
1])                 print("Hola ",elemento[1],"! toma tu pastilla: ",elemento[2],",
cantidad:          ",elemento[3],",          es          hora:          ",elemento[-1])
            hora_ultima_pastilla_alertada=time.strftime("%H:%M") #guarda la hora
actual de laultima pastillatomada para compararla con la hora de la nueva alerta, si
son iguales entonces significa que ya mostrolaalerta una vez y no la vuelve a mostrar
time.sleep(5)
```

El código mostrado, tiene un correcto funcionamiento, ya que luego de horas de prueba en modo autónomo mostró correctamente las alertas de medicinas por pantalla desde el mismo SHELL de Python, tal como se esperaba.



Se realiza la modificación el código cargado en el microcontrolador:

```
#include <SoftwareSerial.h>

SoftwareSerial esp8266(3, 2); // RX |

TX String W =" "; char w ;

String mensaje="";
```

```
void setup()
  {          Serial.begin(9600);                              esp8266.begin(9600);
sendData("AT+RST\r\n",2000);                            //    resetear    módulo
sendData("AT+CIPMUX=1\r\n",1000);  //  configurar  para  multiples  conexiones
sendData("AT+CIPSERVER=1,80\r\n",1000);           // servidor en el puerto 80
pinMode(13,OUTPUT);         //inicializacion  de  led  interno  Arduino  pin  13
digitalWrite(13,LOW);
  }  void
loop()
 {  if (esp8266.available())               // Lo que entra por WIFI à Serial
         { w = esp8266.read() ;
           //Serial.print(w);
           W = W + w ;                     // Vamos montando un String con lo que entra
         }       if (Serial.available())             // Lo que entra por
Serial à WIFI
         {  char s = Serial.read();
esp8266.print(s);
         }
      if ( w == '\n')                     // Sin han pulsado intro, son los enters
automáticos al final de cada línea recibida por wifi en serial
         { if ( W.indexOf("Hola") > 0 )    // Comprobamos si P13 esta incluido en el
string enviado desde http desde python
               { digitalWrite( 13, !digitalRead(13)) ;
mensaje=W.substring(W.indexOf("Hola"),W.indexOf("#"));
mensaje.replace("%20"," ");
mensaje.replace("HTTP/1.1"," ");
                 Serial.println(mensaje);
                 Serial.println("");
}
           if ( W.indexOf("P13") > 0 )    // Comprobamos si P12 esta incluido en el
string enviado desde http desde python
               { digitalWrite( 13, !digitalRead(13)) ;
                 Serial.println("Invirtiendo pin 13");
               }
           W = "" ;  w = ' ' ;                   // Limpiamos las variables
```

```
        }
  }


//funcion sendData a través de serial void
sendData(String comando, const int timeout)
{  long int time = millis(); // medir el tiempo actual para verificar
timeout
   esp8266.print(comando); // enviar el comando al
ESP8266
   while( (time+timeout) > millis()) //mientras no haya
timeout
 {  while(esp8266.available()) //mientras haya datos por
leer
 {
 // Leer los datos disponibles  char c =
esp8266.read(); // leer el siguiente caracter
 Serial.print(c);
 }  }
return
;
}
```

Se realiza la modificación el código en Python para envío a través de http:

```
import MySQLdb
import time
import requests

DB_HOST = 'localhost' #nombre de la ubicación de la base de datos
DB_USER = 'usuario' #nombre del usuario con el que se accede a la base de datos
DB_PASS = 'admin' #contraseña utilizada para acceder a la base de datos
DB_NAME = 'PACIENTES'  #nombre de la base de datos a consultar

datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME] #creación de un vector con la info de
consulta

##################################################################
```

```
#DEFINICION DE FUNCIONES

#Funcion que se conecta a la base de datos y obtiene toda su info guardada
def conectar_DB_all_data(): #genera la info en records y el cursor
    #Crear un cursor llamado db_connection
try:        db_connection=
MySQLdb.connect(*datos)
                    except
MySQLdb.Error as mysqlerror:
        print("Error conectando a la DataBase: %s" %(str(mysql_error)))

    cursor = db_connection.cursor()

#Seleccion y muestreo de todos los datos de la tabla listado

    sql_select_Query="SELECT * from listado"
cursor = db_connection.cursor()
cursor.execute(sql_select_Query)
records=cursor.fetchall()
return(records,cursor)

#Imprime cada uno de los datos de la base de datos listado
def imprime_all_data(records,cursor):

    print("El Numero total de columnas en listado es - ",
cursor.rowcount)     print("Imprimiendo cada valor de las columnas en
listado")     for row in records:
        print("Código de paciente = ",row[0], )
print("Nombre de paciente = ",row[1], )
print("Medicamento = ",row[2], )
print("Dosis por toma = ",row[3], )
print("Hora de la dosis = ",row[4], "\n" )
cursor.close()

#print (records)
#db_connection.close()  #desconexión del servidor

#print("MySQL connection finalizada")

#Comparador de hora, solo compara hora y minuto, ingresa la hora a comparar, #la
compara con la hora del sistema y genera TRUE en caso de igualdad o FALSE caso
contrario
#hora_DB tiene el formato HH:MM:SS
def compara_hora(hora_DB):
igualdad_de_hora=False
    time_actual=time.strftime("%H:%M")
    #print(hora_DB)
    #print(hora_DB[1]==str(":")) #corregir el problema de agregación del 0      if
hora_DB[1]==str(":"):  #asigna un cero al inicio de la hora en el caso de ser de un
digito
        hora_DB="0"+hora_DB     if
time_actual==hora_DB[0:5]:
#print("Las horas son iguales")
igualdad_de_hora=True
    #else:
        #print("no son iguales")
```

```
    #print(time_actual)
#print(hora_DB[0:5])
return(igualdad_de_hora)

#########################################################################
#MAIN CODE

records=conectar_DB_all_data()[0]
cursor=conectar_DB_all_data()[1]
#imprime_all_data(records,cursor)

#print(str(records[0][-1]))
#print  (time.strftime("%H:%M"))
#print(compara_hora("21:20:12"))
#print(compara_hora(str(records[0][-1])))  #obtiene la primera fila del DB pero solo
la hora (-1)
hora_ultima_pastilla_alertada="52:90"
while(True):
    records=conectar_DB_all_data()[0]
cursor=conectar_DB_all_data()[1]
for elemento in records:
        if compara_hora(str(elemento[-1])) and
hora_ultima_pastilla_alertada!=time.strftime("%H:%M"):
if elemento[0]==1:  #paciente 1 en este caso ISAAC
                #print("Pastillas de: ",elemento[1])
#print(elemento[1:3], elemento[-1])
                print("Hola ",elemento[1],"! toma tu pastilla: ",elemento[2],",
cantidad: ",elemento[3],", es hora: ",elemento[-1])
mensaje_a_enviar_http="http://192.168.0.106/Hola "+str(elemento[1])+"! Toma tu
pastilla: "+str(elemento[2])+", cantidad: "+str(elemento[3])+", es hora:
"+str(elemento[-1])+"#"
try:
                    r=requests.get(mensaje_a_enviar_http, timeout=1)
except:
                    print(" ")


            if elemento[0]==2:  #paciente 2 en este caso JOSE
                #print("Pastillas de: ",elemento[1])
#print(elemento[1:3], elemento[-1])
                print("Hola ",elemento[1],"! toma tu pastilla: ",elemento[2],",
cantidad: ",elemento[3],", es hora: ",elemento[-1])
mensaje_a_enviar_http="http://192.168.0.106/Hola "+str(elemento[1])+"! Toma tu
pastilla: "+str(elemento[2])+", cantidad: "+str(elemento[3])+", es hora:
"+str(elemento[-1])+"#"
try:
                    r=requests.get(mensaje_a_enviar_http, timeout=1)
except:
                    print(" ")
            hora_ultima_pastilla_alertada=time.strftime("%H:%M") #guarda la hora
actual de laultima pastillatomada para compararla con la hora de la nueva alerta, si
son iguales entonces significa que ya mostrolaalerta una vez y no la vuelve a mostrar
time.sleep(5)
```

En la cual se comprueba el correcto funcionamiento de la recepción de órdenes en el microcontrolador a través del monitor serial, imprimiendo las órdenes que recibe. En este caso solo es la impresión de la orden, sin embargo, puede adaptarse el código para realizar cualquier tarea que se desee.