UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA: INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del Título de:

INGENIERO ELECTRÓNICO

TEMA:

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD DE CONTROL LOCAL Y REMOTO CON DISPOSITIVOS DE VIGILANCIA,

DESARROLLADO EN EL SOFTWARE PYTHON, CENTRALIZÁNDOSE EN UNA TARJETA RASPBERRY PI.

AUTORES:

ANDY CAMPUZANO BULGARÍN

VANESSA CEDEÑO VÉLEZ

DIRECTOR:

ING. BREMNEN VÉLIZ

GUAYAQUIL, ABRIL DEL 2015

DECLARATORIA DE RESPONSABILIDAD

Los conceptos desarrollados, análisis realizados y conclusiones del presente trabajo son de exclusiva responsabilidad de los autores. A través de la presente declaración, se da el derecho de propiedad intelectual correspondiente a la Universidad Politécnica Salesiana.

Guayaquil, Abril del 2015

(f) _____ (f) _____

Andy Campuzano Bulgarín **Vanessa Cedeño Vélez**

DEDICATORIA

En primer lugar dedico el siguiente trabajo a Dios, por guiarme y haberme permitido llegar hasta aquí, por ser mi fortaleza en los momentos más difíciles, ya que sin el nada de esto fuera posible.

A mis padres, Guber Cedeño Zambrano y Alexandra Vélez Saltos que han sido mi motivación durante todo el proceso de mi carrera profesional. El pilar fundamental en mi vida, donde me han inculcado valores muy importantes que han hecho de mí una mujer de bien en los cuales me ha ayudado a salir adelante y poder obtener mi título de Ingeniera.

A mi hermana Lissette Cedeño Vélez quien siempre ha estado apoyándome en todo momento.

A mi novio Jorge Peñafiel Zapata por ser mi amigo y compañero en todo el transcurso de mi etapa universitaria. Por ser la persona quien me ha apoyado y comprendido en los momentos más difíciles. Por no dejarme vencer y seguir luchando hasta alcanzar la meta. Por su amor incondicional que ha sido muy importante para lograr todo esto.

Vanessa Cedeño Vélez.

DEDICATORIA

Siendo el guía, camino y fuerza, dedico mi trabajo primeramente a Dios por ser incondicional en todo momento hasta en los más difíciles en el transcurso de mi vida.

Luego como es normal a mi familia conformada por mi padre Evaristo Octavio Campuzano Pereira, mi madre Jully Grace Bulgarín Navarrete, mis tres hermanos Yessy Francois Campuzano Bulgarín, Kenyo Roberto Campuzano Bulgarín y Randy Jardel Campuzano Bulgarín, siendo los promotores de la iniciativa que tuve para empezar mi carrera universitaria

Junto a una mujer tan grandiosa como mi madre, como lo es mi tía Tanya Antonieta Bulgarín Navarrete que fue la voz alentadora para seguir adelante en mis caídos de ánimo.

Y como dejar de nombrar a mí amada esposa Connie Alexandra Noriega Ladd y a mí inspirador hijo Bruce Benjamin Campuzano Noriega. Cuyas fuentes de inspiración y fuerza son inimaginablemente grandes, apoyándome en todo lo que estaba a su alcance e inspirándome en todo sólo viéndolos

A todos ellos les dedico mi gran logro sabiendo y diciendo "Que por ellos los hago para estar donde estoy y ser lo que soy"

Andy Campuzano Bulgarín.

AGRADECIMIENTO

Agradezco a mi padre celestial, Dios, por darme la oportunidad de vivir este momento muy especial en mi vida y en la de mi familia. Por llenarme de fuerza y saberme guiar en cada paso que doy. Por no dejarme caer y ser mi luz en mi camino.

A mi papá Guber Cedeño Zambrano y a mi mamá Alexandra Vélez Saltos por ser mi ejemplo a seguir y por estar apoyándome en todo momento. Por su paciencia y amor incondicional que me impulsa salir adelante y lograr alcanzar todas mis metas.

A mis profesores y a mi tutor el Ing. Bremnen Veliz por su apoyo, para lograr finalizar mí trabajo de titulación. Así mismo al Ing. Víctor Huilcapi por todo el aporte brindado durante mi formación profesional.

Vanessa Cedeño Vélez.

AGRADECIMIENTO

A mi gran guía y fuente de fuerza como lo es Dios por ser incondicional en los más difíciles momentos de mi vida.

Agradezco a cada uno de los que conforman mi familia resaltando a mi padre Evaristo Octavio Campuzano Pereira, a mi madre Jully Grace Bulgarín Navarrete, a mis tres hermanos Yessy Francois Campuzano Bulgarín, Kenyo Roberto Campuzano Bulgarín y Randy Jardel Campuzano Bulgarín, y mi grandiosa tía Tanya Antonieta Noemi Bulgarín Navarrete promotores de la iniciativa que tuve para empezar mi carrera universitaria

Y un agradecimiento muy especial a mi esposa Connie Alexandra Noriega Ladd y a mí inspirador hijo Bruce Benjamin Campuzano Noriega. Por su inspiración y fuerza día a día, apoyándome en todo.

A todos los profesores con dignidad y calidad de enseñanza que compartieron conmigo sus conocimientos obteniendo el logro y llegando a la meta de todo estudiante universitario.

A mi tutor el Ing. Bremnen Veliz por su apoyo, para lograr finalizar mí trabajo de titulación.

Así mismo y sin menos importancia al Ing. Víctor Huilcapi por todo el la ayuda brindada durante el transcurso de mi carrera universitaria.

A todos ellos les agradezco infinitamente por permitirme llegar a la gran meta sabiendo y diciendo "Estoy donde estoy y soy lo que soy"

Andy Campuzano Bulgarín.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

AÑO	ALUMNO/S	DIRECTOR DE TESIS	TEMA DE TESIS
2015	CAMPUZANO BULGARIN ANDY JAIR CEDEÑO VELEZ VANESSA ALEXANDRA	ING. BREMNEN VELIZ	"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD, DE CONTROL LOCAL Y REMOTO, CON DISPOSITIVOS DE VIGILANCIA, DESARROLLADO EN EL SOFTWARE PYTHON, CENTRALIZÁNDOSE EN UNA TARJETA RASPBERRY PI."

El trabajo de investigación realizado está basado en el diseño e implementación de un sistema de seguridad donde se concentra gran parte en la tarjeta Raspberry Pi, permitiendo el control de las cámaras y luces. También el sistema está conformado por un dispositivo de acceso biométrico.

El primer control se basa en la ubicación y manejo de cámaras que permite coberturas de ángulos fijos y regulables, dando prioridad a las zonas de acceso hacia el interior de la propiedad, donde cubre en su totalidad las áreas con mayor posibilidad de movimientos.

El control de luces se realizó con un módulo de relés que son controlados por los pines GPIO de la tarjeta Raspberry Pi, bajo un programa con interface en QT Creator y archivos de procesos desarrollados en Python el cual permite al usuario supervisar y tener mayor control de la casa .

La conexión puede manejarse de manera local y remota por medio del Escritorio Remoto hacia la Raspberry Pi, por medio del software XRDP. También se realiza accediendo al ordenador desde el móvil por medio de la aplicación de Escritorio Remoto Google Chrome.

El dispositivo de acceso biométrico contiene visualización de menú, ingreso de datos por teclado y escaneo de huella digital.

Los autores de la investigación eligen un lugar específico para la posición de los dispositivos pertenecientes al sistema optimizando un control global del esquema. El presente trabajo establece que se eleva el nivel de seguridad en propiedades limitadas de terreno.

ABSTRACT

YEAR	AUTHORS	ADVISOR	TITLE
2015	CAMPUZANO BULGARIN ANDY JAIR CEDEÑO VELEZ VANESSA ALEXANDRA	ING. BREMNEN VELIZ	"DESIGN AND IMPLEMENTATION OF A SECURITY SYSTEM , LOCAL CONTROL AND REMOTE MONITORING DEVICES , SOFTWARE DEVELOPED IN PYTHON , CENTERING ON A CARD RASPBERRY PI ."

The research work is based on the design and implementation of a security system which largely focuses on the Raspberry Pi card, allowing the control of the cameras and lights. Also the system is comprised of a biometric access device.

The first control has been based on the location and operation of cameras that allows coverage of fixed and adjustable angles, giving priority to areas of access into the property, which completely covers the areas most likely to move.

The light control is performed with a relay module that are controlled by GPIO pins of the Raspberry Pi card, under a program interface in QT Creator files and processes developed in Python which allows the user to monitor and take greater control of the house.

The connection can be managed locally and remotely via Remote Desktop to the Raspberry Pi, by software XRPD. It is also done by accessing the computer from the mobile application through Google Chrome Remote Desktop.

The biometric access device contains menu display, data entry keyboard and fingerprint scan.

The authors of the research choose a specific location for the position of the devices in the system optimizing overall control scheme. This work establishes that the security level rises in limited field properties.

INTRODUCCIÓN

Un grave problema que aqueja a la sociedad es la inseguridad. Siempre se debe andar con cautela en lugares públicos e incluso en nuestros propios hogares debemos de tomar precaución contra los enemigos de lo ajeno.

En la ciudad de Guayaquil en los últimos años se han registrados 9 robos a vivienda. Por otro lado, en el mercado existen sistemas de seguridad que son costosos y difíciles de usar, por ejemplo un sistema de seguridad con cámaras, sensores de movimientos y parte de domótica cuesta más de $2000.

Por tal motivo, en este proyecto de graduación, se ha decidido crear un prototipo de sistema de seguridad para viviendas que sea eficiente, versátil, y amigable con el usuario, además de tener un control de luces y un monitoreo remoto.

El trabajo de este sistema está enfocado en convertirse en una herramienta muy importante ya que esto ayuda estar en constante monitoreo con nuestras casas aunque estemos en cualquier lugar del mundo.

El objetivo principal del proyecto es mejorar tecnológicamente la seguridad de las personas que habitan en propiedades delimitadas de terreno y a su vez tener una perspectiva de lo que se puede lograr con este tipo de dispositivos como es una minicomputadora Raspberry Pi.

El funcionamiento del sistema se basa en las cámaras, control de luces y el acceso biométrico. Con la minicomputadora Raspberry pi, de manera remota con el software Xrdp logran un mayor control e interactividad con el usuario, ya que desde el teléfono móvil se podrá observar cualquier movimiento dentro del hogar.

Para desarrollar se implementó dispositivos electrónicos de vigilancia; controlados por una interfaz con los programas Python y Qt, dando así mayor eficiencia al sistema.

En el capítulo 1 se detalla puntos primordiales que tiene todo lo relacionado al planteamiento del problema, delimitación, objetivos, metodología, beneficiarios del proyecto y el impacto que tiene el sistema con el usuario.

En el capítulo 2 se realiza el marco teórico donde sobresalen los temas más importantes de la investigación, enfocándose en los elementos aplicados en el transcurso del proyecto.

En el capítulo 3 se describe la manera técnica y desarrollo del proyecto, enfocándose en las características, estructuras, análisis y estudios reales, junto a otros detalles de indagaciones realizadas a sistemas similares.

En el capítulo 4 se analiza minuciosamente los resultados de las pruebas experimentales enfocadas en diferentes etapas que posee el proyecto, tomando en cuentas las dificultades y los logros obtenidos.

CAPÍTULO I

1 EL PROBLEMA

1.1. Planteamiento del problema

En la provincia del Guayas, en la zona Sur del Cantón Guayaquil, ciudadela Los Esteros está ubicada una casa cuyos propietarios continuamente están ausentes, en la misma se encontró un alto nivel de inseguridad, y se encuentra en un callejón de escaso tránsito peatonal. Entre las causas se tiene, que en la zona existen muchos reportes y notificaciones de asaltos a casas (Telégrafo, 2015).

Con este antecedente, los autores analizaron el hecho de que familiares de los dueños realizan una revisión del estado del lugar, determinando este método poco práctico.

Este proyecto se lo realiza en condiciones de menor magnitud, pero a niveles macro o de varias viviendas tendrá un igual desempeño funcional.

1.2. Delimitación del problema

El proyecto se desarrolló en un establecimiento ubicado en la provincia del Guayas, en la zona sur del cantón Guayaquil, ciudadela Los Esteros en el año 2014 y 2015 en el que se realizaron varias pruebas de todos los dispositivos que lo conforman.

El proyecto cuenta con una tarjeta de Raspberry Pi –B el cual trabaja con un sistema operativo basado en el núcleo Linux (software libre), como es raspbian. (V. enero 2014).

Se instaló 3 cámaras VGA (video) donde cuenta con una resolución de 5MP (cámara) con una interfaz USB 2.0, además de esto consta con el soporte de Python 2.2.7, el software QT 4.2 y XRDP.
En conjunto con los demás elementos como el dispositivo de acceso biométrico el cual está compuestos por una pantalla LCD 16*2, módulo lector de huellas rs232,

teclado matricial y un microcontrolador PIC16f887 de Microchip; el dispositivo de control de luces controlado por los PINES GPIO de la Raspberry pi; obteniendo un sistema de control local por una interfaz gráfica y remoto por el Software XRDP.

Demás componentes, no serán objeto de estudio, queda fuera del alcance de la investigación.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar e Implementar un sistema de seguridad de una vivienda, mediante el uso de dispositivos electrónicos de vigilancia; controlado por una interfaz de fácil interactividad desarrollado con los programas Python y Qt, para ayudar a mejorar la seguridad en la residencia.

1.3.2. Objetivos específicos

- Analizar los diferentes tipos de sistema de seguridad implementados en la actualidad y los elementos que componen los mismos.
- Construir un sistema de seguridad, destinado a la vigilancia y control de una propiedad.
- Diseñar e implementar un Dispositivo de Acceso Biométrico con visualización de menú e ingreso de datos por teclado, manejado por un Microcontrolador PIC16F887 como receptor-procesador de señales.
- Desarrollar una interfaz en el software QT 4.2 en conjunto con PYTHON 2.7.7 con visualización y manejo por conexión remota.
- Controlar los elementos del sistema: Cámaras de 5 MP, dispositivos de control de luces y un dispositivo de acceso biométrico localmente por los PINES GPIO de la RASPBERRY PI y remotamente por el Software XRDP.

1.4. Justificación

Como estudiantes de la Universidad Politécnica Salesiana – sede Guayaquil se ha propuesto desarrollar este proyecto debido a la necesidad de integrar nuevas tecnologías con fácil interactividad para la vigilancia de una propiedad, ubicada en la provincia del Guayas, en la zona Sur del Cantón Guayaquil, ciudadela "Los Esteros".

Los autores consideraron la implementación de un sistema de seguridad, gobernado por control local y remoto, equipado con cámaras de seguridad para la transmisión de video en tiempo real, controles de luces y control de acceso, para aplicar los conocimientos relacionados a la emisión y recepción de datos usando tecnología dotada por los programas PYTHON 2.7.7 y QT 4.2 (pertenecientes Raspbian), aprovechando la interactividad de los mismo.

A su vez se usó y fortaleció los estudios de diseño, ensamblaje y construcción de circuitos electrónicos enfocándose en Dispositivos de Acceso Biométrico.

El campo de aplicación de este sistema de seguridad no solo está en la supervisión de una propiedad, sino aplicable para bodegas, locales, empresas e industrias de distintos volúmenes.

1.5. Hipótesis

El diseño del sistema para envíos de capturas de imagen al teléfono móvil, promoverá paso a la adquisición de una herramienta de mayor eficiencia para lograr informar, indicar y prevenir a las personas de cualquier suceso cuando se encuentren fuera de su domicilio.

Con la correcta implementación permitió un seguimiento actualizado y en tiempo real del entorno y áreas definidas lo mismo que será grabado y respaldado en la detección de movimientos para infinidades de utilidades.

1.6. Variables e Indicadores

1.6.1. Variables

- Nivel de Seguridad
- Consumo de Energía
- Tiempo de activación del sistema
- Luces
- Capacidad de Almacenamiento de imágenes
- Velocidad de Transferencia de imágenes

1.6.2. Indicadores

- Robos del Sector
- Encendido del Sistema
- Utilidad del Sistema
- Módulo de Activación
- Capacidad de Memoria SD de la Raspberry Pi
- Capacidad de Memoria Interna de la Raspberry Pi

1.7. Metodología

1.7.1. Métodos

Para el desarrollo del proyecto se llegó a la conclusión de que sean utilizados los siguientes métodos, para lograr una mayor eficiencia y calidad, de esa forma se obtuvo el resultado esperado:
Método Deductivo

Se procedió a realizar el análisis de los procesos de seguridad que son implementados en muchas áreas actualmente, con lo cual se mejoró lo proyectado para la estructura del sistema.

Método Inductivo

Se desarrolló un sistema con control y supervisión en tiempo real, contando con equipos de acceso, dispositivos de vigilancia, interfaces de manejo local y remoto de variables (luces), centralizado en un ordenador de placa reducida (RASPBERRY PI) junto a Controladores Programables de Interrupciones (PIC).

1.7.2. Técnicas

Las técnicas que se aplicaron en el proyecto fueron:

Técnica documental

Para la recopilación de datos que fueron obtenidos durante el transcurso de los proceso de pruebas en el domicilio, se usó la técnica documental donde se concluyó que toda la base del sistema se encontró en conceptos básicos de programación, redes 3 y microcontroladores.

Técnica de campo

Se utiliza el Raspberry Pi para observar el proceso de captura de las cámaras y desarrollo del sistema.

1.7.3. Instrumentos de Investigación y Recolección de datos

Se hace uso de la tarjeta Raspberry Pi para recolección de capturas y videos en tiempo real y manejo de luces; En conjunto al Dispositivo de Acceso Biométrico para la parte de ingreso hacia el interior de la propiedad.

1.8. Población y Muestra

El proyecto fue destinado para los habitantes de una vivienda. Los investigadores tomaron como muestra la totalidad de la población.

1.9. Descripción de la propuesta

Diseñar e implementar un sistema de vigilancia, que podrá controlar cámaras, un dispositivo de acceso biométrico al lugar, y módulos para el control de luces; de manera remota con el software XRDP, y localmente por una interfaz de fácil interactividad con el usuario desarrollado en los lenguajes PYTHON 2.7.7 y QT 4.2, centralizado en una tarjeta RASPBERRY PI.

Para ayudar a mejorar la seguridad en dicha propiedad delimitada de terreno, los autores eligieron una estructura estratégica en la posición de los dispositivos pertenecientes al mismo.

Por lo cual se empleará como receptor: el Microcontrolador PIC16F887 ubicado en el equipo encargado del acceso al lugar (Dispositivo Biométrico), y los PINES GPIO de la tarjeta RASPBERRY PI para el control de luces y la transmisión de video.

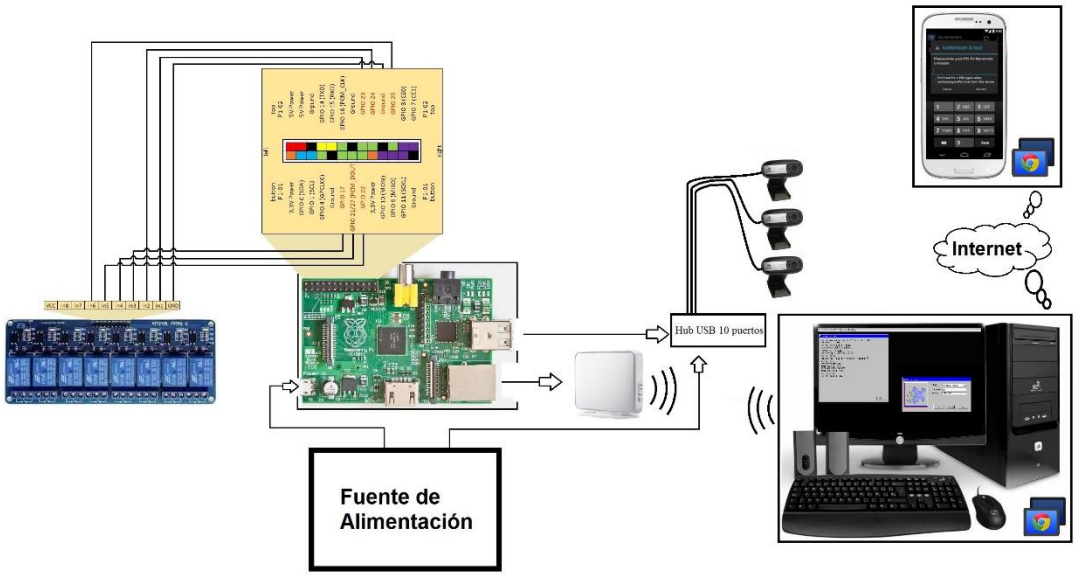


Figura 1.9: Diagrama de Bloques del Sistema

Fuente: Los Autores

1.10. Beneficiarios

Los beneficiarios directos del sistema son los residentes que se encuentran en el domicilio.

1.11. Impacto

Al realizar un sistema de seguridad con los diferentes dispositivos mostrados y al estar interconectados con una microcomputadora como es la Raspberry Pi se incrementara la seguridad debido a la tecnología de punta que se está ofreciendo y otras de las razones es por su fácil manejo e interactividad con el usuario.

El principal obejetivo de este proyecto es concientizar a las personas de lo importante que es la tecnología dentro de un hogar con el fin de dar tranquilidad a la hora de realizar las actividades diarias.

Por otra parte detectar las necesidades reales de formación e información, al momento de promover el aumento de seguridad en la vivienda cuando no este presente ninguna persona. Tener una aceptación positiva a la creación del uso de tecnología como contribución al desarrollo de seguridad de una determinada zona.

Al mismo tiempo lograr un entorno favorable y seguro para los residentes y así eliminar problemas y preocupaciones asociados a la ausencia en las viviendas.

CAPÍTULO II

2 MARCO TEÓRICO

2.1. Raspberry Pi

En su corazón nos encontramos con un chip integrado Broadcom BCM2835, que contiene un procesador ARM11 con varias frecuencias de funcionamiento y la posibilidad de subirla (overclocking) hasta 1 GHz, un procesador gráfico VideoCore IV, y distintas cantidades de memoria RAM.

Las últimas Raspberry Pi cuentan con 512 MB de memoria. Todo ello equivale en la práctica a un ordenador con unas capacidades gráficas similares a la XBOX de Microsoft y con la posibilidad de reproducir vídeo en 1080p.

En la placa nos encontramos además con una salida de vídeo y audio a través de un conector HDMI, con lo que conseguiremos conectar la tarjeta tanto a televisores como a monitores que cuenten con dicha conexión.

En cuanto a vídeo se refiere, también cuenta con una salida de vídeo compuesto y una salida de audio a través de un minijack. Posee una conexión Ethernet 10/100 y, si bien es cierto que podría echarse en falta una conexión Wi-Fi, gracias a los dos puertos USB incluidos podremos suplir dicha carencia con un adaptador Wi-Fi USB de terceros si lo necesitamos.

Los puertos tienen una limitación de corriente, por lo que si queremos conectar discos duros u otro dispositivos se tendrá que hacerlo a través de un Hub USB con alimentación.

En su parte inferior cuenta con un lector de tarjetas SD, lo que da la posibilidad de instalar un sistema operativo en una tarjeta de memoria de 1 GB o más (clase 4 o mejor). De esta forma tenemos también la posibilidad de minimizar el espacio que necesitamos para tener todo un ordenador en un volumen mínimo. (Xataka, 2012)

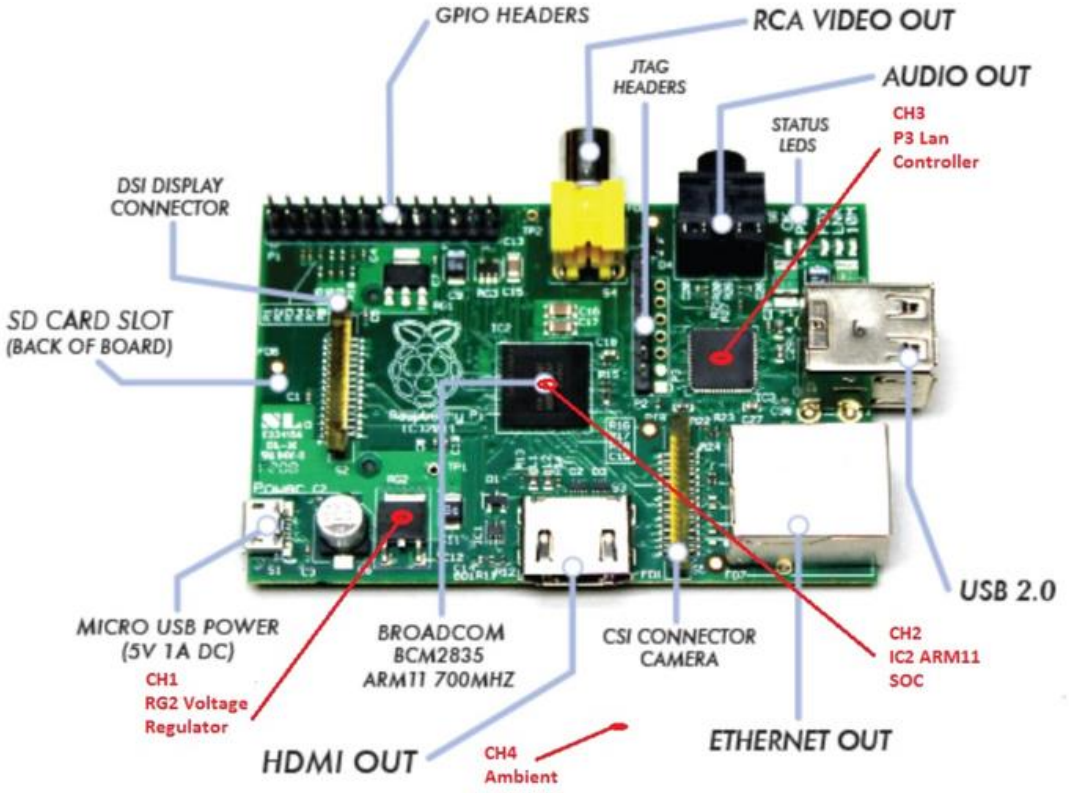


Figura 2.1: Arnaldo Franco (2014). Elementos de la tarjeta. Recuperado de http://hj4afb.blogspot.com/2014/07/host-con-raspberry-pi-para-sdr-primera.html

2.1.1. Sistema Operativo

Raspbian es un sistema basado en Debian creada con el fin de ser útil en el hardware de Raspberry Pi. Raspbian no sólo es un sistema operativo puro; sino que incluye paquetes, programas con precompilados, en un formato que facilita la instalación en la Raspberry Pi. (Raspbian, s.f.)

Raspbian es un software libre que se lo encuentra en la web como un archivo de descarga (en formato .img).

El mismo que por su proceso de instalación es alojado en una tarjeta SD. Permitiendo arrancar su Raspberry Pi con sus respectivas utilidades y funciones. (UBUNTU_FACIL, 2014)

2.1.2. Software Python

2.1.2.1. Introducción

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su escritura dinámica, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. (Rossum, 2009)

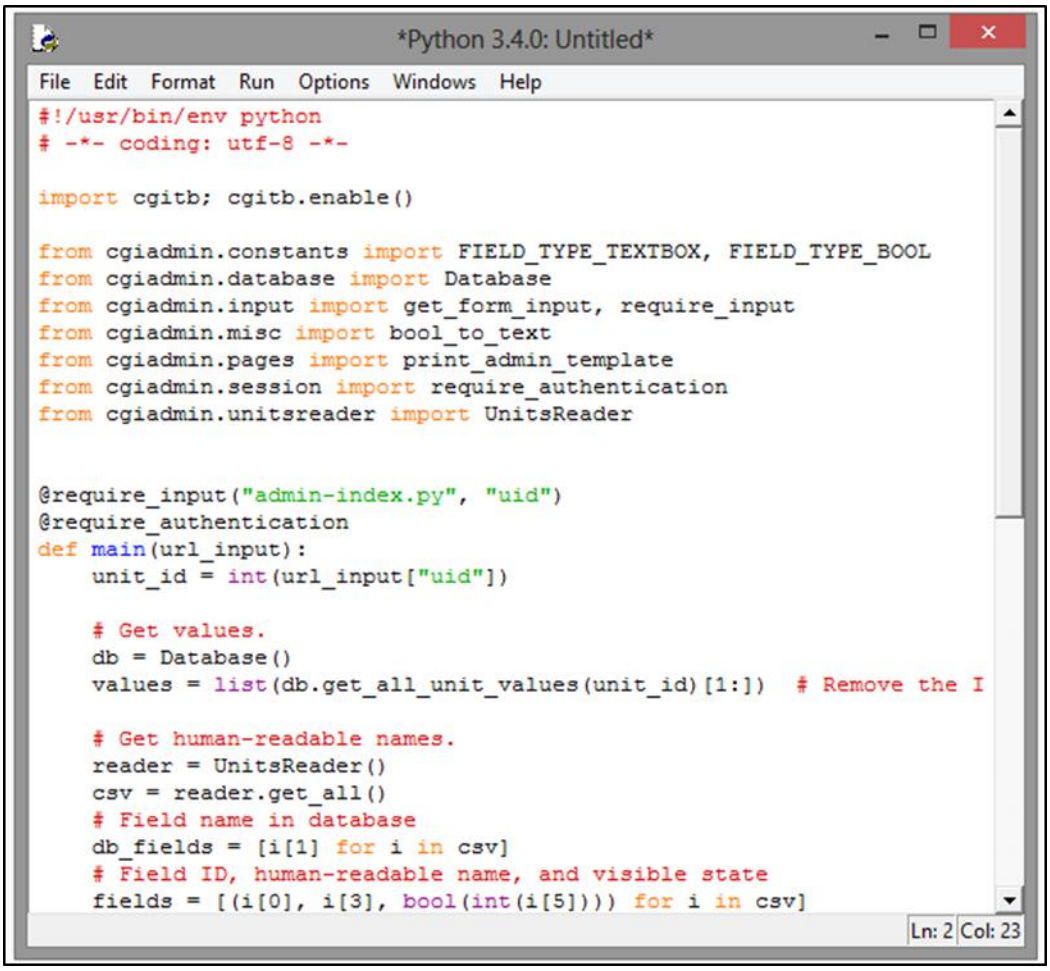


Figura 2.1.2.1: Recursos Python (2014). Consola interactiva de Python. Recuperado de http://recursospython.com/guias-y-manuales/como-empezar-con-python/

2.1.2.2. Características

- Python es un lenguaje de programación creado por Guido van Rossum. A continuación se listan las principales características que este lenguaje posee:
- Usando el lenguaje Python se puede crear todo tipo de programas; programas de propósito general y también se pueden desarrollar páginas Web.
- Debito a la naturaleza de Python de ser Open Source; ha sido modificado para que pueda funcionar en diversas plataformas (Linux, Windows, Macintosh, Solaris, etc...).
- Al ser Open Source es gratuito.
- Al ser un Lenguaje Orientado a Objetos es construido sobre objetos que combinan datos y funcionalidades.
- Al programar en Python no nos debemos preocupar por detalles de bajo nivel, (como manejar la memoria empleada por el programa).
- Se puede insertar lenguaje Python dentro un programa C/C++ y de esta manera ofrecer las facilidades del scripting.
- Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.
- Python tiene una sintaxis muy visual, gracias a que maneja una sintaxis identada (con márgenes), que es de carácter obligatorio. Para separar los bloques de código en Python se debe tabular hacia dentro. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas hechos en Python tengan un aspecto muy similar. (Cuatrorios, 2015)

2.1.3. Software QT

Qt Creator es un IDE (entorno de desarrollo integrado) multiplataforma que se ajusta a las necesidades de los desarrolladores Qt.

Qt Creator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt.

- Editor de código con soporte para C++, QML y ECMAscript
- Herramientas para la rápida navegación del código
- Resaltado de sintaxis y auto-completado de código
- Control estático de código y estilo a medida que se escribe
- Soporte para restructuración de código
- Ayuda sensitiva al contexto
- Plegado de código (code folding)
- Paréntesis coincidentes y modos de selección. (Qt-project, 2014)

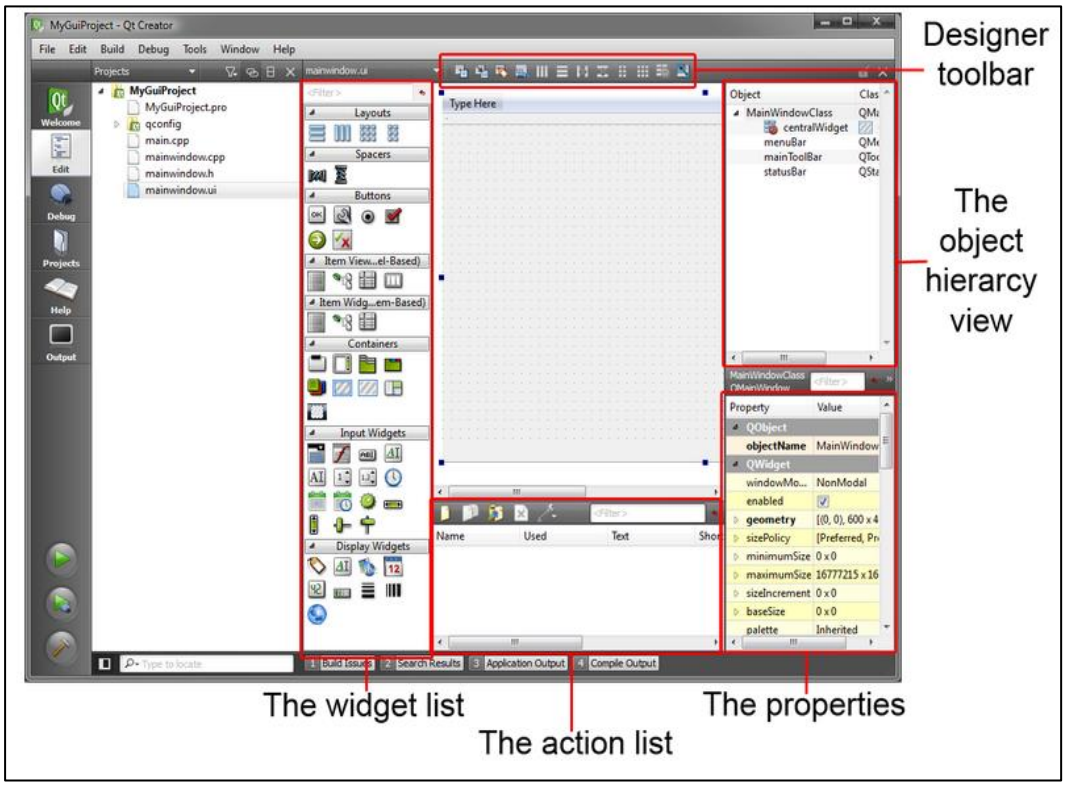


Figura 2.1.3: Uxio (2011). Entorno de Qt-Creator**.** Recuperado de http://uxio0.blogspot.com/2011/03/programar-con-interfaz-grafica-gui-en.html

2.1.4. Software XRDP

Xrdp utiliza el protocolo de escritorio remoto para presentar una interfaz gráfica para el usuario.

El objetivo de este proyecto es proporcionar un servidor completamente funcional terminal de Linux, capaz de aceptar conexiones desde los clientes de escritorio de servidor de terminal propio de Microsoft / remoto.

A diferencia de servidores como: Windows NT / 2000/ 2003/ 2008/2012, xrdp no muestra un escritorio de Windows, sino una ventana de escritorio particular X para el usuario, por sesión. (Sorg, 2013)

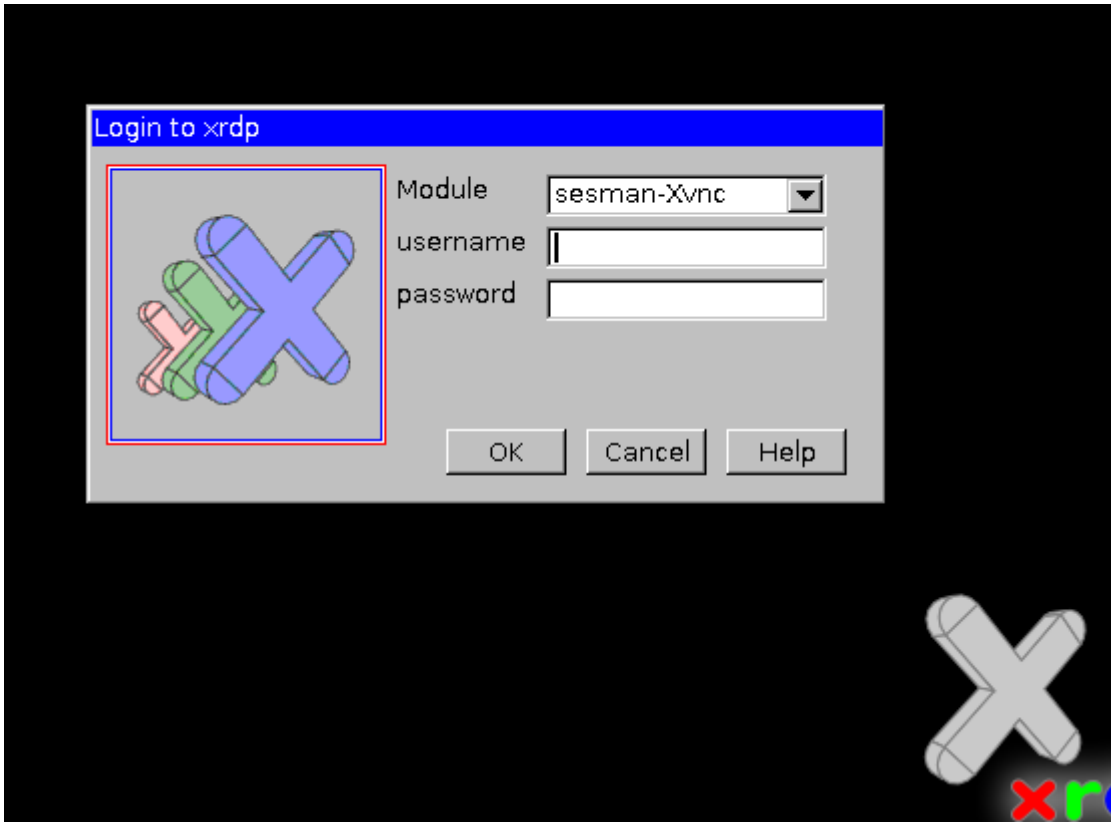


Figura 2.1.4: Griffon (2014). Entorno de ingreso xrdp. Recuperado de http://c-nergy.be/blog/?p=5305

2.1.5. Conectores GPIO

Puerto GPIO del Raspberry Pi se encuentra en la parte superior izquierda de la placa de circuito impreso. Es un puerto de 26 pines, equipado con dos filas de 13 cabeceras de 2,54 mm de sexo masculino.

La separación de estas cabeceras es particularmente importante: 2,54 mm el pin y el espacia entre ellos 0,1 pulgadas. Es un espectáculo muy común en la electrónica, y es el espaciado estándar para la creación de prototipos, plataformas que incluyen Stripboards y placas universales.

Cada pin del puerto GPIO tiene su propio propósito, con varios alfileres que trabajan juntos para formar circuitos particulares (Wiley, 2012).

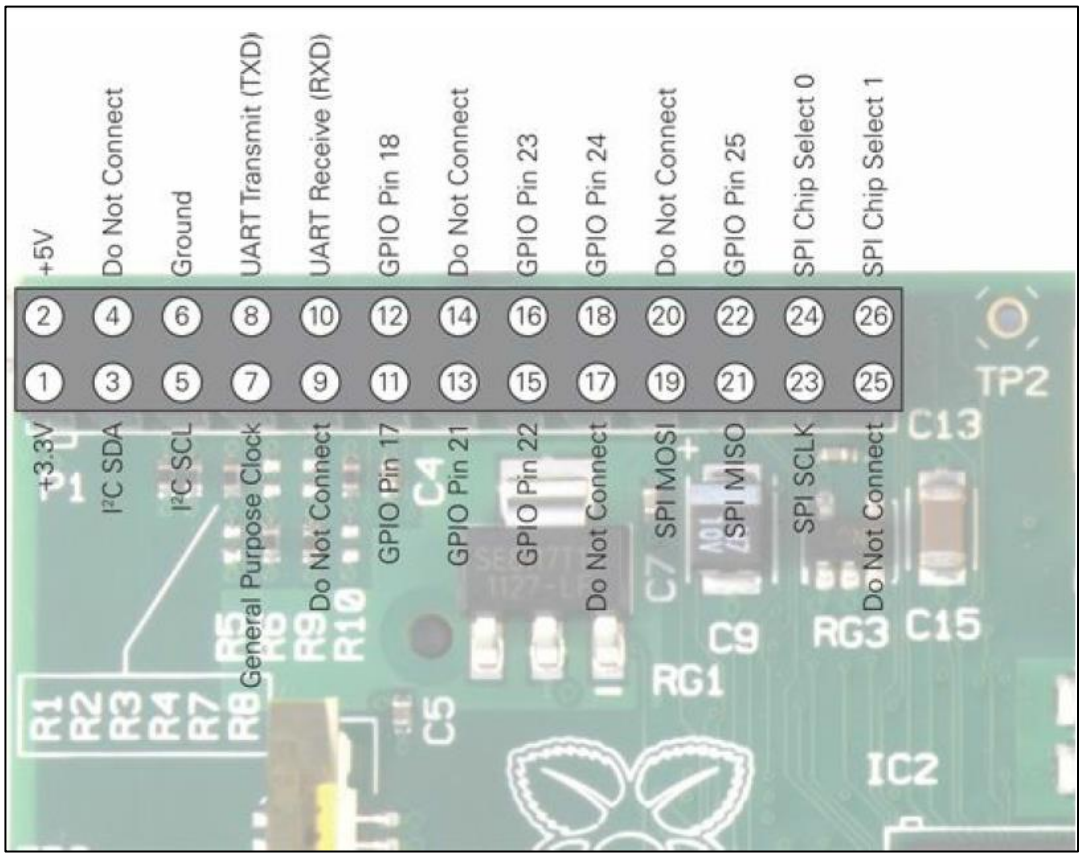


Figura 2.1.5: Wiley (2012). Puerto GPIO de Raspberry Pi y definiciones de pines. Recuperado de (Wiley, 2012)

2.2. Dispositivo de Acceso Biométrico

2.2.1. Módulo Biométrico

Es un dispositivo que provee la lectura de huella digital, enviando un dato con toda la información obtenida, en un texto plano de 1Byte de capacidad.

2.2.2. PIC16F887

El PIC16F887 es un producto conocido de la compañía Microchip. Dispone de todos los componentes disponibles en la mayoría de los microcontroladores modernos. Por su bajo precio, un rango amplio de aplicaciones, alta calidad y disponibilidad, es una solución perfecta aplicarlo para controlar diferentes procesos en la industria, en dispositivos de control de máquinas, para medir variables de procesos etc. Algunas de sus características principales se enumeran a continuación. (MikroElektronika, 2015)

- **Arquitectura RISC**
  - El microcontrolador cuenta con solo 35 instrucciones diferentes
  - Todas las instrucciones son uni-ciclo excepto por las de ramificación
- **Frecuencia de operación:** 0-20 MHz
- **Oscilador interno de alta precisión**
  - Calibrado de fábrica
  - Rango de frecuencia de 8MHz a 31KHz seleccionado por software
- **Voltaje de la fuente de alimentación**
  - Alimentación: 2.0V a 5.5V
  - Consumo: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz) 50nA (en modo de espera)
- **Ahorro de energía:**
  - Modo de suspensión
- **Pines de entrada/salida:**
  - Cantidad: 35
  - Alta corriente de fuente y de drenador para manejo de LED
  - Resistencias pull-up programables individualmente por software
  - interrupción al cambiar el estado del pin
- **Memoria ROM**
  - 8K con tecnología FLASH
  - El chip se puede re-programar hasta 100.000 veces
- **Opción de programación serial en el circuito**
  - El chip se puede programar incluso incorporado en el dispositivo destino.
- **Memoria EEPROM**

- 256 bytes
- Los datos se pueden grabar más de 1.000.000 veces

- **Memoria RAM**
- 368 bytes de

- **Convertidor A/D:**
- 14 canales
- resolución de 10 bits

- **Temporizadores/contadores independientes**
- 3

- **Temporizador perro guardián**
- Si

- **Módulo PWM incorporado**
- Si

- **Módulo USART**
- Mejorado
- Soporta las comunicaciones seriales RS-485, RS-232 y LIN2.0
- Auto detección de baudios

- **Puerto Serie Síncrono Maestro (MSSP)**
- Soporta los modos SPI e I2C. (MikroElektronika, 2015)

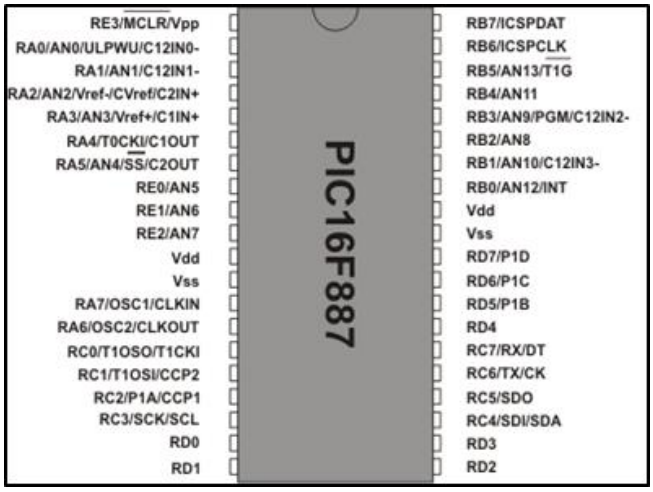


Figura 2.2.2: MikroElektronika (2015). PIC16F887. Por (MikroElektronika, 2015)

2.2.3. Teclado Matricial

Un teclado matricial es un arreglo de botones conectados en filas y columnas, obteniendo una lectura de varios botones con un mínimo número de pines requeridos.

Un teclado matricial 4x4 solamente ocupa 4 líneas de un puerto para las filas y otras 4 líneas para las columnas, de este modo se pueden leer 16 teclas utilizando solamente 8 líneas de un microcontrolador.

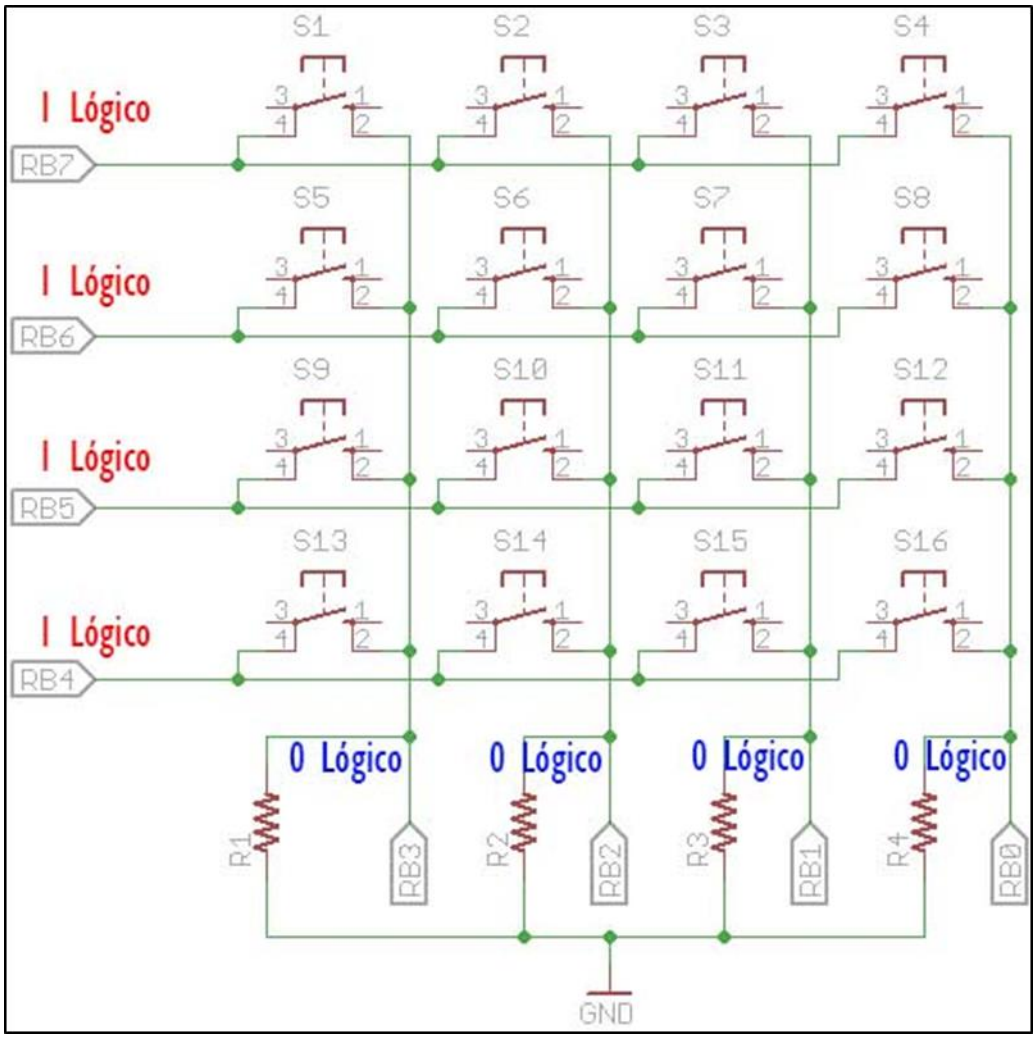


Figura 2.2.3: Rubén (2013). Las conexiones del teclado. Recuperado de http://www.geekfactory.mx/tutoriales/tutoriales-pic/teclado-matricial-con-pic/

2.2.4. Conectores RJ45

La RJ-45 es una interfaz física comúnmente usada para conectar redes de cableado estructurado, (categorías 4, 5, 5e y 6). RJ es un acrónimo inglés de Registered Jack que a su vez es parte del Código Federal de Regulaciones de Estados Unidos.

Posee ocho "pines" o conexiones eléctricas, que normalmente se usan como extremos de cables de par trenzado.

Es utilizada comúnmente con estándares como TIA/EIA-568-B, que define la disposición de los pines o wiring pinout.
Una aplicación común es su uso en cables de red Ethernet, donde suelen usarse 8 pines (4 pares).

Otras aplicaciones incluyen terminaciones de teléfonos (4 pines o 2 pares) por ejemplo en Francia y Alemania, otros servicios de red como RDSI y T1 e incluso RS-232.

Figura 2.2.4: Kevin Nohel García (2013). Tipos de Conectores RJ45. Recuperado de http://inserteladirecciondelblogaqui.blogspot.com/

2.3. Relés

Un relé es un dispositivo electromecánico que permite la conmutación de una línea eléctrica de media o alta potencia a través de un circuito electrónico de baja potencia.

La principal ventaja y el motivo por el que se usa bastante en electrónica es que la línea eléctrica está completamente aislada de la parte electrónica que controla el relé. Es decir, podemos construir un circuito electrónico (un temporizador, una fotocélula, etc.) y, a través de un relé, controlar cualquier tipo de aparato conectado a la red eléctrica. (Gabriela, 2013)

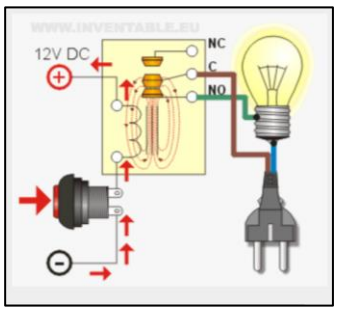


Figura 2.3: Gabriela (2013). Funcionamiento de Relé. Recuperado de http://www.inventable.eu/introduccion-a-los-reles/

2.4. Interruptores Eléctricos

Es un componente que interrumpe o desvía el paso de la corriente eléctrica.

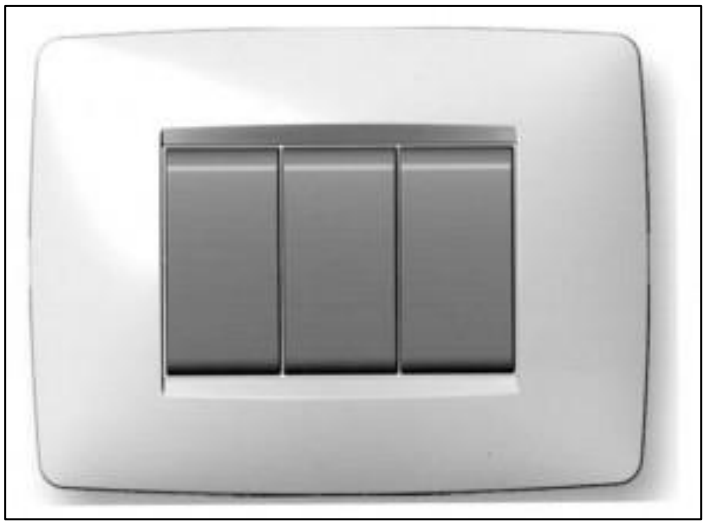


Figura 2.4: Interruptor Eléctrico. Recuperado de http://blog.espol.edu.ec/crielectric/category/instalaciones-domesticas/

CAPÍTULO III

3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1. Especificaciones del Sistema

El sistema consta de las siguientes etapas: captura de Video, control de Luces, módulo principal y acceso al lugar; cada una con características específicas, arquitecturas estratégicas y funcionalidades con alcances determinados.

Dichas etapas trabajan de manera complementaria, por motivo de que una es dependiente a la otra para una funcionalidad óptima del sistema.

3.2. Características y Arquitectura de las etapas del sistema

3.2.1. Análisis de Capturas de Video

La captura de imágenes y video se la realiza con una herramienta de Linux llamada "Motion", que es una aplicación que maneja las cámaras web instaladas, sumado a otras actividades como detección de movimiento, visualización remota entre otras.

Dicha aplicación se configura por medio del archivo "motion.conf" que contiene todas las características y funciones de la aplicación, se complementa con archivos "threadx.conf" donde se encuentran las configuraciones de la cámara que se la identifica con la "x" que es el número de la misma.

3.2.1.1. Imágenes por segundo de video en tiempo real

"Motion" posee la función donde se toma una fotografía a intervalos regulares independientemente del movimiento de la imagen.

Dicho intervalo entre cada captura es un valor de tipo entero, con un rango de valores de 1 a 2147483647, dichas capturas son automáticas y el valor se modifica en el archivo motion.conf en el apartado llamado ‘snapshot_interval’.
Las instantáneas se almacenan en el directorio de destino con un nombre de archivo especificado por la opción ‘snapshot_filename’.

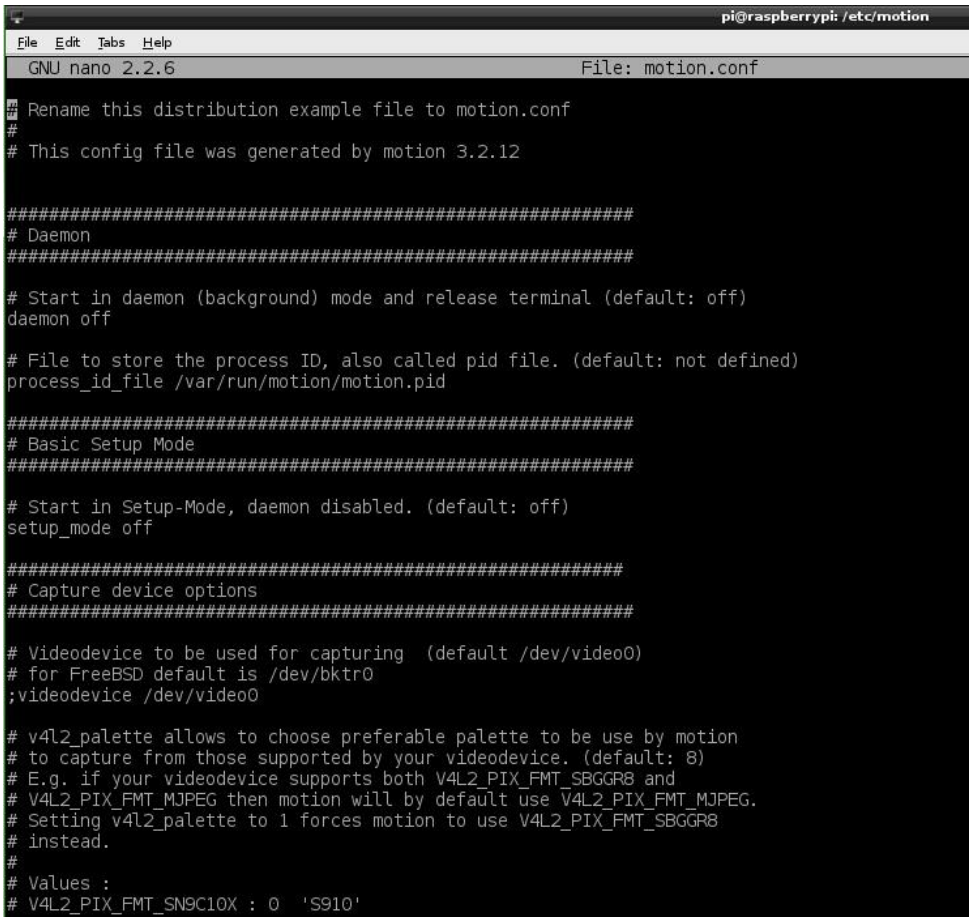


Figura 3.2.1.1: Archivo motion.conf

Fuente: Los Autores

3.2.1.2. Número de Capturas del Sistema

Las capturas realizadas por el sistema van de acuerdo al tiempo de uso del sistema y a los eventos presentados.

Al tiempo de uso del sistema se refiere a que cuando el sistema de vigilancia se inicia comenzará a realizar tomas en el intervalo determinado, pero de una manera

temporal, lo que quiere decir que al cerrar el sistema se eliminan de manera inmediata.

Por otro lado de acuerdo a los eventos presentados se refiere a que mientras el sistema de vigilancia este realizando las tomas temporales puede ocurrir un movimiento por lo que dichas tomas serán enviadas directamente a un repositorio de Google Drive, las mismas que serán guardadas de manera permanente.

3.2.1.3. Ángulos de Cobertura

Para la posición y relación de los ángulos de coberturas de cada cámara se utiliza una estructura estratégica, por motivo de que la colocación de las mismas se la ha realizado con el fin de abarcar la mayor área posible, dando prioridad a las zonas con más posibilidad de movimiento e ingreso.

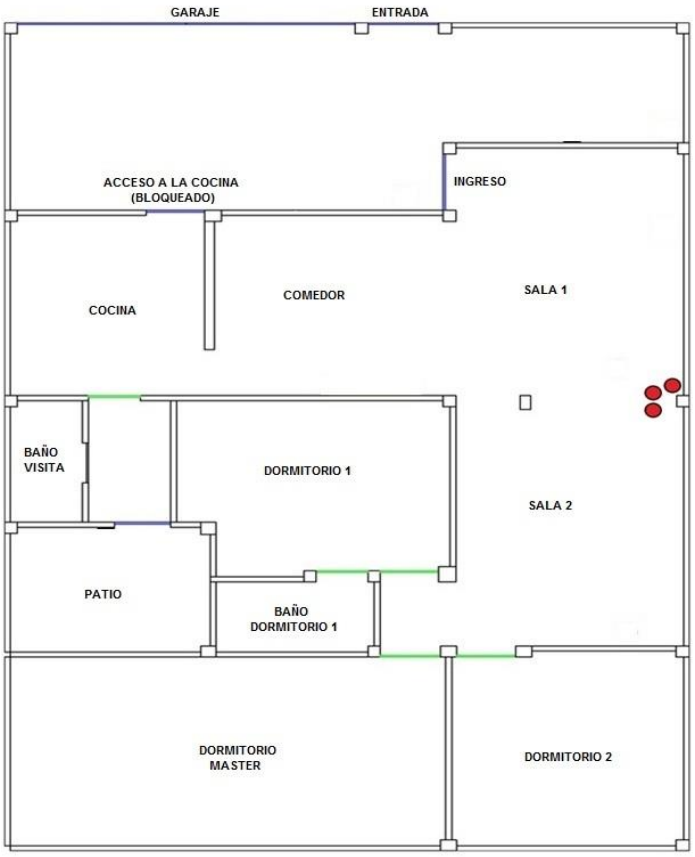


Figura 3.2.1.3: Ubicación de las Cámaras

Fuente: Los Autores

A continuación se muestra un gráfico con el plano de la residencia y las coberturas dadas por cada una de las cámaras y los puntos donde se intersectan, para poder brindar una mayor eficacia en el sistema y así obtener un resultado óptimo en cuanto a la vigilancia de las zonas definidas, llegar a su fin principal al momento de un imprevisto.

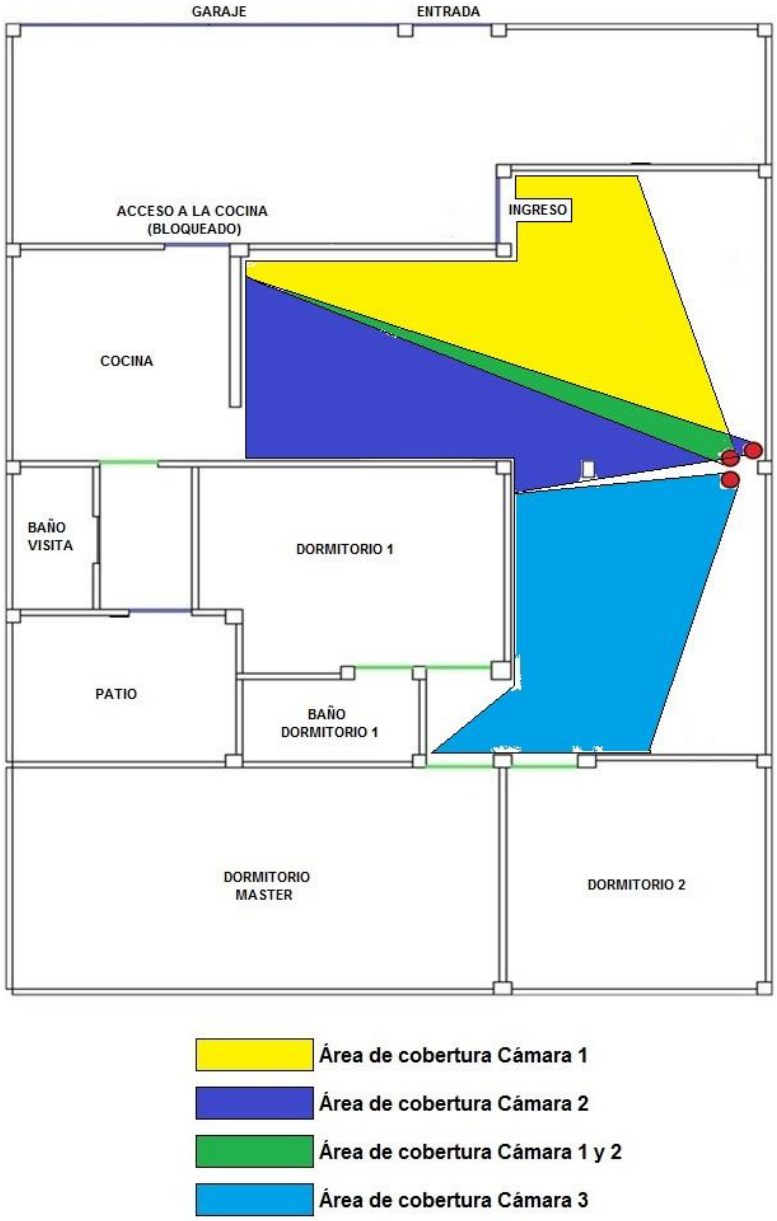


Figura 3.2.1.3: Cobertura de las Cámaras

Fuente: Los Autores

3.2.1.4. Interacción con movimiento

Para lograr la detección de movimiento influye mucho lo que es periodo de activación, por motivo de la claridad del área.

El primer caso se lo tiene cuando el periodo de activación es en la mañana donde existe mayor claridad en las zonas vigiladas, en lo cual las cámaras trabajan de manera cómoda y eficaz sin dependencia de un factor externo.

En cambio, cuando el caso se trata de que el periodo de activación sea en la noche, no se posee claridad en las zonas, dándoles menos visibilidad a las cámaras.

Para evitar esta falencia se colocan sensores de movimiento que enciendan la luz permitiendo realizar las fotografías adecuadas y visibles.

Figura 3.2.1.4: Sensor de movimiento
Fuente: Los Autores

3.2.1.5. Codificaciones realizadas

3.2.1.5.1. Activación y Desactivación de las cámaras

Para lograr la activación de las cámaras conectadas, se edita el código del archivo "motion.conf" y por cada cámara se crea un archivo "threadx.conf" donde la X es el número de cámara.

Las líneas de comando que se editan son las que encontramos en negrillas en el siguiente recorte del código del archivo "motion.conf":

```
############################################################
# Thread config files - One for each camera.
# Except if only one camera - You only need this config file.
# If you have more than one camera you MUST define one thread
# config file for each camera in addition to this config file.
############################################################
# Remember: If you have more than one camera you must have one
# thread file for each camera. E.g. 2 cameras requires 3 files:
# This motion.conf file AND thread1.conf and thread2.conf.
# Only put the options that are unique to each camera in the
# thread config files.

; thread /etc/motion/thread1.conf
; thread /etc/motion/thread2.conf
; thread /etc/motion/thread3.conf
```

El proyecto posee tres cámaras por lo cual debemos tener 3 archivos "threadx.conf" con su respectivo número de identificación en la ubicación "/etc/motion/"

Para activarlas se debe quitar el ";" que tenemos al inicio de cada línea.

Para desactivar alguna cámara se coloca de nuevo en la línea de comando perteneciente a la cámara definida.

Como se solicita tres cámaras se quita el ";" de todas las líneas quedando el código de la siguiente manera:

```
############################################################
# Thread config files - One for each camera.
# Except if only one camera - You only need this config file.
# If you have more than one camera you MUST define one thread
# config file for each camera in addition to this config file.
############################################################
# Remember: If you have more than one camera you must have one
# thread file for each camera. E.g. 2 cameras requires 3 files:
# This motion.conf file AND thread1.conf and thread2.conf.
# Only put the options that are unique to each camera in the
# thread config files.

thread /etc/motion/thread1.conf
thread /etc/motion/thread2.conf
thread /etc/motion/thread3.conf
```

3.2.1.5.2. Alertas de movimientos en capturas

Al momento que se identifica un movimiento se estableció la alerta por medio de un correo eléctrico que es enviado desde una cuenta hacia la misma cuenta cuyos parámetros son configurados en un archivo desarrollado en Python en las siguientes líneas de código:

```
1. # credenciales deGMail
2. self.username = "tu_correo_gmail@gmail.com"
3. self.password = "tu_clave_gmail"
4. self.from_name = "Tu Nombre completo"
5. self.sender = "tu_correo_gmail@gmail.com"
6. self.recipient = "tu_correo_gmail@gmail.com"
7. self.subject = "mensaje de movimiento detectado"
8. self.message = "Detector de movimiento"
9. self.delete_after_upload = "false"
10. self.send_email = "true"
```

En las líneas 2 – 8 son los parámetros para el acceso a la cuenta, direccionamiento y descripción del mensaje.
La línea 9 es para que se borre el mensaje después de ser enviado.
La línea 10 es para enviar el mensaje.
Considerando así este recorte del código de mayor importancia en el archivo desarrollado en Python para el envío del correo electrónico.

3.2.1.5.3. Respaldo de Capturas

Al momento que se identifica un movimiento existe un archivo desarrollado en Python que permite subir las fotos a una cuenta de Google Drive para así obtener un respaldo de los movimientos capturados.

Los parámetros están descritos en un archivo .cfg enlazados por medio de variables vinculadas al archivo desarrollado en Python encargado de subir la foto de respaldo.

3.2.2. Análisis del Control de Luces

El control de luces se la realizó utilizando los conocimientos de eléctrica, como es la activación de un foco desde lugares diferentes.

Para esto se pensó en dos métodos, el primero es la conmutación pero se dedujo que al aplicarlo se produciría mucha inestabilidad al momento de la activación por motivo de que el usuario podía encender la luz y el sistema lo iba a apagar lo cual no era muy acorde a su fin.

El otro método eran los circuitos paralelos en este caso refiriéndose a los interruptores permitiendo la activación ya sea realizada por el usuario y por el sistema también, así esté apagado como se muestra en la figura, lo cual si está acorde con el fin de sistema.

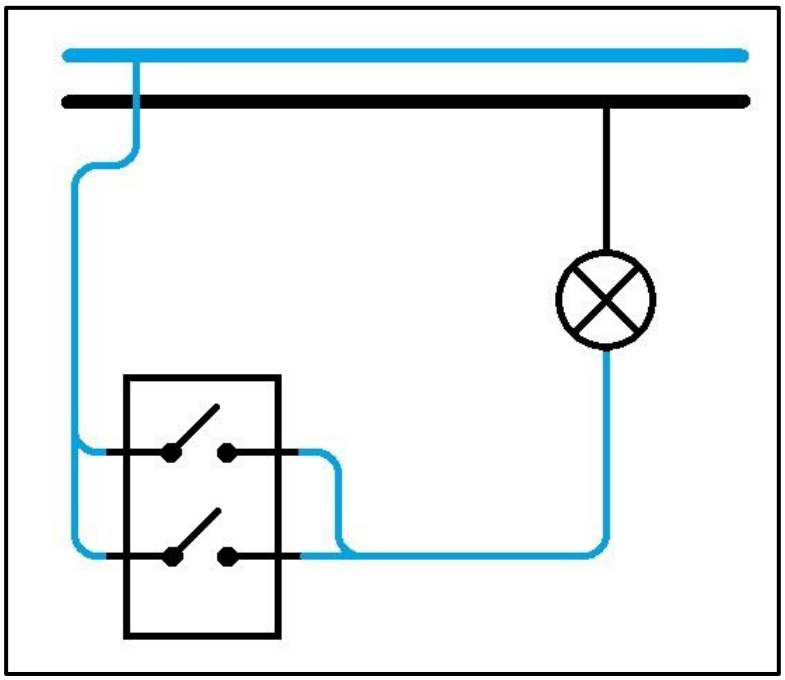


Figura 3.2.2: Interruptores en paralelo

Fuente: Los Autores

3.2.2.1. Elementos controlados

Se sabe que la Raspberry Pi maneja niveles de voltaje por debajo de los 5 VDC, pero al manejar bombillas de 110 VAC, se debe trabajar en una manera de control externa y compatible.

Por lo cual se decidió que los elementos a controlar serían Relés de 5v de activación y que manejan voltajes de 110VAC a 250VAC, como el Módulo de relés de 8 canales.

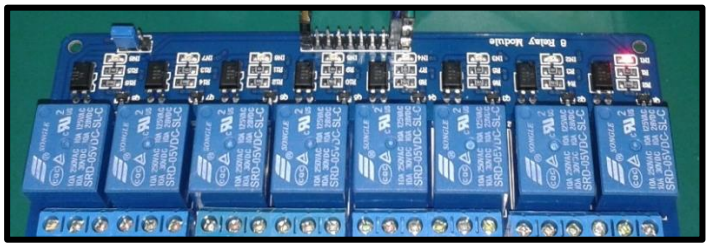


Figura 3.2.2.1: Módulo de relés de 8 canales

Fuente: Los Autores

Este módulo se maneja con los pines GPIO de la Raspberry Pi, en conjunto a un archivo desarrollado en Python que permite encender y apagar un pin determinado, para así activar o desactivar el relé respectivamente.

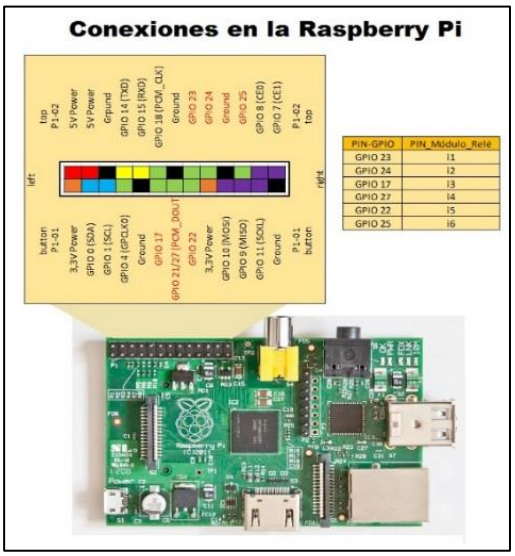


Figura 3.2.2.1: Pines de la Raspberry utilizados por el módulo

Fuente: Los Autores

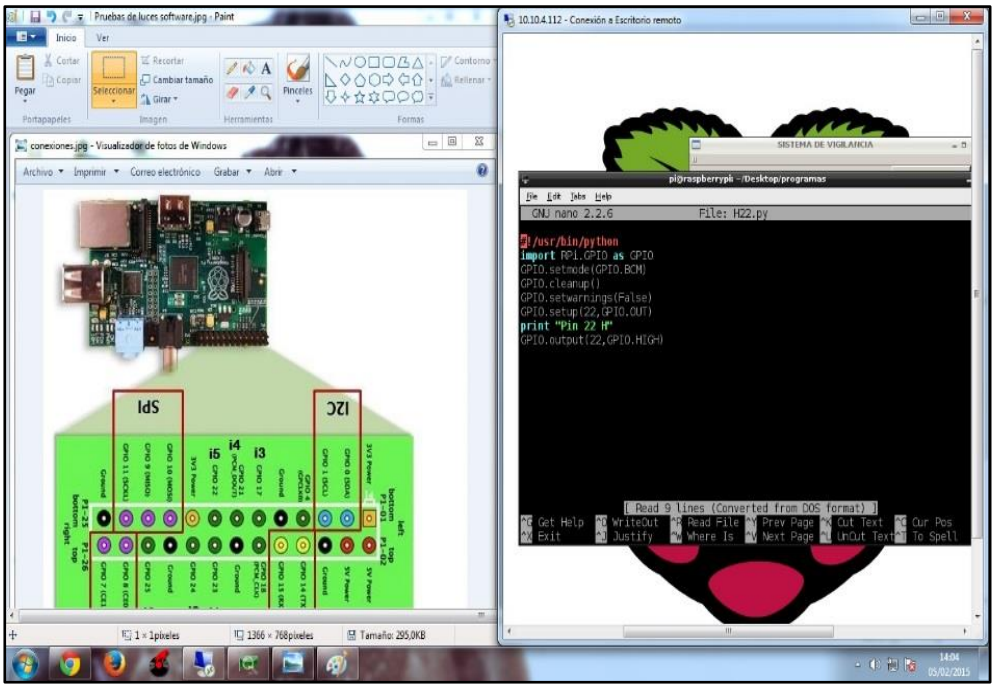


Figura 3.2.2.1: Codificación para el encendido del Pin 22 de la Raspberry pi

Fuente: Los Autores

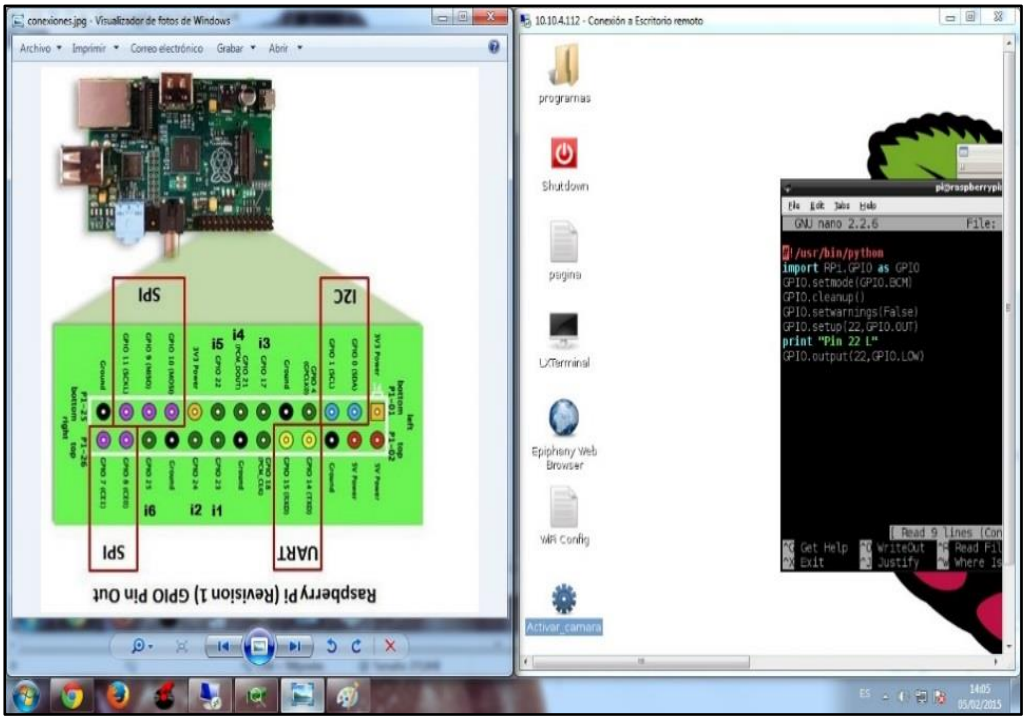


Figura 3.2.2.1: Codificación para el apagado del Pin 22 de la Raspberry pi

Fuente: Los Autores

3.2.2.2. Estudio de esquema del circuito

Para lograr el manejo en la activación y desactivación de las luces en la residencia se optó por estudiar la estructura en la conexión eléctrica, para las luces definidas a controlar.

Lo primero en realizar fue un plano eléctrico de guía para establecer un posible método de conexión o diseño de circuito con cual proveer el mejor y eficaz funcionamiento.

Luego de establecer el circuito se procede a la inspección de los puntos de alimentación eléctrica para las luces definidas.

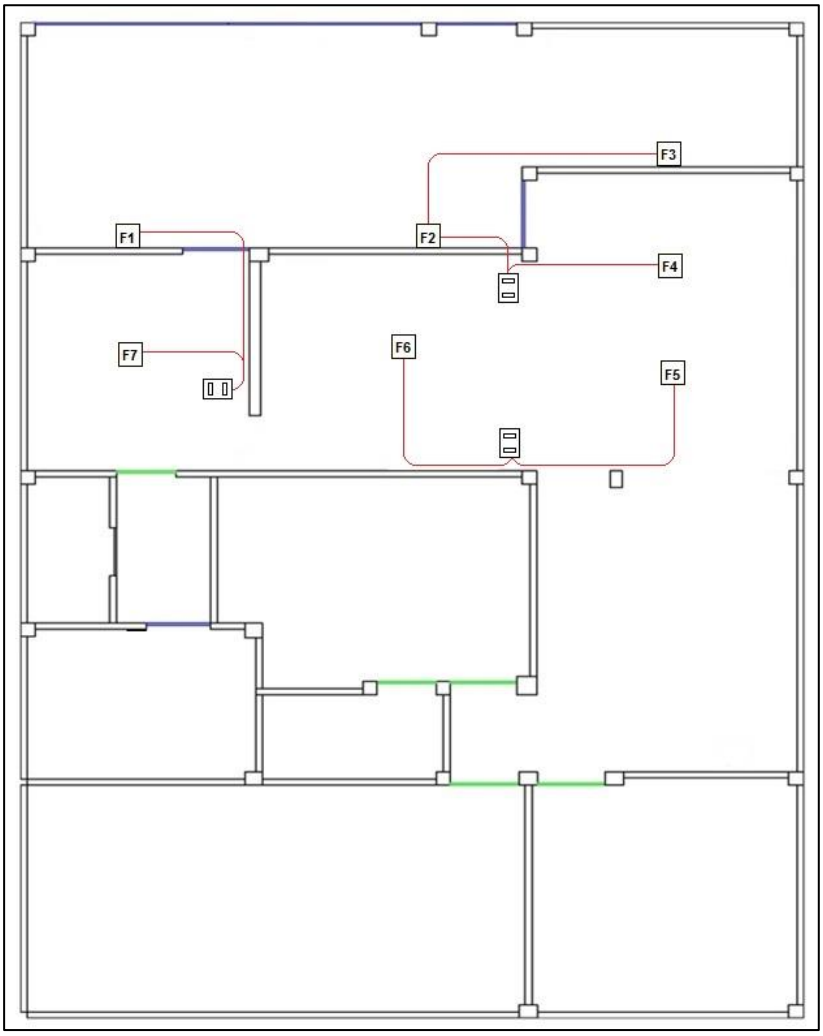


Figura 3.2.2.2: Ubicación de los interruptores controlados

Fuente: Los Autores

3.2.2.3. Diseño del circuito

Al tener establecidos los puntos de alimentación se realizó las pruebas utilizando el método o diseño electrónico que hipotéticamente fue definido, para llegar a obtener un desempeño y funcionamiento óptimo del sistema concluyendo con un provechoso resultado.

Causando incomodidad al usuario al momento de poner en funcionamiento el sistema, tratándose de un diseño en el cual se colocaba el contacto cerrado del relé en serie con el interruptor de la residencia como se muestra en la figura.

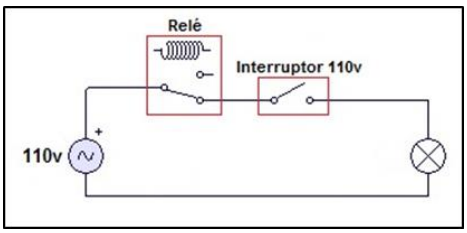


Figura 3.2.2.3: Circuito en serie

Fuente: Los Autores

Este método tuvo el inconveniente que para encender el sistema y se tenga un funcionamiento claro en el control de luces se debía encender todas las luces primero y el sistema se encarga de apagarlos todas, para lograr encenderlas medio del software

Para resolver este malestar se realizó el rediseño del esquema utilizando otro método como lo es el colocar el contacto abierto del relé en paralelo con el interruptor de la residencia como se muestra en la figura

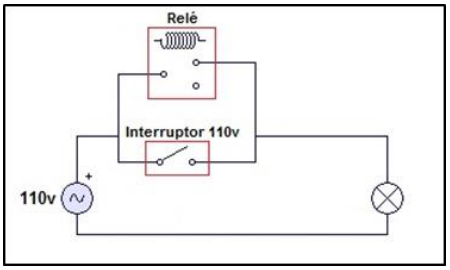


Figura 3.2.2.3: Circuito en paralelo

Fuente: Los Autores

3.2.2.4. Codificaciones realizadas

3.2.2.4.1. Activación de Relé

Para la activación del relé se utilizó un archivo desarrollado en Python que permite emitir 5v por el pin indicado en ##:

1. #!/usr/bin/python
2. import RPi.GPIO as GPIO
3. GPIO.setmode(GPIO.BCM)
4. GPIO.cleanup()
5. GPIO.setwarnings(False)
6. GPIO.setup(##,GPIO.OUT)
7. GPIO.output(##,GPIO.HIGH)

En la primera línea se denota la librería a utilizar en este caso "Python" las siguientes cuatro líneas se encargan de la preparación de los pines, la sexta línea es para configurar al pin ## como salida y la última línea para enviar un alto es decir 5v

3.2.2.4.2. Desactivación de Relé

Para la activación del relé se utilizó un archivo desarrollado en Python que permite emitir 0v por el pin indicado en ##:

1. #!/usr/bin/python
2. import RPi.GPIO as GPIO
3. GPIO.setmode(GPIO.BCM)
4. GPIO.cleanup()
5. GPIO.setwarnings(False)
6. GPIO.setup(##,GPIO.OUT)
7. GPIO.output(##,GPIO.LOW)

En la primera línea se denota la librería a utilizar en este caso "Python" las siguientes cuatro líneas se encargan de la preparación de los pines, la sexta línea es para configurar al pin ## como salida y la última línea para enviar 0v por medio de ese pin.

3.2.3. Análisis de Módulo Principal

3.2.3.1. Elementos del Módulo

Una vez que se tiene el control de luces, los ángulos de coberturas y posición de las cámaras, se procedió a elaborar el módulo principal el mismo que contiene un hub USB de 7 puertos, el módulo de relés y la Raspberry Pi que son alimentados por una Fuente de Poder de 800w incluida en el módulo.

Figura 3.2.3.1: Hub USB de 7 puertos

Fuente: Los Autores

El Hub USB es el encargado de realizar la comunicación con las 3 cámaras que permite la visualización de las áreas determinadas; el módulo de relé se utiliza para el control de luces, dando la parte de domótica al sistema.

Figura 3.2.3.1: Módulo de Relé de 8 Canales

Fuente: Los Autores

El Raspberry Pi es el dispositivo de mayor importancia por ser el encargado de administrar, procesar y dirigir todas las tareas y funciones del sistema.

Figura 3.2.3.1: Raspberry Pi

Fuente: Los Autores

La Fuente de Poder de 800w de potencia para que la alimentación a los elementos antes mencionados, sea eficiente y satisfactoria

Figura 3.2.3.1: Fuente de Poder de 800 w

Fuente: Los Autores

3.2.3.2. Estudio para la estructura del Módulo

Se analizó el tema de que los dispositivos no pueden estar juntos por motivo de que algunos de los mismos no tienen un carcaza protectora y si la poseen son metálicas por lo cual puede ser causa un corto circuito.

Para prevenir esto se hizo una estructura donde cada elemento tendría un lugar aislado del otro por medio de comparticiones de acrílico con topes que permite una estabilidad concreta en la posición de los mismos.

3.2.4. Análisis de Acceso al lugar

3.2.4.1. Elementos controlados

Se sabe que un microcontrolador es un integrado programable y reprogramable que ejecuta instrucciones grabadas en él; Por medio de un software el cual supervisa distintos tipos de periféricos y memorias internas, en compañía de una CPU y memorias externas que permiten el control por medio de entradas y salidas.

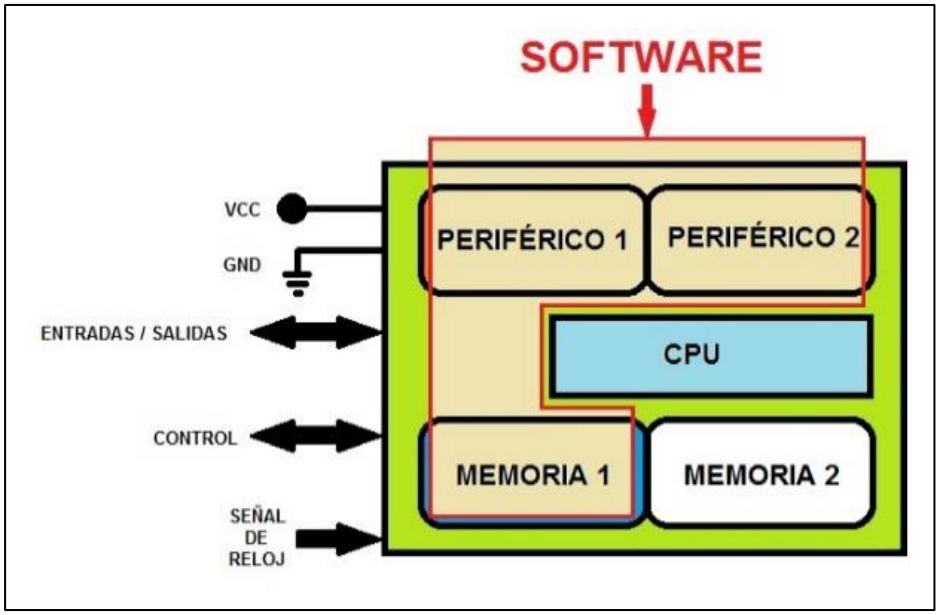


Figura 3.2.4.1: Estructura Interna del microcontrolador

Fuente: Los Autores

Para el proyecto se utilizó el PIC16F887, porque favorecen muchas de las características propias de él como:

- Poseer una gran cantidad de entradas y salidas.
- Poseer una Señal de Reloj interna y configurable.

Este microcontrolador estará encargado de:

1. Procesar los datos recibidos desde el módulo lector de huellas interpretador por la FPU.
2. Comparar los datos recibidos desde el módulo lector de huellas interpretador por la FPU.
3. Permitir el ingreso al menú de acción.
4. Permitir la apertura de la Chapa Eléctrica.
5. Realizar el ingreso de nuevas huellas digitales.

3.2.4.2. Estudio de esquema del Dispositivo de Control

El esquema del Control de Acceso Biométrico se basó en la diferencia entre dos tipos de ingreso biométrico: "Identificación de huella digital" y "Verificación de huella digital", los cuales se explican a continuación:

1. Identificación de huella digital o IDENTIFY

Se basa en el siguiente proceso:

a. Se ingresas las tramas de las huellas digitales determinadas por el usuaria a la FPU

b. Cada vez que se ingrese una huella digital para la comparación, el microcontrolador realiza una búsqueda en las "n" huellas digitales ingresadas en la FPU

c. Una vez comparadas con todas las huellas, al encontrar una trama idéntica se procede a dar acceso.

2. Verificación de huella digital o VERIFY

Se basa en el siguiente proceso:

a. Se ingresas las tramas de las huellas digitales determinadas por el usuario con un número específico de identificación por huella a la FPU.

b. Cada vez que se ingrese una huella digital para la comparación, el microcontrolador realiza una búsqueda directa con el número de identificación.

c. Una vez encontrado el número, se compara las dos tramas y si son idénticas se procede a dar acceso.

3.2.4.3. Diseño del Dispositivo de Control

El diseño del dispositivo se basa en ubicar cada uno de los componentes de manera muy provechosa, para así obtener como resultado un dispositivo que ocupe el espacio menos posible.

El dispositivo está formado por los siguientes componentes:

- Placa principal
- Microcontrolador PIC16F887
- Teclado Matricial
- LCD 16x2
- Lector de Huella Digital RS232
- Módulo de Entrenamiento o FPU
- Relé (Control de Chapa)
- Entrada para programador

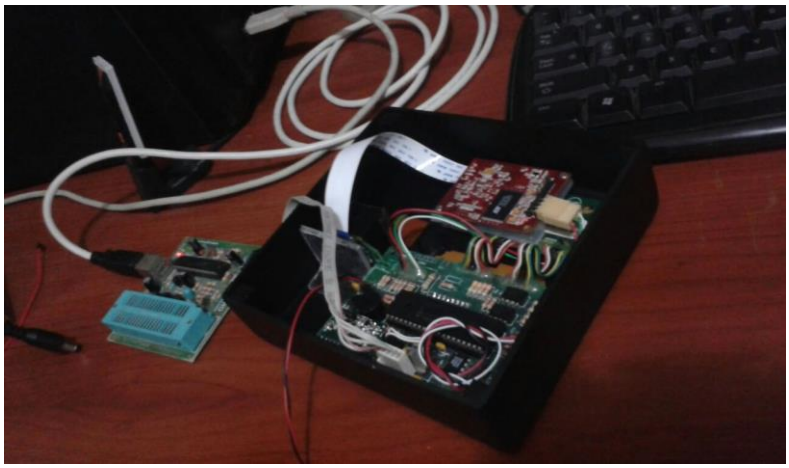


Figura 3.2.4.3: Estructura Interna del Dispositivo Biométrico

Fuente: Los Autores

La programación se basa en una estructura de lazo cerrado en donde se puede realizar las siguientes tareas:

- Ingreso de Huella Dactilar con el proceso llamado “ENROLAR”.
- Eliminación de Huella Dactilar con el proceso llamado “BORRAR”.

Ambas tareas se encontrarán dentro de una única opción del menú principal que lleva en nombre de “Enrolar/Borrar Huella”.

Y para un mejor entendimiento se muestra unos diagramas de bloques con los procesos y tareas que contiene el Dispositivo de Acceso Biométrico.

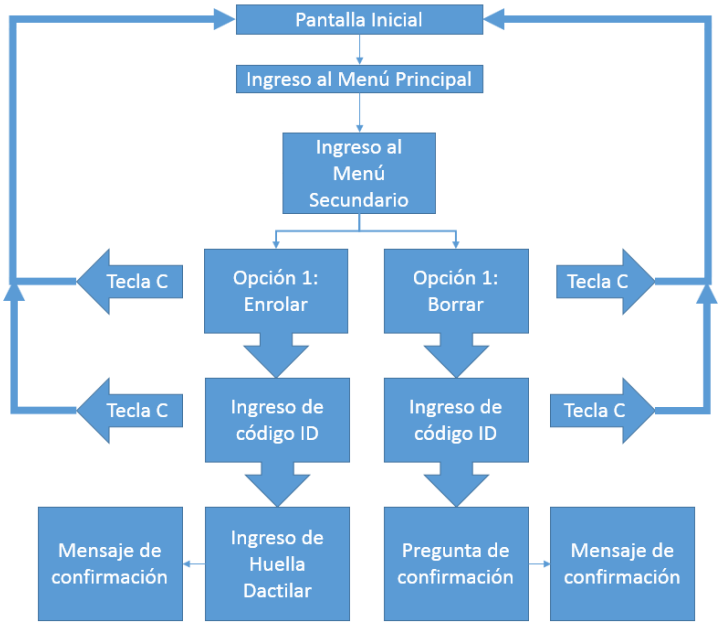


Figura 3.2.4.3: Diagrama de funciones del biométrico

Fuente: Los Autores

3.2.4.4. Codificaciones realizadas

Las codificaciones destinadas para este dispositivo fueron desarrolladas en el lenguaje PICC, por motivo de que el mismo se centraliza en un microcontrolador y la comunicación con un FPU y un módulo biométrico. Para llevar a cabo el enrolamiento de la huella dactilar se basó en el tipo de ingreso llamado “Verificación de huella digital o VERIFY”.

3.2.4.4.1. Enrolamiento del número de identificación

El ingreso o enrolamiento se la desarrolló por partes, como primera se tiene el enrolamiento del número de identificación.

```
int1 INGRESARCODIGOENROLAR(void){
int1  retorno;
InitCodTransicion();
do{
Alarma();
lcd_putc('\f');
lcd_putc("    ENROLANDO");
lcd_putc('\n');
lcd_putc("    ID: ****      ");
delay_ms(100);
do{
opcion=teclado_get();
if (opcion=='0' || opcion=='1' || opcion=='2' || opcion=='3' || opcion=='4' || opcion=='5' ||
opcion=='6' || opcion=='7' || opcion=='8' || opcion=='9'){
lcd_gotoxy(1,2);
codtransicion[0]=opcion;
printf(lcd_putc,"    ID: %c***      ",codtransicion[0]);
delay_ms(500);
do{
opcion=teclado_get();
if (opcion=='0' || opcion=='1' || opcion=='2' || opcion=='3' || opcion=='4' || opcion=='5' ||
opcion=='6' || opcion=='7' || opcion=='8' || opcion=='9'){
lcd_gotoxy(1,2);
codtransicion[1]=opcion;
printf(lcd_putc,"    ID: %c%c**      ",codtransicion[0],codtransicion[1]);
delay_ms(500);
do{
opcion=teclado_get();
if (opcion=='0' || opcion=='1' || opcion=='2' || opcion=='3' || opcion=='4' || opcion=='5' ||
opcion=='6' || opcion=='7' || opcion=='8' || opcion=='9'){
lcd_gotoxy(1,2);
codtransicion[2]=opcion;
printf(lcd_putc,"    ID: %c%c%c*      ",codtransicion[0],codtransicion[1],codtransicion[2]);
delay_ms(500);
do{
opcion=teclado_get();
if (opcion=='0' || opcion=='1' || opcion=='2' || opcion=='3' || opcion=='4' || opcion=='5' ||
opcion=='6' || opcion=='7' || opcion=='8' || opcion=='9'){
  lcd_gotoxy(1,2);
  codtransicion[3]=opcion;
  printf(lcd_putc,"    ID: %c%c%c%c      ",codtransicion[0],codtransicion[1],codtransicion[2],codtransicion[3]);
  delay_ms(1000);
  InitBuffer();
  retorno = 1;
  return retorno;
}
}while(opcion!='C');
}
}while(opcion!='C');
}
}while(opcion!='C');
}
}while(opcion!='C');
}while(opcion!='C');
retorno = 0;
return retorno;
}
```

Figura 3.2.4.4.1: Código para enrolar el número de identificación en el FPU

Fuente: Los Autores

En la figura se describe el código perteneciente al enrolamiento, sólo del número de identificación que se lo ingresa por medio de una trama estándar y que se la realiza por cada número digitado donde se observa las líneas resaltadas.

3.2.4.4.2. Enrolamiento de huella dactilar

La segunda parte es el enrolamiento de la huella que se lo realiza mediante una trama que es la primera línea resaltada en la figura.

En ella se describe el código del enrolamiento de la huella dactilar, las alarmas y el envío al buffer que es donde se guarda la trama y se lo encuentra en la segunda línea resaltada.

```
int1 ENROLARHUELLA(void){
int1 retorno;
x=0;
opcion=' ';
Alarma();
HuellaR:
do{
flag=0;
flg_mensaje=0;
lcd_putc('\f');
lcd_putc("   ENROLANDO:    ");
lcd_putc('\n');
lcd_putc(" INGRESE HUELLA");
lcd_putc('\n');
delay_ms(500);
opcion=teclado_get();
Fprintf(BIO,"ñ0203C5ò");
do{
if(flg_mensaje){
length=strlen(buffer);
if(length!=14){
x++;
if (x==3){
Alarma();
x=0;
InitBuffer();
retorno=1;
return retorno;
}
}else{
goto huellaR;
}
flg_mensaje=0;
flag=1;
}
}while(flag!=1);
}while((x!=3) || (opcion!='C'));
}
```

Figura 3.2.4.4.2: Código para enrolar la huella dactilar en el FPU

Fuente: Los Autores

3.2.4.4.3. Enrolamiento de huella y código

La última parte es el enrolamiento de la huella y el número de verificación que se lo realiza mediante una trama que es línea resaltada en la figura.

En ella se describe el código del enrolamiento de la huella dactilar, y cada transición del número.

```
int1 ENROLARHUELLACODIGO(void){
int1  retorno;
flg_mensaje=0;
flag=0;

Checksum_Enrolar();
Fprintf(BIO,"ñ0204,1,%c%c%c%c%Xò",codtransicion[0],codtransicion[1],codtransicion[2],codtransicion[3],chec
ksum);
do{
if(flg_mensaje){
length=strlen(buffer);
if(length==11){
Alarma();
lcd_putc('\f');
lcd_putc(" HUELLA GRABADA ");
lcd_putc('\n');
lcd_putc("  EXITOSAMENTE ");
delay_ms(1000);
}else{
if(length==14){
if(buffer[10]=='4'){

lcd_putc('\f');
lcd_putc("    HUELLA    ");
lcd_putc('\n');
lcd_putc("  INCORRECTA   ");
delay_ms(1000);
retorno =1;
return   retorno;
}else{
if(buffer[10]=='D'){
lcd_putc('\f');
lcd_putc("    CODIGO   ");
lcd_putc('\n');
lcd_putc("    EN USO   ");
delay_ms(1000);
retorno = 0;
return retorno;
}
}
}
}
flag=1;
flg_mensaje=0;
}
}while(flag!=1);
}
```

Figura 3.2.4.4.3: Código para enrolar la huella y el número en el FPU

Fuente: Los Autores

CAPÍTULO IV

4 ANÁLISIS EXPERIMENTAL DEL SISTEMA

4.1. Análisis Experimental de Capturas de las Cámaras

4.1.1. Resultados y Comparaciones

Realizando las pruebas con las cámaras se obtuvo el siguiente proceso de desarrollo Al inicio se comenzó con 7 cámaras en lo que se pudo notar una gran distorsión en la transmisión. A lo cual se dedujo tres posibles causas:

- Distancia de las cámaras.
- Potencia de la Alimentación.
- Extensiones deterioradas.
- Número de cámaras.

De las cuales se eligió primero la cuarta opción reduciendo de 7 a 5 y por ultimo de 5 a 3 cámaras, con lo cual se obtuvo un mejoramiento en la distorsión.

Después se analizó la tercera opción, dando como solución la revisión y resoldada de los conectores USB al cable UTP, componentes pertenecientes a las extensiones.

Luego de esto se procedió a una nueva prueba pero el problema persistía por lo cual se consideró la segunda causa, y una solución viable fue colocar un HUB ALIMENTADO para las tres cámaras, con lo cual se obtuvo un gran mejoramiento pero no a largo plazo, porque al transcurso de las demás pruebas la calidad de transmisión iba bajando.

Por ese motivo se hizo el cambio de HUB por otro igualmente alimentado y la adaptación de una fuente de alimentación de 800w para el HUB y la Raspberry.

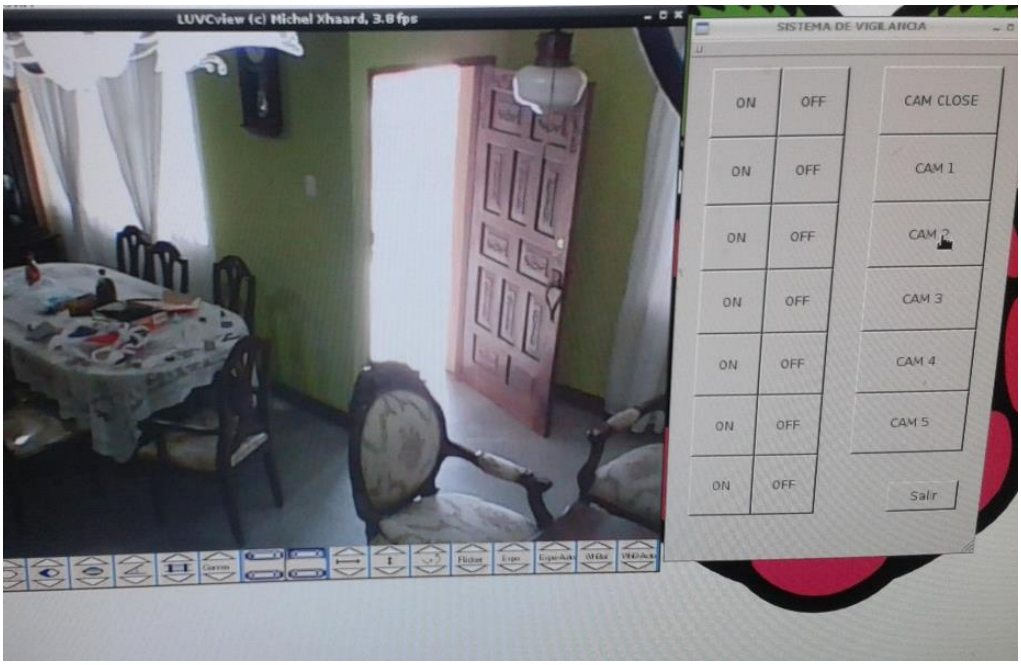


Figura 4.1.1: Prueba de Captura de las cámaras

Fuente: Los Autores

Al realizar ese cambio se notó que el sistema sólo aceptaba una cámara, con eso finalmente se eligió la primera causa que era la distancia a lo cual se acortó y colocó de manera estratégica el enfoque de cada cámara para así tener la misma cobertura.

Atendiendo todas estas posibles causas se obtuvo una óptima transmisión de video, con un enfoque y cobertura bastante amplia.

4.2. Análisis Experimental del Control de luces

4.2.1. Resultados y Comparaciones

Realizando las pruebas con las luces se obtuvo el siguiente proceso de desarrollo: Al inicio se realizó con 3 luces donde trabajaba con normalidad pero al colocar las 6 luces se produjo un tripeo de breaker. A lo cual se dedujo tres posibles causas:

- Corto Circuito.

- Sobre Carga

Para lo cual se escogió la segunda opción, dando como solución la revisión de las conexiones eléctricas realizadas. Pero se notó que todo estaba correcto y se descartó este posible problema.

Luego de esto se analizó estudios sobre la "curva de corriente al arranque con carga" por lo que se dedujo que era el problema, concluyendo que no se podían encender todas las luces a la vez sino una por una, y el apagado si de las dos maneras.

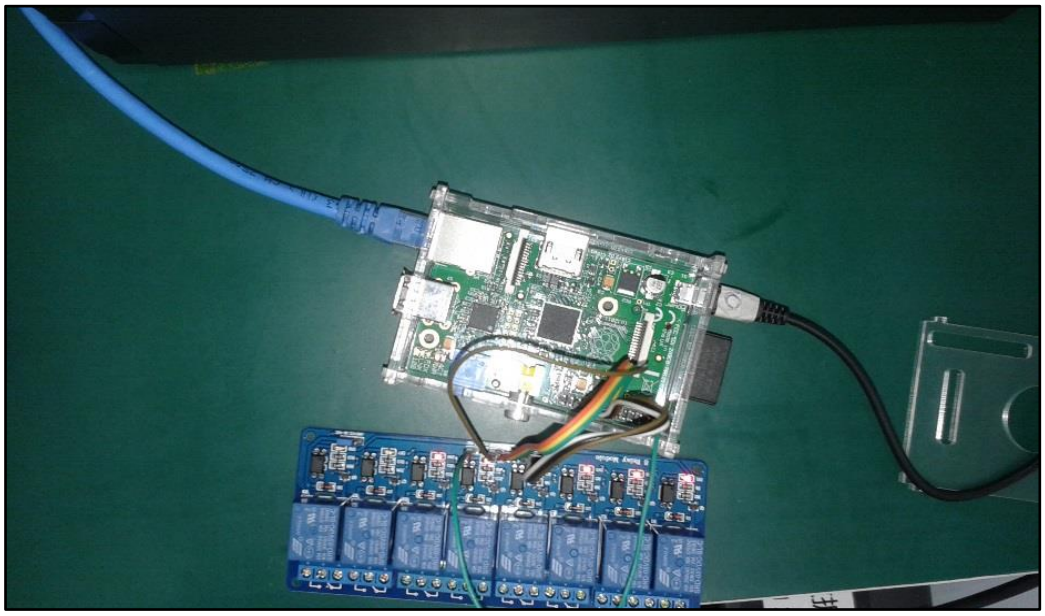


Figura 4.2.1: Prueba de Encendido de Luces

Fuente: Los Autores

4.3. Análisis Experimental de los Dispositivos de Acceso biométrico

4.3.1. Resultado y Comparación

Realizando las pruebas con el Dispositivo Biométrico se obtuvo el siguiente proceso de desarrollo:

Al inicio se realizó con el ingreso de 3 huellas registradas trabajaba con normalidad pero en momentos la lectura de la trama no era exitoso. A lo cual se dedujo tres posibles causas:

- Módulo Dañado.
- Lectura errónea.

Por lo cual se eligió la primera opción, por motivo de que era la de mayor interés dando como solución la revisión del módulo en otro microcontrolador y trabajó con normalidad con lo cual se descartó.

Para esto se aplicó estudios de microcontroladores, analizando el código en el cual se notó que el problema radicaba en el mismo por ese motivo se procedió con una revisión exhaustiva y se rectificó de inmediato.

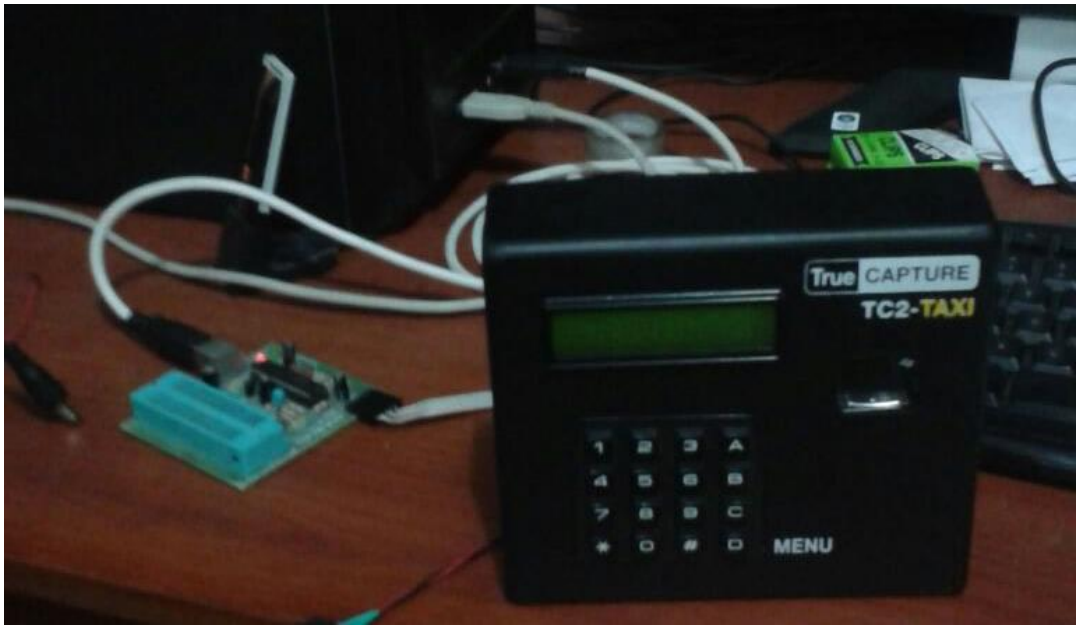


Figura 4.3.1: Pruebas con el Dispositivo Biométrico

Fuente: Los Autores

4.4. Resultados Finales

4.4.1. Módulo Principal

Como resultado del proyecto se obtuvo un módulo principal que es donde se centraliza las tres partes fundamentales del sistema como son: la interface de control con acceso remoto, la domótica y la vigilancia.

Figura 4.4.1: Módulo Principal

Fuente: Los Autores

4.4.1.1. Control y visualización remota

El control del sistema se lleva a cabo en una interface desarrollada en QTCreator, que facilita al usuario el manejo del mismo por medio de una conexión de Escritorio Remoto de Google Chrome. Esto se realiza accediendo al ordenador desde el móvil por medio de la app Escritorio Remoto Google Chrome.

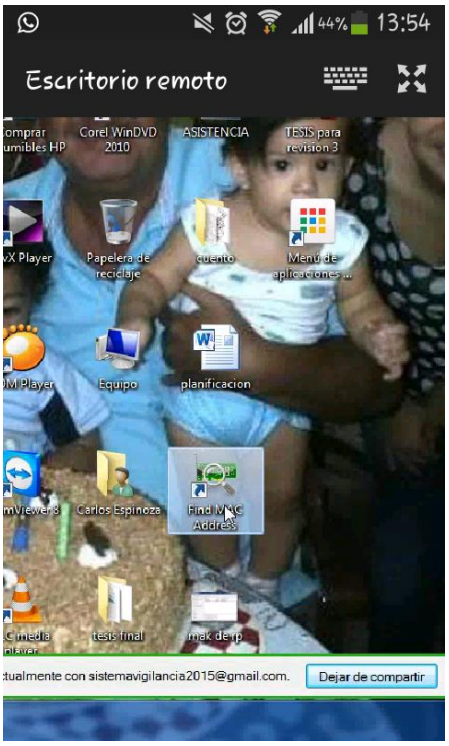


Figura 4.4.1.1: Icono de Find MAC Address en el Escritorio del PC local

Fuente: Los Autores

En el escritorio del ordenador local se busca la aplicación de Windows llamada "Find MAC Address".

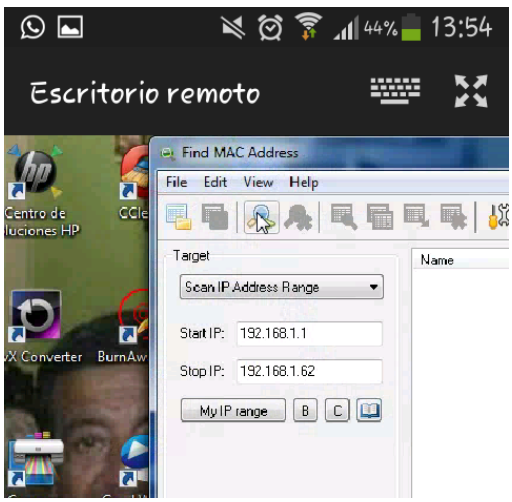


Figura 4.4.1.1: Parámetros de búsqueda en el programa Find MAC Address

Fuente: Los Autores

En la aplicación se establece una IP INICIAL e IP FINAL. Luego se procedió a iniciar la búsqueda. Los resultados se muestran en una lista del lado derecho

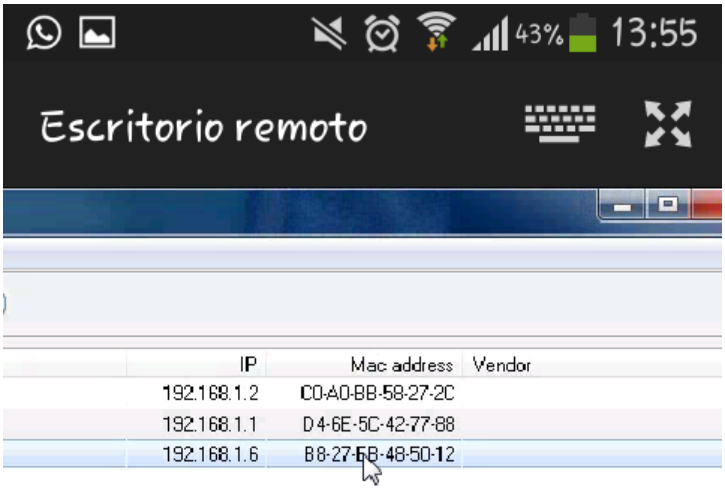


Figura 4.4.1.1: Lista de Equipos conectados a la red

Fuente: Los Autores

En la lista de resultado se busca la MAC Adress de nuestra Raspberry Pi y se ubica la ip de la red. Con este dato se realiza una Conexión a Escritorio Remoto de Windows hacia la IP definida.

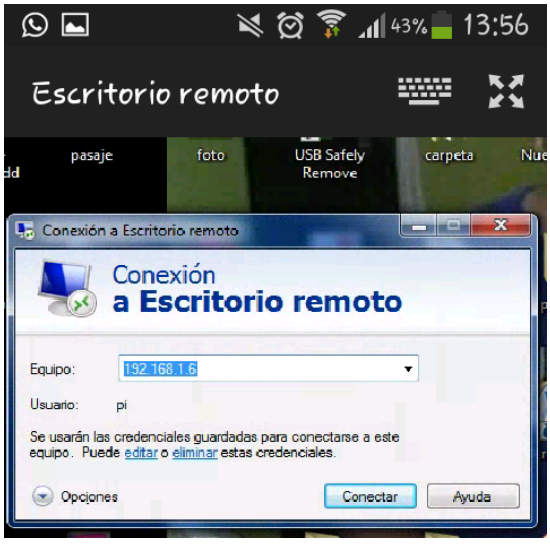


Figura 4.4.1.1: Conexión a Escritorio remoto de Windows

Fuente: Los Autores

Al conectarse a la Raspberry se ingresa el usuario y la contraseña de la misma. Luego mostrará el escritorio donde se busca el icono de la interfaz del sistema llamada “EYES ONE CLICK”

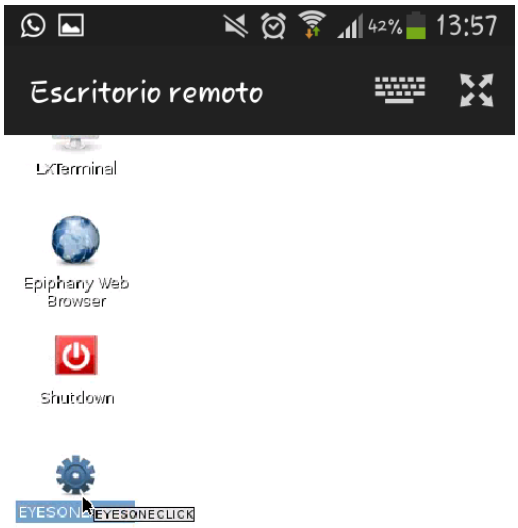


Figura 4.4.1.1: Icono de Interface desarrollada en QTCreator

Fuente: Los Autores

Al dar doble clic aparece una ventana emergente en la cual se elige la opción “EXECUTE”

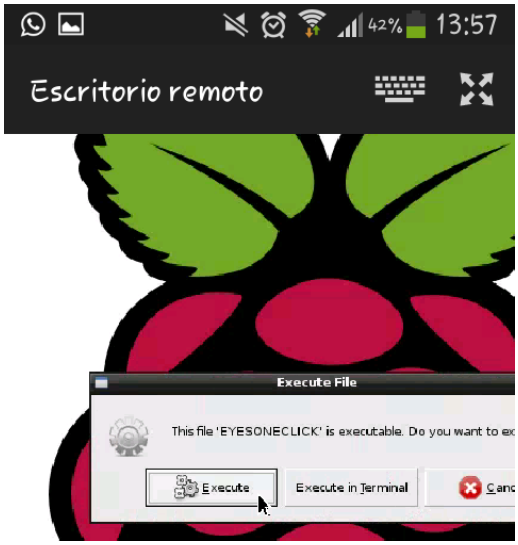


Figura 4.4.1.1: Mensaje de ejecución de la interface desarrollada en QTCreator

Fuente: Los Autores

Y se abrirá la ventana con la interface de control del sistema.

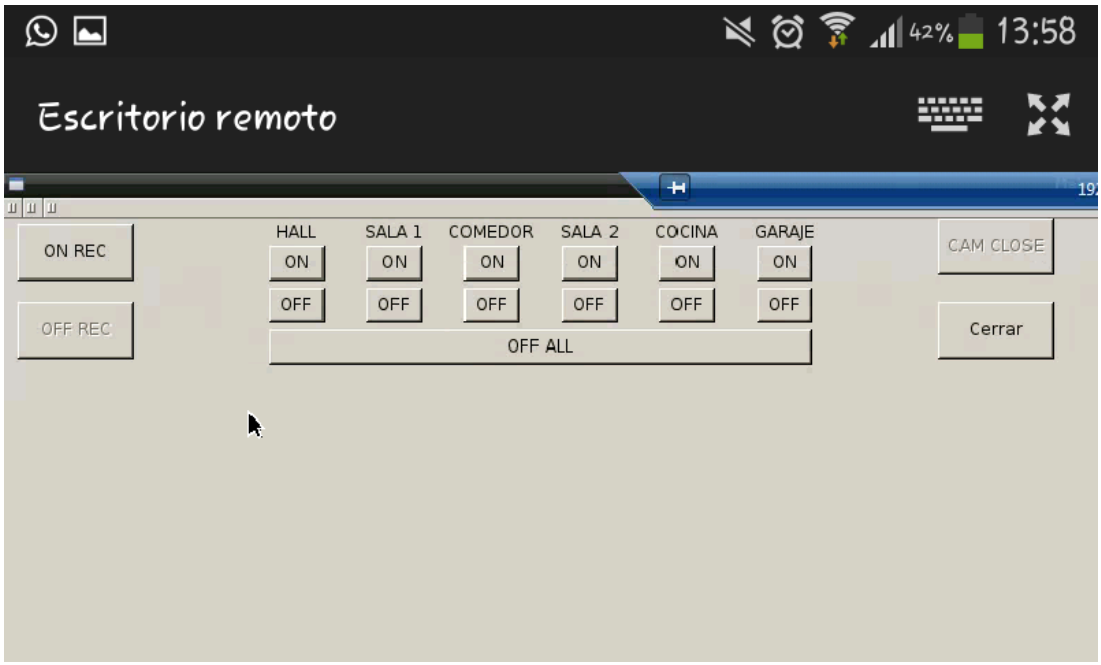


Figura 4.4.1.1: Interface desarrollada en QTCreator

Fuente: Los Autores

4.4.1.2. Domótica (Control de luces)

En la parte de domótica se obtuvo un eficiente resultado ya que por medio de un panel permitió un control exacto de las luminarias definidas del proyecto.

Figura 4.4.1.2: Panel de Control de Luces

Fuente: Los Autores

SALA 1 es el área que se nota primero al ingresar a la vivienda, la cual es alumbrada por un solo foco

Figura 4.4.1.2: Primer foco de la sala

Fuente: Los Autores

COMEDOR es el área que se encuentra en dirección a la cocina, la cual es alumbrada por un solo foco

Figura 4.4.1.2: Foco del comedor

Fuente: Los Autores

SALA 2 que se encuentra junto a la SALA 1, la cual es alumbrada por un solo foco

Figura 4.4.1.2: Segundo foco de la sala

Fuente: Los Autores

COCINA es el área que junto al área del COMEDOR, la cual es alumbrada por un solo foco

Figura 4.4.1.2: Foco de la cocina

Fuente: Los Autores

HALL es el área que se encuentra al lado izquierdo de la entrada principal de la vivienda, la cual es alumbrada por dos focos que se manejan con el mismo interruptor y GARAJE que pertenece al área del lado derecho de la entrada principal, es alumbrada por un solo foco y es en donde se guarda el vehículo.

4.4.1.3. Vigilancia (Cámaras)

En la parte de vigilancia se obtuvo un eficiente resultado ya que por medio de botones se realiza la apertura de la visualización de cada una de las cámaras en la cual permitió una amplia visibilidad de las áreas de mayor posibilidad de movimientos en la vivienda.

La primera área está conformada por SALA 1, SALA2 y parte del COMEDOR

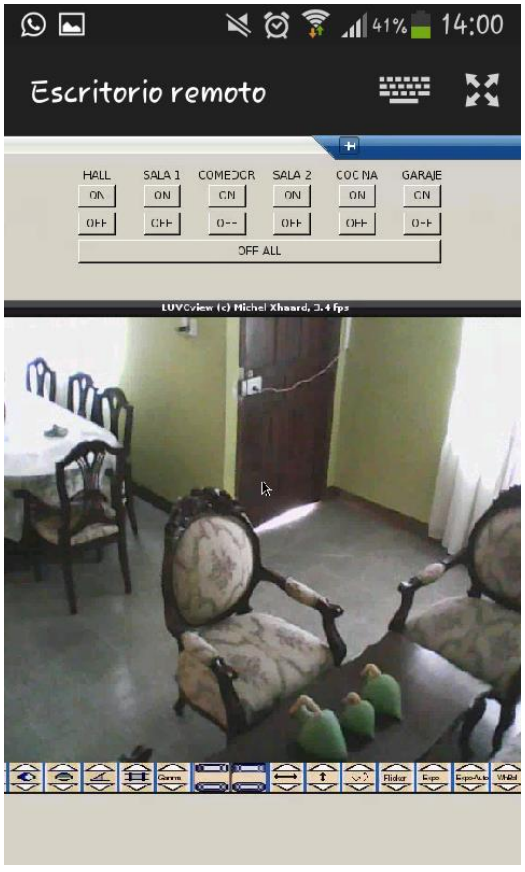


Figura 4.4.1.3: Cobertura de la cámara dirigida a la entrada

Fuente: Los Autores

La segunda área está conformada por el COMEDOR y la entrada de la COCINA

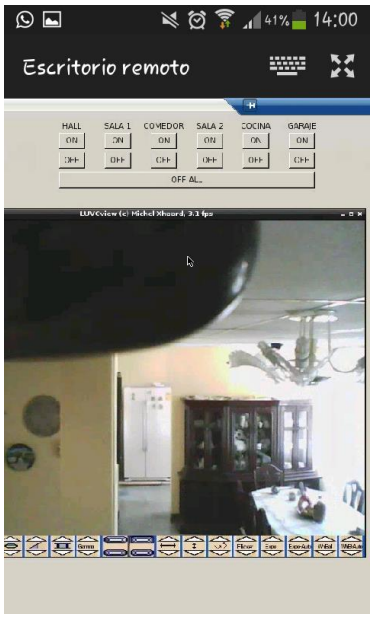


Figura 4.4.1.3: Cobertura de la cámara dirigida a la cocina

Fuente: Los Autores

La última área está conformada por una sección donde existe la activación de un sensor de movimiento y que esta antes de ingresar a los dormitorios.

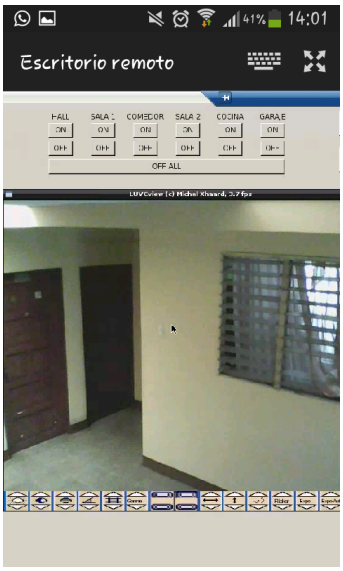


Figura 4.4.1.3: Áreas de la cobertura de la cámara dirigida a los cuartos

Fuente: Los Autores

4.4.2. Acceso Biométrico

En la puerta secundaria para el ingreso de la vivienda se colocó el dispositivo biométrico con los cables al interior de la misma, de tal manera que no fueran visibles para ningún tipo de manipulación.

Figura 4.4.2: Dispositivo Biométrico

Fuente: Los Autores

CONCLUSIONES

Al terminar este trabajo de titulación se obtuvo una gran satisfacción al realizar una investigación más a fondo sobre, identificación de huella digital lo que hoy en día se denomina tecnología biométrica.

Los conocimientos teóricos adquiridos durante la carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana han servido como base en el diseño y construcción de un Dispositivo de Acceso Biométrico y de la estructura de un sistema de seguridad con parte de domótica como es el control de luces el cual permitió tener experiencia de manera física en el manejo de interruptores eléctricos.

Los conocimientos investigados sumados a los adquiridos concluyeron principalmente en la recopilación, transmisión y manejo de datos obtenidos de una manera eficiente con el dispositivo Raspberry Pi.

Una vez teniendo los datos reunidos en el software libre Raspbian se desarrolló una aplicación con base Qt-Creator ayudado con archivos escritos en leguaje PYTHON, demuestra el control de entradas y salidas para el correcto funcionamiento del Sistema de Vigilancia.

RECOMENDACIONES

Una de las recomendaciones que los usuarios que van a manejar el sistema es que lo realicen con las debidas precauciones ya que están manejando dispositivos electrónicos delicados principalmente la Raspberry Pi, porque es la parte central del sistema ya que maneja la totalidad de los dispositivos.

Así también con el Dispositivo de Acceso Biométrico, y el motivo más importante es que posee un módulo lector de huellas, que es sensible a cualquier tipo de golpes, aún siente un dispositivo robusto debe ser tratado con la responsabilidad que el caso amerite.

CRONOGRAMA

	OCTUBRE				NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO			
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
Compra de materiales																								
Colocación de Canaletas																								
Tendido de Cableado																								
Colocación del sensor de movimiento																								
Programación de la Raspberry																								
Construcción de la tarjeta																								
Programación del Microcontrolador																								
Construcción de la caja para el Módulo Principal																								
Construcción de la caja del Dispositivo Biométrico																								
Armado del Módulo Principal																								
Armado del Dispositivo Biométrico																								
Conexión de todos los dispositivos del sistema																								
Pruebas del Sistema																								
Visita Técnica																								
Corrección de problemas																								
Pruebas del Sistema																								

PRESUPUESTO

AREA	EQUIPO	CANT.	P. U.	P. T.
PROCESAMIENTO	Kit (Pantalla touch 7", Raspberry PI, Módulo de Relé)	1	$200.00	$200.00
	Hub USB 7 Puertos alimentados	2	$15.00	$30.00
CONTROL CON DISPOSITIVOS DE ACCESO BIOMETRICO	PIC16f887	2	$7.00	$14.00
	teclado matricial 4x3	1	$7.00	$7.00
	LCD 16*2	1	$17.00	$17.00
	lector de huellas	1	$75.00	$75.00
	Tarjeta Soldada	1	$37.00	$37.00
ENTRADAS Y SALIDAS	Sensor de pared	2	$15.00	$30.00
	Cámaras	3	$112.00	$336.00
ESTÉTICA	Cajetín	1	$40.00	$40.00
	Rollo Cable UTP 50m	1	$30.00	$30.00
	Canaletas Paquete 5u	1	$40.00	$40.00
	Rollo Cable AWG #12	1	$50.00	$50.00
TRANSPORTE	Pagos	1	$100.00	$100.00
INSTRUCTIVO	Impresiones y anillados	1	$44,00	$44,00
TOTAL A PAGAR: $1,050 => $1100			T. COSTOS EQUIPOS	$1,050.00

REFERENCIAS

Cuatrorios. (02 de Marzo de 2015). Catrorios Tecnología. Obtenido de Principales Características del Lenguaje Python: http://www.cuatrorios.org/index.php?option=com_content&view=article&id=161:principales-caracteristicas-del-lenguaje-python&catid=39:blogsfeeds

Gabriela. (18 de Julio de 2013). Introducción a los relés. Obtenido de http://www.inventable.eu/introduccion-a-los-reles/

MikroElektronika. (2015). MikroElektronika. Obtenido de http://www.mikroe.com/chapters/view/81/capitulo-3-microcontrolador-pic16f887/

Qt-project. (22 de Enero de 2014). Qt-project. Obtenido de http://qt-project.org/wiki/Category:Tools::QtCreator_Spanish

Raspbian. (s.f.). Welcome to Raspbian. Obtenido de http://www.raspbian.org/

Rossum, G. v. (2009). Python. (J. Fred L. Drake, Ed.) Obtenido de http://python.org.ar/pyar/Tutorial

Sorg, J. (11 de noviembre de 2013). xrdp. Obtenido de http://www.xrdp.org/

Telégrafo, E. (06 de Febrero de 2015). La violencia en Los Esteros. Obtenido de http://www.telegrafo.com.ec/justicia/item/la-violencia-va-a-la-baja-en-los-esteros.html

UBUNTU_FACIL. (11 de Febrero de 2014). INSTALACIÓN Y CONFIGURACIÓN DE RASPBIAN. Obtenido de http://www.ubuntufacil.com/2014/02/primeros-pasos-con-raspberry-pi-instalacion-y-configuracion-de-raspbian/

Wiley, J. (2012). Raspberry Pi User Guide.

Xataka. (2012).Obtenido de http://www.xataka.com/gadgets/altavoces/raspberry-pi

ANEXOS

Anexo 1: Estado de la casa antes de iniciar (área 1)
Fuente: Los Autores

Anexo 2: Estado de la casa antes de iniciar (área 2)
Fuente: Los Autores

Anexo 3: Estado de la casa antes de iniciar (área 3)
Fuente: Los Autores

Anexo 4: Estado de la casa antes de iniciar (área 4)
Fuente: Los Autores

Anexo 5: Taladrado de pared para cable de cámara
Fuente: Los Autores

Anexo 6: Tendido de cables de para control de luces
Fuente: Los Autores

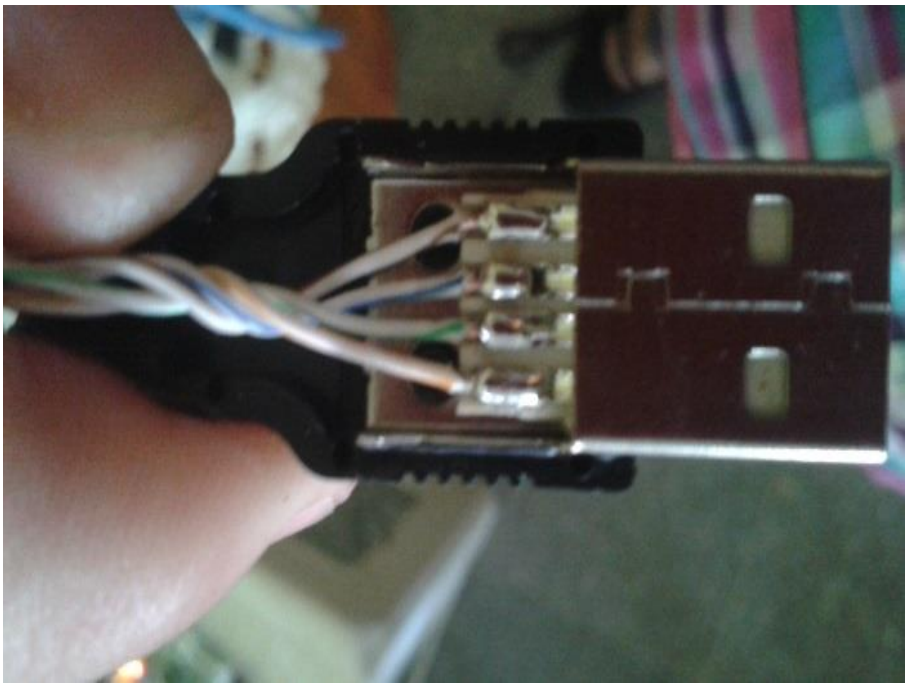
Anexo 7: Soldadura de Conectores USB macho
Fuente: Los Autores

Anexo 8: Orificio inferior para colocación de canaleta
Fuente: Los Autores

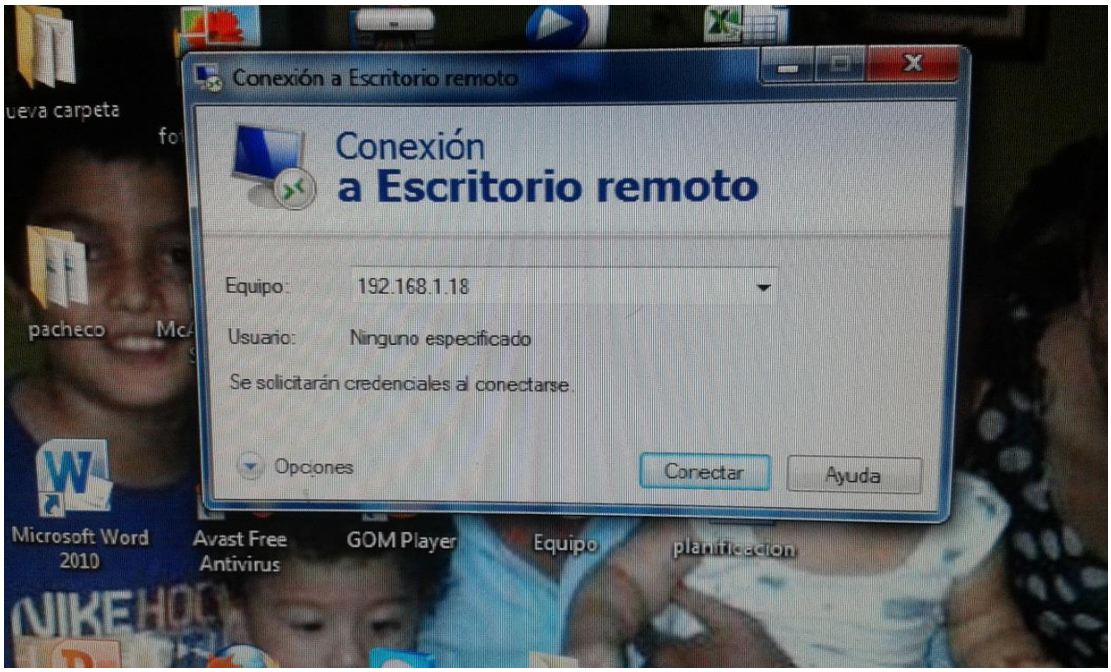
Anexo 9: Orificio superior para colocación de canaleta
Fuente: Los Autores

Anexo 10: Conectores USB machos listos
Fuente: Los Autores

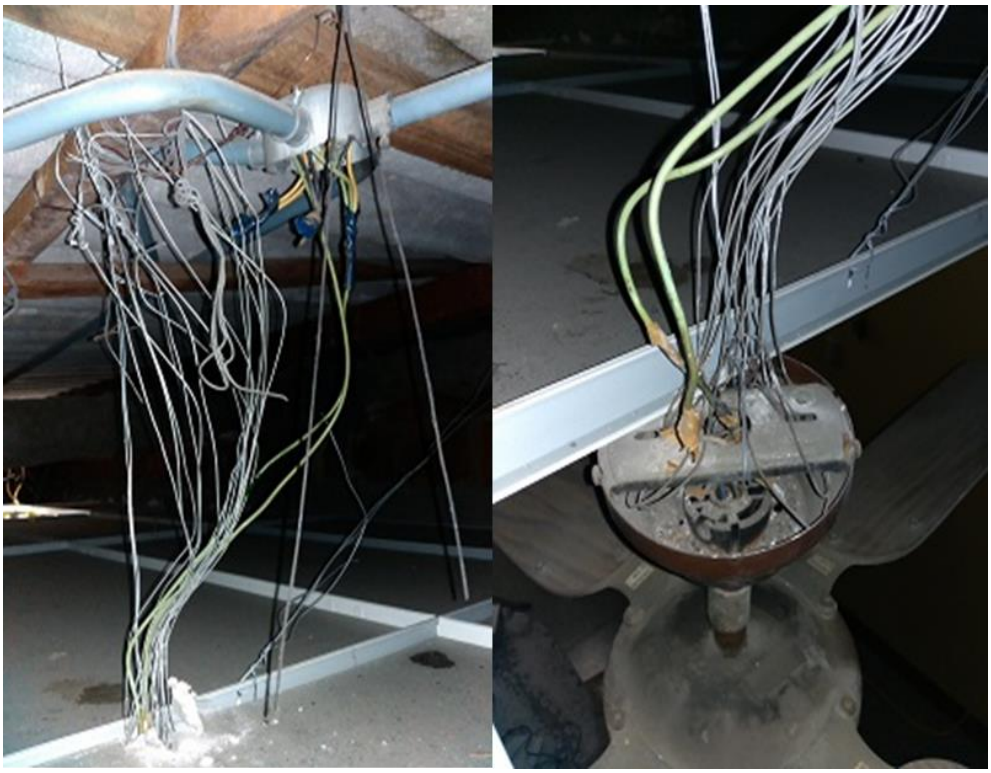
Anexo 11: Codificación de colores para polaridad de extensión
Fuente: Los Autores



Anexo 12: Cobertor del Conector USB macho
Fuente: Los Autores



Anexo 13: Conexión Remota a la Raspberry Pi
Fuente: Los Autores



Anexo 14: Estado Anterior de las conexiones eléctricas
Fuente: Los Autores

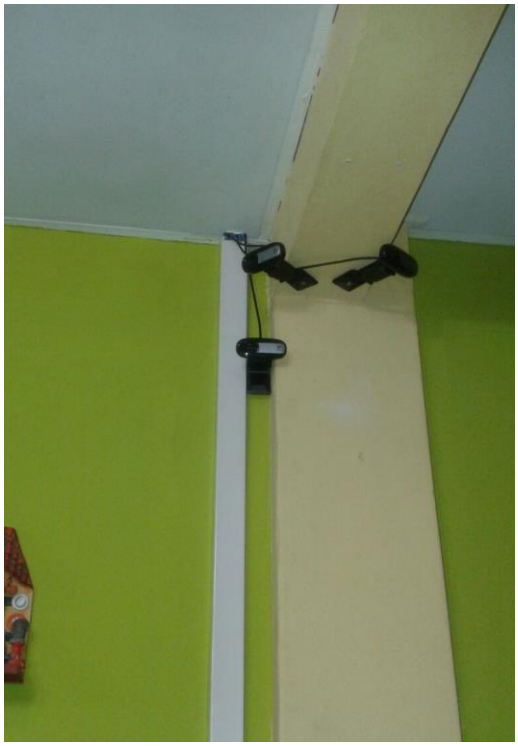
Anexo 15: Estado Actual de las conexiones eléctricas
Fuente: Los Autores

Anexo 16: Tendido Eléctrico Focos-Sala
Fuente: Los Autores

Anexo 17: Tendido Eléctrico Focos-Cocina
Fuente: Los Autores

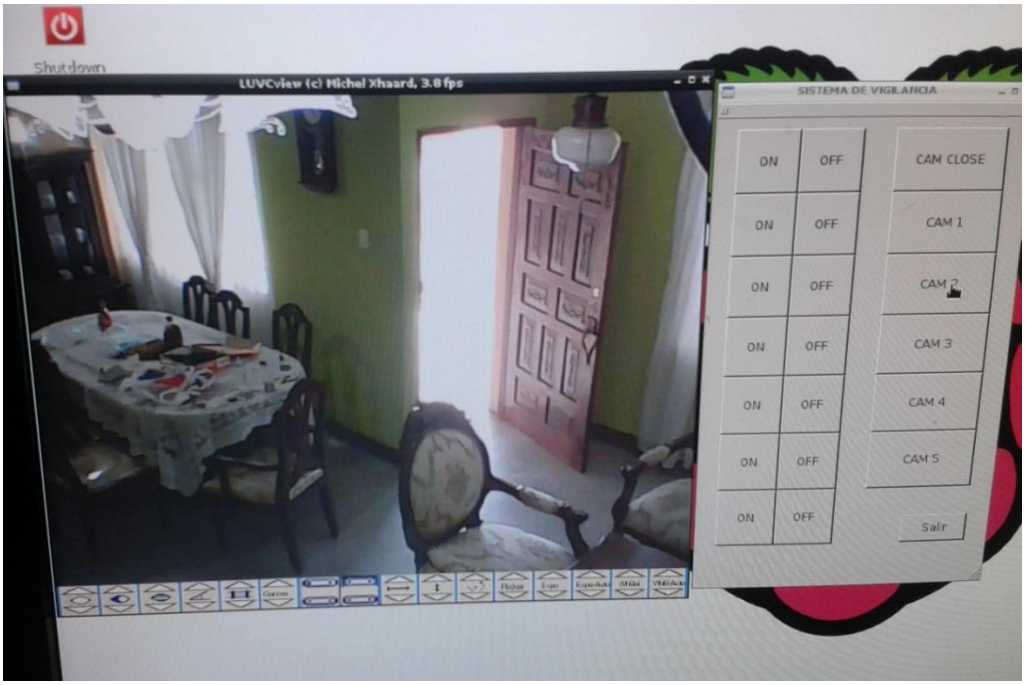
Anexo 18: Limpieza Final de Tumbado
Fuente: Los Autores

Anexo 19: Prueba Final Cámaras y Control de Luces
Fuente: Los Autores



Anexo 20: Ubicación Final de las cámaras
Fuente: Los Autores

Anexo 21: Diseño de Módulo Principal
Fuente: Los Autores



Anexo 22: Monitoreo Remoto
Fuente: Los Autores

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QProcess>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
   void on_btncam1_clicked();
   void on_btncam2_clicked();
   void on_btncam3_clicked();
   void on_btncamclose_clicked();
   void on_btnon1_clicked();
   void on_btnon2_clicked();
   void on_btnon3_clicked();
   void on_btnon4_clicked();
   void on_btnon5_clicked();
   void on_btnon6_clicked();
   void on_btnoff1_clicked();
   void on_btnoff2_clicked();
   void on_btnoff3_clicked();
   void on_btnoff4_clicked();
   void on_btnoff5_clicked();
   void on_btnoff6_clicked();
   void on_btnoffall_clicked();
   void on_btnSTART_clicked();
   void on_btnSTOP_clicked();
   void on_pushButton_clicked();

private:
    Ui::MainWindow *ui;
    QProcess *proceso1, *proceso2, *proceso3, *on1, *on2, *on3,
*on4, *on5, *on6, *off1, *off2, *off3, *off4, *off5, *off6,
*sgrabar, *pgrabar;
};

#endif // MAINWINDOW_H
```

Anexo 23: Programación en QtCreator - mainwindows.h
Fuente: Los Autores

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Anexo 24: Programación en QtCreator - main.cpp
Fuente: Los Autores

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{

    ui->setupUi(this);
    this->setWindowState(Qt::WindowMaximized);
    proceso1 = new QProcess(this);
    proceso2 = new QProcess(this);
    proceso3 = new QProcess(this);
    on1 = new QProcess(this);
    on2 = new QProcess(this);
    on3 = new QProcess(this);
    on4 = new QProcess(this);
    on5 = new QProcess(this);
    on6 = new QProcess(this);
    off1 = new QProcess(this);
    off2 = new QProcess(this);
    off3 = new QProcess(this);
    off4 = new QProcess(this);
    off5 = new QProcess(this);
    off6 = new QProcess(this);
    sgrabar = new QProcess(this);
    pgrabar = new QProcess(this);

}


MainWindow::~MainWindow()
{

    delete ui;

}
```

```
void MainWindow::on_btncam1_clicked()
{
    /*ui->label->setText(x);
    ui->label->adjustSize();*/
    proceso1->start("luvcview -d /dev/video0");
    proceso2->close();
    proceso3->close();
    ui->btncam1->setEnabled(true);
    ui->btncam2->setEnabled(true);
    ui->btncam3->setEnabled(true);
    ui->btnSTART->setEnabled(false);
    ui->btnSTOP->setEnabled(false);
    ui->btncamclose->setEnabled(true);
}


void MainWindow::on_btncam2_clicked()
{
    proceso1->close();
    proceso2->start("luvcview -d /dev/video1");
    proceso3->close();
    ui->btncam1->setEnabled(true);
    ui->btncam2->setEnabled(true);
    ui->btncam3->setEnabled(true);
    ui->btnSTART->setEnabled(false);
    ui->btnSTOP->setEnabled(false);
    ui->btncamclose->setEnabled(true);
}


void MainWindow::on_btncam3_clicked()
{
    proceso1->close();
    proceso2->close();
    proceso3->start("luvcview -d /dev/video2");
    ui->btncam1->setEnabled(true);
    ui->btncam2->setEnabled(true);
    ui->btncam3->setEnabled(true);
    ui->btnSTART->setEnabled(false);
    ui->btnSTOP->setEnabled(false);
    ui->btncamclose->setEnabled(true);
}


void MainWindow::on_btncamclose_clicked()
{
    proceso1->close();
    proceso2->close();
    proceso3->close();
    ui->btncam1->setEnabled(true);
    ui->btncam2->setEnabled(true);
    ui->btncam3->setEnabled(true);
    ui->btnSTART->setEnabled(true);
    ui->btnSTOP->setEnabled(false);
    ui->btncamclose->setEnabled(false);
}
```

```
void MainWindow::on_btnon1_clicked()
{
    on1->start("sudo python /home/pi/EYES_ONE_CLIC/LF0.py");
}

void MainWindow::on_btnon2_clicked()
{
    on2->start("sudo python /home/pi/EYES_ONE_CLIC/LF1.py");
}


void MainWindow::on_btnon3_clicked()
{
    on3->start("sudo python /home/pi/EYES_ONE_CLIC/LF2.py");
}


void MainWindow::on_btnon4_clicked()
{
    on4->start("sudo python /home/pi/EYES_ONE_CLIC/LF3.py");
}


void MainWindow::on_btnon5_clicked()
{
    on5->start("sudo python /home/pi/EYES_ONE_CLIC/LF4.py");
}


void MainWindow::on_btnon6_clicked()
{
    on6->start("sudo python /home/pi/EYES_ONE_CLIC/LF5.py");
}


void MainWindow::on_btnoff1_clicked()
{
    off1->start("sudo python /home/pi/EYES_ONE_CLIC/HF0.py");
}


void MainWindow::on_btnoff2_clicked()
{
    off2->start("sudo python /home/pi/EYES_ONE_CLIC/HF1.py");
}


void MainWindow::on_btnoff3_clicked()
{
    off3->start("sudo python /home/pi/EYES_ONE_CLIC/HF2.py");
}

void MainWindow::on_btnoff4_clicked()
{
    off4->start("sudo python /home/pi/EYES_ONE_CLIC/HF3.py");
}
```

```
void MainWindow::on_btnoff5_clicked()
{
    off5->start("sudo python /home/pi/EYES_ONE_CLIC/HF4.py");
}



void MainWindow::on_btnoff6_clicked()
{
    off6->start("sudo python /home/pi/EYES_ONE_CLIC/HF5.py");
}



void MainWindow::on_btnoffall_clicked()
{

    off1->start("sudo python /home/pi/EYES_ONE_CLIC/HF0.py");
    off2->start("sudo python /home/pi/EYES_ONE_CLIC/HF1.py");
    off3->start("sudo python /home/pi/EYES_ONE_CLIC/HF2.py");
    off4->start("sudo python /home/pi/EYES_ONE_CLIC/HF3.py");
    off5->start("sudo python /home/pi/EYES_ONE_CLIC/HF4.py");
    off6->start("sudo python /home/pi/EYES_ONE_CLIC/HF5.py");

}



void MainWindow::on_btnSTART_clicked()
{

        sgrabar->start("sudo motion");
        ui->btncam1->setEnabled(false);
        ui->btncam2->setEnabled(false);
        ui->btncam3->setEnabled(false);
        ui->btnSTART->setEnabled(false);
        ui->btnSTOP->setEnabled(true);

}



void MainWindow::on_btnSTOP_clicked()
{

    pgrabar->start("sudo pkill motion");
    ui->btncam1->setEnabled(true);
    ui->btncam2->setEnabled(true);
    ui->btncam3->setEnabled(true);
    ui->btnSTART->setEnabled(true);
    ui->btnSTOP->setEnabled(false);

}
```

```
void MainWindow::on_pushButton_clicked()
{
    pgrabar->start("sudo pkill motion");
    off1->start("sudo python /home/pi/EYES_ONE_CLIC/HF0.py");
    off2->start("sudo python /home/pi/EYES_ONE_CLIC/HF1.py");
    off3->start("sudo python /home/pi/EYES_ONE_CLIC/HF2.py");
    off4->start("sudo python /home/pi/EYES_ONE_CLIC/HF3.py");
    off5->start("sudo python /home/pi/EYES_ONE_CLIC/HF4.py");
    off6->start("sudo python /home/pi/EYES_ONE_CLIC/HF5.py");
    proceso1->close();
    proceso2->close();
    proceso3->close();
}
```

Anexo 25: Programación en QtCreator - mainwindow.cpp
Fuente: Los Autores

```
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>764</width>
<height>739</height>
</rect>
</property>
<property name="windowTitle">
<string>MainWindow</string>
</property>
<widget class="QWidget" name="centralWidget">
<widget class="QPushButton" name="btncam1">
<property name="geometry">
<rect>
<x>310</x>
<y>400</y>
<width>50</width>
<height>30</height>
</rect>
</property>
<property name="text">
<string>CAM 1</string>
</property>
</widget>
<widget class="QPushButton" name="btncam2">
<property name="geometry">
<rect>
<x>360</x>
<y>400</y>
<width>50</width>
<height>30</height>
</rect>
```

```
</property>
<property name="text">
<string>CAM 2</string>
</property>
</widget>
<widget class="QPushButton" name="btncam3">
<property name="geometry">
<rect>
<x>410</x>
<y>400</y>
<width>50</width>
<height>30</height>
</rect>
</property>
<property name="text">
<string>CAM 3</string>
</property>
</widget>
<widget class="QPushButton" name="btnon5">
<property name="geometry">
<rect>
<x>470</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btnon2">
<property name="geometry">
<rect>
<x>260</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btnon4">
<property name="geometry">
<rect>
<x>400</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btnon1">
<property name="geometry">
<rect>
```

```
<x>190</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btnon3">
<property name="geometry">
<rect>
<x>330</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btnoff5">
<property name="geometry">
<rect>
<x>470</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
</property>
</widget>
<widget class="QPushButton" name="btnoff2">
<property name="geometry">
<rect>
<x>260</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
</property>
</widget>
<widget class="QPushButton" name="btnoff4">
<property name="geometry">
<rect>
<x>400</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
```

```
</property>
</widget>
<widget class="QPushButton" name="btnoff1">
<property name="geometry">
<rect>
<x>190</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
</property>
</widget>
<widget class="QPushButton" name="btnoff3">
<property name="geometry">
<rect>
<x>330</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
</property>
</widget>
<widget class="QPushButton" name="btnoff6">
<property name="geometry">
<rect>
<x>540</x>
<y>50</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF</string>
</property>
</widget>
<widget class="QPushButton" name="btnon6">
<property name="geometry">
<rect>
<x>540</x>
<y>20</y>
<width>40</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>ON</string>
</property>
</widget>
<widget class="QPushButton" name="btncamclose">
<property name="enabled">
<bool>false</bool>
</property>
<property name="geometry">
<rect>
```

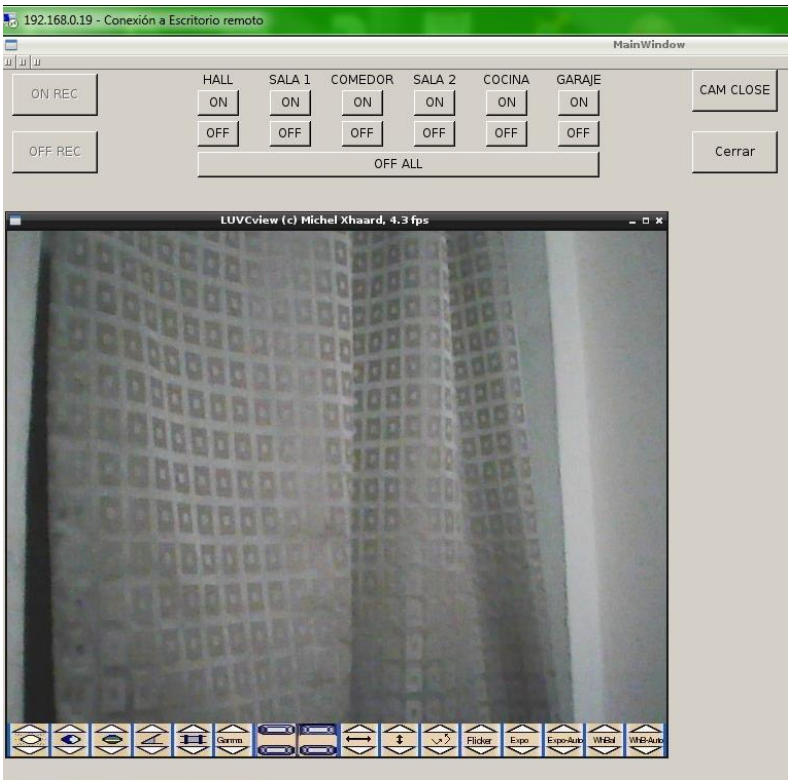
```
<x>670</x>
<y>0</y>
<width>83</width>
<height>41</height>
</rect>
</property>
<property name="text">
<string>CAM CLOSE</string>
</property>
</widget>
<widget class="QPushButton" name="btnSTART">
<property name="geometry">
<rect>
<x>10</x>
<y>4</y>
<width>83</width>
<height>41</height>
</rect>
</property>
<property name="text">
<string>ON REC</string>
</property>
</widget>
<widget class="QPushButton" name="btnSTOP">
<property name="enabled">
<bool>false</bool>
</property>
<property name="geometry">
<rect>
<x>10</x>
<y>60</y>
<width>83</width>
<height>41</height>
</rect>
</property>
<property name="text">
<string>OFF REC</string>
</property>
</widget>
<widget class="QPushButton" name="btnoffall">
<property name="geometry">
<rect>
<x>190</x>
<y>80</y>
<width>390</width>
<height>25</height>
</rect>
</property>
<property name="text">
<string>OFF ALL</string>
</property>
</widget>
<widget class="QLabel" name="label">
<property name="geometry">
<rect>
<x>180</x>
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
```

```
</property>
<property name="text">
<string>HALL</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QLabel" name="label_2">
<property name="geometry">
<rect>
<x>250</x>
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
</property>
<property name="text">
<string>SALA 1</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QLabel" name="label_3">
<property name="geometry">
<rect>
<x>320</x>
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
</property>
<property name="text">
<string>COMEDOR</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QLabel" name="label_4">
<property name="geometry">
<rect>
<x>390</x>
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
</property>
<property name="text">
<string>SALA 2</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QLabel" name="label_5">
<property name="geometry">
<rect>
<x>460</x>
```

```
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
</property>
<property name="text">
<string>COCINA</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QLabel" name="label_6">
<property name="geometry">
<rect>
<x>530</x>
<y>0</y>
<width>60</width>
<height>20</height>
</rect>
</property>
<property name="text">
<string>GARAJE</string>
</property>
<property name="alignment">
<set>Qt::AlignCenter</set>
</property>
</widget>
<widget class="QPushButton" name="pushButton">
<property name="geometry">
<rect>
<x>670</x>
<y>60</y>
<width>83</width>
<height>41</height>
</rect>
</property>
<property name="text">
<string>Cerrar</string>
</property>
</widget>
</widget>
<widget class="QMenuBar" name="menuBar">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>764</width>
<height>22</height>
</rect>
</property>
</widget>
<widget class="QToolBar" name="mainToolBar">
<attribute name="toolBarArea">
<enum>TopToolBarArea</enum>
</attribute>
<attribute name="toolBarBreak">
<bool>false</bool>
</attribute>
</widget>
```

```
<widget class="QStatusBar" name="statusBar"/>
<widget class="QToolBar" name="toolBar">
<property name="windowTitle">
<string>toolBar</string>
</property>
<attribute name="toolBarArea">
<enum>TopToolBarArea</enum>
</attribute>
<attribute name="toolBarBreak">
<bool>false</bool>
</attribute>
</widget>
<widget class="QToolBar" name="toolBar_2">
<property name="windowTitle">
<string>toolBar_2</string>
</property>
<attribute name="toolBarArea">
<enum>TopToolBarArea</enum>
</attribute>
<attribute name="toolBarBreak">
<bool>false</bool>
</attribute>
</widget>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections>
<connection>
<sender>pushButton</sender>
<signal>clicked()</signal>
<receiver>MainWindow</receiver>
<slot>close()</slot>
<hints>
<hint type="sourcelabel">
<x>704</x>
<y>110</y>
</hint>
<hint type="destinationlabel">
<x>606</x>
<y>104</y>
</hint>
</hints>
</connection>
</connections>
</ui>
```

Anexo 26: Programación en QtCreator - mainwindow.ui
Fuente: Los Autores



Anexo 27: Programación en QtCreator - Interface de control desarrollado en QT
Fuente: Los Autores

```
#!/usr/bin/python
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
GPIO.setwarnings(False)
GPIO.setup(##,GPIO.OUT)
print "Pin ## H"
GPIO.output(##,GPIO.HIGH)
```

Anexo 28: Programación en Python - Para enviar 5v por el pin ## y activar el relé conectado en el mismo y permitir que un foco se encienda
Fuente: Los Autores

```
#!/usr/bin/python
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
GPIO.setwarnings(False)
GPIO.setup(##,GPIO.OUT)
print "Pin ## L"
GPIO.output(##,GPIO.LOW)
```

Anexo 29: Programación en Python - Para enviar 0v por el pin ## y desactivar el relé conectado en el mismo y permitir que un foco se apague
Fuente: Los Autores

```
#!/usr/bin/python2.7

import smtplib
from datetime import datetime

import os.path
import sys
import base64

import gdata.data
import gdata.docs.data
import gdata.docs.client
import ConfigParser

class MotionUploader:
    def __init__(self):
        # Se carga la configuracion de conexion

        # credenciales deGMail
        self.username = "tu_correo_gmail@gmail.com"
        self.password = "tu_clave_gmail"
        self.from_name = "Tu Nombre completo"
        self.sender = "tu_correo_gmail@gmail.com"

        self.recipient = "tu_correo_gmail@gmail.com"
        self.subject = "mensaje de movimiento detectado"
        self.message = "Detector de movimiento"

        self.delete_after_upload = "false"
        self.send_email = "true"

        self._create_gdata_client()



    def _send_email(self,msg,imgpath):
        '''Send an email using the GMail account.'''
        senddate=datetime.strftime(datetime.now(), '%Y-%m-%d')
        marker = "MARCADOR001001"
        p1="Date: %s\r\nFrom: %s <%s>\r\nTo: %s\r\nSubject:
%s\r\nContent-Type: multipart/mixed; boundary=%s\r\n--%s\r\n" %
(senddate, self.from_name, self.sender, self.recipient,
self.subject, marker, marker)
        p2="Content-Type: text/plain\r\nContent-Transfer-
Encoding:8bit\r\n\r\n%s\r\n--%s\r\n" % (msg, marker)

        server = smtplib.SMTP('smtp.gmail.com:587')
        server.starttls()
        server.login(self.username, self.password)
        server.sendmail(self.sender, self.recipient, p1+p2)
        server.quit()
    def _upload(self, video_file_path, folder_resource):
        '''Upload the video and return the doc'''
        doc = gdata.docs.data.Resource(type='image',
title=os.path.basename(video_file_path))
        media = gdata.data.MediaSource()
        media.SetFileHandle(video_file_path, 'image/jpeg')
        doc = self.client.CreateResource(doc, media=media,
collection=folder_resource)
```

```
        return doc

    def upload_video(self, video_file_path):
        """Upload a video to the specified folder. Then optionally
send an email and optionally delete the local file."""
        folder_resource = self._get_folder_resource()
        if not folder_resource:
            raise Exception('Could not find the %s folder' %
self.folder)
        doc = self._upload(video_file_path, folder_resource)
        if self.send_email:
            video link = None
            for link in doc.link:
                if 'video.google.com' in link.href:
                    video_link = link.href
                    break
            # Send an email with the link if found
            msg = self.message
            if video_link:
                msg += '\n\n' + video_link
            imgfile = os.path.splitext(video_file_path)[0] +
".jpg"
            self._send_email(msg,imgfile)
            if self.delete_after_upload:
               os.remove(imgfile)
        if self.delete_after_upload:
            os.remove(video_file_path)
    def envia_email(self):
        print('Enviando correo')
        msg="Movimiento detectado"
       self._send_email(msg,"")
if __name__ == '__main__':
    try:
        MotionUploader().envia_email()
    except gdata.client.BadAuthentication:
        exit('Error en los datos de acceso.')
    except gdata.client.Error:
        exit('Error de Login')
    except Exception as e:
        exit('Error: [%s]' % e)
```

Anexo 30: Programación en Python - Archivo para enviar email, activado desde motion.conf cuando se detecta movimiento

Fuente: Los Autores

```
[gmail]
# GMail account credentials
name = My Name
user = gmailusername
password = gmailpassword
sender = me@gmail.com
# Recipient email address (could be same as from_addr)
recipient = me@gmail.com
# Subject line for email
subject = Motion detected
```

```
# First line of email message
message = Video uploaded
[docs]
# Folder (or collection) in Docs where you want the videos to
go
folder = motion
[options]
# Delete the local video file after the upload
delete-after-upload = true
# Send an email after the upload
send-email = true
```

Anexo 31: Programación en Python - Archivo.cfg de los parámetros que necesita el archivo encargado de subir las fotos al google drive
Fuente: Los Autores

```
 #!/usr/bin/python2
import smtplib
from datetime import datetime
import os.path
import sys
import gdata.data
import gdata.docs.data
import gdata.docs.client
import ConfigParser
class MotionUploader:

    def __init__(self, config_file_path):
        # Load config
        config = ConfigParser.ConfigParser()
        config.read(config_file_path)
        # GMail account credentials
        self.username = config.get('gmail', 'user')
        self.password = config.get('gmail', 'password')
        self.from_name = config.get('gmail', 'name')
        self.sender = config.get('gmail', 'sender')
        # Recipient email address (could be same as from_addr)
        self.recipient = config.get('gmail', 'recipient')
        # Subject line for email
        self.subject = config.get('gmail', 'subject')
        # First line of email message
        self.message = config.get('gmail', 'message')
        # Folder (or collection) in Docs where you want the videos
to go
        self.folder = config.get('docs', 'folder')
        # Options
        self.delete_after_upload = config.getboolean('options',
'delete-after-upload')
        self.send_email = config.getboolean('options', 'send-
email')
        self._create_gdata_client()
    def _create_gdata_client(self):
        """Create a Documents List Client."""
        self.client =
gdata.docs.client.DocsClient(source='motion_uploader')
```

```
        self.client.http_client.debug = False
        self.client.client_login(self.username, self.password,
service=self.client.auth_service, source=self.client.source)
    def _get_folder_resource(self):
        """Find and return the resource whose title matches the
given folder."""
        col = None
        for resource in
self.client.GetAllResources(uri='/feeds/default/private/full/-
/folder'):

            if resource.title.text == self.folder:
                col = resource
                break
        return col
    def _send_email(self,msg):

        '''Send an email using the GMail account.'''
        senddate=datetime.strftime(datetime.now(), '%Y-%m-%d')
        m="Date: %s\r\nFrom: %s <%s>\r\nTo: %s\r\nSubject:

%s\r\nX-Mailer: My-Mail\r\n\r\n" % (senddate, self.from_name,
self.sender, self.recipient, self.subject)

        server = smtplib.SMTP('smtp.gmail.com:587')
        server.starttls()
        server.login(self.username, self.password)
        server.sendmail(self.sender, self.recipient, m+msg)
        server.quit()
    def _upload(self, video_file_path, folder_resource):
        '''Upload the video and return the doc'''
        doc = gdata.docs.data.Resource(type='video',
title=os.path.basename(video_file_path))

        media = gdata.data.MediaSource()
        media.SetFileHandle(video_file_path, 'video/avi')
        doc = self.client.CreateResource(doc, media=media,
collection=folder_resource)

        return doc
    def upload_video(self, video_file_path):

        """Upload a video to the specified folder. Then optionally
send an email and optionally delete the local file."""
        folder_resource = self._get_folder_resource()

 if not folder_resource:
            raise Exception('Could not find the %s folder' %
self.folder)
        doc = self._upload(video_file_path, folder_resource)

if self.send_email:
            video_link = None
            for link in doc.link:
                if 'video.google.com' in link.href:
                    video_link = link.href
                    break
            # Send an email with the link if found
            msg = self.message
            if video_link:
```

```
                msg += '\n\n' + video_link
            self._send_email(msg)

if self.delete_after_upload:
            os.remove(video_file_path)

if __name__ == '__main__':
    try:

        if len(sys.argv) < 3:
            exit('Motion Uploader - uploads videos to Google
Drive\n
Usage: uploader.py {config-file-path} {video-file-path}')
        cfg_path = sys.argv[1]
        vid_path = sys.argv[2]

if not os.path.exists(cfg_path):
            exit('Config file does not exist [%s]' % cfg_path)

if not os.path.exists(vid_path):
            exit('Video file does not exist [%s]' % vid_path)
        MotionUploader(cfg_path).upload_video(vid_path)
    except gdata.client.BadAuthentication:
        exit('Invalid user credentials given.')
    except gdata.client.Error:
        exit('Login Error')
    except Exception as e:
        exit('Error: [%s]' % e)
```

Anexo 32: Programación en Python - Archivo.py encargado de subir fotos al google drive con los parámetros descritos en el archivo.cfg

Fuente: Los Autores