UNIVERSIDAD NACIONAL DE INGENIERIA

RECINTO UNIVERSITARIO "SIMON BOLIVAR"

FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

INFORME FINAL DEL TRABAJO MONOGRAFICO PARA OPTAR AL TITULO DE INGENIERO ELECTRONICO

TEMA:

Diseño e Implementación de un Sistema de Notificación y Control de Alarmas por medio del servicio SMS de la red Celular GSM para Dispositivos Móviles.

AUTORES:

Br. Marvin Alejandro Obando Buitrago. Carnet # 2003-18216

Br. Hernán José Ramírez Guerrero. Carnet # 2003-18049

TUTOR:

Ing. Cedrick Elksnherr Dalla-Torre Parrales.

5 de agosto de 2016.

Managua, Nicaragua

DEDICATORIA

Es un gusto para nosotros dedicar esta trabajo monografico:

Con mucho cariño y admiración a nuestros padres que con tanto esfuerzo y dedicación nos han apoyado a lo largo de nuestra vida, y más en la etapa estudiantil, dándonos siempre ejemplo de perseverancia.

Queremos agradecer a todas las personas que estuvieron dedicadas en este proyecto con su apoyo, recomendaciones e ideas. Con mención especial al Ing Gerardo Mena Buitrago y el Ing.Noel Zelaya Perez.

RESUMEN

El concepto de movilidad está cada día más extendido en las comunicaciones. Esto supone una gran ventaja porque permite disponer del servicio en cualquier momento y en cualquier lugar de una forma más personalizada. En el presente trabajo monográfico se expone el diseño e implementación de un sistema de notificación y control de alarmas por medio del servicio de mensajes cortos SMS; a este sistema propuesto se puede acceder remotamente desde cualquier teléfono celular o dispositivo que tenga conexión con la red GSM. Se diseñó un sistema de propósito general, el cual con pequeñas modificaciones, dependiendo de la aplicación, puede ser utilizado en una amplia gama de áreas donde se requiera el control y monitoreo de eventos o magnitudes físicas de una forma autónoma.

El sistema está compuesto por sensores y actuadores que pueden ser controlados por medio de mensajes de texto de cualquier operador móvil. Con esta investigación, se logra presentar el servicio SMS desde un punto de vista práctico, enfocado al desarrollo de aplicaciones basadas en él.

Este documento propone la adopción de modelos de Hardware libre para desarrollar el sistema. Esta es una forma innovadora donde los diseños se distribuyen bajo el amparo de licencias libres, estos ofrecen las mismas cuatro libertades del software libre pero aplicado al plano del Hardware. Están disponibles para que cualquiera los pueda usar, modificar y distribuir.

En la primera parte, se describe de forma general el sistema GSM y la arquitectura de la red subyacente, el servicio SMS, el funcionamiento del modem GSM y la aplicación de los comandos AT. En la segunda, se describen los criterios para la selección del Hardware y Software para el desarrollo de la aplicación, también se describe el diseño e implementación del sistema de notificación y control de alarmas. Se muestran los módulos que conforman el sistema y el diagrama de flujo del programa. En la tercera y última parte, se describe el análisis y presentación de resultados obtenidos en el desarrollo de este proyecto de tesis. Finalmente se describen nuestras conclusiones y recomendaciones para futuros estudios.

ABSTRACT

The concept of communication mobility is becoming more widespread day. This is a great advantage because it allows service available anytime and anywhere in a more personalized way. This thesis details the design and implementation of a system of notification and alarm monitoring via short message service SMS; this proposed system can be accessed remotely from any cell phone or device that has a connection with the GSM network, general purpose system, which with minor modifications, depending on the application, can be used in a wide range of areas where control and monitoring of events or physical quantities in an autonomous manner required was designed.

The system is comprised of sensors and actuators that can be controlled by means of text messages from any mobile operator. With this research, it is possible to present the SMS service from a practical point of view, focused on the development of applications based in the SMS service.

This document proposes the adoption of Free Hardware models to develop the system. This is an innovative way in which the designs are distributed under open source licenses, this license offer the same four liberties of free software but applied to the plane of Hardware. They are available for anyone to be able to use, modify and distribute.

The first part of this thesis generally describes the GSM system and the underlying network architecture, the SMS service, the operation of GSM modem and application of AT commands. In the second, the criteria for the selection of hardware and software for application development are described; the design and implementation of the reporting system and control alarms and the modules that make up the system and program flow diagram are shown. In the third and final part, the analysis and presentation of results in the development of this thesis project is described. Finally our conclusions and recommendations for future studies are described.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

El servicio de mensajes cortos, SMS, está teniendo un gran auge en nuestros días, debido a que es utilizado por una gran cantidad de usuarios a nivel mundial. En el presente documento se utiliza este servicio de una manera práctica para desarrollar un sistema de notificación y control de alarmas para dispositivos móviles.

Este proyecto consiste en un sistema de control en el hogar que permita una interacción bidireccional entre el usuario y el sistema desde cualquier terminal móvil. Los ambientes a controlar serán: acceso a la puerta, encendido y apago de luces, falla de energía comercia y detección de intrusos.

Se inicia con una descripción teórica de la red GSM en general y del servicio SMS en particular. Se describen los protocolos necesarios para la comunicación, centrándose en la capa de transferencia de mensajes, que es la que se utiliza desde las aplicaciones para lograr establecer comunicación entre el teléfono celular y la computadora. Se muestra como es la interfaz entre las aplicaciones y el servicio SMS utilizando un modem GSM, y cómo es posible controlarlo mediante los comandos AT, por medio de una pasarela de mensajería (Gateway).

Luego se muestran los módulos que conforman el sistema, los criterios que se utilizaron para la selección del Hardware y Software implementados en el proyecto los cuales al ser diseñados bajo licencias Open Source[1] nos permiten la integración de múltiples tecnologías a un bajo costo.

Finalmente se muestra un ejemplo de aplicación del sistema de forma sencilla, validando el correcto funcionamiento de control y envió de alertas vía mensajes de textos, obteniéndose los resultados esperados. Se trata de una aplicación básica a partir de la cual se pueden desarrollar aplicaciones mucho más complejas, se provee a otros investigadores interesados la mayor cantidad de información sobre este tema.

[1] El código abierto es el software distribuido y desarrollado libremente.

ANTECEDENTES

A lo largo de la historia han existido diferentes situaciones en las que ha sido necesaria una comunicación a distancia, como en la guerra y en el comercio. Las telecomunicaciones como estudio unificado de las comunicaciones a distancia es una idea reciente que data de mediados del siglo XX. Con la evolución de la tecnología, los sistemas de comunicación digital han sido las que mejor se destacan, en este proceso de transmisiones Inalámbricas. Packet Radio[2], Bluetooth[3], microondas entre otras [1]; Debido a que en nuestro país el uso de internet no posee un rango de penetración en todo el territorio nacional y su costo es muy elevado para la mayoría de la población. La solución más viable es utilizar la red de telefonía inalámbrica GSM[4] debido a que tiene cobertura en la mayor parte del territorio y no se tiene que implementar una nueva infraestructura de redes de comunicación privadas. Utilizando el servicio de mensajes de texto (SMS) como mecanismo de transmisión de datos. A continuación se brinda una pequeña reseña histórica sobre las redes móviles.

El estándar GSM fue desarrollado a partir del año 1982. En la conferencia de telecomunicaciones CEPT de ese año fue creado el grupo de trabajo Groupe Special Mobile o GSM, cuya tarea era desarrollar un estándar europeo de telefonía móvil digital. Se buscó evitar los problemas de las redes analógicas de telefonía móvil, que habían sido introducidos en Europa a fines de los años 1950, y no fueron del todo compatibles entre sí a pesar de usar, en parte, los mismos estándares [2].

La idea de añadir la mensajería de texto a los servicios de usuarios móviles era latente en muchas comunidades de servicios de comunicación móviles al principio de los años ochenta. Los expertos de varias de aquellas comunidades contribuyeron en las discusiones sobre cuáles deberían ser los servicios GSM. En un principio

[2]Es un sistema de comunicación digital. Que emplea un sistema basado en las emisoras de radioaficionados.
[3]Es una especificación industrial para redes inalámbricas de área personal. Que posibilita la transmisión de voz y datos entre diferentes dispositivos.
[4]Sistema Global para las comunicaciones Móviles.

el SMS se pensó como una manera de avisar al usuario, por ejemplo, de llamadas perdidas o mensajes en el buzón de voz [2].

En el año 1992 las primeras redes europeas de GSM-900 iniciaron su actividad, y el mismo año fueron introducidos al mercado los primeros teléfonos móviles GSM, siendo el primero el Nokia 1011 en noviembre de ese año. En los años siguientes, el GSM compitió con otros estándares digitales, pero se terminó imponiendo también en América Latina y Asia. Al igual que en otros países [3].

El desarrollo del Global System for Mobile Communication, en 1996 hizo posible la transmisión de datos desde un móvil utilizando la red telefónica inalámbrica. El Servicio de Mensajes Cortos (SMS) posibilita el envío y recepción de pequeños mensajes de texto que no superen los 160 caracteres a través de la red telefónica inalámbrica. Hasta entonces, el acceso a servicios corporativos o brindados por terceros se encontraba limitado a las conexiones cableadas tradicionales. A partir de la introducción del GSM, la capacidad de acceso desde un conjunto de dispositivos cuyo número supera a las computadoras tradicionales ha propulsado la utilización de nuevos mecanismos de trabajo en el área del desarrollo e interconexión de aplicaciones [4].

A lo largo de los años se han desarrollado grandes avances tecnológicos en el área de seguridad. Gracias a estos se han implementado diferentes formas para gestionar el funcionamiento de la seguridad en el hogar. Esto se ha logrado mediante dispositivos electrónicos que desempeñan funciones específicas de control, los cuales se encuentran conectados a un módulo central, luego este módulo recibe señales de los dispositivos conectados y en caso de una posible invasión alerta a la empresa encargada.

Existen equipos comerciales para aplicaciones de alarmas contra intrusos en hogares, los cuales ofrecen la posibilidad de notificar al usuario hechos como la activación de alarmas, equipos, etc. Este es el caso de la compañía de seguridad C&B por ejemplo, la cual brinda la posibilidad de monitoreo de alarmas contra

intrusos. Con alternativas de comunicación con la estación central de monitoreo por medio de la telefonía convencional, GSM/GPRS y comunicación IP. Además de contar con el módulo de acceso Web para que los clientes revisen el comportamiento del sistema a través de internet. Estos resultan considerablemente más caros que la solución propuesta.

JUSTIFICACION

Debido a la gran difusión de los teléfonos celulares y computadoras, Desarrollamos esta investigación para facilitar su interconexión de tal manera que con ambos dispositivos trabajando juntos se puedan desarrollar nuevos sistemas en Ingeniería. La red GSM presenta una amplia cobertura en todo el país, consideramos como un medio ideal para ser usado como plataforma para la comunicación de nuevas aplicaciones.

Es muy atractivo como ingenieros utilizar esta herramienta como un medio para generar sistemas que sean útiles y tengan aplicación en el mercado actual, es por esto que crear un sistema que por medio de mensajes de texto SMS se puedan controlar eventos. No es algo fuera de nuestro alcance, y amplia nuestra capacidad de interacción con el medio. Un ejemplo puede ser pedir el estado del clima por SMS, también se podría utilizar en el control de equipos de refrigeración, etc. Son muchos los problemas que podrían encontrar solución con una herramienta de este tipo.

También la necesidad de tener un sistema de seguridad en nuestros hogares, hoy en día es de mucha utilidad, pero debido a su alto costo muchas veces no es factible contar con este servicio, este proyecto propone un prototipo de seguridad para viviendas mediantes mensajes de texto debido a que en la actualidad existe un alto porcentaje de personas con el servicio de telefonía celular.

Este proyecto tiene como utilidad facilitar a las personas habitantes de un hogar tener un control vía remota del encendido y apagado de luminaria, notificación de cortes de energía comercial y para la detección de intrusos. Usando hardware de muy bajo costo y software libre.

OBJETIVOS

Objetivo General

Implementar un sistema de notificacion y control de alarma por medio del servicio de mensajes SMS utilizando la red GSM, a través de la integración de controladores electrónicos y un microcomputador.

Objetivos Específicos

- Establecer comunicación entre los dispositivos móviles y el sistema de notificación de alarmas a través de la red inalámbrica GSM para recibir mensajería instantánea cuando ocurra algún evento en el sistema de control.

- Seleccionar e implementar el hardware que permita la interacción entre el teléfono celular y el sistema de notificación de alarmas para controlar al menos dos dispositivos o eventos.

- Analizar y diseñar el software que permita al usuario la comunicación con el sistema de notificación y control de alarmas.

CAPITULO I

MARCO TEORICO

1.1 Red GSM

1.1.1 Introducción a GSM

Formalmente conocida como "Group Special Mobile" (GSM, Grupo Especial Móvil) aunque también llamada Global System for Mobile communications (Sistema Global para las Comunicaciones Móviles), GSM difiere de sus antecesores principalmente en que tanto los canales de voz como las señales son digitales. Para lograr así un moderado nivel de seguridad [5].

GSM tiene cuatro versiones principales basadas en las bandas: GSM-850, GSM-900, GSM-1800 y GSM-1900 [5]. Inicialmente, GSM utilizó la frecuencia de 900 MHz, pero tras su rápida expansión, pronto se saturó el espacio radioeléctrico entorno a esa frecuencia por lo que las redes de telecomunicación pública empezaron a utilizar las frecuencias de 1800 y 1900 MHz, con lo cual es habitual que los equipos móviles de hoy en día sean tribanda[5].

Desde un principio, los creadores de GSM intentaron lograr la compatibilidad con la RDSI en términos de servicios ofrecidos y señalización de control utilizada, sin embargo, las limitaciones del radioenlace en términos de ancho de banda y costes no permitieron que los estandarizados 64 Kbps de tasa de transmisión de un canal B sobre RDSI se alcanzaran en la práctica [5].

Los servicios de telecomunicaciones se pueden dividir en portadores, teleservicios y servicios suplementarios, siendo sin duda el servicio más destacado el de la telefonía. No obstante, una gran variedad de servicios son ofrecidos por la red. Sus usuarios pueden enviar y recibir datos a una velocidad de hasta 9,600 bps a usuarios que utilicen la red telefónica conmutada, RDSI y redes públicas de conmutación de

[5] Referencia a un dispositivo (sobre todo un teléfono móvil) que soporta múltiples radio-bandas de frecuencia utilizadas en comunicaciones.

paquetes y circuitos utilizando una amplia gama de protocolos como X.25 y X.32, con la ventaja adicional de no necesitar módem al ser digital para dialogar con estas redes, excepto para comunicaciones vocales con la PSTN [5].

El único servicio ofrecido por GSM y que no se encuentra en los sistemas analógicos más antiguos es el que realmente nos interesa para este proyecto, el servicio de mensajes cortos o SMS (Short Message Service) [5]. SMS es un servicio bidireccional para mensajes alfanuméricos cortos (hasta 160 bytes).

1.1.2 Arquitectura de la red GSM

En la *Figura 1* se muestra de manera resumida la arquitectura de la red GMS. Esta arquitectura es más compleja y dispone de más elementos que los presentados en esta figura. El objetivo de este trabajo es describir el servicio SMS a nivel de aplicación, sin entrar en demasiados detalles de la red subyacente.

La arquitectura GSM está constituida por tres partes [6]:

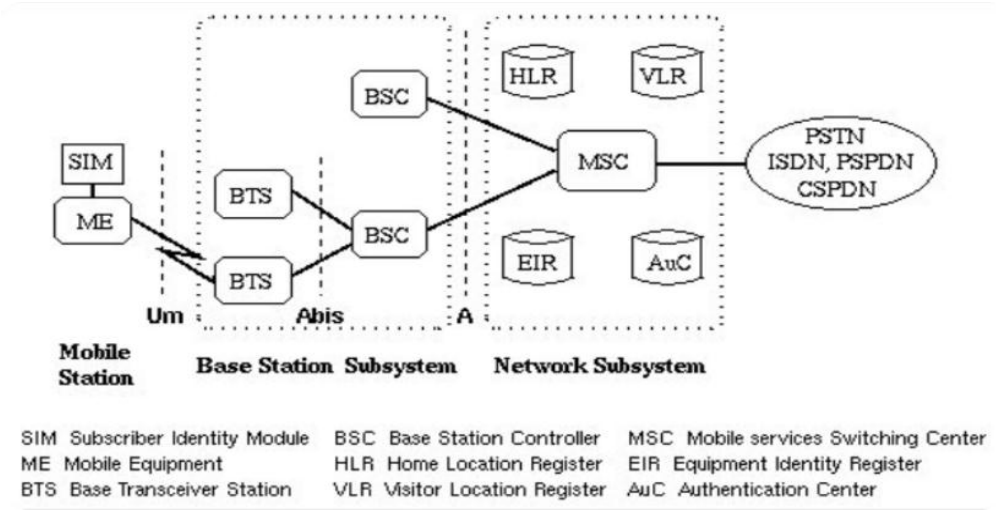

Fig. 1: Arquitectura de la Red GSM.

*1.1.2.1 Subsistema de Radio (**RSS**, Radio SubSystem).*

Se encarga de cubrir y establecer la comunicación entre las estaciones móviles (**MS**) y las estaciones base (**BS**). El interfaz de radio entre ellos se denomina U*m.*

La cual se divide en las siguientes partes [1]:

1) Estación móvil (MS, Mobile Station), es el equipo físico para tener acceso a los diferentes servicios de la red por medio de la interfaz U*m*.
2) Módulo de identificación del abonado (SIM, Subscriber Identity Module) básicamente es la tarjeta que se encarga de la conexión a la terminal del sistema además de tener toda información del abonado como Identidad Internacional del Abonado a un Móvil (IMSI, International Mobile Subscriber Identity), clave de autenticación, guardar la localización, etc.

1.1.2.2 Subsistema de Estaciones Bases (BSS)

Es el encargado de dar cobertura de red a un área determinada. Comprende las funciones de capa física para la conexión de la red con el MS vía interfaz aire U*m*. Este subsistema hace de interfaz entre la parte radio y la parte de red. Dentro de una BSS se identifican las siguientes unidades funcionales [1]:

1.1.2.2.1 BTS (Base Transceiver Station)

La conforman el conjunto de dispositivos capaces de transmitir y recibir información de forma eficiente sobre la interfaz aérea U*m*.

Su labor es doble, dado que la BTS se encarga por un lado de mantener la comunicación con la MS, mientras que por otro la mantiene con su BSC para que sea ésta la que gestione la llamada más allá del nivel físico de la comunicación.

1.1.2.2.2 BSC (Base Station Controller)

Handover, control de las BTS, mapeo de canales radio sobre los canales terrestres. Por un lado se comunica con las BTS a través de un interfaz con canales de

16kbits/s (Abis) y por otro lado se comunica con los MSC a través del interfaz A, con canales de 64kbits/s.

1.1.2.3 Subsistema de red y conmutación (NSS, Network and Switching Subsystem).

Conmutación, gestión de la movilidad, interconexión con otras redes y control del sistema. Esta es la parte más compleja, siendo sus elementos fundamentales los siguientes [1]:

1.1.2.3.1 MSC (Mobile Services Switching Center)

Se trata de la entidad base para el funcionamiento del NSS y se encarga tanto de las labores de conmutación dentro de la red como de posibilitar la comunicación con otras redes.

1.1.2.3.2 GMSC (Gateway Mobile Services Switching Center)

Conexión con otras redes.

1.1.2.3.3 HLR (Home Location Register)

Es un registro utilizado para almacenar y gestionar la información de los usuarios de la red. Puede haber varios o solamente uno, y es la base de datos más importante de la red dado que almacena permanentemente datos de los usuarios.

1.1.2.3.4 VLR (Visitor Location Register).

Este registro se encuentra siempre integrado en la MSC y contiene información temporal sobre usuarios necesaria para que la MSC pueda dar servicio a usuarios visitantes.

1.1.3 Servicios del GSM

1.1.3.1 servicio SMS

El servicio SMS, permite transferir un mensaje de texto entre una estación móvil (MS) y otra entidad (SME) a través de un centro de servicio (SC) [5]. El servicio final

ofrecido es una comunicación extremo-extremo entre la estación móvil (MS) y la entidad (SME). La entidad puede ser otra estación móvil o puede estar situado en una red fija. En el caso de envío de un mensaje entre dos móviles, ambas partes son estaciones móviles. Cuando se envía un mensaje para solicitar algún tipo de servicio de valor añadido, un extremo es una estación móvil y la otra es un servidor que atiende las peticiones [7].

En la norma GSM sólo se especifica la parte de comunicaciones entre las estaciones móviles (MS) y el Centro de servicio. La comunicación entre el Centro de Servicio y las entidades fijas, queda fuera del ámbito de esta norma.

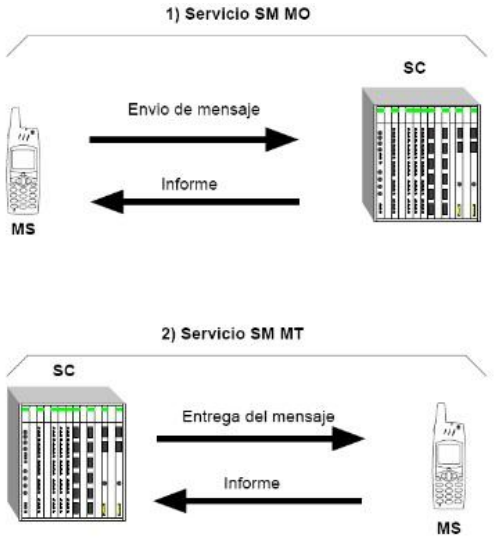

Fig. 2: Servicios Básicos SM MO y SM MT.

El servicio SMS se divide en dos servicios Básicos detallados en la *Figura. 2*:

1. SM MT (Short Message Mobile Terminated Point-to-Point). Servicio de entrega de un mensaje desde el SC hasta una MS, obteniéndose un informe sobre lo ocurrido.

2. SM MO (Short Message Mobile Originated Point-to-Point). Servicio de envío de un mensaje desde una MS hasta un SC, obteniéndose un informe sobre lo ocurrido.

1.1.3.2 Arquitectura de red SMS

Los elementos de red necesarios para proveer el servicio SMS se muestran en la *Figura 3*, son [1]:

1.1.3.2.1 Las Entidades de Mensajería Corta (SME):

Es una entidad que puede enviar o recibir mensajes cortos. Puede ser localizada en la red fija, la estación móvil u otro centro de servicio.

1.1.3.2.2 El Centro de Servicio de Mensaje Corto (SMSC):

Es el responsable de la transmisión, almacenamiento y envío de mensajes cortos entre el SME y la estación móvil.

1.1.3.2.3 El Centro de Conmutación Móvil SMS (SMS GMSC):

Es un centro de conmutación de mensajes encargado de recibir el mensaje del SMSC, interrogar al registro de localización local por la información de encaminamiento, y entregarlo al MSC que da servicio a la estación móvil.

1.1.3.2.4 Registro de Localización Local. (HLR):

Es la base de datos para el almacenamiento permanente y manejo de perfiles de servicio y subscripciones. El HLR provee la información de encaminamiento hacia el cliente indicado. El HLR también informa al SMSC del intento de entrega de un mensaje corto a una estación móvil que ha resultado fallido.

1.1.3.2.5 Registro de Localización del Visitante (VLR):

El VLR es la base de datos que contiene la información temporal acerca de los clientes. Esta información se necesita por el MSC (Mobile Switching Center - MSC) que ejecuta las funciones de conmutación del sistema y las llamadas de control hacia y desde otros teléfonos o sistemas de datos.

1.1.3.2.6 La Estación Móvil (MS):

Es el terminal inalámbrico capaz de recibir y originar mensajes cortos, así como llamadas de voz.

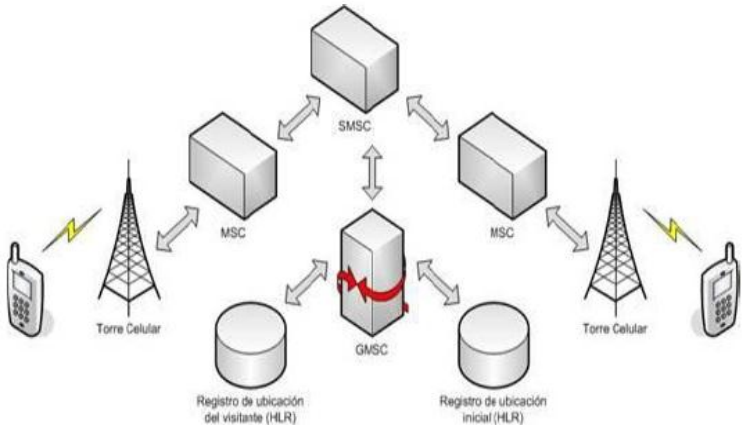

Fig. 3: Elementos de una red SMS.

Para la descripción detallada de la arquitectura, se utiliza un modelo de capas, en el que cada capa o nivel proporciona un servicio a la capa superior, el servicio SMS se implementa mediante el protocolo correspondiente [7]. La arquitectura se divide en 4 capas Las cuales son:

- SM-AL (Short Message Aplication Layer): Capa de aplicación. Envía mensajes de texto cortos entre los diversos usuarios de la red.
- SM-TL (Short Message Transfer Layer): Capa de transferencia. Servicio de transferencia de un mensaje corto entre una MS y un SC (en ambos sentidos) y obtención de los correspondientes informes sobre el resultado de la transmisión. Este servicio hace abstracción de los detalles internos de la red, permitiendo que la capa de aplicación pueda intercambiar mensajes.
- SM-RL (Short Message Relay Layer): Capa de repetición. Proporciona un servicio a la capa de transferencia que le permite enviar TPDU (Transfer Protocol Data Units) a su entidad gemela.
- SM - LL (Short Message Lower Layers): Capas inferiores.

1.1.4 Los Comandos AT

Los comandos AT (se denominan así por la abreviatura de attention) son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal módem. En un principio, el juego de comandos AT fue desarrollado en 1977 por Dennis Hayes como un interfaz de comunicación con un módem para así poder configurarlo y proporcionarle instrucciones, tales como marcar un número de teléfono. Más adelante, con el avance del baudio, fueron las compañías Microcomm y US Robotics las que siguieron desarrollando y expandiendo el juego de comandos hasta universalizarlo [7].

Aunque la finalidad principal de los comandos AT es la comunicación con módems, la telefonía móvil GSM también ha adoptado como estándar este lenguaje para poder comunicarse con sus terminales. De esta forma, todos los teléfonos móviles GSM poseen un juego de comandos AT específico que sirve de interfaz para configurar y proporcionar instrucciones a los terminales [7]. Este juego de instrucciones puede encontrarse en la documentación técnica de los terminales GSM y permite acciones tales como realizar llamadas de datos o de voz, leer y escribir en la agenda de contactos y enviar mensajes SMS, además de muchas otras opciones de configuración del terminal.

Los comandos AT son cadenas ASCII que comienzan por los caracteres AT y terminan con un retorno de carro (LF). Cada vez que el módem recibe un comando, lo procesa y devuelve un resultado, que normalmente es una cadena ASCII salvo que hayamos indicado lo contrario. Al estar la comunicación en ASCII, pondremos utilizar un terminal de comunicaciones desde un ordenador para acceder al módem, bien para configurarlo, bien para hacer pruebas o bien para establecer una comunicación con otro módem.

Para tener acceso a todos esos servicios, y dado que los comandos AT estaban muy extendidos y muy estandarizados, se ha realizado una ampliación, añadiéndose

nuevos comandos. Estos nuevos comandos comienzan por las letras AT+, y se denominan comandos AT+ [7].

Se listan algunos de los comandos AT+ implementados en los módems GSM en la tabla de **Anexos 1** para tener una idea de lo que se puede controlar a través del modem, aunque existen muchos más.

1.1.5 Modem GSM

Los módems GSM no sólo se comportan de forma muy parecida a un modem normal, permitiendo el intercambio de datos con otro modem y utilizándose los comandos AT originales, sino que incluyen muchas más características [7]. Son como pequeños teléfonos móviles, que incluyen su propia tarjeta SIM para poder funcionar y por tanto permiten gestionar la base de datos de teléfonos, la lista de los mensajes SMS recibidos, enviar mensajes SMS, configurar diversos parámetros.

1.1.6 Gateway de mensajería SMS

En general, un Gateway es un programa que permite conectar sistemas que tienen informaciones en diferentes formatos, realizando la conversión entre ellos de la forma más transparente posible.

En un Gateway de mensajería SMS, una de las partes a las que está conectado el Gateway es la red de telefonía celular móvil, sobre la que se envían los mensajes SMS. La otra vertiente del Gateway puede estar conectada a diferentes sistemas o servicios, aunque los casos más comunes incluyen sistemas de correo y redes IP (sistemas cliente/servidor).

El objetivo final del Gateway es enviar mensajes SMS a petición de equipos cliente que no disponen de los medios necesarios y recibir mensajes SMS para que sean procesados o reenviados por otros medios dentro de la red IP [8].

En otras palabras, un Gateway SMS es un programa encargado de traducir los mensajes SMS provenientes de la red de telefonía móvil a un formato legible por otra

red de equipos tipo cliente/servidor (o red IP) que por naturaleza no posee los mismos protocolos de comunicación.

1.2 Comunicación Serial

En telecomunicaciones y computación, la comunicación serial es el proceso de envío de datos de un bit por vez, secuencialmente, sobre un canal de comunicación o un bus de computadora. Contrasta con la comunicación paralela, donde todos los bits de cada símbolo (la más pequeña unidad de datos transmitida por vez) son enviados juntos.

La comunicación serial es utilizada en casi todas las comunicaciones y redes de computadoras, porque los costos de los cables y las dificultades de sincronización hacen a la comunicación paralela poco práctica [9].

Cuando se transmite información a través de una línea serie es necesario utilizar un sistema de codificación que permita resolver los siguientes problemas:

1. **Sincronización de bits:** El receptor necesita saber dónde comienza y donde termina cada bit en la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo (bit para señales binarias).

2. **Sincronización del carácter:** La información serie se transmite por definición bit a bit, pero la misma tiene sentido en palabras o bytes.

3. **Sincronización del mensaje:** Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, por ejemplo, detectar algún error en la comunicación de un mensaje.

1.3 Sensores

Se mostraran los tipos de sensores utilizados en este proyecto, además de proporcionar información sobre su funcionamiento y aplicaciones. Un sensor es un dispositivo que detecta variaciones de: temperatura, intensidad de luz, distancia, aceleración, desplazamiento, presión, fuerza, humedad, etc.

Existen diferentes clasificaciones de sensores, siendo los más importantes los siguientes: formato analógico y formato digital. En los sensores analógicos la señal puede tomar un número infinito de valores dentro de un rango. Normalmente presentan problemas relacionados con la presencia de ruido, interferencias y distorsión. Asimismo, en los sensores digitales la señal solo puede tener un número finito de valores dentro de un rango, es decir, que la función varia de forma discreta.
Se denomina sensor a todo elemento que es capaz de transformar señales físicas como temperatura, posición, longitud etc. en señales eléctricas.

1.3.1 Sensores Infrarrojos

El sensor PIR corresponde a las siglas Pasive Infra Red. Es un dispositivo piro eléctrico (detector de calor). Lo que mide es el cambio de calor, no la intensidad de calor. El calor medido es el calor irradiante cercano al infrarrojo que no es visible.

Este sensor detecta movimiento mediante un promedio del calor irradiado en el tiempo. Como respuesta al cambio el sensor cambia el nivel lógico de su PIN (0-1). Este sensor es de bajo costo y tamaño, por lo que se utiliza en sistemas de alarmas, iluminación y robótica.

Voltaje de alimentación = 5 VDC

Rango de medición = hasta 6 m

Salida = estado de un pin TTL (Transistor-Transistor Logic)

Polaridad de activación de salida seleccionable.

Mínimo tiempo de calibración.

El sensor PIR cuenta con 3 terminales, 2 para alimentación y uno de salida (detección de movimiento).La conexión al microcontrolador solo requiere del uso de este último terminal.

El PIR como se muestra en la *Figura 4* está fabricado de un material cristalino que genera carga eléctrica cuando se expone a la radiación infrarroja. Los cambios en la cantidad de radiación producen cambios de voltaje que son medidos por un amplificador. Este sensor contiene unos filtros especiales llamados Lentes Fresnel que enfocan las señales infrarrojas sobre el elemento sensor. Cuando las señales infrarrojas del ambiente donde está el sensor cambian, el amplificador activa las salidas, para indicar movimiento esta salida permanece activa durante unos segundos lo que permite que el micro controlador sepa si es que hubo movimiento.

Fig. 4: Sensor PIR y radio de cobertura.

El espectro electromagnético de la radiación infrarroja, tiene una longitud de onda más larga que la luz visible no puede ser vista pero si puede ser detectada y los objetos que generan calor también generan radiación infrarroja.

El PIR viene presetiado para la detección del cuerpo humano. Este sensor funciona detectando cambios en el promedio de captura de calor irradiado cerca al infrarrojo (6 metros radio). Es por eso que si uno se queda quieto frente al sensor, este no te detecta más. En teoría si un objeto que no emite calor se mueve el sensor no lo detectaría, por ejemplo un vaso rodando.

1.3.2 Sensores Magnéticos

Los sensores magnéticos detectan una variación en el campo magnético en respuesta a la variación de alguna magnitud física. Están basados en el efecto Hall, por lo que se conocen como sensores de efecto Hall. Como el que se muestra en la *Figura 5*.

Fig. 5: Sensor Magnético.

Es muy común encontrar en el área de alarmas a los sensores magnéticos esto es en muchísimas aplicaciones pero la que vamos a mencionar a continuación es la que nos ayudara a mantener en vigilancia nuestras ventanas o puertas y así cuando se activan si alguien las abre sin antes desconectar la alarma se activara, esta es una aplicación bastante sencilla pero no por eso quiere decir que no puede dársele usos más elaborados como el cual que en lugar de que se active la alarma esta señal se valla a otro circuito de monitoreo y esto haga grabar una video cámara y así no únicamente detectamos a la persona que quiere violar la seguridad de nuestro hogar sino que además lo grabamos y lo localizamos.

La utilización de sensores magnéticos de seguridad ofrece ventajas particulares en casos de condiciones extremas de suciedad, o bien en los casos en que normas de muy elevada higiene deben ser respetadas obligatoriamente. Esto se obtiene gracias a la simplicidad que ofrece la limpieza de sus piezas.

1.4 Relé

El relé o relevador es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

En la *Figura 6* se puede observar un relevador opto aislado. la gran ventaja de los relés electromagnéticos es la completa separación eléctrica entre la corriente de accionamiento, la que circula por la bobina del electroimán, y los circuitos controlados por los contactos, lo que hace que se puedan manejar altos voltajes o elevadas potencias con pequeñas tensiones de control. También ofrecen la posibilidad de control de un dispositivo a distancia mediante el uso de pequeñas señales de control.

Fig. 6: Relé Opto-aislado 5V/30A.

1.5 Timbre Eléctrico.

Un timbre eléctrico es un dispositivo capaz de producir una señal sonora al pulsar un interruptor. Su funcionamiento se basa en fenómenos electromagnéticos.

Consiste en un circuito eléctrico compuesto por un generador, un interruptor y un electroimán. La armadura del electroimán está unida a una pieza metálica llamada

martillo, que puede golpear una campana pequeña. En la *Figura 7* se muestra un timbre eléctrico tipo campana de alta potencia.

Fig. 7: Timbre Eléctrico.

1.6 Alimentación de los subsistemas.

La alimentación de los componentes que integran el sistema es proporcionada por una fuente con dos canales uno de 5VDC y el otro 12VDC. En la *Tabla 1* se pueden observar las especificaciones técnicas. La fuente es de tipo conmutada para garantizar la distribución de energía de forma segura y eficiente.

Para suplir la demanda de corriente de todos los circuitos instalados se requirió el uso de la siguiente fuente de poder que se muestra en la *Figura 8*.

Fig. 8: Fuente de alimentación conmutada.

Tabla 1: Especificaciones técnicas de fuente de alimentación.

Características	MEAN WELL RD-35D
Voltaje de Entrada	100-240VAC 0.8 A 50/60Hz.
Voltaje de Salida	CH1: +5VDC CH2: +12VDC
Corriente de Salida	CH1: 2.2 A CH2: 1.0 A
Potencia	CH1: 35W CH2: 35W
Tipo de Conversión	Conmutada

Para respaldar la fuente de poder ante el fallo de la energía comercial se instaló una UPS, la cual es capaz de entregar hasta 375W operando en 115VAC.

1.7 Arduino UNO

Arduino es una plataforma de hardware libre basada en una sencilla placa con un microcontrolador y un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring. Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. En la *Figura 9* podemos observar el modelo Arduino UNO R3. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Fig. 9: Placa Arduino UNO R3.

Las plataformas Arduino están basadas en los microcontroladores Atmega168, Atmega328, Atmega1280, ATmega8 y otros similares, chips sencillos y de bajo coste que permiten el desarrollo de múltiples diseños.

Al ser open-hardware, tanto su diseño como su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

Arduino UNO es una placa basada en el microcontrolador ATMega328. Tiene 14 pines de E/S digital (6 de los cuales pueden ser utilizados como salidas PWM), 6 entradas analógicas, un oscilador de cristal de 16MHz, conexión USB, un botón de reset, cabeceras ICSP y una entrada de alimentación. Contiene todo lo necesario para hacer funcionar el microcontrolador, simplemente se conecta a un ordenador mediante un cable USB o se alimenta con pilas o adaptadores a corriente continua.

Arduino UNO se diferencia del resto de placas de Arduino en que no hace uso del driver para el chip USB-a-serial FDTI. En lugar de esto, utiliza un ATMega8U2 programado para comportarse como un conversor USB a serie [10].

En resumen Arduino UNO consta de las siguientes características:

Tabla 2: Características de Arduino UNO.

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para E/S digital	14 (6 pueden usarse como salida PWM)
Pines de entrada analógica	6
Corriente por pin E/S	40 mA
Corriente en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

1.8 Raspberry Pi

El Raspberry Pi es una computadora basada en una arquitectura ARMv6 y un GPU de Broadcom capaz de decodificar videos FullHD en varios codecs. Incluye puertos USB 2.0, salida de video HDMI y compuerto analógico, Ethernet, lector de tarjetas de memoria y salida de audio de 3.5mm estándar, puede correr una distribución especial de Linux incluyendo un entorno gráfico [11]. En la *Figura 10* se observa el modelo B.

Fig. 10: Raspberry Pi model B.

Una fundación sin fines de lucro que lleva el mismo nombre lo diseño y fabrico. Se pensó como una forma económica de presentar a niños y jóvenes en edad escolar la programación en computadoras, aunque ha encontrado otros nichos importantes por parte de la comunidad de usuarios. Ideal para proyectos de control y automatización.

Estas son sus especificaciones:

Tabla 3: Características de Raspberry Pi.

SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)
CPU:	ARM1176JZF-S a 700 MHz (familia ARM11)
GPU:	Broadcom VideoCore IV, 59, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia) ,57 1080p30 H.264/MPEG-4 AVC
Memoria (SDRAM):	512 MB (compartidos con la GPU)
Puertos USB 2.0:	2 (vía hub USB integrado)
Entradas de vídeo:	Conector [[MIPI] CSI que permite instalar un módulo de cámara desarrollado por la RPF.
Salidas de vídeo:	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD
Salidas de audio:	Conector de 3.5 mm, HDMI
Almacenamiento integrado:	SD / MMC / ranura para SDIO
Conectividad de red:	10/100 Ethernet (RJ-45) vía hub USB
Periféricos de bajo nivel:	8 x GPIO, SPI, I²C, UART
Consumo energético:	700 mA, (3.5 W)
Fuente de alimentación:	5 V vía Micro USB o GPIO header
Dimensiones:	85.60mm × 53.98mm64 (3.370 × 2.125 inch)

1.9 Modem 3G ZTE

Con un aspecto similar a una memoria USB, el modem 3G es un pequeño Gadget que permite a un usuario acceder a Internet a través de su PC portátil cuando no dispone de ninguna conexión a Internet o cuando no se encuentra dentro de una zona WiFi.

El modem USB 3G, es una especie de modem inalámbrico de tipo WiFi, utiliza la red de los operadores de telefonía para conectarse a Internet. Al igual que los teléfonos móviles, posee un lugar reservado para una tarjeta SIM. Para que funcione, es necesario que previamente se haya suscrito a un plan en un operador de telefonía.

lEl Modem ZTE MF626 HSDPA/EDGE/GPRS que se muestra en la *Figura 11* es un Modem inalámbrico multi-modo 3G compatible con redes HSDPA, WCDMA, EDGE, GPRS, y GSM. Con la interfaz USB se puede conectar a una computadora portátil e integrar la funcionalidad de un Modem combinando perfectamente la comunicación móvil con Internet. Soporta datos a través de la red de telefonía móvil ayudándolo a romper con las limitaciones de una red fija comunicándose inalámbrica mente a toda hora en cualquier lugar donde haya cobertura. En la *Tabla 4* se observan las

Fig. 11: Modem 3G ZTE MF626.

Especificaciones Técnicas:

Tabla 4: Características de Modem 3G USB.

Estándares de red:	HSDPA/ WCDMA/ EDGE/ GPRS/ GSM
	HSDPA 850MHz
Frecuencias:	HSDPA 1900/2100MHz: 1920-1980 MHz, 2110-2170 MHz
	GSM 850MHz
	GSM 900MHz: 880-915 MHz / 925-960 MHz
	GSM 1800MHz: 1710-1785 MHz, 1805-1880 MHz
	GSM 1900MHz: 1850 -1910 MHz, 1930 -1990 MHz
Consumo de corriente Tiempo de espera:	80mA Max: 850mA
Máxima transferencia de potencia:	WCDMA: 200mW, GSM/GPRS: 2W

1.10 Kannel

Kannel fue en primer lugar una pasarela WAP, pero amplió sus funcionalidades posteriormente para dar soporte a WAP usando SMS como transporte de los datos y para obtener páginas web. Básicamente, en una instalación mínima, aparece lo que en Kannel denominan "BearerBox", el componente que comunica con el centro servidor de mensajes (SMSC) o un teléfono WAP que puede sustituirlo en parte. Con este BearerBox comunican los otros componentes, "SMSBox" y "WAPBox", que se encargan de acceder a los servidores HTTP que tienen el contenido que se desea acceder, y que realizan las conversiones/compresiones que establece el protocolo WAP para el envío de los mismos a los teléfonos. En la *Figura 12* podemos observar su arquitectura

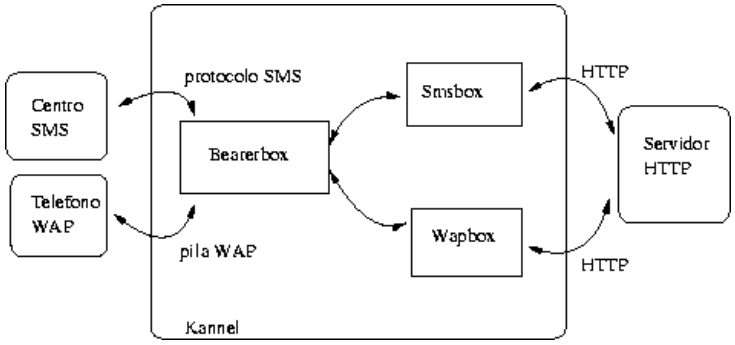

Fig. 12: Arquitectura de Kannel.

En una instalación orientada a SMS, desde el teléfono celular se envía un mensaje que es recibido por el SMSC u otro teléfono móvil, que recibe el mensaje para pasárselo al BearerBox; el mensaje es encaminado al SMSBox, que accede a una página web (puede ser un cgi que genere la página dinámicamente, para ejecutar comandos o lanzar programas, por ejemplo) y el resultado es devuelto al BearerBox de nuevo para ser devuelto al teléfono [12].

Kannel está orientado principalmente al trabajo directo con centros servidores de mensajes (SMSC) e implementa muchos protocolos de acceso a SMSCs de diferentes fabricantes. Es en la conexión directa con un SMSC cuando se obtienen los mejores resultados de rendimiento; sin embargo, no siempre se tiene disponible una conexión de este tipo con la red GSM, por lo que también implementa conexión mediante módem GSM o teléfono móvil con módem incorporado.

1.11 LAMP

Las siglas LAMP se emplean para definir el trabajo conjunto con los siguientes programas utilizados para crear sitios web dinámicos o servicios: Linux, Apache, MySQL y uno de los siguientes lenguajes de programación: Perl, Python o PHP [13].

Todos los elementos que forman LAMP son software libre, de modo que disfrutan de las siguientes ventajas propias del mismo:

- Libertad de copia y distribución. Se pueden conseguir gratuitamente en línea. Hay muchísimas fuentes donde conseguir cualquiera de las distribuciones.
- Libertad de modificación. Junto a los programas ejecutables, se pueden obtener su código fuente.

1.11.1 Linux

Linux está basado en los estándares Unix, y surgió a principios de los años 90, desde entonces, ha ido incrementándose de forma espectacular el número de desarrolladores.

Lo que es propiamente Linux es el núcleo del sistema operativo, que ha ido implementando soporte para una gran parte del hardware actual. Dicho núcleo viene arropado por librerías y utilidades distribuidas bajo la licencia libre GPL[6] o similares (de aquí la denominación GNU/Linux).

Linux, entre muchas otras cosas, es multitarea, multiusuario, multiplataforma, multiprocesador, tiene protección de la memoria entre procesos, soporta muchísimos tipos de sistemas de archivos, dispone de una amplia variedad de protocolos de red soportados en el núcleo y, finalmente, permite compartir por red ficheros e impresoras, incluso con otros sistemas operativos.

1.11.2 Apache

El servidor Apache es un servidor web HTTP[7] de código abierto, para plataformas Unix que implementa el protocolo HTTP/1.1. Cuando comenzó su desarrollo en el año 1995 se basó inicialmente en el código del popular NCSA HTTPd 1.3 pero más tarde fue rescrito por completo.

[6] Licencia de Software que garantiza a los usuarios la libertad de usar, compartir y modificar el software.
[7] Es el protocolo de transferencia de hipertexto usado en cada transacción de la w orld wide web.

Es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable.

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir hipertextos, páginas web o paginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonido. Sin embargo, el hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un formato de archivo y HTTP es un protocolo.

1.11.3 MySQL

MySQL es un servidor de bases de datos relacionales muy rápido y robusto. Es un sistema de gestión de bases de datos multiusuarios, publicado bajo la licencia GPL (GNU, Licencia Pública) desarrollado desde abril de 2009 por la compañía Oracle. Existen diferentes arquitecturas para los sistemas de gestión de bases de datos, pero la más extendida, y la que más éxito ha tenido, es la arquitectura relacional.

El corazón de las aplicaciones web dinámicas es la base de datos que almacena la información de las aplicaciones de manera ordenada en las cuales los datos se organizan a través de tablas relacionadas. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

1.11.4 PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

1.11.5 Python

Python es un lenguaje de programación de propósito general aplicado a menudo en roles de scripting. Comúnmente se define como un lenguaje de scripts orientado a objetos una definición que combina soporte para POO[8] con una orientación general hacia funciones de script.

Las principales características citadas por usuarios de Python son las siguientes:

- Calidad del software. Para muchos, Python se centra en la legibilidad, la coherencia y la calidad del software en general se distingue de otras herramientas de scripting en el mundo. El código de Python está diseñado para ser legible, y por lo tanto, reutilizable y mantenible, mucho más que el de lenguajes de script tradicionales.
- Productividad del desarrollador. Python aumenta la productividad muchas veces más allá que lenguajes compilados o estáticamente escritos como C,

[8] Programación Orientada a Objetos.

C++ y Java. El código de Python es habitualmente entre un tercio y un quinto del tamaño de su equivalente en código C++ o Java. Esto significa que hay menos que escribir, depurar, y menos para mantener después de hecho.

1.12 Lenguaje de Programación Arduino

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software [13].

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el popular lenguaje de programación de alto nivel Processing. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino. Algunos ejemplos son: Java, Flash (mediante ActionScript), Processing, Pure Data, Etc.

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato de sistemas y lenguajes puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar Arduino mediante esta gran variedad.

CAPITULO II

DISEÑO E IMPLEMENTACION DEL SISTEMA

En este capítulo se describe el proceso del diseño del sistema, así como también todos los componentes que conforman este proyecto. Existen distintas maneras de poder realizar este proyecto. Nosotros evaluamos las diferentes tecnologías existentes en el mercado y utilizando como criterio principal el uso de Hardware de muy bajo costo y Software con licencia de código abierto para que se puedan desarrollar sistemas más robustos y sean escalables.

2.1 Diseño del Sistema de Control

Cumpliendo con el objetivo general del proyecto se diseñó e implemento un sistema de propósito general para la notificación de alarmas mediante el servicio de mensajes cortos de la red GSM. En la *Figura 13* se muestra el diagrama de bloques de los dispositivos que conforman el sistema.

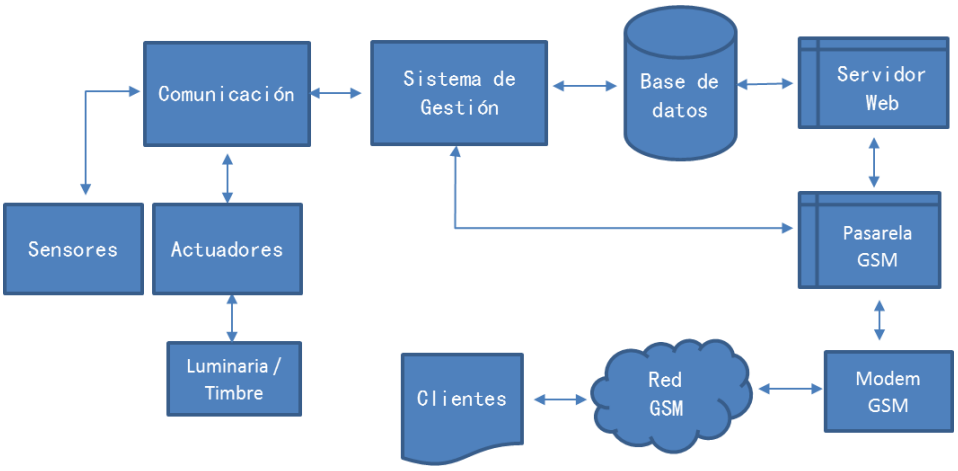

Fig. 13: Diagrama de Bloque del Sistema

Con la implementación de este sistema lo que se pretende es brindar una herramienta practica para el control y notificación de alarmas en un hogar a través de un teléfono celular.

A continuación se describe paso a paso el proceso de envío y recepción de mensajes del sistema.

Un usuario de telefonía móvil solicita el servicio de alarma por medio de mensaje de texto SMS. El mensaje es recibido por el dispositivo móvil conectado a la pasarela de mensajes. El modem GSM recibe el mensaje por medio de comandos AT y se comunica con el computador por medio del puerto USB, estableciéndose un medio físico por el cual los mensajes de texto son transmitidos a la computadora para su procesamiento y ejecución. También es el encargado de enviar la información de respuesta que el sistema reporta al usuario.

Un Gateway SMS es un programa encargado de traducir los mensajes SMS provenientes de la red de telefonía móvil a un formato legible por otra red de equipos tipo cliente/servidor que por naturaleza no posee los mismos protocolos de comunicación.

Los mensajes enviados por el usuario son entregados al sistema por medio del Modem GSM y se ejecuta un script de servicio web desarrollado en PHP, por medio del servidor web Apache, la información obtenida se procesa y almacena en una base de datos.

La tarjeta de control se utiliza con la aplicación de monitoreo y control de alarmas, está conectada al puerto serial de la PC, desde donde se reciben y entregan las señales del proceso que se desea controlar, esta tarjeta posee en la etapa de salida, relés que son manejados por transistores los cuales se saturan de acuerdo al estado lógico de los bits de datos del puerto serial, los relés tienen la capacidad de corriente como para poder activar o desactivar equipos eléctricos o electrónicos. En la etapa de entrada posee sensores conectados cuyos bits de estados son introducidos por el puerto serial.

Desarrollar un sistema como el que se describe, tiene un costo muy bajo en comparación con otros sistemas existentes en el mercado, todo esto mediante la tecnología GSM y el uso de microcontroladores.

2.2 Instalación en vivienda.

El proyecto esta implementado en una vivienda prototipo para lo cual se realizó un plano arquitectónico y se le implemento el diagrama unifilar de las instalaciones eléctricas y electrónicas necesarias para el funcionamiento del proyecto como son: panel de control, sensores de movimiento y sensores magnéticos. Esta vivienda está dividida en zonas, se puede observar en el plano arquitectónico: el dormitorio, sala común y cocina. Como se observan en la *Figura 14.*

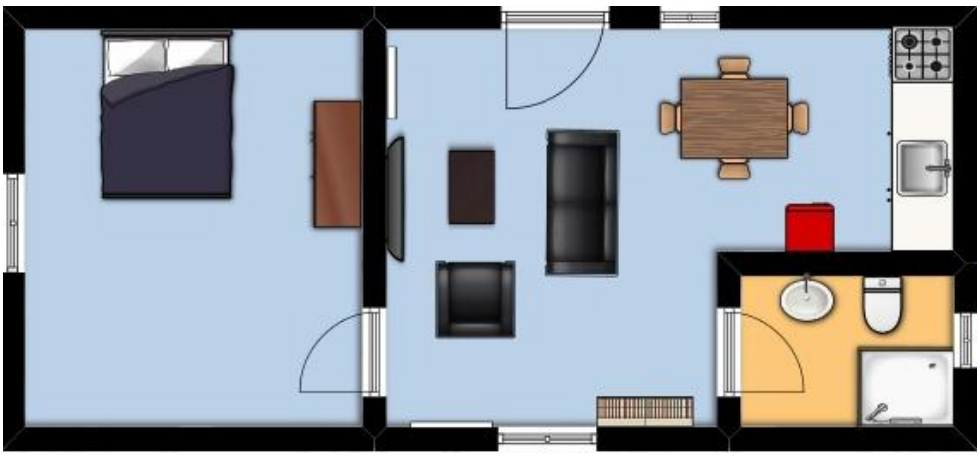

Fig. 14: Plano de Vivienda donde se instaló el sistema.

En la sala común se ubicó el sensor de movimiento PIR, este sensor es de fácil instalación y no necesita de calibración tiene tres pines uno de alimentación a 12 VDC otro pin de GND y el ultimo es de señal además detecta movimiento a una distancia de seis metros a la redonda. En el momento que detecta algún movimiento este sensor automáticamente envía una señal bajo o un cero lógico a la tarjeta de recepción de señal. Este sensor estará conectado como dispositivo de entrada hacia la placa Arduino ATMEGA 3 y se encuentra instalado en el techo de la vivienda.

En el marco de la puerta principal de la vivienda se instaló un sensor magnético, este sensor es fácil de instalar es económico y funciona como un punto continuo cuando las dos placas están juntas. Una vez que se separan deja de haber un punto de continuidad esa señal llega a la tarjeta (receptora de señales) en el momento que se abra la puerta se indicara la activación del sensor. Al activarse debido a la pérdida del campo magnético envía una señal en alto (+5V) a un pin de entrada digital en la placa Arduino que activara la alarma.

2.3 Sensor Infrarrojo

Los sensores que se utilizan en este proyecto son sensor de movimiento PIR, sensor de movimiento magnético debido a que estos son los sensores utilizados en los sistemas de vigilancia y detección. Estos sensores estarán conectados como dispositivos de entrada hacia la placa Arduino ATMEGA 3 se encuentran instalados en el techo de la vivienda, y los sensores magnéticos se instalan en molduras de ventanas y marcos de puertas.

Su componente principal es el sensor piroeléctrico. Se trata de un componente electrónico diseñado para detectar cambios en la radiación infrarroja recibida. Generalmente dentro de su encapsulado incorporan un transistor de efecto de campo que amplifica la señal eléctrica que genera cuando se produce dicha variación de radiación recibida, tienen numerosas lentes de Fresnel o segmentos de espejo, un alcance efectivo de unos diez metros (treinta pies), y un campo de visión menos de 180 grados.

El detector de movimiento de infrarrojos pasivos (PIR) *Figura 15*. Son sistemas de seguridad que reconocen la energía infrarroja emitida por el calor de un cuerpo. Estos detectores están diseñados para activar la alarma sólo cuando hay un aumento repentino en los niveles de energía se envía una señal al pin del controlador Arduino, activando la alarma. EL sensor se polariza con un voltaje nominal entre 9V y 12VDC. Posee 4 terminales, con velocidad de detección de 0.2m/s a 3.5m/s. Un

patrón de cobertura de 4m X 3m en 360º a una altura de 2.4 metros sobre el nivel del suelo. Posee una salida de relé de señal NC para detección de movimiento. Al estar configurado en modo Relé, el DG467 funciona como un detector de movimiento estándar comunicando sus señales de alarma y de sabotaje mediante los relés. Los terminales GRN y YEL no son usados en el modo relé. En el modo Relé, la configuración del detector sólo puede ser modificada mediante los puentes.

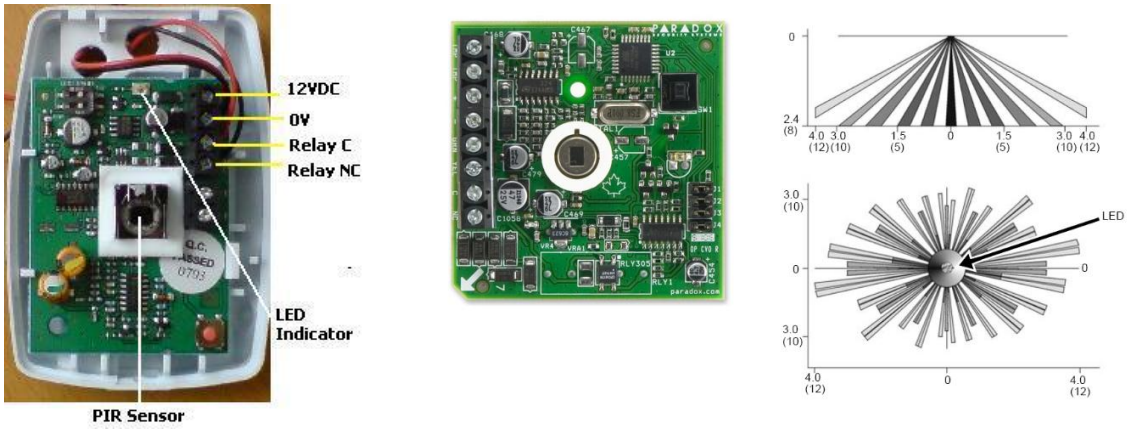

Fig. 15: Sensor PIR y radio de cobertura

2.4 Sensor Magnético

Es un sensor que forma un circuito cerrado por un imán y un contacto (Reed Switch) muy sensible que al separarse, cambia el estado (se puede programar como NC o NA) provocando un salto de alarma. El reed switch o interruptor de lengüeta es un interruptor eléctrico activado por un campo magnético.

Cuando los contactos están normalmente abiertos se cierran en la presencia de un campo magnético; cuando están normalmente cerrados se abren en presencia de un campo magnético. El reed switch como se observa en la *Figura 16* consiste en un par de contactos ferrosos encerrados al vacío dentro un tubo de vidrio. Cada contacto está sellado en los extremos opuestos del tubo de vidrio. El tubo de vidrio puede tener unos 10 mm de largo por 3 mm de diámetro. Al acercarse a un campo

magnético, los contactos se unen cerrando un circuito eléctrico. La rigidez de los contactos hará que se separen al desaparecer el campo magnético.

Lo utilizamos en una puerta, colocando una parte del sensor en el marco y otra en la puerta. Al activarse debido a la pérdida del campo Magnético envía una señal en alto (+5V) a un pin de entrada digital en la placa Arduino que activara la alarma.

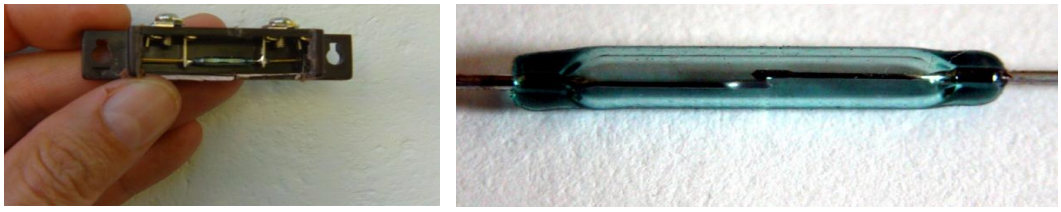

Fig. 16: Sensor magnético y tubo de vidrio al vacío.

2.5 Modulo de Relé y Timbre eléctrico

Utilizamos el módulo de relés SPDT (Single Pole Double Thow) debido a que es un elemento básico de control tipo On-Off en los sistemas de alarma y detección, su fácil manejo de cargas AC-DC, no requiere salidas analógicas en el diseño de este Sistema. Los elementos finales conectados al módulo de relés son cargas tipo AC en

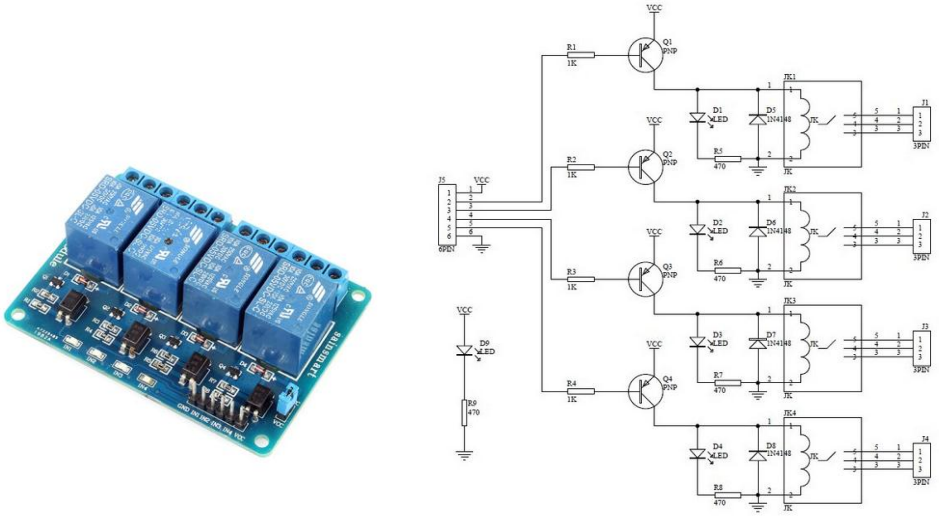

Fig. 17: Módulo de Relés y esquema de conexión.

este caso un timbre y una luminaria que puede ser del tipo estroboscópica o de cualquier tipo. En la *Figura 17* se muestra el módulo de relés y el esquema de conexión.

El dispositivo que produce la señal sonora es un timbre eléctrico de corriente alterna de 110VAC con una corriente de consumo de 170mA y una potencia 90db.

2.6 Detección falla de energía

Se utiliza una fuente de 5VDC como se observa en la *Figura 18*. Para verificar si se encuentra funcionando con energía de la red comercial o energía de respaldo entregada por la UPS al Sistema, fue elegido debido que se acopla fácilmente al controlador Arduino con 5VDC y a la UPS-CDP, sin necesidad de diseñar un circuito se conecta a la toma no respaldada de la UPS.

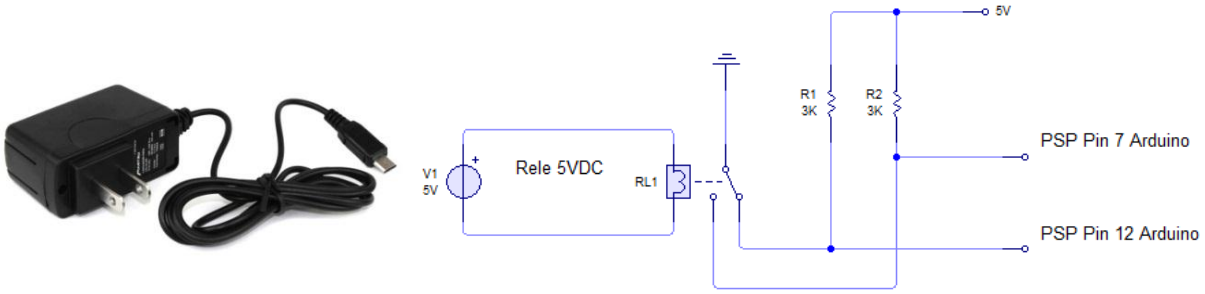

Fig. 18: Cargador de 5VDC y esquema de conexión.

2.7 Controlador del sistema de detección

Utilizamos la placa de desarrollo Arduino Uno R3 para el control del sistema de alarma debido a su bajo costo, simplicidad y su código es liberado bajo licencia de Código Abierto (GPL), reduce sustancialmente el tiempo de Desarrollo en comparación a un PIC[9], debido que posee un Entorno de Desarrollo (IDE) para programar la placa [14]. Proporciona circuitos de soporte para conexiones de sensores y actuadores lo que facilita el diseño del hardware, en contraposición al PIC

[9] Familia de microcontroladores tipo RISC.

en el que hay que construir una interfaz. Como se observa en la *Figura 19*. No es necesario construir fuente de alimentación ya que posee integrado un regulador de voltaje, su voltaje de alimentación varia en un rango de 7VDC a 12VDC y su voltaje de Operación es de 5VDC. Posee un puerto de comunicación USB desde donde se puede alimentar la placa y establecer comunicación serial con una PC a través de un convertidor serial RS232 a TTL serial [14].

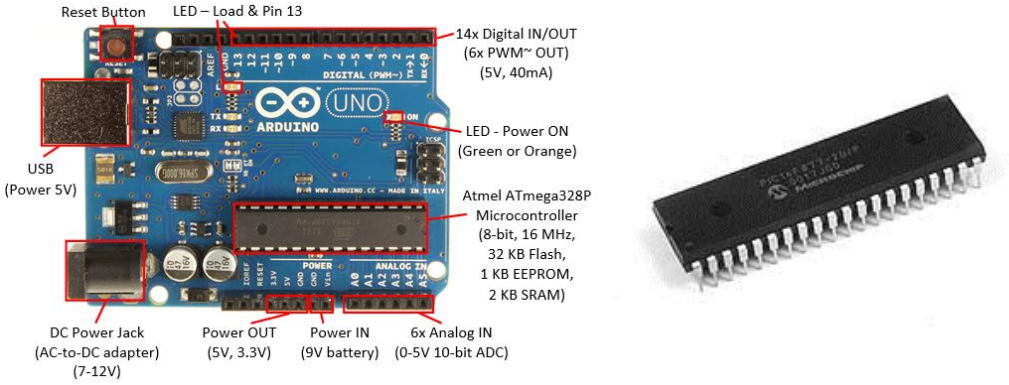

Fig. 19: Placa Arduino vs Microcontrolador PIC.

2.8 Ordenador Raspberry Pi

La Raspberry Pi es un ordenador de placa reducida de arquitectura ARM referencia que funciona con sistema operativos basados en Linux como Debian.

El bajo costo del ordenador y su versatilidad, tamaño reducido y mejor manejo de la potencia eléctrica es útil en entornos industriales o sistemas embebidos como controlador de interfaz y como un servidor. Esta arquitectura se usa en entornos industriales o en sistemas embebidos dentro de otros que sirven como controladores e interfaces.

Debido a los grandes niveles de integración y reducción de componentes y conectores, los computadores en una tarjeta suelen ser más pequeños, livianos, más confiables y con un mejor manejo de la potencia eléctrica que los computadores de múltiples tarjetas. En el desarrollo de este sistema se utiliza una Raspberry Pi modelo B con sistema operativo Debian wheezy. A continuación se observa el modo grafico GNOME de la Raspbery Pi ejecutando la terminal (consola). Como se observa en la *Figura 20*.

Fig. 20: Interfaz gráfica sistema operativo Debian.

2.9 Programación del Controlador de Detección

A continuación se describe el diseño e implementación del software de control del sistema, Primero se declaran las varíales que utilizamos para almacenar el estado de los sensores como se observa en la *Figura 21*.

```
4  int sensor = 0;          // Sensor de Presencia PIR
5  int door= 0;             // Senosr de Deteccion Magnetico
6  int unionf = 0;          // Estado del Detector de Energia Comercial
7  int psp = 0;             // Estado del Detector de Energia de Respaldo UPS
8  int habilitar=0;         // Condicion de Armado del Sistema de Deteccion
9  int incomingByte;        // Buffer del Puerto Serial
```

Fig. 21: Declaración de variables programa Arduino

El **Void Setup** es el bloque en el programa en el que inicializamos los pines de entrada y salida de la placa Arduino, inicializamos el puerto serial y configuramos a una velocidad de 9600 bps. Como se muestra en la *Figura 22*.

```
16 void setup() {
17   // initialize the pins as an output:
18   Serial.begin(9600);
19   pinMode(4, OUTPUT); //Led Indicador
20   pinMode(2,INPUT);//sensor
21   pinMode(8,INPUT);//puerta
22   pinMode(5,OUTPUT);//salida rele
23   pinMode(12,INPUT);//union fenosa
24   pinMode(7,INPUT);//energia alterna
25   pinMode(13,OUTPUT); //Dispositivo a Activar
26 }
```

Fig. 22: Bloque Void Setup programa Arduino.

La función **Void Loop** en el programa se encarga de ejecutar el programa cada vez que la placa base es energizada, es el bloque donde se escribe todas las instrucciones de programa, en el ponemos en alto o bajo los Relés dependiendo del caso , y obtenemos el nivel del detector de energía si se encuentra en bajo (0VDC) o en alto (5VDC), leemos el puerto serial si se encuentra disponible, y si obtenemos una serie de caracteres definidos ejecutamos las instrucciones, el carácter G cuando es recibido la placa base envía una cadena de texto indicando el estado de Energía , cuando los caracteres “X” y “Y” son recibidos la placa base cambia el estado de los pines de salida para apagado o encendido de la luminaria respectivamente, transmitimos por la conexión serial con la PC (Raspberry Pi) una cadena indicando si el dispositivo fue activado o desactivado (Luminaria). En la *Figura 23* se observa el código del programa.

```
void loop(){
  // read the state of the pushbutton value:
  digitalWrite(13,LOW); // Se pone la Salida del Pin 13 en Bajo
  digitalWrite(4,HIGH); // Se pone en Alto el pin 4 para el Rele
  digitalWrite(5,HIGH); // Se pone en Alto el pin 5 para el Rele
  digitalWrite(9,LOW);  // Se pone en bajo el LED Indicador Alarma
  psp=digitalRead(7);   // Se lee el Detector de Energia de Respaldo
  unionf=digitalRead(12);  // Se lee el Detector de Energia de Energia Comercial
if (Serial.available()) {
  incomingByte = Serial.read(); // Si el serial Esta Disponible lo que se reciba es guardado en la Variable incomingbyte
}
if (unionf == HIGH){
  if (incomingByte=='G' && tx1 == 0){
Serial.println("Energia Estable");
tx1=1;
tx2=0;
incomingByte=0;
}
}
if (incomingByte=='X'){
digitalWrite(13,LOW);
Serial.println("Dispositivo Desactivado");
incomingByte=0;

if (incomingByte=='Y'){
digitalWrite(13,HIGH);
Serial.println("Dispositivo Activado");
incomingByte=0;
}
if (psp == HIGH ){
  if (incomingByte=='G'&& tx2 == 0){
Serial.println("Energia UPS");
tx1=0;
tx2=1;
incomingByte=0;
}
}
```

Fig. 23: Función Void Loop programa Arduino

Cuando detectamos el carácter H por medio del puerto serial entramos en un bucle While donde se activa el sistema de alarma poniendo en alto el ping 9 que es la salida al LED rojo indicando que el sistema ha sido activado enviando una cadena de texto por medio del puerto serial . A continuación se lee el estado de cada sensor y si el sensor es activado se pone en estado alto (5VDC) al ocurrir esto la placa Arduino envía una cadena de texto indicando si fue una “Violación a la Puerta” (Sensor Magnético) o una “Intrusión Detectada” (Sensor PIR) por el puerto serial .El dispositivo de salida luminaria ejecuta las mismas condiciones que en la explicación anterior junto con el detector de energía por lo tanto el código no es necesario mostrarlo.

Para desactivar el sistema de alarma y escapar del ciclo While el carácter a recibir es L a lo cual la placa Arduino envía una cadena de texto indicando que el sistema ha sido desactivado y el programa regresa al bloque void loop así sucesivamente. En la *Figura 24* se muestra el código para desactivar el sistema de control. En la *Figura 25* se observa el diagrama de flujo del sistema de gestión. En la *Figura 26* se observa el diagrama de conexión del sistema de control.

```
65  if (incomingByte == 'H') {
66  Serial.println("Sistema Armado");
67  for (int i=0;i<2;i++){
68    digitalWrite(9,HIGH);
69    delay(500);
70    digitalWrite(9,LOW);
71    delay(500);
72  }
73  habilitar = true;
74  }
75  incomingByte=0;
76  while (habilitar == true) {
77  digitalWrite(9,HIGH);
78  sensor=digitalRead(2);
79  door=digitalRead(8);
80  unionf=digitalRead(12);
81  psp=digitalRead(7);
82  if (Serial.available()) {
83    incomingByte = Serial.read();
84  }
85  if (sensor == HIGH){
86    if (incomingByte=='A' && tx3==0){
87        Serial.println("Intrusion Detectada");
88  digitalWrite(4,LOW);
89  digitalWrite(5,LOW);
90  tx3=1;
91  tx4=0;
92  incomingByte=0;
93     }
94  }
95  if (door == HIGH){
96     if (incomingByte=='A' && tx4==0){
97        Serial.println("Violacion a la puerta");
98     digitalWrite(4,LOW);
99     digitalWrite(5,LOW);
100 tx3=0;
101 tx4=1;
102 incomingByte=0;
103   }
131 if (incomingByte=='L'){
132 habilitar=false;
133 tx3=0;
134 tx4=0;
135 Serial.println("Sistema Desarmado");
136 incomingByte=0;
137 }
138 incomingByte=0;
139 }
140
141 }
```

Fig. 24: Código para desactivar el sistema de control.

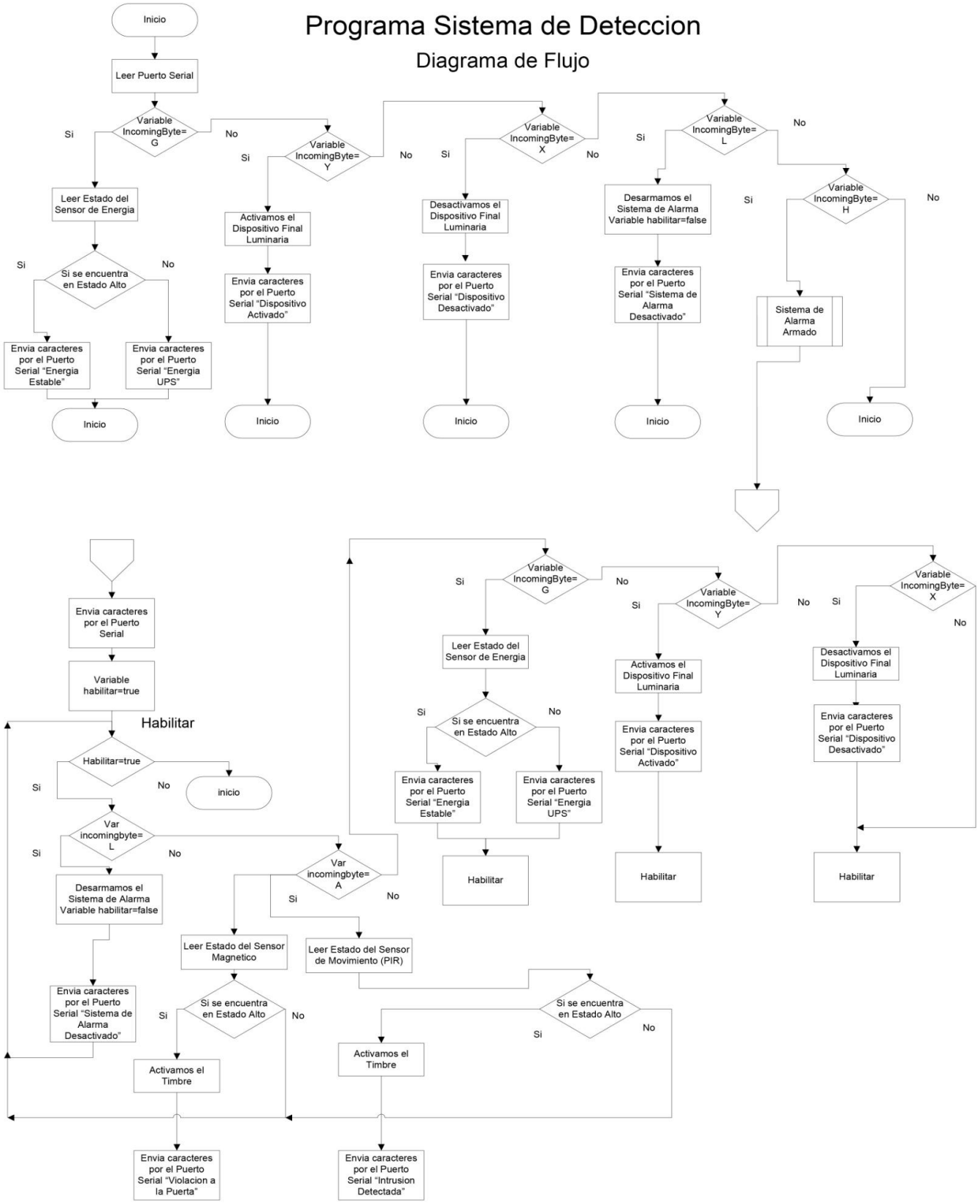

Fig. 25: Diagrama de Flujo del Programa del sistema de gestión.

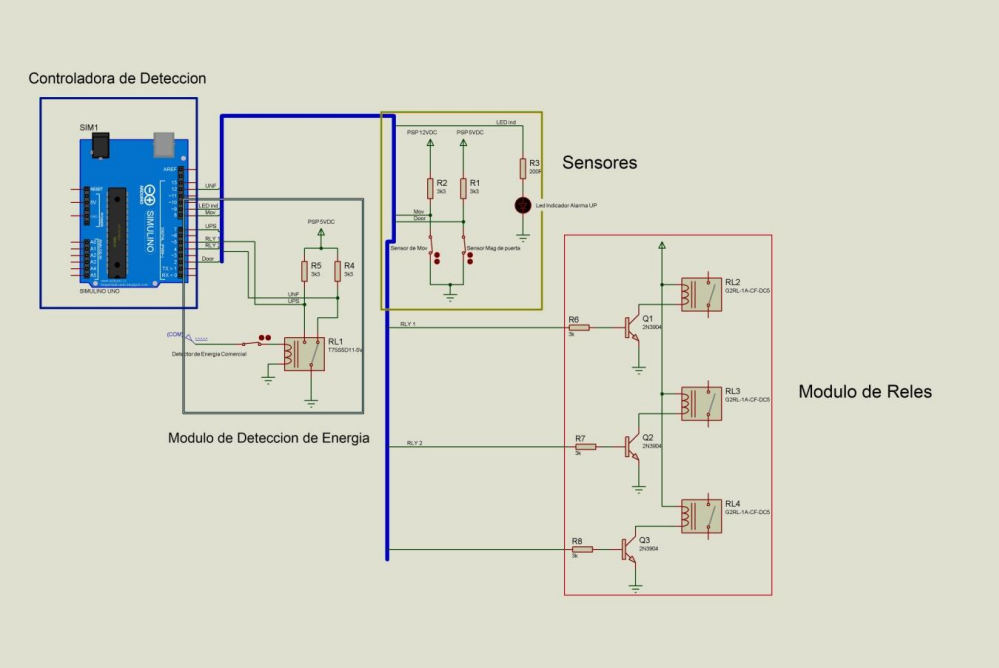

Fig. 26: Diagrama de conexión del sistema de control.

2.10 Modulo de Energía de Respaldo

El sistema debe ser capaz de estar operativo bajo corte en el suministro eléctrico por lo tanto debe ser instalado un modulo de respaldo de alimentación. Con el uso de una UPS ahorramos el diseño de una fuente DC y cargador de baterías, ventajas de manejar cargas AC y mejoramos la calidad de la energía eléctrica que llega a las cargas, filtrando subidas y bajadas de tensión y eliminando armónicos de la red.

Se utiliza un sistema de alimentación ininterrumpida (en ingles UPS) marca CDP modelo B-UPR 505 con una capacidad de 300W como modulo de energía de respaldo lo que proporciona autonomía al sistema de alarma de operación en un lapso de 2 horas. En la *Figura 18* se observa el módulo de energía de respaldo.

La alimentación de los componentes electrónicos y los módulos que integran el sistema es proporcionada por una fuente con dos canales uno de 5VDC y el otro 12VDC. La fuente es de tipo conmutada para garantizar la distribución de energía de forma segura y eficiente. El modelo que utilizamos es MEAN WELL RD-35D debido a su tamaño.

Fig. 27: UPS modelo B-UPR 505

2.11 Software de Control del Sistema

Python es un lenguaje de programación de propósito general aplicado a menudo en roles de scripting. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

El modulo de gestión del sistema es un programa (script) escrito en python es el encargado de la parte de gestión del sistema. Su funcionamiento es el de establecer comunicación serial con el modulo de detección (Placa Arduino) para obtener los valores de los sensores, este programa se encarga de consultar a una base de datos los datos de los usuarios inscritos al sistema los datos puntuales los número de teléfono de los usuarios y su estado si esta de alta o de baja en el sistema y por ultimo enviar mensajes SMS a los usuarios del sistema, a través de una pasarela de mensajería, los eventos ocurridos como la activación del sistema y la detección de movimiento o presencia en los sensores.

Fue desarrollado en python y no en otro lenguaje debido a que el ordenador en este caso una Raspberry Pi cuenta con el lenguaje instalado por defecto, librerías que son útiles como la librería serial que es robusta y sencilla de implementar. En la *Figura 28* se observa diagrama de bloque de la comunicación entre el sistema de gestión, el modulo de la base de datos, de detección y la pasarela de mensajería SMS

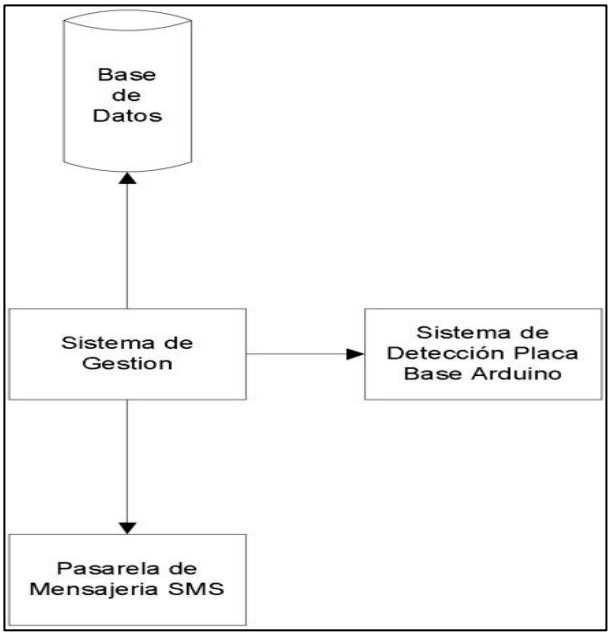

Fig. 28: Diagrama de bloque sistema de gestión y base de datos

En la *Tabla 5* obtenemos el protocolo de intercambio de información entre el programa de gestión y la placa Arduino estos son los datos que son transmitidos y recibidos entre ellos.

Tabla 5: Protocolos de intercambio de información.

PROTOCOLO DE INTERCAMBIO DE INFORMACION		
Caracteres de Contenido a Enviar	**Lista de Acciones**	**Respuestas**
H	Activar Sistema de Alarma	Sistema Armado
L	Desactivar Sistema de Alarma	Sistema Desarmado
G	Obtener el Estado del Sensor de Detección de Energía	Energía Estable, Energía UPS
A	Obtenemos el Estado de los Sensores de Movimiento y Presencia	Intrusión Detectada, Violación a la puerta
Y	Activamos el Dispositivo Final en este caso Luminaria	Dispositivo Activado
X	Desactivamos el Dispositivo Final en este caso Luminaria	Dispositivo Desactivado

2.12 Programación del Sistema de Gestión

Importamos los módulos de Python a utilizar en el programa, declaramos las variables en este caso los caracteres de contenido y creamos una función de salida del sistema cada vez que el usuario quiebre el programa o el Sistema Operativo se apague o reinicie(señales siginit y sigterm) como se observa en la *Figura 29*.

```
4   import serial
5   import MySQLdb
6   import pdb
7   from send_sms import send2smsbox
8   from funcion2 import arduino_verificar
9   from read import poder
10  from sms_notificacion import envios
11
12  luces='on'
13  luces_on='Y'
14  luces_off='X'
15  status='A'
16  off = 'Apagado'
17  out5= ''
18  Servicio  = 'Alarma'
19  energia ='G'
20  input='H'
21  output='L'
22  def on_exit(sig,func=None):
23      if conexion:
24          # conexion.close()
25          #log.debug('cerrando')
26           ser.close()
27      sys.exit(1)
28
29  #Este mensaje se envia cuando el OS solicita cerrar el programa
30  signal.signal(signal.SIGTERM,on_exit)
31  #Este mensaje se envia cuando el usuario envia ctrl+c
32  signal.signal(signal.SIGINT,on_exit)
```

Fig. 29: Código de importación de módulos de Python.

Luego localizamos si se encuentra la placa Arduino, conectada por un cable USB al ordenador, en el sistemas operativo Debian el dispositivo puede ser enlistado en 4 posibles puertos /dev/ttyACM del 0-4, lo hacemos utilizado una estructura de control **for** .Si encontramos el puerto iniciamos la conexión serial con la placa Arduino si no se encuentra se envía un mensaje de texto a los usuarios activos del sistema

indicando que la placa Arduino no se encuentra conectada y finaliza el programa. A como se observa en la *Figura 30*.

```
# Variable para saber si hemos encontrado el puerto o no
bEncontrado = False
# Hacemos un bulce para recorrer los puertos que queremos comprobar
for iPuerto in range(0, 4):
    try:
        # Puerto que vamos a probar
        PUERTO = '/dev/ttyACM' + str(iPuerto)
        # Velocidad
        VELOCIDAD = '9600'
        # Probamos ha abrir el puerto
        Arduino = serial.Serial(PUERTO, VELOCIDAD)
        # si no se ha producido un error, cerramos el puerto
        Arduino.close()
        # cambiamos el estado del la variable para saber si lo hemos encontrado
        bEncontrado = True
        # Salimos del bucle
        break
    except:
        # Si hay error, no hacemos nada y continuamos con la busqueda
        pass

# Si encontramos el puerto?
if bEncontrado:
    # Mostramos el puerto donde esta el arduino
    print('el puerto del arduino es: ' + '/dev/ttyACM' + str(iPuerto))
    ser = serial.Serial(
    port='/dev/ttyACM' + str(iPuerto),
    baudrate=9600)
    #ser.open()
    ser.isOpen()
```

Fig. 30: Código de conexión USB de placa Arduino.

Creamos un ciclo **while** true en el que generamos una conexión a la base de datos utilizando la dirección local el nombre de usuario, el password y el nombre de la base de datos para obtener los datos, creamos un cursor y un espacio en memoria para hacer la consulta a una tabla llamada comando obtenemos el dato de activación del

sistema en este caso una variable texto con el valor alarma. A como se muestra en la *Figura 31*.

```
while 1:
    conexion = MySQLdb.connect("localhost","root","raspberry","KannelProduccion")
    cursor = conexion.cursor()
    sql2=("SELECT comando FROM tbl_Comando WHERE comando='%s '" %(Servicio))
    cursor.execute(sql2)
    respuesta2 =cursor.fetchone()
    rowsaffected = cursor.rowcount
    conexion.commit()
    conexion.close()
```

Fig. 31: Código de conexión a la base de datos.

Si el valor obtenido en la consulta existe ejecutamos un bloque **try catch** escribimos en el puerto serial enviando en este caso el Carácter H a la placa Arduino si no podemos enviar el dato ejecutamos una excepción que se encarga de enviar un mensaje de texto a los usuarios del Sistema que el Dispositivo no se encuentra.

Si podemos enviar el dato por el puerto serial, creamos una variable tipo string vacía para almacenar dato y ejecutamos la lectura del puerto serial donde obtenemos los datos que nos envía la placa Arduino, si el dato es distinto de vacío enviamos un mensaje de texto con este dato a los usuarios inscritos en el sistema utilizando una pasarela de mensajería. Como se observa en la *Figura 32*. El programa continua siguiendo la misma lógica en los casos de los diferentes caracteres de contenido por lo tanto no vamos a redundar en la explicación de los otros caracteres. El programa completo se encuentra en los **Anexos 2** de este documento.

```
if rowsaffected ==1:
        try:
            ser.write(input +'\r\n')
        except serial.SerialException:
            print "muerto el puerto"
            tester =arduino_verificar()

        salida=''
        time.sleep(1)
    print "%s" %(Servicio)
   # print "%s" %(respuesta2[0])
    #ser.write(output +'\r\n')
    estado='on'
        time.sleep(2)
        try:

        while ser.inWaiting()>0:
            salida += ser.read(1)
    except IOError:
            arduino_verificar()

    if salida != '':

        conexion = MySQLdb.connect("localhost","root","raspberry","KannelProduccion")
        cursor = conexion.cursor()
        sql3 = ("TRUNCATE TABLE tbl_Comando")
        cursor.execute(sql3)
        conexion.commit()
        conexion.close
        respuesta=envios(conexion,texto4)

            if respuesta is not None:
                print "Mensaje Enviado"
    print ">>"+ salida
```

Fig. 32: Código de bloque try catch

2.13 Diseño e Implementación de la Base de Datos

Se utilizó MySQL como sistema de gestión de base de datos relacional del sistema, por su fiabilidad, velocidad, que es licencia GPL, está muy ligada a PHP para aplicaciones web, y por la gran cantidad de datos que puede manejar, el almacenamiento de datos permite la escalabilidad.

Para instalar MySQL en el ordenador que utiliza el sistema Debian (Distribución Linux) escribimos este comando en consola:

pi@raspberrypi:~ **$ sudo apt-get install mysql**

Instalamos phpMyAdmin que es una herramienta de código libre y abierto escrito en PHP con la que podemos manejar la administración de MySQL o MariaDB. Con el uso de un navegador web. Se puede realizar diversas tareas como crear, modificar o borrar bases de datos, tablas, campos o filas; ejecución de sentencias SQL; o la gestión de usuarios y permisos [15].

pi@raspberrypi:~ **$ sudo apt-get install phpmyadmin**

Para el desarrollo de la base de datos, creamos tablas con la Información de los usuarios, A continuación en la *Figura 33* se describe el modelo relacional de esta base de datos.

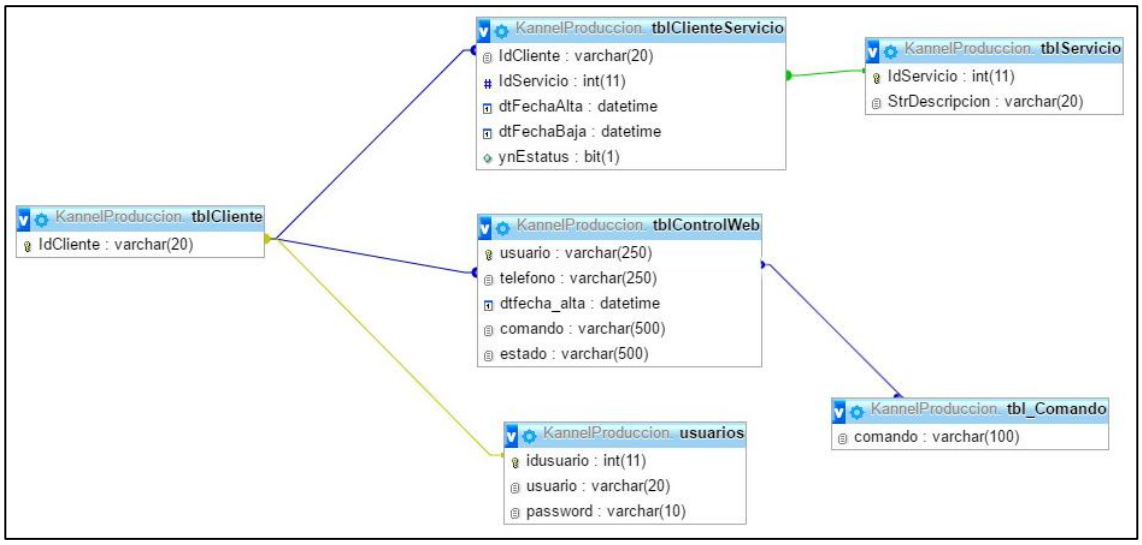

Fig. 33: Modelo relacional de base de datos clientes

La tabla tblCliente guarda el valor del número telefónico de un usuario, en la tabla tblClienteServicio guardamos el número de teléfono del usuario con el servicio al que se encuentra suscrito, la fecha en la que se dio de alta, el estatus es 1 si el usuario de dio de alta y el usuario se ha dado de baja del sistema el estatus es 0. También guardamos la hora en que se dio de baja.

La tabla tblServicio guarda los distintos servicios que pueda tener el sistema con un identificador único en este caso del tipo autoincrementa.

La tabla tblControlWeb guardamos la tabla con los datos de los usuarios que accedan al sistema por medio de una página web aquí se guarda el último registro de quien activo o desactivo el sistema, Este registro puede ser consultado desde la web.

En la tabla Usuarios tenemos el número del Usuario con su password correspondiente para acceder desde la página web.

La tabla Comando se guarda el comando de activación y desactivación que es consultado por el programa de gestión para enviar datos al dispositivo de detección (Placa Arduino).

2.14 Estructura y Comandos de acceso SMS

Para teléfonos móviles, SMS usa el codificado PDU en el cual el mensaje es encapsulado. Esta estructura se le da al teléfono móvil para realizar el evento de enviar o recibir mensajes SMS.

2.14.1 Leer un mensaje SMS

El comando AT+CMGR=1 lee el mensaje en la de ubicación 1. Cada ves que llega un nuevo mensaje es indexado en una dirección de memoria del teléfono móvil. Como se observa en la *Figura 34.*

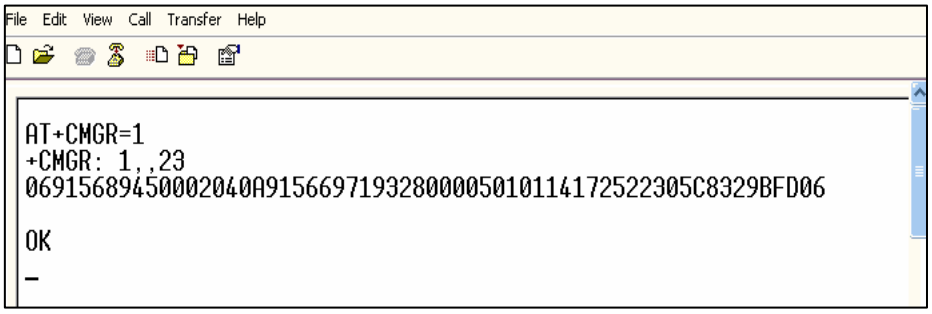

Fig. 34: Lectura de un mensaje SMS.

El SMS recibido puede ser decodificado en base a la siguiente estructura. Como se muestra en la *Tabla 6*.

Tabla 6: Estructura de mensaje recibido.

Byte	Dato	Lo que significa el dato	Definición
0	06	N = 6, "91" - 1 byte "56 89 45 00 02" - 5 bytes	Largo del SMSC - N Tamaño del SMSC + Numero.
1	91	numero Internacional	Tipo de numero SMSC 0x81 – numero de plan por defecto 0x91 – numero Internacional 0xA1 – Numero de plan Nacional
2-6	5689450002	Numero SMSC = +65-98540020	El Numero SMSC actual
7	04	Siempre es 04	Primer Octeto del msj. SMS
8	0A	M = 10 bytes	El largo del tipo y numero del remitente
9	91	Numero Internacional	Tipo de numero de remitente 0x81 - numero de plan por defecto 0x91 - numero Internacional 0xA1 - Numero de plan Nacional
10 to 14	5669719328	El numero del remitente actual : +65-96173982	Numero del remitente.
15	00	-	Protocolo identificador
16	00	-	Esquema de código de datos

17 to 23	50101141725223	05 – Año 01- Enero 11 – Dia del Mes 14:27:25:32 – Hora	Dias y Horas. 50101141725223->05-01-11-14-27-25-32
24	05	Largo del Msj. Actualmente recibido , en este caso "Hello"	Largo del Msj. Actual en hexadecimal.
25 to 29	C8329BFD06	Este es el Nuevo Mensaje Recibido "Hello"	Mensaje Actual Recibido codificado en formato de 7 bits

El mensaje recibido es representado por un formato de 7 bit. El mensaje "Hello" seria representado por la cadena "0x48 0x65 0x6C 0x6F" de 7 bits. Como se observa en la Figura 35.

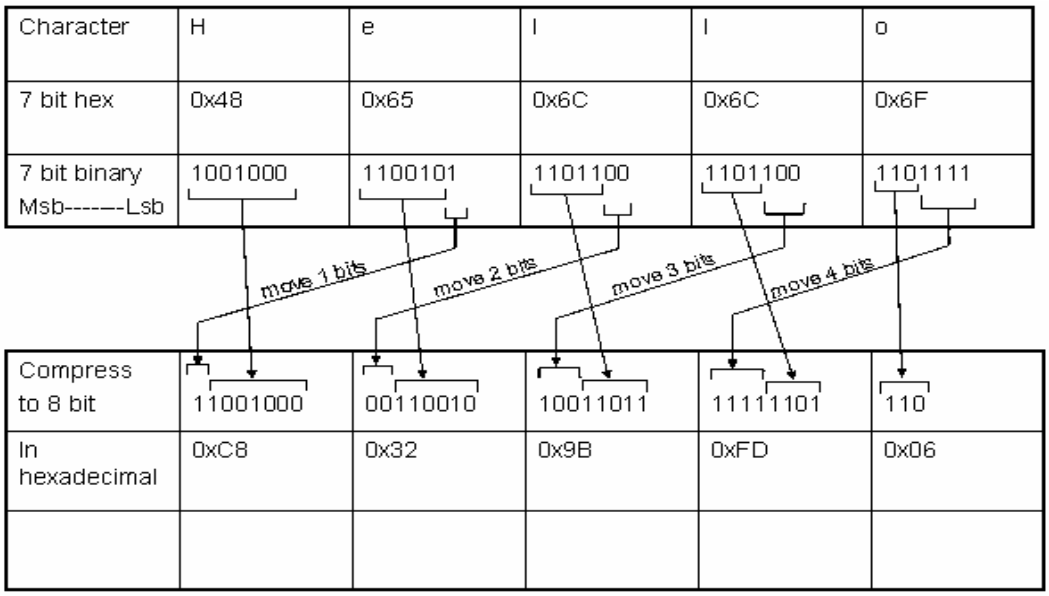

Fig. 35: Modelo de compresión de un mensaje SMS

2.14.2 Enviar un SMS

Para enviar un mensaje SMS, se usa el comando GSM: AT+CMGS. Como se observa en la *Figura 36*.

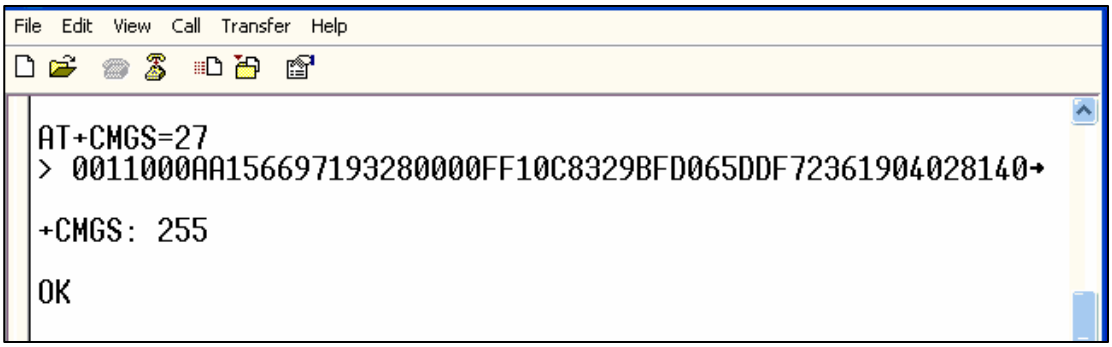

Fig. 36: Envio de un mensaje SMS.

El comando AT+CMGS = 27, pre-notifica al celular que el PDU que será enviado contiene 27 bytes de datos. El dato pasado al teléfono después del "AT+CMGS" contiene al PDU. La siguiente *Tabla 7* muestra la estructura del mensaje enviado:

Tabla 7: Estructura del mensaje enviado.

Byte	Dato	Significado del Dato	Definición
0	00	Usa la Información SMSC interna del	Largo del Información SMSC. Aquí el largo es 0, lo que significa que el SMSC guardado en el teléfono tiene que ser usado.
1	11	Siempre 11	Primer octeto del envió de mensaje.
2	00	-	Mensaje de referencia. El valor "00" configura el numero de mensaje por referencia del teléfono.

3	0A	Largo total de "A1" + "566971932	Largo del tipo y del número de recipiente.
4	A1	Numero de Plan Nacional	Tipo de numero de recipiente 0x81 - Numero de plan por defecto 0x91 - Numero Internacional 0xA1 - Numero de plan Nacional
5 to 9	5669719328	Numero del SMS de destino:	El numero del SMS en el recipiente.
10	00	-	Protocolo Identificador
11	00	-	Esquema de codificación de dato.
12	FF	Ignorado	Periodo de Validez
13	10	El Msj. Posee un largo de 16 caracter es	Largo del Mensaje actual en Hexadecimal.
14 to 27	C8329BFD065DDF723619 04028140	Este es el Msj. Codificado a enviar	El Mensaje codificado a enviar.

2.15 Instalación y Configuración de la Pasarela SMS

La pasarela de Mensajería que utilizamos es Kannel es de licencia BSD software libre. En una instalación orientada a SMS, desde el teléfono celular se envía un mensaje que es recibido por el SMSC u otro teléfono móvil, que recibe el mensaje que accede a una página web para generar páginas dinámicas, comandos o lanzar aplicaciones. Instalamos la pasarela siguiendo usando este comando para instalar la plataforma:

pi@raspberrypi:~ $ ***sudo apt –get install kannel***

Editamos el archivo de configuración [16] *kannel.conf* en el directorio /etc/kannel

 # Sample configuration file for Kannel bearerbox on Debian.
 # See the documentation for explanations of fields.
group = core
admin-port = 13000 ***// Puerto Administrativo***
admin-password = bar
admin-allow-ip = "127.0.0.1"
smsbox-port = 13001 ***// Puerto para el Envio de Mensaje***
*log-file = "/var/log/kannel/bearerbox.log****" //localizacion del archivo log***
box-allow-ip = "127.0.0.1" ***// direccion ip permitida a comunicarse***

#SMSC
group = smsc
smsc = at
smsc-id = out
modemtype = ztemodem
device = /dev/ttyUSB2 ***//localización del Modem en el Sistema Operativo***
speed = 9600 ***//Velocidad del Dispositivo***
sms-center = +5058105137 ***//número del centro de SMSC***
log-file = "/var/log/kannel/modem.log" ***//localizacion del archivo log del modem***
log-level = 0

 # Modem SETUP
group = modems ***//Configuración del Modem GSM ZTE***
id = ztemodem
name = "ZTE"
#detect-string = "ATZ"
detect-string = "ATQ0 V1 E1 S0=0 &C1 &D2
+FCLASS=0;+CNMI=1,2,0,1,0;+CMGF=1" ***//Cadena de detección y configuración***
init-string = "AT+CNMI=1,2,0,0,0"
keepalive-cmd = "AT+CBC;+CSQ" ***//keep alive consultando potencia***

 # SEND-SMS USERS
group = sendsms-user
username = tester ***// usuario para envío de Mensajes***
password = foobar ***// password para envío de Mensajes***

#SMS SERVICE 'Activacion'
group = sms-service

keyword = alta
#text = "Gracias por usar este servicio"
get-url = http://localhost/AltaProduccion.php?phone=%p&destino=%P&text=%a
*max-messages = **0 // al obtener el texto con la cadena alta ejecutamos este script en php***
catch-all = true

#SMS SERVICE 'Baja'
group = sms-service
keyword = baja
#text = "Gracias por usar este servicio"
*get-url = http://localhost/BajaProduccion.php?phone=%p&destino=%P&text=%a **// al obtener el texto con la cadena alta ejecutamos este script en php***
max-messages = 0
catch-all = true

2.16 interfaz HTTP para enviar mensajes SMS

Después de haber configurado la plataforma de mensajería Kannel podemos enviar mensajes de texto a través de HTTP. Por ejemplo, usando un navegador web. Escribimos una URL como la siguiente:

http://localhost:13013/cgi-bin/sendsms?username=foo&password=bar&to=numero del telefono&text=texto a Enviar

Esto es una petición HTTP GET para enviar el mensaje creamos una función en PHP. A continuación se muestra la función implementada.

```
function EnviarSMS ($IdCliente,$strMensaje)
{
    $hostname="localhost:13013" ;
    $rutaCGI= "/cgi-bin/sendsms";
    $Usuario="tester";
    $Pass="foobar";
    $from="88888900"   ;
    $strUrl = "http://"
.$hostname.$rutaCGI."?username=".$Usuario."&password=".$Pass."&to=".urlenco
de($IdCliente)."&text=".urlencode($strMensaje)."&from=".$from;
    $curl_handle = curl_init ($strUrl);
 //    Perform the GET and get the data returned by the    server.
        $result = curl_exec ($curl_handle) or die ('Error en la Ejecución
del Servidor HTTP');

        //    Close the CURL handle
        curl_close ($curl_handle);
}
```

2.17 Procesamiento de los Mensajes de Entrada

Los mensajes SMS que obtenemos de la pasarela de mensajería son procesados por medio de un servidor HTTP. Elegimos Apache porque es de código abierto y es para plataformas UNIX (BSD, LINUX) es usado principalmente para enviar páginas web estáticas y dinámicas en Internet. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP y Python. Una aplicación del lado del servidor es cualquier programa o conjunto de instrucciones diseñadas con la finalidad de que un servidor web las procese para realizar alguna acción. Las aplicaciones del lado del servidor están escritas mediante algún lenguaje de programación. En este sistema el servicio web es desarrollado en PHP.

Obtenemos el número del usuario y el texto del SMS por medio de una petición $GET, esta información es proporcionada por la pasarela de mensajería. Si el contenido del SMS es alta alarma verificamos en la base de datos en la tblCliente si no existe el usuario (número de teléfono) lo insertamos en la tabla tblClienteServicio el número de teléfono, el Servicio que es alarma, la hora y fecha y YnEstatus=1, esto crea un usuario valido en el sistema y le enviamos un SMS informándole que fue dado de alta en el sistema usando la función EnviarSMS que fue descrita en la sección anterior del documento. La baja de un usuario actualiza el campo YnEstatus=0 lo que significa que dio de baja el servicio de alarma de la misma manera se le envía un mensaje SMS informándole que se fue dado de baja. En la *Figura 37* se observa la inserción en la base de datos el numero y el servicio activado.

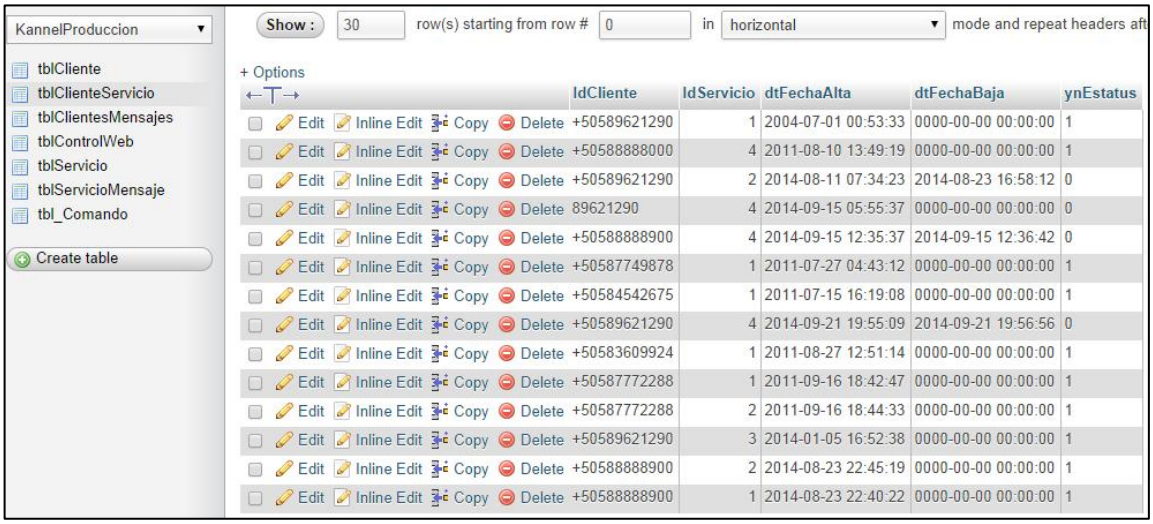

Fig. 37: Inserciones en la Base de Datos del número y el servicio activado.

En la ejecución del script se hacen consultas SQL a la base de datos donde se inserta la información necesaria el número del teléfono, la hora y fecha de la ejecución, y el comando a llevarse a cabo y si el cliente tiene permisos de ejecutar el comando esta es la parte de información física almacenada en el disco duro y que puede ser consultada en cualquier momento. En la *Figura 38* se observa el diagrama de flujo para el procesamiento del mensaje entrante.

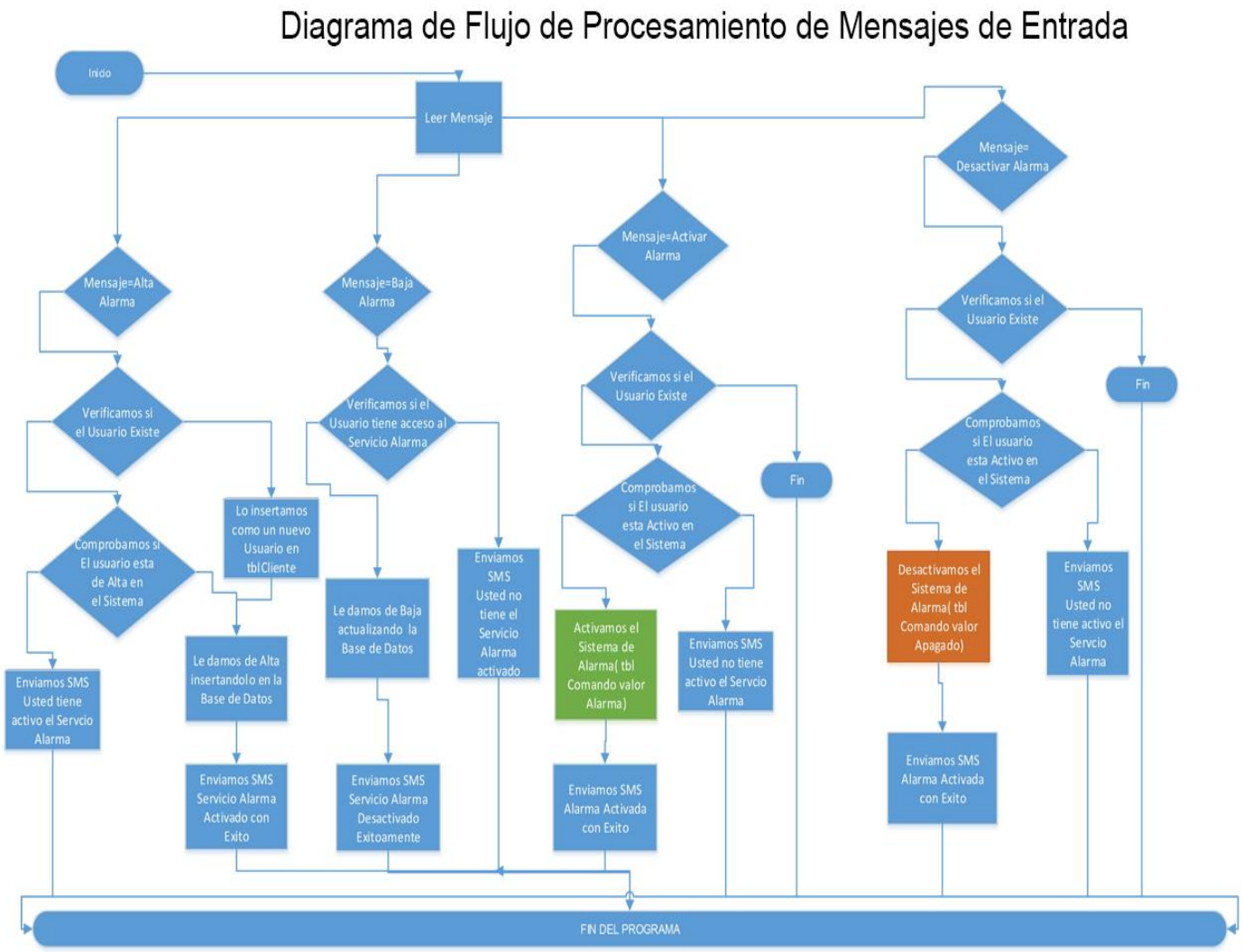

Fig. 38: Diagrama de flujo de procesamiento de mensaje entrante.

Esta etapa es importante puesto que hace llegar al circuito de control las órdenes para activar o desactivar el sistema de control. Los comandos que se utilizaron fueron palabras como: activar alarma, apagar alarma, para encender y apagar respectivamente; En la siguiente *Tabla 8* se observa el juego de comandos para interactuar con el sistema de control y notificación de alarmas:

Tabla 8: Comandos SMS para el sistema de control de alarmas.

Comando Texto SMS	Descripción
alta alarma	Dar de alta a usuarios al servicio
baja alarma	Dar de baja a usuarios al servicio
activar alarma	Activar el Funcionamiento de la Alarma
desactivar alarma	Desactivar el Funcionamiento de la Alarma
encender Dispositivo	Encender Luminaria
apagar Dispositivo	Apagar Luminaria

2.18 Descripción de acción del Sistema

A continuación se describen paso a paso el proceso de envío y recepción de mensajes de la aplicación.

- Un usuario de telefonía móvil se inscribe al Sistema de Alarma.
- El mensaje SMS es recibido por el dispositivo móvil conectado al servidor (Modem conectado a la RaspberryPi).
- La pasarela de mensaje obtiene el contenido del texto y el número telefónico del usuario compara el texto con un keyword establecido si son idénticos ejecuta un script php por medio de una petición HTTP.
- El servidor interpreta ejecuta consultas e inserciones un una base de datos Si el usuario se inscribió correctamente o hubo errores es comunicado a través de un mensaje de texto.
- Para la activación del sistema de alarma el usuario envía un SMS con un texto predefinido, el sistema obtiene el dato y ejecuta un script donde se confirma si

el número que envío el mensaje es un usuario que esta de alta en el sistema y escribe en una tabla de la base de datos.

- El programa de gestión en Python lee la base de datos si el campo de la tabla hay un valor especifico se encarga de abrir la comunicación serial y enviar un carácter de contenido al sistema de detección si el programa de detección responde correctamente, el programa de gestión envía un mensaje SMS al usuario indicando la acción realizada.

A continuación el script que se encarga de darle de alta al usuario:

```
<?php
    require_once ('Funciones.inc');
    //$phone=$_GET["phone"];
    //$texto= $_GET["text"];
   //Verifica Conexion
  $conexion=mysql_connect("localhost",user","password")  or die("Problemas
en la conexion");
  mysql_select_db("KannelProduccion",$conexion) or
  die("Problemas en la selección de la base de datos");

  $query = mysql_query("SELECT IdCliente FROM tblCliente WHERE IdCliente
LIKE '$phone'");
   if (!mysql_num_rows($query){
 $query1 = mysql_query("Insert INTO tblCliente(IdCliente) values
('$phone')") or die ("Error!");
  }
  if (strstr($texto,' ')){
    $Servicio = trim(strstr($texto,' '));
$queryServ = mysql_query("SELECT IdServicio FROM tblServicio WHERE
strDescripcion LIKE '$Servicio'");
   if(mysql_num_rows($queryServ)){
        $fila=mysql_fetch_array($queryServ);
        $IdServicio = $fila['IdServicio'];
        $queryClientServ = mysql_query("SELECT * FROM tblClienteServicio
WHERE IdCliente LIKE '$phone' and IdServicio = '$IdServicio'");
if(!mysql_num_rows($queryClientServ)){
 $query1 = mysql_query("Insert INTO
tblClienteServicio(IdCliente,IdServicio,dtFechaAlta,ynEstatus) values
('$phone',$IdServicio,now(),1)") or die ("Error!");
 $MensajeActivar= "Servicio '$Servicio' activado Correctamente";
  EnviarSMS($phone,$MensajeActivar);
$phonepass= service_alarma($phone,$conexion);
 if ($phonepass==true){
$querypasswd=mysql_query("Insert INTO usuarios(usuario,password) values
('$phone','$phonepass')") or die("Error en la Insercion del Password
Warning");
```

```
echo $phonepass;
$MensajePassword="Este es su password para gestionar el Sistema de Alarma
por medio de Internet: $phonepass";
EnviarSMS($phone,$MensajePassword)
}

else{

$aviso_pass= "Usted ya tiene un password para este Servicio $Servicio";

EnviarSMS($phone,$aviso_pass);
$Mensaje = RandomMensaje($phone,$IdServicio);
EnviarSMS($phone,$Mensaje); }
else{

$queryClientServUpdate = mysql_query("SELECT * FROM tblClienteServicio
WHERE IdCliente LIKE '$phone' and IdServicio = '$IdServicio' and
ynEstatus='1'");

 if (mysql_num_rows($queryClientServUpdate)){
 $MensajeActivar= "Usted ya tiene activo el servicio '$Servicio'";
 EnviarSMS($phone,$MensajeActivar);
   }
    else{
  $query1 = mysql_query("Update tblClienteServicio set dtFechaAlta = now(),
ynEstatus=1 where IdCliente LIKE '$phone' and IdServicio = '$IdServicio'")
or die ("Error!");
   $MensajeActivar= "Servicio '$Servicio' Reactivado";
   EnviarSMS($phone,$MensajeActivar);
       }
       }
       }
       else{
       $MensajeActivar= "No existe el servicio";
       EnviarSMS($phone,$MensajeActivar);
             }

      }
                else{
                    $MensajeActivar= "No especifico el Servicio";
                    EnviarSMS($phone,$MensajeActivar);
                }

   ?>
```

A continuación el script que se encarga de darle de baja al usuario.

```
<?php
    require_once ('Funciones.inc');
    $phone=$_GET["phone"];
    $texto= $_GET["text"];
 //Verifica Conexion
  $conexion=mysql_connect("localhost","root","raspberry")
  or die("Problemas en la conexion");
  mysql_select_db("KannelProduccion",$conexion) or
  die("Problemas en la seleccion de la base de datos");
  $query = mysql_query("SELECT IdCliente FROM tblCliente WHERE IdCliente
LIKE '$phone'");
  if (!mysql_num_rows($query)){
  $MensajeBaja= "Su Numero no se encuentra registrado";
  EnviarSMS($phone,$MensajeBaja);
  }

   else{
        if (strstr($texto,' ')){
                    $Servicio = trim(strstr($texto,' '));
                    $queryServ = mysql_query("SELECT IdServicio FROM
tblServicio WHERE strDescripcion LIKE '$Servicio'");
                    if(mysql_num_rows($queryServ)){
                    $fila=mysql_fetch_array($queryServ);
                    $IdServicio = $fila['IdServicio'];
                    $queryClientServ = mysql_query("SELECT * FROM
tblClienteServicio WHERE IdCliente LIKE '$phone' and IdServicio =
'$IdServicio'");
if(!mysql_num_rows($queryClientServ)){
$MensajeBaja= "Usted no tiene el Servicio '$Servicio' activado";
EnviarSMS($phone,$MensajeBaja);
}
Else {
$queryClientServUpdate = mysql_query("SELECT * FROM tblClienteServicio
WHERE IdCliente LIKE '$phone' and IdServicio = '$IdServicio' and
ynEstatus='0'");
 if (mysql_num_rows($queryClientServUpdate)){
 $MensajeBaja= "Usted ya tiene Desactivado el servicio '$Servicio'";
 EnviarSMS($phone,$MensajeBaja);
 }
 else{
 $query1 = mysql_query("Update tblClienteServicio set dtFechaBaja = now(),
ynEstatus=0 where IdCliente LIKE '$phone' and IdServicio = '$IdServicio'")
or die ("Error!");
$MensajeBaja="El Servicio '$Servicio' ha Sido Desactivado Exitosamente";
EnviarSMS($phone,$MensajeBaja);
}
}
}
else{
$MensajeBaja= "No existe el servicio";
EnviarSMS($phone,$MensajeBaja);
```

```
}
}
  else{
  $MensajeBaja= "No especifico el Servicio";
  EnviarSMS($phone,$MensajeBaja);
      }
   }
  ?>
```

A continuación el script que se encarga de activar el sistema de alarma (Detección).

```
<?php
  require_once ('Funciones.inc');
  require("php_serial.class.php");

  $phone=$_GET["phone"];
  $texto=$_GET["text"];

//Verificar conexion con la base de datos
$conexion=mysql_connect("localhost","root","raspberry")or die("Problemas en
la Conexion con MySQl");
  mysql_select_db("KannelProduccion",$conexion) or die("Problemas en la
Seleccion de la Base de Datos");
    $query = mysql_query("SELECT IdCliente FROM tblCliente WHERE IdCliente
LIKE '$phone'");
if (!mysql_num_rows($query)){
$aviso= "Usted no tiene acceso a este servicio ";
EnviarSMS ($phone,$aviso);
}
if (strstr($texto,' ')){
$Servicio = trim(strstr($texto,' '));
$queryServ = mysql_query("SELECT IdServicio FROM tblServicio WHERE
strDescripcion LIKE '$Servicio'");
if(mysql_num_rows($queryServ)){
$fila=mysql_fetch_array($queryServ);
$IdServicio = $fila['IdServicio'];
$queryClientServ = mysql_query("SELECT * FROM tblClienteServicio WHERE
IdCliente LIKE '$phone' and IdServicio = '$IdServicio'");

if(mysql_num_rows($queryClientServ)){
$serial = new phpSerial();
//Puerto Serial
$serial->deviceSet("/dev/ttyACM0/");
//Parametros de Configuracion 9600 8-N-1, so
$serial->confBaudRate(9600);
$serial->confParity("none");
$serial->confCharacterLength(8);
$serial->confStopBits(1);
$serial->confFlowControl("none");
$serial->deviceOpen();
```

```
$serial->sendMessage("H");
$serial->deviceClose();
$MensajeActivar= "Servicio '$Servicio' activado Correctamente";
EnviarSMS($phone,$MensajeActivar);
}
}
}
?>
```

Los otros casos que no mencionamos tienen la misma similitud con los códigos anteriormente adjuntos, los scripts completos para su análisis se encuentran en los **Anexos 2** de este documento. En la *Figura 39* observamos el diagrama de interconexión del sistema.

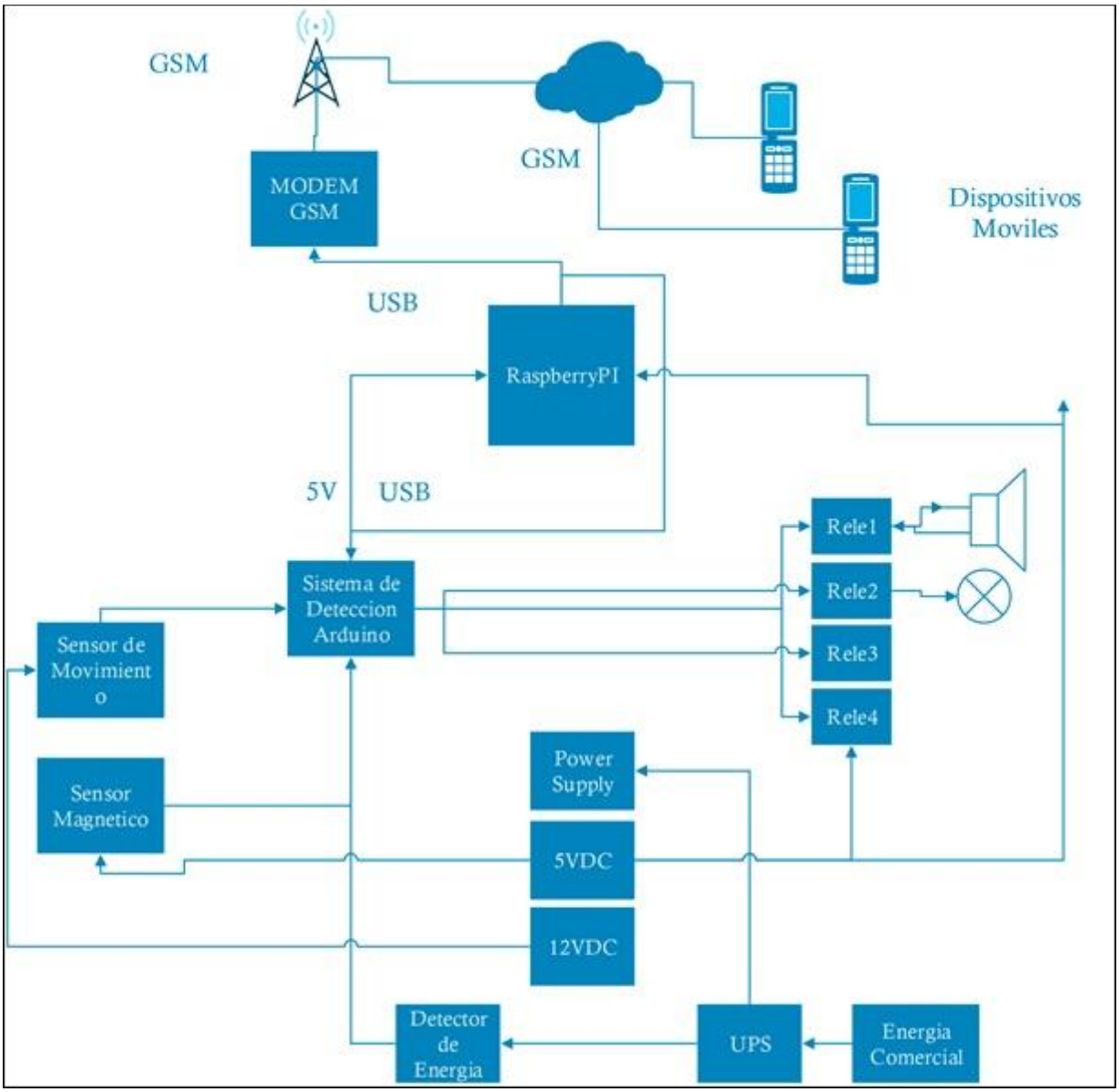

Fig. 39: Diagrama de interconexión del sistema.

2.19 Gastos de materiales del sistema

A continuación se describen los gastos de materiales del proyecto y se evidencia la factibilidad del mismo.

Entre los gastos recurrentes que se presentan en la implementación del sistema se encuentra el pago de USD$10 mensuales para la activación de paquetes de

mensajes de textos. Se puede observar en la tabla el costo total del sistema, y para entrar en el mercado se lo ha fijado en 180 USD (Ciento ochenta dólares americanos). Como se observa en la *Tabla 9.*

Tabla 9: Costos de materiales del Proyecto.

Elementos del sistema de control			
Elementos	**Cantidad**	**Precio/ Unidad**	**Total**
Modem GSM	1	20	20
Raspberry Pi	1	40	40
Caja de Plástico	1	10	10
Arduino	1	25	25
Módulo de 4 Relay	1	10	10
Sensor magnético	1	10	10
Sensor PIR	1	10	10
Tarjeta de memoria SD	1	10	10
Adaptador de 5 V	1	5	5
Bateria UPS	1	30	30
Paquete de SMS Recurrente	1	10 Mensual	10
Total del sistema			$180

CAPITULO III

ANALISIS Y PRESENTACION DE RESULTADOS

Teniendo en cuento los objetivos planteados, se cumplieron los objetivos al crear un sistema de alarmas (SMS) funcional y de bajo costo que puede ser adaptado a las necesidades del usuario, debido a que la mayor parte del funcionamiento es controlado por software.

El sistema demostró fiabilidad, escalabilidad para monitorear y controlar procesos de otra índole como en domotica y procesos industriales.

En el siguiente capítulo se describen los resultados obtenidos del presente proyecto, basados en el diseño explicado en el capítulo anterior. En el siguiente capítulo se describen los resultados obtenidos del presente proyecto, basados en el diseño explicado en el capítulo anterior.

3.1 Instalación

Para la instalación del sistema se analizó la vivienda para poder ubicar los dispositivos de manera adecuada, tomando en cuenta los requerimientos de seguridad. La *Figura 40* muestra la vivienda en que se instaló el sistema de notificación de alarmas.

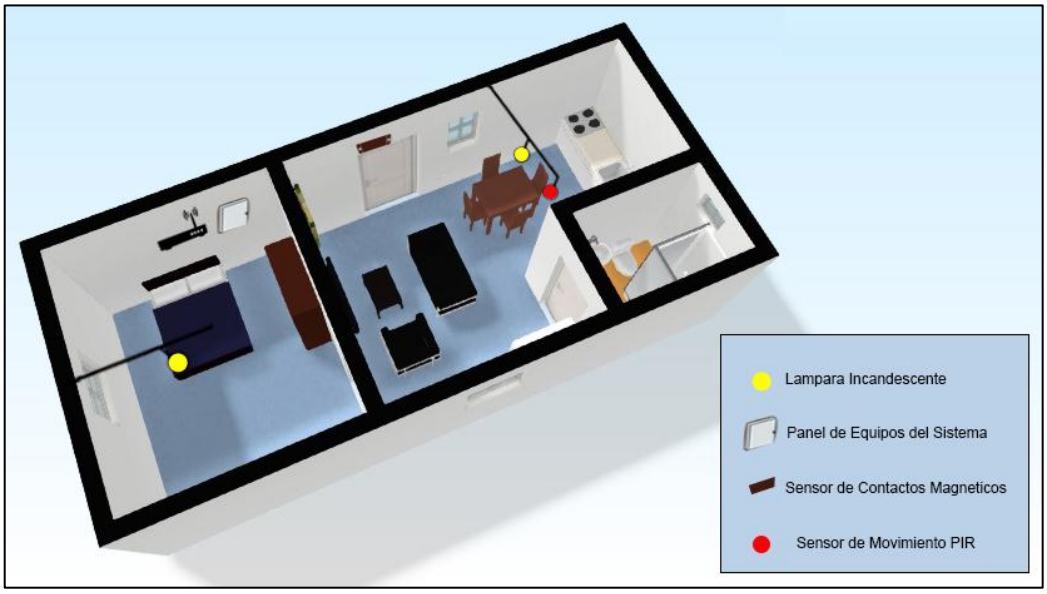

Fig. 40: Vivienda en que se instaló el sistema

Se buscó que la ubicación del sistema de control preste facilidad para el acceso a la red GSM, ya que la fiabilidad del sistema depende de la calidad de la red celular. Para la ubicación del panel central se buscó un lugar que preste la facilidad para el cableado de todos los dispositivos, ya que, este es el cerebro del sistema y de este salen todas las conexiones, es por esto que la ubicación de este debe ser estratégica. La *Figura 41* muestra la ubicación de la central. Con todas sus conexiones, en la parte izquierda está la fuente de poder que lo alimenta con voltajes de 12Vcd y 5Vcd, en la parte superior la tarjeta Raspberry Pi con el modem GSM. Puesto que se instaló una UPS posee una sola toma de corriente. En la parte central se observa la tarjeta Arduino y el módulo de Relés donde se conectan los distintos dispositivos como son: iluminación, timbre electrico y los sensores.

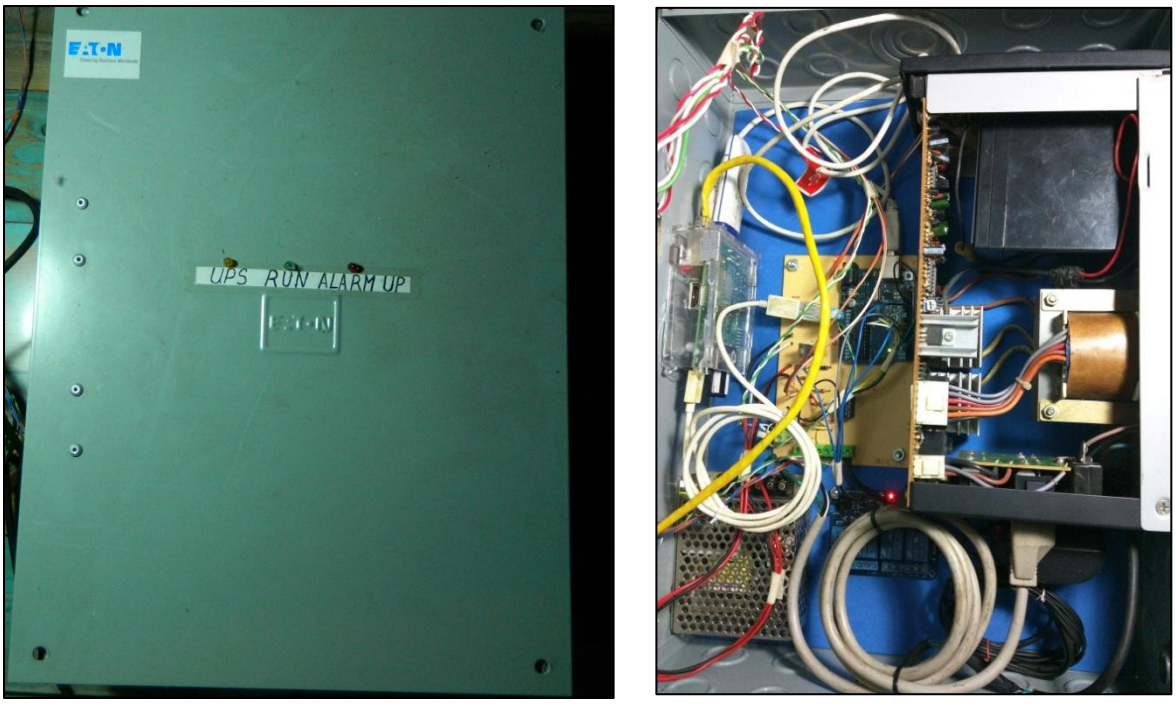

Fig. 41: Modulo central del sistema.

Los dispositivos que conforman el sistema de seguridad son: contacto magnético para puertas y sensor infrarrojo de movimiento, se observan e la *Figura 42*, los sensores alertan sobre un intruso en la vivienda mediante la activación de un timbre en la vivienda y un SMS al usuario, solo en caso que la seguridad del domicilio este activada.

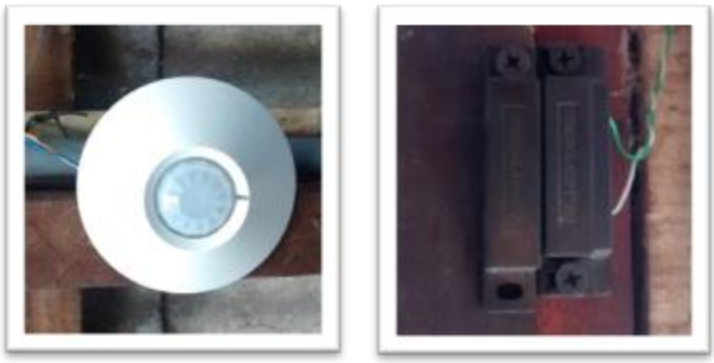

Fig. 42: Sensor infrarrojo de movimiento y contacto magnético

La parte de control de dispositivos se aplicó a la iluminación, que es solo una pequeña muestra de la cantidad de dispositivos que se podrían controlar. Para la iluminación la central solamente funciona como interruptor. El sistema eléctrico de la vivienda se encarga de proporcionar el voltaje de 110VCA para alimentar la bujía, Este dispositivo lo observamos en la *Figura 43*.

Fig. 43: Iluminación conectada al sistema.

3.2 Resultados

Como resultado de este proyecto fue un sistema de control y notificación de alarmas por medio del servicio SMS, instalado y probado en una vivienda. A continuación se muestran las pantallas con las salidas de datos correspondientes a la aplicación en su etapa de pruebas y etapa final.

3.2.1 Leyendo mensaje entrante

Como primer resultado se envió el mensaje de texto Alta alarma al sistema solicitando dar de alta al usuario como se observa en la *Figura 44*.

Fig. 44: Mensaje de texto Alta alarma y recepción de pasarela SMS.

Se recibió el mensaje de activación correctamente por parte de la pasarela de mensajes SMS, se agrego un nuevo usuario al servicio. Insertándolo a las base de datos con el servicio activo de manera correcta.

3.2.2 Modo de detección de alarma por SMS

Se realizó con éxito la activación del modo detección mediante el envío de mensajes de texto Activar alarma y se recibió el mensaje de confirmación por parte del sistema. También se realizó con éxito por parte del sistema el envío de mensajes de texto al abrirse la puerta y detectar movimiento como se observa en la *Figura 45*.

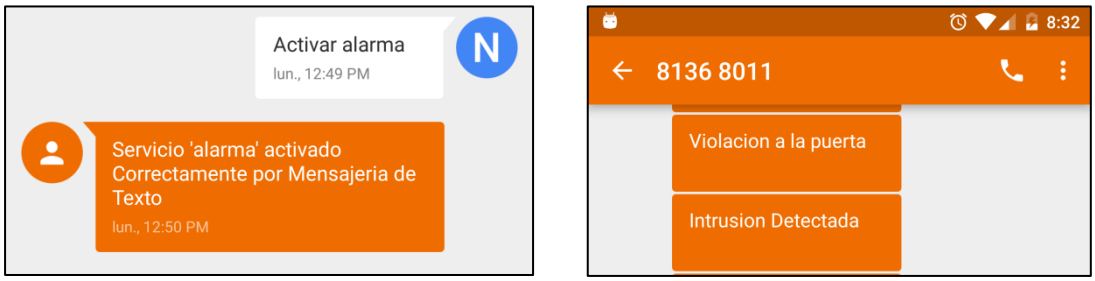

Fig. 45: Activación de alarma y notificación de intruso.

en la siguiente *Figura 46* se observa el modo detección activado y el envio de un mensaje SMS.

Fig. 46: Programa Principal Monitoreando Arduino y Base de Datos

Se obtuvieron los mensajes de texto de todos los eventos establecidos, dar de alta al usuario de baja, la activación de la alarma, su desactivación y la detección de intrusos y los estados de energía del sistema si está alimentando de la energía comercial o por su sistema de respaldo.

El Sistema estuvo en un ambiente de prueba en la vivienda por espacio de una semana sin tener ningún error en él envió de mensajes. El sistema es un prototipo y presento altas prestaciones relacionados con la fiabilidad y escalabilidad del sistema para desarrollar diferentes aplicaciones asociadas al tema.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Se realizó con éxito el diseño e implementación del sistema de notificación de alarmas gracias a la facilidad en la integración de las plataformas Arduino y Raspberry Pi.
- La notificación de alarmas y control del hogar se puede hacer desde cualquier parte del país con cobertura GSM si es vía SMS en dispositivos móviles desde gama alta a baja.
- El sistema actúa siempre y cuando el número del usuario este registrado en la base de datos con el servicio activo.
- Al trabajar con el sistema operativo Linux como es el caso del presente proyecto, se hace más fácil el desarrollo de una aplicacion cliente servidor, debido a la cantidad de herramientas y a la documentación disponible en internet.
- Se pueden personalizar los sistemas de notificación y control de alarmas de acuerdo a las necesidades de cada usuario sin incurrir en grandes costos de hardware especializado.

RECOMENDACIONES

- Se recomienda la instalación de una UPS para el respaldo de energía. Debido a que en el país la energía comercial presenta interrupciones momentáneas o permanentes del servicio y se necesita un suministro ininterrumpido de energía.
- El presente proyecto puede servir como punto de partida para ser ampliado en alcance. diseñar e implementar otras interfaces además de la explicada en el presente documento.
- Instalar el sistema en un lugar estratégico, es decir poco visible pero no sin descuidar la calidad de señal en el módulo GSM, para envío y recepción de los SMS.
- Se pueden añadir al sistema entradas para la conexión de sensores de gas, cerraduras eléctricas, etc. y con esto desarrollar un sistema más robusto.
- Con la constante evolución de la tecnología se tiene que innovar este tipo de sistemas y utilizar las nuevas herramientas tecnológicas como aplicaciones móviles .
- Se recomienda para mejor tiempo de respuesta del sistema utilizar el modelo de Raspberry Pi 2 que tiene una velocidad de inicialización del sistema operativo más rápido que su predecesora.
- Una opción viable para proveer redundancia y acceso remoto al sistema es el desarrollo de una interfaz web interactiva para el control y monitoreo del sistema. Esto a la vez provee acceso remoto desde cualquier lugar.
- Si en el Sistema implementamos un acceso web, el sistema tendría un costo recurrente de web hosting y de un dominio para brindarle al sistema acceso remoto por medio de Internet, la modalidad de pagos de este servicio es anual con un costo de unos 40 a 60 dólares americanos aproximadamente.
- Para que el sistema posee tolerancia a fallos en la red GSM se recomienda el uso de una sistema de redundancia por medio de acceso via Web al sistema y

a notificaciones por correo electrónico, en caso de que el acceso o falla en la red GSM ya que el único canal hacia la red GSM es un modem o varios, con un acceso web es una ruta alterna para establecer comunicación con el sistema.

- EL sistema de respaldo de energía con que cuenta el sistema tiene un intervalo de tiempo de trabajo de aproximadamente de 2 horas, si este sistema será implementado en lugares con fallas en el suministro eléctrico recomendamos diseñar un sistema de respaldo con un inversor y baterías de ciclo profundo o con células fotovoltaicas.

BIBLIOGRAFIA

[1] Wayne Tomasi, *Sistemas de comunicaciones electrónicas*.: Pearson Educación, 2003.

[2] Eugenio Rey, *Telecomunicaciones Moviles*. Barcelona: MARCOMBO, S.A., 1998.

[3] O. Cesar Augusto Jara and C. Wilfredo Alberto Almeida. (2011) [Online]. http://www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-91420.pdf

[4] Sergio Machuca and Sergio Nesmachnow. (2003) [Online]. http://cita2003.fing.edu.uy/articulosvf/54.pdf

[5] José Luis Valenzuela González Oriol Sallent Roig, *Principios de Comunicaciones Móviles*.: Ediciones UPC, 2003.

[6] Theodore S. Rappaport, *Wireless communications : principles and practice*.: Prentice Hall, 2002.

[7] Juan Gonzalez Gomez. (2002, Junio) [Online]. http://www.iearobotics.com/personal/juan/doctorado/sms/sms.pdf

[8] Andres Hernandez Seco. (2001, Marzo) [Online]. http://www.alamin.org/es/faq.es.html

[9] Jorge Roberto Alvarado and Cesar Oswaldo Arevalo. (2010) [Online]. http://dspace.espoch.edu.ec/bitstream/123456789/366/1/38T00175.pdf

[10] Massimo Banzi, *Getting Started with Arduino*.: Maker Media, Inc, 2009.

[11] Simon Monk, *Programming the Raspberry Pi: Getting Started with Python*., 2012.

[12] Andreas Fink, Bruno Rodrigues, and Stipe Tolj. (2014, Septiembre) [Online]. http://www.kannel.org/download/kannel-userguide-snapshot/userguide.pdf

[13] Wikipedia. [Online]. https://es.wikipedia.org/wiki/LAMP

[14] Wikipedia. [Online]. https://es.wikipedia.org/wiki/Arduino

[15] Wikipedia. [Online]. https://es.wikipedia.org/wiki/PhpMyAdmin

[16] Kannel.org. [Online]. http://www.kannel.org/userguide.shtml

[17] Carlos A. Reyes, *Microcontroladores Programacion en Basic*., 2006.

[18] J. Padron. (2006, Septiembre) Investigar, reflexionar y actuar en la practica docente. [Online]. http//padron.entretemas.com/

ANEXOS

Anexo 1: Tabla de comandos AT y AT+

Las siguientes listas contienen los comandos AT más usados dividida en 5 grupos:

1. **Comandos generales**

a) **AT+CGMI**: Identificación del fabricante

b) **AT+CGSN**: Obtener número de serie

c) **AT+CIMI**: Obtener el IMSI.

d) **AT+CPAS**: Leer estado del modem

2. **Comandos del servicio de red**

a) **AT+CSQ**: Obtener calidad de la señal

b) **AT+COPS**: Selección de un operador

c) **AT+CREG**: Registrarse en una red

d) **AT+WOPN**: Leer nombre del operador

3. **Comandos de seguridad:**

a) **AT+CPIN**: Introducir el PIN

b) **AT+CPINC**: Obtener el número de reintentos que quedan

c) **AT+CPWD**: Cambiar password

4. **Comandos para la agenda de teléfonos**

a) **AT+CPBR**: Leer todas las entradas

b) **AT+CPBF**: Encontrar una entrada

c) **AT+CPBW**: Almacenar una entrada

d) **AT+CPBS**: Buscar una entrada

5. **Comandos para SMS**

a) **AT+CPMS**: Seleccionar lugar de almacenamiento de los SMS

b) **AT+CMGF**: Seleccionar formato de los mensajes SMS

c) **AT+CMGR**: Leer un mensaje SMS almacenado

d) **AT+CMGL**: Listar los mensajes almacenados

e) **AT+CMGS**: Enviar mensaje SMS

f) **AT+CMGW**: Almacenar mensaje en memoria

g) **AT+CMSS**: Enviar mensaje almacenado

h) **AT+CSCA**: Establecer el Centro de mensajes a usar

i) **AT+ WMSC**: Modificar el estado de un mensaje

Anexo 2: Código fuente del sistema de detección.

PROGRAMA ARDUINO

```
// Arduino

// Se declaran las variables que utilizar para almacenar el
Estado de los Sensores
int sensor = 0;          // Sensor de Presencia PIR
int door= 0;             // Senosr de Deteccion Magnetico
int unionf = 0;          // Estado del Detector de Energia
Comercial
int psp = 0;             // Estado del Detector de Energia de
Respaldo UPS
int habilitar=0;         // Condicion de Armado del Sistema de
Deteccion
int incomingByte;        // Buffer del Puerto Serial
int tx1=0;
int tx2=0;
int tx3=0;
int tx4=0;
int tx5=0;
int tx6=0;
void setup() {
  // initialize the pins as an output:
  Serial.begin(9600);
  pinMode(4, OUTPUT); //Led Indicador
  pinMode(2,INPUT);//sensor
  pinMode(8,INPUT);//puerta
  pinMode(5,OUTPUT);//salida rele
  pinMode(12,INPUT);//union fenosa
  pinMode(7,INPUT);//energia alterna
  pinMode(13,OUTPUT); //Dispositivo a Activar
}
void loop(){
  // read the state of the pushbutton value:
  digitalWrite(13,LOW); // Se pone la Salida del Pin 13 en Bajo
  digitalWrite(4,HIGH); // Se pone en Alto el pin 4 para el
Rele
  digitalWrite(5,HIGH); // Se pone en Alto el pin 5 para el
Rele
```

```
  digitalWrite(9,LOW);  // Se pone en bajo el LED Indicador
Alarma
  psp=digitalRead(7);   // Se lee el Detector de Energia de
Respaldo
  unionf=digitalRead(12);  // Se lee el Detector de Energia de
Energia Comercial
if (Serial.available()) {
  incomingByte = Serial.read(); // Si el serial Esta Disponible
lo que se reciba es guardado en la Variable incomingbyte
}
if (unionf == HIGH){
  if (incomingByte=='G' && tx1 == 0){
Serial.println("Energia Estable");
tx1=1;
tx2=0;
incomingByte=0;
}
}
if (incomingByte=='X'){
digitalWrite(13,LOW);
Serial.println("Dispositivo Desactivado");
incomingByte=0;

if (incomingByte=='Y'){
digitalWrite(13,HIGH);
Serial.println("Dispositivo Activado");
incomingByte=0;
}
if (psp == HIGH ){
  if (incomingByte=='G'&& tx2 == 0){
Serial.println("Energia UPS");
tx1=0;
tx2=1;
incomingByte=0;
}
}
// if it's a capital H (ASCII 72), armar sistema:
if (incomingByte == 'H') {
Serial.println("Sistema Armado");
for (int i=0;i<2;i++){
  digitalWrite(9,HIGH);
  delay(500);
  digitalWrite(9,LOW);
```

```
  delay(500);
}
habilitar = true;
}
incomingByte=0;
while (habilitar == true) {
digitalWrite(9,HIGH);
sensor=digitalRead(2);
door=digitalRead(8);
unionf=digitalRead(12);
psp=digitalRead(7);
if (Serial.available()) {
  incomingByte = Serial.read();
}
if (sensor == HIGH){
  if (incomingByte=='A' && tx3==0){
      Serial.println("Intrusion Detectada");
digitalWrite(4,LOW);
digitalWrite(5,LOW);
tx3=1;
tx4=0;
incomingByte=0;
   }
}
if (door == HIGH){
   if (incomingByte=='A' && tx4==0){
      Serial.println("Violacion a la puerta");
   digitalWrite(4,LOW);
   digitalWrite(5,LOW);
tx3=0;
tx4=1;
incomingByte=0;
  }
}
if (unionf == HIGH){
if (incomingByte=='G' && tx5==0){
Serial.println("Energia Estable");
tx5=1;
tx6=0;
incomingByte=0;
}
}
if (psp == HIGH){
```

```
if (incomingByte=='G' && tx6==0){
Serial.println("Energia UPS");
tx5=0;
tx6=1;
incomingByte=0;
}
}
if (incomingByte=='X'){
      digitalWrite(13,LOW);
      Serial.println("Dispositivo Desactivado");
      incomingByte=0;
}
if (incomingByte=='Y'){
      digitalWrite(13,HIGH);
     Serial.println("Dispositivo Activado");
     incomingByte=0;
}
if (incomingByte=='L'){
habilitar=false;
tx3=0;
tx4=0;
Serial.println("Sistema Desarmado");
incomingByte=0;
}
incomingByte=0;
}

}
```

CONFIGURACION DE LA PASARELA DE MENSAJERIA KANNEL

```
 Sample configuration file for Kannel bearerbox on Debian.
# See the documentation for explanations of fields.
#
#
# HTTP administration is disabled by default. Make sure you set
the
# password if you enable it.

group = core
admin-port = 13000
admin-password = bar
admin-allow-ip = "127.0.0.1"
```

```
smsbox-port = 13001
log-file = "/var/log/kannel/bearerbox.log"
box-allow-ip = "127.0.0.1"
#dlr-storage = mysql


#SMSC
group = smsc
smsc = at
smsc-id = out
modemtype = ztemodem
device = /dev/ttyUSB2
speed = 9600
sms-center = +5058105137
log-file = "/var/log/kannel/modem.log"
log-level = 0
sim-buffering = true
#allowed-smsc-id = out

# Modem SETUP
group = modems
id = ztemodem
name = "ZTE"
#detect-string = "ATZ"
detect-string = "ATQ0 V1 E1 S0=0 &C1 &D2
+FCLASS=0;+CNMI=1,2,0,1,0;+CMGF=1"
init-string = "AT+CNMI=1,2,0,0,0"
keepalive-cmd = "AT+CBC;+CSQ"


#SMSBOX SETUP
group = smsbox
bearerbox-host = localhost
sendsms-port = 13013
sendsms-chars = "0123456789 +-"
log-file = "/var/log/kannel/smsbox.log"
log-level = 0


# SEND-SMS USERS
group = sendsms-user
username = tester
password = foobar
```

```
#SMS SERVICE 'Activacion'
group = sms-service
keyword = alta
#text = "Gracias por usar este servicio"
get-url =
"http://localhost/AltaProduccion.php?phone=%p&destino=%P&text=%
a"
max-messages = 0
catch-all = true

#SMS SERVICE 'Baja'
group = sms-service
keyword = baja
#text = "Gracias por usar este servicio"
get-url =
"http://localhost/BajaProduccion.php?phone=%p&destino=%P&text=%
a"
max-messages = 0
catch-all = true

#SMS SERVICE 'Envi'
group = sms-service
keyword = envio
#text = "Gracias por usar este servicio"
get-url = "http://localhost/EnvioMensajeServicio.php?phone=%p"
max-messages = 0
catch-all = true

#SMS SERVICE 'Notas'
group = sms-service
keyword = nota
#text = "Gracias por usar este servicio"
get-url =
"http://localhost/nota.php?phone=%P&destino=%p&text=%a"
max-messages = 3
catch-all = true

#SMS SERVICE 'Alarma'
group = sms-service
keyword = activar
#text = "Gracias por usar este servicio"
```

```
get-url =
"http://localhost/activar2.php?phone=%p&destino=%P&text=%a"
max-messages = 0
catch-all = true


#SMS SERVICE 'Alarma'
group = sms-service
keyword = Apagado
#text = "Gracias por usar este servicio"
get-url =
"http://localhost/desactivar.php?phone=%p&destino=%P&text=%a"
max-messages = 0
catch-all = true
```

CODIGO FUENTE SISTEMA DE GESTION
PYTHON

```
#! /usr/bin/python
import sys, time,signal
import serial
import MySQLdb
import pdb
from send_sms import send2smsbox
from funcion2 import arduino_verificar
from read import poder
from sms_notificacion import envios

luces='on'
luces_on='Y'
luces_off='X'
status='A'
off = 'Apagado'
out5= ''
Servicio = 'Alarma'
energia ='G'
input='H'
output='L'
def on_exit(sig,func=None):
    if conexion:
        # conexion.close() log.debug('cerrando')
         ser.close()
    sys.exit(1)
```

```
#Este mensaje se envia cuando el OS solicita cerrar el programa
signal.signal(signal.SIGTERM,on_exit)
#Este mensaje se envia cuando el usuario envia ctrl+c
signal.signal(signal.SIGINT,on_exit)
# Variable para saber si hemos encontrado el puerto o no
bEncontrado = False
# Hacemos un bulce para recorrer los puertos que queremos
comprobar
for iPuerto in range(0, 4):
    try:
        # Puerto que vamos a probar
        PUERTO = '/dev/ttyACM' + str(iPuerto)
        # Velocidad
        VELOCIDAD = '9600'
        # Probamos ha abrir el puerto
        Arduino = serial.Serial(PUERTO, VELOCIDAD)
        # si no se ha producido un error, cerramos el puerto
        Arduino.close()
        # cambiamos el estado del la variable para saber si lo
hemos
        # encontrado
        bEncontrado = True
        # Salimos del bucle
        break
    except:
        # Si hay error, no hacemos nada y continuamos con la
busqueda
        pass

# Si encontramos el puerto?
if bEncontrado:
    # Mostramos el puerto donde esta el arduino
    print('el puerto del arduino es: ' + '/dev/ttyACM' +
str(iPuerto))
    ser = serial.Serial(
    port='/dev/ttyACM' + str(iPuerto),
    baudrate=9600)
    #ser.open()
    ser.isOpen()
    while 1:

        conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
```

```
n") # Se crea la conexion con la Base de Datos usando pass,
user, y la base de Datos
     cursor = conexion.cursor() # se crea el cursor para
generar un dump
     sql2=("SELECT comando FROM tbl_Comando WHERE comando='%s
'" %(Servicio)) # se consulta la tabla de la Activacion de la
Alarma
     cursor.execute(sql2)
     respuesta2 =cursor.fetchone()
     rowsaffected = cursor.rowcount # Se obtiene el Numero de
Columnas Afectadas por la Consulta
     conexion.commit()
        conexion.close()
     if rowsaffected ==1: # Error si la Tabla esta vacia o la
Consulta devuelve null error por que no podas bajar de un
arreglo o tupla un valor si es nulo
            try:
                ser.write(input +'\r\n')
            except serial.SerialException:
                print "muerto el puerto"
                tester =arduino_verificar()
                #if tester is None:
                    #print "La arduino fue desconectada Sistema
"
                    #sys.exit(1)
            salida=''
            time.sleep(1)
         print "%s" %(Servicio)
         estado='on'
            time.sleep(2)
            try:

             while ser.inWaiting()>0:
                 salida += ser.read(1)
         except IOError:
                arduino_verificar()
         if salida != '':
                telefono=88888900
                 shortcode=9899
                texto1 = salida
                envio=send2smsbox('tester','foobar')
                envio.mclass=1
                envio.priority=0
```

```
              conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
                cursor = conexion.cursor()
             sql3 = ("TRUNCATE TABLE tbl_Comando")
             cursor.execute(sql3)
          conexion.commit()
             conexion.close
                respuesta=envio.send(telefono,shortcode,texto1)
          #ser.flushOutput()
                if respuesta is not None:
                    print "Mensaje Enviado"
          print ">>"+ salida

        conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
     cursor =conexion.cursor()
     sql4 = ("SELECT comando FROM tbl_Comando WHERE comando='%s
'" %(off))
        cursor.execute(sql4)
     respuesta3 = cursor.fetchone()
     rowsaffected2 = cursor.rowcount
     conexion.commit()
     conexion.close()

        if rowsaffected2 ==1:
            try:
                ser.write(output +'\r\n')
         except serial.SerialException:
                print "Se perdio el puerto"
                tester2 = arduino_verificar()
            out2 = ''
         time.sleep(1)
         #ser.write(output +'\r\n')
            while ser.inWaiting()>0:
             out2 = ser.read(1)
            print ""+ out2
        #if out != '' :
            print rowsaffected2
            telefono=88888900
         shortcode=9899
            texto="Sistema Alarma The Fallen Desactivado"
```

```
            texto2 = out2
            envio=send2smsbox('tester','foobar')
            envio.mclass=1
            envio.priority=0
          conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
            cursor = conexion.cursor()
          sql5 = ("TRUNCATE TABLE tbl_Comando")
          cursor.execute(sql5)
          conexion.commit()
          conexion.close
            respuesta2=envio.send(telefono,shortcode,texto)
          print "Por aqui vamos"
            if respuesta2 is not None:
                print "Mensaje Enviado, Sistema Desactivado"

        try:
            ser.write(energia +'\r\n')
        except serial.SerialException:
            print "Se perdio el puerto"
            tester3 = arduino_verificar()
        time.sleep(1)
         out3 =''
        try:
            while ser.inWaiting()>0:
             out3 += ser.read(1)
        except IOError:
            arduino_verificar()
        if out3 != '':
         texto3= out3
            conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
            supply = envios(conexion,texto3)
          if supply is not None:
                print "Mensaje Enviado"

        try:
            ser.write(status +'\r\n')
            #pdb.set_trace()
        except serial.SerialException:
            print "Se perdio el puerto"
```

```
            #tester8 = arduino_verificar()
        time.sleep(1)
        #tester4=poder() out5 =''
        try:
         out5=''
            while ser.inWaiting()>0:
                out5 += ser.read(1)
        except IOError:
            arduino_verificar()
        #out5='' pdb.set_trace()
        if out5 != '':
            texto4= out5
            print texto4
            conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
            marvin = envios(conexion,texto4)
            texto5 = "Alerta movimiento detectado!!!!!!!!!!"
            if marvin is not None:
                print "Mensaje Enviado por Activacion de los
Sensores alguien se encuentra aqui"
        else:
            print "La cadena esta vacia!"

        conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n") # Se crea la conexion con la Base de Datos usando pass,
user, y la base de Datos
     cursor = conexion.cursor() # se crea el cursor para
generar un dump
     #time.sleep(2)
     sql_luces=("SELECT * FROM tbl_luces WHERE estado='%s '"
%(luces)) # se consulta la tabla de la Activacion de la Alarma
     cursor.execute(sql_luces)
     respuesta_luces =cursor.fetchone()
     rowsaffected = cursor.rowcount # Se obtiene el Numero de
Columnas Afectadas por la Consulta
     #print "%s" %(rowsaffected) pdb.set_trace()
     conexion.commit()
        conexion.close()
     if rowsaffected ==1: # Error si la Tabla esta vacia o la
Consulta devuelve null error por que no podas bajar de un
arreglo o tupla un valor si es nulo
```

```
            try:
                ser.write(luces_on +'\r\n')
            except serial.SerialException:
                print "muerto el puerto"
                tester =arduino_verificar()
                #if tester is None:
                    #print "La arduino fue desconectada Sistema
"
                    #sys.exit(1)
            salida_luz=''
            time.sleep(1)
         print "%s" %(luces_on)
        # print "%s" %(respuesta2[0])
         #ser.write(output +'\r\n')
         estado='on'
            time.sleep(2)
            try:

             while ser.inWaiting()>0:
                 salida_luz += ser.read(1)
         except IOError:
                arduino_verificar()
         if salida_luz != '':
               texto12 = salida_luz
                envio=send2smsbox('tester','foobar')
                envio.mclass=1
                envio.priority=0
              conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
                cursor = conexion.cursor()
             sql_off = ("TRUNCATE TABLE tbl_luces")
             cursor.execute(sql_off)
          conexion.commit()
             conexion.close

respuesta=envio.send(telefono,shortcode,texto12)
          #ser.flushOutput()
                if respuesta is not None:
                    print "Mensaje Enviado"
          print ">>"+ salida_luz
#print "%s" %(respuesta)
else:
```

```
    conexion =
MySQLdb.connect("localhost","root","raspberry","KannelProduccio
n")
    texto6 = "El Dispositivo no se encuentra conectado por
favor Verifique, final de la Ejecucion !!!!"
    no_device = envios(conexion,texto6)
    if no_device is not None:
        print "Mensaje Enviado por que no se encuentra el
Dispositivo"
    print("El dispositivo no se Encuentra conectado Por Favor
Verifique")
exit()
CODIGO FUENTE SMS_NOTIFICATION.PY
import sys,time,signal,pdb
import serial
import MySQLdb
from send_sms import send2smsbox
# -*- coding: UTF8 -*-

def envios(conexion,texto):
    conexion = conexion
    texto = texto
    shortcode = 9897
    cursor = conexion.cursor()
    sql = "SELECT IdCliente from tblClienteServicio where
IdServicio =4 and ynEstatus=1"
    cursor.execute(sql)
    response = cursor.fetchall()
    if response is not None :
        sms=send2smsbox('tester','foobar')
        sms.mclass=1
        sms.priority=0
        #texto= "Prueba de Envio"
        for registro in response:
            enviar=sms.send(registro[0],shortcode,texto)
            print registro[0]
            #break
            if enviar is not None:
                print "Mensaje Enviado"
        conexion.commit()
        conexion.close()
```

CODIGO FUENTE ALTA AL USUARIO

```
<?php
    require_once ('Funciones.inc');
    $phone=$_GET["phone"];
    $texto= $_GET["text"];



  //Verifica Conexion
  $conexion=mysql_connect("localhost","root","raspberry")
  or die("Problemas en la conexion");
  mysql_select_db("KannelProduccion",$conexion) or
  die("Problemas en la seleccion de la base de datos");

  $query = mysql_query("SELECT IdCliente FROM tblCliente WHERE
IdCliente LIKE '$phone'");
   if (!mysql_num_rows($query))
           {
                   $query1 = mysql_query("Insert INTO
tblCliente(IdCliente) values ('$phone')") or die ("Error!");
           }

               if (strstr($texto,' ')){
                   $Servicio = trim(strstr($texto,' '));
                   $queryServ = mysql_query("SELECT IdServicio
FROM tblServicio WHERE strDescripcion LIKE '$Servicio'");
                   if(mysql_num_rows($queryServ)){
                       $fila=mysql_fetch_array($queryServ);
                   $IdServicio = $fila['IdServicio'];

                        $queryClientServ = mysql_query("SELECT
* FROM tblClienteServicio WHERE IdCliente LIKE '$phone' and
IdServicio = '$IdServicio'");
                       if(!mysql_num_rows($queryClientServ)){
                         $query1 = mysql_query("Insert INTO
tblClienteServicio(IdCliente,IdServicio,dtFechaAlta,ynEstatus)
values ('$phone',$IdServicio,now(),1)") or die ("Error!");
                         $MensajeActivar= "Servicio
'$Servicio' activado Correctamente";
                         EnviarSMS($phone,$MensajeActivar);
                               $phonepass=
service_alarma($phone,$IdServicio,$conexion);
                             if ($phonepass==true){
```

```
     $querypasswd=mysql_query("Insert INTO
usuarios(usuario,password) values ('$phone','$phonepass')") or
die("Error en la Insercion del Password Warning");
                                    echo $phonepass;
                                    $MensajePassword="Este es su
password para gestionar el Sistema de Alarma por medio de
Internet: $phonepass";

     EnviarSMS($phone,$MensajePassword);
                                    }
                                 else{
                                        $aviso_pass= "Usted ya
tiene un password para este Servicio $Servicio";

EnviarSMS($phone,$aviso_pass);
                                    }
                              $Mensaje =
RandomMensaje($phone,$IdServicio);
                              EnviarSMS($phone,$Mensaje);
                            }
                            else{
$queryClientServUpdate = mysql_query("SELECT * FROM
tblClienteServicio WHERE IdCliente LIKE '$phone' and IdServicio
= '$IdServicio' and ynEstatus='1'");

                              if
(mysql_num_rows($queryClientServUpdate)){
                              $MensajeActivar= "Usted ya tiene
activo el servicio '$Servicio'";
                              EnviarSMS($phone,$MensajeActivar);
                              }
                              else{
                              $query1 = mysql_query("Update
tblClienteServicio set dtFechaAlta = now(), ynEstatus=1 where
IdCliente LIKE '$phone' and IdServicio = '$IdServicio'") or die
("Error!");
                              $MensajeActivar= "Servicio
'$Servicio' Reactivado";
                              EnviarSMS($phone,$MensajeActivar);
                              }
```

```
                        }

                    }
                    else{
                     $MensajeActivar= "No existe el servicio";
                     EnviarSMS($phone,$MensajeActivar);
                    }

                }
                else{
                    $MensajeActivar= "No especifico el
Servicio";
                    EnviarSMS($phone,$MensajeActivar);
                }



?>
```

CODIGO FUENTE BAJA AL USUARIO

```
<?php
    require_once ('Funciones.inc');

    $phone=$_GET["phone"];
    $texto= $_GET["text"];



  //Verifica Conexion
  $conexion=mysql_connect("localhost","root","raspberry")
  or die("Problemas en la conexion");
  mysql_select_db("KannelProduccion",$conexion) or
  die("Problemas en la seleccion de la base de datos");

  $query = mysql_query("SELECT IdCliente FROM tblCliente WHERE
IdCliente LIKE '$phone'");
   if (!mysql_num_rows($query))
            {
             $MensajeBaja= "Su Numero no se encuentra
registrado";
             EnviarSMS($phone,$MensajeBaja);
            }
```

```
   else{
                if (strstr($texto,' ')){
                    $Servicio = trim(strstr($texto,' '));
                    $queryServ = mysql_query("SELECT IdServicio
FROM tblServicio WHERE strDescripcion LIKE '$Servicio'");
                    if(mysql_num_rows($queryServ)){
                        $fila=mysql_fetch_array($queryServ);
                    $IdServicio = $fila['IdServicio'];
                         $queryClientServ = mysql_query("SELECT
* FROM tblClienteServicio WHERE IdCliente LIKE '$phone' and
IdServicio = '$IdServicio'");
                        if(!mysql_num_rows($queryClientServ)){
                                $MensajeBaja= "Usted no tiene el
Servicio '$Servicio' activado";
                                EnviarSMS($phone,$MensajeBaja);
                        }
                        else{
                            $queryClientServUpdate =
mysql_query("SELECT * FROM tblClienteServicio WHERE IdCliente
LIKE '$phone' and IdServicio = '$IdServicio' and
ynEstatus='0'");
                            if
(mysql_num_rows($queryClientServUpdate)){
                            $MensajeBaja= "Usted ya tiene
Desactivado el servicio '$Servicio'";
                            EnviarSMS($phone,$MensajeBaja);
                            }
                            else{
                            $query1 = mysql_query("Update
tblClienteServicio set dtFechaBaja = now(), ynEstatus=0 where
IdCliente LIKE '$phone' and IdServicio = '$IdServicio'") or die
("Error!");
                            $MensajeBaja="El Servicio
'$Servicio' ha Sido Desactivado Exitosamente";
                            EnviarSMS($phone,$MensajeBaja);
                            }


                        }

                    }
                    else{
                     $MensajeBaja= "No existe el servicio";
```

```
                    EnviarSMS($phone,$MensajeBaja);
                   }

               }
               else{
                   $MensajeBaja= "No especifico el Servicio";
                   EnviarSMS($phone,$MensajeBaja);
               }

    }

?>
```

CODIGO FUENTE ACTIVAR ALARMA

```
<?php
  require_once ('Funciones.inc');
  require("php_serial.class.php");

  $phone=$_GET["phone"];
 $texto=$_GET["text"];

//Verificar conexion con la base de datos
$conexion=mysql_connect("localhost","root","raspberry")or
die("Problemas en la Conexion con MySQl");
  mysql_select_db("KannelProduccion",$conexion) or
die("Problemas en la Seleccion de la Base de Datos");
    $query = mysql_query("SELECT IdCliente FROM tblCliente
WHERE IdCliente LIKE '$phone'");
        if (!mysql_num_rows($query))
            {
                $aviso= "Usted no tiene acceso a este servicio
";
                EnviarSMS ($phone,$aviso);
            }
            if (strstr($texto,' ')){
                $Servicio = trim(strstr($texto,' '));
                    $queryServ = mysql_query("SELECT IdServicio
FROM tblServicio WHERE strDescripcion LIKE '$Servicio'");
                    if(mysql_num_rows($queryServ)){
                        $fila=mysql_fetch_array($queryServ);
                    $IdServicio = $fila['IdServicio'];
```

```
                    $queryClientServ = mysql_query("SELECT *
FROM tblClienteServicio WHERE IdCliente LIKE '$phone' and
IdServicio = '$IdServicio'");
                        if(mysql_num_rows($queryClientServ))
                            {

                                $serial = new phpSerial();



                                //Puerto Serial

                                $serial-
>deviceSet("/dev/ttyACM0/");



                                //Parametros de Configuracion 9600
8-N-1, so

                                $serial->confBaudRate(9600);

                                $serial->confParity("none");

                                $serial->confCharacterLength(8);

                                $serial->confStopBits(1);

                               $serial->confFlowControl("none");


                                    $serial->deviceOpen();
                                        $serial->sendMessage("H");


                                        $serial->deviceClose();
```

```
                            $MensajeActivar= "Servicio
'$Servicio' activado Correctamente";
                             EnviarSMS($phone,$MensajeActivar);


                          }

                       }


                    }
?>
```

CODIGO FUENTE DESACTIVAR ALARMA

```
<?php
  require_once ('Funciones.inc');
  require("php_serial.class.php");

  $phone=$_GET["phone"];
 $texto=$_GET["text"];

//Verificar conexion con la base de datos
$conexion=mysql_connect("localhost","root","raspberry")or
die("Problemas en la Conexion con MySQl");
  mysql_select_db("KannelProduccion",$conexion) or
die("Problemas en la Seleccion de la Base de Datos");
    $query = mysql_query("SELECT IdCliente FROM tblCliente
WHERE IdCliente LIKE '$phone'");
        if (!mysql_num_rows($query))
            {
                $aviso= "Usted no tiene acceso a este servicio
";
                EnviarSMS ($phone,$aviso);
            }
            if (strstr($texto,' ')){
                $Servicio = trim(strstr($texto,' '));
                    $queryServ = mysql_query("SELECT IdServicio
FROM tblServicio WHERE strDescripcion LIKE '$Servicio'");
                    if(mysql_num_rows($queryServ)){
                        $fila=mysql_fetch_array($queryServ);
                    $IdServicio = $fila['IdServicio'];
```

```
                    $queryClientServ = mysql_query("SELECT *
FROM tblClienteServicio WHERE IdCliente LIKE '$phone' and
IdServicio = '$IdServicio'");
                        if(mysql_num_rows($queryClientServ))
                            {

                                $serial = new phpSerial();



                                //Puerto Serial

                                $serial-
>deviceSet("/dev/ttyACM0/");



                                //Parametros de Configuracion 9600
8-N-1, so

                                $serial->confBaudRate(9600);

                                $serial->confParity("none");

                                $serial->confCharacterLength(8);

                                $serial->confStopBits(1);

                               $serial->confFlowControl("none");


                                    $serial->deviceOpen();
                                        $serial->sendMessage("L");


                                        $serial->deviceClose();
```

```
                            $MensajeActivar= "Servicio
'$Servicio' activado Correctamente";
                             EnviarSMS($phone,$MensajeActivar);


                          }

                       }


                   }
?>
```

CODIGO FUENTE ACTIVAR Y DESACTIVAR LUMINARIA

```
<?php
  require_once ('Funciones.inc');
  //require("php_serial.class.php");

   $phone=$_GET["phone"];
   $texto=$_GET["text"];
    $comando="on";
    $comando2="off";

//Verificar conexion con la base de datos
  $conexion=mysql_connect("localhost","root","raspberry")or
die("Problemas en la Conexion con MySQl");
    mysql_select_db("KannelProduccion",$conexion) or
die("Problemas en la Seleccion de la Base de Datos");
    $queryoff = mysql_query("SELECT IdCliente FROM tblCliente
WHERE IdCliente LIKE '$phone'");
     if(!mysql_num_rows($queryoff)==1)
     {
       $prohibido= "Usted no tiene acceso a este servicio";
       echo $prohibido;
       EnviarSMS($phone,$prohibido);
     }


     else
     {
         strstr($texto,' ');
        $orden = trim(strstr($texto,' '));
```

```
        //$orden = explode(" ",$texto);
        echo("$orden");

         if($orden===$comando)
         { //echo("$orden");
           $luces_query=mysql_query("SELECT * from tbl_luces");
           if (!mysql_num_rows($luces_query))
             {
                $query_insert=mysql_query("INSERT INTO
tbl_luces (estado) values('$orden')");
                $aviso=("Las luces han sido activadas");
                EnviarSMS($phone,$aviso);

             }
           else{
                  $query_insert2=mysql_query("UPDATE tbl_luces
SET estado='$orden'");
                  $aviso2=("Las luces han sido activadas");
                  EnviarSMS($phone,$aviso2);




               }

         }



     }

      if($orden===$comando2){
           $luces_query=mysql_query("SELECT * from tbl_luces");
           if (!mysql_num_rows($luces_query))
             {
                $query_insertoff=mysql_query("INSERT INTO
tbl_luces (estado) values('$orden')");
                $aviso3="Las luces han sido activadas";
                EnviarSMS($phone,$aviso3);

             }
          else{
```

```
                    $query_insertoff2=mysql_query("UPDATE
tbl_luces SET estado='$orden'");
                    $aviso4=("EL estado de las luces de la casa
es '$orden'");
                    EnviarSMS($phone,$aviso4);




                }




            }
     ?>
```

CODIGO FUENTE Funciones.inc

```
      <?php

function EnviarSMS ($IdCliente,$strMensaje)
{
    $hostname="localhost:13013" ;
    $rutaCGI= "/cgi-bin/sendsms";
    $Usuario="tester";
    $Pass="foobar";
    $from="88888900"   ;
    $strUrl = "http://"
.$hostname.$rutaCGI."?username=".$Usuario."&password=".$Pass."&
to=".urlencode($IdCliente)."&text=".urlencode($strMensaje)."&fr
om=".$from;
    $curl_handle = curl_init ($strUrl);

        //curl_setopt ($curl_handle, CURLOPT_URL,$urllink);

        curl_setopt($curl_handle, CURLOPT_HEADER, 0);
        //curl_setopt($curl_handle, CURLOPT_POST, 1);
        curl_setopt($curl_handle, CURLOPT_RETURNTRANSFER, 1);
        //curl_setopt ($curl_handle, CURLOPT_POSTFIELDS,
$data_string);
```

```
        //    Perform the GET and get the data returned by the
        //    server.
        $result = curl_exec ($curl_handle) or die ('There has
been an error');

        //    Close the CURL handle
        curl_close ($curl_handle);

}

function generaPass(){
    //Se define una cadena de caractares. Te recomiendo que
uses esta.
    $cadena = "1234567890";
    //Obtenemos la longitud de la cadena de caracteres
    $longitudCadena=strlen($cadena);

    //Se define la variable que va a contener la contraseña
    $pass = "";
    //Se define la longitud de la contraseña, en mi caso 10,
pero puedes poner la longitud que quieras
    $longitudPass=4;

    //Creamos la contraseña
    for($i=1 ; $i<=$longitudPass ; $i++){
        //Definimos numero aleatorio entre 0 y la longitud de
la cadena de caracteres-1
        $pos=rand(0,$longitudCadena-1);

        //Vamos formando la contraseña en cada iteraccion del
bucle, añadiendo a la cadena $pass la letra correspondiente a
la posicion $pos en la cadena de caracteres definida.
        $pass .= substr($cadena,$pos,1);
    }
    return $pass;
}
?>
```

Anexo 3: Ficha Técnica de Componentes.