"El saber de mis hijos hará mi grandeza"

UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE INVESTIGACIÓN EN FÍSICA

Ingeniería en Tecnología Electrónica

CONTROL DE ACCESO Y SEGURIDAD INTELIGENTE

TESIS QUE PRESENTA:

SERGIO ALEJANDRO ROSALES NÚÑEZ

PARA OBTENER EL TÍTULO DE

INGENIERO EN TECNOLOGÍA ELECTRÓNICA

DIRECTOR DE TESIS

DR. ALEJANDRO GARCÍA JUÁREZ

Hermosillo, Sonora **Octubre 2015**

Universidad de Sonora

Repositorio Institucional UNISON

AGRADECIMIENTOS

Le doy gracias a mi mamá por haberme apoyado siempre, en cualquier situación por brindarme la oportunidad de estudiar e inculcarme los valores necesarios para poder sobresalir en la vida. También agradezco a mis hermanos por estar dispuestos siempre para brindarme apoyo.

Le agradezco la confianza y dedicación de su tiempo al Dr. Alejandro García, al igual que su apoyo que siempre estaba dispuesto a brindarme con cualquier problema que se me presentara.

ÍNDICE

Capítulo 1

1.1.- Introducción general

El ser humano siempre ha tenido el impulso de satisfacer sus necesidades básicas, esto lo ha llevado a evolucionar para poder controlar, de cierta manera, su supervivencia. Sin embargo, también han surgido necesidades que ahora son importantes satisfacer. Una de ella es la seguridad la cual, siempre ha sido un tema muy importante a tratar, tanto en la época pasada como en la actual. La seguridad típica usada en cualquier lugar y desde hace mucho tiempo es poner cerraduras de metal y si se requiere más eficiencia, se agregan cerraduras extras. En los últimos años en cuanto a investigación se refiere, no se ha aportado algo nuevo relacionado con las cerraduras a no ser variantes sobre lo ya conocido, aunque por ello no dejan de ser aportaciones importantes. La mayoría de los hogares y edificios actuales usan sistemas de seguridad muy comunes y conocidos por todas las personas, sensores magnéticos en puertas y en ciertas ocasiones en ventanas, lo que provoca que al activarse dichos sensores suene una alarma, que si el usuario no se encuentra cercas de ese lugar, no se percatará de cualquier incidente ocurrido. Otro método común que se usa es mediante una cámara de seguridad, que tiene sólo la típica función de grabar y por consecuencia no podrá avisar si hay un problema de seguridad, por lo que hoy en día no es una buena opción tener dichos sistemas que no alertan al usuario cuando éste se encuentra lejos de la propiedad.

Mientras la tecnología avanza, los sistemas de seguridad se vuelven más sofisticados, quedando atrás todo lo relacionado con la manipulación humana y se crean sistemas con menor costo a mayor plazo y, algunas veces, más eficientes que teniendo manipulación humana. Una computadora puede realizar funciones muy similares que el hombre, por ejemplo el reconocimiento facial, de voz, de colores, etc. Generalmente, el sistema de seguridad típico que se tiene en todas partes es el monitoreo mediante video, donde si ocurre un acontecimiento no deseado se opta por revisar la grabación a la hora deseada y ver lo que se grabó. También se puede dar el caso que la persona encargada de monitorear

la grabación en tiempo real se percate de algún acontecimiento no deseado y se actúe con las medidas necesarias. Con la ayuda de la tecnología, las grabaciones se pueden monitorear vía internet, gracias a los nuevos sistemas de grabación que incluyen conexión a internet con la capacidad de servidor y por lo tanto, mientras el sistema esté conectado a internet, se pueda acceder desde cualquier lugar y hora.

Un aspecto fundamental en la creación de ciertos sistemas inteligentes es detectar la presencia humana. El tópico interacción hombre-máquina ha comenzado a atraer una gran atención en la última década. El objetivo es poder crear interfaces más inteligentes capaces de extraer información sobre el contexto o las acciones a realizar mediante una interacción natural con el usuario. En la actualidad gracias a la tecnología, se puede crear un sistema que sea capaz de controlar el acceso y monitorear un lugar mientras el usuario permanezca fuera del mismo. Con la creación de la visión artificial, se ha dado un gran avance al procesamiento de imágenes digitales y debido a lo anterior, se ha llegado al punto que pueda existir un monitoreo que pueda ser capaz de detectar ciertas anomalías que nosotros le programaremos.

1.2.- Objetivo

En esta tesis se pretende crear un sistema de seguridad inteligente enfocado a las viviendas y edificios empresariales. Dicho sistema abarcará desde el ingreso a los edificios hasta el intento de intrusión no deseado, el cual incluirá vigilancia en tiempo real, sensores en ventanas, puertas e incluso tendrá sensores hasta en los muros que conforman la habitación, los cuales se activarán si hay un golpe muy fuerte. Lo que hace inteligente al sistema de seguridad planteado es la capacidad de enviar alertas por medio de la mensajería instantánea al usuario cuando es detectado un intruso, para que de esa manera el usuario pueda actuar a tiempo.

1.3.- Metodología

Básicamente, el sistema de seguridad funcionará de acuerdo al siguiente diagrama:

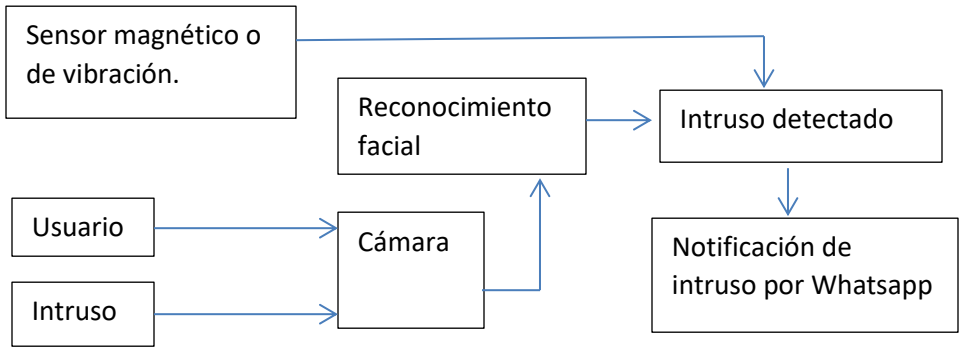

Figura 1.1. Diagrama de la estructura del sistema

Mediante una cámara y con la ayuda de los sensores magnéticos y de vibración serán los dispositivos que se encargarán de detectar las presencias humanas y con el apoyo de procesamiento computacional se podrá clasificar entre usuario e intruso para poder enviar correctamente una notificación vía Whatsapp.

1.4.- Distribución de la tesis

La propuesta de tesis está dividida en básicamente 7 secciones que describen lo siguiente:

Capítulo 1. Se da una introducción breve sobre el problema de la seguridad y se describen ciertas soluciones que puedan resolver dicho problema. Se describe el objetivo y la metodología que se usará.

Capítulo 2. Se describen aspectos importantes relacionados con la visión artificial, como lo es su definición, los procesos que se llevan a cabo, los algoritmos más importantes que hay, las herramientas que se pueden usar para su desarrollo.

Capítulo 3. Se analizan los sistemas embebidos, su diferencia con los ordenadores, la forma en que se usan, aplicaciones básicas, describe la tarjeta embebida a utilizarse y los sistemas operativos que conllevan.

Capítulo 4. Se da una introducción sobre los protocolos de comunicación y los tipos más usados. Se ejemplifican los protocolos a utilizarse y se dan características importantes de dichos protocolos.

Capítulo 5. Se describe el sistema que se usará para el control de acceso, el cual consiste en el reconocimiento de rostros y el algoritmo que se implementó, así como también los métodos existentes para realizar dicho reconocimiento.

Capítulo 6. En este capítulo se describe la configuración y programación de los puertos de entrada/salida del sistema embebido que se utilizará con el apoyo de los sensores a implementarse. También se describe la configuración de la aplicación usada para el envío de notificaciones mediante el uso de la mensajería instantánea.

Capítulo 7. Por último, en este capítulo se da la conclusión sobre el desarrollo de este trabajo y el trabajo a futuro que se pudiera implementar en el sistema para mejorarlo.

Capítulo 2

2. Visión Artificial

2.1- Introducción a la visión artificial

La visión artificial es un campo de la inteligencia artificial cuyo propósito es la adquisición de imágenes, generalmente en dos dimensiones, para luego procesarlas digitalmente mediante un sistema de CPU (computadora, microcontrolador, DSP, etc), con el fin de extraer y medir determinadas propiedades de las imágenes adquiridas de tal manera que se puedan analizar e interpretar.

La visión artificial es un campo de estudio diverso y relativamente nuevo. En los inicios de la computación era complicado procesar incluso conjuntos moderadamente grandes de datos de imagen. No fue hasta finales de los años setenta que emergió un estudio más concentrado de dicho campo. El inicio de la visión artificial, desde el punto de vista práctico, fue marcado por Larry Roberts, el cual, en 1961 creó un programa que podía "ver" una estructura de bloques, analizar su contenido y reproducirla desde otra perspectiva, demostrando así a los espectadores que esa información visual que había sido mandada al ordenador por una cámara, había sido procesada adecuadamente por él [1].

2.2.- Componentes de la visión artificial

Una aplicación de visión artificial se compone de una o varias cámaras que capturarán y digitalizarán una serie de imágenes bajo configuraciones diferentes de iluminación. Estas imágenes, una vez digitalizadas, deben ser procesadas por una computadora, donde se programarán los algoritmos de procesado digital de imagen necesarios para conseguir extraer la información buscada en la escena. En esta sección se pueden descubrir las ventajas que aporta una aplicación de visión artificial a un proceso de fabricación así como

ver ejemplos de posibles aplicaciones que pueden ser desarrolladas utilizando técnicas de visión artificial.

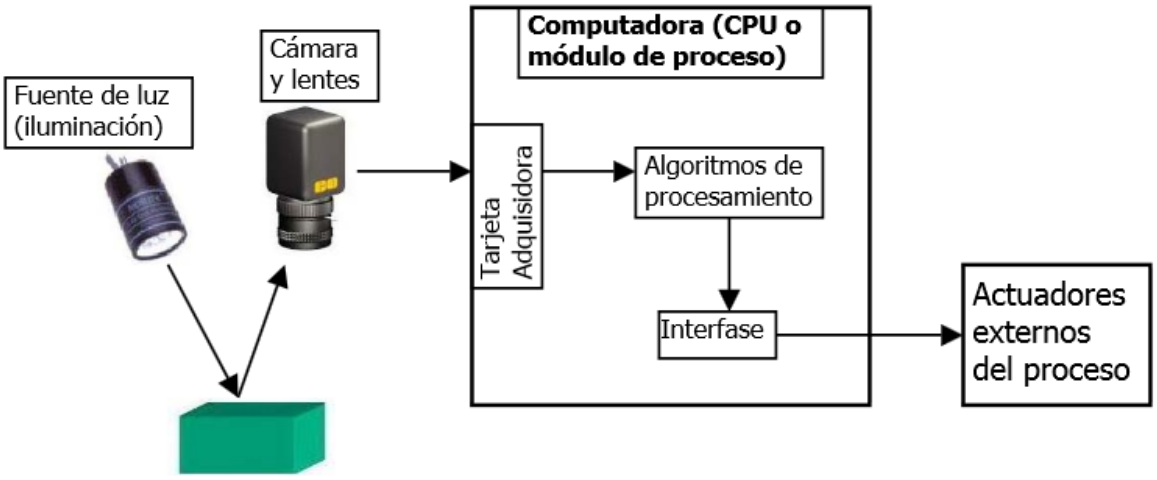

Figura 2.1. Sistema de visión artificial.

La figura 2.1 muestra claramente los componentes más básicos para un sistema de visión artificial. Con la ayuda de una fuente de luz, una cámara puede detectar cualquier objeto y enviar la información a un CPU, donde posteriormente es procesada y se relaciona con parámetros predefinidos, por ejemplo reconocer la forma del objeto, el color, el tamaño, etc. A continuación se describen algunas características que se deben de tomar en cuenta con cada componente: [2]

2.2.1.- La iluminación

Un aspecto importante que se debe tomar en cuenta cuando se realiza un sistema de visión artificial es la iluminación, ya que luz es un factor de vital importancia que afecta de forma crucial a los algoritmos de visión que se vayan a utilizar bajo esas condiciones. Una cámara no ve en sí al objeto, sino que adquiere la luz que los objetos reflejan hacia ella. La iluminación se puede considerar la parte más crítica dentro de un sistema de visión artificial. Las cámaras, de momento, son mucho menos sensibles y versátiles que la visión humana y las condiciones de iluminación se deben optimizar al máximo para que una cámara pueda capturar una imagen que el ojo humano podría distinguir sin necesidad de

una iluminación tan especializada. Los objetivos de la iluminación son: optimizar el contraste, normalizar cualquier variación de la iluminación ambiente y simplificar el proceso de tratamiento posterior de la imagen (si se utilizan filtros por software el tiempo de procesado se incrementa) [3].

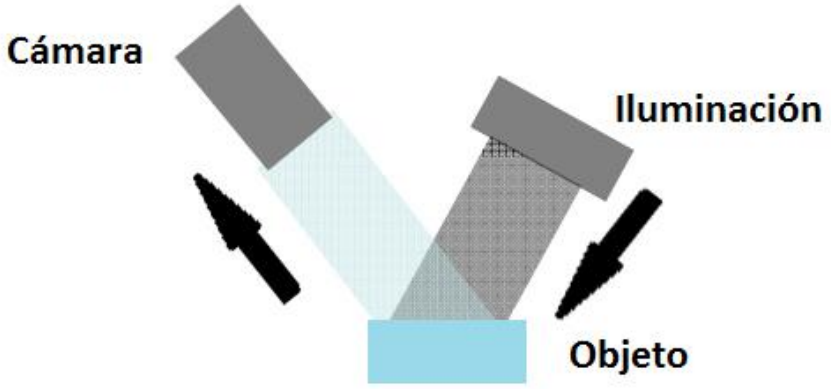

Figura 2.2. Reflejo de la luz de un objeto hacia una cámara.

2.2.2.- Cámara

Una cámara digital contiene una lente que refleja la luz a un sensor óptico para después ser procesada mediante circuitos electrónicos. En la actualidad, existen dos tipos de tecnologías utilizadas en los sensores de cámaras digitales, los CCD (Charge Coupled Device) y CMOS (Complementary Metal Oxide Semiconductor). Las dos tecnologías usan semiconductores de metal-óxido (MOS) y están distribuidos en forma de matriz.

Cada matriz contiene celdas o pixeles que almacenan cargas eléctricas. La carga eléctrica almacenada en cada píxel, dependerá en todo momento de la cantidad de luz que incida sobre el mismo. Mientras más nivel de energía luminosa incida sobre el píxel, mayor será la carga que este adquiera. Posteriormente la carga eléctrica es digitalizada mediante una conversión A/D para que pueda ser manipulada fácilmente [4].

- **Sensor CCD**

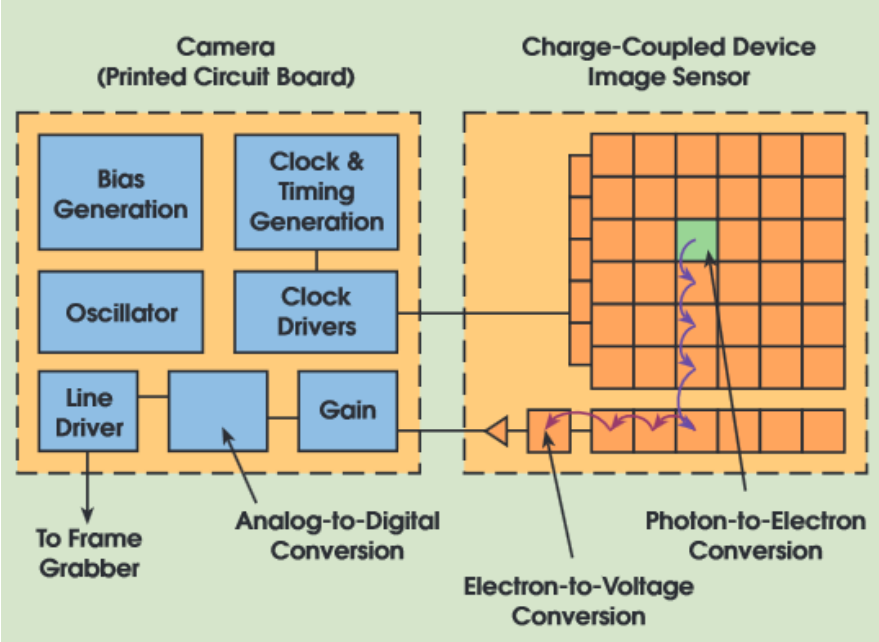

Figura 2.3. Sensor CCD.

En los sensores CCD, se hace una lectura de los valores a cada una de las celdas, donde hay una conversión de luz a carga eléctrica. Posteriormente dicha carga es transportada, con la ayuda de un registro de desplazamiento, se realiza un corrimiento vertical lento y uno horizontal rápido en la última fila del arreglo de pixeles, en cuyo extremo se realiza la lectura. Para acelerar el proceso de adquisición se pueden tener múltiples puntos de lectura, pero se aumenta el tamaño y la complejidad de la circuitería. Después se llega a un dispositivo que realiza la conversión de carga eléctrica a voltaje. Los procesos anteriores se realizan internamente en el sensor, como se puede observar en la figura 2.3. Por lo tanto la conversión de analógico-digital se realiza con circuitos externos, provocando de esa manera gasto de energía adicional pero reduciendo en algunas ocasiones el ruido que se puede generar internamente en el sensor [4].

- **Sensor CMOS**

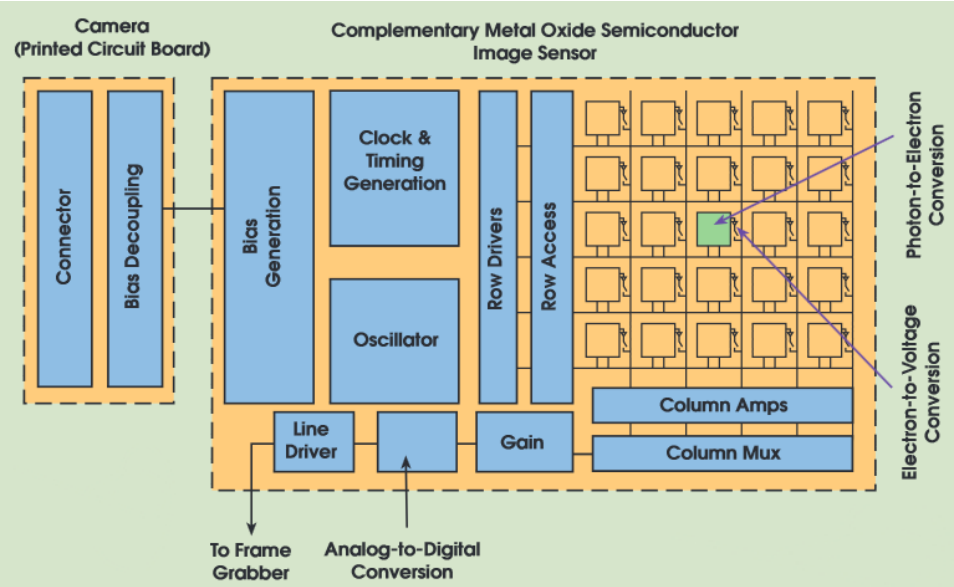

Figura 2.4. Sensor CMOS.

En el sensor CMOS, cada celda es independiente y ahí mismo se genera la conversión de carga eléctrica a voltaje debido a que se realiza la lectura de la carga en cada pixel, por lo tanto no se realiza movimiento de la carga acumulada. Los voltajes medidos en cada pixel son luego movidos por registros de desplazamiento en un esquema similar al movimiento de cargas de una CCD. La diferencia principal en comparación a los sensores CCD, es que la conversión analógico-digital se realiza sin usar circuitos externos, es decir, todo el trabajo se lleva a cabo dentro del sensor generando de esa manera un menor consumo de energía y menor tamaño físico de una cámara. En la figura 2.4 se observa que internamente el sensor realiza más tareas que un sensor CCD, por lo que el ruido térmico que afecta gravemente a una imagen se reduce considerablemente [4].

Entre las diferencias más destacables que hay entre los sensores CCD y CMOS es la capacidad de respuesta y rango dinámico de una imagen. La capacidad de respuesta es la cantidad de señal entregada del sensor por unidad de energía óptica de entrada, donde en este caso el sensor CMOS es superior, debido a que los elementos de ganancia son más fáciles de introducir, por ejemplo los transistores internos que llevan con bajo consumo de energía y alta ganancia de amplificación. Por otra parte, los sensores CCD tienen un alto

consumo de energía debido a los amplificadores, pero hoy en día ese problema se está resolviendo.

El rango dinámico es el coeficiente entre la saturación de los píxeles y el umbral por debajo del cual no captan señal, donde el sensor CCD es muy superior debido a que los transistores utilizados para amplificación reducen considerablemente el ruido, haciéndolo menos sensible a la luz y para tolerar los bordes de luz de una forma superior que el sensor CMOS, aunque éste último resulta ser más sensible a la luz. La elección entre cualquiera de las dos tecnologías dependerá del uso específico que se le quiera dar a una cámara. En este proyecto es más conveniente el uso de una cámara CMOS, debido a que la iluminación que se usará será la luz solar o luz de las lámparas que comúnmente se usan en una casa y el sensor CMOS resulta ser más sensible a la luz [5].

2.3.- Introducción al procesamiento de imágenes

En los últimos años los estudios centrados para modificar e interpretar una Imagen han crecido enormemente, situándolo como el vértice sobre el que giran buena parte de las estrategias y herramientas esenciales en varias áreas del conocimiento, como en comunicación, marketing, seguridad y de todo tipo de organizaciones (desde empresariales a políticas, pasando por fundaciones, universidades, sistemas inteligentes, entre otros). Tanta es la importancia que las imágenes han llegado a poseer en nuestro mundo que algunos autores consideran que la nuestra se ha convertido en una sociedad de pseudo acontecimientos, es decir, una sociedad donde las representaciones de la realidad acaban siendo más importantes que la realidad misma [6].

Una imagen natural capturada con una cámara, un telescopio, un microscopio o cualquier otro tipo de instrumento óptico presenta una variación de sombras y tonos continua. Imágenes de este tipo se llaman imágenes analógicas. Para obtener una imagen digital se puede digitalizar una imagen analógica o simplemente usar una cámara digital, que contiene internamente los circuitos necesarios para digitalizarla.

Una imagen en forma genérica es una representación en 2 dimensiones de un objeto de 2 o 3 dimensiones. Esta representación puede ser definida por colores o diferentes niveles de

gris. Las imágenes digitales son fotos electrónicas tomadas de una escena o escaneadas de documentos, fotografías, manuscritos, textos impresos e ilustraciones. Se realiza una muestra de la imagen digital y se confecciona un mapa de ella en forma de cuadrícula de puntos o elementos de la figura (píxeles). A cada píxel se le asigna un valor en función de la tonalidad de la imagen (negro, blanco, matices de gris o color), el cual está representado en un código binario (ceros y unos). Los dígitos binarios (bits) para cada píxel son almacenados por una computadora en una secuencia, y con frecuencia se los reduce a una representación matemática (comprimida). Luego la computadora interpreta y lee los bits para producir una versión analógica para su visualización o impresión [6].

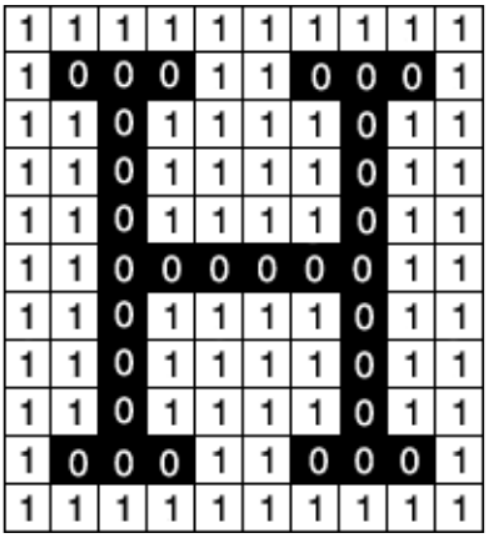

Figura 2.5. Representación de una imagen digital.

En la figura 2.5 se muestra cómo es la interpretación de una imagen, donde cada cuadro representa un pixel que puede tomar, en este caso, sólo dos valores y por consecuencia dos colores, negro (para 0) y blanco (para 1). Una imagen digital está formada por una matriz de píxeles (a x b x c), donde a y b representan la anchura y altura y c es la profundidad de color o profundidad de bit, podríamos decir que es la tercera dimensión de la matriz, la que permite que cada píxel pueda tener un número determinado de colores distintos [6].

Para saber la cantidad de colores que puede tener una imagen se examina la profundidad de píxel, que es una unidad de medida binaria porque cada píxel está formado por bits. Cuando se dice que la profundidad de píxel es de 1 bit, la imagen tiene dos colores o dos niveles de gris representados por los valores 0 y 1 que puede ser blanco o negro, como se observa en

la figura 2.6. Dichas imágenes también son llamadas mapa de bits, debido a que sólo muestra la distinción de 2 colores [7].

Figura 2.6. Imagen de 1 bit por pixel.

Una profundidad de píxel de 8 bits permite que cada pixel pueda tener 256 colores distintos o 256 niveles distintos de grises que provienen de combinar las series de ocho elementos posibles obtenidas al combinar ceros y unos. La figura 2.7 muestra una imagen de 8 bits sin color con un solo canal, que comúnmente es llamada imagen a escala de grises y se observa una mejor calidad y representación real de la imagen que la de 1 bit [7].

Figura 2.7. Imagen de 8 bits por pixel.

Una imagen a color se puede obtener con 8 bits y un canal con color indexado, es decir, la cantidad de colores es limitada y debido a que solamente se tiene un canal, sólo es posible tener 256 colores diferentes en cada pixel, por lo tanto, los colores se asignan a los píxeles con un método indirecto, usando una tabla de búsqueda llamada *mapa de colores*. La figura 2.8 muestra un ejemplo de una imagen de 8 bits con color indexado y aunque sea una imagen a color, con 256 colores no se tiene una buena claridad de la imagen y resulta ser inferior a la imagen a escala de grises [7].

Figura 2.8. Imagen de 8 bits por pixel con color indexado.

Si la profundidad del píxel es de 24 bits podemos llegar a 16 millones de colores distintos en cada píxel. Cada canal puede tener un máximo de 8 bits, por lo tanto con la combinación de los 3 canales se hace un total de 24 bits. Debido a que se usan 3 canales (rojo, azul y verde), la imagen puede tener una buena calidad y se tendrá una representación muy parecida a la imagen real que captan nuestros ojos [7].

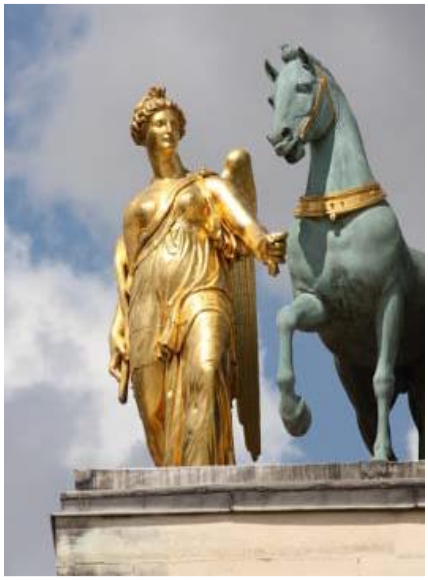

Figura 2.9. Imagen de 24 bits por pixel.

2.4.- Algoritmo de Viola-Jones

La detección y reconocimiento de un rostro es una tarea que los seres humanos realizan de forma cotidiana sin realizar esfuerzo alguno. El reconocimiento facial es un método no intrusivo y las imágenes faciales son probablemente las características biométricas más utilizadas por los seres humanos para hacer un reconocimiento personal. Pero traducir a nivel algorítmico un proceso de diferenciación y reconocimiento facial es un asunto extremadamente difícil, debido a que el rostro humano tiene una alta variabilidad en su apariencia y eso ocasiona problemas al querer clasificarlo u obtener sus rasgos principales. Antes de poder reconocer un rostro primeramente se debe detectarlo y existen diferentes métodos para realizar esa función. Uno de los algoritmos de detección de rostros ampliamente utilizado es el propuesto por P. Viola y M. Jones. Este algoritmo, basado en etapas de clasificación de complejidad creciente, es considerado como uno de los mejores en cuanto a tiempo de ejecución y efectividad de detección [8].

El método de Viola-Jones usa una colección de características para clasificar ventanas de la imagen. Un conjunto de clasificadores débiles es obtenido a partir de un conjunto de entrenamiento. Cada uno de estos clasificadores es una función simple compuesta de sumas de rectángulos, seguidas de la aplicación de un umbral. Estos clasificadores débiles son

linealmente combinados para formar un único clasificador más complejo y preciso. La metodología que se usa en el algoritmo de Viola-Jones consiste en 3 etapas, que se observan en la figura 2.10 [8].

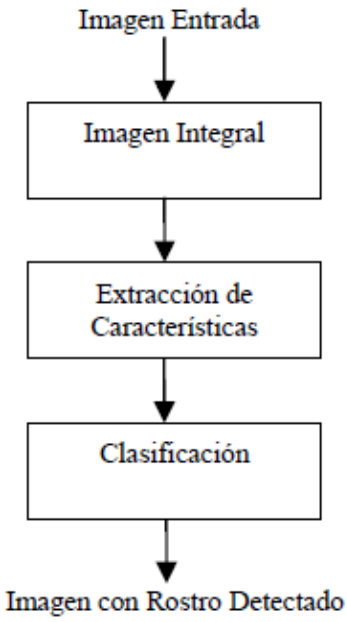

Figura 2.10. Etapas del algoritmo de Viola-Jones.

2.4.1.- Imagen integral

Una contribución fundamental del algoritmo de Viola-Jones consiste en acelerar el cálculo del valor del rasgo al trabajar no con la imagen original sino con una imagen integral. La integral de una imagen respecto un punto (x,y) consiste en la suma de los píxeles por arriba y a la izquierda de dicho puntos, como se muestra en la figura 2.11.

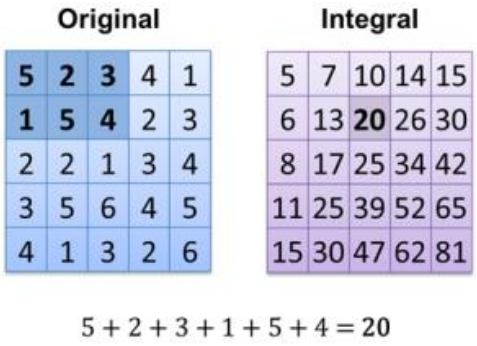

Figura 2.11. Obtención de una imagen integral.

A partir de la imagen integral se puede calcular rápidamente la suma de todos los puntos contenidos en un rectángulo cualquiera de la imagen utilizando sólo los cuatro valores asociados a sus esquinas. Lo anterior permite que el cálculo de la suma de los puntos contenidos en un rectángulo de tamaño arbitrario pueda ser realizado en un tiempo constante utilizando sólo cuatro operaciones, por lo que el cálculo del valor de un rasgo se reduce considerablemente. La figura 2.12 muestra la obtención de un cuadro gris utilizando la imagen integral [8].

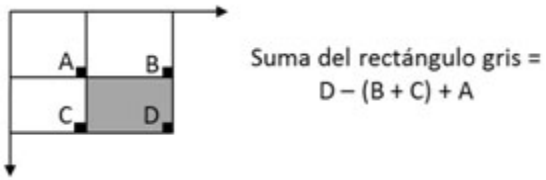

Figura 2.12. Utilización de la imagen integral.

2.4.2.- Extracción de características

En imágenes las características de cada objeto se extraen al aplicar ciertas funciones que permitan la representación y descripción de los objetos de interés de la imagen (patrones). Se utilizan ciertos rasgos de clasificación, como las formas geométricas, que ayudan al algoritmo para detectar zonas de una imagen (o ventana) que pueda contener partes de un rostro. Los rasgos de clasificación utilizados por el algoritmo Viola-Jones son estructuras simples compuestas por dos, tres o cuatro rectángulos, como se muestra en la figura 2.13 [9].

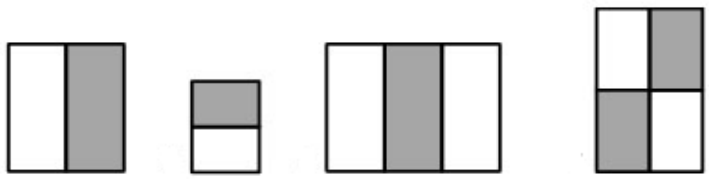

Figura 2.13. Ejemplos de rasgos de clasificación.

La extracción de características es un paso en el reconocimiento de patrones en el cuál las medidas u observaciones son procesadas para encontrar atributos que puedan ser usados

para asignar los objetos a determinada clase. Las estructuras simples mostradas en la figura 2.13 pueden ser asociadas a partes comunes de un rostro, como las correspondientes a los ojos, la nariz, la frente, el pelo, entre otros rasgos, como se muestra en la figura 2.14 [9].

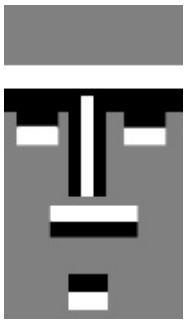

Figura 2.14. Aproximación de un rostro con estructuras simples.

2.4.3.- Clasificación

Un conjunto de distintas características (o filtros de Haar) corresponden a un clasificador simple y su complejidad vendrá dada por el número de características que posea. El conjunto de clasificadores simples en cascada constituye el detector final. Viola y Jones presentan un método para combinar los clasificadores fuertes en una estructura en forma de cascada. Las primeras etapas de la cascada poseen pocos clasificadores pero permiten descartar rápidamente aquellas ventanas que no contienen rostros, de tal manera que se pueda invertir la mayor parte del tiempo de procesamiento en aquellas zonas que posiblemente contengan una cara [9].

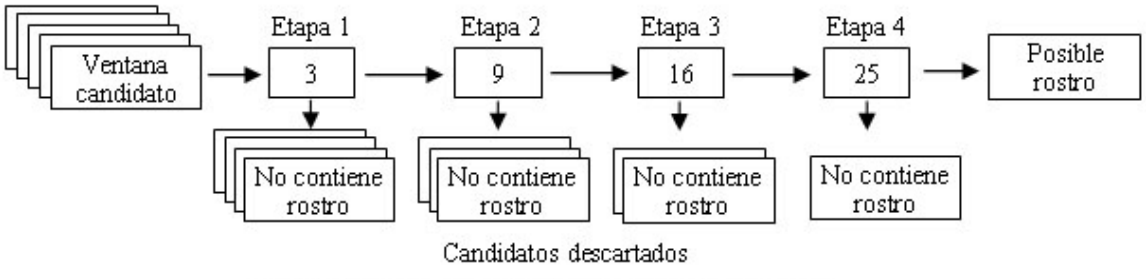

Figura 2.15. Estructura en cascada.

El algoritmo de clasificadores se muestra en la figura 2.15, donde en la primera etapa se obtiene muestra de una parte de la imagen y se realizan filtros de tal manera que se localice

un posible rostro, posteriormente la siguiente etapa examinaría el posible rostro que se detectó en la etapa anterior y así se ejecutaría el algoritmo sucesivamente, es decir, todos los posibles rostros se filtran en cada etapa generando de esa manera rostros más finos. Al final se detecta el rostro con pequeños márgenes de diferencias y al final, sólo se toma el rostro de la etapa final. La figura 2.16 muestra el algoritmo en cascada que se realiza al detectar un rostro [9].

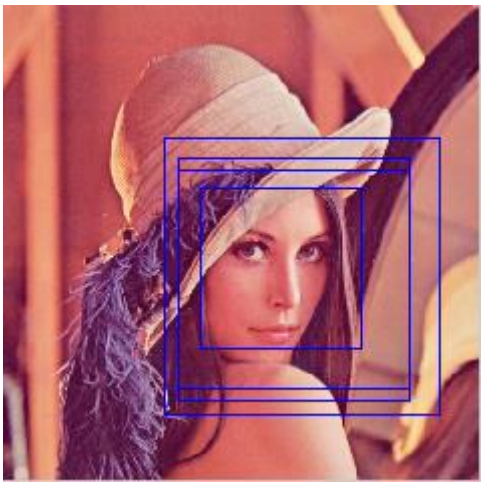

Figura 2.16. Diferentes etapas de la detección de rostros.

2.5.- Librería OpenCV

La librería OpenCV (Open Source Computer Vision) es una librería de código abierto, desarrollada por Intel. Esta librería proporciona un alto nivel de funciones para el procesado de imágenes y contiene más de 2500 algoritmos que permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios, etc. La librería es multiplataforma, y puede ser usada en plataformas Mac OS X, Windows y Linux [10].

Referencias:

[1] Universidad Nacional de Quilmes. (2005). *Aspectos de un sistema de visión artificial*. Recuperado de: http://iaci.unq.edu.ar

[2] Cáceres, J. (2006). *La visión artificial y las operaciones morfológicas en imágenes binarias.* Recuperado de http://www.dsi.uclm.es

[3] Microscan (2006). *Introducción a iluminación de visión artificial*. Recuperado de http://www.microscan.com

[4] Litwiller, D. (2001). CCD vs CMOS: Facts and Fiction. *PHOTONICS SPECTRA*. Recuperado https://teledynedalsa.com/corp/

[5] Litwiller, D. (2005). CCD vs CMOS: Maturing Technologies, Maturing Markets. *PHOTONICS SPECTRA*. Recuperado https://teledynedalsa.com/corp/

[6] Biblioteca de la Universidad de Cornell. (2003). *Llevando la teoría a la práctica: tutorial de digitalización de imágenes*. Recuperado de http:// www.library.cornell.edu

[7] Dominguez, J. (s.f.). *GIMP aplicaciones didácticas: La imagen digital.* Recuperado de http://www.ite.educacion.es

[8] Delgado, M. (2012). *Extracción automática de caras en imágenes captadas con móviles android.* Tesis de maestría no publicada, Universidad Politécnica de Cataluña, Barcelona, España.

[9] Vega, A. (2011). *Detección y seguimiento de rostros*. Tesis de ingeniería no publicada, Universidad Carlos III de Madrid, Madrid, España.

[10] Open Source Computer Vision Library. (2015). *Introduction*. Recuperado de http://docs.opencv.org

Capítulo 3

3. Sistemas embebidos y software libre

3.1.- Introducción a los sistemas embebidos

Utilizar una computadora para crear cualquier tipo de sistema es muy costoso y resulta complicado y voluminoso usar tarjetas externas para hacer lectura de parámetros que una computadora normalmente no puede realizar. Los sistemas embebidos son sistemas diseñados mediante una combinación de hardware y software para cumplir una o varias funciones específicas. Se encuentran disponibles a cada momento de nuestra vida. Se tienen como ejemplos de sistemas embebidos el horno de microondas, el auto, el elevador, el equipo de audio, el avión, que por lo general todos son controlados mediante el uso de computadoras que normalmente no poseen una pantalla, un teclado o disco duro.

En la figura 3.1 se muestra cómo está conformado un sistema embebido. La parte central y principal es un CPU, que puede estar unido a diferentes componentes con funciones específicas. La parte del software puede ser usado para controlar los componentes de hardware y el funcionamiento general del sistema. Una parte es el puerto de diagnóstico, que cumple la función de monitorear los sistemas conectados. La memoria usada puede ser una RAM, ROM, EEPROM, disco duro, etc. que se usa para almacenar procesos y configuraciones predeterminadas o para almacenar información adicional a la requerida. También se puede tener una interfaz humana, la cual puede ser desde un simple indicador de luz hasta una visión robótica en tiempo real. Otra parte son los sistemas auxiliares, como puede ser la fuente de alimentación, disipadores, puertos de comunicación, entre otros. Adicionalmente, un sistema embebido puede tener un FPGA/ASIC y convertidor analógico-digital o digita-analógico para tener una facilidad al adquirir o enviar información externa [1].

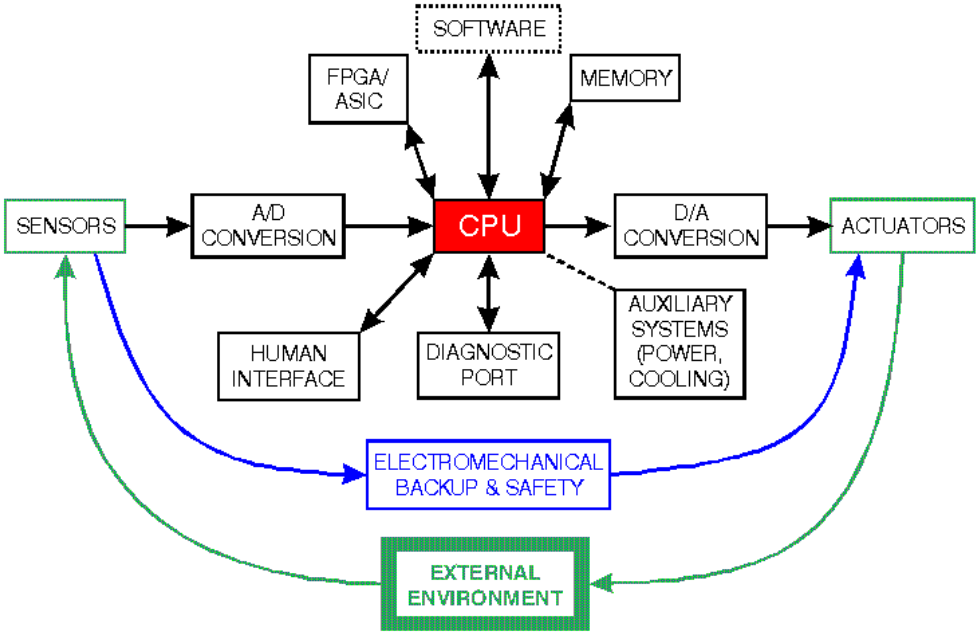

Figura 3.1. Composición de un sistema embebido.

Generalmente, los sistemas embebidos son usados para realizar funciones en tiempo real. Un sistema en tiempo real es una combinación de software y hardware que interaccionan constantemente con el entorno físico y reacciona a cualquier estímulo proveniente de dicho entorno, realizando funciones establecidas por el usuario en un tiempo determinado, donde el tiempo es un parámetro muy importante en dichos sistemas. Un ejemplo claro de un sistema de tiempo real es el control de tráfico aéreo [2].

En la era moderna, toda la tecnología tiende a crear dispositivos de poco tamaño, con lo cual el costo también se podría reducir considerablemente. Una característica de los sistemas embebidos es el poco tamaño que pueden tener y eso es bueno tanto para el consumidor como para el fabricante. Por ejemplo, si se desea transportar muchos dispositivos de gran tamaño y a larga distancia, se requeriría un avión y el costo sería enorme debido al peso de los dispositivos. En cambio, si son dispositivos muy pequeños, se puede transportar vía terrestre o de igual forma en avión, pero con el costo muy reducido y eso se ve reflejado notablemente en el precio de los dispositivos cuando son lanzados al mercado [2].

Por otra parte, gracias al hardware implementado en los sistemas embebidos es posible crear un sistema para detectar sus propios errores y seguir en funcionamiento apoyándose de los procesadores implementados. Dado que muchos sistemas embebidos son concebidos para ser producidos en miles o millones de unidades, el costo por unidad es un aspecto importante a tener en cuenta en la etapa de diseño. Generalmente, los sistemas embebidos emplean procesadores muy básicos, relativamente lentos y memorias pequeñas para minimizarlos costos. Por consiguiente, el consumo de energía que consumen los sistemas embebidos es muy bajo y por lo tanto, muchas veces no se requieren disipadores de calor. Debido a que dichos sistemas no consumen demasiada energía, el período de vida de los mismos se incrementa considerablemente, de tal manera que pueden durar funcionando constantemente sin tener tanto degrado físico [2].

3.2.- Sistemas operativos libres

El software es un código que, como la mayoría de los sistemas que el hombre fabrica, tiene autor y leyes que protegen sus derechos, por lo tanto no pueden ser modificados o copiados. Hoy en día, disponer de un software robusto, eficiente y económico resulta algo complicado. De igual manera, la optimización de un sistema operativo dirigido a sistemas con pocos recursos es una tarea difícil de realizar y es por esa razón que se optan diferentes alternativas a las conocidas comúnmente, debido a que la mayoría de los sistemas operativos o software no pueden ser modificados y si se llega a realizar cualquier modificación se estaría cometiendo un delito.

Una alternativa es usar software libre, que ha existido desde hace más de 20 años. El software libre es aquel que puede ser distribuido, modificado, copiado y usado; por lo tanto, debe venir acompañado del código fuente para hacer efectivas las libertades que lo caracterizan. Dentro de software libre hay, a su vez, matices que son necesarias tener en cuenta. Por ejemplo, el software de dominio público significa que no está protegido por el copyright, por lo tanto, podrían generarse versiones no libres del mismo, en cambio el software libre protegido con copyleft impide a los redistribuidores incluir algún tipo de restricción a las libertades propias del software así concebido, es decir, garantiza que las modificaciones seguirán siendo software libre [3].

Cuando se adquiere un software propietario, habitualmente se recibe una copia del programa ejecutable y una licencia que permite ejecutarlo en un número determinado de computadoras. Esta licencia deja bien claro que, en realidad, lo que se adquiere no es el programa en sí, sino el derecho de que se pueda ejecutar, pues el programa sigue siendo propiedad de la empresa que lo fabrica. Por lo tanto, no se está autorizado a realizar ningún cambio o modificación en el programa, aspecto que tampoco es posible de hacer ya que no se tiene acceso al código fuente.

Al adquirir una copia de un programa libre (bien sea gratuito o habiendo pagado), se obtiene no solamente la libertad para la ejecución, sino que es posible ver su código fuente y realizar modificaciones para que se adapte a las diferentes necesidades de cada usuario. El programa no tiene un propietario, por lo que puede modificar o dar el uso que se considere oportuno.

El sistema operativo de un ordenador se puede definir como el software encargado de gestionar y manejar el hardware del equipo. Crea una capa de abstracción sobre la complejidad de los circuitos y conexiones eléctricas que componen una computadora proporcionando una interfaz amigable. Por ejemplo, cuando se copian archivos sólo es necesario indicar al sistema operativo los archivos de origen y dónde se desean pegar, por lo que no es necesario saber que por debajo de eso hay un disco duro compuesto de platos, éstos a su vez de sectores, y éstos de bloques y de un cabezal que se recorren al realizar al escritura [3].

3.2.1.- Sistema operativo GNU/Linux

En los últimos años, el software libre no se ha perfilado como una alternativa válida para muchos usuarios, empresas y, cada vez más, instituciones y gobiernos. Actualmente, GNU/Linux es uno de los sistemas operativos más fiables y eficientes que podemos encontrar. Aunque su naturaleza de software libre creó inicialmente cierta desconfianza por parte de usuarios y empresas, GNU/Linux ha demostrado estar a la altura de cualquier otro sistema operativo existente [4].

GNU/Linux es un sistema operativo libre desarrollado por voluntarios de todo el mundo.

Sus principales características son:

- Multitarea: se pueden realizar varias actividades a la vez (navegar por Internet, editar un documento, compilar un programa, etc.).

- Multiusuario: varios usuarios pueden trabajar concurrentemente en una única computadora con varias terminales (teclado y monitor) de forma que tengan la sensación de que es el único que está trabajando en el sistema. Cada usuario almacena sus datos (programas, documentos de texto, imágenes,...) en una cuenta privada o "home". Notar que para que sea multiusuario es imprescindible que sea multitarea.

- Conectividad: permite las comunicaciones en red y el acceso a recursos remotamente. Por ejemplo, podemos acceder a nuestros datos situados en una máquina a través de otro equipo, conectados ambos a Internet

- Multiplataforma: se puede instalar en multitud de dispositivos, desde todo tipo de computadoras, portátiles y servidores hasta videoconsolas o incluso teléfonos móviles.

- Libre: su código fuente está disponible. Cualquiera puede usarlo, modificarlo y distribuir. Una consecuencia de esto es que es gratis.

- Y muchas más características técnicas que se escapan del ámbito de esta asignatura

Una gran ventaja al utilizar software libre es el apoyo de grandes empresas que están involucradas y que dan un gran soporte para que los proyectos de software libre sigan adelante. En este caso, GNU/Linux es apoyado por multitud de grandes empresas de la informática entre las que podemos encontrar a IBM, Novell, Oracle, Hewlett-Packard, Sun Microsystems y Google. Otra ventaja de usar GNU/Linux es la variedad de aplicaciones existentes que bien pueden ayudar a resolver necesidades del usuario o puede ser modificada para adaptarlas a las necesidades que se desean [4].

3.3.- Raspberry Pi

Los sistemas embebidos necesitan muchas veces apoyo de software para trabajar eficientemente y en la mayoría de los casos, el uso de un sistema operativo es la mejor opción. La combinación de un sistema operativo y un sistema embebido forman la mejor herramienta para trabajar cualquier tipo de proceso, como por ejemplo: sistemas de adquisición de datos, sistemas en tiempo real, comunicaciones inalámbricas, sistemas de automatizaciones, etc.

La Raspberry pi es un pequeño sistema embebido que cumple con características similares a la de una computadora, ya que también tiene la capacidad de tener un sistema operativo y además, tiene la ventaja de controlar dispositivos adicionales mediante sus puertos de entrada/salida de propósito general (GPIO en inglés) [5].

En la figura 3.2 se muestra cómo es físicamente la Raspberry Pi y se observa que cumple con las características típicas de un sistema embebido, donde lo más destacable es el tamaño reducido que tiene. Las especificaciones de hardware de la Raspberry Pi son las siguientes:

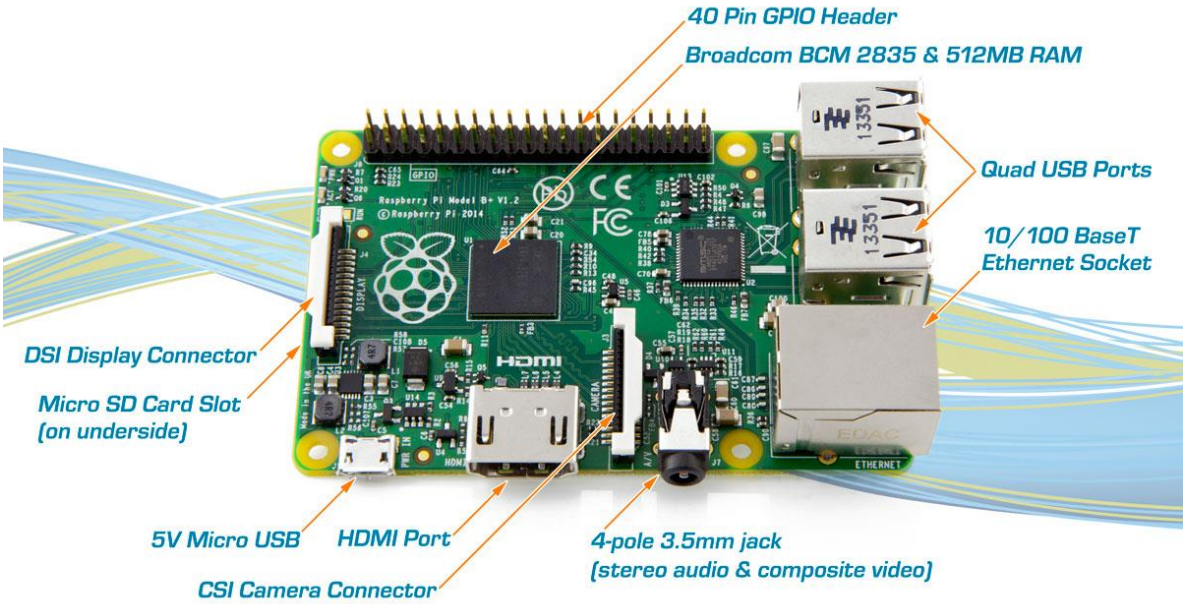

Figura 3.2. Raspberry Pi

- Procesador ARM 176JZF-S a 700 MHz expandible hasta 1GHz con overclock
- 512Mb de memoria RAM
- GPU Dual Core VideoCore IV® Multimedia Co-Processor
- 4 puertos USB
- Puerto de Ethernet 10/100
- Salida HDMI
- Alimentación por puerto Micro USB a 5V
- Puerto para Micro SD
- 40 pines programables
- Poco consumo de energía (de 1 a 3 W de consumo aproximadamente)

Las características importantes que hacen a la Raspberry la mejor opción para el sistema de seguridad de este trabajo es la potente tarjeta de video que tiene, la cual es superior a la de otros sistemas embebidos como la de BeagleBone, que su tarjeta de video es de sólo 1 núcleo. Otra característica importante es el poco consumo de energía, provocando crear grandes sistemas sin consumir tanta energía.

La Raspberry Pi tiene soporte suficiente para poder utilizar varias distribuciones del sistema operativo libre GNU/Linux y eso es otra gran ventaja a favor. También existen algunas distribuciones optimizadas para los sistemas embebidos y en el caso de la Rasberry Pi, tiene una distribución modificada de GNU/Linux llamada Raspbian, que es el sistema operativo que se usa en este proyecto de sistema de seguridad. Aunque dicho sistema operativo está optimizado para la Raspberry Pi, tiene las mismas funciones y aplicaciones que cualquier distribución GNU/Linux [6].

Referencias:

[1] Galiana, A. (s.f). *Sistemas Embebidos*. Recuperado de la base de datos de la Universidad Politécnica de Valencia.

[2] Koopman, P. (1996). *Embedded System Design Issues*. Trabajo presentado en la Conferencia Internacional de Diseño de Cómputo 96'. Disponible en http://users.ece.cmu.edu

[3] GNU Operating System. (2015). *What is free software?*. Recuperado de http://www.gnu.org/philosophy/free-sw.html

[4] Baig, R. y Aulí, F. (2003). *Sistema operativo GNU/Linux básico*. Recuperado de http://softlibre.unizar.es/manuales/linux/868.pdf

[5] Computing at School. (2012). *The Raspberry Pi Education Manual*. Recuperado de downloads.raspberrypi.org

[6] Raspberry Pi Foundation. (s.f). *What is a Raspberry Pi?*. Recuperado de https://www.raspberrypi.org/;

Capítulo 4

4. Redes computacionales

4.1.- Introducción a las redes computacionales

El control de acceso y seguridad inteligente planteado en este trabajo utilizará diferentes métodos de comunicación, tanto en el envío de notificaciones como en la comunicación con el sensor de vibración. Por lo tanto, es conveniente describir los aspectos básicos y teóricos que describen las comunicaciones que se utilizarán.

Las redes computacionales inician a finales de los años 50, cuando los Estados Unidos crearon un organismo llamado ARPA (Advanced Research Projects Agency) en 1957, el cual tenía como objetivo impulsar a los Estados Unidos en el ámbito tecnológico para ser aplicado al entorno militar. En dicho organismo se estaban desarrollando técnicas relacionadas con la transmisión de varias comunicaciones usando sólo un cable, siendo lo anterior el principio de la comunicación en paquetes o denominada formalmente tecnología de conmutación de paquetes y dichas técnicas constituyen la base fundamental para la transmisión en las redes computacionales [1].

No fue sino hasta el año de 1967 en donde ARPA sugiere aspectos relacionaos con la construcción de su propia red que fuera utilizada por todo el público, por lo que varias universidades y empresas importantes se reunieron para proponer proyectos relacionados con la creación de dicha red. Después de años de espera, en 1969 se construye la primera red de computadoras de la historia llamada ARPANET, utilizándose también el primer protocolo de comunicación llamado NCP. Los acontecimientos antes mencionados dieron el inicio de todas las comunicaciones computacionales que se usan en la actualidad [1].

La base de la transmisión de la información en las redes computacionales es el bit (0 ó 1). Toda la transmisión de información se basa en la transmisión de bits, el cual es el nivel más físico de la comunicación entre computadoras y se representa como variación de voltaje. La velocidad con la que dichos voltajes son trasmitidos se denomina velocidad de transmisión

o conocido comúnmente en inglés, “baud rate”. Dicha velocidad suele medir los bits por segundo que un dispositivo es capaz de transmitir y se expresa como “bps”.

En una comunicación de simple datos, lo más importante es transportar la información de un punto a otro sin obtener ningún error. Para poder realizar una comunicación desde un punto A hacia un punto B se utiliza un tipo codificación con el fin de que la información no pueda ser revelada en cualquier otro punto. De esta manera, tanto en el transmisor como en el receptor se encontrarán dispositivos o software dedicados a realizar tanto la codificación como la decodificación. El lenguaje que utilizan las computadoras para realizar la comunicación se le conoce como protocolo.

4.2.- Modelo OSI

Para reducir la complejidad del diseño de una red, la gran mayoría de ellas están formadas por una composición o pilas de capas, una encima de la otra. Lo anterior se realiza para ocultar detalles que se realizan en una comunicación de una capa inferior a una capa superior y desde una capa n de una computadora a una capa n de otra computadora.

La arquitectura de red más popular que existe hoy en día es una propuesta realizada por la Organización Internacional de Normas (ISO), la cual propone una estandarización de la arquitectura de la red. Dicha propuesta recibe el nombre de Modelos de referencia OSI (Interconexión de Sistemas Abiertos), la cual describe cómo viaja la información entre aplicaciones de computadoras que se encuentran conectadas en una red. El modelo OSI está compuesto por 7 capas, las cuales tienen una tarea en específico para poder realizarla sin necesidad de la ayuda de capas superiores e inferiores, por lo que si se desea sustituir un nivel por otro mismo nivel, no afectarán de ninguna forma a otro nivel y la comunicación se realizará sin ningún inconveniente. Las 7 capas o niveles del modelo OSI son los siguientes: [2]

- Capa física: se relaciona con la transmisión de bits puros a través de un canal de transmisión. En esta capa se definen los niveles y tiempo de cambio del voltaje, velocidad de transmisión, distancia máxima de la conexión, etc.

- Capa de enlace de datos: en esta capa se realiza la transformación de bits puros a una transmisión fiable, de modo que enmascara los errores dividiendo los datos de entrada en tramas secuenciales y por lo tanto, el receptor devuelve una confirmación cuando los datos envueltos en una trama hayan sido correctos.
- Capa de red: es la responsable de las funciones que permiten la interconexión de sistemas a través de una o varias redes. También realiza la tarea de hacer el direccionamiento lógico, conexión y desconexión de redes, detección de errores, etc.
- Capa de transporte: su función es dividir datos, provenientes de las capas superiores, en unidades más pequeñas para pasarlos a la capa de red, asegurando que dichos datos lleguen correctamente hacia el lado que se desean transportar. Cuando realiza lo anterior, oculta todos los detalles de la red a las capas superiores. Realiza también la multiplexación, el control de flujo, gestión de circuitos virtuales, comprobación y recuperación de errores.
- Capa de sesión: esta capa permite a los usuarios establecer sesión entre ellos utilizando distintas máquinas, ofreciendo varios servicios como el control de diálogo (control de quién va a transmitir) o la sincronización (puntos de referencias que ayudan a reanudar si existe alguna interrupción).
- Capa de presentación: Proporciona las funciones de codificación, conversión, compresión y cifrado de los datos provenientes de la capa de aplicación, de tal manera que la computadora destino pueda entender la información que se le envía.
- Capa de aplicación: Proporciona el mecanismo por el cual el usuario o aplicación puede utilizar los recursos de la red, por lo que identifica los participantes de la comunicación, sincroniza y determina los recursos disponibles a cada usuario o aplicación.

4.3.- Protocolo de comunicación

Un protocolo de comunicación es un conjunto de reglas que rigen la forma en que fluyen los datos y la estructura de la construcción de dichos datos para que se pueda realizar una comunicación entre computadoras. Cada capa del modelo OSI utiliza un protocolo distinto

para transmitir la información, por lo que cada protocolo establece normas sintácticas para el tratamiento de datos de manera que puedan llegar a la computadora destino, se puedan interpretar y mostrarse correctamente al usuario destino [2].

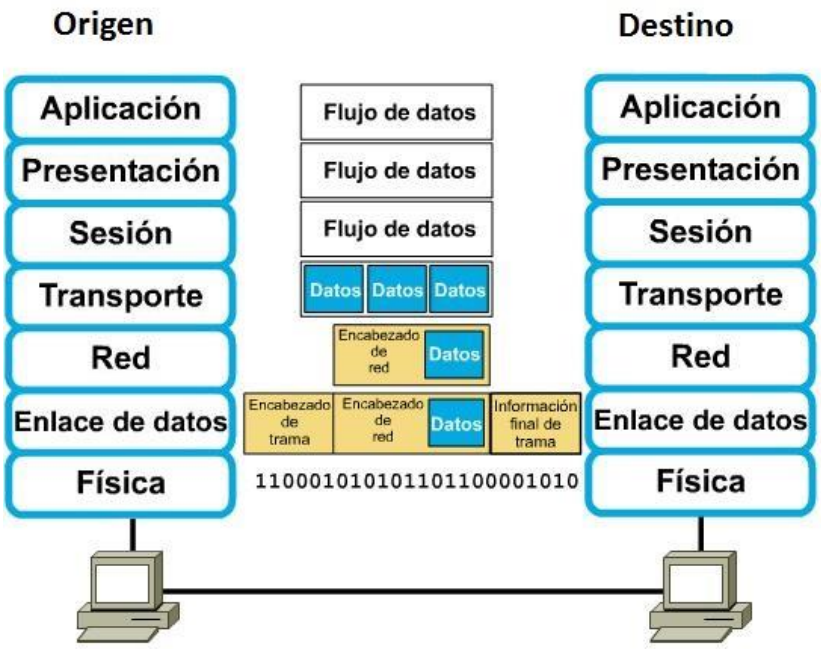

Figura 4.1. Capas del modelo OSI.

La figura 4.1 muestra cómo es la forma del protocolo utilizado en cada capa, en donde se observa que protocolo es muy diferente en cada capa. Los niveles superiores del modelo OSI son los que interaccionan directamente con las aplicaciones y sistemas operativos en cambio, las capas de niveles inferiores tienen una interacción más física. La transmisión de datos siempre se realiza en la capa física, por lo que si desea transmitir información desde la capa de aplicación, tiene que ser enviada capa por capa hasta llegar a la capa física. Debido a que cada protocolo es diferente, mientras los datos fluyan a niveles inferiores, se va ocultando información de una capa a otra. Cuando los datos se transmitan por la capa física y lleguen al usuario destino, la capa física del destino no podrá saber qué tipos de datos se enviaron, eso sólo lo sabrán las capas muy superiores, como la capa de presentación y la de aplicación [3].

Cada capa puede tener diferentes protocolos, que a la vez deben de tener una estructura similar. Los protocolos más importantes utilizados en cada capa son los siguientes:

- Capa de aplicación: SSH (Secure Shell), HTTP (Hipertext Transfer Protocol), FTP (File Transfer Protocol), XMPP (Extensible Messaging and Presence Protocol), entre otros.

- Capa de presentación: Terminal Virtual y RCP (Remote Procedure Call).

- Capa de sesión: NetBIOS (File Sharing and Name Resolution protocol) y SOCKS.

- Capa de transporte: TCP (Transmission Control Protocol) y UDP (User Datagram Protocol).

- Capa de red: IPv4 (Internet Protocol version 4), IPv6 (Internet Protocol version 6) y IPX (Internetwork Packet Exchange).

- Capa de enlace de datos: Ethernet, VLAN (Virtual Local Area Network), CDP (Cisco Discovery Protocol) y ATM.

- Capa física: RS-232, RS-485, la mayoría de los dispositivos de comunicación serial, DSL, SONET/SDH, Bluetooth, CAN Bus, USB, etc.

4.3.1.- Protocolo XMPP

XMPP (Extensible Messaging and Presence Protocol) es un protocolo de comunicación de código abierto situado en la capa de aplicación del modelo OSI. Se diseñó para el uso de comunicaciones en tiempo real que abarca varias áreas de aplicación, como la mensajería instantánea, chat por voz o video, etc. Anteriormente era llamado Jabber y tiene como base principal el uso del lenguaje XML, el cual tiene una facilidad en el manejo de almacenamiento de información o la creación de base de datos. Las aplicaciones más conocidas que usan el protocolo XMPP para realizar sus comunicaciones son Whatsapp y el chat de Facebook [4].

Las principales ventajas del uso del protocolo XMPP son las siguientes:

- Es abierto: el protocolo de XMPP es gratuito, abierto, público y comprensible. Además, existen múltiples implementaciones de código abierto para Servidores XMPP, como numerosos clientes y librerías de desarrollo.

- Es libre: XMPP es libre porque no solo se puede ver cómo funciona, sino además el usuario tiene la libertad de implementarlo él mismo, la libertad de adaptarlo a sus necesidades, sin requerir la aprobación de nadie.

- Es extensible: usando el potencial del lenguaje XML, cualquiera puede extender el protocolo de XMPP para una funcionalidad personalizada. Claro que para mantener la interoperabilidad, las extensiones comunes son controladas por la XMPP Software Foundation.

- Es descentralizado: cualquiera puede montar su propio servidor de XMPP, además está libre de patentes y no depende de ninguna empresa de modo que se puede usar para siempre con total libertad.

- Es seguro: Soporta seguridad en la capa de transporte y cualquier servidor de XMPP puede ser aislado de la red pública XMPP.

De manera general, en cuanto a la arquitectura se refiere, XMPP se usa en el modo de Cliente-Servidor descentralizada, de modo que cuando el usuario necesite realizar una comunicación hacia otra computadora o dispositivo, la información no pasa necesariamente por un servidor central, si no que llega directamente hacia el destino, ya que cualquier cliente tiene la capacidad de hacer su propio servidor XMPP. La figura 4.2 muestra la comunicación entre 2 clientes que utilizan el protocolo XMPP para enviar información, en donde se muestra que la comunicación es directa, es decir, no hay un servidor que reciba el mensaje y lo reenvíe. En cambio, la figura 4.3 muestra la comunicación entre 2 clientes que desean mandar un correo electrónico, donde antes de llegar al cliente destino, la información pasa por un servidor central que lo reenvía [5].

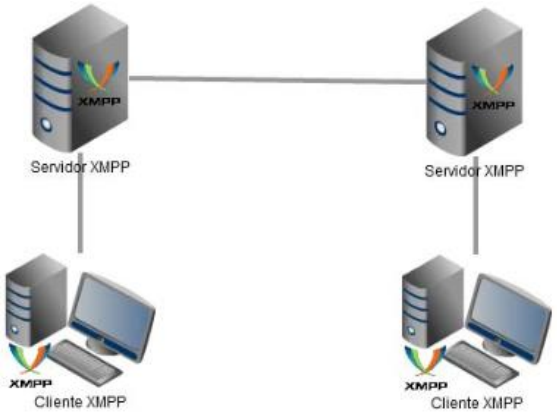

Figura 4.2. Comunicación entre clientes XMPP.

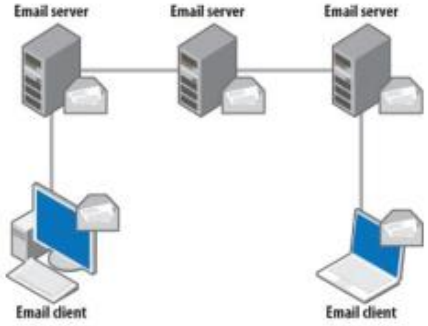

Figura 4.3. Envío de un correo electrónico.

Existen diversos tipos de protocolos para la mensajería instantánea pero como cada protocolo tiene su propio lenguaje de codificación, resulta imposible realiza una comunicación de una red, por ejemplo SMS (Short Message Service), a otra diferente, como Email. XMPP permite traducir la información utilizando el protocolo XMPP a otros protocolos, de tal manera que el mensaje llegue sin ningún problema de decodificación. Para realizar la función anterior, es necesario usar un servidor que tenga dicho servicio o usar un servidor externo es decir, que nuestro servidor se conecte a otro que si tenga la función de traducir el mensaje.

4.4.- Bus de comunicación I^2C

La I^2C (Inter-Integrated Circuit) es un bus de comunicación serial síncrono creado por Philips Semiconductors a principios de los 80's el cual, como su nombre lo dice, se usa para facilitar la comunicación entre circuitos integrados como los microcontroladores, memorias, sensores, entre otros, ya que posee sólo líneas para la señal. También tiene la capacidad de tener múltiples maestros en una comunicación, es decir, todos los dispositivos conectados en el protocolo de I^2C pueden tanto enviar como recibir mensajes. El bus de comunicación serial se encuentra en la capa física del modelo OSI y es apoyado con un protocolo que se encuentra en la capa de enlace de datos, debido a que funciona mediante el envío de tramas [6].

4.4.1.- Arquitectura del bus I^2C.

Las dos señales que usa el bus I^2C son:

- SDA (System Data): es la línea por la que se transfieren los datos entre los dispositivos conectados al bus.
- SCL (System Clock): esta línea transporta los pulsos de reloj que sincroniza la comunicación.

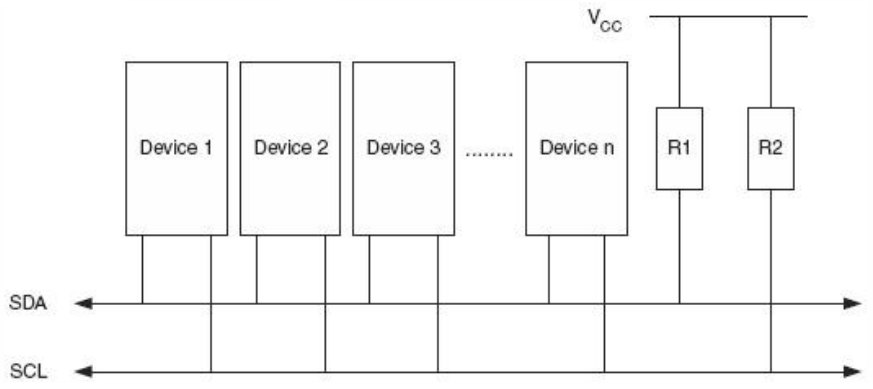

Figura 4.4. Arquitectura del bus I^2C.

En la figura 4.4 se muestran las conexiones de las líneas. Se observa que la señal de reloj SCL es común para todos y sólo podrá ser activada por el maestro, por lo que los esclavos no podrán iniciar una comunicación. La línea de datos SDA va conectada a todos los

dispositivos, ya que por dicha línea se realiza la transmisión de datos. Una consideración que se tiene que tener en cuenta es que las líneas SDA y SCL son de tipo drain abierto, algo similar al colector abierto, por lo que necesitan tener resistencias de pull-up para realizar correctamente los cambios de niveles de la señal tanto del reloj como de los datos.

4.4.2.- Protocolo del bus I^2C.

Para realizar una transferencia de datos, el bus I^2C cuenta con 2 tipos de clasificación para los dispositivos, maestro o esclavo. La diferencia radica en que sólo el maestro puede iniciar la transmisión o petición de datos. Si en un bus hubiera 2 maestros, cualquiera de ellos puede iniciar la transferencia de datos. Para iniciar dicha transferencia se debe de enviar un bit de inicio, el cual hace que la línea SDA se ponga en bajo, para poder realizar la transferencia de datos. En estado de reposo o inactivo, las líneas SDA y SCL permanecen en alto. Al terminar la transferencia, el maestro debe de enviar ahora un bit de parada para colocar el bus en estado inactivo. La figura 4.5 muestra el bit de inicio y de parada, los cuales van en sincronía con la señal de reloj. Cuando el bus está listo para la transferencia por cada pulso alto de reloj, el maestro envía los datos hasta el dispositivo deseado [6].

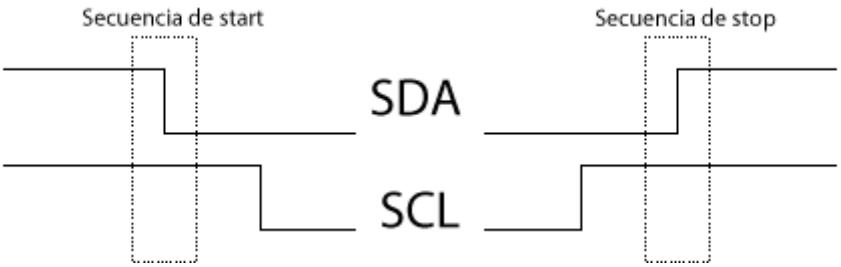

Figura 4.5. Bit de inicio y parada en el bus I^2C.

Debido a que el bus I^2C usa un protocolo que se encuentra en la capa de enlace de datos, la información se envía empaquetada en una trama, la cual contiene todos los datos necesarios, incluyendo el bit de inicio y parada. La figura 4.6 muestra la composición de la trama y los bits que la conforman.

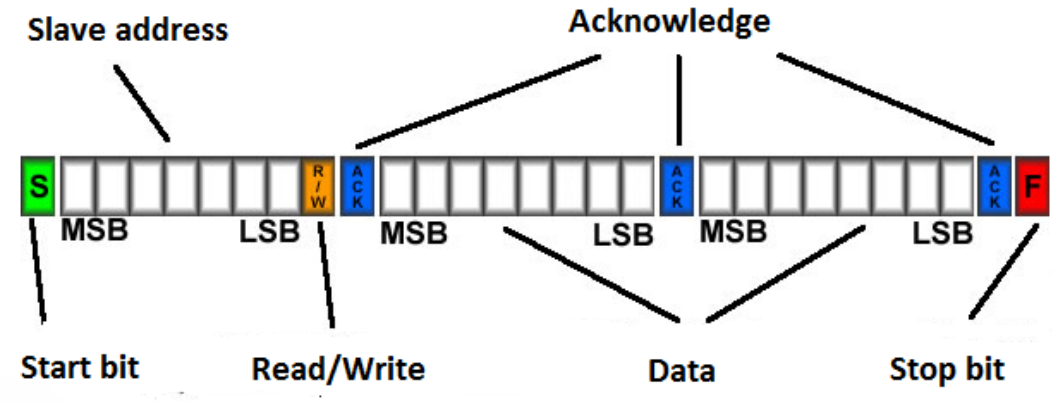

Figura 4.6. Trama de datos en el bus I^2C.

Los datos de las tramas se transfieren en secuencias de 8 bits y cada bit es transmitido cuando hay un pulso de reloj en alto. Los primeros 8 bits, sin incluir el de inicio, contienen la dirección del dispositivo destino que generalmente es un esclavo o también puede ser un maestro. De esos 8 bits 7 son para la dirección y 1 es para decidir si queremos leer o escribir datos. Después hay un bit de reconocimiento (ACK) el cual sirve para decirle al dispositivo maestro que el esclavo o el receptor reconocieron la secuencia de bits. Los próximos 8 bits son para enviar los datos con los bits más significativos teniendo de igual manera un bit de reconocimiento. Posteriormente están 8 bits que representan los bits menos significativos de los datos y también al final hay un bit de reconocimiento. Por último se envía el bit de parada, lo que significa que la trama ya fue enviada y dependiendo si se transmitió una lectura o escritura se recibirá respuesta del dispositivo receptor [6].

Referencias:

[1] Sánches, A. y López, J. (2000). Redes. México: McGraw-Hill.

[2] Tanenbaum, A. y Wetherall, D. (2012). Redes de computadoras. México: Pearso Educación.

[3] Kaufman, M. (2000). Networking for Embedded Systems. Disponible en: http://comp.ist.utl.pt/ec-sc/acetatos/ch8-1x-2.pdf

[4] Maset, M. (2003). Protocolo de aplicacion Whatsapp. Disponible en: http://www.uv.es/~montanan/redes/trabajos/WhatsApp.pdf

[5] XMPP Satandards Foundation. (s.f). XMPP. Disponible en: http://xmpp.org/

[6] Irazabal, J. y Blozis, S. (2003). I^2C manual. Philips Semiconductors.

Capítulo 5

5. Sistema de control de acceso

5.1.- Introducción

La primera etapa que se realiza en el sistema de seguridad inteligente es el control de acceso, el cual consiste en el reconocimiento del rostro de las personas para clasificar entre usuarios e intrusos y enviar un mensaje de alerta por medio de Whatsapp al usuario, donde se notificará sólo si se ha detectado un intruso y de lo contrario no se realizará ninguna acción.

La librería OpenCV cuenta con 3 métodos para realizar el reconocimiento de rostro, los cuales son: Eigenfaces, Fisherfaces y mediante histogramas de patrones binarios locales (Local Binary Patterns Histograms o LBPH). Los 3 métodos anteriores realizan el reconocimiento comparando el rostro detectado con una base de datos rostros [1].

El reconocimiento de rostro se realiza en tiempo real y muestra una etiqueta con el nombre de la persona que se detectó. El proceso antes mencionado consta de 4 partes las cuales son: la detección y procesamiento del rostro detectado, procesamiento con los rostros almacenados y el entrenamiento con la base de datos para detectar. En todo el algoritmo que se describirá en este capítulo se utilizó la versión de la librería OpenCV 2.4.9 [1].

El método LBPH necesita una gran cantidad de rostros en la base de datos para realizar el reconocimiento, por lo tanto queda descartado. El método de Fisherfaces y Eigenfaces funcionan de una manera muy similar y es sólo decisión de cada uno seleccionar cualquiera de los dos algoritmos. Fisherfaces funciona muy eficiente cuando se tienen diferentes cambios de iluminación, es decir, se podrá detectar un rostro aunque se tenga poca luz. En cambio, Eigenfaces puede tener algunos problemas cuando no se tiene una iluminación adecuada, pero es muy eficaz cuando se tiene imágenes distorsionadas tanto en la base de datos como en tiempo real. Debido a que la iluminación es fácil de manipular, se opta por usar el método de Eigenfaces [2].

5.2.- Detección de rostro

5.2.1.- Extracción de características

En la sección 2.4 del capítulo 2 se describió el proceso de detección de rostro aplicando el algoritmo de Viola-Jones, el cual también es implementado por la librería OpenCV. Lo primero que se realiza es la extracción de características, las cuales realizarán la función de describir el rostro mediante ciertas figuras geométricas. En la figura 5.1 se muestra las formas usadas por OpenCV que, básicamente, cumplen la función de compararse con una imagen de entrada para detectar un objeto deseado (rostros, formas geométricas, etc.) y responden con un valor de 0 si no se detectó parte del objeto deseado o 1 si se ha detectado alguna parte de dicho objeto. Con la ayuda de la imagen integral, el proceso se realiza rápidamente y se generan muchas respuestas hasta completar el objeto deseado [3].

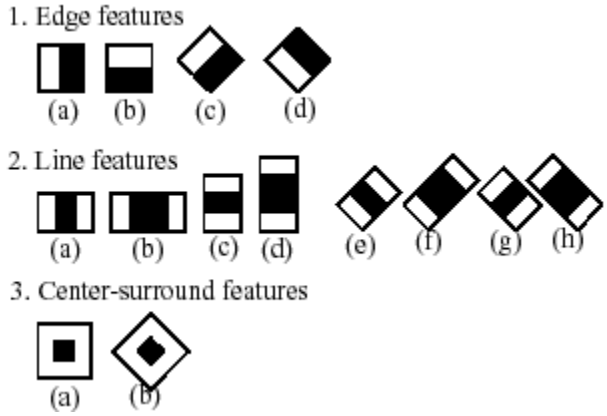

Figura 5.1. Figuras usadas por OpenCV, que reciben el nombre de Haar-like features.

Posteriormente se realiza la clasificación del objeto deseado que en este caso es un rostro. OpenCV cuenta con funciones para clasificar los ojos, la boca, la nariz y rostros de personas, de las cuales ésta última es la que se requiere y se encuentra almacenada en el archivo "haarcascade_frontalface_alt_tree.xml ". Para cargarlo en el código simplemente se crea una variable string con `string fn_haar` y después se coloca la ruta donde se encuentra con `fn_haar="/home/pi/opencv-2.4.9/data/haarcascades/haarcascade_frontalface_alt_tree.xml"`[1] [4].

5.2.2.- Acceder a la cámara

Para ver los rostros es necesario usar una cámara, donde en este caso se utilizó la Pi camera, la cual se observa en la figura 5.2. Cuenta con una resolución de 5 MP (Mega Pixeles) y puede tomar fotos de hasta 2592x1944 pixel. Los videos los puede capturar con las resoluciones de 1080p a 30 fps (fotogramas por segundo), 960p a 45 fps, 720p a 60 fps, 640x480 a 90fps y 320x240 a 120 fps. La Pi camera tiene las dimensiones de 25 mm x 20 mm x 9 mm, la cual se observa en la figura 5.2, por lo tanto es posible tener una cámara de control de acceso de poco volumen y al ser especial para un sistema embebido, su consumo de energía es mínimo [5].

Figura 5.2. Cámara oficial de la Raspberry Pi.

Para controlar la GPU (unidad de procesamiento gráfico) de la Raspberry Pi se utiliza una aplicación llamada MMAL API (Multi-media Abstraction Layer), la cual realiza una interfaz de bajo nivel con la GPU, es decir, se comunica directamente con varios procesos y parámetros que se relacionan directamente con la reproducción de imágenes y videos. También se utiliza una herramienta que generalmente está instalada en el sistema operativo Raspbian por default y es llamada RASPIVID. Dicha herramienta es utilizada para la creación de videos y también será usada en este algoritmo para controlar la cámara de la Raspberry Pi. Por lo tanto, la aplicación MMAL API controlara factores de muy bajo nivel y RASPIVID lo de alto nivel [6].

Para el algoritmo de reconocimiento de rostros se usará librerías de la aplicación MMAL API para crear el componente de la cámara y se pueda mostrar el video en tiempo real de la persona que se quiere detectar. En OpenCV se usa simplemente la función `VideoCapture::open()` para acceder a la cámara, pero en este algoritmo se usó `status = mmal_component_create()`.

5.2.3.- Características de la imagen para la detección de rostro

Cuando se inicializa la cámara y se comienza la captura de video, se debe de tomar en cuenta ciertas consideraciones al momento que el programa toma cada imagen por segundo que está capturando el video. Dichas consideraciones son características particulares que debe de tener cada imagen capturada para ser procesada correctamente con el clasificador "haarcascades" que es para detectar el rostro.

Lo primero que se debe realizar es tener la imagen capturada a escala de grises. En este algoritmo, para aumentar la velocidad de captura y vista previa del resultado del proceso, se usa solamente la escala de grises. Lo anterior se realiza configurando el video que captura la herramienta RASPIVID con la función `state->graymode = 1`. La imagen se captura con la función `cvCreateImage()` y posteriormente es almacenada en una variable de tipo Mat (Mat es un tipo de objeto disponible en la librería OpenCV que almacena componentes de las imágenes) llamada "gray". También la imagen debe contar con un tamaño adecuado para que la función de detectar rostro sea más eficiente. Generalmente dicha función detecta rostros que tengan un tamaño de 20x20 pixeles, por lo tanto la imagen capturada debe ser mayor a esa cantidad. En la documentación de la librería OpenCV se recomienda un tamaño igual o superior a 240x240 pixeles, pero dicho tamaño también dependerá de la resolución que se utilice desde la cámara. En este caso, se utilizó una resolución de 320x240 pixeles [7] [8].

5.2.4.- Procesamiento del rostro detectado

Cuando la imagen ya ha sido transformada a escala de grises y con el tamaño adecuado, se prosigue a realizar la detección de rostros usando la función `CascadeClassifier.detectMultiScale(parámetros)` la cual se basa en el algoritmo de Viola-Jones para realizar la detección. Los parámetros más importantes que se tienen en la función antes mencionada son los siguientes: [7]

- Imagen: Variable que contiene la imagen con el rostro a detectar. Debe ser una imagen de tipo CV_8U (8 bit por pixel y cada pixel puede tener un valor de 0 a 255, es el tipo de imagen que tiene por default los objetos Mat). En este trabajo la variable tiene el nombre de "gray" [1].

- Objeto: Variable donde se almacenará la localización del rostro detectado. En el código que se usó, el objeto se llama "faces" [1].

- Factor de escala. Determina la escala en que se debe buscar la imagen. Con una escala grande, la detección se puede realizar más rápido, pero con posibles errores y con una escala normal, la detección toma mayor tiempo pero es más precisa. En este trabajo se utilizó 1:1, que es la escala normal y es la típica que se usa [1].

- Bandera. Es usada para optimizar la detección de objetos, agregando funciones para ese fin. En las nuevas versiones de la librería OpenCV no se usa ese parámetro, pero en este trabajo se usó una función llamada CV_HAAR_SCALE_IMAGE, la cual asigna memoria directa cuando se almacena un rostro detectado, por lo que provoca más velocidad en el algoritmo de detección [1].

- Tamaño. Es el tamaño máximo y mínimo que puede tener una cara detectada. Por default el tamaño es de 20x20, pero como la cámara de la Raspberry pi se utilizará a una distancia cercana, se recomienda un valor mayor, por lo que en este caso se selección el valor de 80x80 pixeles [1] [7].

Después de utilizar la función escrita anteriormente, se detecta fácilmente un rostro y las coordenadas se almacena en una variable llama “faces”, la cual posteriormente será utilizada para compararse con rostros almacenados. Ya que se ha detectado un rostro, la imagen es recortada, escalada y guardada en una variable con el nombre de “face_resized”. Al rostro detectado se le dibuja un rectángulo amarillo, el cual se muestra en el video de vista previa que se genera al ejecutar el programa.

5.3 Procesamiento de los rostros almacenados

El algoritmo de reconocimiento de rostro puede distinguir el rostro de varias personas, siempre y cuando sus rostros se encuentren en la base de datos. La eficacia que tendrá el programa de reconocimiento de rostro depende mucho de los procesos a realizarse en las imágenes, ya sea recortarla, cambiarla a escala de grises, escalarla, procurar que los ojos estén alineados, etc., por lo tanto lo más conveniente es realizar dichos procesos externamente, es decir, evitar que dichos procesos se realicen en el programa principal para obtener un mejor rendimiento. Las imágenes de la base de datos tendrán características especiales que se deben tomar en cuenta para obtener mejores resultados en el reconocimiento de los mismos [1] [8].

5.3.1 Preparación de la base de datos

Con la ayuda de la librería OpenCV, se puede crear fácilmente un programa que genere la base de datos a partir de cualquier imagen y dicho programa se ejecutara sólo una vez. El programa que se utiliza para la preparación de la base de datos realiza las siguientes funciones: [7]

- Cambiar el tamaño de la imagen. El primer proceso que realiza el programa es dar un tamaño adecuado a todas las imágenes detectadas para que se pueda tener un mejor rendimiento en los siguientes procesos. El primer proceso se realiza con la función `resize()`[7].

- Transformar a escala de grises. En el capítulo 2 se mencionó que el procesamiento de imágenes siempre será más rápido y eficiente cuando se tengan las imágenes a escala de grises, por lo que en este caso no es la excepción. Transformar una imagen a escala de grises es sencillo con OpenCV y se realiza con la función `cvtColor(frame, grayframe, CV_BGR2GRAY)`, en donde "frame" es la imagen original, "grayframe" es la imagen ya convertida y CV_BGR2GRAY es el parámetro para convertir la imagen a escala de grises [7].

- Detectar rostro. Utilizando la función `face_cascade.detectMultiScale()`, la cual ya ha sido explicada en la sección anterior de este capítulo. Si un rostro es detectado se prosigue a ejecutar el siguiente procesos y en caso contrario, el programa muestra un mensaje de error [7].

- Detectar ojos. En este caso, se toma en cuenta la localización de los ojos y si la persona de la imagen usara lentes, también es posible detectar su rostro, gracias a que OpenCV cuenta con las herramientas necesarias para detectar ojos y rostros cuando una persona use lentes. Los archivos que se deben inicializar para usar los clasificadores necesarios son haarcascade_eye_tree_eyeglasses.xml y haarcascade_eye.xml. Para detectar los ojos se usa la función `eyes_cascade.detectMultiScale()` la cual tiene los mismos parámetros que la detección de rostros. Se compara el tamaño de los ojos detectados, por lo que si no resultan igual a cierta medida establecida, se utiliza `glasses_cascade.detectMultiScale()` para afirmar si los ojos detectados usan lentes [7].

- Recortar imagen. Cuando el rostro y los ojos son detectados, se recorta la imagen de tal manera que sólo permanezca el rostro y lo anterior se realiza con la función `CropFace()`. El tamaño del rostro será configurable al momento de ejecutar el programa. Lo recomendable es un tamaño de 100x100 pixeles [8].

- Ecualizar histograma de la imagen. La ecualización del histograma consiste en transformar de manera uniforme la intensidad de una imagen, que en este caso es que la intensidad de la escala de grises. Este proceso es opcional y se puede desactivar al ejecutar el programa. La ecualización del histograma se realiza con la función `equalizeHist( grayframe, grayframe)`,donde el primer "grayframe" es la imagen de entrada y el otro es la imagen de salida [8].

Al momento de ejecutar el programa de la preparación de las imágenes, se cuentan con ciertos parámetros los cuales son: el porcentaje de recortado de la imagen, el tamaño de la imagen final, prefijo de la imagen, tamaño de la imagen de entrada y activar o desactivar la ecualización de histograma de la imagen. Cuando se crea la imagen final, se nombra con el prefijo que se desee. Por ejemplo, si a la imagen se le colocara el prefijo "im", todas las imágenes finales obtendrán de nombre "im1.jpg", "im2.jpg" y así sucesivamente. La figura 5.3 muestra un ejemplo de la base de datos de una sola persona.

Figura 5.3. Base de datos de rostros.

Después hay que crear un documento de texto que contenga la dirección de las imágenes utilizando el siguiente formato: "/home/pi/camcv/base/nombre_imagen.jpg;0", donde el

parámetro “0” significa una etiqueta que estará ligada a una persona, por lo que si se tienen muchas imágenes de una sola persona, se debe poner sólo su etiqueta. El documento de texto es leído por el programa y carga las imágenes en una pila con el nombre de “images” y las etiquetas en “labels”. En la figura 5.4 se observa un ejemplo del documento de texto que se debe crear [1].

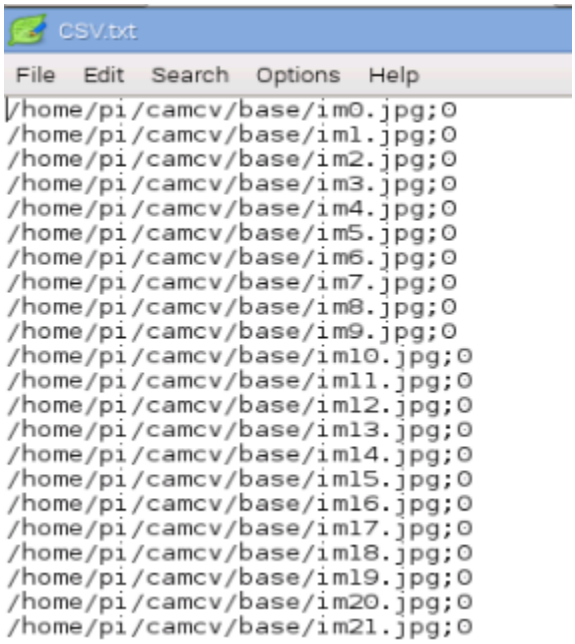

Figura 5.4. Documento de texto con rostros de la base de datos.

5.4.- Entrenamiento y reconocimiento de rostros

Si la base de datos esta lista, el siguiente proceso es obtener parámetros importantes de las imágenes para ser comparadas con las imágenes detectadas en tiempo real y por último aplicar funciones de reconocimiento de rostro.

5.4.1.- Modelo Eigenfaces

Eigenfaces constituye uno de los primeros intentos para definir el espacio-rostro, de modo que al comprimir el dato facial en una pequeña y compacta firma biométrica que pueda servir como modelo para un reconocimiento facial basado en el Análisis de Componentes

Principales. Dicho análisis es un modelo lineal estadístico que reduce o comprime variables de tal forma que existan menos variables con la menor pérdida de información posible. Las nuevas imágenes generadas son llamadas Eigenfaces y se muestran en la figura 5.5 [9].

Figura 5.5. Imágenes de tipo Eigenfaces.

La figura 5.6 muestra el algoritmo usado en el método Eigenfaces, donde en primera instancia se tiene una base de dato de imágenes para extraer información de ellas y ese proceso se llama entrenamiento. Después dichas imágenes son transformadas en Eigenfaces (E). Enseguida se calcula el vector de peso (W) de la contribución de cada Eigenfaces en las imágenes originales. Luego se realiza el proceso antes mencionado pero en vez de usar las imágenes de la base de datos se calcula el vector de peso (Wx) de una imagen de entrada desconocida (X). Posteriormente se comparan los dos vectores Wx y W mediante el promedio de la distancia (D) que hay entre ellos. Si la diferencia entre ellos es muy lejana, el resultado no se considera como verdadero y es descartado, pero si la diferencia entre ellos es cercana, el resultado es guardado y se genera un resultado correcto [9].

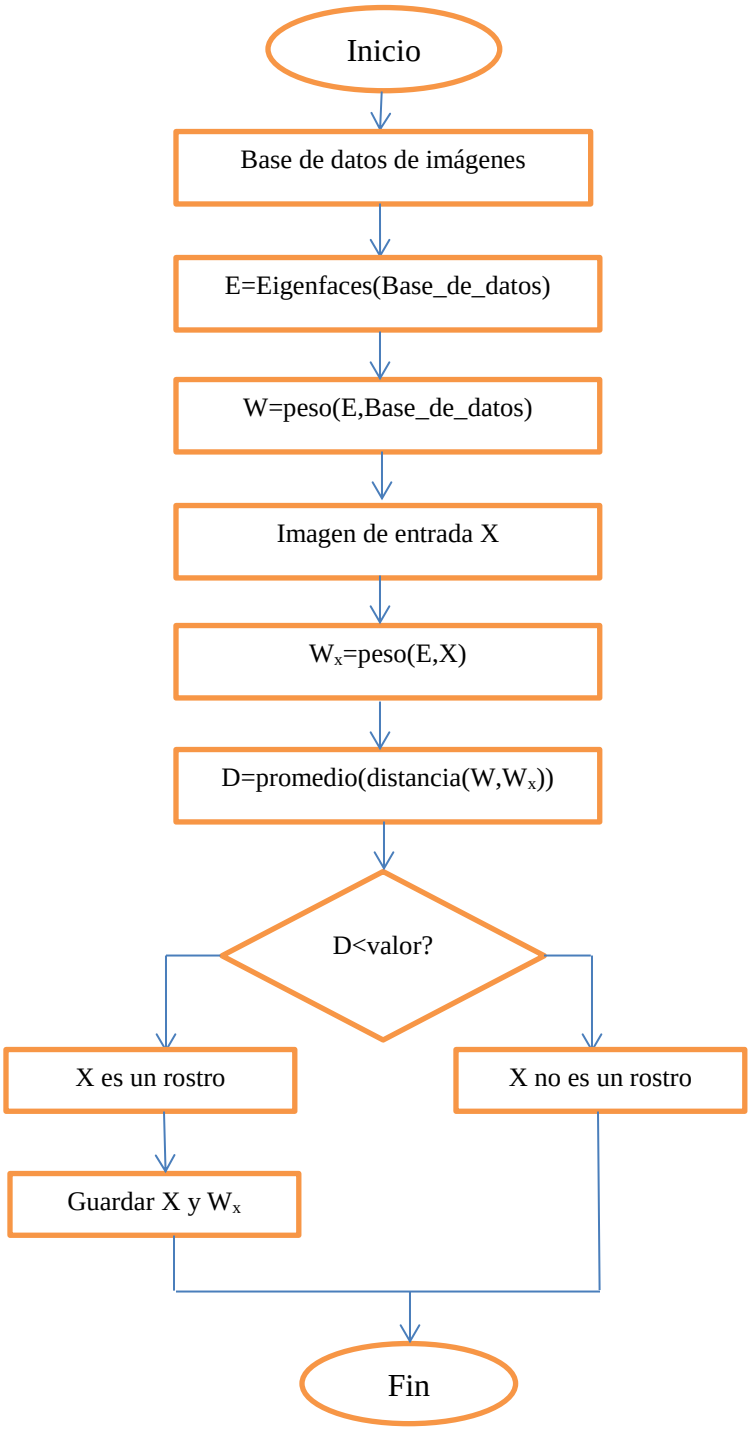

Figura 5.6. Algoritmo de Eigenfaces.

5.4.2.- Funciones de entrenamiento y predicción

Para entrenar con la base de datos, primero es necesario crear un modelo nuevo de Eigenfaces con la función `Eigenfaces model` y para entrenar con la base de datos simplemente se usa la función `model.train(images, labels)`, donde el parámetro "images" es la pila que contiene las imágenes de la base de datos y el parámetro "labels" la pila de las etiquetas de cada persona [8].

Posteriormente se realiza la predicción del rostro la cual se realiza con la siguiente función: `model.predict(face_resized,prediction,predicted_confidence)`, donde el parámetro "face_resized" es el rostro ya procesado leído por la cámara, "prediction" es la etiqueta del rostro que se detectó y "predicted_confidence" es la coincidencia de la predicción del rostro o también se puede traducir como la distancia que hay entre dos Eigenface la cual se describe en la sección 5.4.1. Un valor recomendado para esa distancia es de 4500, por lo que en el código se compara que, si el resultado es mayor a 4500, la coincidencia del rostro es aceptable. Después de la misma manera se comparará la etiqueta resultante. Ya que el rostro sea reconocido, se dibuja un cuadro verde alrededor y se muestra la etiqueta con el nombre a quien pertenece. La figura 5.7 muestra el resultado que se obtiene al ejecutar el programa descrito en este capítulo [8].

Figura 5.7. Programa de reconocimiento de rostros.

En caso de que el rostro no coincida con ninguno almacenado en la base de datos, se dibuja el cuadro verde alrededor del rostro sin mostrar ninguna etiqueta y se escribe en un documento de texto la palabra intruso, el cual será leído por otro programa para enviar notificación de intruso mediante la mensajería instantánea Whatsapp. Dicha notificación pudo ser incluida dentro de este código de reconocimiento de rostro, pero al momento de enviar la alerta surge un problema de rendimiento que afecta directamente al algoritmo de reconocimiento de rostro y la retrasa considerablemente, por lo que se opta por separar el programa del envío de notificación con el programa de reconocimiento de rostro.

Referencias:

[1] Open Source Computer Vision Library. (2015). *Face Recognition with OpenCV*. Recuperado de http://docs.opencv.org

[2] Belhumeur, P., Hespanha, J., Kriiegman, D. (1997). *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. Recuperado de http://ieeexplore.ieee.org

[3] Kadir, K., Kamaruddin, M., Nasir, H., Safie, S., y Bakti, Z. (2014). *A Comparative Study between LBP and Harr-like features for Face Detection Using OpenCV*. Trabajo presentado en 4th International Conference on Engineering Technology and Technopreneuship, Kuala Lumpur, Malaysia. Recuperado de: http://ieeexplore.ieee.org

[4] Arubas, E. (2013). *Face Detection and Recognition (Theory and Practice)*. Recuperado de: http://eyalarubas.com/face-detection-and-recognition.html

[5] OmniVision Technologies. (2009). *Color CMOS QSXGA 5MP image sensor with OmniBSI technology*. Recuperado de: www.ovt.com

[6] Embedded Linux Wiki. (2009). *Rpi Camera Module*. Recuperado de: http://elinux.org/Rpi_Camera_Module

[7] Lelis, D. (2012). *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing Ltd. Recuperado de: https://www.packtpub.com/

[8] Raufast, P. (Junio de 2013). OpenCV and Pi Camera Board [Publicación de blog]. Recuperado de: https://thinkrpi.wordpress.com/opencv-and-pi-camera-board/

[9] Pissarenko, D. (2002). *Eigenface-based facial recognition*. Recuperado de: http://citeseerx.ist.psu.edu

Capítulo 6

6. Sistema de seguridad

6.1.- Introducción

Una parte muy importante de cualquier sistema de seguridad es el monitoreo de las partes de fácil acceso que puede tener un delincuente. Las puertas y ventanas son los accesos más fáciles para entrar a un lugar, por lo que es recomendable monitorear continuamente esas zonas. Hoy en día existen sistemas que hacen ese trabajo, pero la notificación que se realiza es poco primitiva, ya que se realiza mediante alarmas auditivas. También existen servicios que ofrecen algunas empresas los cuales consisten en que ellos mismos monitorean los accesos, pero dicho servicio suele ser costoso. Con la ayuda de una Raspberry Pi, se pueden monitorear las puertas y ventanas pero con la diferencia de que la activación puede ser manipulada y en este caso, enviarse como notificación por mensajería instantánea. Debido a que el monitoreo de puertas y ventanas es bien conocido por cualquier delincuente, también se agregará el monitoreo por vibración y de igual forma enviar una notificación.

6.2.- Puertos GPIO de la Raspberry Pi

Los puertos de entrada/salida de propósito general o GPIO (General Purpose Input/Output) de la Raspberry Pi son una potente herramienta que se controlan mediante software y por lo tanto, pueden realizar cualquier acción deseada. La figura 6.1, muestra los puertos de la Raspberry Pi, donde se observa que tiene 40 puertos, con la posibilidad de realizar comunicación serial gracias a los pines Tx y Rx y comunicación I^2C con los pines SDA y SCL. También cuenta con varios puertos para propósito general, en donde algunos será utilizados para el monitoreo de sensores. Una especificación muy importante que se debe tener en cuenta con los puertos GPIO es que funcionan con 3.3 volts, por lo que si se usan los típicos 5 volts, los puertos podrían dañarse [1].

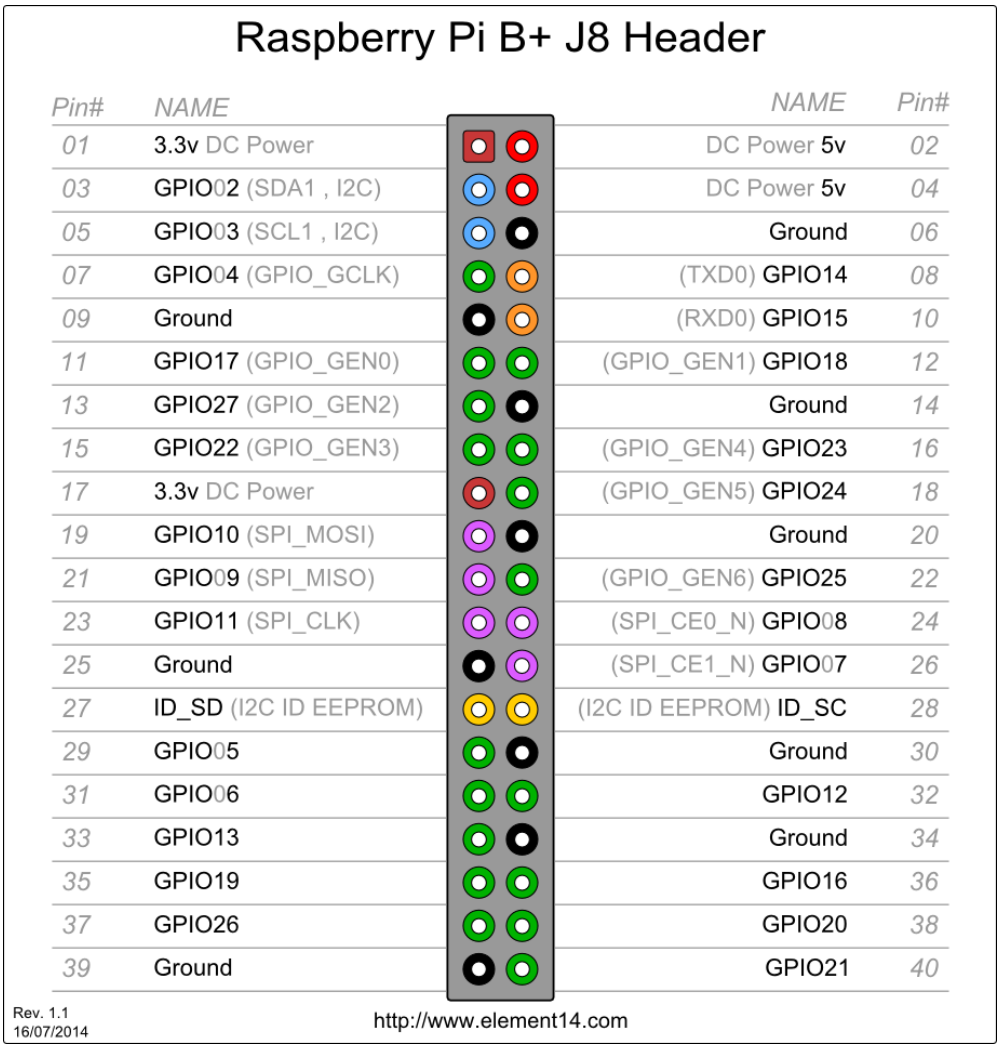

Figura 6.1. Puertos GPIO de la Raspberry Pi.

Existen 3 formas para controlar los puertos, por medio del lenguaje de programación de consola llamado Bash, con el lenguaje C y con el lenguaje Python. Se opta por usar el lenguaje de programación Python debido a la facilidad de programación al realizar procesos pequeños, por ejemplo el monitoreo de pines. También otra ventaja que tiene es el soporte que hay de Python hacia los puertos de la Raspberry Pi, los cuales se adaptan sin ningún problema a dicho lenguaje de programación. Hay 2 formas en que los pines de la Raspberry Pi pueden ser numerados, conforme a la colocación física (pines del 1 al 40, obsérvese figura 6.1) o por numeración de la compañía Broadcom que es la creadora del circuito integrado que contiene los pines (desde GPIO 2 hasta GPIO 27) [2].

6.2.1.- Monitoreo de sensores magnéticos

Un par se sensores magnéticos son utilizados en una puerta y una ventana para su monitoreo. Para usar los puertos de la Raspberry Pi con Python, primero se debe agregar la librería usando el comando `import RPi.GPIO as io`, donde también se simplifica el nombre de la librería usando solamente "io". Se utilizó la numeración conforme al circuito integrado de Broadcom, por lo que se requiere colocar la función `io.setmode(io.BCM)` para activar dicha configuración. Forzosamente se debe decidir una forma de numeración, ya que si no se declara ninguna de las 2 disponibles, los pines no se podrán usar [2].

Los pines GPIO 13 y GPIO 19 se usaron para el monitoreo de una puerta y una ventana los cuales, son nombrados sensor1 y sensor2 usando las funciones `sensor1= io.input(13)` y `sensor2 = io.input(19)`. Los sensores se conectan en pull-up, por lo que se preguntará si las entradas son iguales a 0 para ejecutar una acción, que en este caso es el envío de notificación.

6.2.2.- Monitoreo de la vibración

Hay varias formas de detectar la vibración, una de las más usadas es mediante un acelerómetro, el cual detecta los cambios de movimiento que se generen. En este trabajo se utilizó un acelerómetro de tipo micromecánico o MEMS (Micro-Electro-Mechanical-System) el cual, provoca un cambio en la capacidad de dos condensadores que contiene internamente y que es proporcional movimiento que se detecte, por lo que el voltaje de salida también tendrá una variación [3].

El acelerómetro que se utilizó para detectar la vibración fue el MPU-6050, que se muestra en la figura 6.2. Dicho acelerómetro funciona con un voltaje de 3.3 volts, puede dar tanto salida analógica como digital, donde ésta última se obtiene a partir de un convertidor analógico-digital de 16 bits la obtención de la señal y configuración del dispositivo se realiza mediante la comunicación I^2C, tiene un rango de medición desde +- 2g (la fuerza G= 9.80665 m/s^2) hasta +- 16g [4].

Figura 6.2. MPU-6050.

Gracias a que la Raspberry Pi cuenta con los pines necesarios para realizar una comunicación de I^2C, el acelerómetro puede ser fácilmente controlado mediante Python y de esa manera adaptarse al mismo programa para la detección de los sensores magnéticos. El acelerómetro tiene predeterminada la dirección 0x68, la cual siempre es necesaria al momento de realizar configuraciones por medio de la comunicación I^2C. Con la ayuda de la función para leer un dato `bus.read_byte_data()` o `read_word()` y para escribir `bus.write_byte_data()` se pueden utilizar para manipular los registros. Al momento de detectarse una vibración, se genera una aceleración en los ejes X, Y, Z para posteriormente convertirse a digital y almacenarse en los registros. Dichos registros se encuentran en la dirección 0x3b para un movimiento en X, 0x3c para uno en Y y 0x3d para un movimiento en Z [4].

Cuando se instala el acelerómetro por primera vez en un lugar deseado, es necesario realizar una calibración. Las aceleraciones en X, Y, Z se dan en bits/g, por lo que al momento de obtener los valores, la documentación del MPU-6050 dice que se obtienen 16384 bits/g de sensibilidad, por lo tanto hay que dividir el dato leído en el registro entre 16384 para obtener el resultado en fuerza G y a consecuencia tener mayor control en la calibración. El programa monitorea los 3 ejes, por lo que si uno de los ejes sobre pasa el valor indicado al momento de calibrar, se enviará una notificación.

6.3.- Envío de notificaciones

Hoy en día la mensajería instantánea es la herramienta más común para realizar un intercambio de mensajes de manera rápida y por lo tanto, se utilizará ese medio para el envío de notificaciones que se generen en el control de acceso y sistema de seguridad inteligente. Whatsapp es la aplicación más común que se usa en la actualidad para el intercambio de mensajes y es compatible con la mayoría de los sistemas operativos para los teléfonos celulares, por lo que no será necesario instalar programas adicionales en dichos teléfonos para recibir notificaciones. Whatsapp es un software que funciona con base al protocolo XMPP y por lo tanto, tiene las mismas características que cualquier software de mensajería instantánea que funcione con dicho protocolo [5].

Debido a que Whatsapp sólo es para teléfonos celulares, será necesario el apoyo de una aplicación para que se comunique con sus servidores. Yowsup es una librería desarrollada en Python que permite interactuar con Whatsapp. Para transmitir mensajes usando dicha librería, es necesario un número de teléfono celular para obtener el usuario y contraseña que Whatsapp otorga al momento de utilizarlo o dar de alta el número. Cuando se usa una aplicación basada en XMPP se genera un usuario y contraseña, por lo tanto Yowsup necesita dicha información para poder funcionar. Yowsup cuenta con las herramientas necesarias para registrar un número en el protocolo XMPP [6].

El primer paso a realizar para usar Yowsup es crear un documento de texto de configuración que contenga los siguientes datos:

```
cc=código del país
phone=número de teléfono
id=Esto se deja en blanco
password=Este dato también se deja en blanco
```

Posteriormente se realiza el pedido de la contraseña utilizando el `comando python yowsup-cli registration –c config.txt -r sms`, donde config.txt es el documento de texto de configuración creado y se utilizó la ventana de comandos de Linux como intérprete de Python. Si Yowsup fue ejecutado correctamente, se recibirá un mensaje de texto con un código de 6 números al estilo xxx-xxx que será usado para obtener la contraseña. Con el comando `python yowsup-cli registration -c`

`config.txt -R xxx-xxx` se da de alta el número y se responde en la ventana de comandos con la información de la figura 6.3 [6].

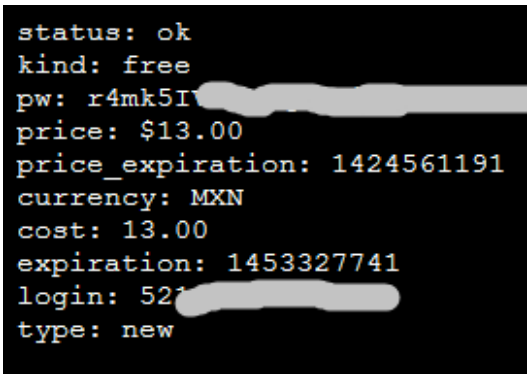

Figura 6.3. Información obtenida al dar de alta un número en Whatsapp.

La información importante que se observa en la figura 6.3 es la contraseña, que es nombrada "pw". Dicha contraseña se debe agregar al documento config.tx en donde dice "password". Después de realizar lo anterior, ya es posible mandar mensajes desde una computadora a un teléfono celular por medio de Whatsapp, usando solamente el comando `python yowsup-cli demos -c config.txt -s número_de_teléfono_destino "mensaje de prueba"` [6].

6.3.1.- Envío de mensaje al detectar intrusos

Una parte importante del sistema del sistema de seguridad es el envío de notificaciones cuando se detecte un intruso. Cuando un rostro que no se encuentre en la base de datos es detectado, se genera la palabra intruso en un documento de texto. El rendimiento del programa de reconocimiento de rostro fue optimizado para la Raspberry Pi por lo tanto, si la notificación de Whatsapp se realiza desde el mismo programa, éste último puede sufrir un pequeño congelamiento y se puede perder el monitoreo de rostro por algunos segundos. Por la razón antes mencionada, se opta por generar las alertas y el uso de sensores con un programa separado y usando Python, es decir, habrá 2 programas corriendo simultáneamente.

La primera fase del sistema de seguridad está compuesta por el reconocimiento de rostros. Un documento de texto llamado identidad.txt será monitoreado por el programa generador de alertas y comparará si lo leído es igual a la palabra intruso. Si es afirmativo, el programa ejecuta la función `yowsup-cli demos -c config.txt -s número_de_teléfono_destino "intruso"`, enviando la palabra "intruso" por Whatsapp al número de teléfono celular que se le especifica.

La segunda fase del sistema consiste en la detección de la activación de los sensores magnéticos. Debido a que los sensores se encuentran en pull-up, el programa monitorea el programa si existen en las entradas de la Raspberry Pi un 0, si es así se ejecuta la función de enviar la notificación. Para los sensores existen 3 condiciones, si se activa el interruptor magnético 1, se envía el mensaje por Whatsapp "sensor 1 activado". Si se activa el interruptor magnético restante, se envía "sensor 2 activado". Por último, si se activan los 2 interruptores magnético se envía "sensor 1 y 2 activados".

La tercera y última fase del sistema consiste en la detección de la vibración. El sensor de vibración que se usará puede ser instalado en una puerta, ventana, muros, etc. En este caso se hizo la prueba en una puerta. Se monitorean los 3 ejes XYZ para detectar la vibración. El programa compara la aceleración de cada eje mencionado con un valor que se tiene como base al momento de realizar la calibración. Debido a que la aceleración es tipo de fuerza relacionado con el movimiento, se monitorea al valor absoluto obtenido del acelerómetro digital para detectar aceleraciones hacia cualquier dirección. Cuando se sobrepasan los valores establecidos, el programa activa la función `yowsup-cli demos -c config.txt -s número_de_teléfono_destino "puerta golpeada"` enviando de esa manera el mensaje de alerta.

Referencias:

[1] Pi Blog. (s.f). *Raspberry Pinout*. Recuperado de http://pi.gadgetoid.com/pinout

[2] Raspberry Pi Tutorials, Videos & Reviews. (2013). *Setting up RPi.GPIO, numbering systems and inputs*. Recuperado de http://raspi.tv

[3] Rincon, R., Ambrosio, R. y Mireles, J. (2013). *Análisis y caracterización de un acelerómetro capacitivo fabricado con tecnología polymump's*. Recuperado de http://www.smctsm.org.mx/

[4] InvenSense Inc. (2013). *MPU-6000 and MPU-6050 Product Specification*. Recuperado de http://store.invensense.com

[5] Maset, M. (2003). *Protocolo de aplicacion Whatsapp*. Disponible en: http://www.uv.es/~montanan/redes/trabajos/WhatsApp.pdf

[6] Tarek, T. (2015). *The python WhatsApp library*. Recuperado de https://github.com/tgalal/yowsup

Capítulo 7

7. Conclusión y trabajos a futuro

Los sistemas inteligentes son un gran avance tecnológico que ayudan a tomar en cuenta ciertos factores que, por problemas de precio y componentes, no se podría lograr fácilmente. En este trabajo se presenta un ejemplo de ello, un control de acceso y seguridad inteligente que cualquier persona puede realizar con conocimientos básicos de electrónica e informática. Se utilizan sistemas que actualmente están teniendo un gran desarrollo en cuanto a su investigación y aplicación, como lo es los sistemas embebidos y la visión artificial.

Hoy en día cualquier sistema que sea inteligente suele tener un precio elevado debido a que realizan actividades cercanas a la de un humano, por ejemplo la visión artificial y el envío de mensajes instantáneos. Este trabajo muestra un sistema inteligente que tiene un precio reducido de inversión y que puede ser modificado con las necesidades que una persona en particular necesite. También puede ser adaptado a casi cualquier lugar y no es necesario un mantenimiento constante, por lo que tampoco necesitará invertir posteriormente.

Este trabajo es sólo una base de sistemas más complejos y que puede ser mejorado con muchas modificaciones en cuanto a software y configuración se refiere. El reconocimientos de rostros y monitoreo de sensores usan un programa individual, por lo que pueden ser adaptados en uno solo usando programación más avanzada y usando solamente un lenguaje de programación. También, se podrían utilizar 2 cámaras, 1 encargada del reconocimiento de rostro y otra encargada de transmitir el monitoreo por internet. Otra acción que se pudiera realizar es crear un programa que realice reconocimiento de rostro y pueda ser monitoreado mediante internet, pero la programación a realizarse sería complicada. Adicionalmente se puede utilizar un módem GSM para enviar mensaje de texto o hacer autollamadas. Por último, el sistema de seguridad funciona realizando una configuración pero no es posible reconfigurarla a distancia, por lo que sería una gran mejora que el sistema de seguridad tuviera la posibilidad de aceptar comandos para que realice tanto configuraciones como acciones adicionales que se pudieran agregar.

Índice de Figuras