UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN – León

Facultad de Ciencias y Tecnología

Sistema de detección de movimiento para uso residencial, con notificación a móviles, utilizando el microcomputador Raspberry PI.

Tesis para optar al título de
INGENIERO EN TELEMATICA

Presentado por:

Br. Omar Octavio Zapata Romero.

Br. Darwin Raúl Rivera Zeas.

Tutor:

Msc. Aldo Rene Martinez.

Julio 2016

RESUMEN

El internet de las cosas, Realidad Aumentada, Servicios de nube, Domótica, Desarrollo multiplataforma, son sólo algunas de los miles de tecnologías que están causando revuelo y haciendo historia en la actualidad en la que vivimos, nuevos dispositivos, aplicaciones, términos como “wereables” y los pasos agigantados en los que la tecnología avanza, es cada vez es más increíble. Día a día nuestro existir se torna más complejo, el hombre siempre ha buscado la manera de simplificar o solucionar las cosas de la forma más sencilla posible, pero esto por otro lado implica que algún entusiasta se siente a programar una aplicación, diseñar un producto, un componente electrónico o un algoritmo por muchas horas seguidas. Y no solo esto, deben de existir plataformas, centros de datos, Internet, en fin, un sin número de personas creando cosas complejas para que nuestra vida sea más sencilla.

Es por eso que nos hemos decidido a realizar este trabajo con el propósito de contribuir de manera objetiva con el desarrollo de tecnologías libres de patentes y no privativas, el cual consiste en un sistema de detección de movimiento para uso residencial de bajo coste y poco consumo energético. En dicho trabajo incluimos un microcomputador Raspberry Pi B+ como controlador de sensores, presentaremos un prototipo que obtendrá datos de un sensor infrarrojo pasivo (PIR) y los transmitirá a un servidor que se encuentra alojado en una plataforma de Nube Microsoft Windows Azure, el cual tiene un servicio LAMP (Linux Apache MySql PHP) configurado y que en conjunto con el servicio gratuito GCM (Google Cloud Messaging) perteneciente a la compañía Alphabet, enviarán un mensaje de alerta a un dispositivo en específico con sistema operativo Android.

Dicho proyecto contribuirá al posterior estudio y desarrollo de diversas aplicaciones, en los diferentes campos que este documento incluye, con la oportunidad de hacer de este un sistema más inteligente, que solventaría múltiples necesidades poco a poco, debido a su fácil escalabilidad.

Dedicatoria

A nuestros Padres y Maestros, de no ser por ellos no hubiese sido posible este milagro.

Agradecimiento

A Dios y a nuestros amigos que siempre estuvieron dándonos ánimos para continuar a pesar de todas las dificultades.

INDICE

INTRODUCCIÓN

Cada día se hace más grande la cantidad de tecnología desarrollada alrededor nuestro, es imposible no percatarse de todo lo que está ocurriendo en el mundo, los deseos de aprender sobre cada una de estas tecnologías, es cada vez más grande, tecnologías como Servicios de nube publica, desarrollo de aplicaciones para móviles, domótica con hardware libre y el uso de Software Libre, son cada vez más importantes para el desarrollo de la humanidad y el conocimiento.

Estas tecnologías son de código abierto pero no significa que sean solo gratuitas, o de libre acceso, significa que todos y cada uno de nosotros podemos aprender de ellas y con ellas, tomar conocimiento compartido, modificarlo y compartido nuevamente, encontrando más de una sola solución a un problema y tapando así todos los huecos que podrían entorpecer nuestros propósitos finales, dichas tecnologías nos dejan conocerlas y estudiarlas en su totalidad, lo que les hace fáciles de aprender y contribuyen en gran escala al desarrollo del todo tecnológico.

PLANTEAMIENTO DEL PROBLEMA

En Nicaragua el problema de la seguridad electrónica residencial, es un hecho y el bajo acceso a la información Tecnológica hacen que nos enfrasquemos en un cuello de botella respecto a soluciones que podríamos utilizar. Es un poco evidente que las opciones en nuestro país son limitadas, un gran ejemplo son las pocas opciones que poseemos en cuanto a proveedores de servicios que se han tornado básicos, como el internet o el servicio telefónico. El poco acceso a la información tecnológica es muchas veces el causante de que el progreso se estanque, esto no sólo sucede en nuestro país sino a nivel mundial, por otro lado en Nicaragua existen un grupo de comunidades que van creciendo poco a poco en pro de diferentes tecnologías, entre ellas las tecnologías que se desarrollaran en este documento, buscando así soluciones más viables no solo para el bolsillo del Nicaragüense sino también para contribuir en el desarrollo de nuestro país y el mejoramiento de la calidad de vida.

Los servicios de detección de intrusos que actualmente se ofrecen en nuestro país se encuentran sobrevaluados en comparación con sus homónimos en otros países de Latinoamérica y el mundo. Estos sistemas se contratan y se pagan en mensualidades a un alto costo dentro de Nicaragua.

Lo que se propone es una solución Tecnológica Libre con funciones de un sistema de detección de intrusos, el cual tiene amplias oportunidades de crecimiento como producto, un bajo costo de implementación y una alta contribución a que nuestra Nicaragua sea un mejor lugar, el sistema de detección, consume muy poca energía y no requiere de mensualidad alguna más que del servicio de internet ya contratado, al momento de la detección de movimiento se generara una alerta que se mostrara como notificación Push en un teléfono celular con sistema operativo Android.

JUSTIFICACION

La realización de este proyecto fue con la intención de proporcionar una alternativa de bajo costo en la implementación de un sistema de detección de intrusos en un hogar o negocio, por medio de sensores de movimiento conectados a una Raspberry Pi B+, la cual permite enviar notificaciones al dispositivo móvil del dueño del local al momento de detectar movimiento en el rango de detección de los sensores.

También pretendemos utilizar hardware libre no solo por su bajo costo de implementación, sino porque también permiten realizar diversas mejoras que en un futuro ampliará los métodos de detección de intrusos y automatizará diferentes servicios, todo esto para crear un sistema de seguridad eficiente y confiable.

Originalidad:

Este tema agrupa el estudio de diversas tecnologías de código abierto y a su vez integra servicios de nube que son la vanguardia y el siguiente paso de la informática que actualmente está en auge y desarrollo.

Alcance:

El Sensor de movimiento PIR que conectado a la placa Raspberry Pi B+, detecta movimientos en un rango de 5 - 8 metros y que mediante un script envía notificaciones push a móviles con sistema operativo Android.

Producto:

Se entregará un sistema de detección de movimiento que permita vigilar nuestro hogar con notificaciones remotas en tiempo real.

Impacto:

Con la realización de esta práctica investigativa se pretende aportar conocimiento a las tecnologías libres y principalmente incentivar tanto a informáticos como a no informáticos a incursionar en tecnologías de este tipo.

ANTECEDENTES

Podemos encontrar una cantidad increíble de proyectos de detección de intrusos que podemos encontrar por todo el Internet usando Raspberry Pi y diferentes sensores. En el sitio oficial de Raspberry Pi existe una comunidad de Hackers encargados de recopilar todos los proyectos, permitiendo a entusiastas de estas tecnologías a replicarlos y mejorarlos.

En Estados Unidos

PrivateEyePi - Julio 2013

PrivateEyePi es sistema de alarmas configurable, automatizable y sobre todo de código abierto que puedes construir tú mismo. Su creador solamente identificado por "Gadjet Nut" quien documentó minuciosamente el sistema completo y proveyó de una lista de partes incluyendo precios, esquemas de conexión del circuito y todo el código necesario para la realización exitosa del sistema, puedes usar sensores de movimiento, switches armados en puertas o una mezcla de ambos. Existen instrucciones para añadir cámaras a la configuración e incluso se le puede agregar sensores de temperatura para verificar que el sistema de calefacción esté funcionando correctamente mientras no estás en casa y ser capaz de monitorear todo desde una computadora o teléfono celular.

Las notificaciones se efectúan mediante correo electrónico y también cuenta con un sitio web donde puedes entrar a monitorear los datos.

En España y Latinoamérica

SecurOS – agosto 2014

SecurOS es un sistema de seguridad creado por @fpaez (Francisco Páez) un entusiasta de la electrónica de origen español, La idea principal de Páez es montar una cámara de seguridad con Raspberry Pi activada por un sensor de movimiento HC-SR501 y que pueda capturar fotografías cuando detecte movimiento gracias al módulo de cámara Pi NoIR.

Las imágenes que son tomadas con la cámara pueden ser consultadas de manera remota desde un móvil o computador a través de un sitio web, este módulo de cámara que se ocupa en este proyecto es ideal para ambiente con muy poca luz o ninguna luz, y es ideal para monitorear el paso de personas por un área determinada o controlar movimientos de una mascota.

Proyecto Acompaña – septiembre 2014

Desarrollado por el Dr. Bernardo Alarcos en la Universidad de Alcalá como una cátedra, el proyecto acompaña fue creado para mejorar la autonomía de las personas mayores alargando el tiempo en el que pueden vivir en su hogar de forma razonable y segura, haciendo supervisión no intrusiva de su actividad en el hogar, ayudando en la detección temprana de problemas y prevención de los mismo. Basado en una serie de sensores distribuidos e instalados en toda la casa de habitación el proyecto acompaña es uno de los más completos que encontramos

Nicaragua

Proyectos relacionados con la seguridad domiciliar en nuestro país podemos mencionar la empresa Claro Nicaragua la cual ofrece un servicio de un sistema de vigilancia orientado a las pequeñas y medianas empresas en el país, este sistema garantiza principalmente la seguridad de instalaciones, equipos y mercadería mediante un monitoreo en vivo las 24 horas del día, los siete días de la semana.

Este sistema funciona con equipos de grabación, transmisión y almacenamiento de imágenes que se puede monitorear a través del internet empresarial de claro desde cualquier lugar en cualquier dispositivo con conectividad a internet.

Los equipos de vigilancia involucrados en este servicio deben ser comprados, pueden ser pagados en plazos y disponen de un completo soporte técnico telefónico y presencial para resolver cualquier duda sobre el funcionamiento del mismo, los clientes empresariales pueden adquirir este servicio por una renta mensual desde 66.99 dólares más impuestos al valor agregado (IVA).

En nuestro país se han realizado muy pocos proyectos utilizando tecnologías hardware libre, aunque existen diversas comunidades comprometidas para que esto cambie en los próximos años, así como lo son la comunidad Fedora Nicaragua, Debian Nicaragua incluso existe una lista de correos para una creciente comunidad de Raspberry Pi.

Estas comunidades han estado realizando actividades de diferentes índoles desde Defcon, Fudcon, Flisol, otras ferias relacionadas con tecnologías de código abierto y reuniones casuales para compartir sobre experiencias en este mundo de la tecnología no privativa.

En el 2014 se impartió el primer curso de hardware libre involucrando Raspberry Pi Modelo B y Arduino Uno en el Fudcon (Fedora User and Developers Conference) Managua 2014 realizado en La Universidad de Ciencias Comerciales UCC Managua. En este taller se aprendieron aspectos básicos de estas tecnologías y cómo empezar a manipularlas.

Algunos proyectos que nos gustaría mencionar son el proyecto de "**Control de flujo energético**" Realizado por estudiantes de la Universidad Nacional Autónoma de Nicaragua UNAN- León a manos de los bachilleres Hellen Solís, Roberto Barreto y José Lainez, realizado en enero 2016. Y el proyecto "**Modelación de un sistema automatizado para la administración del invernadero del campus agropecuario de la UNAN-León**" por el Lic. Arnoldo Contreras y el Lic. Jairo Martínez, realizado en septiembre del 2015. Ambos proyectos realizados en nuestra alma mater y con el mismo propósito que sería la automatización de servicios.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar un sistema de detección de movimientos de bajo coste, implementado con el microcomputador Raspberry Pi B+, Sensores de Movimientos Infrarrojo Pasivo y con notificaciones remotas en tiempo real a un dispositivo móvil.

OBJETIVOS ESPECIFICOS

- Configurar los componentes para detectar y anunciar el movimiento utilizando la plataforma Raspberry Pi B+ y el sensor PIR, enviando el identificador del sensor y un mensaje personalizado hacia el servidor Azure.

- Procesar la información en el servidor LAMP ejecutándose sobre Azure, recibida desde la Raspberry Pi, para enviar el token y el mensaje, hacia al servidor GCM de Google.

- Desarrollar una aplicación para móviles en Android, que permita recibir las notificaciones PUSH enviadas desde el servidor GCM de Google.

MARCO TEORICO

Raspberry PI

Raspberry Pi es una placa computadora de bajo costo desarrollada en Reino Unido por la Fundación Raspberry Pi, con el principal objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

En realidad, se trata de una diminuta placa base de 85 x 54 milímetros (del tamaño aproximado de una tarjeta de crédito) en el que se aloja un chip Broadcom BCM2835 con procesador ARM hasta a 1 GHz de velocidad (modo Turbo haciendo overclock), GPU VideoCore IV y 512 Mbytes de memoria RAM (Las primeras placas contaban con sólo 256MB de RAM).

Para que funcione, necesitamos de un medio de almacenamiento (**Raspberry Pi** utiliza tarjetas de memoria SD o microSD en sus últimos modelos), conectarlo a la corriente utilizando cualquier cargador micro USB de al menos 1000mah para las placas antiguas y de al menos 2000mah para las modernas, y si lo deseamos, guardarlo todo utilizando una carcasa para que todo quede bien protegido y su apariencia sea más estética.

En función del modelo que escojamos, dispondremos de más o menos opciones de conexión, aunque siempre dispondremos de al menos un puerto de salida de video HDMI y otro de tipo RCA, mini Jack de audio y un puerto USB 2.0 (modelos A y A+, B dispone de dos USB, B+ y Raspberry Pi 2 disponen de 4 USB) al que conectar un teclado y ratón.

GPIO

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

Estos GPIO´s se encuentran no solo en Micro controladores Raspberry sino también en diferentes Chips.

De los modelos de Raspberry pi podemos destacar que RPI A+, B+, 2B y Zero poseen 40 pines en su GPIO así como también el nuevo modelo 3. Los modelos A y B tienen solamente los primeros 26 pines.

Los Pines de la Raspberry en resumen son la puerta al internet de las cosas y si son usados de manera incorrecta puedes dañar tu Raspberry.

El orden de los Pines muchas veces puede ser complicado de memorizar por eso en muchas ocasiones es más recomendado usar una tabla de referencia a ellos, los colores dentro de la tabla también hacen referencia a los protocolos que utilizan algunos de los pines, por ejemplo, el **GPIO de color verde celeste en nuestra tabla** son los del tipo GPIO básico o simplemente usados para apagar, encender y colectar información, por ejemplo, un diodo led. **Los de color Rosado chicha en nuestra tabla** (GPIO 2 y GPIO 3) son los pines que permiten conectar con dispositivos que corran con el protocolo I²C (Inter Integrated Circuit) por sus siglas en inglés, en español como “Circuito Integrado Interno”, este protocolo es un bus de datos serial desarrollado en 1982, los dispositivos conectados que utilizando este protocolo generalmente usan dos pines a la misma vez funcionando uno como maestro y el segundo como esclavo, I²C funciona muy bien para operaciones pequeñas y su muy bajo coste de producción lo hace ideal para estar integrado en una Raspberry Pi, sus usos pueden ser el control numérico ascendente o descendente por ejemplo el volumen de un televisor.

El protocolo I²C tuvo gran importancia en el pasado en el área de las tarjetas y chips. La tarjeta sanitaria utilizada en Alemania hasta finales del 2014 era una tarjeta I²C, es decir, debajo de la

superficie dorada del chip, se encontraba una simple I²C-EEPROM, que podía ser leída y escrita por el lector de tarjetas a través de un protocolo de bus I²C.

Los Pines que se encuentran en color **Celeste Pastel en nuestra tabla** (GPIO 7, GPIO 8, GPIO 9, GPIO 10, GPIO 11) es un estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados a equipos electrónicos. Su funcionalidad en resumen es bastante similar a la de los pines con I²C ya que también funciona con un esclavo y un maestro pero este puede realizar múltiples operaciones en cascada y está más integrado a procesadores ARM, MIPS o PowerPC, mayormente son usados para comunicarse con una variedad de periféricos como:

- Sensores: temperatura, presión, pantallas táctiles, controles de videojuegos.
- Dispositivos de control: Codecs de audio, potenciómetros digitales.
- Lentes de cámara: Canon EF Lens Mount.
- Comunicación, Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 802.11.
- Memoria: Flashy EEPROM
- Relojes en tiempo real.

Los Pines con **color Lila en nuestra tabla** (GPIO 14 y GPIO 15) están bajo el estándar UART que son las siglas de Universal Asynchronous Reciever Transmitter en español Transmisor Receptor Asíncrono Universal, este es un hardware que controla los puertos y dispositivos de serie, se encuentra integrado y actualmente es utilizado en diversas placas bases o tarjetas adaptadoras, así como lo está en el grupo de pines de la Raspberry.
El dispositivo traduce los datos entre caracteres (usualmente bytes) en una computadora y en formato de comunicación asíncrona serial el cual encapsula esos caracteres en "Start Bytes". Los pines UART usualmente son parte de un circuito integrado usado para comunicaciones seriales sobre un computador o dispositivo con puerto serial y en su mayoría incluidos en micro controladores.

UART es muy útil a la hora de trabajar en robótica, algunos ejemplos de lo que podemos hacer con UART serían: controlar un LCD, Bluetooth, Wireless, hacer un data logger, depurar código, probar sensores.

El Pin 27 y 28 que se encuentran paralelos y de color amarillo en nuestra tabla con etiqueta ID_SD e ID_SC son pines reservados para ID EEPROM, en tiempo de arranque estas interfaces I2C serían consultadas para ver si poseen un EEPROM que identifique la tarjeta conectada con ellos y permita configuraciones automáticas en los pines de entrada y salida (opcionalmente también Controladores Linux), También conocidos como "DO NOT USE" (DNU) por sus siglas en inglés o en español "NO USAR" ya que es de uso exclusivo para conectar EEPROM, se recomienda dejarlos desconectados si no se requiere el uso de EEPROM.

Los Pines 1 y 17 en nuestra tabla de color naranja son pines de poder o pines de 3v3 que es la manera políticamente correcta para referirse a números que poseen puntos decimales, en algunas partes de Europa por ejemplo se usan comas para separar enteros de dígitos fraccionarios. Para evitar la ambigüedad en situaciones internacionales se está colocando la letra “v” en el lugar donde el punto decimal debería estar. 3v3 es 3.3 y 1v3 es 1.8 por ejemplo.

Los Pines 2 y 4 que se encuentran en color rojo pastel en nuestra tabla son los pines que alimentan con 5V a nuestros sensores para mayor aprovechamiento de estos ya que las Raspberry solo poseen 2 pines es usual usar un Breadboard para conectar tantos sensores como se desee.

Los pines de alimentación de 5v están conectados directamente a la alimentación de entrada de la Pi y son capaces de proporcionar totalidad de la corriente de la fuente de alimentación.

Finalmente tenemos los Pines GND (Ground) o Tierra en español, Pines 6, 9, 14, 20, 25, 30, 34 y 39 de color gris en nuestra tabla, los cuales están conectados entre sí y se usan para llevar a tierra cualquier derivación indebida de la corriente eléctrica a los elementos con los que se puede estar en contacto.

Raspberry Pi B+

General Pourpuse Input/Output

Entrada/Salida de proposito general

Figura 1, Raspberry Pi B+ - GPIO

Figura 1, GPIO Raspberry Pi B+

Python

Python es el lenguaje de programación que es recomendado por los fundadores de la **Raspberry Pi**, pues piensan que es un lenguaje de sintaxis sencilla y clara que puede venir bien en los temas de educación.

Es un lenguaje interpretado o de script, fuertemente tipado y dinámico, es multiplataforma y es orientado a objetos. Además, es un lenguaje bastante potente y con muchas librerías que nos ayudan a realizar casi cualquier cosa.

IDLE de Python en Raspberry:

La introducción más fácil a Python es a través del IDLE, un entorno de desarrollo Python el cual se encuentra pre instalado en la mayoría de las distribuciones Linux para Raspberry.

Figura 2, IDLE - Python.

El IDLE (Integrated Development and Learning Enviroment) nos da un REPL (Read-Evaluated-Print-Loop) por sus siglas en ingles que básicamente es una interfaz en la que pueden ingresar comandos Python y esta devuelve el resultado en orden de ejecución.

Cabe mencionar que la versión actual de Python es la 3 lanzada en el año 2008, pero aún se mantiene el soporte para la versión 2 que fue lanzada en el año 2000 ya que existen muchas aplicaciones que dependen de librerías que no existen aún en la versión 3.

Sensor PIR

Sensor Infrarrojo Pasivo o PIR por sus siglas en Ingles en un sensor electrónico que mide la luz infrarroja que irradia desde los objetos en su campo visible. Más comúnmente usado en detectores de movimiento.

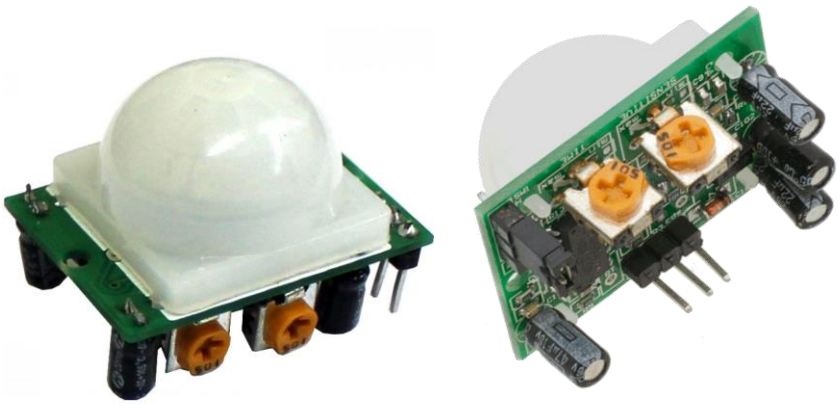


Figura 3, Sensor Infrarrojo Pasivo (PIR).

Principios operativos:

Todos los objetos con una temperatura sobre el Cero absoluto emiten energía calorífica en forma de radiación. Usualmente esta radiación es invisible al ojo humano porque irradia en longitudes de onda infrarroja. Pero puede ser detectada con dispositivos electrónicos creados con este fin como lo es el PIR.

El termino pasivo en el PIR se refiere a que este dispositivo no genera o irradia ninguna energía para propósito de detección, trabaja enteramente detectando energía emitida por otros objetos, el sensor PIR no mide el calor sino que detecta la radiación infrarroja emitida o reflejada de un objeto

Detector de movimiento.

Un detector de movimiento es usado para detectar movimiento de personas, animales y otros objetos. Comúnmente usado en alarmas de seguridad y activación automática de sistemas de luz.

HC-SR501

Descripción:

El módulo HC-SR501 tiene 3 pines de conexión +5v, OUT (3,3v) y GND, y dos resistencias variables de calibración (Ch1 y RL2).

- Ch1: Con esta resistencia podemos establecer el tiempo que se va a mantener activa la salida del sensor. Una de las principales limitaciones de este módulo es que el tiempo mínimo que se puede establecer es de más o menos 3s. Si cambiamos la resistencia por otra de 100K, podemos bajar el tiempo mínimo a más o menos 0,5 s.
- RL2: Esta resistencia variable nos permite establecer la distancia de detección que puede variar entre 3-7m.

La posibilidad de mantener activa la salida del módulo durante un tiempo determinado nos permite poder usarlo directamente para prácticamente cualquier aplicación sin necesidad de usar un micro controlador.

Características:

- Sensor piro eléctrico (Pasivo) infrarrojo (También llamado PIR)
- El módulo incluye el sensor, lente, controlador PIR BISS0001, regulador y todos los componentes de apoyo para una fácil utilización
- Rango de detección: 3 m a 7 m, ajustable mediante trimmer (Sx)
- Lente fresnel de 19 zonas, ángulo < 100º
- Salida activa alta a 3.3 V
- Tiempo en estado activo de la salida configurable mediante trimmer (Tx)
- Re disparo configurable mediante jumper de soldadura
- Consumo de corriente en reposo: < 50 µA
- Voltaje de alimentación: 4.5 VDC a 20 VDC.

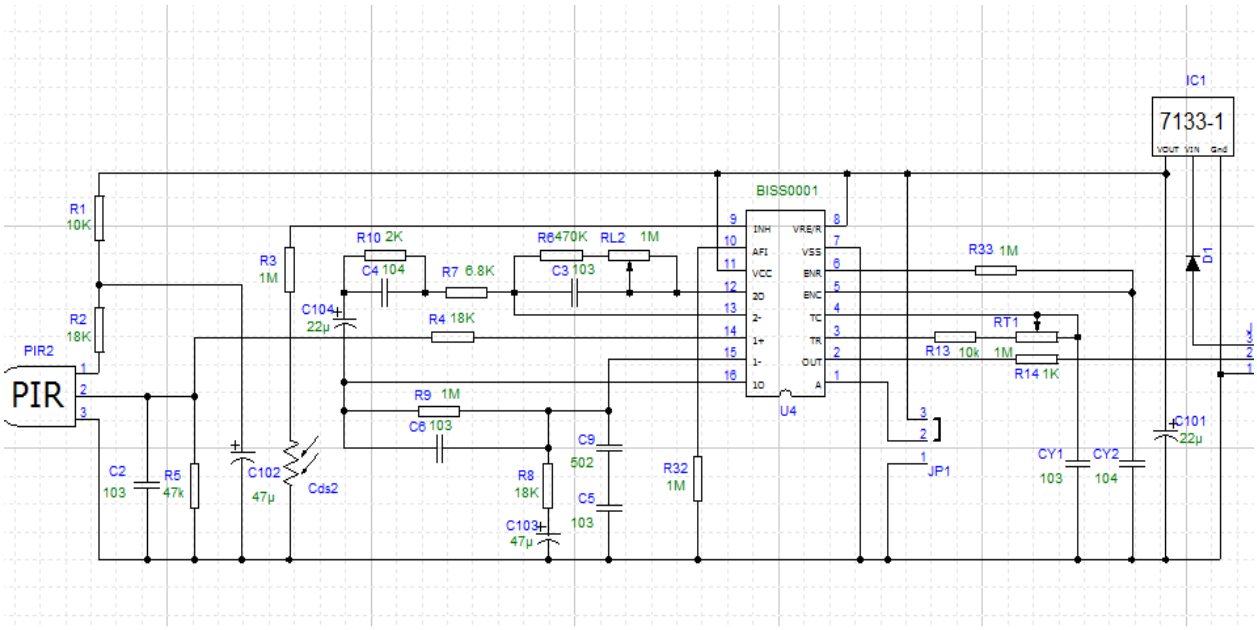


Figura 4, Esquema lógico del sensor de movimiento HC-SR501

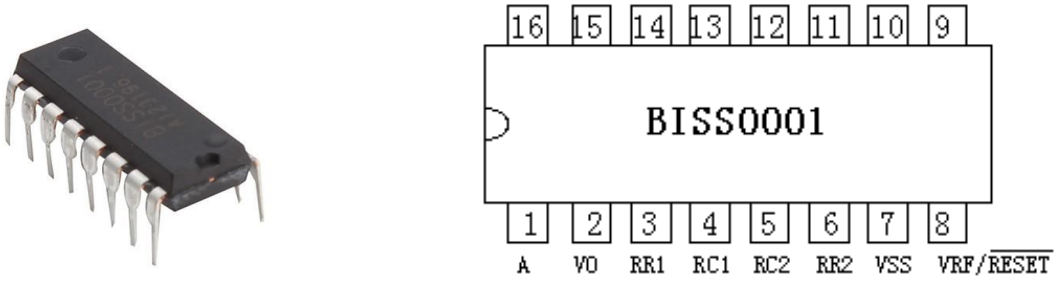


Figura 5, BISS0001 Micro Power PIR Motion Detector IC.

Lente Fresnel de 19 zonas.

Es un tipo de lente compacto desarrollado por el físico francés Augustin-Jean Fresnel para Faros Náuticos originalmente, el diseño permite la construcción de un lente con apertura larga y una corta distancia focal sin el peso y volumen de material que debería usarse en una lente de diseño convencional.

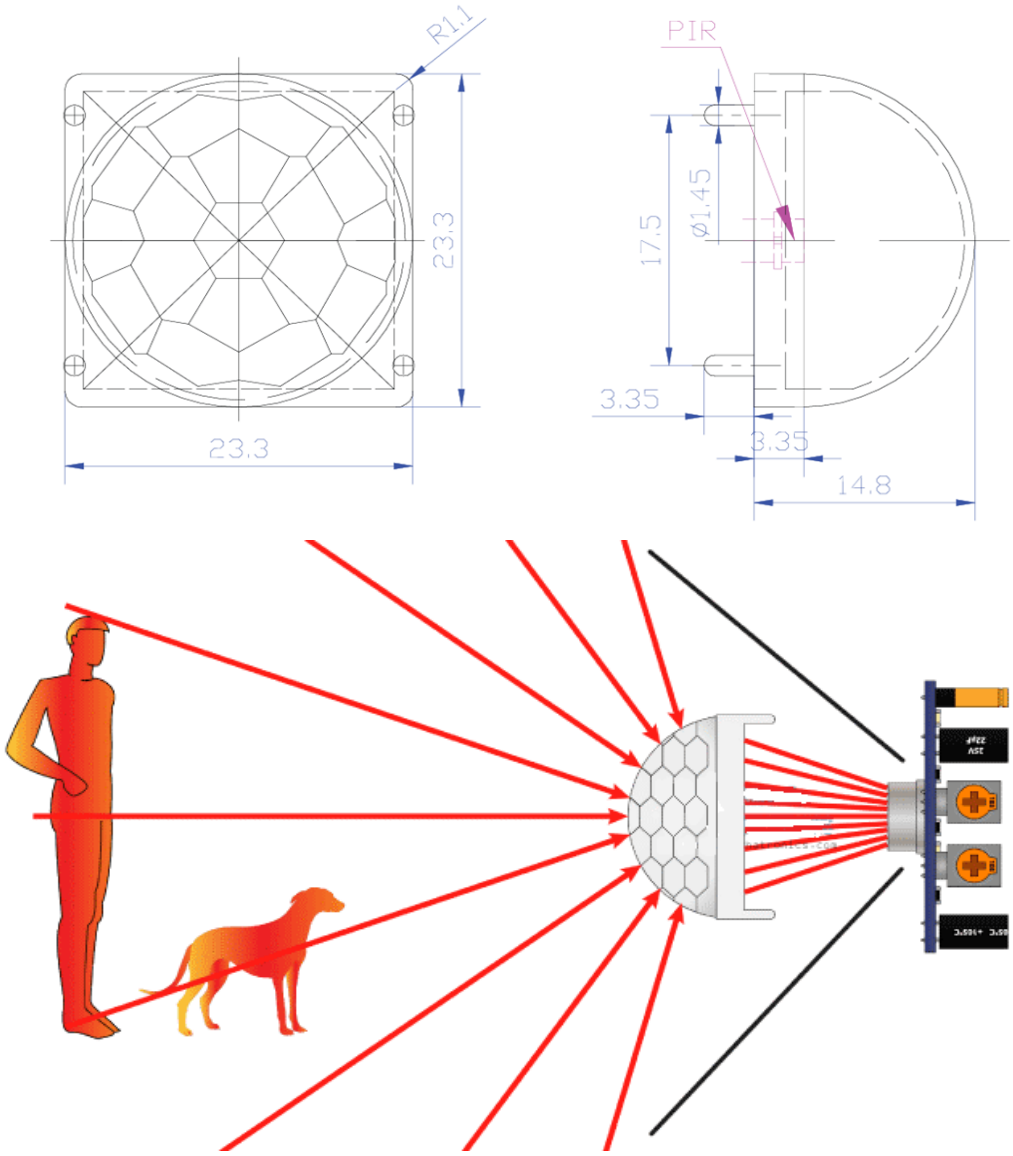


Figura 6, Lente Fresnel de 19 zonas

Computación en la nube.

Servicios de nube:

Según la real academia de la ingeniería Computación en la nube es: La utilización de las instalaciones propias de un servidor web albergadas por un proveedor de Internet para almacenar, desplegar y ejecutar aplicaciones a petición de los usuarios demandantes de las mismas.

También podemos decir que el “Cloud Computing” se refiere a la entrega bajo demanda de recursos informáticos y aplicaciones a través de Internet con un sistema de precios basado en el consumo realizado. ¿Pero qué es exactamente? ¿Es solamente sobre alojamiento web? ¿Es algo viejo vestido con ropa nueva?, ¿porque las organizaciones deberían considerar su utilización?

A través del tiempo y desde que el paradigma de la computación en la nube nació, su significado y propósito ha evolucionado rápidamente y lo seguirá haciendo, muchos profesionales de TI concluyen en diferentes definiciones según su punto de vista profesional y basado en lo que este paradigma ofrece. Como no existe acuerdo acerca que precisamente constituye la computación en la nube, este sigue ofreciendo un paradigma prometedor que puede permitir a las empresas encarar el mercado volátil de forma ágil y de la mejor Costo-eficiencia forma posible.

Recientemente una definición poco más concisa ha emergido: Externalizar las actividades de TI a uno o más terceros que tienen alta disponibilidad de recursos para solventar las necesidades de las organizaciones de manera fácil y eficiente. Estas necesidades pueden incluir componentes hardware, redes, almacenamiento, sistemas de software y aplicaciones en adición pueden incluir elementos de infraestructura como lo son espacio físico, equipos de enfriamiento, electricidad, sistemas contra fuego y recursos humanos para mantener todos estos elementos funcionando de la manera más óptima.

En este modelo los usuarios facturan por su uso de la infraestructura de TI más que por la compra, instalación y manejo de estos. Esta estructura nos permite una escalabilidad flexible tanto para incrementar los recursos como para decrementarlos en respuesta a la fluctuación de necesidades del mercado.

Software como servicio

El **Software como servicio** SaaS por sus siglas en inglés “Software as a Service” se encuentra en el nivel más alto de la pirámide y caracteriza una aplicación completa ofrecida como un servicio, por demanda, vía multi-tendencia que significa una sola instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes. Las aplicaciones que suministran este modelo de servicio son accesibles a través de un navegador web o de cualquier aplicación diseñada para tal efecto y el usuario no tiene control sobre ellas, aunque en algunos casos se le permite realizar algunas configuraciones. Esto le elimina la necesidad al cliente de instalar la aplicación en sus propios computadores, evitando asumir los costos de soporte y el mantenimiento de hardware y software.

Un gran ejemplo de este modelo es el servicio de Google Docs y su aplicación para Crear y Editar documentos en línea al igual que su homónimo Office Web App de Microsoft.

Figura 7, Google docs – Office Web App. (SaaS).

Plataforma como servicio

La **Plataforma como servicio** PaaS por sus siglas en inglés o “Platform as a Service” es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o APIs pre-configurados y listos para integrarse sobre una tecnología concreta de desarrollo (por ejemplo, un sistema Linux, un servidor web, y un ambiente de programación como Perl o Ruby). Las ofertas de PaaS pueden dar servicio a todas las fases del ciclo de desarrollo y pruebas del software, o pueden estar especializadas en cualquier área en particular, tal como la administración del contenido.

Ejemplos comerciales son Google App Engine, que sirve aplicaciones de la infraestructura Google; Microsoft Azure, una plataforma en la nube que permite el desarrollo y ejecución de aplicaciones codificadas en varios lenguajes y tecnologías como .NET, Java y PHP o la Plataforma G, desarrollada en Perl. Servicios PaaS como éstos permiten gran flexibilidad, pero puede ser restringida por las capacidades disponibles a través del proveedor.

En este modelo de servicio al usuario se le ofrece la plataforma de desarrollo y las herramientas de programación por lo que puede desarrollar aplicaciones propias y controlar la aplicación, pero no controla la infraestructura.

Figura 8, Google app engine – Windows Azure. (PaaS).

Infraestructura como servicio

La **infraestructura como servicio** IaaS por sus siglas en Ingles “Infraestructura as a Service” también llamada en algunos casos “Hardware as a Service” o Hardware como servicio, es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo, a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo —desde procesamiento en lotes (“batch”) hasta aumento de servidor/almacenamiento durante las cargas pico. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales (respectivamente). Otro ejemplo es Joyent, cuyo producto principal es una línea de servidores virtualizados, que proveen una infraestructura en demanda altamente escalable para manejar sitios web, incluidas aplicaciones web complejas escritas en Python, Ruby, PHP y Java.

Figura 9, Joyenet, Windows Azure, Rackspace, Amazon EC2, Oracle Cloud. (IaaS).

Otros Ejemplos de Computación en la nube:

- Box (sitio web) - desarrollado por Box Inc.
- Campaign Cloud - desarrollado por ElectionMall Technologies
- Doitle ajaxplorer - desarrollado por Doitle
- Dropbox - desarrollado por Dropbox
- Google Drive - desarrollado por Google
- iCloud - desarrollado por Apple
- OneDrive - desarrollado por Microsoft (Antes SkyDrive)
- OwnCloud - desarrollado por OwnCloud Inc.
- Salesforce.com - desarrollado por Salesforce.com Inc.
- SugarSync - desarrollado por SugarSync
- Ubuntu One - desarrollado por Canonical->(Cerrado)
- Wuala - desarrollado por LaCie
- Dataprius - desarrollado por Dataprius
- CISCO Metapod – desarrollado por CISCO.
- Docker – desarrollada por Docker.
- Heroku – adquirido por Salesforce.com
- Oracle Cloud – desarrollada por Oracle.
- Apache CloudStack – desarrollada por Apache.
- FUJITSU Cloud IaaS Trusted Public S5 – desarrollada por Fujitsu.
- IEEE Cloud Computing – desarrollada por IEEE.
- Eucalyptus – desarrollada por Eucaliptus Systems, inc.

GCM (Google Cloud Messaging)

Mensajería Google en la nube (GCM) es un servicio gratuito lanzado en 2012 por Google, que permite a los desarrolladores enviar mensajes entre los servidores y las aplicaciones clientes Esto incluye los mensajes desde los servidores de aplicaciones a los clientes, y los mensajes de las aplicaciones cliente a los servidores.

Descripción de la arquitectura:
La implementación del GCM incluye un Servidor de Conexiones de Google, un servidor de aplicaciones en tu ambiente que interactúe con el servidor de conexiones vía protocolos HTTP o XMPP y finalmente una aplicación cliente.

Figura 10, Google Cloud Messaging

Actualmente GCM tiene soporte para las plataformas iOS, Android y Chrome, y se encuentra en constante desarrollo. La plataforma es completamente gratuita y posee muchas características de implementación, en el último cuatrimestre Google anuncio su nueva versión FCM o “Firebase Cloud Messaging” que sería la sucesora de GCM en la cual se mejoran las características ya integradas a GCM.

Android App

Android Es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, *tabletas* o teléfonos; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró.

Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Los dispositivos de Android venden más que las ventas combinadas de Windows Phone e IOS.

Periodo	Android	iOS	Windows Phone	BlackBerry OS	Otros
2015Q2	**82.8%**	**13.9%**	**2.6%**	**0.3%**	**0.4%**
2014Q2	**84.8%**	**11.6%**	**2.5%**	**0.5%**	**0.7%**
2013Q2	**79.8%**	**12.9%**	**3.4%**	**2.8%**	**1.2%**
2012Q2	**69.3%**	**16.6%**	**3.1%**	**4.9%**	**6.1%**

Android comenzó en el mercado comercial con su versión 1.6 o “Donut” junto con un Android Market en el 2008 el cual fue rediseñado para presentar en forma de catálogo aplicaciones gratuitas y pagadas. Desde entonces ha ido mejorando sus versiones liberando APIs para los desarrolladores de la mejor manera posible.

Las siguientes versiones de Android siguen las letras del alfabeto haciendo mención a postres y dulces 2.1 “Eclair”, 2.2 “Froyo”, 2.3 “Gingerbread”, 3.0 “Honeycomb”, 4.0 “Ice Cream Sandwich”, 4.1 “Jelly Bean”, 4.4 “KitKat”, 5.0 “Lollipop” y finalmente 6.0 “Marshmallow”.

METODOLOGIA IMPLEMENTADA

Materiales Utilizados

Materiales Hardware:

Materiales	Descripción	Costo
Raspberry Pi B+	Ordenador de placa reducida o de bajo coste	$35.00
Sensor PIR HC-SR501	Un **sensor infrarrojo pasivo** es un sensor electrónico que mide la luz infrarroja radiada de los objetos situados en su campo de visión.	$2.00
MicroSD	Raspberry Pi utiliza tarjetas de memoria SD o microSD, como medio de almacenamiento de su sistema operativo	$5.00
Materiales Complementarios	2 Push buttons (2 pulsadores), 2 leds, 2 resistencias de 220 Ohm, 12 cables duros (hard wires) de 10 cm, 1 protoboard (placa de pruebas).	$2.00
Cable de Energía para Raspberry Pi	Proporciona los 5v de energía para encender la Raspberry pi	$2.00
Teléfono Android	Teléfono con sistema operativo Android	$100.00
TOTAL	**Total, en materiales usados para el sistema de seguridad**	**$146.00**
Laptop	Samsung Series 5 - Intel Core i5 2467M, Processor 1.6GHz, 8GB DDR3 RAM, 500GB Hard Drive	$525.00
TOTAL	**Total, para la realización de este proyecto**	**$671.00**

Materiales Software

Software	Descripción	Costo
Android Studio	Software libre utilizado para desarrollar la aplicación Android del proyecto	$0
Raspbian	Sistema operativo libre y gratuito basado en Debian y optimizado para el hardware de la Raspberry Pi.	$0
Python	Lenguaje de programacion utilizado para programar el sensor PIR	$0
LAMP	Linux, Apache, MySql, PHP	$0
Total:		$0

Etapas del Trabajo

Etapa 1: Recopilación de la información

Para la realización del proyecto principal estuvimos recolectando información, implementando métodos de investigación básicos y realizando pruebas individuales con diferentes herramientas de hardware libre para el mayor entendimiento de estos equipos.

Aplicamos conocimientos adquiridos durante la carrera para algunas partes de desarrollo final de la práctica principal otro tipo de aptitudes necesarias fueron debidamente investigadas.

Etapa 2: Diseño del equipo.

En esta etapa realizamos un esquema básico de lo que se quería lograr, iniciamos el prototipo del sistema de seguridad y aplicamos un método básico de diseño de producto.

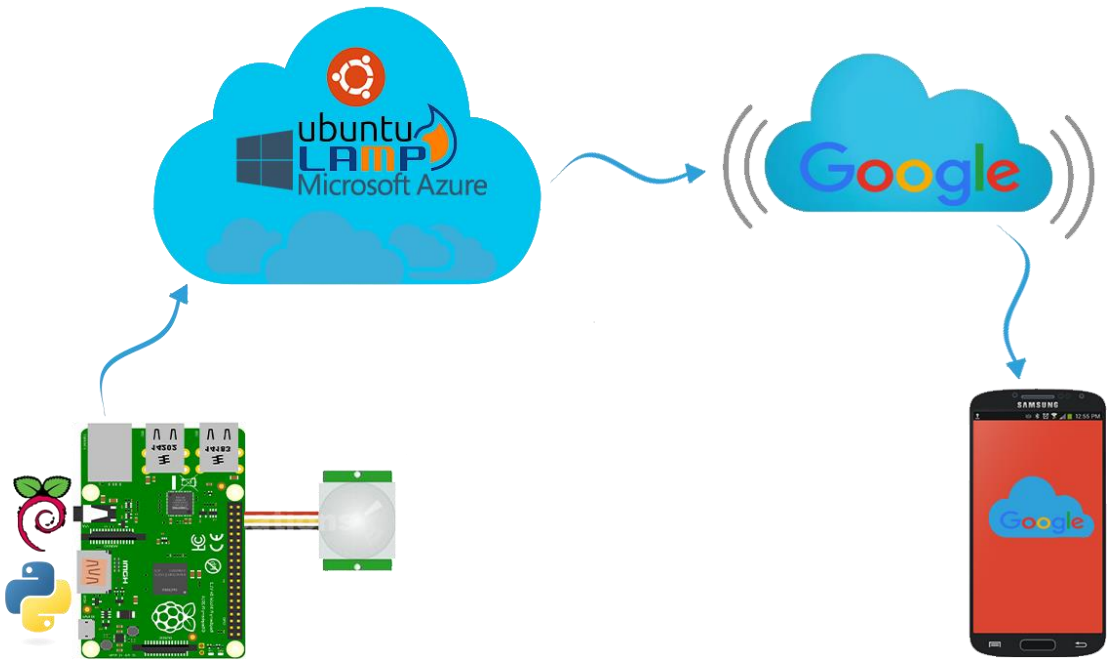


Figura 11, Diseño lógico del funcionamiento del sistema de seguridad.

Etapa 3: Comprobación de conexión entre dispositivos

Parte 1: Durante la primera parte estudiamos como se comportaba el sensor de movimiento y aprendimos el funcionamiento de sus reguladores analógicos Ch1 y RL2.

Parte 2: En la segunda parte trabajamos con la máquina virtual en la plataforma Azure de servicios de nube como lo son la configuración del servidor y la instalación de servicios necesarios para la realización del producto en cuestión.

Parte 3: Trabajamos con el servicio gratuito de Google, GCM con el fin de aprender del servicio, así como la manera de integrarlo al proyecto.

Parte 4: Aplicando el conocimiento adquirido durante la carrera trabajamos sobre el IDE de Android para finalizar el ciclo de comunicación entre los diferentes nodos.

Etapa 4: Pruebas del sistema.

En esta etapa realizamos las pruebas de campo con el producto terminado, así como evaluación de errores y escenarios del servicio.

Etapa 5: Informe Final

Recopilamos todas las anotaciones realizadas durante la realización de las etapas anteriores para estructurar un documento detallado en donde indicamos paso a paso la configuración y el funcionamiento de nuestro proyecto y sus resultados.

Implementación del sistema

En esta sección abordaremos el desarrollo de las diferentes etapas del proyecto de manera más detallada.

Recopilación de la información.

En esta parte realizamos varias investigaciones referentes a microcomputadores Raspberry Pi con el fin de familiarizarse con el entorno de trabajo que posteriormente se deseaba realizar.

Fueron necesarias horas considerables de trabajo sobre la Raspberry Pi, no por cuestiones de complejidad sino con fines de estudio de las diferentes maneras de trabajar con ella. Se hicieron pruebas con diferentes sistemas operativos, así como Pidora, Raspbian y Snappy ubuntu principalmente.

Al final decidimos trabajar con Raspbian por ser un sistema operativo ligero y con un amplio soporte y comunidad de respaldo. A partir de acá se realizaron varias practicas pequeñas principalmente con el sensor PIR para entender mejor el funcionamiento de los reguladores de distancia sensorial y tiempo que se mantiene activa la salida del sensor.

Diseño del equipo

Durante esta etapa y ya teniendo conocimientos sobre el alcance de nuestro computador de bajo coste gracias a la constante investigación previa, diseñamos el sistema de seguridad aplicando un método de desarrollo de producto por integrales.

Definición del esquema de funcionamiento.

El funcionamiento del sistema se basa en la detección de movimiento gracias a un sensor infrarrojo pasivo ubicado en un lugar estratégico para la detección de intrusos, el momento en que el sensor detecta movimiento se ejecuta un script que envía un identificador al servidor, este lo busca en la base de datos de dispositivos y se encarga de reenviarlo al servicio de notificación de Google que es el encargado de enviar una notificación al dispositivo móvil Android que contiene la aplicación previamente instalada y el token registrado.

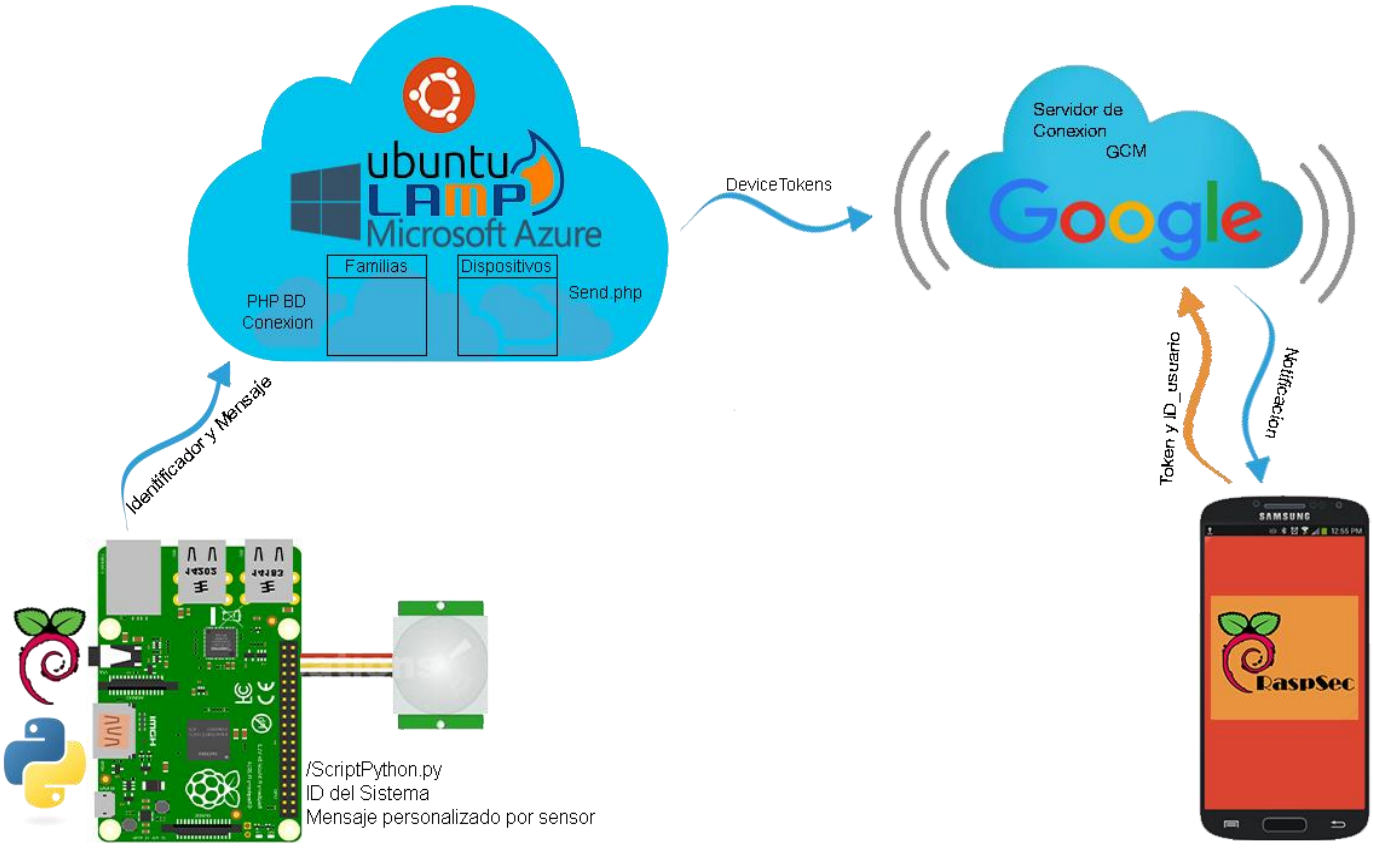


Figura 12, Descripción del diseño lógico del funcionamiento una vez lograda la comunicación entre ellos.

Creación y configuración del equipo

Raspberry PI

Empezaremos con la configuración de la Raspberry Pi, existen una variedad de formas para instalar el sistema operativo en nuestro computador, para usuarios nuevos en esta área recomendamos altamente el uso del software NOOBS el cual está diseñado para facilitar la gestión de instalación del SO, considerémoslo como un asistente de instalación amigable.

La vía utilizada para este proyecto fue escribiendo la imagen directamente sobre la Tarjeta SD. Podemos encontrar una serie de imágenes de diferentes sistemas operativos en el sitio oficial de Raspberry.

Para realizar la instalación de esta forma usaremos el comando $dd el cual puede sobre escribir datos de cualquier partición de tu sistema si se especifica el dispositivo equivocado se puede perder la partición primaria de Linux de tu PC o laptop.

Paso 1:

$df –h

Con este comando miraremos los dispositivos que están montados actualmente en tu PC. Conectamos la Tarjeta SD en el Slot o adaptador a tu PC. Corremos nuevamente el comando $df – h y ahora el dispositivo que aparece es tu tarjeta SD, usualmente debería ser algo como /dev/sdb1, con esto tendremos claro cuál será la ruta para utilizar el comando $dd sin ningún peligro. En nuestro caso tenemos /dev/sdb1 y /dev/sdb2 nótese que nos aparecen más de una partición ya que como tenemos un sistema previamente instalado, as imágenes para Raspberry utilizan más de una partición.

Lo siguiente es desmontar todas las particiones que te muestra el comando df asignadas al SD utilizaremos el comando $umount.

$umount /dev/sdb1

$umount /dev/sdb2

En esta parte vamos a escribir la imagen en la SD asegurándonos de utilizar la ruta de nuestra imagen .img en el campo if= y la ruta a la SD en el campo of= en este caso /dev/sdb, esta parte es muy importante ya que un error puede costarte los datos de tu disco principal, debemos asegurarnos que la ruta que le pasamos es la que señala a la SD en su totalidad y no solo una de sus particiones, ejemplo /dev/sdb y no /dev/sdb1 en nuestro caso.

$dd bs=4M if=/home/omar/2016/Pimages/2016-05-27-raspbian-jessie.img of=/dev/sdb

El tamaño de bloque a utilizar o bs= (Block Size) será de 4M el cual es el tamaño recomendado, en algunas ocasiones tiende a fallar, en esos casos se recomienda usar 1M aunque esto puede tardar varios minutos más.
El comando $dd no nos muestra ninguna información de progreso y pueda parecer que la terminal está congelada, pero solo hay que esperar un poco para que este termine de copiar los datos de la imagen en la SD, se recomienda también que si el proceso toma más de 5 minutos, abrir otra terminal y utilizar el comando $pkill –USR l –n –x dd, el progreso se mostrara en la terminal original y no en la terminal donde se ejecutó el comando y puede que no se muestre automáticamente a causa de tiempo de buffer.

Al finalizar la copia con el comando $dd correremos el comando $sync para asegurarnos de que la cache se ha descargado y que es seguro desmontar la SD.

En esta parte procedemos a colocar la SD en nuestra Raspberry Pi. Para la primera vez se recomienda usar un monitor conectado a ella el que facilitara su configuración.

Una vez el sistema copiado en la SD y apropiadamente configurado lo básico (Idioma, hora, acceso a internet, etc.), se procedió a la configuración del Servidor LAMP, esta configuración es bastante básica, pueden encontrar mucha información en el sitio oficial de Raspberry.

El servidor LAMP por los momentos no se usara pero posteriormente se pretende crear una interfaz local a la cual los usuarios puedan acceder para verificar logs, encender o apagar servicios a través de una aplicación web.

A partir de acá haremos las pruebas y configuraciones vía SSH.

Se crearon 2 Scripts en la Raspberry para controlar el sistema, el primero creado fue el que controlaba al sensor en nuestro caso es el que llamaremos PruebaPIR1.py y el segundo que creamos lo llamamos botón.py este segundo fue creado para controlar el sistema de forma física y no tener que iniciar el script desde el terminal vía SSH.

El primer script que analizaremos es el del botón.py, este script fue colocado al final del archivo /etc/rc.local para que se inicie al momento de la carga del sistema.

CODIGO:

```
# El código empieza por las librerías de Python que se utilizan
import RPi.GPIO as GPIO
# Al declararlo así podemos referirnos al GPIO como GPIO en el resto del código.
import time
import os, signal
GPIO.setmode(GPIO.BCM)
# Se indican los pines GPIO que se utilizan y se asigna como variable

led = 25
led2 = 12
pbon = 18
pboff = 16

# Con GPIO.setup establecemos los estados mandatorios
GPIO.setup(led, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(pbon, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(pboff, GPIO.IN, pull_up_down = GPIO.PUD_UP)

GPIO.input(pboff) == False
while True: # dentro del while tendremos los ciclos con los que controlaremos los botones físicos para controlar el Sistema.

    if GPIO.input(pbon) == False:
                print "esta encendido"
                GPIO.output(led, GPIO.LOW)
                GPIO.output(led2, GPIO.HIGH)
                os.system("sudo pkill -f PruebaPIR1.py")
                os.system("sudo python /media/TESIS/PruebaPIR1.py")
                time.sleep(1)
        if GPIO.input(pboff) == False:
                GPIO.output(led, GPIO.HIGH)
                GPIO.output(led2, GPIO.LOW)
                os.system("sudo pkill -f PruebaPIR1.py")
                os.system("sudo pkill -f PruebaPIR1.py")
                print "sensor deberia estar apagado"
                time.sleep(1)
```

Luego analizaremos el código que controla el sensor PIR

CODIGO

```
import RPi.GPIO as GPIO
import time
import webbrowser
import urllib2, urllib
import os

sensor = 4
pboff = 16
led2 = 12
led = 25

GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor, GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(pboff, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(led, GPIO.OUT)

previous_state = False
current_state = False

def sendNotification(): #Definiremos la funcion paraenviar la notificación
mydata=[('msg','Demostracion'),('id','familia1')] # aquí tendremos el id del dispositivo y el mensaje que se enviara al servidor con estos datos se identificara en la base de datos que se encuentra en Azure.
        mydata=urllib.urlencode(mydata)
        path='http://13.90.250.14/send.php'  # se pasa la URL del servidor al que se va a enviar
        req=urllib2.Request(path, mydata)
        req.add_header("Content-type","application/x-www-form-urlencoded")
        page=urllib2.urlopen(req).read()
        print page

while True: # El ciclo que está verificando el estado del sensor
   time.sleep(0.1)
   previous_state = current_state
   current_state = GPIO.input(sensor)
   if current_state != previous_state:
      new_state = "HIGH" if current_state else "LOW"
      print("GPIO pin %s is %s" % (sensor, new_state))
      if new_state == 'HIGH':
         sendNotification() # llamado a la función si el sensor detecta movimiento
   if GPIO.input(pboff) == False:
            print "esta apagado"
            GPIO.output(led, GPIO.HIGH)
        GPIO.output(led2, GPIO.LOW)
            os.system("sudo pkill -f boton.py")
            os.system("sudo python /media/TESIS/boton.py")
            time.sleep(0.1)
```

Ambos códigos funcionan en armonía permitiendo que el sistema funcione si errores, al encender la Raspberry el código que controla los botones se inicia automáticamente, al presionar el botón de encendido el script de botones invoca el script del sensor y al presionar dos veces el botón de apagado se cancela el script del sensor y se mata el proceso respectivamente. Se implementó de esta forma como medida de seguridad. Otra característica que posee es que al iniciar el script del sensor se espera 20 segundos para darle tiempo al usuario de salir de la habitación.

Ahora se procede a realizar el circuito utilizando el sensor de movimiento.

Se realizan pruebas con el sensor para calibrarlo.

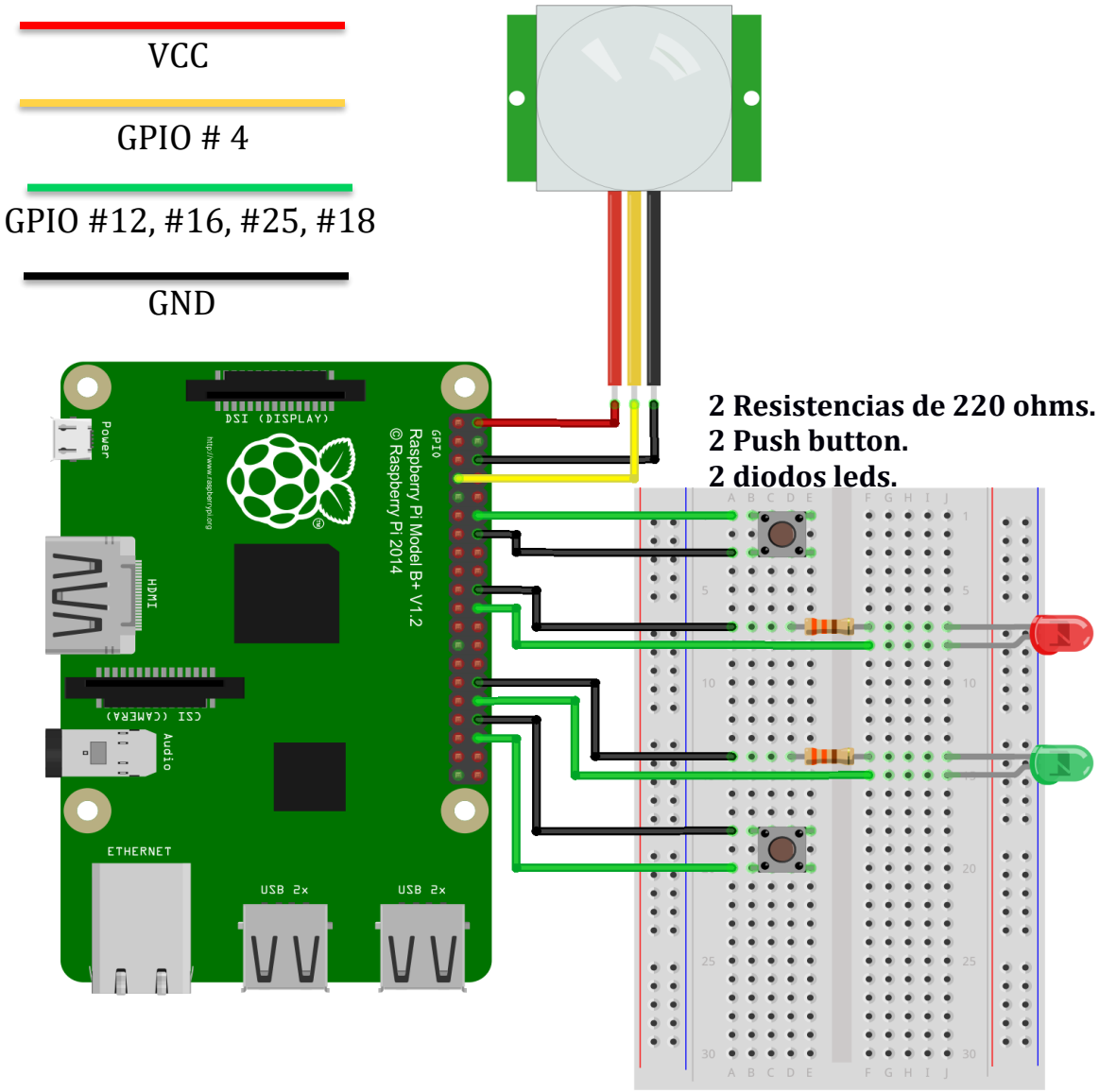


Figura 13, Esquema de conexiones físicas:

En el esquema lógico del circuito el GND se encuentra en un solo lugar ya que como mencionamos anteriormente los GND están todos conectados entre sí y distribuidos entre los pines del GPIO para facilitar la conexión de los proyectos.

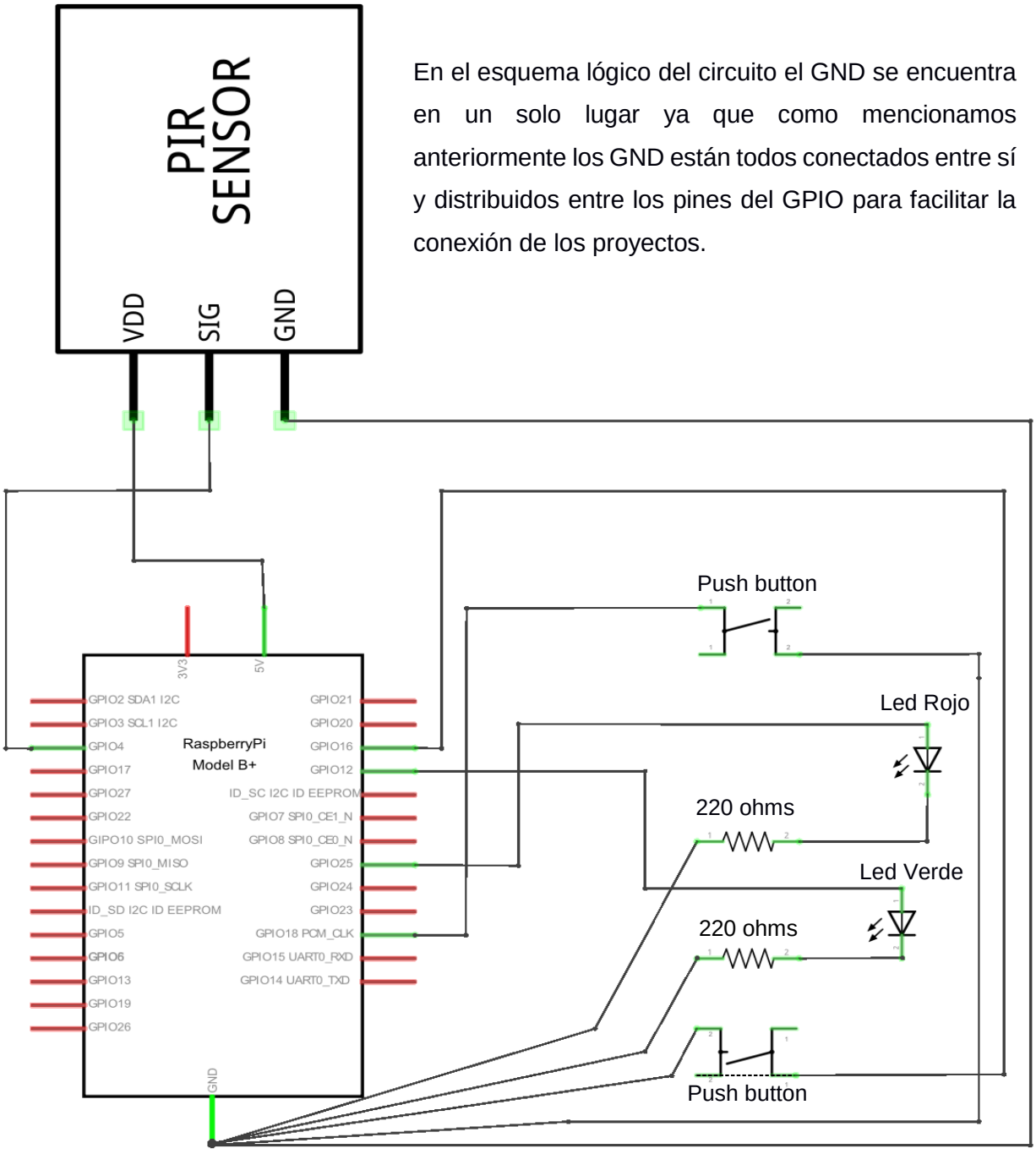


Figura 14, Esquema de conexión Lógica del circuito.

Servidor en el Cloud

Para la creación de este servidor o máquina virtual en la nube ocuparemos la nube de Microsoft sobre la plataforma de Windows Azure, las universidades que cuentan con cuentas Dreamspark para sus estudiantes tienen acceso a un código que les permite no solo descargar de forma gratuita Software Microsoft, sino que tienen la posibilidad de utilizar ese código para crear una cuenta gratuita en Windows Azure por un año con crédito mensual de 150 Dólares para libre uso de los servicios.

En nuestro caso creamos una Maquina que corre con Ubuntu Server 14.04 posee un procesador compartido y 768MB de memoria RAM que bastaran para utilizarse como base de datos de nuestro sistema de seguridad.

Luego de configurar el Ubuntu server procedimos a conectarnos vía SSH utilizando la IP publica del servidor proporcionado por la plataforma a través de un Terminal Linux. Ahí procedimos a instalar y configurar un Servidor LAMP para la gestión de usuarios y comunicación con el servicio de GCM de Google, una vez configurado el servidor LAMP procedemos a crear la base de datos: Las dos tablas se relacionan ID con ID_USUARIO cuando un usuario tenga más de un dispositivo y/o cuando se quieran notificar solamente a usuarios predeterminados de una familia.

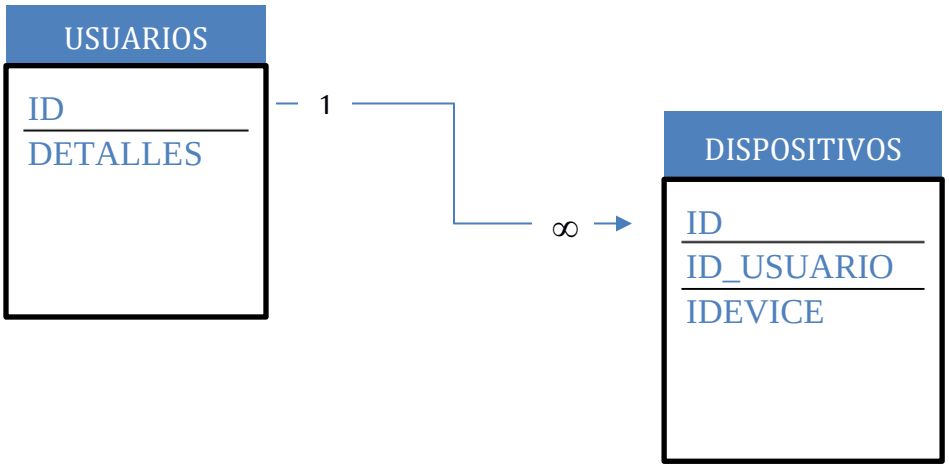


Figura 15, Diseño de la base de datos utilizada.

Y el código que se encarga de escuchar los dispositivos, crear la conexión a la base de datos y enviar el token y el mensaje al servidor de conexión de Google es el siguiente:

CODIGO

```
<?php // API access key from Google API's Console
define('API_ACCESS_KEY', 'AIzaSyD2A8S-j_AJgK7jkMOBIb4R4Rd0fES18Zk');

$iduser = $_POST['id'];

$servername = "localhost";
$username = "root";
$password = "159753Geek";

// Crea la conexión
$conn =mysqli_connect("localhost", "root", "passwd", "db_security"); ;

// Verifica que la conexión sea satisfactoria.
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";

if ($stmt = $conn->prepare("SELECT idevice FROM dispositivos WHERE id_usuario=?")) {
   $stmt->bind_param("s", $iduser);
   $stmt->execute();
   $result;
   $stmt->bind_result($result);

        while($stmt->fetch())
        {
                echo $result;
                $registrationIds =array($result);
                $msgp = $_POST['msg'];
                // prep the bundle
                $msg = array (
                        'message' => $msgp
                        //'title' => 'This is a title. title',
                        //'subtitle'        => 'This is a subtitle. subtitle',
                        //'tickerText'      => 'Ticker text here...Ticker text here...Ticker text here',
                        //'vibrate'         => 1,
                        //'sound' => 1,
                        //'largeIcon'       => 'large_icon',
                        //'smallIcon'       => 'small_icon'*/
                         );
                        $fields = array (
                        'registration_ids' => $registrationIds,
                        'data' => $msg );

                $headers = array (
                        'Authorization: key=' . API_ACCESS_KEY,
                        'Content-Type: application/json' );

                $ch = curl_init();
```

```
                curl_setopt( $ch,CURLOPT_URL,'https://android.googleapis.com/gcm/send');
                curl_setopt( $ch,CURLOPT_POST, true );
                curl_setopt( $ch,CURLOPT_HTTPHEADER, $headers );
                curl_setopt( $ch,CURLOPT_RETURNTRANSFER, true );
                curl_setopt( $ch,CURLOPT_SSL_VERIFYPEER, false );
                curl_setopt( $ch,CURLOPT_POSTFIELDS, json_encode( $fields ) );
                $result2 = curl_exec($ch );
                curl_close( $ch );
                echo $result2;
                echo $registrationIds[0];
                 echo $msg[0];
        }

   $stmt->close();
}

 ?>
```

Luego verificamos el otro código que le ayudara a nuestra aplicación registrar en la base de datos la casa a la que pertenece.

CODIGO

```
13.90.250.14/rapsecurity.php13.90.250.14/rapsecurity.php<?php
// Crear conexion
$conn =mysqli_connect("localhost", "root", "passwd", "db_security");
$id_usuario=$_POST['id_usuario'];
$idevice=$_POST['idevice'];
// Verificar conexion
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";


if ($stmt = $conn->prepare("insert into dispositivos(id_usuario,idevice) values(?,?)")) {
   $stmt->bind_param("ss", $id_usuario,$idevice); // s sambas son strg
   $stmt->execute(); //ejecuta la sentencia
}
?>
```

Google Cloud Messaging

Para empezar a trabajar con este servicio gratuito de Google hay que registrarse como desarrollador y crear una aplicación en el Google console developer y seguir un proceso para crear una aplicación y luego generar un API Key que será necesaria para que la aplicación se comunique con el servicio, también hay que habilitar el servicio (API GCM) e instalar el SDK de Google Play Services.

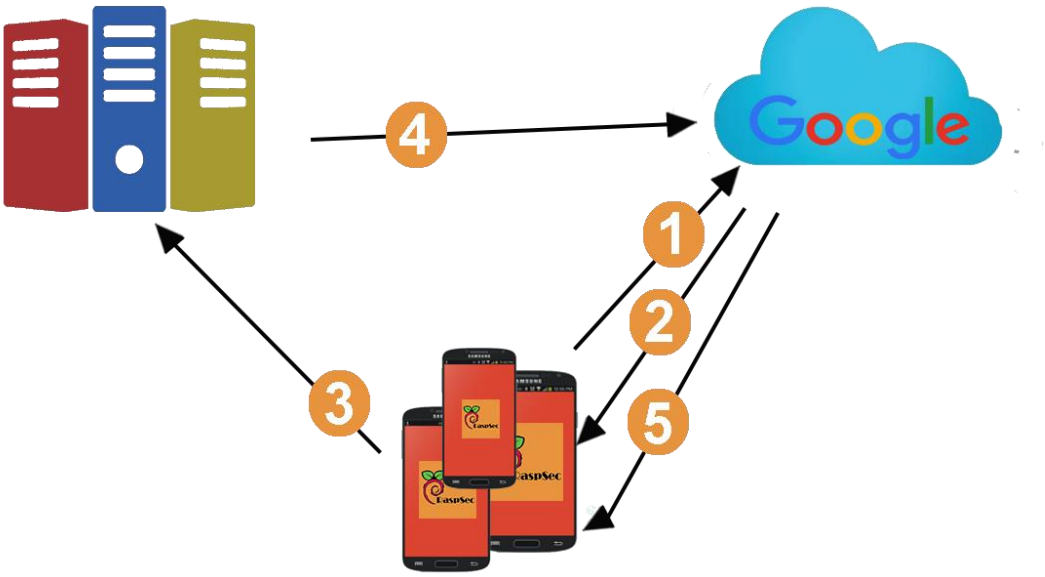


Figura 16, Funcionamiento GCM y Servidor en Azure

1- La aplicación se conecta con el GCM y se registra en él.
2- Luego del registro satisfactorio el GCM manda un token de registro. Este token es único para identificar cada dispositivo.
3- El dispositivo manda el token de registro y se inserta el grupo al que pertenecerá en nuestro servidor para guardarlo en la base de datos MySql.
4- Cuando el servidor de aplicaciones reciba una notificación de la Raspberry se mandará una petición al servidor GCM junto con el token identificado por el grupo ingresado para que este envié un mensaje push al dispositivo.
5- El servidor GCM identifica el dispositivo usando el token de registro e inicializa la petición al mensaje push.
6- El dispositivo recibe la notificación.

Android

Para la aplicación Android utilizaremos el IDE Android Studio, lo instalamos sobre un Linux (Fedora 23). En eta parte analizáremos dos porciones del código que se desarrolló para que la aplicación se ajustara a las necesidades del sistema de seguridad.

Empezaremos con esta parte del código en el MainActivity.java en la cual vemos lo que sucede una vez se inicia la aplicación en el dispositivo, se muestran los controles y se crea el intent que nos ayudara a guardar los datos en nuestra base de datos del grupo al que pertenece el dispositivo.

CODIGO

```
// chequea los servicios de google
   if (checkPlayServices()) {
      //obtenemos info del mantenimiento aislado, sobre si el disp se registró o no
      SharedPreferences settings = getSharedPreferences("RegistroToken", Context.MODE_PRIVATE);

      if (!settings.getBoolean("registrado",false))
      {

         btnEnviar.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
               // Lanza el Intent Service para registrar esta aplicación en GCM
               btnEnviar.setEnabled(false);
              // registra el dispositivo con la función siguiente
               RegistrarDispositivos(txtGrupo.getText().toString());
btnEnviar.setVisibility(View.INVISIBLE); //ocultamos el botón
               txtGrupo.setVisibility(View.INVISIBLE); //ocultamos el textview
            }
         });
      }
      else //si ya está registrado oculta botones
      {
         btnEnviar.setVisibility(View.INVISIBLE);
         txtGrupo.setVisibility(View.INVISIBLE);
      }

   }
}
```

El RegistrationIntenServer.java

En este fragmento encontramos donde se genera el token y se prepara toda la información para ser enviada una vez esté formada la cadena vía POST

CODIGO

```
InstanceID instanceID = InstanceID.getInstance(this);//generamos token
String token = instanceID.getToken(getString(R.string.gcm_defaultSenderId),
      GoogleCloudMessaging.INSTANCE_ID_SCOPE, null);
// [END conseguir token]
Log.i(TAG, "GCM Registration Token: " + token); //para lanar la info en la consola evento logs
SharedPreferences settings = getSharedPreferences("RegistroToken", Context.MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("registrado", true);
editor.commit();// guardar cambios
// Crear los datos variables para enviarlos al servidor
String idCodigo= intent.getStringExtra("idGrupo"); // se recupera la variable del inent anterior pasado a la intención

//se arma la cadena de los parámetros que se enviaran al servidor y que sean guardados en la BD
String data = URLEncoder.encode("id_usuario", "UTF-8")
      + "=" + URLEncoder.encode(idCodigo, "UTF-8");

data += "&" + URLEncoder.encode("idevice", "UTF-8") + "="
      + URLEncoder.encode(token, "UTF-8");

String text = "";
BufferedReader reader=null;

// Send data
try
{

   // Define la URK donde se enviaran los datos
   URL url = new URL("http://13.90.250.14/rapsecurity.php"); // se termina de acoplar para ser enviado al servidor

   // Se envía vía POST

   URLConnection conn = url.openConnection();



 conn.setDoOutput(true);
   OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
   wr.write( data );
   wr.flush();

   // respuesta del servidor
```

```
    reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    StringBuilder sb = new StringBuilder();
    String line = null;

    // leer respuesta del servidor
    while((line = reader.readLine()) != null) {

        sb.append(line + "\n");
    }

    AlertDialog.Builder dlgAlert  = new AlertDialog.Builder(this);
    dlgAlert.setMessage("Dispositivo Registrado!");
    dlgAlert.setTitle("RapSecurity");
    dlgAlert.setPositiveButton("OK", null);
    dlgAlert.setCancelable(true);
    dlgAlert.create().show();
   Toast.makeText(this,sb.toString(),Toast.LENGTH_LONG);
}
catch(Exception ex)
{

}
finally
{
    try
    {

        reader.close();
    }

    catch(Exception ex) {}
}
```

CONCLUSIONES

Concluimos que con el uso de estos equipos de bajo costo se pueden construir sistemas de seguridad eficientes, así como lograr solventar tareas básicas domesticas de manera automatizada.

También podemos afirmar que es posible la utilización de servicios de nube para interconectar el sistema de seguridad realizado con dispositivos móviles de manera remota.

Se comprobó un óptimo desempeño del sistema gracias a las múltiples pruebas reales que se realizaron sobre este.

RECOMENDACIONES

Dentro de las recomendaciones tenemos una increíble cantidad de sugerencias, Ideas y futuras implementaciones a realizar.

- La posibilidad de que la conexión se realice en dos vías permitiendo que desde nuestra aplicación móvil podamos controlar remotamente diferentes tipos de servicios no se descarta.
- Reemplazar el Servicio GCM por un servicio propio al crear sobre el servidor de Windows Azure un Servicio Móvil propio que cuente con las características necesarias para garantizar la comunicación entre el servidor y el Dispositivo/aplicación específica que se desea alcanzar.
- La instalación de cámaras sobre Servo Motores para ejecutar un script que controle el instante en la que detecta el movimiento y se envié una orden a la cámara para tomar una ráfaga de imágenes y enviarlas al dispositivo.
- La implementación de switches para puertas para mantener un control de que puerta fue abierta en que momento.
- No se descarta llegar a un convenio con la policía para mantener un servidor donde ellos reciban notificaciones en tiempo real y gestionar la seguridad ya sea de instalaciones domiciliares tanto como edificios del gobierno y/o privadas.
- La creación de un sitio web local donde el usuario podrá verificar los Logs e historial del Sistema de seguridad.

BIBLIOGRAFIA

Gus. (Oct 1, 2015). Raspberry Pi Motion Sensor using a PIR Sensor. Marzo 2016, de pimylifeup Sitio web: https://pimylifeup.com/raspberry-pi-motion-sensor/

mct. (Sep 1, 2006). SPI - Serial Peripheral Interface. Marzo 2016, de mct.net Sitio web: http://www.mct.net/faq/spi.html

Francesc. (Ago 19 2015). Cámara de seguridad con Raspberry Pi (SecurOS). Mayo 2016, de fpaez.com Sitio web: http://fpaez.com/camara-de-seguridad-con-raspberry-pi-securos/

Liz Upton. (Jul 2013). PrivateEyePi – a DIY home alarm system. Mayo 2016, de raspberrypi.org Sitio web: https://www.raspberrypi.org/blog/privateeyepi-a-diy-home-alarm-system/

elnuevodiario.com.ni. (23 Marzo 2015). Servicio de video-vigilancia para Pymes. Mayo 2016, de elnuevodiario.com.ni Sitio web: http://www.elnuevodiario.com.ni/economia/355987-servicio-video-vigilancia-pymes/

elnuevodiario.com.ni. (13 Marzo 2015). Claro ofrece vigilancia para Pymes. Mayo 2016, de elnuevodiario.com.ni Sitio web: http://www.elnuevodiario.com.ni/economia/empresas/355204-copa-airlines-aerolinea-mas-puntual-america-latina/

Wikipedia. (sin fecha). Universal asynchronous receiver/transmitter. Mayo 2016, de wikipedia.org Sitio web: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

MICROCONTROLLER UART TUTORIAL. (Sin fecha), de societyofrobots.com Sitio web: http://www.societyofrobots.com/microcontroller_uart.shtml

C. GORDON BELL· J. CRAIG MUDGE· JOHN E. McNAMARA . (Ago 1979). Bell Computer Engineering. U.S.A.: Digital Press.

GPIO: Raspberry Pi Models A and B. Sin Fecha, de raspberrypi.org Sitio web: https://www.raspberrypi.org/documentation/usage/gpio/

@RogueHAL13, @ardadev & @nixijav. (Sin Fecha). Aprende los pines de la Pi y sus add-ons. Junio 2016, de pinout.xyz Sitio web: https://es.pinout.xyz/pinout/pin35_gpio19

More on Python. Junio 2016, de raspberrypi.org Sitio web: https://www.raspberrypi.org/documentation/usage/python/more.md

Sensor de movimiento PIR HC-SR501. Junio 2016, de http://electronilab.co Sitio web: http://electronilab.co/tienda/sensor-de-movimiento-pir-hc-sr501/

Micro Power PIR Motion Detector IC. Junio 2016, de seeedstudio Sitio web: http://www.seeedstudio.com/wiki/images/2/2f/Twig_-_BISS0001.pdf

Fresnel zone plate theory and equations. Junio 2016, de zoneplate Sitio web: http://zoneplate.lbl.gov/theory

Cibernat. (Sin Fecha). Computación en la nube . Junio 2016, de Cibernat Sitio web: http://cibernat.com/articulos/computacion-en-la-nube

Qusay F.. (Feb 2011). Demystifying Cloud Computing. Junio 2016, de DATA MINING Sitio web: http://static1.1.sqspcdn.com/static/f/702523/10181434/1294788395300/201101-Hassan.pdf?token=YpUn07SQ1eom0uFoqxzAQ4u5Bbo%3Dhttps://tools.ietf.org/html/rfc6120

Firebase Cloud Messaging. Junio 2016, de Firebase Sitio web: https://firebase.google.com/docs/cloud-messaging/#proximos_pasos

(Mayo 18, 2016). Google Cloud Messaging: Overview. Junio 2016, de Google developer Sitio web: https://firebase.google.com/docs/cloud-messaging/#proximos_pasos

(November 5, 2007). Industry Leaders Announce Open Platform for Mobile Devices. Junio 2016, de open handset alliance Sitio web: http://www.openhandsetalliance.com/press_110507.html

ANEXOS

Comunidades en Tecnologías de código abierto en Nicaragua

http://www.debian.org.ni/

http://fedora.org.ni/

http://guluca.org/

http://www.ubuntu.org.ni/

http://rubyni.com/

https://groups.drupal.org/nicaragua

https://wpnicaragua.org/

http://mapanica.net/#8/13.000/-85.000

http://pythonicaragua.github.io/

http://mozilla-nicaragua.blogspot.com/

http://www.enlinux.org/

Figura 17, Logo de comunidades de Tecnologías de código abierto en Nicaragua, Mozilla Nicaragua, Comunidad de software libre bilwi, Ubuntu Nicaragua, GUL-UCA, Clinux, Fedora Nicaragua, Rubi-NI, Linux Enterprise Nicaragua. Wordpress Nicaragua, Debian Nicaragua, Drupal Nicaragua, Python Nicaragua.

Configuración GCM

La creación de una nueva aplicación es muy sencilla pero los pasos son un poco confusos. Esta es una breve guía de creación de una aplicación y configuración de GCM.

1. **Nos dirigimos al sitio de desarrolladores de Google. Nos podemos loguear con nuestra cuenta de Gmail.**

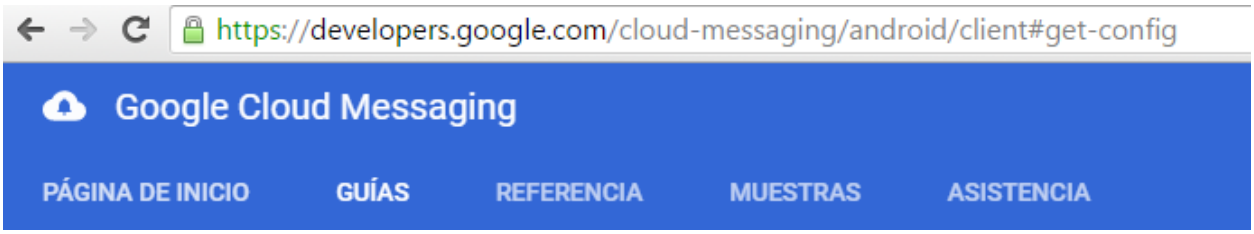


Figura A

2. **En la página principal del sitio damos clic en "Prueba con Android" o bien "Try it on Android".**
3. **Lo primero que hay que hacer es conseguir el proyecto de ejemplo. Para ello usaremos:**

```
$git clone https//github,com/googlesamples/google-services.git
```

4. **Abrimos Android studio File>Open y buscamos el lugar donde clonamos el repositorio de Google Services. Y abrimos google-services/android/gcm**
5. **Ahora necesitamos el archivo de configuración en el mismo sitio donde nos encontramos.**

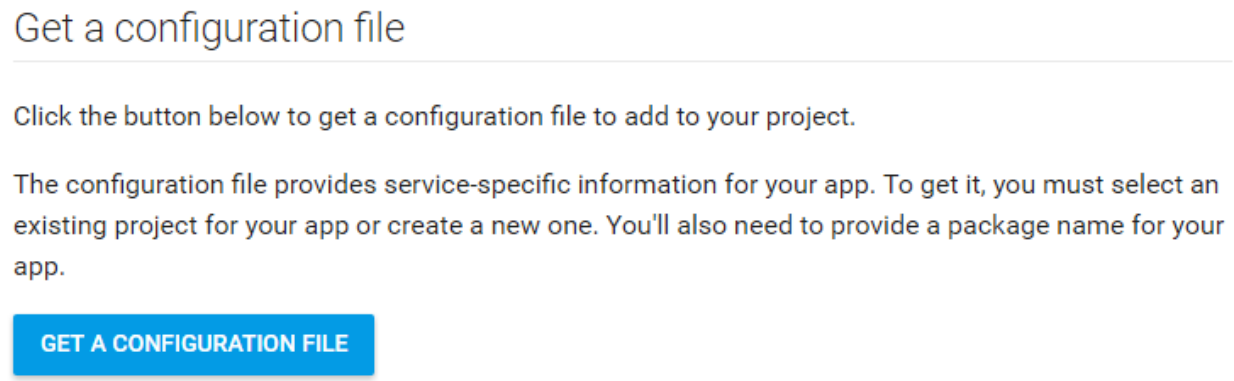


Figura B

6. **Estando ahí seleccionamos el nombre de nuestra aplicación y el nombre de empaquetado que usaremos en nuestro proyecto de Android studio. Aquí el proyecto fue agregado al Google Developers Console**.

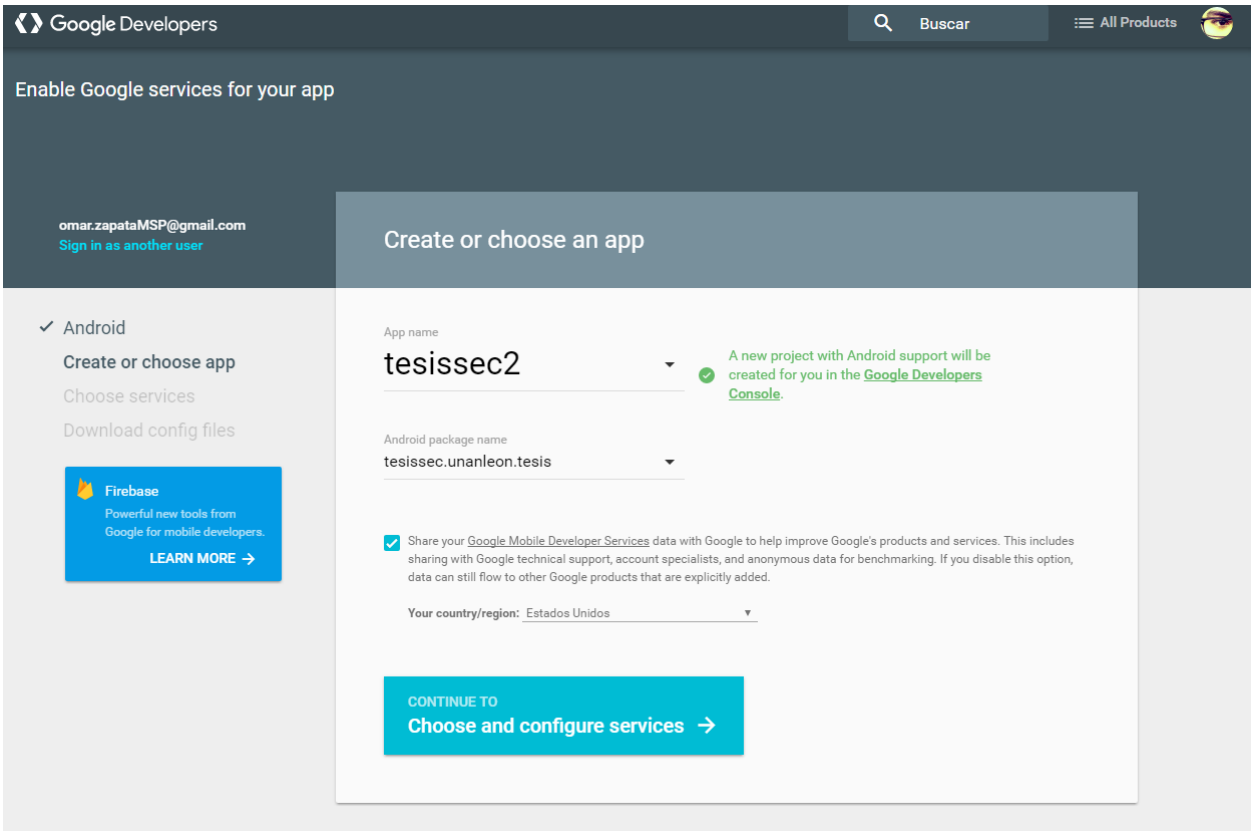


Figura C

7. **Aquí seleccionamos los servicios que vamos a utilizar de la plataforma de aplicaciones para desarrolladores. En nuestro caso habilitaremos el Servicio de Google Cloud Messaging.**

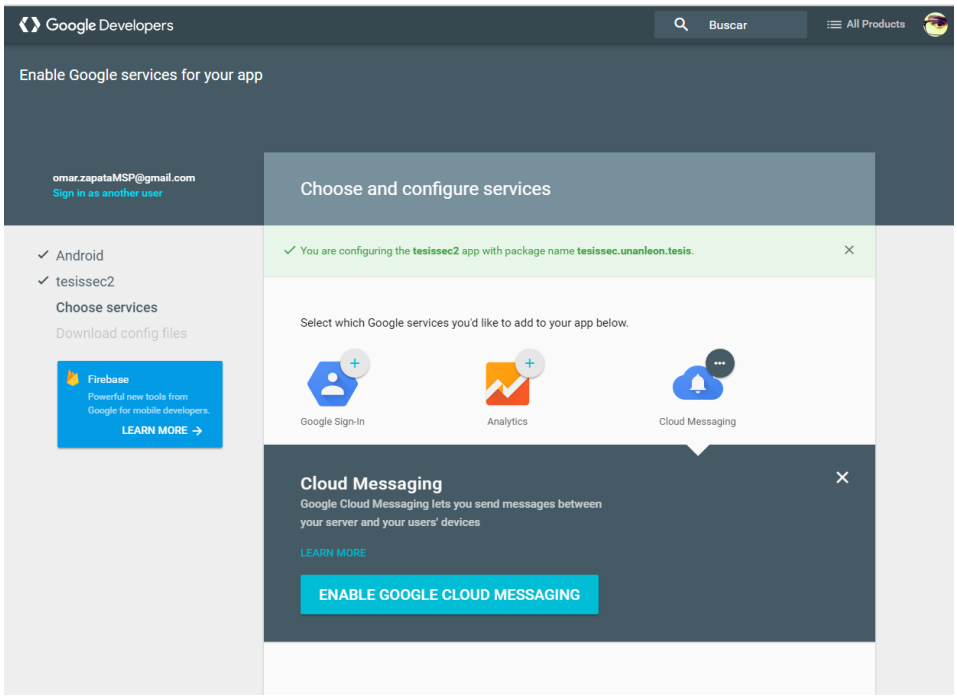


Figura D

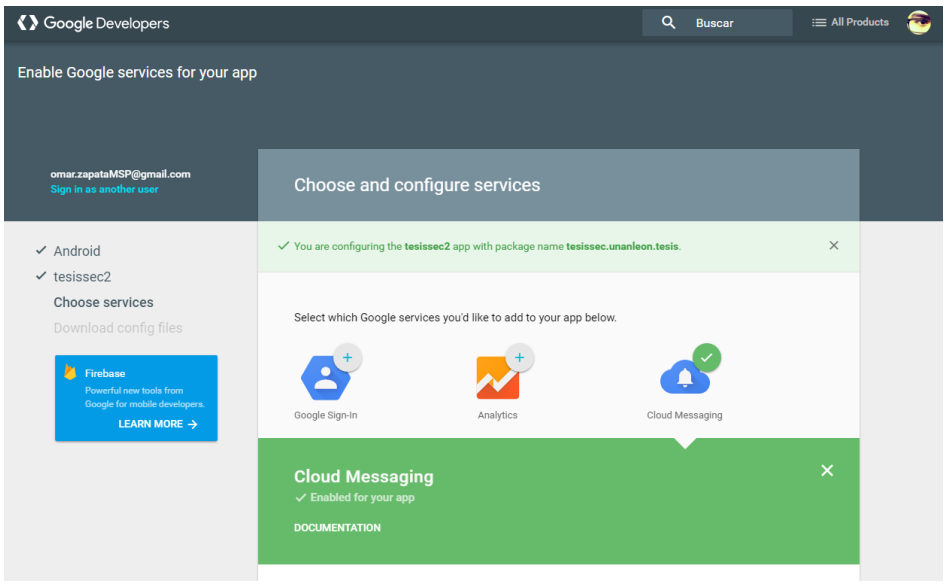


Figura E

8. **Una vez habilitado el sitio nos dará el Server Api Key y el Sender ID con el que identificaremos nuestra aplicación en el Google Connection Server. Y generaremos nuestros archivos de configuración.**

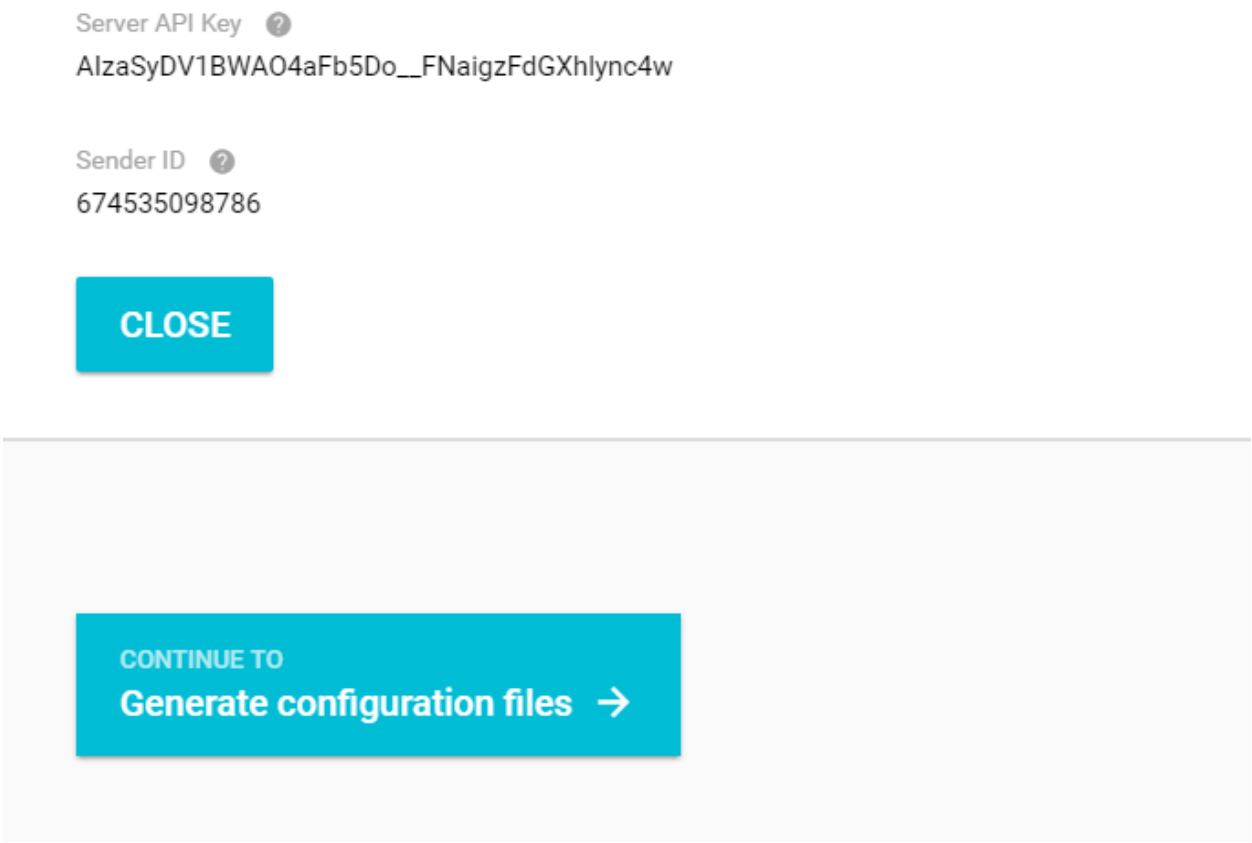


Figura F

9. **Luego descargamos el archivo JSON el cual contiene detalles de configuración así como keys e identificadores para los servicios que habilitamos.**

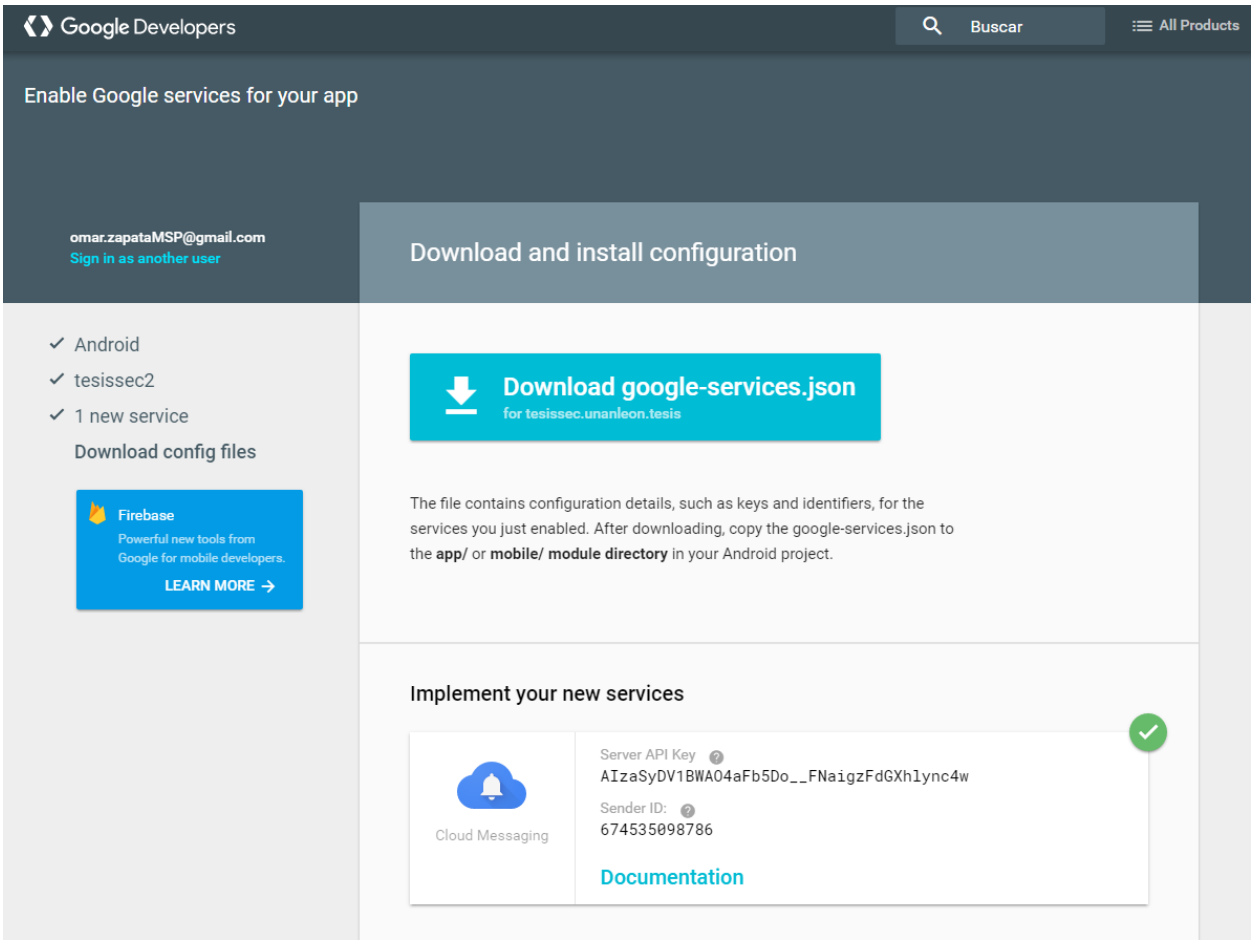


Figura G

10. **Una vez que tenemos el archivo JSON lo copiamos a la carpeta donde se encuentra nuestro proyecto.**
11. **Realizamos los cambios correspondientes a nivel de proyecto y a nivel de aplicación en los build.gradle.**

Agregamos la dependencia a nivel de proyecto:

classpath 'com.google.gms:google-services:2.0.0-alpha6'

Agregamos el plugin a nivel de aplicación:

apply plugin: 'com.google.gms.google-services'

12. Tendremos que instalar el SDK de Google Play Services en nuestro gestor de paquetes de Android Studio.

El punto clave es que se debe referenciar la librería simplemente agregando un archivo .jar, si estas usando Android studio esta es la dependencia que se debe agregar a la sección de dependencias en el “build.gradle” de tu aplicación.

```
dependencies {
  compile "com.google.android.gms:play-services-gcm:9.0.0"
}
```

13. En este punto hay que editar el App Manifest

Se agregan lo siguiente a la app manifest.

- Tu <nombre de paquete de aplicación> + “.permission.C2D__MESSAGE” el permiso para evitar que otras aplicaciones Android registren y reciban los mensajes de aplicación. El nombre de permisos debe ser exactamente igual que en este ejemplo. De otra forma la aplicación no recibirá ningún mensaje.
- Una declaración de GcmReciever, la cual maneja los mensajes enviados desde el GCM a tu aplicación. Porque este servicio necesita permiso para recibir mensajes desde GCM, se agrega.

Com.google.android.c2m.permission.SEND al receiver.

- Una declaración de GcmListenerService, el cual habilita varios aspectos que manejan los mensajes así como detección de diferentes tipos de mensajes desde el servidor. Determinando el estado de los mensajes que van hacia el servidor y automáticamente mostrando notificaciones simples en nombre de la aplicación.
- Un servicio que extiende InstanceIDListenerService, para manejar la creación, rotación y la actualización del registro de tokens.
- Opcionalmente, el permiso android.permission.WAKE_LOCK es la aplicación que evita que el procesador se duerma cuando un mensaje es recibido.
- Si la característica del GCM es crucial para el funcionamiento de tu aplicación hay que asegurarse de establecer android: minSdkVersion=”8” o mayor en el manifest. Con esto nos aseguramos que la aplicación no puede ser instalada en entornos donde no funcionaría apropiadamente.

14. Ahora podemos lanzar nuestra aplicación de ejemplo.

Primero hay que asegurarse de que el API key ha sido provista en el GcmSender.java. Dentro de Android Studio damos clic en el botón lanzar y seleccionamos el dispositivo asociado. Una vez que la aplicación carga completamente en el dispositivo podemos correr el siguiente comando para enviar una notificación a todas las instancias registradas a nuestra aplicación de forma de prueba.

./gradlew run –Pmsg=”<message>” en Linux o Mac.

.\gradlew.bat run –Pmsg=”<message>” en Windows.

Como funciona.

Por cada dispositivo, primero recibe un token de registro para GCM desde la Instancia API ID. Este token es usado para identificar la instancia en el dispositivo. Se recupera el token usando IntentService como es mostrado.

Instalación del sistema operativo en la Raspberry pi utilizando NOOBS

Paso 1. Descarga del Sistema Operativo

Este se descarga directamente del sitio oficial de Raspberry Pi (http://www.raspberrypi.org/downloads), en este link encontrara que hay diferentes versiones disponibles. Si no sabe cuál descargar, le recomendamos que inicialmente utilice NOOBS, el cual contiene las siguientes versiones de sistemas operativos:

- Archlinux
- OpenELEC
- Pidora
- RISC OS
- RaspBMC
- Raspbian

Paso 2. Descomprimir el archivo ZIP

Seleccione el archivo con el nombre NOOBS_vx_x.ZIP . Después de unos segundos el archivo empezara a descargarse, esto tomara bastante tiempo ya que el archivo es de 1.1 GB. Tenga en cuenta que este es un archivo ZIP una vez más, es necesario tener una herramienta para descomprimir estos archivos.

Paso 3. Descomprimir el archivo en el escritorio

Una vez el archivo ha descargado por completo, descomprima el archivo en lugar conocido como en el “escritorio” en una carpeta que puede llamar “NOOBS”.

Paso 4. Copiar o mover el contenido de la carpeta en la tarjeta SD

Cerciórese de que la tarjeta SD está en la ranura de su PC o portátil. Luego abra la carpeta donde descargo el sistema operativo. Ahora proceda a copiar o mover todo el contenido, le recomendamos copiar el contenido. Para eso, seleccionamos todos los archivos (presione ***Ctrl+a***), luego copiamos (***Ctrl+c***) o cortamos (***Ctrl+x***) los archivos en la tarjeta SD. Como segunda opción, puede seleccionar todos los archivos (***Ctrl+a***) y luego hacer clic derecho > enviar a > “G:\xxxx”. Esta última acción moverá todos los archivos del Raspberry Pi a la memoria, sin dejar copia en el disco duro.

Con estos pasos, el sistema operativo ha sido copiado en nuestra tarjeta SD y ahora podemos proceder a instalar el sistema operativo.

Instalación y configuración inicial del Raspberry Pi (raspi-config)

Una vez la tarjeta ha sido preparada y NOOBS ha sido copiado, el sistema operativo se reinicia y el primer programa que se ejecuta es Pi Recovery, este programa sirve para instalar la versión del sistema operativo deseada. El siguiente tutorial muestra la instalación y configuración de Raspbian.

En la primera pantalla inicial, seleccione Raspbian para instalar el sistema operativo. Este proceso tomara entre 15-20 minutos.

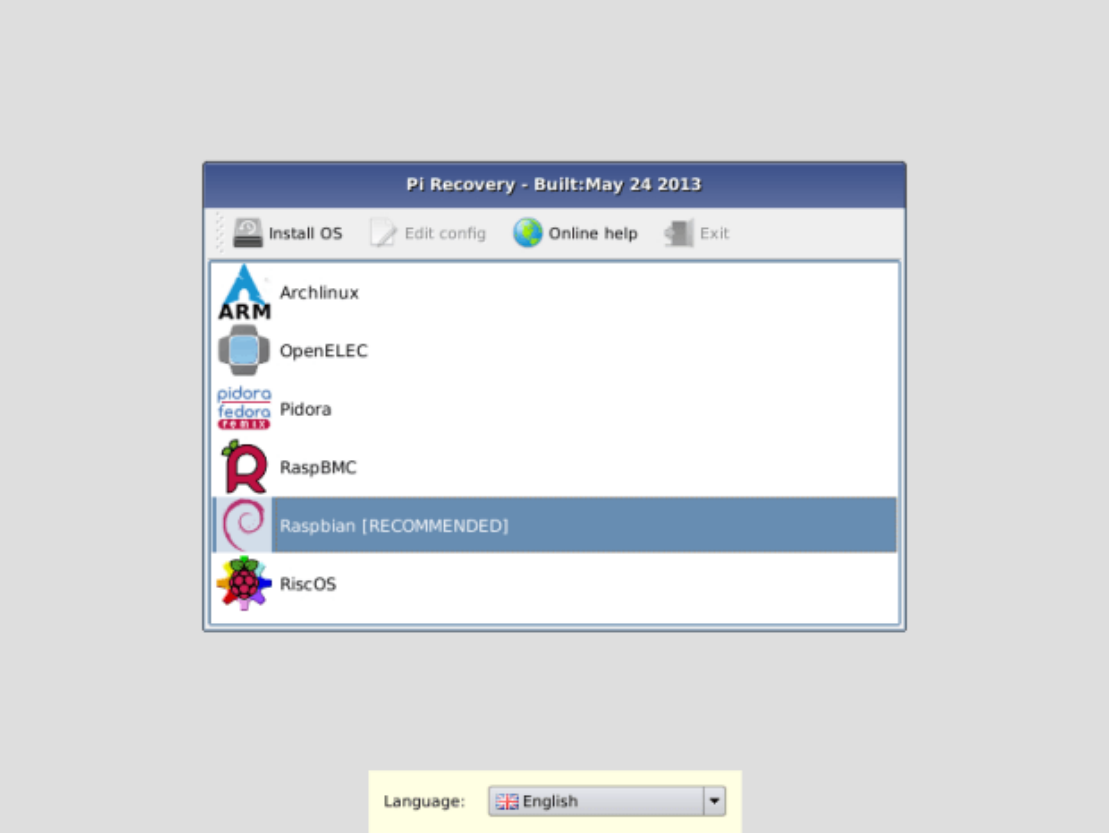


Figura H, Pantalla inicial, seleccionar sistema operativo

Después que Raspbian “Wheezy” ha sido instalado, toca configurar el Sistema operativo para que funcione en español. Los siguientes pasos le muestra las diferentes opciones disponibles.

Configuración inicial de Raspbian

Con el primer inicio de nuestra **Raspberry Pi con Raspbian**, se nos mostrará **rapi-config**, una herramienta de configuración de Raspbian para configurar las opciones principales, que les describo a continuación:

1 – Expandir el sistema: Esto se encarga de que el sistema de archivos de Raspbian ocupe todo el espacio de la tarjeta SD (si se instala con NOOBS creo que no es necesario hacerlo).

2 – Cambio de contraseña: Por defecto el usuario es pi y la contraseña Raspberry, recomendable cambiarla.

3 – Activar inicio en modo escritorio: Con esta opción podemos elegir que nuestra Raspberry Pi inicie en modo de escritorio directamente.

4 – Opciones de internacionalización: Con esto podemos configurar el idioma de nuestra Raspberry Pi, la localización del teclado que usamos, etc.

5 – Activar cámara: Esto es para activar la cámara de Raspberry Pi (que hay que adquirir aparte).

6 – Añadir a Rastrack: Esto es para tema de clusterización.

7 – Overclock: Opciones de Overclocking, para subir la velocidad del microprocesador (con el correspondiente aumento de temperatura y la disminución del tiempo de vida que supone).

8 – Opciones Avanzadas: Aquí podemos cambiar el nombre de la máquina; elegir el uso de memoria para la GPU; y entre otras opciones más, lo que me parece más importante resaltar, activar el servidor SSH para poder acceder a la Raspberry remotamente vía SSH desde cualquier terminal.

9 – Información sobre raspi-config

Todas estas opciones se acceden con un menú controlable desde el teclado, y son bastante sencillas y explicativas. **Consejo:** si no tenemos claro que hace alguna de estas opciones, mejor no tocarla.

Primer contacto con Raspbian

Una vez salgamos de configurar por primera vez nuestro **Raspbian** accederemos a una consola o terminal, en el que podremos ejecutar comandos. Al ser una **distribución Linux** basada en **Debian** (como Ubuntu), muchos de los comandos son iguales que en el resto de distribuciones Linux y especialmente en el caso de **Ubuntu y Debian**. Algunos **comandos** que nos vendrán bien con **Raspbian** son los siguientes:

- **raspi-config**: Nos mostrará las opciones de configuración.
- **startx**: Para iniciar el modo gráfico con el escritorio.
- **omxplayer <nombre-de-archivo>**: Para reproducir archivos multimedia desde consola.

Hay que comentar que, aunque **Raspbian** tenga **modo Escritorio**, no es como en Ubuntu, es mucho más limitado y no corre tan liviano. **Manejarnos con soltura con la consola nos vendrá bien para usar una Raspberry Pi y aprovechar todo su potencial**.

Creación de la base de datos en - LAMP.

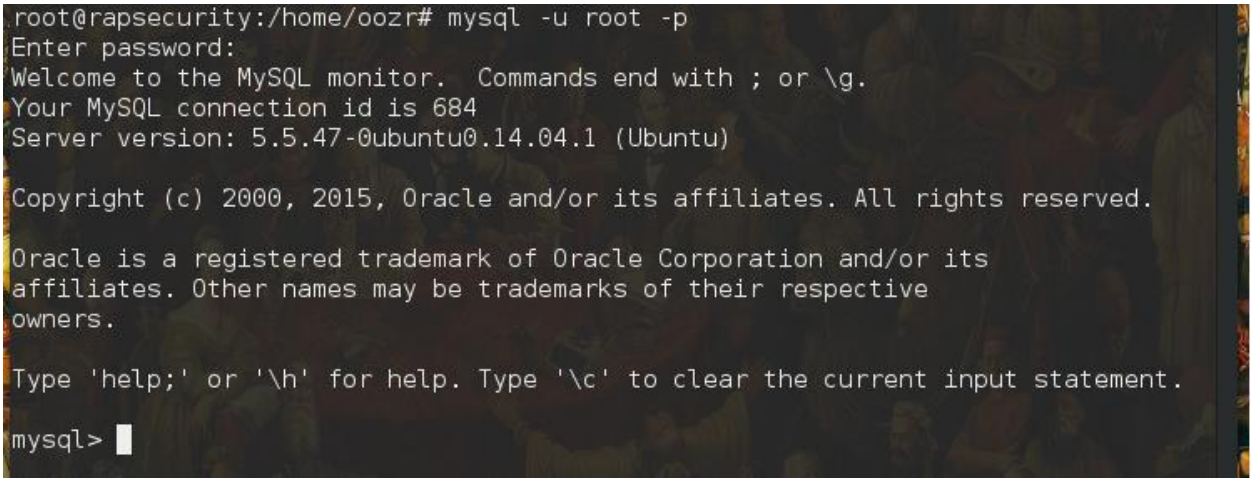
En esta parte del proyecto se trabajó conectados vía SSH al servidor en La Nube desde el terminal Linux.

Paso 1: conectarse vía SSH.

$sudo ssh oozr@<dirección ip>

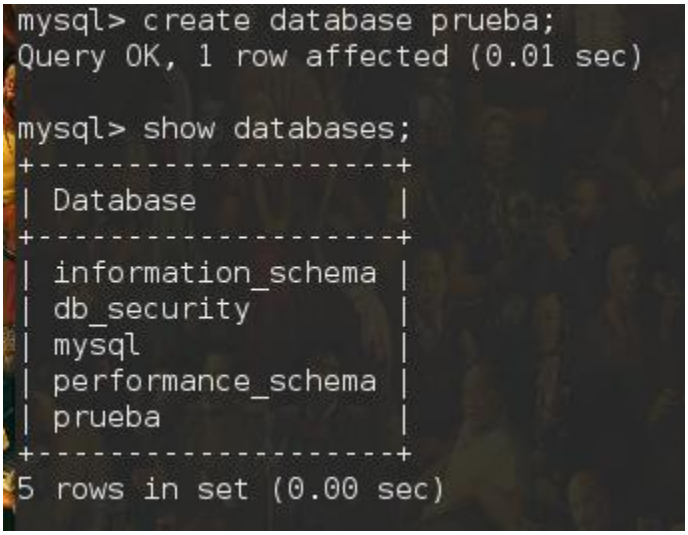
Paso 2: una vez dentro de nuestro servidor proseguiremos a conectarnos con la base de datos.

$sudo mysql –u root –p



Paso 3: Proseguimos en la creación de la base de datos que llamaremos prueba.

mysql> create database prueba;

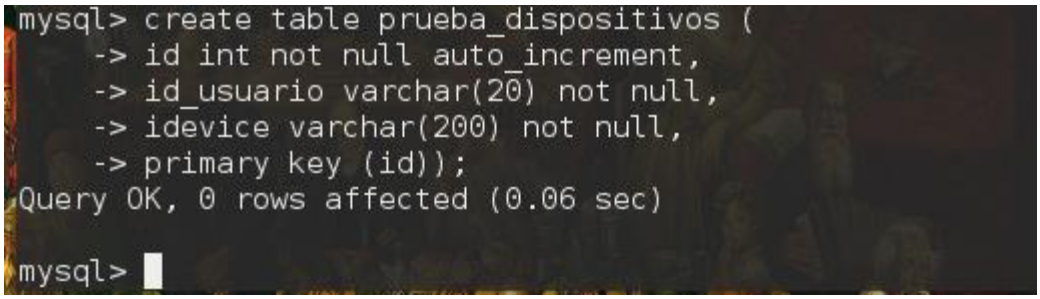


Paso 4: Seleccionamos la base de datos que recién creamos para crear las tablas en ella.

mysql> use db_security

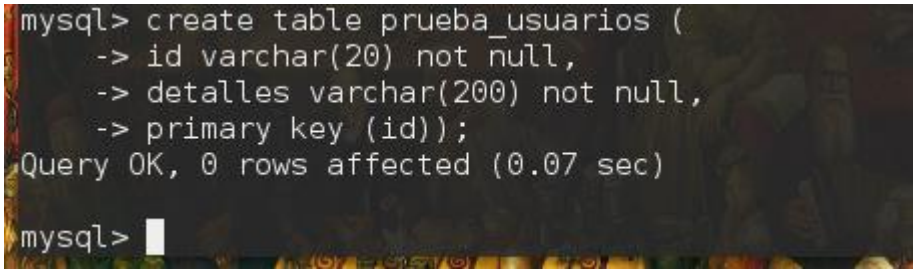
Paso 5: creamos la primera tabla y sus campos. Crearemos una tabla llamada usuarios que tendrá un campo "ID" y uno llamado "Detalles" tabla que se usara para llevar el control cuando un usuario tenga más de un dispositivo a notificar.

```
mysql> create table usuarios (
                id varchar(20) not null,
                detalles varchar(200) not null,
                primary key (id));
```



Paso 6: procederemos a crear la siguiente tabla que mantendra los dispositivos con sus respectivos Tokens y grupos.

mysql> create table prueba_dispositivos (
id int(5) not null auto_increment,
id_usuario varchar(20), not null,
idevice varchar(200) not null,
primary key(id));



Con esto basta para tener una pequeña base de datos de usuarios que podremos escalar luego. Podemos ver el contenido de nuestra base de datos con el comando.

Mysql> show tables;

INDICE DE IMÁGENES

Configuración GCM y Raspberry Pi (Anexos)