DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO INALÁMBRICO DE VARIABLES FÍSICAS MEDIANTE EL EMPLEO DE VISIÓN ARTIFICIAL PARA LA INTERPRETACIÓN DE LAS ESCALAS NUMÉRICAS DE INSTRUMENTOS ANALÓGICOS INDUSTRIALES”

AUTORES: LEANDRO GABRIEL CORRALES TIBÁN
CATHERINE LISETH GÁLVEZ JÁCOME

DIRECTOR: ING. EDWIN PRUNA P.

LATACUNGA

2017

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO INALÁMBRICO DE VARIABLES FÍSICAS MEDIANTE EL EMPLEO DE VISIÓN ARTIFICIAL PARA LA INTERPRETACIÓN DE LAS ESCALAS NUMÉRICAS DE INSTRUMENTOS ANALÓGICOS INDUSTRIALES"** realizado por el señor **LEANDRO GABRIEL CORRALES TIBÁN** y la señorita **CATHERINE LISETH GÁLVEZ JÁCOME**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **LEANDRO GABRIEL CORRALES TIBÁN** y a la señorita **CATHERINE LISETH GÁLVEZ JÁCOME** para que lo sustenten públicamente.

Latacunga, 16 de enero del 2017

ING. EDWIN PATRICIO PRUNA PANCHI
DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Nosotros, **LEANDRO GABRIEL CORRALES TIBÁN**, con cédula de identidad N° 050250394-9, y **CATHERINE LISETH GÁLVEZ JÁCOME**, con cédula de identidad N° 050389207-7, declaramos que este trabajo de titulación **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO INALÁMBRICO MEDIANTE EL EMPLEO DE VISIÓN ARTIFICIAL PARA LA INTERPRETACIÓN DE LAS ESCALAS NUMÉRICAS DE INSTRUMENTOS ANALÓGICOS INDUSTRIALES"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 16 de enero del 2017

Leandro Gabriel Corrales Tibán	Catherine Liseth Gálvez Jácome
C.C.:050250394-9	C.C.: 050389207-7

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros, **LEANDRO GABRIEL CORRALES TIBÁN** y **CATHERINE LISETH GÁLVEZ JÁCOME**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO INALÁMBRICO MEDIANTE EL EMPLEO DE VISIÓN ARTIFICIAL PARA LA INTERPRETACIÓN DE LAS ESCALAS NUMÉRICAS DE INSTRUMENTOS ANALÓGICOS INDUSTRIALES"** cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Latacunga, 16 de enero del 2017

Leandro Gabriel Corrales Tibán

C.C.: 050250394-9

Catherine Liseth Gálvez Jácome

C.C.: 050389207-7

DEDICATORIA

Quiero dedicar este trabajo de investigación principalmente a Dios por concederme la vida y la sabiduría para poder realizarlo, de manera especial y con gran cariño a mis padres Rafael y Lida por esforzarse y estar al pendiente de mi persona brindándome su apoyo y enseñanzas mediante un ejemplo de responsabilidad y perseverancia para alcanzar mis sueños, también a mis hermanos Bryan y Wendy por compartir su cariño sincero y colaborar conmigo en los momentos necesarios, y a mis demás familiares por estar prestos a brindarme un concejo valioso que me impulse a continuar siendo un persona correcta.

CATHERINE

Todo mi esfuerzo lo dedico en principio a Dios por iluminar el camino en cada aspecto de mi vida, a mis padres por su constante e incansable sacrificio orientado a apoyarme en el alcance de mis propósitos, a todos mis familiares que siempre han estado dispuestos a brindarme alegrías y su ayuda de una u otra manera, y en especial, el esmero invertido en éste trabajo lo dedico a mi tío Marco, quien más que un tío fue un amigo, que me incentivó a estudiar ésta interesante carrera llena de posibilidades, y me brindó un desinteresado apoyo al inicio de esta etapa, así como a lo largo del tiempo que me acompañó, que Dios lo tenga en su Gloria.

GABRIEL

AGRADECIMIENTO

A nuestras respectivas familias por su ayuda incondicional y apoyo moral a lo largo de nuestra existencia.

Nuestra más sincera gratitud a cada uno de los docentes del Departamento de Eléctrica y Electrónica y a todos quienes a más de ser excelentes profesionales que nos han sabido compartir sus conocimientos, demuestran día a día su gran calidad humana expresándonos amablemente sus consejos y amistad.

De manera muy especial, agradecemos al Ing. Edwin Pruna por brindarnos generosamente su apoyo y amistad como docente y tutor de este proyecto, y por habernos motivado a involucrarnos en el fascinante mundo de la investigación.

De la misma forma al PhD. Oscar Chang por su confianza y por brindarnos lineamientos importantes en el desarrollo de éste proyecto.

CATHERINE, GABRIEL

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de titulación trata del diseño y desarrollo de un sistema de monitoreo de bajo costo aplicando visión artificial en el reconocimiento y la digitalización de la medida de los instrumentos analógicos comúnmente utilizados en las industrias, como son el termómetro, el manómetro y el rotámetro, que miden las variables de temperatura, presión y flujo respectivamente; presentando la característica de ser un sistema no invasivo que permite obtener datos de las variables en tiempo real a través de una red inalámbrica basada en tecnología WIFI. Los algoritmos correspondientes al procesamiento de imágenes se han programado haciendo uso de las librerías de OpenCV y la aplicación de redes neuronales artificiales en el reconocimiento de formas, para ello se utiliza una plataforma de desarrollo embebida que soporta la compilación y ejecución de códigos escritos en lenguaje C++ sobre un sistema operativo libre; y dónde además las funciones de transmisión inalámbrica están desarrolladas en base al protocolo UDP. Los datos correspondientes a cada variable son visualizados y almacenados en una PC que actúa como un sistema servidor de la red inalámbrica, para ello se ha elaborado un HMI en el software LabVIEW que, además, se complementa con diferentes lenguajes de programación para la creación de una página web orientada al servicio de monitoreo remoto.

PALABRAS CLAVE:

- **PROCESAMIENTO DE IMÁGENES.**
- **REDES NEURONALES ARTIFICIALES.**
- **PLATAFORMAS DE DESARROLLO EMBEBIDAS.**
- **SOFTWARE LABVIEW.**
- **REDES INALÁMBRICAS.**

ABSTRACT

This titulation work is the design and development of a low cost monitoring system by applying artificial vision recognition and scanning measurement of analog instruments commonly used in industries such as thermometer, pressure gauge and flow meter, measuring variables temperature, pressure and flow respectively; presenting the characteristic of being a non-invasive system that allows for variable data in real time over a wireless network based on WIFI technology. The algorithms for the image processing are programmed using OpenCV libraries and application of Artificial Neural Networks for pattern recognition, for this, is used a development embedded platform that supports the compilation and execution of codes written language used C++ on a free operating system; and also where wireless transmission functions are developed based on the UDP protocol. Corresponding to each variable data are displayed and stored on a PC that acts as a server system of the wireless network, for it has developed an HMI in LabVIEW software that also includes different programming languages for creating a website oriented to a remote monitoring service.

KEYWORDS:

- **IMAGE PROCESSING.**
- **ARTIFICIAL NEURAL NETWORKS.**
- **EMBEDDED DEVELOPMENT PLATFORMS.**
- **LABVIEW SOFTWARE.**
- **WIRELESS NETWORKS.**

CAPÍTULO I

1. ASPECTOS GENERALES

1.1. Antecedentes

En la actualidad el desarrollo tecnológico ha crecido exponencialmente, de tal forma que se ha llegado a obtener sistemas computacionales de bajo costo con microprocesadores capaces de ejecutar millones de instrucciones por segundo, haciendo posible el desarrollo de aplicaciones complejas de distintos ámbitos, como es el caso de la visión artificial.

La visión artificial tiene como propósito obtener la información visual de la imagen para extraer características relevantes; existen distintas utilidades que han sido beneficiosas en el campo industrial, donde se integra sistemas de adquisición de imágenes, dispositivos de entrada/salida y redes de ordenadores para el control de equipos y procesos destinados a la fabricación. En general, los sistemas de visión artificial están destinados a realizar inspecciones visuales de alta velocidad y funcionamiento continuo.

Los equipos utilizados en la industria continuamente mejoran su tecnología para asegurar una operación fiable, muchos de ellos son utilizados para medir variables físicas que son de interés en los procesos de producción, sin embargo los costos son muy elevados, incluso en las grandes compañías la migración a tecnologías de última generación no ocurre de forma instantánea y frecuente, debido a que financieramente no resulta conveniente si los sistemas existentes se mantienen en funcionamiento.

Existen proyectos de investigación enfocados a brindar soluciones similarmente efectivas pero a menor precio, debido a que los recursos por los que está conformados se basan en plataformas embebidas que manejan el concepto de hardware abierto y software libre, por lo que se presentan como una alternativa al ordenador, pues su capacidad de procesamiento es suficiente para ejecutar un sistema operativo y algoritmos complejos. Adicionalmente, están

construidos con características que permiten la conexión de periféricos y establecer diferentes protocolos de comunicación.

1.2. Planteamiento del problema

Los equipos utilizados en la industria para medir, monitorear y registrar una variable física poseen un costo muy elevado, el cual no resulta favorable para las pequeñas y medianas empresas. Considerando que el desarrollo tecnológico es indispensable para el mejoramiento y abaratamiento de costos de producción, lo ideal es que todas las entidades se acojan a dicha tendencia para verse beneficiadas por un progresivo crecimiento; sin embargo, en muchos de los casos no se produce tal desarrollo a causa de la escasez de recursos para la repotenciación tecnológica en los procesos.

En varias empresas utilizan medidores analógicos de distintas variables físicas como son voltaje, corriente, presión, temperatura, flujo, nivel, entre otras; donde se requiere registrar la medida manualmente, con el fin de utilizar los datos para actividades de mantenimiento de la maquinaria, verificación del desenvolvimiento normal del proceso o el control de consumo de los insumos utilizados en la planta, en algunos casos una o varias personas, debidamente capacitadas, deben realizar la recolección de los datos, en ocasiones los lugares donde se debe acudir son de difícil acceso, por lo que es necesario tomar ciertas medidas de seguridad que, sin embargo, no mitigan al cien por ciento los riesgos existentes.

El tiempo empleado en el registro manual, y en la mayoría de casos un posterior traspaso de los datos a un formato digital, también es un factor a tener en cuenta, pues esta tarea debe realizarse en un período de tiempo definido acorde a las necesidades, lo cual involucra el empleo de tiempo de trabajo de los colaboradores encargados, mismo que podría ser mejor aprovechado en otras actividades que beneficien a la compañía.

Además, otro aspecto importante a ser considerado es que el sistema de registro de datos de forma manual implica la utilización de papel, debido a que la actividad debe llevarse a cabo cada cierto tiempo paulatinamente se van sumando grandes cantidades que en algún momento deberán ser desechadas, esto se traduce en destrucción de bosques y contaminación del medio ambiente, a pesar de que existan políticas de reutilización y reciclado de papel dentro de la institución.

1.3. Justificación

El proyecto de investigación que se pretende realizar está orientado a brindar soluciones de monitoreo y registro de variables a un costo relativamente bajo, aprovechando las características que actualmente presentan los sistemas computacionales embebidos, y considerando que los sistemas de visión artificial no necesariamente requieren de cámaras de características especiales, sino que puede hacerse uso de cámaras web que son muy comunes en el mercado.

En el área de la instrumentación existen equipos digitales de medición con características que les brindan confiabilidad en su operación, la cual está estandarizada y permite escalabilidad entre la tecnología empleada en una industria, el sistema de monitoreo inalámbrico por visión artificial propone una eficacia próxima a equipos especializados mediante el empleo de tecnología embebida, y al mismo tiempo ser adaptable a software y hardware existentes y privativos, para aprovechar las utilidades de sistemas que se encuentran en operación dentro de un proceso.

El beneficio que brinda el proyecto no solo se presenta en el ahorro al emplear tecnología de cómodo acceso, sino también en el hecho de que se consigue una mejor organización de la información de las mediciones de las variables de los procesos dentro de la empresa; esto significa que los colaboradores, encargados del funcionamiento de los procesos, tengan el continuo conocimiento de cómo se encuentra operando la planta.

Adicionalmente, el registro de los datos puede utilizarse para llevar a cabo actividades de mantenimiento predictivo, e identificar a tiempo anomalías para ejecutar un mantenimiento correctivo o preventivo. Por otra parte, el personal administrativo puede hacer uso de la información para llevar el control del empleo de insumos y evaluar económicamente el rendimiento de la producción y los procesos.

Es importante resaltar que mediante este sistema de monitoreo y registro digital se impulsa el concepto de desarrollo sostenible, debido a la reducción del papel empleado en los registros técnicos; y propone la intención de salvaguardar la integridad del talento humano al no requerir acudir constantemente a los sitios de difícil acceso para registrar una medición.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un sistema de monitoreo inalámbrico de variables físicas mediante el empleo de visión artificial para la interpretación de las escalas numéricas de instrumentos analógicos industriales.

1.4.2. Objetivos específicos

- Investigar acerca de los recursos de hardware y software necesarios en la implementación de un sistema de visión artificial que emplee tecnología de costo relativamente bajo.
- Desarrollar algoritmos para la interpretación de las escalas numéricas de los instrumentos analógicos industriales basados en la utilización de redes neuronales artificiales.
- Desarrollar un algoritmo de transmisión de datos de forma inalámbrica por WIFI de tal forma que el sistema opere en tiempo real.
- Diseñar un programa en el software LabVIEW que actúe como un sistema servidor de la red inalámbrica para organizar, almacenar, y presentar los datos a nivel local y remoto.

- Comparar el funcionamiento del sistema implementado con el de los equipos electrónicos industriales existentes en el Laboratorio de Redes Industriales y Control de Procesos de la Universidad de las Fuerzas Armadas ESPE extensión Latacunga.

1.5. Delimitación

Análisis del comportamiento del sistema frente a equipos construidos específicamente para sensar variables físicas, con el fin de determinar si es conveniente la operación en plantas industriales; considerando el desarrollo de algoritmos de procesamiento de imágenes óptimos basados en redes neuronales artificiales, ejecutados sobre plataformas embebidas y aplicados a la lectura de la magnitud de una variable sobre un instrumento analógico que se encuentra en operación en una determinada planta didáctica, ubicada en el laboratorio de Redes Industriales y Control de Procesos de la Universidad de las Fuerzas Armadas ESPE extensión Latacunga, bajo ambientes de iluminación controlada.

CAPÍTULO II

2. FUNDAMENTACIÓN TEÓRICA

2.1. Instrumentos analógicos industriales

Dentro del entorno industrial existen numerosas variables que deben ser detectadas de forma simultánea, de modo que, para adquirir información de forma visual se utilizan instrumentos indicadores con respuesta analógica y digital; siendo los instrumentos analógicos industriales equipos que basan su funcionamiento en principios mecánicos, generalmente poseen una escala graduada fija y un índice móvil, por los cuales cuantifican y proporcionan información necesaria, y de forma comprensible para el operador, del estado de variables físicas como por ejemplo presión, temperatura, flujo, voltaje y amperaje. (Dunn, 2005; Rivas, 2007)

2.1.1. Clases de instrumentos analógicos

Existen diferentes maneras de clasificar a los instrumentos de medición, en cada una se consideran distintos parámetros; sin embargo, en este trabajo se considera la clasificación en función de la variable de proceso donde son empleados, es decir, la atención se centra específicamente en las señales medidas sin contemplar el funcionamiento interno; resultando la existencia de instrumentos de nivel, humedad y punto de rocío, pH, frecuencia, velocidad, fuerza, entre otros. (Creus Solé, 2011; Calderón & Sánchez Montero, 2004)

Algunos de los instrumentos más utilizados en la industria, y a la vez los empleados en este trabajo, se describen a continuación.

a. Rotámetro

El rotámetro (Figura 1) es empleado en la medición de flujo o caudal; corresponde a los elementos de área variable, caracterizados por el cambio de área que se produce entre el elemento primario en movimiento y el cuerpo del

medidor. Se asimilan a una placa-orificio de diámetro interior variable dependiendo del caudal y de la fuerza de arrastre producida por el líquido.

En el rotámetro, un flotador cambia su posición dentro de un tubo en relación al flujo del líquido; el fluido ingresa por la parte inferior del tubo y hace que el flotador suba hasta que éste se encuentre en equilibrio entre su peso, la fuerza de arrastre del fluido y la fuerza de empuje del fluido sobre el flotador. El caudal depende de la viscosidad y peso específico del líquido, así como de los valores de la sección interior del tubo que cambia según sea el punto de equilibrio del flotador.

Los rotámetros son adecuados para la medida de pequeños caudales de líquidos limpios, llegando a valores máximos que pueden alcanzar los 3.5 m^3/min en agua y 30 m^3/min en aire; no obstante, pueden utilizarse en la medida de mayores caudales instalándose como rotámetros bypass en la tubería. (Creus Solé, 2011; Caibe Yanzapanta, 2012)

Figura 1: Rotámetro.

Fuente: (PCE Instruments, 2016)

b. Manómetro

Existen diferentes tipos de manómetros industriales (como el que se indica en la Figura 2), muchos de ellos basan su funcionamiento en un tubo de Bourdon, que es un tubo de sección elíptica que forma un anillo casi completo, cerrado por

un extremo. Al aumentar la presión en el interior del tubo, éste tiende a enderezarse y el movimiento es transmitido a la aguja indicadora, por un sector dentado y un piñón. La forma, el material y el espesor de las paredes del tubo Bourdon dependen de la presión que se requiera medir.

Este instrumento mide la presión relativa, ya que la presión del fluido se encuentra dentro del elemento, mientras que en el exterior actúa la presión atmosférica. La exactitud de este tipo de manómetros depende en gran parte del tubo, por esta razón los tubos Bourdon son fabricados con normas estrictas por parte de los fabricantes. (Creus Solé, 2011)

Figura 2: Manómetro.

Fuente: (WIKA, 2016)

c. Termómetro

Los termómetros industriales, como el indicado en la Figura 3, generalmente son de tipo bimetálicos, que se fundamentan en el distinto coeficiente de dilatación de dos metales diferentes, tales como latón, monel o acero y una aleación de ferroníquel laminados conjuntamente; las láminas bimetálicas pueden ser rectas o curvas, formando espirales o hélices.

Los termómetros, en la mayoría de casos, contienen pocas partes móviles, sólo la aguja indicadora sujeta al extremo libre de la espiral o hélice y el propio elemento bimetálico. El eje y el elemento están sostenidos con cojinetes, y el

conjunto está construido con precisión para evitar rozamientos. No hay engranajes que exijan un mantenimiento. El uso de termómetros industriales es admisible para servicio continuo con exactitudes que llegan a ±1%. (Creus Solé, 2011)

Figura 3: Termómetro.

Fuente: (WIKA, 2016)

2.1.2. Importancia de los instrumentos analógicos industriales

Los instrumentos analógicos industriales todavía son ampliamente utilizados en las industrias a nivel de campo, esto se debe a que en la mayoría de los procesos industriales existen parámetros críticos que deben ser monitoreados, y es crucial mantener un seguimiento de los mismos para evaluar el estado de la planta; adicionalmente se ha determinado que los operadores e inspectores son muy propensos a detectar, de forma más fácil, la deflexión de un indicador que reconocer el cambio de un valor en un display digital, lo cual resulta favorable en la inspección continua y constante de variables para evitar la devastación de un proceso. (Jaffery & Dubey, 2013; Yang, y otros, 2014; Gellaboina, Swaminathan, & Venkoparao, 2013)

La continuidad de la utilización de instrumentos analógicos también se debe a que los fabricantes los construyen con características apropiadas para cada

situación que se presenta en varias industrias, por lo que existe una combinación de un alto grado de tecnología de medición, operación simple, robustez y flexibilidad. Además, se presentan muchas otras ventajas en la utilización de estos instrumentos a más de la visualización fácil y rápida de la disminución o aumento de la variable, entre ellas están el bajo costo, el no requerimiento de fuentes de alimentación y la facilidad de adaptación a diferentes tipos de escalas. (WIKA, 2016; Hernández, 2012)

2.1.3. Problemas que presentan los instrumentos analógicos

Existen algunas desventajas e inconvenientes en la utilización de instrumentos analógicos industriales, entre ellas se considera que las lecturas por parte de los operadores presentan errores cuando el instrumento presenta varias escalas, no pueden emplearse como parte de un sistema de procesamiento y adquisición de datos de tipo digital, así como que en ciertas circunstancias la rapidez de lectura es baja porque depende del observador. De igual manera varios instrumentos se ven afectados por agentes externos, como por ejemplo, pueden proporcionar medidas erróneas cuando la temperatura ambiente está fuera del rango permitido, y/o producirse un deterioro por la existencia de vibraciones mecánicas, corrosión o vapor en exceso. (Hernández, 2012; WIKA, 2013; WIKA, 2016)

Los problemas que típicamente se presentan y son de mayor consideración tratan del error de paralaje, producido cuando la lectura se realiza en el momento que el operador no se encuentra en una línea de observación perpendicular al plano de la escala y el indicador del instrumento; así como el error de interpolación que se presenta cuando el observador redondea la lectura que ha realizado debido a que índice del instrumento no coincide de forma exacta con la graduación de la escala. (Creus Solé, 2011; Higuera)

2.1.4. Industrias que utilizan instrumentos analógicos

Debido a que las características de los instrumentos analógicos industriales les permiten operar en entornos extremos, éstos son empleados en varias

industrias de procesos críticos como de petróleo y gas, combustibles alternativos, centrales eléctricas, refinería y petroquímica; así también en aplicaciones generales como instalaciones hidráulicas móviles, operaciones de aguas residuales, sistemas de Calefacción, Ventilación y Aire Acondicionado (HVAC), refrigeración y ventilación de alimentos y bebidas, entre otros. (WIKA, 2016)

En la mayoría de procesos intervienen diferentes variables, es importante mencionar que una de las que aparecen con más frecuencia es la presión, por lo que los manómetros generalmente son los instrumentos más utilizados; por otra parte, los rotámetros son ampliamente utilizados en procesos químicos, ya que requieren de líquidos que deben ser bombeados y almacenados, como por ejemplo, en la industria de obtención de carbono activado, alimenticia y farmacéutica. (WIKA, 2016; Caibe Yanzapanta, 2012; Vacca, Scaglia, Serrano, Godoy, & Mut, 2014)

2.2. Sistemas de visión artificial

La visión artificial o visión por ordenador es una rama de la inteligencia artificial que tiene por objeto modelar la imagen digital suministrada por un sensor de forma que un ordenador sea capaz de interpretarla y utilizarla como base para la toma de decisiones de algún proceso. De forma más precisa, trata de la capacidad para deducir la estructura y las propiedades del mundo tridimensional a partir de imágenes bidimensionales. (Escribano Sánchez & Escardino Malva, 2003; Prieto, Febres, Cerrolaza, & MIquelarena, 2010)

Un sistema de visión artificial es un sistema autónomo que reemplaza algunas de las tareas del sistema de visión humano, siendo capaz de extraer información del entorno para realizar tareas o actividades definidas previamente que pueden estar relacionadas a la identificación e inspección de objetos y reconocimiento de caracteres en una imagen hasta la interpretación tridimensional de escenas complicadas. (Valencia López & Abril Cañas, 2007)

Así también es un área científica-técnica en continuo desarrollo debido a la existencia de un elevado número de aplicaciones que pueden crearse para desempeñar importantes funciones, algunos de los campos en dónde se emplea la visión artificial es la robótica, la seguridad, los procesos de inspección automática, navegación de vehículos, análisis de imágenes médicas, etc. (Rodríguez Villoria, 2013; Andrade Miranda, López Encalada, & Chávez Burbano, 2009)

2.2.1. Etapas de un sistema de visión artificial

Dentro de un sistema de visión artificial están incluidas varias y diferentes técnicas, tales como el procesamiento de imágenes que trata de la captura, transformación, codificación de imágenes; o como el reconocimiento de formas, la cual se basa en la teoría estadística de decisiones y enfoques neuronales de aprendizaje enfocados a la clasificación de patrones. Para llevar a cabo estos procesos, se realizan las etapas que se describen a continuación. (Sobrado Malpartida, 2003)

a. Adquisición de la imagen

Para la adquisición de la imagen se emplea un sistema que, generalmente en varias aplicaciones, está compuesto por dos elementos; el primer dispositivo es la cámara, encargada de transformar las señales luminosas que aparecen en la escena, en señales analógicas capaces de ser transmitidas. Consta de dos partes, el sensor, que captura las propiedades del objeto en forma de señales luminosas y lo transforma en señales analógicas, y la óptica que se encarga de proyectar los elementos adecuados de la escena ajustando una distancia focal adecuada. (Sobrado Malpartida, 2003)

El sistema de iluminación también es muy importante, ya que la escena debe estar adecuadamente iluminada, de acuerdo a los requerimientos, para el correcto desempeño del sistema de visión artificial; pues una apropiada iluminación puede ayudar en el procesamiento y análisis de imágenes mediante

la reducción de ruido, reflexión y el mejoramiento del contraste de la imagen, así también, se evita el desarrollo de algoritmos complejos para solucionar problemas generados por falta de claridad de la escena o aparición de sombras no deseadas, lo cual incrementa el coste computacional, es decir, es posible lograr una alta velocidad de procesamiento utilizando la menor cantidad de algoritmos como sea posible. (Zhang, y otros, 2014; Sánchez Reinoso, 2011)

Existen varias técnicas dentro de los sistemas de iluminación, siendo importante resaltar que la iluminación frontal (se indica en la Figura 4) es la más utilizada, la cual consiste en colocar la fuente de luz y la cámara en la misma posición frontal hacia el objeto; es bastante útil en aplicaciones de inspección de superficies no reflectoras, en algunos casos existen complicaciones debido a que no se puede obtener un buen contraste de la imagen por la aparición de brillo. Algunos dispositivos de iluminación comerciales son los anillos y los domos, que emplean LEDs, tubos fluorescentes o lámparas incandescentes, disponibles en espectros blancos, rojo, infrarrojo, ultravioleta, entre otros. (Sierra Álvarez, 2012; Sobrado Malpartida, 2003; Universidad Nacional de Quilmes, 2005; De la Fuente López & Trespademe, 2012)

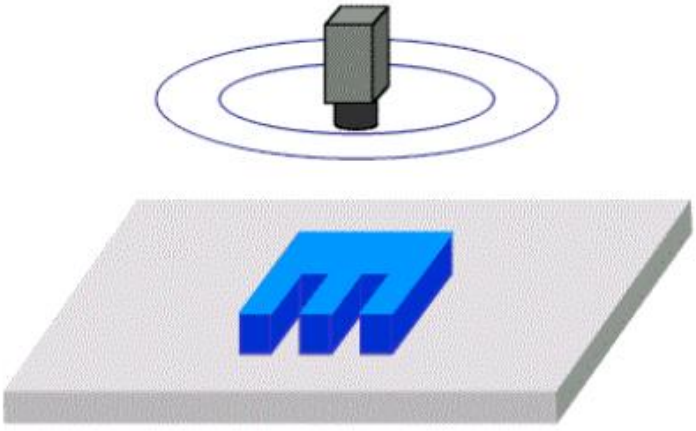


Figura 4: Iluminación frontal.

Fuente: (Sobrado Malpartida, 2003)

b. Procesamiento de imágenes previo

En el procesamiento de la imagen, se considera que una imagen monocromática en dos dimensiones se puede representar de la forma $f(x, y)$ como se indica en la Figura 5. El valor de la amplitud de f en coordenadas espaciales (x, y) es un escalar positivo el cual tiene un significado físico que se determina de la fuente de donde se origina la imagen. La función $f(x, y)$ es caracterizada por la multiplicación de la iluminancia $i(x, y)$ que contiene la cantidad de una fuente de iluminación incidente en la escena observada, y la reflectancia $r(x, y)$ que corresponde a la cantidad de iluminación reflejada por los objetos que se encuentran en la escena. (Sánchez Reinoso, 2011; Loaiza Quintana, Manzano Herrera, & Múnera Salazar, 2012; Valencia López & Abril Cañas, 2007)

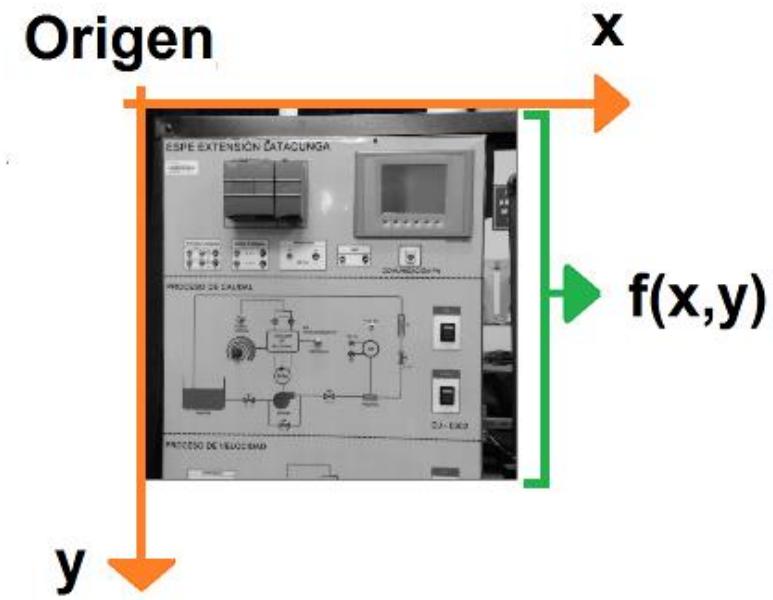


Figura 5: Representación de una imagen en un plano bidimensional.

Se vuelve necesaria la transformación de la imagen original en otra imagen en la cual hayan sido eliminados o reducidos los problemas de ruido o de iluminación, especialmente cuando el entorno se ve afectado por iluminación variable. Esto permite la adecuación de las imágenes digitales adquiridas en base

a los requerimientos del sistema, es decir, la importancia del procesamiento de la imagen radica en el mejoramiento de la misma, de modo que el objeto y las formas contenidas tengan mayores posibilidades de éxito en la identificación. En las siguientes secciones se describe algunas de las técnicas comúnmente utilizadas en procesamiento de imágenes. (Sobrado Malpartida, 2003; Martínez Libreros & Potes Blandón, 2013)

c. Segmentación

La segmentación es aquel proceso mediante el cual una imagen se descompone en regiones o varios conjuntos de píxeles que pueden corresponder a objetos, parte de objetos o contornos. Este proceso es el encargado de evaluar si cada pixel de la imagen genera una imagen binaria, es decir, se representa con 1 (blanco) los pixeles que pertenecen a dicho objeto y 0 (negro) a los que no corresponden. (Sánchez Reinoso, 2011; Calvo Tejedor, 2013)

d. Extracción de características

Las características pueden presentarse en forma de líneas, círculos, elipses y varias otras formas determinadas, así también es posible tomar en cuenta las características locales de la imagen, de tal forma que la información pueda relacionarse con patrones externos; específicamente mediante la extracción de características se pretende reducir la cantidad de información de la imagen a ser analizada. (Sobrado Malpartida, 2003; Muñoz Manso, 2014)

e. Localización y reconocimiento de objetos

Esta etapa corresponde a la localización de un objeto o una forma contenida en la imagen. Muchas veces consiste en detectar figuras geométricas definidas. Es importante que exista un reconocimiento de objetos en base a la obtención de patrones, en otras palabras el reconocimiento y la localización de objetos van a la par debido a que son dependientes entre sí. Para realizar esta etapa es importante considerar las características representativas del objeto en la imagen

como la reflectancia, intensidad, variación del valor de los píxeles, entre otros. (Parra Ramos & Regajo Rodríguez, 2006; Muñoz Manso, 2014)

f. Interpretación de la escena

Cuando se ha obtenido la información requerida en las etapas anteriores se procede a interpretar la escena contenida en la imagen, esta fase se relaciona con el campo de la inteligencia artificial, y se considera la relación existente entre los objetos de interés que han sido reconocidos y localizados, con el conocimiento de los eventos, reglas, restricciones o parámetros del mundo real que intervienen en la aplicación para la toma de una decisión. (Muñoz Manso, 2014; Rodríguez Villoria, 2013)

2.2.2. Algoritmos utilizados en procesamiento de imágenes

a. Conversión de una imagen RGB a escala de grises

Una imagen en escala de grises está representada por medio de una matriz bidimensional de $m \times n$ elementos, mientras que una imagen de color RGB es representada por una matriz tridimensional $m \times n \times p$. Por lo que la conversión de RGB a escala de grises es necesaria para disminuir el costo computacional en el procesamiento de la imagen, debido a que se realizan operaciones sobre una sola matriz. (Cotrina Escandón & Peña Álvarez, 2011; Valencia López & Abril Cañas, 2007)

El ojo humano percibe diferentes intensidades de luz en función del color que observe en el rango del espectro visible; en base a este análisis, el proceso de conversión de una imagen RGB a escala de grises se realiza por medio de una media ponderada de los componentes de color de cada pixel. El valor de intensidad en nivel de grises puede ir desde 0 hasta 255, siendo 0 la representación del negro absoluto y el blanco absoluto está representado por 255. En la Figura 6 se indica un ejemplo de una imagen RGB transformada a escala de grises. (Cárdenas Vera & LLerena Pizarro, 2012; Sobrado Malpartida, 2003)

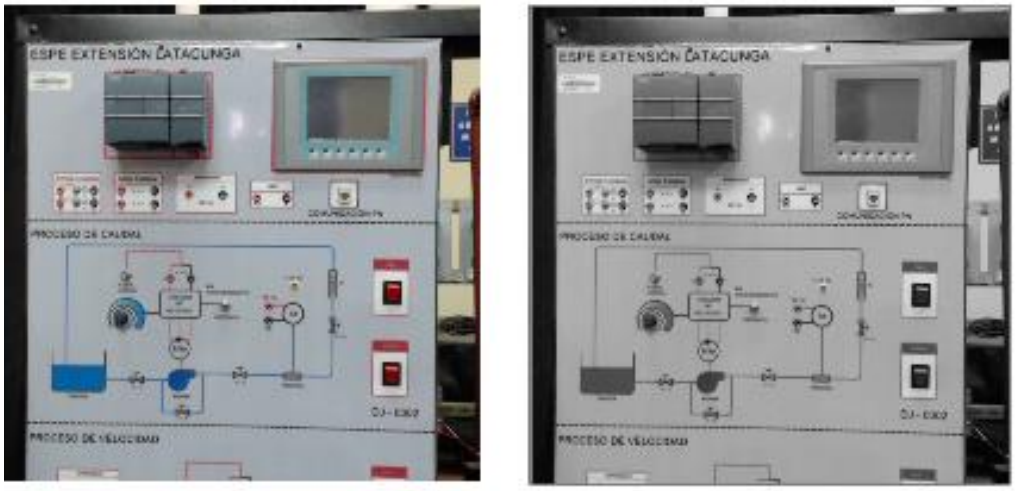


Figura 6: Ejemplo de la conversión de una imagen RGB (izquierda) a una imagen en escala de grises (derecha).

b. Binarización de una imagen

Existen aplicaciones que no necesitan varios niveles de gris, por el hecho que las imágenes pueden estar representadas por pocos niveles; la binarización permite reducir el nivel de gris hasta llegar a tener dos valores, es decir convierte una imagen de escala de grises a una imagen en blanco y negro (0 y 1) dependiendo del umbral establecido, en la Figura 7 se indica un ejemplo.

Figura 7: Ejemplo de la conversión de una imagen en escala de grises a una imagen binarizada.

Fuente: (Muñoz Manso, 2014)

Con este tipo de imágenes se consigue reducir el procesamiento al mínimo, ya que reducir la representación de la imagen hace que se aproveche la memoria y la potencia computacional. Adicionalmente, es posible obtener las propiedades geométricas de los objetos que componen la imagen de forma más fácil y rápida. (Muñoz Manso, 2014)

c. Filtrado de la imagen

Existen variaciones aleatorias de la intensidad o color en una imagen, a este suceso se lo conoce como ruido y se manifiesta como píxeles aislados que toman un tono diferente al de sus vecinos y en la mayoría de casos debe ser atenuado o eliminado mediante un proceso de filtrado; es originado durante la adquisición, donde intervienen factores ambientales o la calidad del dispositivo de captura, así también, durante la transmisión de las imágenes, debido a la interferencia del canal utilizado. Algunos tipos de ruido se mencionan a continuación. (Sada Pezonaga & Sanz Delgado, 2015; Esqueda Elizondo & Palafox Maestre, 2005)

- **Ruido Gaussiano:** El valor de los píxeles de una imagen cambian de acuerdo a una distribución gaussiana; produce un efecto en el que se generan valores aleatorios que no varían considerablemente respecto al valor original del píxel, resultando en pequeñas variaciones en la imagen.
- **Ruido Impulsivo:** Es también conocido como sal y pimienta, debido a que el valor que toma el píxel no tiene ninguna relación con el valor real, es decir los valores son muy altos o muy bajos. El efecto resultante es que en la imagen se visualizan píxeles que toman valores máximos y/o mínimos (puntos que tienden a ser blancos y/o negros).
- **Ruido Multiplicativo:** Se origina cuando una imagen obtenida resulta de la multiplicación o composición de dos señales.

Los filtros son ampliamente utilizados en el procesamiento de imágenes digitales, debido a que a través de los mismos no solo se consigue eliminar el ruido que posee la imagen, sino que además, es posible realzar los bordes de los objetos contenidos en la imagen y realizar un proceso de suavizado de la

imagen, que trata de reducir la cantidad de variaciones de intensidad entre píxeles vecinos. (Sada Pezonaga & Sanz Delgado, 2015)

Es posible clasificar a los filtros en dos grupos importantes, en el dominio de la frecuencia, donde las operaciones se realizan sobre la transformada de Fourier de la imagen, y en el dominio del espacio o también conocido como procesamiento espacial, que trata de operaciones realizadas directamente sobre los píxeles de la imagen, donde se define un entorno de vecindad sobre el píxel a transformar, que se entiende como una ventana de $m \times n$ píxeles centrada sobre el píxel mencionado.

Uno de los filtros más utilizados, principalmente en algoritmos de reconocimiento de bordes, y a la vez empleado en la sección de desarrollo de este trabajo es el filtro Gaussiano, utilizado ampliamente para suavizar la imagen, produciendo un efecto de emborronamiento uniforme, lo cual disminuye el nivel de detalle y el nivel de ruido contenido, especialmente es apto para atenuar el ruido sal y pimienta.

Se produce un efecto negativo en la utilización de este filtro, debido a la pérdida de nitidez, por lo que es necesario realizar una evaluación y/o compensación entre la cantidad de ruido que se va a eliminar y la nitidez que se va a perder. En la Figura 8 se indica la aplicación del Filtro Gaussiano sobre una imagen. (Sada Pezonaga & Sanz Delgado, 2015; Esqueda Elizondo & Palafox Maestre, 2005)

Figura 8: Ejemplo de una imagen que ha sido filtrada con un filtro Gaussiano, a la izquierda la imagen original y a la derecha el resultado del filtro para un $\sigma=2$.

Fuente: (MathWorks, 2016)

El filtro Gaussiano corresponde a la aplicación de una matriz de convolución con una distribución Gaussiana de dos dimensiones dada por,

$$F(x,y) = \frac{1}{2\pi\,\sigma^2} e^{-\frac{x^2+y^2}{2\,\sigma^2}} \quad (1)$$

que, específicamente, es utilizada para calcular cada uno de los coeficientes de la matriz, máscara o kernel de convolución como una aproximación discreta, siendo x y y las coordenadas de la máscara con el punto central considerado como origen; el parámetro σ corresponde a la desviación estándar que determina la intensidad del filtro, una representación se indica en la Figura 9. La operación de la aplicación del filtro Gaussiano, mediante convolución, se puede resumir en que cada píxel de la imagen filtrada resultante es calculado como un promedio ponderado de los píxeles de la imagen original, siendo el valor de ponderación dependiente de la distancia del punto que se está calculando. (Aguilar Vergara, 2008; Paguay Paguay & Urgilés Ortiz, 2012)

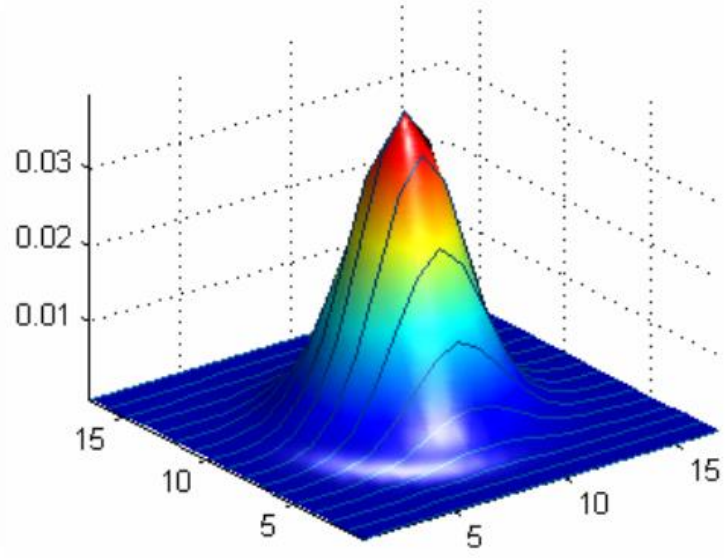


Figura 9: Representación de un filtro Gaussiano bidimensional con un kernel de 17x17 y desviación estándar de valor 2.

Fuente: (Paguay Paguay & Urgilés Ortiz, 2012)

d. Detección de bordes aplicando Canny

La extracción de bordes de una imagen (un ejemplo se observa en la Figura 10) es uno de los algoritmos más útiles en las aplicaciones de visión artificial; a un borde se lo puede definir como una frontera o zonas de píxeles entre dos regiones locales cuyos niveles de gris varían de forma significativa, una respecto a la otra. Para determinar los bordes es necesario el uso de técnicas basadas en el gradiente, dado por,

$$\mathrm{G}[f(x,y)] = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2}$$

que es una medida de cambio en una función, y una imagen es considerada como un vector de muestras de alguna función continua de niveles de intensidad; sin embargo, en la práctica lo que se hace es aplicar máscaras o filtros de imagen, que son matrices que van recorriendo cada pixel de la imagen y realizando una operación determinada. Los operadores que más se utilizan en la detección de contornos son Canny y Sobel. (Muñoz Manso, 2014)

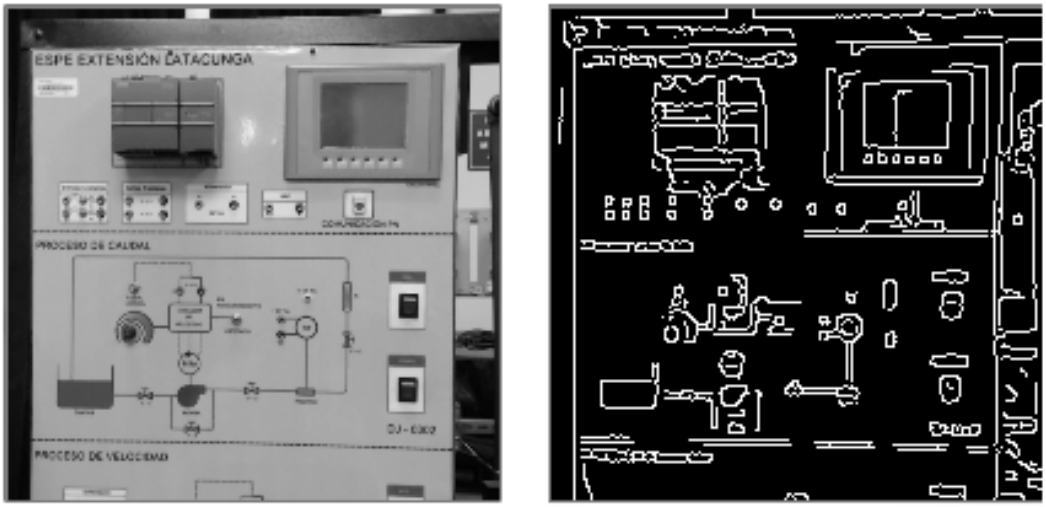


Figura 10: Ejemplo de una imagen donde se han extraído los bordes (derecha) de una imagen en escala de grises (izquierda).

El algoritmo de Canny fue desarrollado por John F. Canny en 1986, se caracteriza por ser una técnica óptima para la detección de contornos, está basado en múltiples etapas que cumplen una serie de criterios que maximizan la probabilidad de detectar bordes verdaderos mientras minimizan la probabilidad de la aparición de los falsos; los principales propósitos del algoritmo de Canny son los siguientes: (Jaramillo, Fernández, & Martínez de Salazar, 2010; Martínez Libreros & Potes Blandón, 2013; OpenCV, 2016)

- **Baja tasa de error:** Corresponde a una buena detección de únicamente los bordes existentes.
- **Buena localización:** La distancia entre los píxeles de los bordes detectados y los bordes reales debe ser mínima.
- **Respuesta mínima:** Lo cual corresponde a solamente una respuesta por borde, es decir, cuando existen dos respuestas para el mismo borde, uno debe considerarse como falso.

Para llevar a cabo la detección de bordes cumpliendo los criterios mencionados, el algoritmo de Canny está estructurado en etapas secuenciales, el primer paso consiste en suavizar la imagen y filtrar cualquier ruido existente

mediante un filtro Gaussiano, el siguiente paso es encontrar la intensidad de los bordes calculando la gradiente de la imagen aplicando el operador de Sobel, que corresponde a una máscara de convolución de 3x3 especificada en la Figura 11, el cual estima la magnitud del gradiente en la dirección x y y mediante la fórmula,

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3}$$

de igual manera, se calcula la dirección del borde mediante la expresión,

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{4}$$

considerando que se realiza un redondeo a uno de los cuatro ángulos posibles (0, 45, 90 o 135 grados).

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figura 11: Máscaras de convolución del operador de Sobel.

Fuente: (Maini & Aggarwal, 2009)

Cuando las direcciones de los bordes han sido encontradas, se aplica el procedimiento llamado supresión del no máximo (non-maximum suppression), que consiste en eliminar el valor de cualquier píxel (fijar el valor del píxel en cero) que no es considerado para ser un borde, lo que genera líneas finas en la respuesta de salida. Finalmente, se utiliza un proceso de histéresis, como un medio de eliminación de líneas no deseadas; donde intervienen dos umbrales, alto $T1$ y bajo $T2$, y se cumple los criterios que, si cualquier valor de gradiente del píxel tiene un valor mayor a $T1$ se toma en cuenta inmediatamente como un píxel que forma parte de un borde, si el valor de gradiente del píxel está entre $T1$ y $T2$ es aceptado únicamente si está conectado a un píxel que supera el valor de $T2$, por último todos los píxeles con valores inferiores a $T2$ son rechazados como parte de un borde. (Maini & Aggarwal, 2009; OpenCV, 2016)

2.3. Redes Neuronales Artificiales (RNA)

2.3.1. Antecedentes

En síntesis las Redes Neuronales Artificiales (RNA) tratan de reproducir el procedimiento de solución de un problema, de una forma similar a como lo realiza el cerebro humano, el cual se basa en aplicar el conocimiento ganado a través de la experiencia de determinadas situaciones, de tal forma que una RNA aprende a tomar decisiones y realizar clasificaciones tomando como ejemplos problemas resueltos o también llamados patrones.

Las RNA se orientan a brindar solución a problemas que no tienen solución computacional precisa, que no están basados en modelos matemáticos, o que requieren algoritmos muy extensos, como es el caso de reconocimiento de imágenes, pues las RNA están en la capacidad de generalizar una respuesta o información que está fundamentada en datos experimentales o bases de datos, los cuales se definen por el experto que conoce el proceso o el problema. (Ponce, 2010)

a. Aplicaciones

Las RNA son aplicadas en diferentes áreas, por citar algunos ejemplos se tiene: en *Robótica* el control dinámico de trayectorias, controladores y sistemas ópticos; en *Manufactura* el control de la producción y del proceso, análisis y diseño de productos, diagnóstico de fallas en el proceso y maquinarias, inspección de calidad mediante sistemas de visión artificial; en *Medicina* el análisis de electroencefalograma y electrocardiograma; en *Seguridad* el reconocimiento de huellas dactilares y voz; entre otros. (Ponce, 2010)

b. Ventajas y desventajas

Las RNA específicamente se enfocan al procesamiento de información para generar una respuesta, en base a esto presentan ciertas ventajas frente a otros sistemas que tienen relación con la inteligencia artificial, entre ellas está: la posibilidad de sintetizar algoritmos a través de un proceso de aprendizaje;

únicamente es necesario estar relacionado con los datos de trabajo mas no conocer a fondo los detalles matemáticos de un proceso; proveen solución a problemas no lineales; una de las más importantes ventajas trata de que las RNA son robustas debido a que continúan trabajando a pesar de que algunos elementos de procesamiento (neuronas) fallen a diferencia de algunos algoritmos de programación tradicional.

Así mismo es importante mencionar algunas consideraciones que, de acuerdo a la aplicación, podrían resultar como desventajas, por ejemplo: es necesario realizar varias pruebas para determinar la arquitectura adecuada de la RNA, es decir, el número de capas y neuronas en cada capa; debe entrenarse las RNA para cada problema, por lo que el entrenamiento puede ser largo y consumir tiempo de procesamiento de la CPU, así mismo, es necesaria una gran cantidad de datos patrones; para añadir nuevos conocimientos a la RNA es necesario modificar el algoritmo de entrenamiento considerando los anteriores conocimientos para lograr un efecto unificado, por lo que se vuelve necesario utilizar un esquema de aprendizaje; la utilización de RNA puede resultar en un aspecto complejo para un observador externo que requiera realizar modificaciones. (Ponce, 2010)

c. Modelo biológico

Una neurona es una célula biológica que procesa información, puede ser dividida en tres partes principales: dendrita, soma y axón (Figura 12). Las dendritas colectan señales de otras neuronas y las transmiten al soma. El soma es la unidad central de procesamiento, en él se genera una señal de salida si el total de las señales de entrada exceden un determinado umbral; ésta señal de salida se trasmite al axón que entrega la misma señal a otras neuronas.

La unión entre dos neuronas se denomina sinapsis; cada neurona pre sináptica (que envía la señal) comúnmente se conecta con más de 10000 neuronas post sinápticas (receptoras de señal). Las señales de las neuronas consisten en pulsos eléctricos cortos que tienen una amplitud de 100 mV y

típicamente una duración de 1 a 2 ms; de este modo las neuronas son elementos de todo o nada, es decir, se activan o permanecen inactivas; por lo que la información transmitida en la señal se identifica en el número y tiempo de pulsos, mas no en la forma de cada pulso, porque éstos son parecidos. (Brauer Luna, 2013)

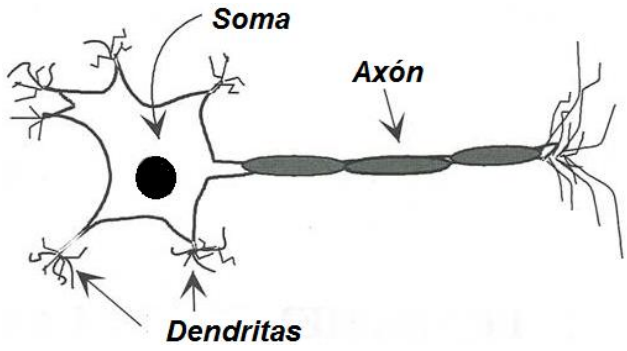


Figura 12: Partes principales de una neurona biológica.

d. Modelo artificial

En términos computacionales una RNA está compuesta por una serie de neuronas artificiales o unidades de procesamiento interconectadas mediante enlaces, basadas en propiedades de redes neuronales biológicas, y usan un modelo matemático o computacional para procesar información; su estructura se representa en la Figura 13. (Escolando Ruiz, Cazorla, Alfonzo, Colomina, & Lozano, 2003)

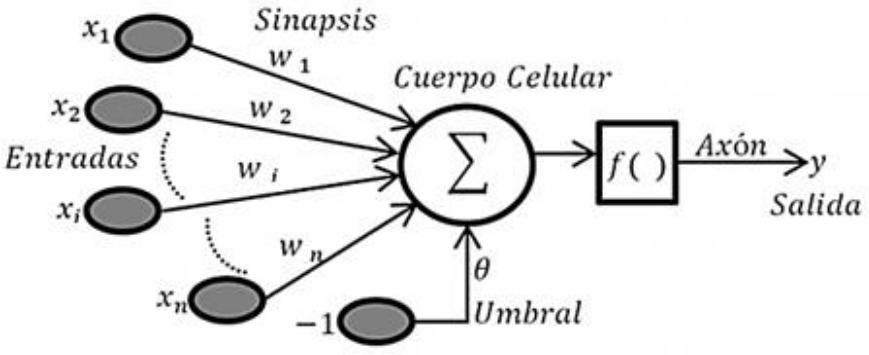


Figura 13: Modelo de una neurona artificial.

Fuente: (Martín & Sanz, 2006)

Las neuronas artificiales simples fueron introducidas por McCulloch y Pitts en 1943, donde se establece que una RNA se compone por los siguientes elementos: (Ponce, 2010)

- Un conjunto de unidades de procesamiento o neuronas.
- Un estado de activación o salida para cada neurona.
- Conexiones entre neuronas, definidas por un peso que determina el efecto de una señal de entrada en la neurona.
- Una regla de propagación, que determina la entrada efectiva o neta de una neurona a partir de las entradas.
- Una función de activación que define un nivel de activación basándose en la entrada efectiva.
- Un método para reunir la información, correspondiente a la regla de aprendizaje.
- Un ambiente en el que el sistema va a operar que proporciona las señales de entrada

Básicamente consisten en que cada neurona recibe la entrada de otras neuronas o fuentes externas y procesa la información para obtener una salida que se propaga a otras neuronas; el procesamiento trata de una sumatoria de la multiplicación del conjunto de datos de entrada por los valores asociados a las neuronas (pesos sinápticos), conocida como entrada neta, y ésta se inserta a una función matemática que determina la activación de la neurona; dentro de la sumatoria es posible añadir un valor de umbral θ que afecte la activación de salida, estableciendose de la forma:

$$y = f(\sum_i x_i \, w_i + \theta) \tag{5}$$

donde los x_i son los valores de las entradas a las neurona; los w_i son los pesos sinápticos, que determinan la influencia de cada entrada en la activación de la neurona, si el w_i es positivo la conexión es exitatoria mientras que si es negativo la conexión es inhibitoria; y f es la función de activación. (Escolando Ruiz, Cazorla, Alfonzo, Colomina, & Lozano, 2003)

2.3.2. Fundamentos de RNA

a. Estructuras y modelos de RNA

El número de modelos de RNA es alto, a partir del primero se han desarrollado diferentes modelos estableciéndose diferencias entre los mismos, tales como: funciones matemáticas de salida, valores aceptables de entrada, topologías, algoritmos de aprendizaje, entre otras. Por citar algunas: Perceptón, Perceptón multicapa, Redes Hopfield, Redes de neuronas de base radial, entre otras. (Brauer Luna, 2013; Alonso & Jara, 2013)

En este trabajo se utilizará la estructura perceptón multicapa (Figura 14), el cual contiene una capa de entrada, una o más capas ocultas y una capa de salida; todas las neuronas de una capa están conectadas a las neuronas de la capa antecedente, las señales fluyen en la misma dirección (hacia adelante); al realizar la operación entre las entradas y los pesos sinápticos se obtiene un escalar, que desde el punto de vista matemático se puede utilizar para la clasificación de problemas multiclase, debido a esto se dice que puede tomar como entradas valores continuos o discretos; generalmente tiene como función de activación la función sigmoide. La fase de ejecución de un perceptón multicapa es una operación computacional muy rápida que puede realizarse en tiempo real. (Alonso & Jara, 2013; Llanos, 2012)

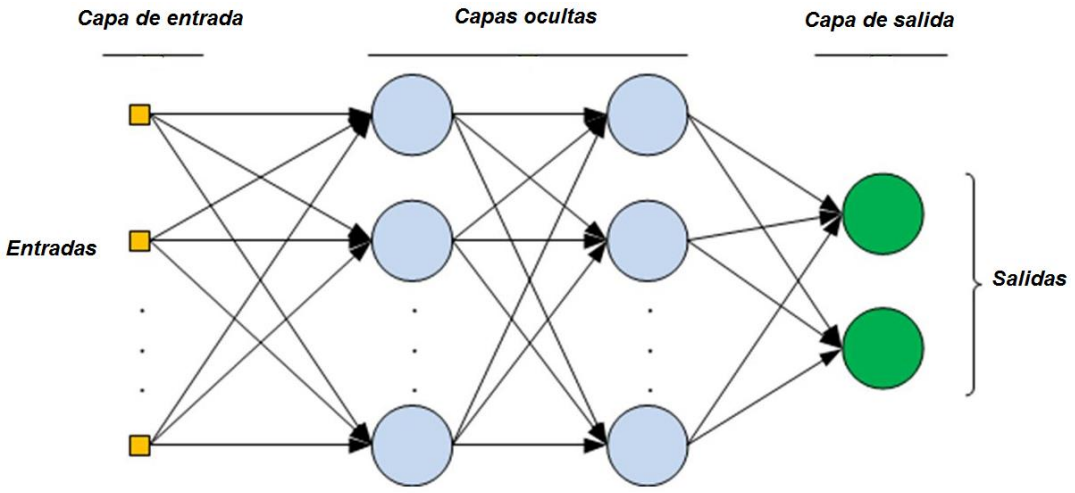


Figura 14: Estructura de un perceptón multicapa.

b. Funciones de activación

La regla que logra establecer el efecto de la entrada neta o efectiva en la activación de la neurona se denomina función de activación; el propósito principal de la misma consiste en evitar la saturación de la neurona, es decir, limitar el valor de salida de la neurona, pues de no hacerlo se irían produciendo valores de entradas a las neuronas siguientes cada vez mayores, hasta alcanzar valores no manejables. (Ponce, 2010) (Escolando Ruiz, Cazorla, Alfonzo, Colomina, & Lozano, 2003)

Entre las funciones de activación que se emplean en las RNA están la función escalón, función lineal, la función Gaussiana; las más utilizadas corresponde a la función tangente hiperbólica y la función sigmoidal, que se empleará en este trabajo (Figura 15). El valor de salida dado por la función sigmoidal es cercano a uno de los valores asintóticos (0 y 1), es decir, en la mayoría de casos el valor de salida está comprendido en la zona alta o baja del sigmoide; es derivable en todo su intervalo, lo cual resulta importante para los algoritmos de aprendizaje de descenso por gradiente (como se verá en la sección correspondiente al algoritmo de Backpropagation). (Ponce, 2010; Escolando Ruiz, Cazorla, Alfonzo, Colomina, & Lozano, 2003)

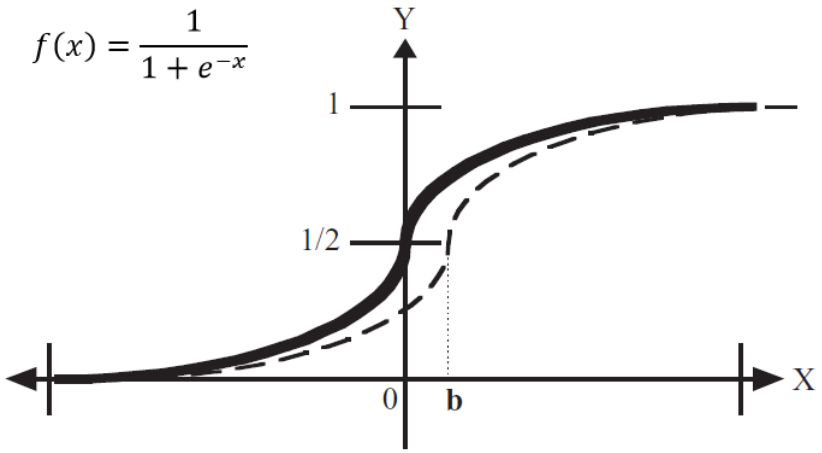


Figura 15: Función de activación sigmoide.

Fuente: (Ponce, 2010)

2.3.3. Entrenamiento de las RNA

El entrenamiento es un proceso de configuración de una RNA para que las entradas produzcan las salidas deseadas a través del fortalecimiento de las conexiones; esto se puede realizar a partir de un proceso que vaya cambiando los pesos de la RNA hasta encontrar los adecuados para que la función de transferencia de la red se ajuste al objetivo; de este modo se dice que el conocimiento de la RNA se encuentra en los pesos de las conexiones entre las distintas capas que forma la RNA. (Ponce, 2010; Llanos, 2012)

a. Tipos de entrenamiento

El entrenamiento o aprendizaje de las RNA puede dividirse en supervisado o asociativo y no supervisado o auto-organizado. El supervisado trata de la introducción de entradas que corresponden a determinadas salidas, que puede ser a través de un agente externo o por el mismo sistema, en cambio en el entrenamiento no supervisado se orienta a encontrar características estadísticas entre agrupamientos de patrones en las entradas. (Ponce, 2010)

El aprendizaje de tipo supervisado, que se utilizará en este trabajo, consiste en inferir una función dado un conjunto de datos de entrenamiento, el cual, consiste en pares de datos entrada-salida; no requiere que el usuario sea un experto debido a que no son necesarios conocimientos de cómo definir una función de recompensa, y además resulta ser intuitivo. (Brauer Luna, 2013)

Las funciones de recompensa o más conocidas como reglas para el ajuste de pesos se utilizan durante el entrenamiento de las RNA, una de ellas y la más usada es conocida como regla delta, la cual implica el ajuste de los pesos a través de la diferencia entre la activación actual y la deseada, está dada por:

$$\Delta w_{jk} = \gamma\, y_j\, (d_k - y_k) \tag{6}$$

donde y_j es la salida de una neurona j que envía una señal a una neurona k, y esta a su vez produce una salida y_k, γ es una constante de proporcionalidad que representa una tasa de aprendizaje y d_k es la activación deseada. (Ponce, 2010)

b. Algoritmo de Backpropagation

Existen varios algoritmos para ajustar pesos y obtener el desempeño deseado de la RNA, el más utilizado es el denominado Backpropagation (retro propagación del error) que es una generalización de la regla delta para el caso multicapa y está dado por el método del gradiente descendiente; se basa en propagar el error de salida de cada capa hacia la capa anterior, de forma que se ajusten los pesos de cada capa de manera independiente; esto se realiza mediante el uso de un conjunto ordenado de entradas y salidas deseadas, y la comparación entre salida deseada y la salida real de la RNA. (Escribano Sánchez & Escardino Malva, 2003; Llanos, 2012)

Este algoritmo de entrenamiento corresponde a un proceso iterativo; en cada iteración los pesos sinápticos son modificados; básicamente está fundamentado en los siguientes pasos: (Ponce, 2010)

1. Se inicia forzando las entradas de la RNA con un conjunto de datos o ejemplos conocidos; así también se inicializan pesos sinápticos aleatorios.
2. Se determina los valores de las señales de salida de cada neurona en cada capa que está produciendo la RNA.
3. Se realiza una comparación entre los valores de salida de cada neurona de la capa de salida con los valores objetivos (Target); ésta diferencia corresponde al valor del error de la señal.
4. Es imposible conocer los valores de error de cada neurona de la o las capas intermedias, se debe propagar cada error obtenido (en el paso 3) de regreso a todas las neuronas anteriores correspondientes, es decir, a aquellas neuronas cuya salida ha sido una entrada de la última neurona.
5. Se modifican los pesos sinápticos mediante una compensación que utiliza los valores obtenidos en esa iteración de entrenamiento.

De esta manera se dice que luego de varias iteraciones y a medida que el error desciende, hasta tender a cero, la RNA aprende a comportarse según el ejemplo que se le ha dado; una características de éste algoritmo es que el error

se va disminuyendo de forma exponencial a medida que atraviesa capas hasta el principio de la red, lo cual resulta ser un problema, porque en una red muy profunda (con muchas capas ocultas), se entrenan únicamente las últimas capas. (Brauer Luna, 2013; Llanos, 2012; Ponce, 2010)

Con respecto a la parte matemática de éste algoritmo se tiene que: para calcular el gradiente del error de cada neurona de cada capa se necesita conocer el estado interno de la neurona, al que se le aplica la derivada de la función de activación, y el error de salida de la neurona. (Llanos, 2012) La derivada de la función sigmoide está dada de la siguiente forma:

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = y(1-y) \tag{7}$$

Se demuestra que cada neurona de la capa de salida k es responsable de una cantidad de error δ_k obtenida de la diferencia entre la salida deseada para esa neurona d_k y la obtenida y_k, multiplicada por la derivada de la función de activación $f'(net_k)$, estableciéndose de la siguiente forma:

$$\delta_k = (d_k - y_k) f'(net_k) \tag{8}$$

De esta forma la regla de actualización de los pesos w_{jk} que relacionan la capa oculta j con la capa de salida k está dada por:

$$w_{jk} = w_{jk} + \eta\ \delta_k\ y_j \tag{9}$$

Además, para conocer el error de salida en las capas ocultas, el error se propaga hacia atrás desde la capa de salida; se considera que una neurona oculta de la capa j es responsable de una parte del error δ_k de cada una de las neuronas a las que está conectado, entonces el error cometido en la neurona de la capa j, δ_j, será la suma de errores de las neuronas de capa siguiente ponderados por los pesos correspondientes. (Escolando Ruiz, Cazorla, Alfonzo, Colomina, & Lozano, 2003)

$$\delta_j = f'(net_j) \sum_k w_{jk}\ \delta_k \tag{10}$$

2.4. Software libre para procesamiento de imágenes

2.4.1. OpenCV

OpenCV (Open Source Computer Vision) es una librería de código abierto desarrollada inicialmente por Intel; está escrita en lenguaje C y C++ y puede ejecutarse en los sistemas operativos Linux, Windows y Mac OS X. Se ha utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos.

OpenCV está diseñado para la eficiencia computacional y un enfoque en aplicaciones de tiempo real; uno de sus principales objetivos es proveer una infraestructura de visión artificial fácil de usar, para la realización de aplicaciones sofisticadas de visión de una manera mucho más rápida. La librería de OpenCV contiene más de 500 funciones que abarcan muchas áreas en la visión, que incluyen inspección de productos en industrias de manufactura, imágenes médicas, interfaz de usuario, seguridad, calibración de cámaras, visión estéreo y robótica.

La licencia de código abierto que posee OpenCV permite desarrollar aplicaciones o productos comerciales usando todo o una parte de OpenCV, de tal forma que los desarrolladores no están en la obligación de dar apertura al código de su producto o aportar con mejoras del software al dominio público; sin embargo, existe una extensa lista de comunidades de usuarios en la web donde se abren foros de temas relacionados al software, se incluyen grandes compañías como Microsoft, Intel o Google, así como centros de investigación como Stanford, MIT, entre otros. (Bradski & Kaehler, 2008; Pacheco Sánchez, 2011)

Figura 16: Logotipo del software OpenCV.

Fuente: (Wikipedia, 2016)

2.4.2. Estructura y módulos de OpenCV

OpenCV se estructura en los componentes principales que se indican en la Figura 17; en CV están contenidos los algoritmos básicos de procesamiento de imágenes y de visión artificial de alto nivel, MLL corresponde a la biblioteca de aprendizaje de máquina, HighGUI contiene rutinas y funciones de carga y almacenamiento de imágenes y videos, y en CxCore están las estructuras de los datos básicos. (Bradski & Kaehler, 2008)

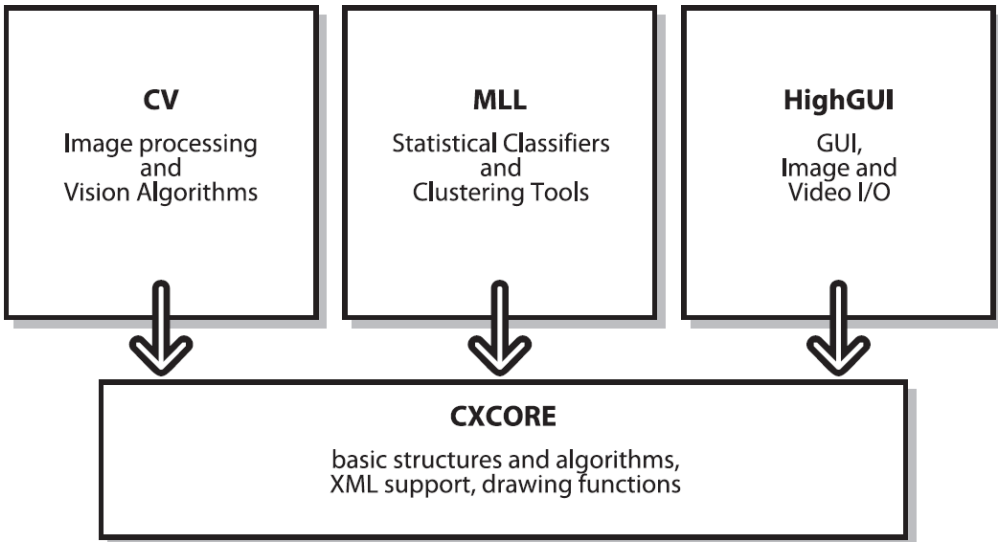


Figura 17: Esquema de la estructura básica de OpenCV.

Fuente: (Bradski & Kaehler, 2008)

Además, en OpenCV se incluyen cientos de algoritmos de visión artificial basados esencialmente en Interfaces de Programación de Aplicaciones (API) de alto nivel y funciones muy eficientes en C++, desde las versiones OpenCV 2.x, que están agrupadas en una estructura modular, por lo que existen paquetes que incluyen varias librerías compartidas o estáticas; algunos de los módulos disponibles más destacados se describen a continuación: (OpenCV, 2014)

- **core:** Corresponde a un módulo compacto que define las estructuras básicas, como por ejemplo el arreglo multidimensional Mat y otras funciones usadas por otros módulos.
- **imgproc:** Es un módulo de procesamiento de imágenes donde se incluyen el filtrado de imágenes lineal y no lineal, las transformaciones geométricas de la imagen, histogramas, entre otras.
- **video:** Trata de un módulo de análisis donde están comprendidos la estimación de movimiento, la sustracción de fondo y los algoritmos de seguimiento de objetos.
- **calib3d:** En este módulos son incluidos algoritmos básicos de visión múltiple geométrica, calibración de cámaras simples o estéreo, estimación de posición de objetos, elementos de reconstrucción en 3D, entre otros.
- **features2d:** Corresponden a detectores de características sobresalientes, descriptores y comparadores.
- **objdetect:** Está comprendida la detección de objetos e instancias de clases predefinidas como por ejemplo personas, rostros, ojos, automóviles, etc.
- **highgui:** Es un módulo que comprende interfaces de fácil manejo para la captura de video e imágenes, además de capacidades simples de Interfaces de Usuario (UI).
- **gpu:** Corresponde a algoritmos aceleradores a través de una Unidad de Procesamiento Gráfico (GPU) de diferentes módulos de OpenCV.

2.5. Sistemas embebidos para procesamiento de imágenes

Los sistemas embebidos, empotrados o integrados pueden ser definidos como sistemas computacionales de procesamiento que están diseñados para realizar una o varias funciones de un objetivo específico y operan en condiciones definidas, en su mayor parte, integran tecnologías basadas en uno o varios microprocesadores, Procesadores Digitales de Señales (DSPs), Arreglos de Compuertas Programables por Campo (FPGAs) y/o demás circuitos integrados; pueden estar compuestos por elementos adicionales como partes eléctricas, mecánicas o electromecánicas, además, el software que está incluido difiere del software de un ordenador convencional, debido a que es necesario que exista más control del tiempo de respuesta y manejo de recursos de memoria, pues los sistemas tiende a operar en tiempo real.

Otras características importantes tratan del bajo consumo de energía, la interacción con interfaces de hardware, alta escalabilidad, tamaño compacto, poseen plataformas de desarrollo flexible, además, éstos pueden ser sistemas independientes o formar parte de un sistema mayor, así también, deben estar en la capacidad de reaccionar ante cualquier eventualidad. Muchos dispositivos embebidos son producidos en miles o millones de unidades, por lo que esto influye en el costo por unidad, considerándose en su mayor parte como relativamente bajo; son ampliamente utilizados en aplicaciones médicas, robótica, telecomunicaciones, electrónica del automóvil, entre otras.

El rendimiento de los procesadores y los grandes avances de la tecnología utilizada en las nuevas generaciones de sistemas embebidos, referente al continuo enfoque de tecnologías de procesamiento paralelo como CPUs multi-núcleo, han permitido que éstos estén en la capacidad de ejecutar algoritmos sofisticados y relacionados al procesamiento de imágenes, de tal forma que se integran en sistemas de detección visual como medios para la adquisición de datos y la toma de decisiones inteligentes con tiempos de respuesta inmediatos; las múltiples aplicaciones de visón artificial embebida también han sido incluidas en sistemas de control y supervisión de industrias, como por ejemplo, de

manufactura, ciencias de la vida, transporte, etc. (Ramos, 2015; López Sandoval, y otros, 2015; Vargas Soria, 2015; National Instruments, 2013)

En los siguientes literales se detalla las características de hardware de las plataformas embebidas utilizadas en este trabajo; adicionalmente se incluye la descripción de la cámara que opera en conjunto con los dispositivos; así mismo, se describe el software necesario para la configuración y operación de las plataformas embebidas.

2.5.1. BeagleBone Black Revisión C (BBB)

La plataforma BeagleBone Black (BBB), que se indica en la Figura 18, proviene del desarrollo de las tarjetas BeagleBoard de Texas Instruments (TI) lanzadas en 2008, las tarjetas BBB fueron lanzadas en 2013 bajo el concepto de Open Source Hardware (Hardware Libre) y están diseñadas para proporcionar una plataforma de desarrollo de bajo costo muy poderosa, flexible, expandible y que permite la creación de proyectos innovadores. BBB utiliza el procesador ARM Cortex A8 que opera a 1 GHz o 2000 millones de instrucciones por segundo, con el conjunto de instrucciones ARMv7, que es una de las CPU más comunes entre los sistemas embebidos actuales de alto desempeño.

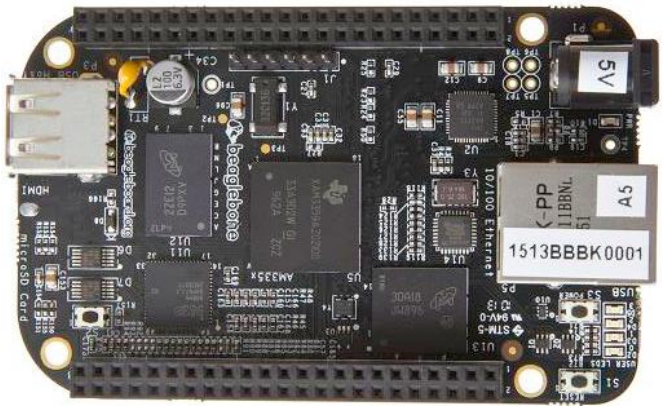


Figura 18: Plataforma embebida BeagleBone Black.

Fuente: (BeagleBoard.org Foundation, 2016)

Es posible programar la BBB haciendo uso del lenguaje Python, C/C++, con comandos en línea de Linux. Sólo tiene un puerto USB, por tanto, si se necesitara conectar un mouse, un teclado o periféricos adicionales, como módulos de comunicación inalámbrica, se debe conectar un HUB USB para su expansión. Consta de dos buses externos tipo DIP de 46 pines cada uno, lo que nos proporciona un total de 92 pines de conexión, con 7 canales ADC a 12 bits, 8 salidas PWM y puertos seriales; cuenta con el soporte de multitud de distribuciones Linux y Android, incluyendo varias versiones de Ubuntu. En la Tabla 1 se resume las principales características generales de la plataforma embebida BBB. (Arenas, 2014)

Tabla 1

Características generales de la tarjeta BeagleBone Black

BeagleBone Black	
Procesador	Texas Instruments Sitara AM3358BZCZ100 ARM-A8 de 32 bits a 1 GHz; cuenta con 2 PRUs (Unidades de tiempo real programables) y motor gráfico SGX530
Memoria SDRAM	512 MB
Memoria de almacenamiento Flash	4 GB, 8 bit Embedded MMC (EMMC)
Puertos USB	Puerto USB Client 2.0 de conector mini USB y puerto USB Host 2.0 de socket tipo A a 500 Ma
Fuente de poder	A través del mini USB o el Jack DC (5 V)
Consumo	300 – 500 mA a 5 V
Ethernet	Conecto RJ45, 10/100 Mbps
Conector SD/MMC	microSD a 3,3 V
Salida de audio y video	micro-HDMI
Especificaciones de entrada y salida	65 posibles entradas y salidas digitales (GPIO); 8 salidas PWM; 4 temporizadores; 7 entradas analógicas de 1,8 V máximo; 4 puertos de transmisión y recepción; 1 puerto serial unidireccional solo de transmisión; 4 puertos SPI

Fuente: (Embedded Linux Wiki, 2016; Pucha Ortíz, 2016; Molloy, 2014)

2.5.2. Raspberry Pi 3 Model B

La tarjeta Raspberry Pi 3 Model B, que se indica en la Figura 19, es la tercera versión de un dispositivo embebido, o también conocido como un mini ordenador, por parte de la Fundación Raspberry Pi que tiene como uno de sus fines hacer económicamente más accesibles los ordenadores en el ámbito de la enseñanza; su bajo costo y sus prestaciones muy aceptables han permitido su ingreso en la informática y en el desarrollo de proyectos que son asistidos por una gran comunidad web. (Machado Da Silva, 2016)

Figura 19: Plataforma embebida Raspberry Pi 3.

Fuente: (Microsoft, 2016)

El modelo consta con un procesador Broadcom BCM2387 Quad-Core ARM Cortex- A53 de 64 bits operando a 1.2 GHz, que corresponde a una velocidad diez veces más rápida a la primera generación de Raspberry Pi, además de una memoria RAM de 1 GB que permite la ejecución de aplicaciones más complejas; principalmente difiere del modelo predecesor (Raspberry Pi 2 Model B) en la capacidad de conectividad inalámbrica, debido a que está integrada una antena y un módulo Broadcom BCM43438 que permite la conexión de redes LAN inalámbricas 802.11 b/g/n y Bluetooth 4.1 (Bluetooth Classic y LE), sin embargo, mantiene compatibilidad en el conector GPIO de 40 pines para integración de tarjetas adicionales. El resumen de las demás características de la tarjeta Raspberry Pi 3 se detalla en la Tabla 2.

No posee una memoria interna de almacenamiento, por lo que el sistema operativo no viene preinstalado, por tanto, el usuario debe precargarlo en una tarjeta SD; para emplearlo como un ordenador es necesario una fuente de 5 V por micro USB, conectar un monitor con entrada digital HDMI, así como teclado y mouse en las entradas USB. Por defecto se puede programar en el lenguaje Python, aunque también existe la posibilidad de hacerlo en C/C++ y Java. (SEEED WIKI, 2016; Adafruit, 2016; Arenas, 2014)

Tabla 2

Características generales de la tarjeta Raspberry Pi 3

Raspberry Pi 3	
Procesador	Broadcom BCM2387 Quad-Core ARM Cortex-A53 de 64 bits a 1,2 GHz
Unidad de Procesamiento Gráfico (GPU)	Dual Core VideoCore IV Multimedia Co-Processor a 400 MHz
Memoria SDRAM	1 GB (compartido con la GPU)
Puertos USB	Puerto USB Client 2.0 de conector micro USB; 4 puertos USB Host 2.0 de socket tipo A
Fuente de poder	A través del micro USB (5,1 V; 2500 mA) o del GPIO header
Consumo	800 mA a 5 V
Ethernet	Conecto RJ45, 10/100 Mbps
Conectividad inalámbrica	WIFI 802.11 b/g/n; Bluetooth 4.1; Bluetooth Low Energy (BLE)
Conector SD/MMC	microSD
Entrada de video	Camera serial interface (CSI) de 15 pines
Salida de video	HDMI; conector RCA (PAL y NTSC); interfaz DSI para panel LCD
Salidas de audio	Conector jack de 3,5 mm; HDMI
Especificaciones de entrada y salida	27 posibles entradas y salidas digitales (GPIO) que operan a 3,3 V; 1 puerto de transmisión y recepción (UART0); 1 bus I2C; 2 SPI

Fuente: (Raspberry Pi Foundation, 2016; WIKIPEDIA, 2016; RS, 2016; Microsoft, 2016)

2.5.3. Cámara web Logitech C920

En este trabajo la cámara Logitech C920 (Figura 20) de conexión USB es utilizada como parte del sistema embebido de visión artificial para la adquisición continua de imágenes, debido a que es compatible con las plataformas embebidas (en el Capítulo III se presentarán el argumento para definir a la cámara como compatible); a continuación se presentan algunas especificaciones importantes por parte del fabricante:

- Calidad de video Full HD 1080p (hasta 1920x1080 píxeles) para Windows en la aplicación Skype.
- Calidad de video HD 720p (1280x720 píxeles) para clientes compatibles.
- Capacidad de grabación de video Full HD (hasta 1920 x 1080 píxeles).
- Compresión de video H.264.
- Capacidad de corrección automática de iluminación escasa.
- Lente Carl Zeiss con enfoque automático de 20 pasos.
- Certificación USB 2.0 de alta velocidad.
- Resolución óptica real de 3 Mega Pixeles (MP) y mejorada por software de 15 MP.
- Frecuencia de cuadro máxima de 30 cuadros por segundo (fps) a 1080p.

Los requisitos de sistema que presenta la cámara web Logitech C920 especifican que es compatible con sistemas operativos Windows Vista, Windows 7, Windows 8, Windows 10 o posterior (32 bits o 64 bits); Mac OS 10.6 o posterior; Chrome OS; Android 5.0 o posterior. De igual manera, para una grabación de video HD 1080p requiere de al menos un procesador Intel Core 2 Duo a 2,4 GHz y 2 GB de RAM o más. (Logitech, 2011; Loja Bravo, 2015; Logitech, 2016)

Figura 20: Cámara web Logitech C920.

Fuente: (Logitech, 2011)

2.5.4. Sistemas operativos para dispositivos embebidos

Existen diferentes distribuciones de sistemas operativos para los dispositivos embebidos, en el caso de la tarjeta BeagleBone Black existen versiones de Android, Debian, Amstrong, Ubuntu, entre otros; mientras que se presentan versiones Raspbian, Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IoT Core y demás para la tarjeta Raspberry Pi 3; debido a que son sistemas operativos abiertos, es posible encontrar en la web diferentes versiones de las distribuciones nombradas para cada dispositivo incluyendo versiones modificadas por algunos usuarios. (BeagleBoard.org Foundation, 2016; Raspberry Pi Foundation, 2016)

En este proyecto se ha utilizado sistemas operativos Linux en los dispositivos embebidos, que además son recomendados en los respectivos sitios web oficiales, estos corresponden a una versión de Debian específica para BeagleBone Black y Raspbian en la tarjeta Raspberry Pi 3; ambos basados en el sistema operativo Debian que es considerado como una distribución de alta calidad, estable y escalable, que cuenta con más de 43000 paquetes, los cuales se definen como software pre compilado y empaquetado en un formato de instalación sencilla en la máquina; por otra parte, Raspbian cuenta con más de

35000 paquetes optimizados para un mejor rendimiento de la tarjeta Raspberry Pi. (Debian, 2016; Debian Proyects, 2016)

Los sistemas operativos Linux son adecuados para utilizarse en sistemas embebidos, debido a que el núcleo (kernel) de Linux es altamente configurable para trabajar sobre una gran variedad de hardware que puede estar limitado (por tamaño de memoria, respuesta en tiempo real, consumo de potencia, etc.), es decir, el sistema de configuración permite elegir únicamente los elementos necesarios para un sistema particular; otras características que son representativas en Linux se presentan a continuación. (García Moya & Barriga Barros, 2012; López Ulloa & Miranda Salvatierra, 2015)

- Es software libre, por lo que no es necesario adquirir una licencia y puede ser modificado por cualquier usuario.
- Tiene capacidad multitarea, es decir, es posible ejecutar varios programas simultáneamente sin que un programa interfiera en la ejecución de otro.
- Permite a más de un usuario utilizar las mismas aplicaciones a la vez trabajando en las mismas o diferentes terminales.
- Existen varias comunidades web enfocadas a la resolución de problemas de programación, actualizaciones, entre otras.

2.5.5. PuTTY

El software Putty, que su interfaz se indica en la Figura 21, es una plataforma de licencia libre disponible en Internet, desarrollado inicialmente para el sistema operativo Windows, recientemente está disponible en varias plataformas UNIX; la función que desempeña es actuar como un cliente SSH, Telnet, Rlogin, TCP raw o serial para establecer una comunicación; en este trabajo se hará uso de la opción cliente SSH del inglés Secure Shell, el cual corresponde a un protocolo de red cifrado para la comunicación segura de datos, que permite acceder a máquinas remotas (consideradas como servidor) a través de una red, de tal modo que es posible manejar completamente un ordenador, incluyéndose la ejecución

de comandos en el servidor y otros servicios de red seguros. (WIKIPEDIA, 2016; Meister, 2007; Eng Hee, 2016)

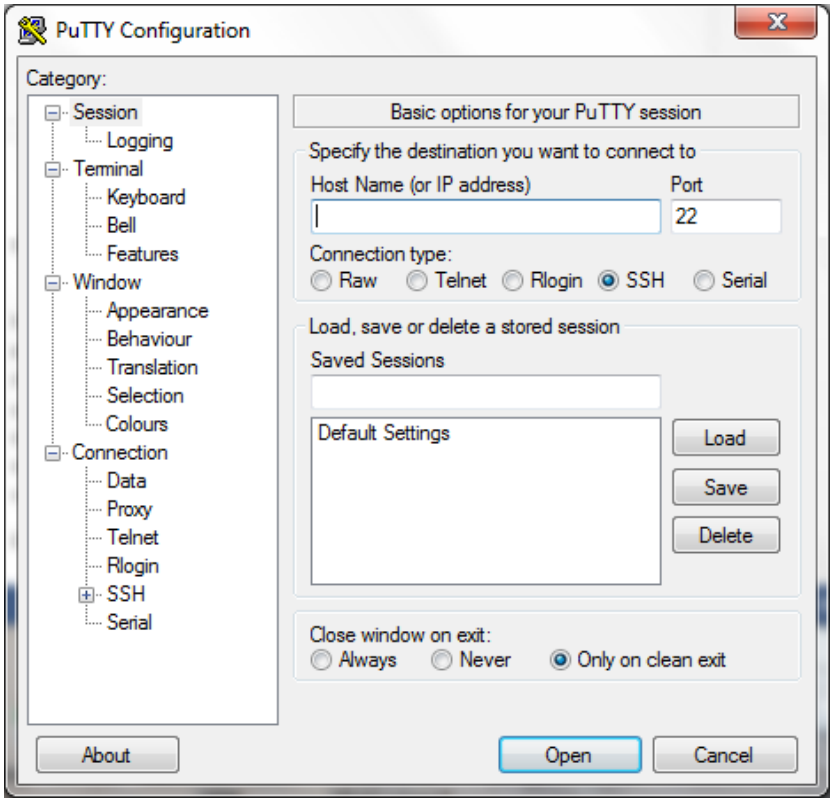


Figura 21: Interfaz del software Putty.

2.6. Sistemas de comunicación

2.6.1. Modelo OSI y TCP/IP

El modelo OSI (del inglés Open System Interconnection) es un lineamiento para las comunicaciones por red, creada por la Organización Internacional de Estándares (ISO), con el fin de promover la accesibilidad universal y la interoperabilidad entre productos de diferentes fabricantes; varios estándares y protocolos cumplen con los lineamientos del modelo OSI. Este modelo está constituido por siete capas, en cada una existe un conjunto de funciones que se deben realizar para que los paquetes de datos se transporten desde el origen al destino dentro de una red, a continuación se describen de forma breve: (Medrano

Chimborazo & Ramos Cárdenas, 2011; Universida Nacional del Centro de la Provincia de Buenos Aires)

- **Aplicación:** Es el nivel más cercano al usuario, se refiere a las funciones que proporcionan soporte a las aplicaciones o tareas del sistema, como por ejemplo el acceso a bases de datos.
- **Presentación:** Las funciones se relacionan con la representación de los datos en un computador, la información es procesada en forma binaria y adaptada a una forma de interpretación más accesible, como la transformación de datos binarios a caracteres, es decir se establece una sintaxis y la semántica de la información transmitida, así como la estructura de los datos.
- **Sesión:** Se encarga de establecer, administrar y sincronizar el diálogo entre aplicaciones remotas para el intercambio de información, es decir, define el proceso de establecimiento y terminación de una sesión.
- **Transporte:** Las funciones de este nivel se encargan de asegurar la integridad de los datos, es decir la calidad de la comunicación, pues en este nivel se realizan las retransmisiones cuando las tramas de información son corrompidas o tienen errores; intervienen protocolos como TCP (Transmission Control Protocol) o UDP (User Datagram Protocol).
- **Red:** Es la encargada de proporcionar conectividad y seleccionar la ruta entre dos sistemas, algunas funciones son asignar direcciones de red únicas, interconectar subredes distintas y/o realizar un control de congestión de las rutas.
- **Enlace:** Su función es adaptar la información codificada en forma binaria en formatos o tramas definidas de acuerdo al protocolo utilizado, es decir, trata del envío ordenado de bits de información ensamblados en tramas, además emplea sencillos algoritmos de detección de errores para validar la integridad de la trama.
- **Física:** Se encarga de las comunicaciones físicas entre dispositivos por donde se transmitirán los datos, involucran las características físicas y/o

eléctricas para el envío y recepción de bits, es decir, se relaciona con componentes y conectores mecánicos, tipos de cables, sistemas de transmisión inalámbrica, niveles de tensión, entre otros.

A más de la existencia del modelo OSI, existe la arquitectura TCP/IP que corresponde a una colección de protocolos abiertos y ampliamente soportados, de los cuales destacan el TCP (Transmission Control Protocol) y el IP (Internet Protocol); es utilizado por la red Internet, por lo que esta estructura se aplica en varias aplicaciones telemáticas, como las redes locales y corporativas, de tal forma que trabaja en cualquier medio de red, hardware y sistema operativo existente. La estructura TCP/IP posee cuatro capas, la correspondencia de cada capa con el modelo OSI se indica en la Figura 22, se describen rápidamente a continuación: (Medrano Chimborazo & Ramos Cárdenas, 2011; Universida Técnica del Norte; Mc Graw Hill Education, 2016)

- **Aplicación:** En este nivel se encuentran protocolos que soportan servicios de conexión remota como Telnet; correo electrónico como SMTP (Simple Mail Transfer Protocol); transferencia de archivos de texto, gráficos, sonidos e imágenes como HTTP (Hipertext Transfer Protocol).
- **Transporte:** Se encarga de manejar la comunicación que se da entre la fuente emisora y el destino receptor a través de la red, proporciona comunicación bidireccional entre los nodos, asegura la integridad de los datos, el orden de su llegada y la posibilidad de retransmisión en caso de errores, también se emplean los protocolos TCP y UDP.
- **Capa de red (Internet):** Tiene como propósito seleccionar la mejor ruta para el envío de paquetes por la red; interviene el protocolo IP (Internet Protocol).
- **Acceso a red:** Se encarga de definir los procedimientos para efectuar la interfaz con el hardware de la red y poder tener acceso al medio de transmisión, es decir, tiene relación con todos los componentes necesarios para lograr un enlace físico; una función importante trata de la asignación de direcciones IP a las direcciones físicas, así como el encapsulamiento de los paquetes IP en tramas.

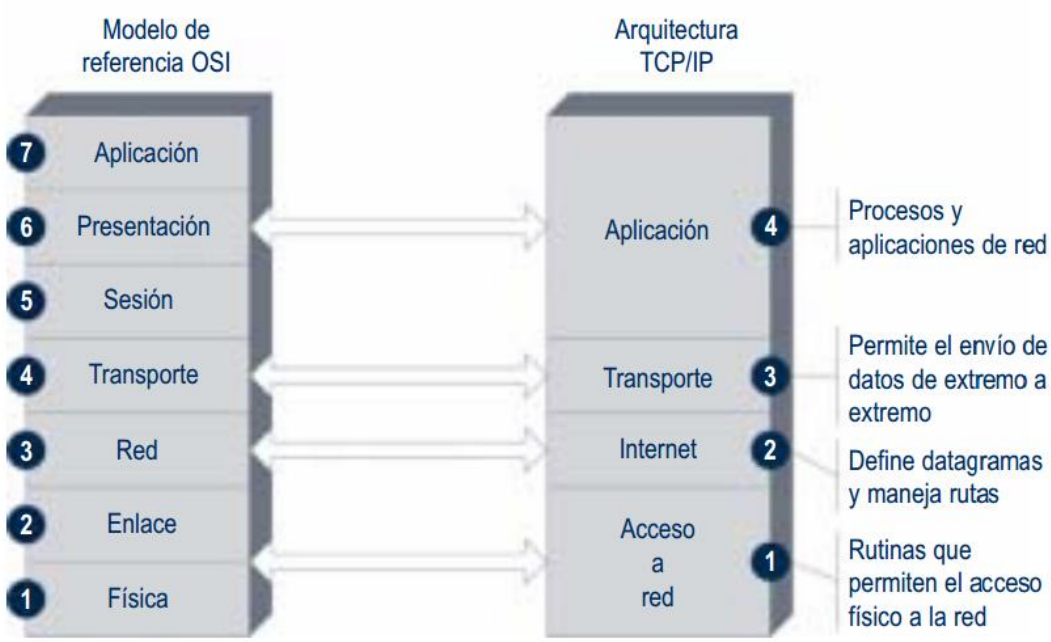


Figura 22: Correspondencia del modelo OSI con TCP/IP.

Fuente: (Mc Graw Hill Education, 2016)

2.6.2. Protocolo de Internet (IP)

Trata de un protocolo no orientado a conexión, lo que quiere decir que cuando dos equipos requieren conectarse entre sí no intercambian información para establecer la sesión; los datos son enviados en bloques conocidos como paquetes o datagramas que contienen información suficiente para propagarse a través de la red sin la necesidad de establecer conexiones permanentes, y tampoco comprueba si se han producido errores de transmisión.

El funcionamiento consiste en que para cada datagrama, que contiene la información en su cabecera y a continuación los datos, se consulta la dirección de origen y la compara con la dirección de destino, si resulta que las direcciones de origen y destino corresponden a equipos de la misma red, el datagrama se envía de forma directa, por el contrario, es necesario la intervención de una puerta de enlace para la facilidad de envío a redes diferentes.

Las direcciones IP corresponden a números de 32 bits, divididos en cuatro grupos de 8 bits traducidos a un equivalente decimal comprendido de 0 a 255, que representan la dirección unívoca de todo dispositivo conectado en la red; estas direcciones proporcionan datos correspondientes al número de red y el número de host; se relaciona con una dirección de máscara que se encarga de indicar que bits de la dirección IP corresponde al indicador de red (bits equivalentes de la máscara en 1) y cuales al host (bits equivalentes de la máscara en 0). (Mc Graw Hill Education, 2016)

2.6.3. Protocolos TCP y UDP

TCP (Transmission Control Protocol) es uno de los principales protocolos de la capa de transporte del modelo TCP/IP, corresponde a un protocolo orientado a conexión, lo cual se refiere a que las aplicaciones solicitan una conexión de inicio confiable y sincronizada con el destino previo a la transferencia de datos de forma fiable, y existe un control del estado de la transmisión entre emisor y receptor.

La transmisión de datos se produce entre un cliente y un servidor donde TCP se caracteriza por garantizar una transmisión segura y fiable de datos por las redes, esto se realiza a través de un sistema de acuse de recibo, es decir TCP retransmite el datagrama hasta que recibe el acuse de recibo, de forma que se entrega los datos en secuencia sin errores, pérdida o duplicación.

Una característica adicional que presenta TCP trata de la optimización del ancho de banda de la red, es decir, existe un control dinámico del flujo de datos entre las conexiones, de tal modo que, cuando existe una sobrecarga de datos en el buffer de entrada del receptor TCP hace que el emisor disminuya la velocidad de transmisión; así también, TCP permite varias conexiones simultáneas. Para que la comunicación y los controles de recibo de datos sean posibles, se agrega un encabezado a los paquetes de datos a razón de sincronizar las transmisiones y garantizar su recepción. (NATIONAL INSTRUMENTS, 2016; Torres Faría, 2009; Universida Técnica del Norte)

En la Figura 23 se indica el formato de un segmento TCP; los campos se describen de forma breve, PUERTO FUENTE y PUERTO DESTINO corresponden a los números de puertos TCP que identifican a los programas; NÚMERO DE SECUENCIA indica la posición de los datos del segmento en el flujo de transmisión; NÚMERO DE ACUSE DE RECIBO indica la cantidad de bytes que la fuente debe recibir; en HLEN está la longitud del encabezado del segmento, correspondiente a múltiplos de 32 bits; RESERVADO es un campo de 6 bits; CODIGO comprende 6 bits que indican aspectos como el inicio y fin de conexión, acuse válido de recibo, etc.; VENTANA informa el número de datos que se aceptarán en cada envío de segmento; SUMA DE VERIFICACIÓN comprueba la integridad de los datos y el encabezado; APUNTADOR DE URGENCIA indica los datos urgentes en el segmento; y OPCIONES, generalmente se usa para definir el tamaño de segmento que se va a recibir. (Universidad Tecnológica de la Mixteca)

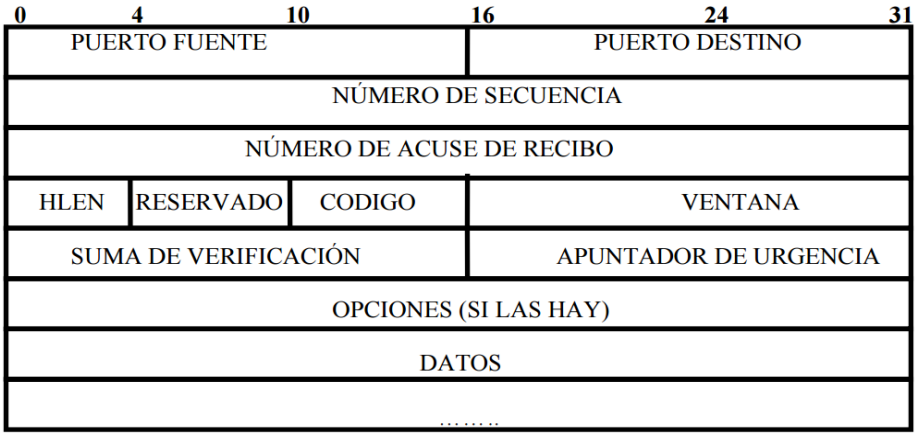


Figura 23: Formato de segmento TCP.

Fuente: (Universidad Tecnológica de la Mixteca)

UDP (User Datagram Protocol), en contraste a TCP es un protocolo no orientado a conexión debido a que no emplea una sincronización entre el origen y el destino; está basado en el intercambio de datagramas o paquetes enteros, a diferencia de TCP que lo hace a través de bytes individuales; es considerado

como no confiable debido a que no emplea un control del flujo de datos, así como tampoco ordena los paquetes.

Una consideración importante trata de que un paquete UDP admite utilizar la dirección de broadcast de IP (255.255.255.255), pues se orienta a no conexión, con ello se permite enviar un mismo paquete a varios destinos simultáneamente; de forma teórica, se puede enviar datagramas de cualquier tamaño, pero, generalmente UDP no es adecuado para enviar grandes datagramas porque no es tan fiable como TCP.

UDP se utiliza en aplicaciones donde fiabilidad de envío de paquetes no es crítica, es decir, una aplicación que transmita datos informativos a un destino de forma suficientemente frecuente, de modo que la pérdida de algunos segmentos no sea un problema; una ventaja es que UDP es un protocolo de rendimiento más alto, en la Figura 24 se muestra la estructura de un datagrama UDP, a diferencia de TCP el encabezado se divide en únicamente cuatro campos de 16 bits. (NATIONAL INSTRUMENTS, 2016; Candelas Herías & Pomares Baeza, 2009; Universidad Tecnológica de la Mixteca)

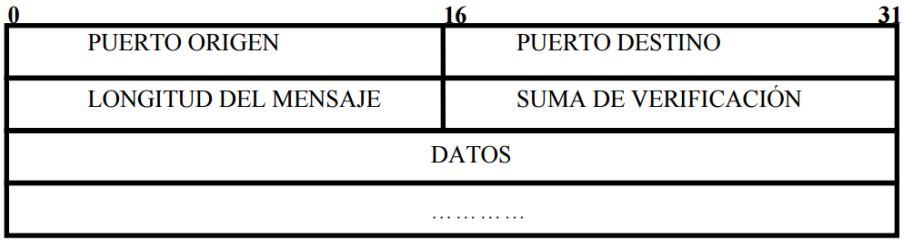


Figura 24: Formato de datagrama UDP.

Fuente: (Universidad Tecnológica de la Mixteca)

2.6.4. Ethernet

Ethernet, corresponde al protocolo IEEE 802.3, es el estándar más popular en las redes de área local (LAN), está basado en un método de transmisión de datos de nombre Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD), de tal forma que antes de que un nodo envíe un dato a través de la

red primero escucha y detecta si otro nodo está transfiriendo información, si no es así, el nodo transmite la información por la red, todos los nodos escuchan y el nodo correspondiente recibe la información; si existe el caso de que dos nodos requieran enviar información al mismo tiempo reconocerán la colisión y esperarán un tiempo para el reenvío.

Las velocidades de envío de los paquetes por Ethernet corresponden a 10 Mbps en el caso de Ethernet estándar, 100 Mbps en Fast Ethernet y 1000 Mbps utilizando el Gigabit Ethernet. La trama de Ethernet es de longitud variable, pero no es inferior a 64 bytes ni superior a 1518 bytes, incluidos el encabezado, los datos y código de detección de errores CRC (del inglés Cyclic Redudancy Check), en la Figura 25 se muestra un diagrama de la forma como se encapsula un dato a partir de una trama Ethernet y como se relaciona con las capas del modelo TCP/IP. (Universidad Tecnológica de la Mixteca)

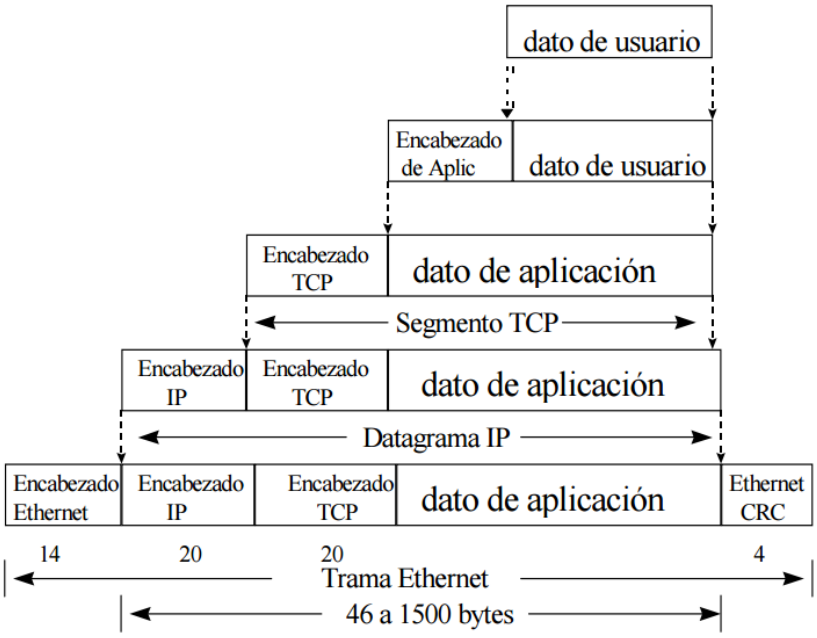


Figura 25: Encapsulado de un dato a partir de una trama Ethernet.

Fuente: (Universidad Tecnológica de la Mixteca)

2.6.5. WIFI

Una red WIFI es una red de comunicaciones de datos, por lo tanto, permite conectar servidores, PC, impresoras, y demás dispositivos, con la característica de no existir la necesidad de cableado; las especificaciones generales de funcionamiento de una red WIFI son las mismas que las de una red con cableado (Ethernet), la diferencia es que el WIFI utiliza el aire como medio de transmisión.

Los dispositivos que incorporan tecnología WIFI correspondiente los estándares IEEE 802.11, permiten la creación de redes de área local conocidas como WLAN, y que son completamente compatibles con los de cualquier otro fabricante que utilice estos estándares. En principio existía el estándar 802.11b, sin embargo, para evitar problemas de compatibilidad de dispositivos y la interoperabilidad de las redes, el término WIFI se extendió a todos los aparatos provistos con tecnología 802.11: 802.11a, 802.11b, 802.11g, 802.11n. (Cruz Dorado, 2014; Colegio Oficial de Ingenieros de Telecomunicación, 2004)

a. Elementos de una red WIFI

La composición de una red WIFI se analiza en tres elementos principales, el punto de acceso (AP), que es el dispositivo que gestiona la información transmitida y la hace llegar a destino, también proporciona la unión entre la red WIFI y la red fija; la antena que se encarga de proporcionar la cobertura necesaria a un usuario para el acceso a la red WIFI; y la terminal WIFI que puede tener forma de dispositivo externo WIFI, se instala en el PC del usuario o bien puede encontrarse ya integrado, adicionalmente se pueden encontrar otros terminales con capacidad de comunicación, como adaptadores WIFI tipo USB y teléfonos móviles, que disponen de accesorios (internos o externos) para conectarse a redes WIFI. (Diputación de Badajoz; Colegio Oficial de Ingenieros de Telecomunicación, 2004)

b. Topologías de una red WIFI

Se analiza dos topologías, por un lado las redes sin infraestructura, las cuales no necesitan un sistema fijo que interconecte algunos elementos de la arquitectura. Son redes que no han tenido un importante éxito comercial. Los ejemplos más habituales que podemos encontrar son las redes AD HOC (comunicación punto a punto), y las redes MESH (utilizan puntos de acceso que trabajan con diferentes canales de frecuencia).

También están la red en modo infraestructura que trabaja utilizando puntos de acceso que realizan las funciones de coordinación; presenta una eficiencia superior a la red AD HOC, ya que este modo gestiona y transporta cada paquete de información en su destino, mejorando la velocidad del conjunto. En este modo de funcionamiento, la tarjeta de red se configura automáticamente para utilizar el mismo canal radio que utiliza el punto de acceso más próximo de la red. (Colegio Oficial de Ingenieros de Telecomunicación, 2004) (Diputación de Badajoz)

c. WIFI en la industria

La tecnología Industrial Wireless LAN (IWLAN), de igual forma, se basa en el estándar WLAN IEEE 802.11 y ofrece aplicaciones especiales aptas para severos requisitos de tiempo real y de redundancia que deben cumplir las aplicaciones del sector industrial. Debido a la tecnología inalámbrica de enlace con sistemas de automatización y terminales industriales se alcanza una mayor flexibilidad, las tareas de mantenimiento se simplifican, los gastos de reparación y los tiempos improductivos se reducen.

Para la protección contra el acceso indebido, los equipos basados en IWLAN ofrecen modernos mecanismos estándar para la identificación del usuario y el cifrado de datos, y al mismo tiempo se integran sin problemas en esquemas de seguridad existentes. Dicha comunicación se basa en normas internacionales, como IEEE 802.11, y utiliza frecuencias de 2,4 GHz y 5 GHz, así como velocidades de transferencia de hasta 450 Mbits/s. Los sistema de comunicación

inalámbrica WIFI se asocian a equipos industriales como PLCs, Touch Pannels, ordenadores y redes Ethernet. (SIEMENS, 2012)

2.7. LabVIEW

National Instruments impulsa el desarrollo de aplicaciones de ingeniería y ciencia, lo que ha promovido el progreso de una gran variedad de industrias, por esta razón existe el software LabVIEW, el mismo que es un entorno de desarrollo integrado, diseñado para la acelerar la productividad de ingenieros y científicos que realizan sistemas de ingeniería. LabVIEW trabaja con un lenguaje de programación gráfico mediante el cual se puede construir exitosamente aplicaciones únicas, además permite realizar la visualización de datos y el diseño de la interfaz de usuario; la incorporación del mismo con hardware y software de diferente proveedor, lo cual garantiza el uso todas las herramientas según la necesidad; y el registro de datos para crear reportes; entre otras. (National Instruments, 2016)

El software LabVIEW facilita el desarrollo de aplicaciones virtuales de los sistemas de instrumentación, pues contiene una colección completa de controles e indicadores, que pueden ser personalizados y programados con el fin de crear interfaces de usuario altamente personalizadas, donde se pueda visualizar los datos y enviar comandos por parte del operador. (National Instruments, 2016; Lajara Vizcaíno & Pelegrí Sebastiá, 2011)

Existe un sitio web que contiene herramientas de código abierto para LabVIEW, el mismo fomenta el uso y desarrollo de código abierto y software libre para la comunidad LabVIEW; inicialmente existen tres herramientas gratuitas disponibles para descargar, utilizar, modificar, redistribuir, contribuir a las mismas y usar libremente en los proyectos de LabVIEW. La herramientas son las siguientes: LabVNC es una herramienta multiplataforma que permite controlar de forma remota un VI (aplicación creada en LabVIEW) a través del internet, LabSQL es un conjunto de herramientas para usar en LabVIEW los componentes de ADO (del inglés ActiveX Data Objects) de Microsoft de tal forma que se pueda conectar

y ejecutar consultas SQL en cualquier base de datos compatible con ODBC (del inglés Open DataBase Connectivity), LabPerl es un API (del inglés Application Programming Interface) que permite que Perl y LabVIEW se comuniquen. (National Instruments, 2013; Travis, 2004)

LabVIEW posee una opción que permite el acceso a una aplicación (VI) de forma remota, por tanto dispone de un servidor propio y dos mecanismos que son los paneles remotos y la publicación web, donde el esquema de comunicación es cliente-servidor, el cliente puede ser un navegador web o el propio LabVIEW y el servidor puede configurarse para permitir o denegar el acceso a ciertos usuarios. Además este software permite crear servicios web, para intercambiar datos con otras aplicaciones usando el estándar abierto XML (Lenguaje de Marcado Extensible), dichas aplicaciones pueden ser desarrolladas en distintos lenguajes de programación y correr en distintas plataformas o sistemas operativos; también existe la posibilidad de publicar el contenido estático público junto a los métodos HTTP (del inglés hyperText Transfer Protocol) VIs del servicio web para que los usuarios accedan mediante solicitudes HTTP a dicho contenido. (National Instruments, 2016; Rodríguez & Hernández, 2013; National Instruments, 2013; Larson, 2014)

2.8. XAMPP

XAMPP es una distribución de Apache que puede descargase de forma gratuita, y permite instalar fácilmente un servidor web local en cualquier sistema operativo por ser multi-plataforma, lo que hace posible realizar pruebas sin necesidad de contratar un hosting (servicio que provee espacio dentro del internet para publicar un sitio web). La herramienta de instalación XAMPP permite de manera automática tener en el ordenador la aplicación servidor (Apahe), el sistema gestor de la bases de datos (MySQL), el lenguaje de programación en secuencia de comandos (PHP), y el lenguaje de programación que admite la extracción de informes de ficheros de texto para la generación de reportes (Perl); además dependiendo de la versión de XAMPP se pueden instalar diferentes

complementos como PHPmyAdmin que es una interfaz gráfica diseñada para administrar la gestión MySQL desde cualquier navegador. (Mikoluk, 2013; Prellezo Gutiérrez, 2002; Alfonzo, Medina, Quintana, & Salazar, 2012)

PHP es un lenguaje de programación en secuencia de comandos (scripts) con el que se puede realizar soluciones para la web puesto que fue creado con este fin, además PHP se integra directamente en las páginas web HTML (del inglés HyperText Markup Language) y es interpretado por el servidor, es decir al realizar una solicitud HTTP al servidor web este ejecuta los comandos de la página PHP y devuelve el resultado en formato HTML, por lo tanto el usuario nunca observará el código PHP. (Hanke)

2.9. Especificación HTML5 para la creación de páginas web

HTML es el lenguaje de marcas de hipertexto (etiquetas que identifican la estructura del texto) más utilizado para describir contenido, o datos, en la World Wide Web, la última versión de este lenguaje es HTML5 e incluye características tanto mejoradas como nuevas, es decir contiene elementos de marcación existentes y nuevos elementos que ayudan a los diseñadores Web a ser más expresivos en la semántica de su codificación. De esta forma no se satura la página con etiquetas de div al contrario se puede incluir las etiquetas article, section, header, footer y otras más.

Sencillamente HTML5 contiene tres características las cuales son estructura, estilo y funcionalidad, por lo tanto, se le considera como el resultado de la combinación de HTML, CSS y JavaScript pero nunca esto ha sido declarado oficialmente, aunque estas tecnologías son altamente dependientes y trabajan como una sola unidad organizada bajo la especificación HTML5; de esta forma HTML se encarga de la estructura, la misma que debe proveer forma, organización y flexibilidad, CSS presenta esa estructura y su contenido en la pantalla y JavaScript permite realizar mejoras en el entorno que se mostrará al usuario logrando de esta manera páginas web dinámicas .

La tecnología CSS es un complemento creado para superar las limitaciones y reducir la complejidad de HTML. Antes los atributos que proveían estilos esenciales para los diferentes elementos se los programaba dentro de las etiquetas HTML, pero a medida que fue evolucionando el lenguaje el desarrollo del código era más complejo; por lo tanto, CSS es la forma de separar la estructura de la presentación. El lenguaje de estilo (CSS) trabaja en conjunto con HTML, ya que, proporciona estilos visuales a los elementos del documento como pueden ser tamaño, fondo, tipo de letra, color, bordes, entre otros. Por esta razón también se lo describe como aquel lenguaje que presenta el código HTML o aplica un estilo.

Javascript (JS) es un complemento usado para varios propósitos, entre ellos el desarrollo web, al ser un lenguaje de programación interpretado depende de un entorno de ejecución que podría ser un navegador web, el cual posee un motor de interpretación que transforma el código JS en código de máquina, por esta razón se alcanzan velocidades de ejecución altas que beneficia a la velocidad de respuesta de las páginas web, debido a que el código JS puede ser procesado por el mismo cliente que pide la información al servidor y no por este último.

Para el desarrollo de un documento HTML, CSS o JavaScript no es indispensable un programa de desarrollo web, ya que, al ser un archivo de texto se puede desarrollar sobre un Bloc de Notas en Windows o cualquier editor de texto, únicamente debe ser grabado con la extensión .html, .css, .js respectivamente. (Goldstein, Lazaris, & Weyl, 2011; Gauchat, 2012)

CAPÍTULO III

3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Este capítulo trata del desarrollo práctico que se ha realizado para cumplir con la propuesta del presente proyecto de investigación; en el literal 3.1 se describe de forma breve el diseño contemplado del sistema de monitoreo de variables físicas basado en visión artificial, así como se introduce las consideraciones generales del sistema y se justifica el porqué de la utilización de técnicas, software y/o hardware de algunas de sus etapas, las mismas que se detallan del literal 3.2 al 3.8.

3.1. Diseño del sistema de monitoreo propuesto

3.1.1. Diseño general de la estructura del sistema

En este proyecto se propone una solución en cuanto a la implementación de un sistema de monitoreo de variables físicas utilizando dispositivos embebidos de bajo costo; la operación del sistema estará basado en el reconocimiento y la digitalización de la medida leída en instrumentos analógicos a partir de técnicas de visión artificial; se ha considerado utilizar los instrumentos analógicos como elementos primarios de medida, pues los antecedentes describen que éstos aún son ampliamente utilizados en las industrias, por su robustez y funcionalidad que facilita a los operadores distinguir cambios de valor de las variables de proceso de forma sencilla y rápida por la deflexión del indicador; por lo que se aprovecha los recursos existentes dentro de la planta.

El sistema debe tener la característica de ser no invasivo, es decir, que no es necesario realizar modificaciones directas a los elementos de la planta de los cuales depende el proceso, tales como tubería o válvulas; así también, está orientado a la adquisición continua de datos en tiempo real, adecuado para la función de monitoreo; se adiciona la función de transmisión de datos a través de una red inalámbrica WIFI a un sistema servidor donde se centraliza, almacena y presenta toda la información adquirida, y además actúa como un servidor de una

página web que proporciona un servicio de monitoreo remoto, específicamente en la misma red y fuera del área de supervisión.

La Figura 26 corresponde al esquema de diseño del sistema propuesto, en éste se indica que los algoritmos de visión artificial se ejecutan en cada plataforma embebida, éstas a la vez transmiten los datos a un sistema servidor realizado en LabVIEW donde se supervisa el estado de las variables del proceso; éste a la vez se complementa con el software XAMPP y otros leguajes de programación (XAMP, HTML 5, CSS, PHP, JAVASCRIPT) para la creación de la base de datos y la página web; todo el sistema opera en una única red inalámbrica, por lo que se considera la adición de niveles de usuarios y contraseñas de seguridad para el acceso a la información y la modificación de configuraciones.

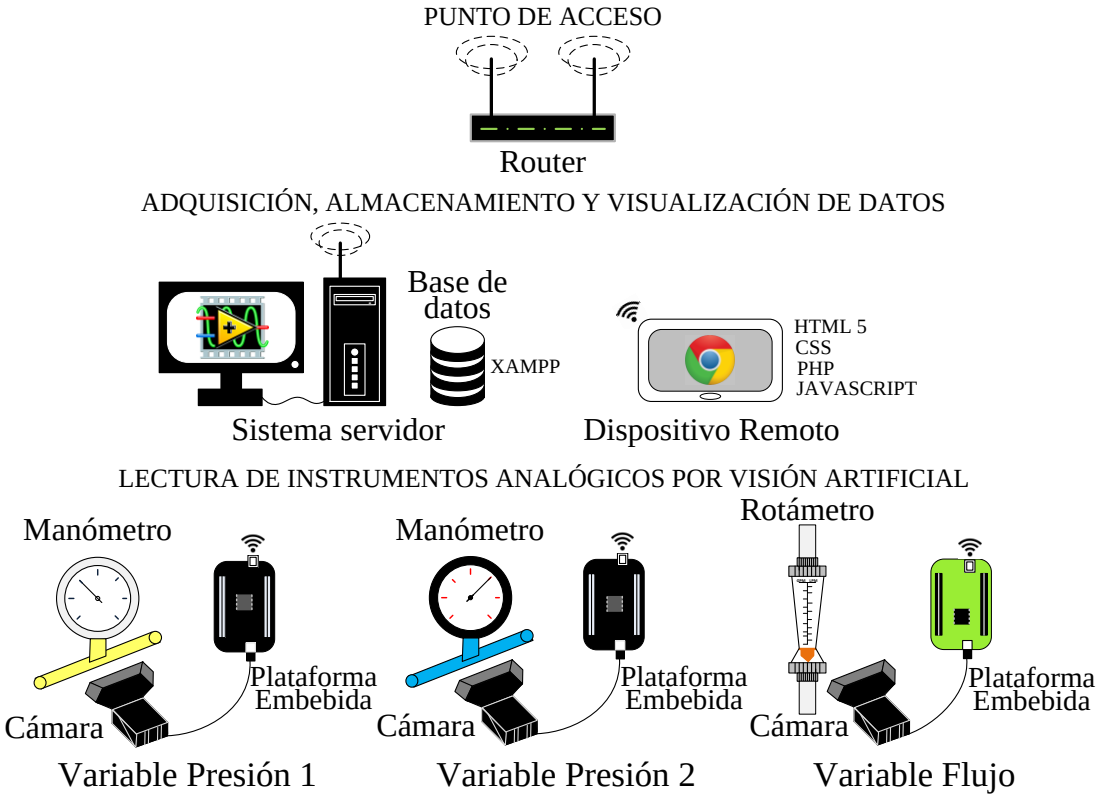


Figura 26: Esquema del sistema de monitoreo basado en visión artificial.

3.1.2. Consideraciones del algoritmo y los sistemas embebidos

El algoritmo encargado de determinar el estado de la variable de un proceso mediante la adquisición, procesamiento y análisis de imágenes en tiempo real, debe diseñarse para ser adaptable y operar mediante el reconocimiento y seguimiento de formas de objetos correspondientes a cualquier tipo de instrumento analógico y sus respectivos indicadores (agujas, flotadores y demás punteros); por tal motivo se considera la utilización de Redes Neuronales Artificiales (RNA) aplicadas como elementos que proporcionan inteligencia y memoria a un sistema de visión artificial; esto debido a que con un proceso de entrenamiento definido, característico de las RNA, es posible que éste sistema aprenda a reconocer y diferenciar diferentes formas de interés o Regiones de Interés (ROI) contenidas en una imagen que proporcionen información relevante para los fines de la aplicación, en este caso la lectura del valor de la variable. En la Figura 27 se indica el esquema del modo de operación del sistema propuesto, donde se especifican las etapas del algoritmo de visión artificial.

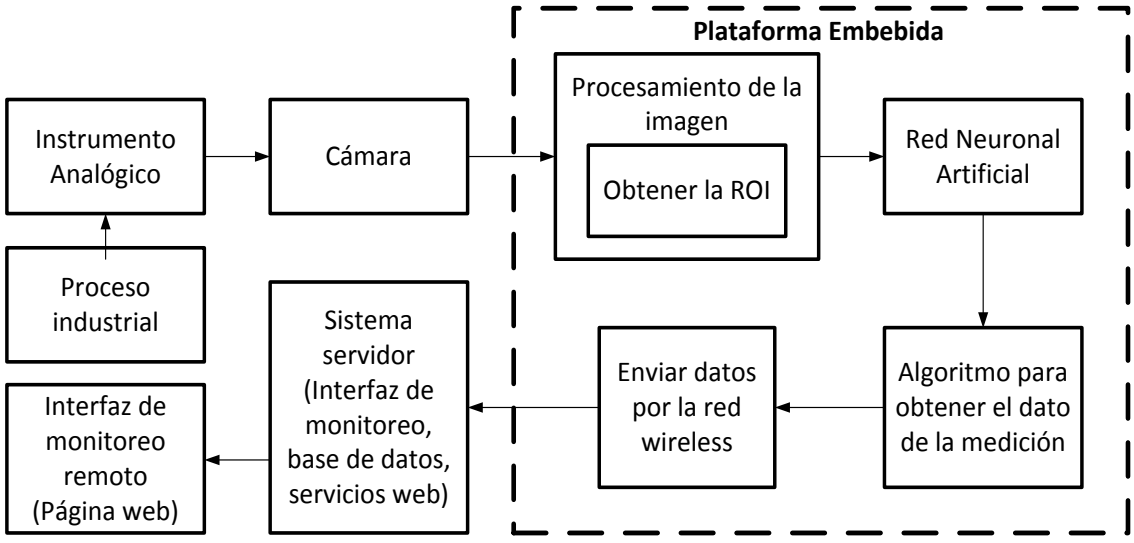


Figura 27: Esquema de operación del sistema de monitoreo de variables físicas basado en visión artificial con RNA.

Debido a que el algoritmo emplea RNA es necesario considerar un espacio de memoria donde se almacenarán el valor de los pesos de conexión de cada neurona empleada (sinapsis), además, el algoritmo debe poseer rutinas de emergencia en el caso de no poder cumplir con su propósito, esto debido a que forma parte de un sistema embebido que opera de forma autónoma; en base a estas consideraciones se ha optado en utilizar las plataformas BeagleBone Black y Raspberry Pi 3, que poseen características de velocidad de procesamiento y almacenamiento necesarias para el desarrollo de rutinas complejas de procesamiento de imágenes y cálculo de extensas operaciones iterativas que involucran las RNA; así como la posibilidad de comunicación con dispositivos externos y la adición de periféricos; es importante aclarar que la tarjeta BeagleBone Black no posee un módulo interno de comunicación WIFI, sin embargo es posible añadir un adaptador WIFI por el puerto USB.

Es necesario el empleo de sistemas operativos basados en Linux en las tarjetas embebidas utilizadas, debido a que se pueden agregar los paquetes de edición, programación, compilación de programas, comunicación y drivers de periféricos que únicamente serán utilizados. Un aspecto importante a considerar es definir la cámara que se utilizará como elemento de adquisición de imágenes, ya que existen algunos modelos específicos que son compatibles y recomendados, principalmente con la tarjeta BeagleBone Black.

Se utilizará la cámara web Logitech C920, cuyos requisitos se indican en la Tabla 3, donde se determina que funciona de forma adecuada en dispositivos con procesador que opere al menos a 1 GHz, que es la frecuencia de operación de la tarjeta BeagleBone Black; la frecuencia de operación de la plataforma Raspberry Pi 3 es superior (1,2 GHz) por lo que también es compatible con la cámara seleccionada, que además permite la adquisición de video a una velocidad máxima de 30 cuadros por segundo (FPS) con una resolución de imagen de 640 x 480 pixeles y un adecuado soporte de drivers sobre el sistema operativo Linux. (Shah, 2014)

Debido a que el sistema propuesto está basado en visión artificial se contempla la utilización de un sistema de iluminación, en la etapa de adquisición de imágenes, en los casos donde se presente la necesidad a causa de determinados factores que compliquen el procesamiento, análisis e interpretación de imágenes, por ejemplo la baja o ausente iluminación en el ambiente.

Tabla 3

Requisitos de la cámara web Logitech C920

Requisitos del sistema		
Requisito básico	CPU mínima de 1 GHz CPU recomendad Core 2 Duo a 2,4 GHz o superior	RAM mínima de 256 MB RAM recomendad de 2 GB
Requisito HD	CPU mínima Core 2 Duo a 2,4 GHz o superior CPU recomendada i7 Quad Core a 2,6 GHz o superior	RAM mínima de 2 GB RAM recomendad de 4 GB

Fuente: (Logitech, 2016)

3.1.3. Consideraciones de la transmisión de datos de forma inalámbrica

Además de obtener el dato de la variable por visión artificial, la plataforma embebida debe transmitir el dato al sistema servidor, y debido a que el dato debe actualizarse de forma continua en tiempo real es necesario utilizar un protocolo de transmisión de datos de gran velocidad, para ello se ha considerado basar la conectividad inalámbrica de un modo similar a cómo lo realizan equipos industriales.

Los protocolos industriales están basados en un modelo de cuatro capas parecido a TCP/IP con la diferencia que el nivel más bajo corresponde a la capa física como el modelo OSI y el nivel superior o capa de aplicación contiene el

protocolo CIP (del inglés Common Industrial Protocol) que es propio de las redes determinísticas utilizadas en la industria, tales como EtherNet/IP, DeviceNet, ControlNet; específicamente éste protocolo abarca el conjunto de servicios y extensas librerías para las aplicaciones de automatización industrial por ejemplo el control, seguridad, soporte de dispositivos de entradas/salidas analógicas/digitales y HMIs, gestión de la red, integración de redes ofimáticas e Internet, entre otras. (Fábregas, 2009; ODVA, 2016)

A partir de lo anterior, se considera que la etapa de comunicación inalámbrica a través de WIFI de este proyecto opere de forma similar al modelo que emplea EtherNet/IP que se muestra en la Figura 28, refiriéndose específicamente al protocolo UDP en la capa de transporte, el cual gestiona información que debe ser transmitida en tiempo real y está relacionada a datos de valores de entradas y salidas que intervienen en el control y monitoreo de un proceso.

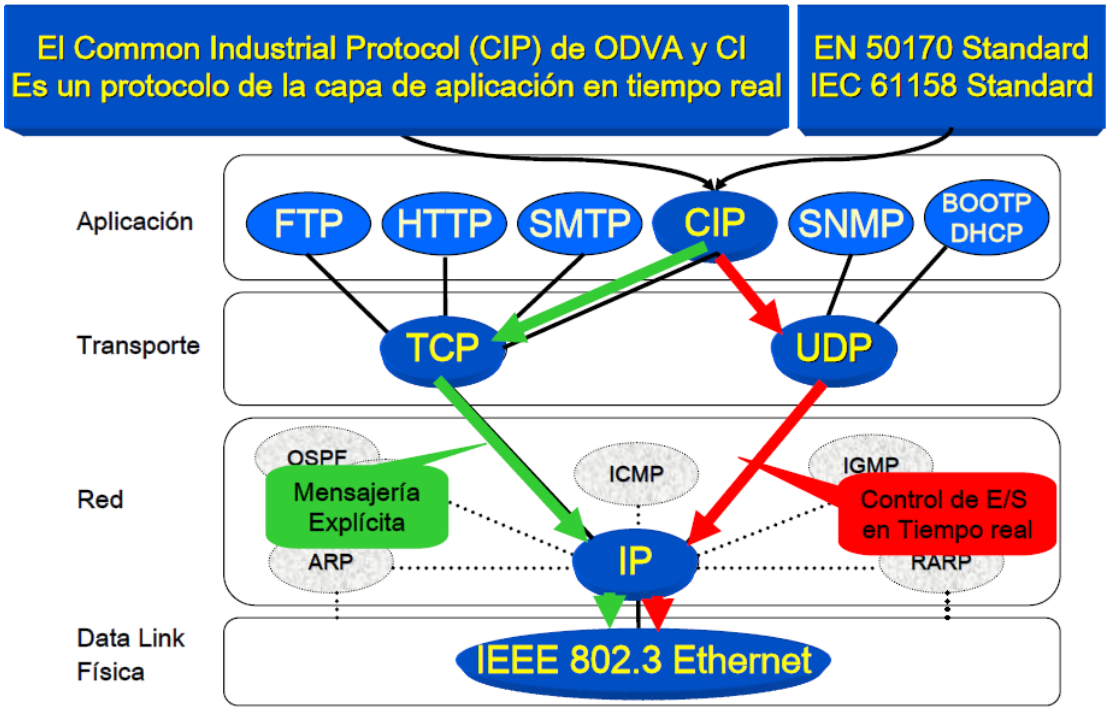


Figura 28: Modelo de comunicación industrial EtherNet/IP.

Fuente: (Fábregas, 2009)

La característica de velocidad de transmisión que presenta UDP al emplear encabezados cortos y enviar paquetes de información completos resulta adecuada para el sistema propuesto, debido a que la tarjeta embebida estará enviando datos constantemente hacia el sistema servidor, por lo que la pérdida de paquetes que puede producirse en ocasiones no alterará a la actualización del dato correspondiente al valor la variable vista en el HMI; así también, es importante considerar que los datos que se manejan en el sistema embebido representan valores numéricos o mensajes cortos, mas no cadenas de caracteres extensas, lo que justifica el empleo de éste protocolo.

Los demás niveles del modelo operan de forma transparente y corresponden de la siguiente manera, la capa física se ve reflejada en la utilización de un punto de acceso o enrutador WIFI que gestione la transferencia de datos por aire, el protocolo IP utilizado en la capa de red, y en la capa de aplicación se tiene todo el conjunto de servicios integrados en LabVIEW en el sistema servidor, así como las librerías de código C++ que permiten la comunicación en el lado de las tarjetas embebidas.

3.1.4. Consideraciones del sistema servidor

Para el sistema servidor se considera que éste debe estar conformado de servicios a modo de un Sistema de Control Supervisorio y de Adquisición de Datos (SCADA) donde se centraliza la información de los procesos, es decir, donde se visualiza el estado de cada una de las variables del proceso en un HMI intuitivo, para el caso de este proyecto que trata únicamente de monitoreo se considera los servicios de alarmas, tendencias de variables, registro de los valores de las variables e históricos de alarmas, así también seguridades que bloquen o habiliten ciertas funciones del HMI.

Existe una consideración importante con respecto al registro de los datos correspondientes a los valores de las variables y el histórico de las alarmas, ya que le debe ser posible al usuario utilizar estos datos en el momento que él crea conveniente, aún si el sistema está en funcionamiento y se siguen registrando

más datos; LabVIEW posee funciones directas para almacenar datos, un ejemplo es la exportación de datos a hojas de Excel, sin embargo, no es posible abrir ésta hoja mientras la aplicación de LabVIEW se ésta ejecutando.

En base a lo anterior se ha contemplado comunicar internamente LabVIEW con una base de datos que pueda gestionarse de forma independiente, por lo que se hará uso del software XAMPP, que actúa como un servidor local y utiliza simultáneamente MYSQL, para la creación de bases de datos, y APACHE, que le permite operar como un servidor web que funciona sin internet; de modo que los datos almacenados puedan ser visualizados y descargados en un formato de Excel, PDF o texto plano desde cualquier ordenador o dispositivo de la red mientras el sistema está en funcionamiento.

El sistema servidor también debe proporcionar información del estado de las variables del proceso de forma remota, con el fin de proveer una opción de supervisión ocasional fuera del cuarto de control a través de dispositivos que se encuentran conectados en la misma red inalámbrica; por lo que se considera el desarrollo de una interfaz gráfica (página web) que pueda ser visualizada desde un navegador web como Google Chrome, ésta deberá desplegar el estado del valor de las variables y las alarmas, así como contener los diagramas de los procesos para una mejor interpretación por parte del usuario; se considera que el acceso a éste servicio debe estar permitido para uno o varios miembros del personal, de tal modo que estará equipado con seguridades de acceso a través del nombre de usuario y contraseña que se agregarán desde la estación centrar (HMI del sistema servidor).

LabVIEW posee la herramienta Web Publishing que permite publicar en la web la interfaz del VI diseñado, sin embargo no se considera ésta opción porque existe la limitación de que solo puede operar las funciones de la interfaz un usuario a la vez; en éste servicio uno o más usuarios accederían al mismo tiempo desde diferentes dispositivos y deben escribir el nombre de usuario y contraseña en casillas de texto, es decir, requerirían manipular una función del HMI. En base a esto se considera la alternativa del desarrollo de una página web haciendo uso

de HTML5, CSS3 y PHP que se comunicará con la aplicación de LabVIEW a través de su Servicio Web para extraer los datos que serán mostrados, además, se conectará con la base de datos en XAMPP donde estarán registrados los nombres de los usuarios y contraseñas para el control del acceso.

3.2. Configuraciones de las plataformas de desarrollo embebidas

3.2.1. Configuraciones iniciales

Las tarjetas utilizadas operan con un sistema operativo basado en Linux, para cada una se ha utilizado el sistema operativo que recomienda cada una de sus Fundaciones que brindan soporte de su manejo; esto corresponde a las versiones disponibles más recientes de la fecha de inicio de la utilización de las tarjetas, es decir, la versión de Raspbian de marzo 2016 para la Raspberry Pi 3 obtenida del enlace https://www.raspberrypi.org/downloads/raspbian/ y la versión Debian 8.4 para BeagleBone Black de mayo 2016 obtenida de https://beagleboard.org/latest-images; las guías de inicio de los sitios web oficiales muestran cómo cargar los sistemas operativos en las tarjetas.

Los sistemas operativos mencionados corresponden a versiones con paquetes preinstalados, como el escritorio y demás programas que tiene un fin educativo; existen otras versiones de sistemas operativos (que vienen únicamente con la terminal de comandos sin ningún paquete) que también se pueden utilizar en este proyecto siguiendo los procedimientos que se irán explicando a continuación; no obstante, la razón de utilizar éstas versiones está en que se realizará, más adelante, una comparación del desempeño de ejecución de ambas tarjetas considerando el software que proveen y recomiendan sus respectivas Fundaciones.

En este trabajo se ha considerado manejar las tarjetas embebidas de forma remota desde un ordenador para realizar las tareas relacionadas a la instalación de software y paquetes, así como la programación, compilación y ejecución del algoritmo propuesto; es así que se utiliza el software PuTTY para establecer una

conexión segura Ethernet de tipo SSH por el puerto 22; la dirección IP es propia de cada tarjeta, es importante mencionar que en el caso de la BeagleBone Black se emula esta conexión a través del puerto USB Client (Mini USB) y la dirección IP está predefinida por el sistema operativo como 192.168.7.2, en cambio en la Raspberry Pi 3 se utilizará el propio puerto Ethernet para éste propósito y se definirá una dirección IP (más adelante en esta sección -192.168.10.20-).

Es importante que al momento de desarrollar y ejecutar el algoritmo, que involucra procesamiento de imágenes, se desplieguen las pantallas donde se visualiza la secuencia de imágenes capturas y procesadas que forman un video, esto con el fin de cumplir únicamente con actividades de ajuste de la posición de la cámara y verificación de funcionamiento de los procedimientos dentro del algoritmo, ya que posteriormente la tarjeta quedará empotrada y sólo enviará datos.

Para esto es necesario utilizar un software denominado Xming X Server (se puede encontrar en http://www.straightrunning.com/XmingNotes/) que actúa como un complemento de PuTTY para que en el ordenador, desde donde se accede a las tarjetas, sea posible la apertura de pantallas gráficas. Una vez instalado se ejecuta Xming V Server, éste aparecerá únicamente en la barra de tareas de Windows con la locación 0.0 como se indica en la Figura 29.

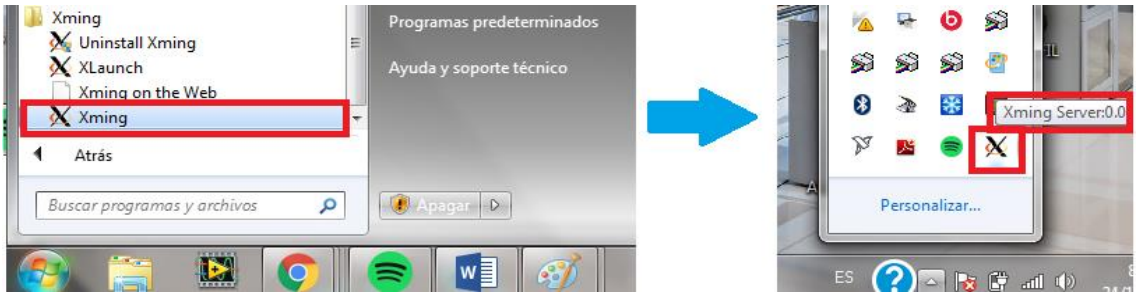


Figura 29: Ejecución del software Xming X Server (complemento de PuTTY).

El modo de configurar PuTTY para establecer una conexión con una tarjeta se indica en la Figura 30, es importante resaltar las configuraciones que se relacionan con Xming X Server para la apertura de pantallas, que se observa en

la pantalla de configuraciones de la derecha, donde habilitarse la casilla *Enable X11 forwarding* y debe escribirse la locación del display, por defecto se interpreta como 0.0 si la casilla de texto se deja en blanco.

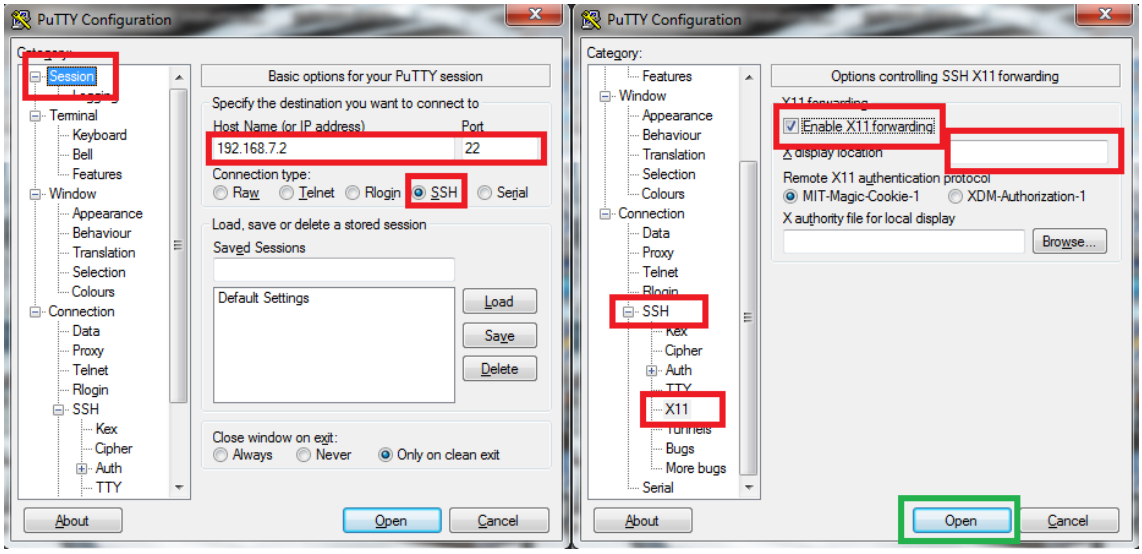


Figura 30: Configuraciones del software PuTTY para establecer una conexión Ethernet con la tarjeta embebida.

Una vez establecida la conexión es necesario iniciar sesión para iniciar a hacer uso de la tarjeta embebida, para el caso de la BeagleBone Black se puede acceder directamente como ***root*** lo cual permite tener privilegios de administrador (o como usuario ***debian*** y contraseña ***tmppwd***), en cambio para la tarjeta Rasberry Pi 3 se debe acceder con usuario ***pi*** y pedirá una contraseña que será ***raspberry***, es importante mencionar que estos usuarios y contraseñas vienen por defecto definidos en los sistemas operativos correspondientes. En la Figura 31 se indica cómo se ha iniciado sesión con una tarjeta BeagleBone Black, en primera instancia se distingue la versión de sistema operativo (Debian GNU/Linux 8).

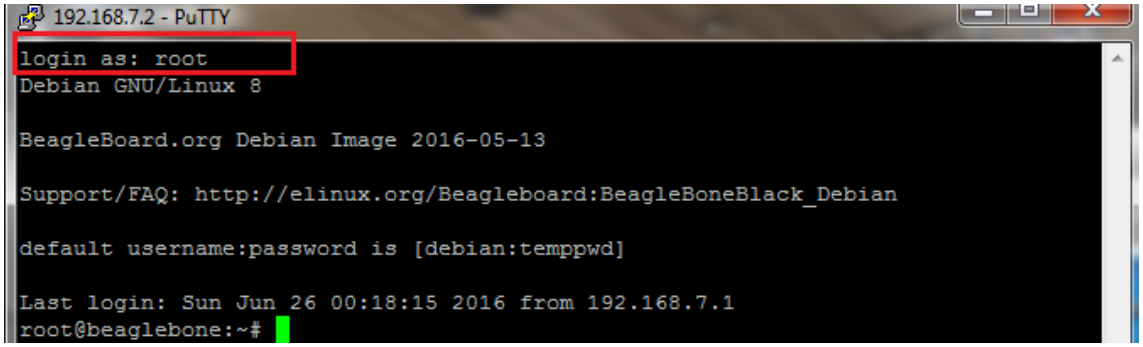


Figura 31: Inicio de sección como root para acceder al manejo de una tarjeta BeagleBone Black.

Para que Raspberry Pi 3 pueda manejarse a través de PuTTY primero es necesario definir una dirección IP para conectarse con el ordenador por Ethernet; para ello, en primera instancia, se debe utilizar a esta tarjeta como un computador convencional, es decir, conectándolo a un monitor por el puerto HDMI y a periféricos de entrada (teclado y mouse) por los puertos USB, de esta forma se debe acceder a la pantalla de configuración de red ***Network Preferences*** como se indica en la Figura 32 donde queda definida una IP 192.168.10.20, y en el ordenador se establecerá una IP relacionada (192.168.10.x) como se indica en la Figura 33.

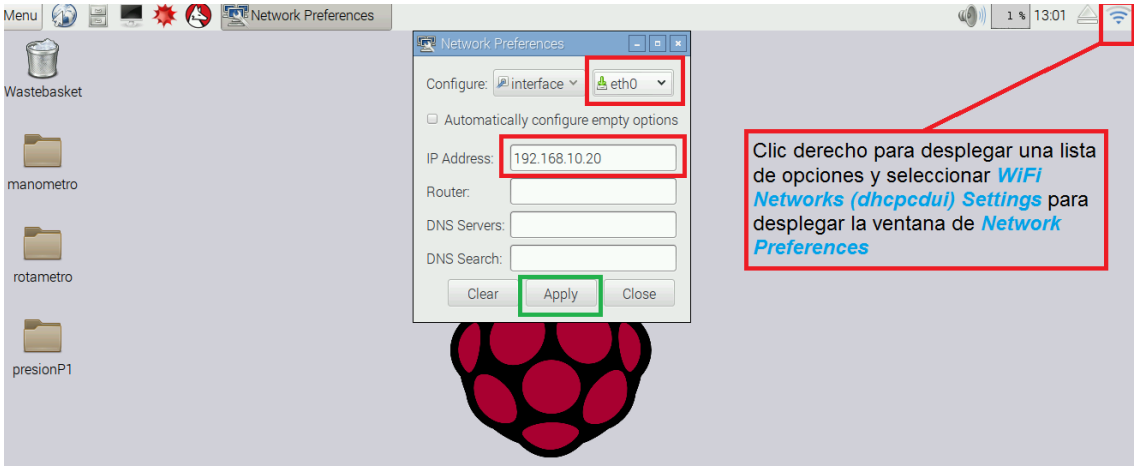


Figura 32: Configuración que establece una IP fija para la conexión Ethernet en la tarjeta Raspberry Pi 3 (utilizada como un computador convencional).

Nota: *También será necesario utilizar la tarjeta Raspberry Pi 3 como un computador convencional, cuando se descargue paquetes de software desde Internet previo a su instalación (porque se requiere del puerto Ethernet para la conexión con el modem), y definir las configuraciones de la conexión inalámbrica (similar a la configuración de la dirección IP para la conexión Ethernet), lo cual se indica en las secciones 3.2.2 y 3.2.3.*

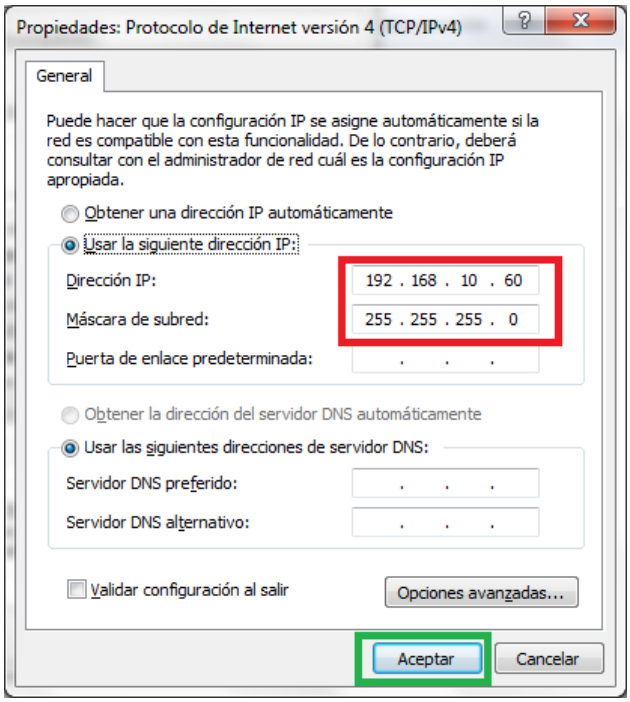


Figura 33: Definición de una IP (192.168.10.60) en el ordenador desde dónde se manejará la tarjeta Raspberry Pi 3 a través de PuTTY.

En la Figura 34 se puede observar cómo se ha producido el acceso remoto a la tarjeta Raspberry Pi 3 utilizando PuTTY de la misma manera como se indicó para la BeagleBone Black, pero utilizando la IP 192.168.10.20 que ya se encuentra definida; en los recuadros celestes se indica que se ha ingresado el usuario y contraseña mencionados anteriormente, se puede apreciar que en la contraseña no muestran los caracteres y se puede apreciar que el sistema operativo Raspbian también se muestra como una versión de Debian GNU/Linux.

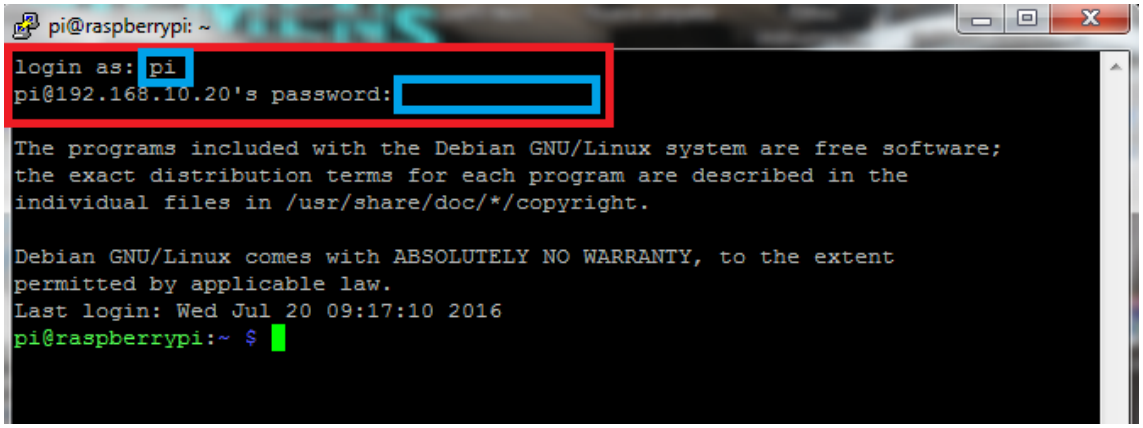


Figura 34: Inicio de sesión de la tarjeta Raspberry Pi 3 en PuTTY.

Es necesario indicar que se utilizan algunos comandos básicos que se ejecutan en la terminal de un sistema operativo Linux y sirven para acceder y moverse a través de carpetas donde se modificarán códigos de configuración o se editará el código del algoritmo. Estos comandos básicos se pueden observar en la Figura 35, ***cd*** permite acceder a una carpeta, ***cd ..*** sirve salir de la carpeta actual, ***ls*** despliega todo los elementos contenidos en una carpeta, y para crear una nueva carpeta o directorio se usa el comando ***mkdir nombre***; de esta manera se creará el directorio donde estarán contenidos todos los archivos de código o datos que se usarán en el proyecto (recuadro celeste).

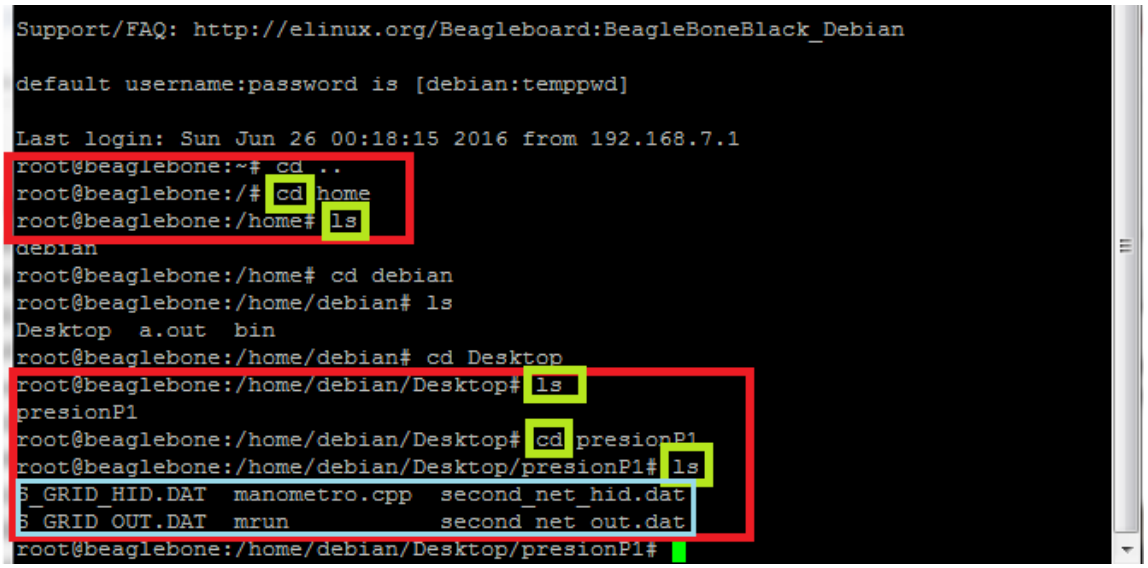


Figura 35: Manejo de comandos básicos en la terminal.

3.2.2. Instalación de OpenCV

En la imagen del sistema operativo Debian para BeagleBone Black están preinstaladas por defecto las librerías de OpenCV (Molloy, 2014), de tal forma que en este caso no es necesario realizar algún procedimiento; no es así en la tarjeta Raspberry Pi 3, por lo que se sigue el proceso de instalación que se indica a continuación, es importante resaltar que éste procedimiento puede realizarse en cualquier hardware embebido con sistema operativo Linux.

Por facilidad de manejo durante la instalación se recomienda utilizar la tarjeta Raspberry Pi 3 como un computador convencional, esto debido a que es necesario descargar paquetes de instalación, es decir, se conectará la tarjeta a un modem con acceso a Internet a través de Ethernet para que el proceso de descarga e instalación sea mucho más rápido; para posibilitar la conexión a Internet, momentáneamente debe habilitarse la opción de ***Automatically configure empty options*** de la ventana ***Network Preferentes*** que se indica en la Figura 36.

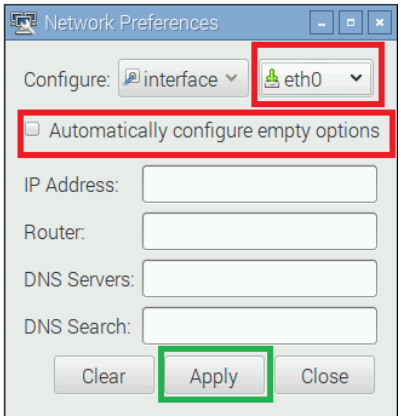


Figura 36: Configuración de red automática para establecer una conexión a Internet por Ethernet.

Para verificar la conexión a internet y las actualizaciones de los paquetes ya instalados, como en toda instalación que se realiza en un sistema operativo Linux, se inicia ejecutando los siguientes comandos en la terminal como administrador, esto es anteponiendo el comando ***sudo***:

```
pi@raspberrypi:~$ sudo apt-get update
pi@raspberrypi:~$ sudo apt-get upgrade
```

Es un requisito instalar primero los paquetes de dependencias de software que corresponden a procesos de compilación, entre ellos se incluye el lenguaje de programación C++, en el cual será editado todo el código del algoritmo, para ello se digita la siguiente composición de comandos:

```
pi@raspberrypi:~$ sudo apt-get –y install build-essential cmake cmake-curses-gui pkg-config
```

A continuación se instala un segundo bloque de paquetes de dependencias específicas para que sea posible la compilación de las librerías de OpenCV, algunas de ellas tienen que ver con el manejo de formatos de imágenes y video, se digita lo siguiente en la terminal:

```
pi@raspberrypi:~$ sudo apt-get –y install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev libavcodec-dev libavformat-dev libswscale-dev libv41-dev libeigen3-dev libxvidcore-dev libx264-dev libgtk2.0-dev
```

Se procede a descargar OpenCV desde la fuente oficial, en este caso se ha utilizado la versión OpenCV 3.1.0 que viene comprimida (.zip), disponible en el sitio web https://github.com/Itseez/opencv/archive/3.1.0.zip, para ello se utiliza el comando ***wget*** de la siguiente manera:

```
pi@raspberrypi:~$ sudo wget https://github.com/Itseez/opencv/archive/3.1.0.zip
```

A continuación se debe extraer la carpeta que contiene todos los archivos de instalación de la carpeta comprimida descargada; luego se accederá a esta carpeta llamada *opencv-3.1.0* para crear una nueva carpeta (*release*) donde se compilará OpenCV, se ejecutan los siguientes comandos uno por uno:

```
pi@raspberrypi:~$ unzip opencv-3.1.0.zip
pi@raspberrypi:~$ cd opencv-3.1.0
pi@raspberrypi:~$ sudo mkdir release
pi@raspberrypi:~$ cd release
pi@raspberrypi:~$ sudo ccmake ../
```

Posterior a la ejecución de los comandos anteriores se desplegará en la terminal una extensa lista de opciones modificables (como se indica en la Figura

37) para la configuración de OpenCV previo a su instalación definitiva, en este caso se ha utilizado la configuración que viene por defecto, por lo que se presiona la tecla '***c***' para que quede establecida la configuración y luego se presiona la tecla '***g***' para generar automáticamente el archivo de la configuración (denominado *makefile*) que empezará a ejecutarse (este proceso durará varias horas). Al finalizar ésta ejecución del *makefile* deberá ejecutarse los siguientes comandos para finalizar la instalación de OpenCV:

```
pi@raspberrypi:~$ make
pi@raspberrypi:~$ sudo make install
```

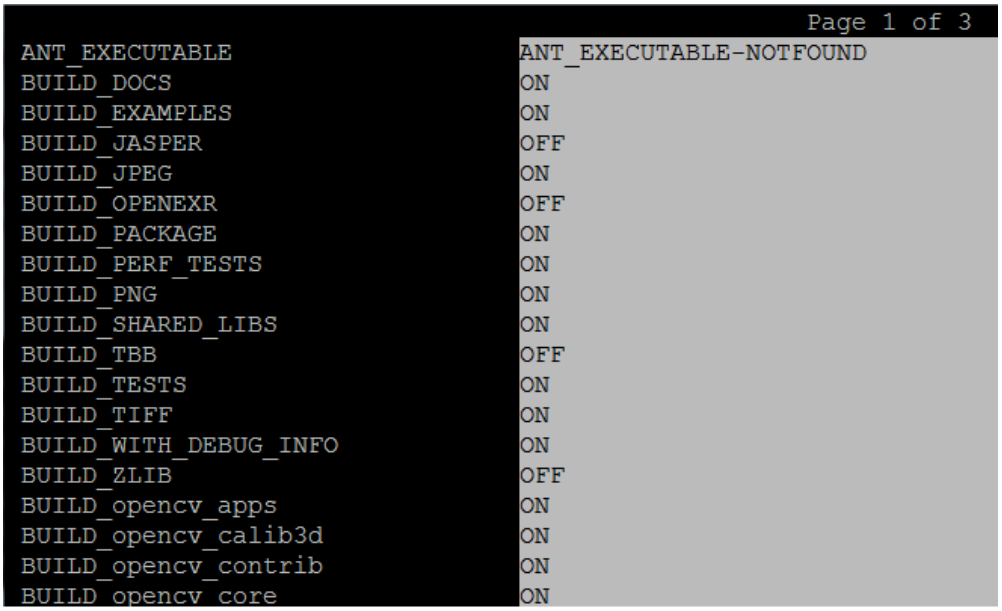


Figura 37: Lista de opciones de configuración de OpenCV.

Finalizado el proceso de instalación en la tarjeta embebida, se procede a establecer la ruta dónde se han instalado todas las librerías de OpenCV para que sea posible la compilación de los programas que empleen éstas librerías, por defecto se encuentran en la ruta ***/usr/local/lib***; se puede verificar lo mencionado, como se observa en la Figura 38, accediendo a la ruta utilizando los comandos:

```
pi@raspberrypi:~$ cd ..
pi@raspberrypi:/$ cd usr
pi@raspberrypi:/usr$ cd local
```

```
pi@raspberrypi:/usr/local$ cd lib
pi@raspberrypi:/usr/local/lib$ ls
```

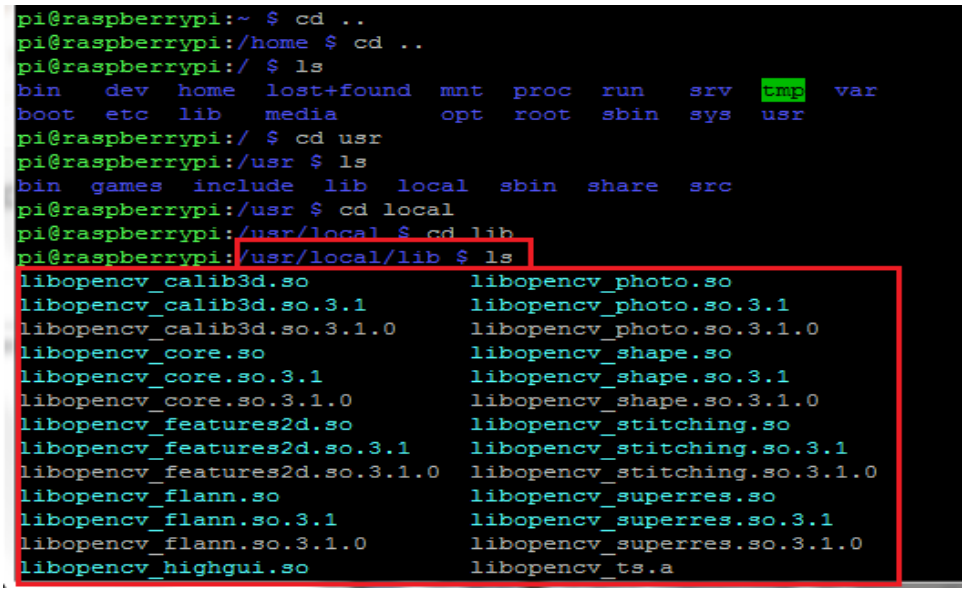


Figura 38: Verificación de la instalación de las librerías de OpenCV en la ruta */usr/local/lib*.

Para establecer la ruta dónde se encuentran las librerías se debe editar el archivo oculto ***.bashrc*** (ejecutando el comando ***ls -a*** se despliegan todos los archivos contenidos en una carpeta incluyendo los ocultos), es necesario utilizar el comando ***nano***, el cual corresponde a un editor de texto propio de GNU/Linux, que permite la edición del archivo como se indica en la Figura 39.

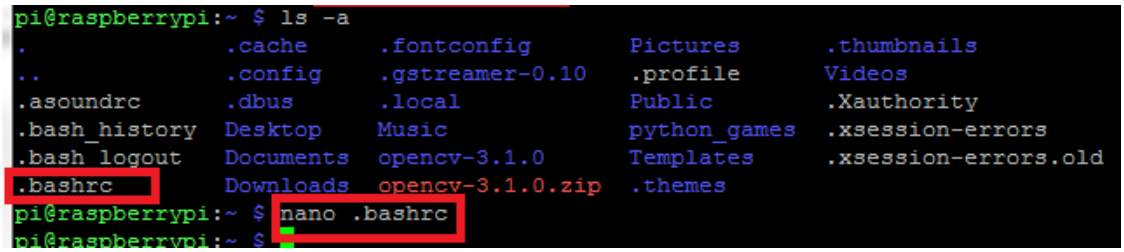


Figura 39: Apertura de edición del archivo *.bashrc*.

Una vez abierto el archivo ***.bashrc*** se escribe la línea de comando contenida en el siguiente recuadro, al final de todo el código contenido como se indica en la Figura 40; finalmente se debe salir del modo de edición con la combinación de

teclas ***Ctrl+x*** y pedirá guardar los cambios presionando la tecla '***y***'. Al final de este paso se debe reiniciar el dispositivo para que los cambios se efectúen.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

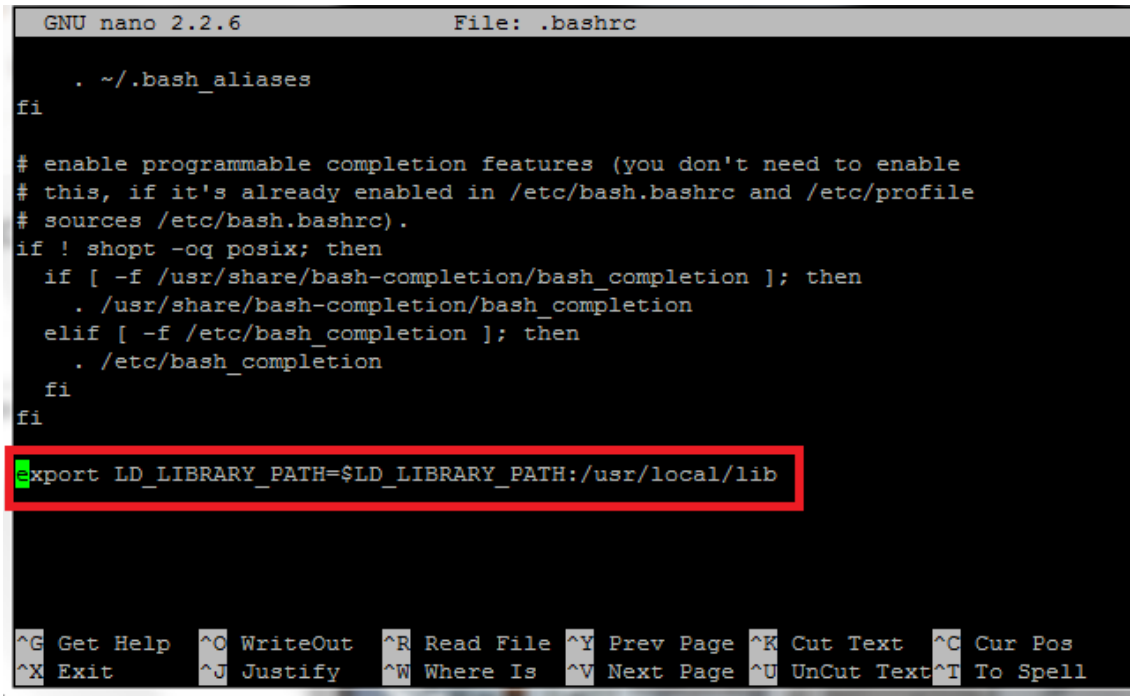


Figura 40: Establecimiento de la ruta dónde se encuentran instaladas las librerías de OpenCV.

Nota: *Luego de instalar OpenCV en un dispositivo embebido puede ser necesario expandir la capacidad de almacenamiento de la tarjeta microSD, que inicialmente se encuentra definida por la imagen del sistema operativo que fue instalada; existe información en la web para realizar éste procedimiento, por ejemplo para la BagleBone Black se puede encontrar en el sitio http://elinux.org/Beagleboard:Expanding_File_System_Partition_On_A_microSD*

3.2.3. Configuraciones de la conexión inalámbrica WIFI

Para iniciar con las configuraciones de red inalámbrica de las tarjetas embebidas es necesario disponer de una red WIFI, para lo cual se utiliza un router WIFI (D-LINK modelo DIR-905L) y se lo configura siguiendo un procedimiento

convencional, semejante al que se sugiere en el enlace ftp://ftp.dlinkla.com/pub/DIR-905L/DIR-905L_A1_QIG_v1.20(DLA).pdf, por nombrar algunos parámetros importantes definidos se tiene: nombre de la red (SSID) *VISION*, contraseña *computervision*, puerta de enlace predeterminada *192.168.0.1*, máscara de red *255.255.255.0* y asignación de IP dinámica por parte del router (protocolo DHCP).

En tarjeta BeagleBone Black no está incluido un módulo de comunicación inalámbrica por lo que es necesario utilizar un adaptador WIFI tipo USB, pero, con el requisito de que éste adaptador esté basado en un *chipset* de la serie *REALTEK RTL8192* para que sea compatible con la tarjeta (Molloy, 2014). Para este trabajo se ha utilizado un adaptador WIFI de bajo costo LB-LINK modelo BL-WN2210; adicionalmente es necesario incluir un HUB USB que no requiera de una fuente de poder adicional (adaptador de un puerto USB a varios puertos USB) porque se requiere dos puertos USB, para el adaptador WIFI y la cámara Logitech C920. Al momento de utilizar el adaptador WIFI y el HUB USB no es suficiente con energizar la tarjeta a través del puerto mini USB, se debe conectar una fuente de 5 V y 2 A por el Jack DC.

Se debe instalar el *firmware* del adaptador WIFI en la tarjeta BeagleBone Black, para ello ésta debe tener acceso a internet conectándola a un modem a través del puerto Ethernet; para conocer cuál es el paquete de instalación que corresponde al chipset *REALTEK RTL8192* y si está disponible (debido a que siempre existen actualizaciones de paquetes para Linux), utilizando PuTTY, se introducen los siguientes comandos en la terminal:

```
root@beaglebone:~# sudo apt-get update
root@beaglebone:~# apt-cache search RTL8192
```

En la terminal se desplegará el nombre del paquete, en este caso se presenta como *firmware-realtek*, de tal forma que se procede a la instalación ejecutando la línea de comando que se indica:

```
root@beaglebone:~# sudo apt-get install firmware-realtek
```

A continuación se debe acceder al archivo *interfaces* que contiene una serie de comandos de configuraciones de red y se encuentra en la ruta *etc/network*; para iniciar a editar el archivo se utiliza el editor de texto *nano* ejecutando los siguientes comandos en la terminal:

```
root@beaglebone:/# cd etc/network
root@beaglebone:/etc/network# nano interfaces
```

Al final del contenido del archivo *interfaces* se introduce los comandos de configuración para establecer una red inalámbrica *wlan0*, que se encuentran enmarcados en la Figura 41; aquí se definen los parámetros propios de la red inalámbrica establecida (VISION), así como la dirección IP estática que se le asignará a la tarjeta BeagleBone Black (en este caso 192.168.0.30).

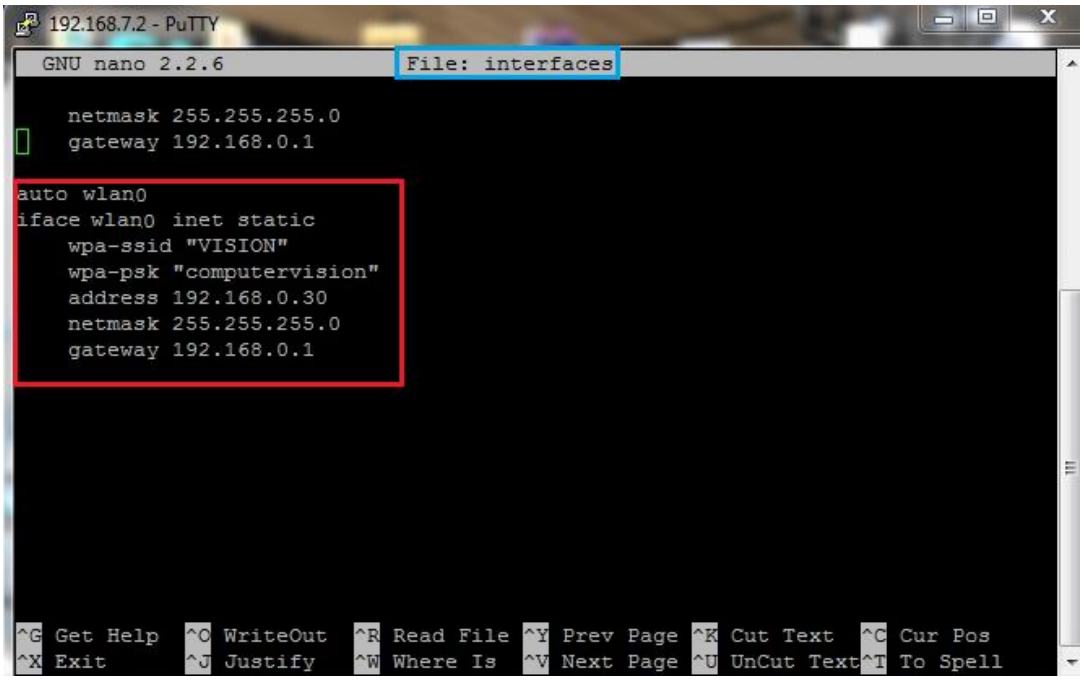


Figura 41: Configuración de una red inalámbrica WIFI en el archivo *interfaces*.

Para que los cambios realizados en el archivo *interfaces* produzcan la conexión de la tarjeta a la red WIFI, se debe reinicializar la red *wlan0* configurada utilizando los comandos i***fdown*** para desactivarla e ***ifup*** para activarla; se puede verificar que se hayan establecido los parámetros definidos (SSID, IP, etc.)

utilizando el comando ***ifconfig -a*** que desplegará en la terminal la información de todas las redes instauradas en la tarjeta, incluyendo la información de *wlan0*, los comandos se introducen de la siguiente manera:

```
root@beaglebone~# ifdown wlan0
root@beaglebone~# ifup wlan0
root@beaglebone~# ifconfig -a
```

En el caso de la tarjeta Raspberry Pi 3, que está equipada de fábrica con un módulo WIFI, el sistema operativo permite configurar y agregar redes inalámbricas de forma sencilla desde las herramientas del escritorio (si se emplea a la tarjeta como un computador convencional) como se observa en la Figura 42; dónde solo es necesario conectarse a la red e introducir la clave, y accediendo a la ventana de *Network Preferences* se establece la dirección IP estática que se ha definido como 192.168.0.20 para ésta tarjeta embebida.

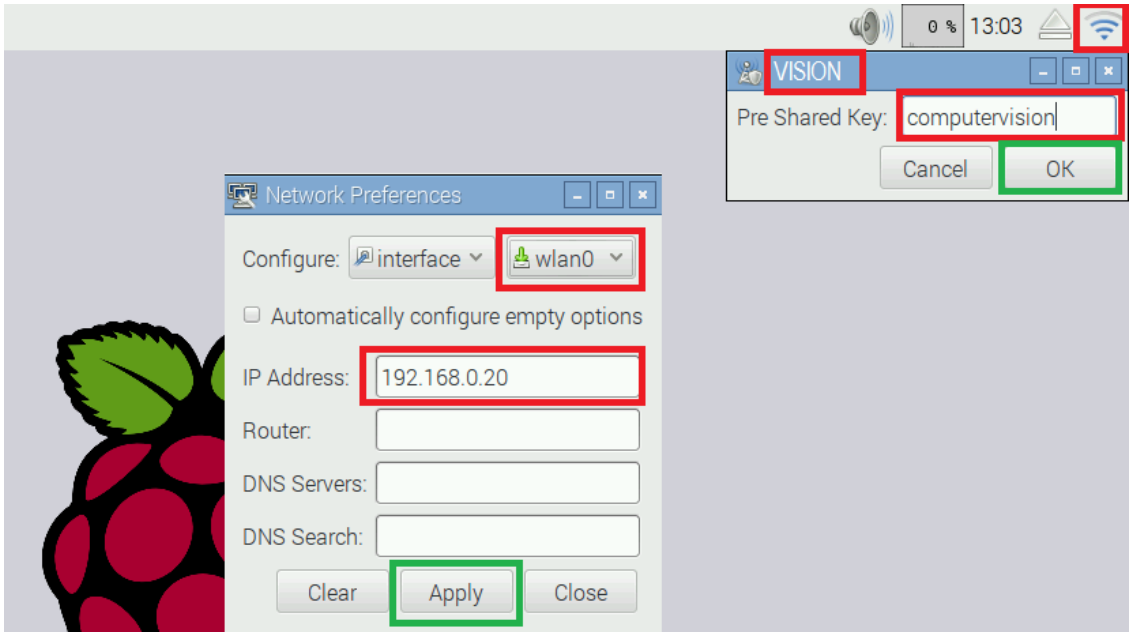


Figura 42: Configuración de una red inalámbrica WIFI en el archivo *interfaces*.

También es posible realizar modificaciones de una red inalámbrica (o agregar una nueva) utilizando la terminal de Raspbian, siendo posible acceder desde PuTTY; en la Figura 43 se indica el acceso en modo lectura al archivo *interfaces*

mediante en comando ***more*** donde, a diferencia de Debian para BeagleBone Black, está establecida una nueva ruta de un archivo *wpa_supplicant.conf* que contiene todos los parámetros de un red inalámbrica *wlan0* y el archivo *dhcpcd.conf* en la ruta */etc/* donde se definen las direcciones IP estáticas; sin embargo, esto resulta un procedimiento más complejo por el manejo de un mayor número de archivos del sistema operativo.

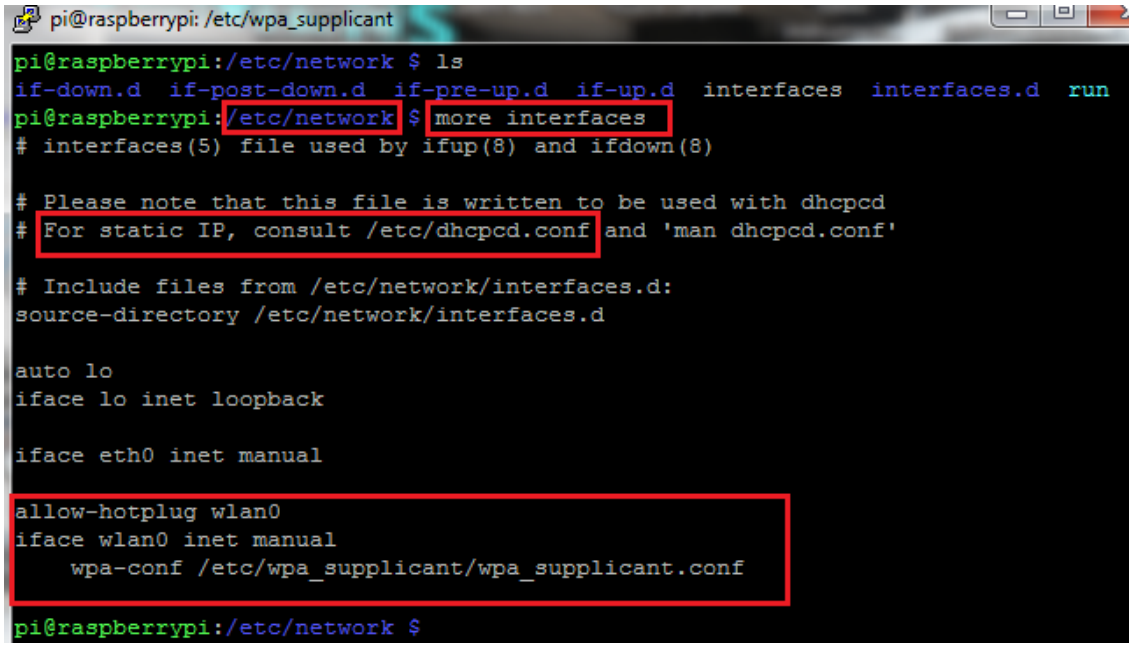


Figura 43: Configuración de una red inalámbrica WIFI en el archivo *interfaces*.

Tanto en la tarjeta BeagleBone Black como en la tarjeta Raspberry Pi 3 es posible comprobar que éstas se encuentran en red haciendo un *ping* desde la terminal a cualquier dispositivo que esté conectado a la misma red; en este caso se hará un *ping* al ordenador que actúa como sistema servidor del sistema del monitoreo, que se lo ha establecido con la dirección IP 192.168.0.102, como se indica en la Figura 44. Para detener la ejecución del ping, y cualquier otro comando o programa que se esté ejecutando en la terminal, se debe utilizar la combinación de teclas ***Ctrl+c***.

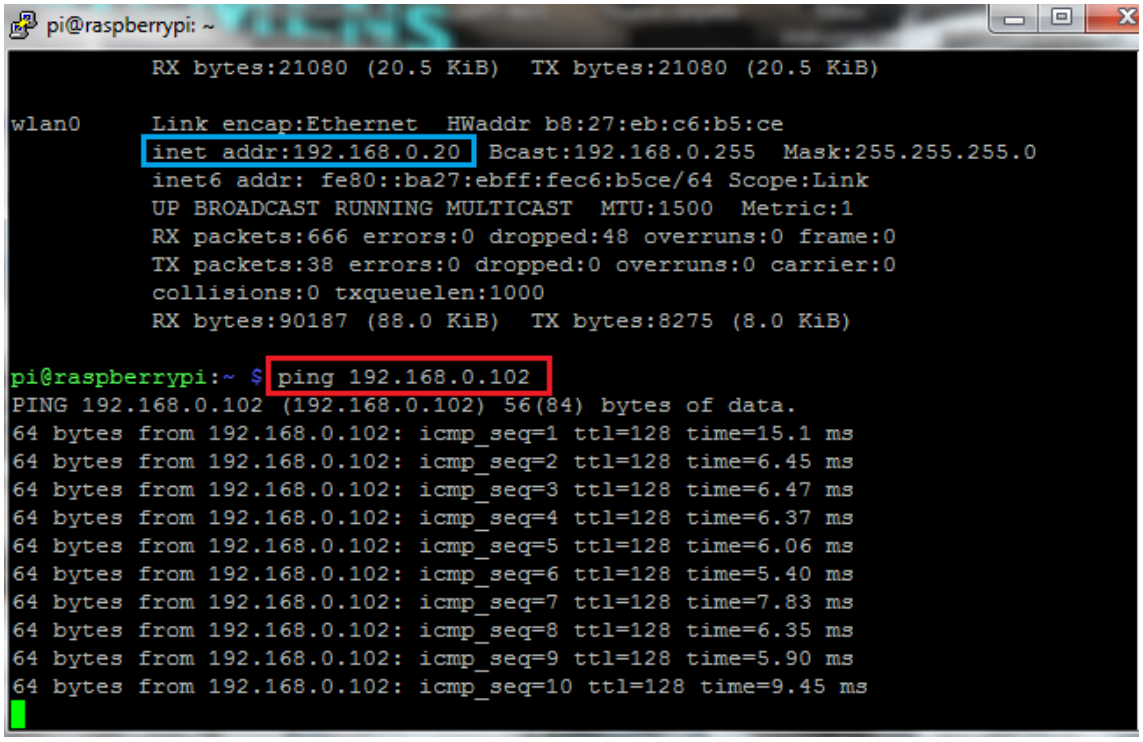


Figura 44: Comprobación de la conexión a la red inalámbrica mediante un ping al dispositivo con dirección IP 192.168.0.102.

3.2.4. Configuración del arranque de la tarjeta

Las tarjetas que se utilizan en este trabajo deben operar como una plataforma embebida, es decir, que inicien la ejecución del algoritmo que ha sido programado apenas son energizadas; para ello se debe editar el archivo ***rc.local***, ubicado en la ruta */etc/*, que se ejecuta mientras Linux está iniciando y antes de que sea necesario iniciar una sesión de usuario; en este archivo se especifica los comandos y el archivo ejecutable (generado luego de la compilación del algoritmo escrito en C++ que se especifica en la sección 3.3) que deben ejecutarse luego de los recursos esenciales del sistema operativo.

Como se observa en la Figura 45 se crea un script en el archivo rc.local que, en primer lugar, reinicia la red *wlan0* configurada para garantizar la conectividad inalámbrica con los parámetros definidos; se procede a la apertura de la ruta donde se ha almacenado el archivo ejecutable del algoritmo desarrollado, en este

caso corresponde algoritmo de lectura de la medida del rotámetro que se lo ha denominado como *rrun* y se lo ejecuta mediante el comando ***./rrun***; finalmente se termina el script de arranque con el comando *exit 0*.

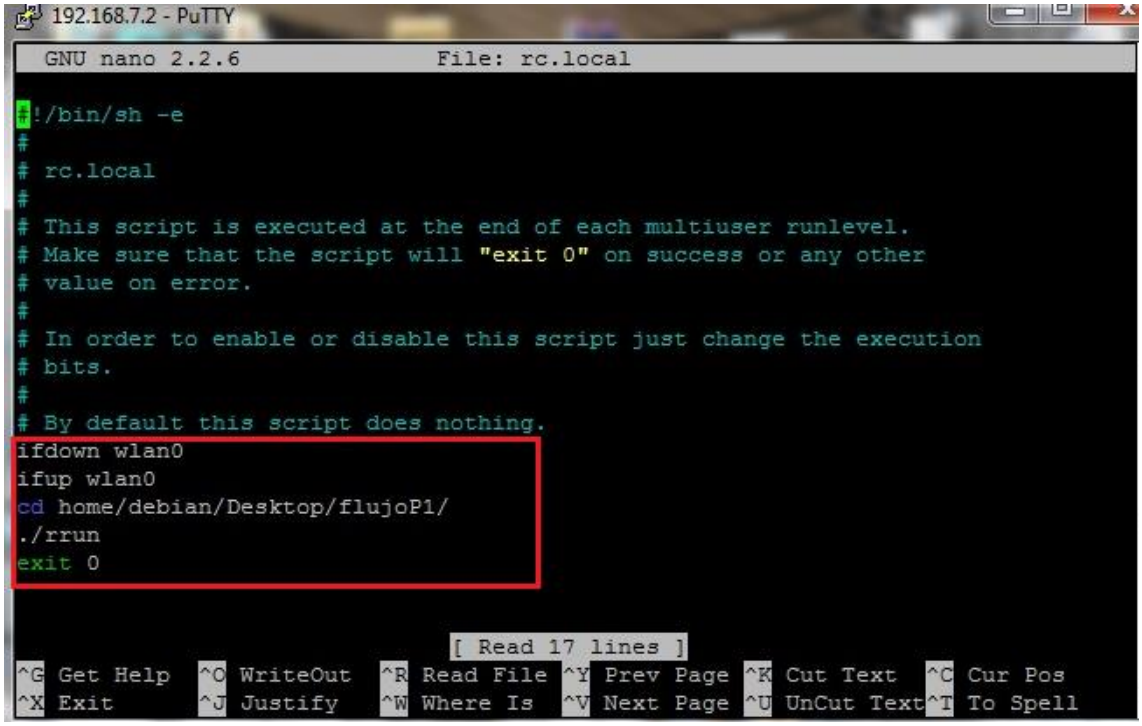


Figura 45: Configuración del arranque de una tarjeta embebida con sistema operativo Linux en el archivo *rc.local*.

3.2.5. Sumario del hardware embebido utilizado en la operación del sistema de monitoreo

En la Figura 46 se observa todos los componentes de hardware que se encuentran asociados en cada una de las tarjetas embebidas, mismos que han sido mencionados en el transcurso de esta sección; se ha considerado que la tarjeta Raspberry Pi 3 sea utilizada para ejecutar el algoritmo de lectura de la medida del rotámetro, y la BeagleBone Black para la lectura de la medida presentada por un manómetro, en este caso es necesario la adición de un sistema de iluminación frontal basado en tecnología LED (los detalles de este requerimiento se explican en la sección 3.3.4).

El sistema de iluminación es suministrado de 5 V a través de los pines 1 (GND) y 5 (5 V) del bloque P8 y representa un consumo de corriente aproximado de 20 mA. Las tarjetas microSD donde están cargadas las imágenes de los respectivos sistemas operativos son de 8 GB y de clase 10, según las recomendaciones de las páginas web oficiales de soporte de las tarjetas. Ambas tarjetas se energizan con adaptadores de 110 V AC a 5 V DC de al menos 2 A de salida.

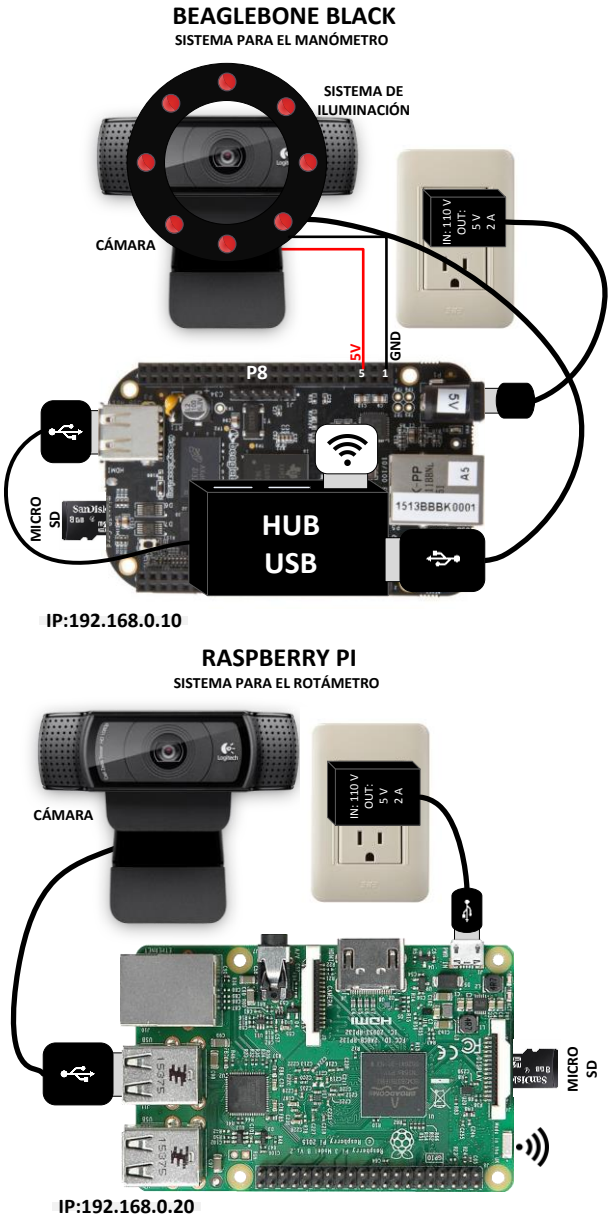


Figura 46: Conjunto del hardware utilizado por el sistema propuesto.

3.3. Desarrollo del algoritmo de visión artificial para la lectura de la medida en la escala de los instrumentos

El algoritmo de visión artificial creado está basado en la utilización de RNA para el reconocimiento de formas correspondientes a las escalas de los instrumentos y sus respectivos indicadores; por esta razón se inicia especificando la estructuras de las RNA utilizadas y su operación (secciones 3.3.1 y 3.3.2) que intervienen y/o dependen de las demás rutinas (descritas de la sección 3.3.3 a la 3.3.6), y tienen como fin determinar el valor de la variable que se lee en un instrumento de forma automática, emulando la acción de lectura que hace un operador por medio del sentido de la vista. Se explicará secciones del código, las versiones completas correspondientes al manómetro y rotámetro es encuentran en el Anexo A y el Anexo B respectivamente.

El algoritmo se ha implementado considerando que se ejecutará en una plataforma embebida (como se menciona en la sección 3.1.2), por lo que las rutinas que involucran procesamiento de imágenes y cómputos de RNA se han desarrollado de un modo que cumplen con sus respectivos propósitos sin complicar la velocidad de procesamiento de la tarjeta, así también se contempla que la ejecución iniciará automáticamente y operará como un bucle sin fin, de tal manera que únicamente será necesario energizar y/o reiniciar las tarjetas embebidas por parte de un operador.

En la Figura 47 se estructura el modo de operación del algoritmo propuesto, generalizado para cualquier tipo de instrumento analógico que posea una escala y un indicador móvil; los procesos enmarcados en color azul corresponden a rutinas de inicialización y necesarias para las siguientes etapas, los enmarcados en color verde definen la operación normal y continua del algoritmo, el color naranja corresponde a rutinas emergentes que se habilitan cuando no sea posible determinar el valor de la variable, y el color rojo representa a un aviso de error, el cual indica que definitivamente no será posible determinar el valor de la variable porque no ha sido posible que el algoritmo encuentre e identifique la imagen de la escala instrumento.

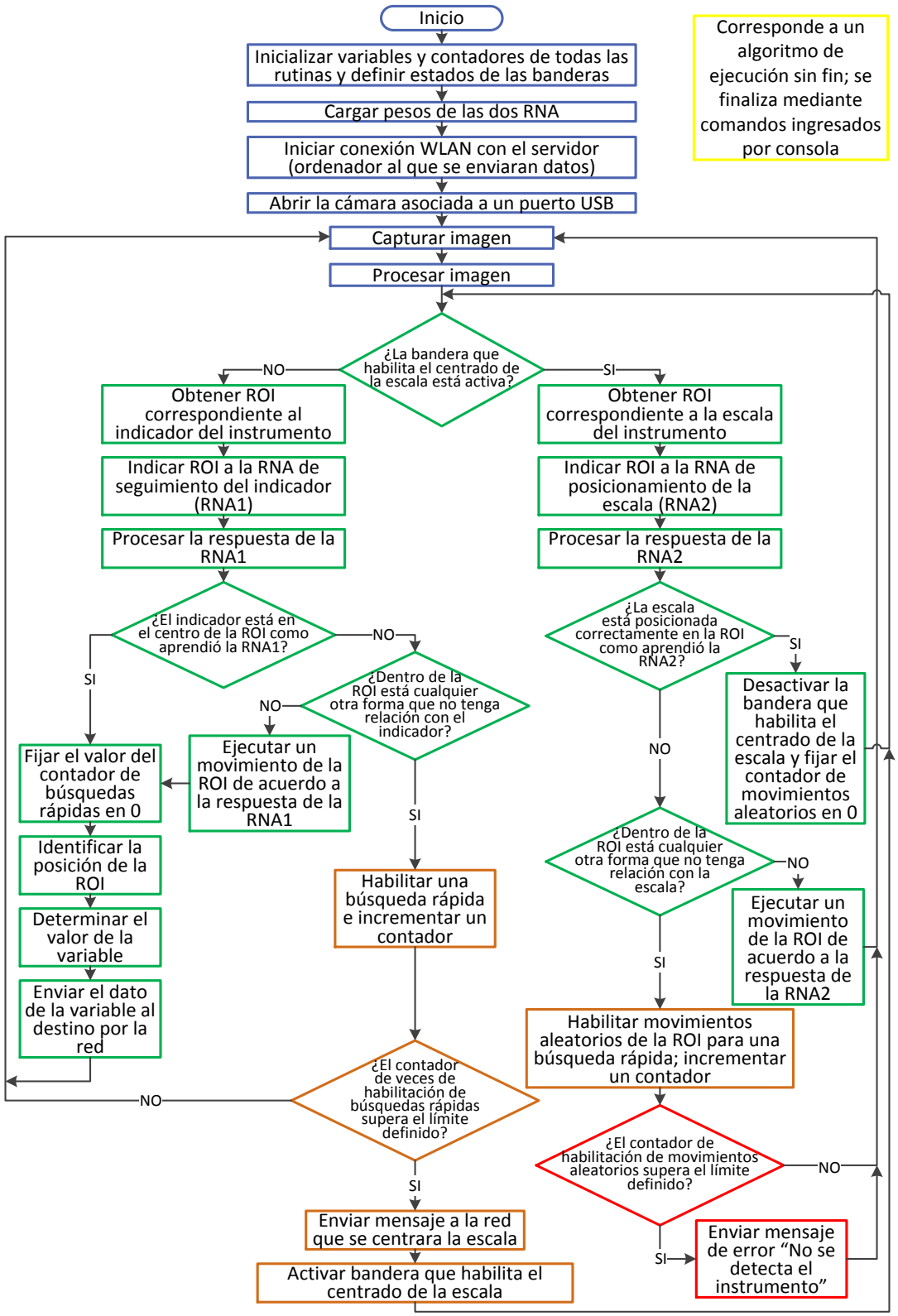


Figura 47: Diagrama de flujo del algoritmo de ejecución del sistema embebido.

Es importante mencionar que es posible compilar y ejecutar los algoritmos en cualquiera de las dos tarjetas utilizadas, pero, se ha definido que en la BeagleBone Black se utilice para el algoritmo correspondiente al manómetro y para el rotámetro la Raspberry Pi 3. El algoritmo se ha escrito en lenguaje C++ (utilizando el editor de texto *nano* de Linux para crear un archivo **.cpp*) de tal manera que se utiliza el compilador g++ para realizar la compilación del programa y crear el archivo ejecutable. Se define un directorio donde se almacena el programa C++ y los archivos que contienen datos de las RNA (pesos de conexión entre neuronas), mediante la terminal se accede a este directorio y se ejecuta el conjunto de comandos que se indica a continuación para compilar el programa que corresponde al manómetro:

```
root@beaglebone:/home/debian/Desktop/presionP1# g++ -o mrun manometro.cpp `pkg-config opencv --cflags --libs`
```

El comando ***–o*** define el nombre del archivo ejecutable denominado ***mrun***, ***manometro.cpp*** es el nombre del archivo editado en lenguaje C++ y el comando ***`pkg-config opencv --cflags --libs`*** especifica al proceso de compilación que se ha utilizado librerías de OpenCV dentro del código para que sean tomadas en cuenta y no se generen errores de compilación; el directorio donde se almacena todo este conjunto de archivos corresponde a */home/debian/Desktop/presionP1*. De forma similar en el caso del algoritmo correspondiente a la lectura de la medida en el rotámetro se tiene:

```
pi@raspberrypi:/home/debian/Desktop/flujoP1$ g++ -o rrun rotametro.cpp `pkg-config opencv --cflags --libs`
```

3.3.1. Estructura de las RNA

El algoritmo está compuesto por dos RNA de tres capas que tienen como tarea identificar las formas contenidas en una región de interés (ROI), extraída de la etapa de procesamiento de imágenes (sección 3.3.4), que es una imagen binaria con píxeles de color negro y blanco asignados con valores de 0.1 y 0.9 respectivamente; éstos valores son asignados, uno a uno, a cada neurona de la

capa de entrada de una RNA. El tamaño de la ROI utilizada se define identificando las características de la imagen que brindarán información de la forma de la escala del instrumento y el indicador, que serán identificados por la respectiva RNA.

Una de las RNA es empleada en la rutina de reconocimiento y posicionamiento de la región de la escala del instrumento analógico, la capa de entrada consta de 20000 neuronas suministradas de valores de una ROI de 100x200 píxeles en el caso del manómetro; el número de neuronas para la capa escondida son 30 y 7 neuronas en la capa de salida. La otra RNA se emplea en el seguimiento del indicador del manómetro, consta de una capa de entrada de 1000 neuronas correspondiente a una ROI de 50x20 píxeles, la capa escondida está conformada por 20 neuronas y la capa de salida por 6 neuronas. La Figura 48 contiene la estructura de las RNA descritas.

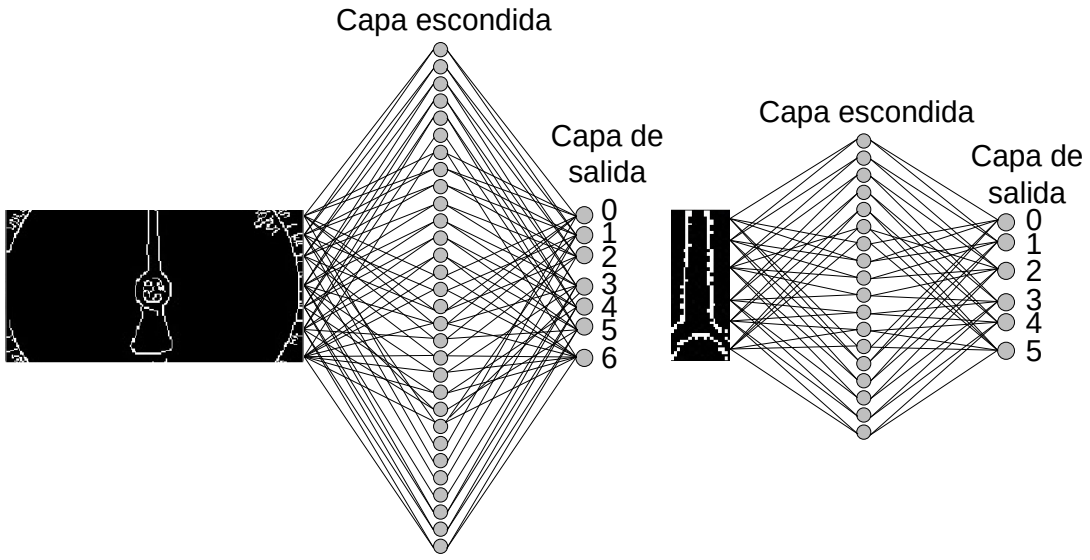


Figura 48: Estructuras de las RNA para el reconocimiento de la escala del manómetro (izquierda) y seguimiento del indicador (derecha).

La estructura las RNA utilizadas se precisa en tres capas, el número de neuronas de las capas entrada, salida y escondida se definen en las constantes *N_IN*, *N_SAL* y *N_HID* respectivamente; la capa de entrada está representada por un vector *Entrada* que contiene los valores de los píxeles blancos (0.9) y

negros (0.1); las capas de salida y escondida se definen en el código como una estructura de programación orientada a objetos, de modo que se los considera como objetos con sus respectivos elementos.

Los elementos de las capas de salida y escondida corresponden a: una matriz de valores de pesos de las conexiones sinápticas entre neuronas, un vector del valor de salida de cada neurona (comprendidos entre 0 y 1) y un vector de error del valor de la salida de cada neurona con respecto a los valores objetivos (utilizado únicamente para el proceso de entrenamiento al igual que el vector *Target* y la ganancia *Eta*). La ganancia *Gain* será utilizada por la función de activación. El código presentado a continuación contiene la estructura de la RNA utilizada en el reconocimiento del indicador del manómetro:

```
#define N_IN    1000    //NEURONAS DE ENTRADA
#define N_SAL   6       //NEURONAS DE SALIDA
#define N_HID   20      //NEURONAS ESCONDIDAS
float  Eta=0.025, Gain=0.5;
float Target[N_SAL];
float Entrada[N_IN];
struct  capa_salida{
            float  pesos[N_SAL][N_HID];
            float  error[N_SAL];
            float  out[N_SAL];
    } c_salida;
struct  capa_escondida{
            float  pesos[N_HID][N_IN];
            float  error[N_HID];
            float  out[N_HID];
    } c_escondida;
```

Para el caso de la RNA utilizada en el reconocimiento de la escala del manómetro se definen los mismos parámetros, pero, como se utilizará dentro del mismo código definido anteriormente, es necesario variar el nombre de los términos y el número de neuronas que se utilizan en cada capa, como se indica a continuación:

```
#define n_N_IN    20000     //NEURONAS DE ENTRADA
#define n_N_SAL   7         //NEURONAS SALIDAS
#define n_N_HID   30        //NEURONAS ESCONIDIDAS
float  n_Eta=0.25, n_Gain=0.5;
float n_Target[n_N_SAL];
float n_Entrada[n_N_IN];
```

```
struct  n_capa_salida{
            float  pesos[n_N_SAL][n_N_HID];
            float  error[n_N_SAL];
            float  out[n_N_SAL];
    } n_c_salida;
struct  n_capa_escondida{
            float  pesos[n_N_HID][n_N_IN];
            float  error[n_N_HID];
            float  out[n_N_HID];
    } n_c_escondida;
```

En el rotámetro se utiliza una ROI de 200x70 píxeles que contiene la zona de la escala y alimenta a una capa de entrada de 14000 neuronas; el número de neuronas para la capa escondida son 30 y 7 neuronas en la capa de salida. La RNA que se encarga del seguimiento del flotador consta de una capa de entrada de 600 neuronas de la extracción de ROI de 30x20 píxeles; la capa escondida está conformada por 20 neuronas y la capa de salida por 6 neuronas; esto se observa en la Figura 49. El código que define la estructura de la RNA para éste caso difiere, únicamente, del caso del manómetro en el número de las neuronas de entrada, por lo que se especifica la cantidad en las constantes *N_IN* y *n_ N_IN*.

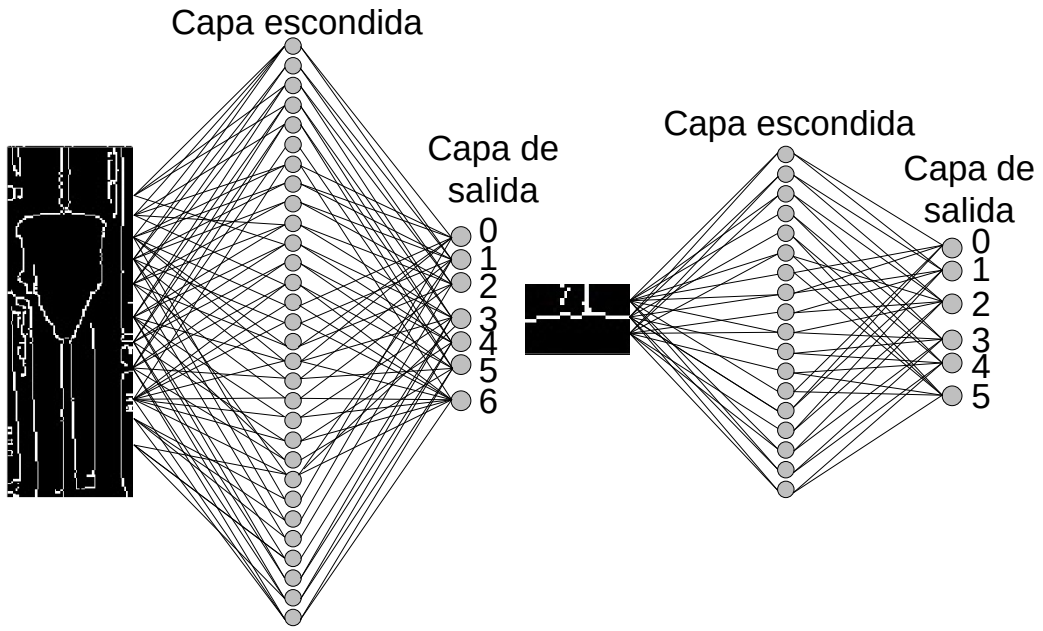


Figura 49: Estructuras de las RNA para el reconocimiento de la escala del rotámetro (izquierda) y seguimiento del flotador (derecha).

En esta sección también es importante describir la forma como se realiza el acceso al valor de los pesos sinápticos o conexiones entre neuronas que intervienen en los cálculos de las RNA; para esto se ha creado una función *CargarPesos,* cuya función es cargar de valores flotantes a los vectores *pesos* que son elementos de los objetos capa escondida (*c_escondida*) y capa de salida (*c_salida*), esto lo realiza accediendo al contenido de los archivos S_GRID_HID.DAT (contiene el valor de los pesos sinápticos de las neuronas de la capa escondida) y S_GRID.OUT.DAT (contiene el valor de los pesos sinápticos de las neuronas de la capa de salida) que han sido creados y modificados al finalizar el proceso de entrenamiento de las RNA; es necesario incluir el encabezado *#include <stdio.h>* para el manejo de funciones de apertura, búsqueda, lectura y cierre de archivos. Lo descrito se define en las siguientes líneas de código:

```
#include <stdio.h>
int CargarPesos(void){
	int k, i, aux;
	float numero;
	FILE *in_file;
	aux=0;
	in_file = fopen ("S_GRID_HID.DAT", "rt");
	fseek(in_file, 0, SEEK_SET);
	for (k=0; k<N_HID; k++){
		for (i=0; i<N_IN; i++){
			fscanf(in_file, "%f", &numero);
			c_escondida.pesos[k][i]=numero;
		}
	}

	fclose(in_file);
	in_file = fopen ("S_GRID_OUT.DAT", "rt");
	fseek(in_file, 0, SEEK_SET);
	for (k=0; k<N_SAL; k++){
		for (i=0; i<N_HID; i++){
			fscanf(in_file, "%f", &numero);
			c_salida.pesos[k][i]=numero;
		}
	}
	fclose (in_file);
	return(aux);
}
```

Es importante resaltar que se ha considerado emplear una cantidad de neuronas en cada capa, determinada de forma experimental, que permita el

cumplimiento del propósito de cada RNA, el tiempo empleado en el entrenamiento y, sobre todo, no represente un alto costo computacional en la tarjeta embebida porque se complicaría el seguimiento del indicador en tiempo real; esto debido a que al aumentar el número de neuronas en cada capa se incrementa el número de pesos de conexión entre neuronas y, por lo tanto, de iteraciones de cálculos de la RNA en el algoritmo (definidos en la sección 3.3.2), lo que disminuye la velocidad de adquisición de cuadros de imagen que capturan el movimiento del indicador del instrumento analógico.

3.3.2. Operación de las RNA

El comportamiento de las neuronas de la capa de salida de cada RNA define el funcionamiento del algoritmo propuesto, es decir, de todas las rutinas relacionadas al proceso de lectura del valor de la variable, que está basado en ubicar correctamente la escala del instrumento para luego identificar el indicador y seguirlo cuando éste se mueva, así se podrá determinar su ubicación dentro de la escala; para esto es necesario ejecutar desplazamientos de los recuadros que extraen porciones de la imagen procesada (ROIs) para suministrar de valores de entrada a las RNA correspondientes (como se indicó en las Figuras 48 y 49). Los desplazamientos ocasionan la aparición de varias formas de imagen binaria dentro de éstas ROIs, que serán identificadas por la activación de las neuronas de la capa de salida según corresponda.

En la *RNA de reconocimiento de la región de la escala* la activación de cada una de las siete neuronas de la capa de salida se interpreta de la siguiente manera: las tres primeras (de la 0 a la 2) se utilizan para determinar el desplazamiento de la ROI en el eje horizontal, siendo la activación de la neurona 1 la que indica cuando la ROI se encuentra en el centro, la neuronas 0 y 2 se activan cuando existe una desviación de un píxel o más hacia la izquierda y la derecha respectivamente; las siguientes tres neuronas son empleadas para determinar el desplazamiento en el eje vertical, las desviaciones se distinguen hacia arriba (neurona 3), hacia abajo (neurona 5) y el centro (neurona 4); y la última neurona 6 se activa cuando dentro de la ROI esté cualquier otra forma que

no corresponda a una porción de la escala del instrumento, esto como un método de emergencia que habilita una búsqueda rápida de la escala del instrumento.

Respecto a las *RNA empleadas para el seguimiento dinámico del indicador*, las neuronas de la capa de salida solo se activan una a la vez, por lo que es posible detectar si el indicador está ubicado justo en el centro de la ROI (activación neurona 2) o si existe una desviación de ángulo a la izquierda o derecha en el caso del manómetro, y de píxeles hacia arriba o abajo en el caso del rotámetro; las neuronas 1 y 3 se activan ante una desviación de 1 a 5 grados o píxeles en la dirección correspondiente según el instrumento, están enfocadas a brindar exactitud en la determinación de la posición del indicador, mientras que las neuronas 0 y 4 reconocen desviaciones de 6 a 20 grados en el indicador del manómetro y de 6 a 10 píxeles en el caso del borde del flotador del rotámetro, éstas son consideradas como elementos que introducen velocidad al seguimiento dinámico.

El cumplimiento de la activación de las neuronas de la capa de salida, según lo definido anteriormente, dependen del proceso de entrenamiento (sección 3.4), y se deben a la ejecución de los cálculos de las RNA que se producen cada vez que el algoritmo requiere reconocer una porción de imagen; los cálculos, que se realizan con cada una de las neuronas de la capa escondida y de la capa de salida, corresponden principalmente a una ponderación de su activación. La activación de las neuronas está basada en el modelo sigmoidal,

$$s_k(u) = \frac{1}{1+e^{-g\,u}} \tag{11}$$

donde s_k es el valor de salida de la neurona k, g es una ganancia positiva y u es el valor de la entrada efectiva dada por la sumatoria:

$$u = \sum_{j=1}^{n} w_j\, z_j \tag{12}$$

donde w_j es el valor del peso de conexión sináptica de la neurona con una entrada j (establecido en el proceso de entrenamiento) y z_j es el valor de la entrada j.

Estas operaciones se realizan con todas las neuronas contenidas en la RNA; se inicia con las neuronas de la capa escondida en la función *void calcular_capa_escondida(void)*, a cada neurona le corresponde como entradas los valores de los píxeles de la ROI (asignados 0.9 para blancos y 0.1 para negros) contenidos en el vector *Entrada[i]* y sus respectivos pesos de conexión contenidos en la matriz *c_escondida.pesos[k][i]*, los resultados de las multiplicaciones sucesivas entre éstos parámetros se van sumando para luego aplicarse la función de activación implementada en la función *float sigmoide(float u)* (requiere de funciones contenidas en la cabecera *#include <math.h>*), el resultado final corresponde al valor de activación de la neurona *k* y se almacenan en el vector *c_escondida.out[k]*.

El mismo procedimiento se aplica para las neuronas de la capa de salida, detallado en la función *calcular_capa_salida(void)*, solo que los valores obtenidos en *c_escondida.out[k]* corresponden ahora como entradas de cada una de éstas neuronas y los resultados se obtienen en *c_salida.out[k]*. Lo mencionando se estructura en el código que se presenta a continuación:

```
float sigmoide(float u){
	float out;
	if (u>30) u=30; if (u<-30) u=-30; //para valores de u fuera del intervalo -30 a 30 el resultado de la función sigmoide tienden mucho más 0 y 1 respectivamente
	float out=pow((1+exp(-Gain*u)),-1);
	return(out);
}
void calcular_capa_escondida(void){
	for (int k=0; k<N_HID; k++){
		c_escondida.out[k]=0;
		for (int i=0; i<N_IN; i++){
		    c_escondida.out[k]=c_escondida.out[k]+c_escondida.pesos[k][i]*Entrada[i];
		}
		c_escondida.out[k]=sigmoide(c_escondida.out[k]);
	}
}
void calcular_capa_salida(void){
	for (int k=0; k<N_SAL; k++){
		c_salida.out[k]=0;
		for (int i=0; i<N_HID; i++){
		    c_salida.out[k]=c_salida.out[k]+c_salida.pesos[k][i]*c_escondida.out[i];
		}
		c_salida.out[k]=sigmoide(c_salida.out[k]);
		if(c_salida.out[k]<0.1) c_salida.out[k]=0.1; //limita una salida mínima de 0.1
	}
}
```

3.3.3. Adquisición de imágenes

En primer lugar es necesario crear objetos de clase *Mat* que, básicamente, corresponden a matrices bidimensionales (de uno, dos, tres o cuatro canales) donde se almacenan imágenes, en *frame* se almacenará la secuencia de imágenes adquiridas a través de la cámara, las imágenes adquiridas son a color de tipo RGB (Red Green Blue), por lo que *frame* se define automáticamente de tres canales.

La adquisición de imágenes se realiza con la clase *VideoCapture* creando un objeto *cap* y especificando la cámara (asignada como dispositivo de video 0), tanto la adquisición como el procesamiento de imágenes corresponden a funciones de las librerías de OpenCV, de modo que se incluyen los encabezados de los módulos correspondientes descritos en la sección 2.4.2, específicamente *#include <opencv2/highgui/highgui.hpp>* está relacionado a la captura de imágenes o video; como un requerimiento de C++ se definen los espacios de nombre mediante using namespace std y *using namesoace cv* que organizan el código en grupos lógicos para evitar conflictos de identificación de funciones ya que se usan varias librerías de OpenCV conjuntamente con otras de C++.

La creación de *cap* se realiza dentro de la función principal *main* (resaltada en verde para identificarla en la explicación de las demás secciones del código), allí también se configura la captura a una dimensión relativamente pequeña de 320x240 píxeles para evitar el incremento de procesamiento de la CPU, así como la apertura de la cámara a través de la función *cap.isOpened()*; la captura continua se realiza y se asigna a *frame* a través de la línea de código cap >> frame dentro de un *while(true)* infinito que contiene todas las demás rutinas del algoritmo (resaltado en verde para su identificación en otras secciones); luego éste *while* debe finalizarse la captura mediante *cap.release()* en caso de generarse un *return 0* en el programa. El código de ésta sección se identifica de la siguiente manera:

```
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
using namespace std;
using namespace cv;
Mat frame;// se utiliza para almacenar las imágenes capturadas
Mat im_ROI; //se utiliza para almacenar la sección de interés
Mat img_gray; //se utiliza en las rutinas de procesamiento de imagen
int main(){
        VideoCapture cap(0);
        cap.set(CV_CAP_PROP_FRAME_WIDTH,320);
        cap.set(CV_CAP_PROP_FRAME_HEIGHT,240);
        if(!cap.isOpened()) return -1;
        while(true){
                cap >> frame;
                //CODIGO DE OTRAS RUTINAS: PROCESAMIENTO DE IMÁGENES, OPERACIONES DE RNA, CÁLCULO DE VARIABLES, ETC.
        }
}
cap.release();
```

3.3.4. Procesamiento de imágenes

Para las rutinas de procesamiento de imágenes será necesario el manejo de colores de píxeles de tal modo que se declaran constantes del tipo *cvScalar* donde se especifican los niveles de cada color RGB siendo blanco todos los parámetros en el valor más alto (255) y negro en el más bajo (0); las funciones que se utilizarán como filtros y detectores de bordes están contenidas en la declaración de los encabezados *#include <opencv2/imgproc/imgproc.hpp>*; el código de procesamiento de imágenes se encuentra dentro del *while*(true) infinito luego de *cap >> frame* como se indica en la siguiente estructura de código:

```
#define WHITE   cvScalar(255, 255, 255, 0)
#define BLACK   cvScalar(0,0,0,0)
//RED: cvScalar(0,0,255, 0) //GREEN: cvScalar(0,255,0,0) //BLUE: cvScalar(255,0,0,0)
int main(){
        //CODIGO DE OTRAS RUTINAS INCLUIDA ADQUISICIÓN DE IMÁGENES
        while(true){
                cap >> frame;
//CODIGO DE PROCESAMIENTO DE IMÁGENES CORRESPONDIENTE A CADA INSTRUMENTO
    }
}
```

Para el desarrollo del algoritmo, en este trabajo, se ha considerado clasificar a los instrumentos analógicos en dos tipos según su forma geométrica: circular y cilíndrica, que son formas comunes en las cuales se presentan los instrumentos donde la lectura se estima en base a la posición del indicador sobre una escala. El desarrollo se describe para un manómetro (forma circular) y un rotámetro (forma cilíndrica); algunas rutinas se diferencian para cada caso (incluido la etapa de procesamiento de imágenes), sin embargo, la estructura del algoritmo presentada en la Figura 47 es la misma.

a. Manómetro

Algunos instrumentos, como el manómetro, a menudo se encuentran ubicados a la intemperie, por lo que éste está expuesto a condiciones de luminosidad variable, lo cual resulta un problema desde la perspectiva de la etapa de procesamiento de imágenes, pues es muy probable que se generen sombras o no sea posible distinguir el objeto a causa de la ausencia de luz; otro factor a considerar es que la escala la recubre una capa de vidrio, ésta superficie implica la dificultad de procesar una imagen sin ruido porque es posible que se generaren reflejos de forma aleatoria.

Estos inconvenientes se evitan utilizando un sistema de iluminación frontal basado en tecnología LED, de forma que el consumo energético es mínimo, y el color corresponde al rojo; éste color comúnmente es utilizado en aplicaciones de visión artificial, experimentalmente se comprobó que no altera la sensibilidad de la cámara al no perderse formas de interés dentro de la imagen (Advanced ilumination, 2016; National Instruments, 2016).

Utilizando el sistema de iluminación frontal no es necesario implementar algoritmos complejos de eliminación de sombras y reconstrucción de formas de imagen que incrementan el procesamiento de la CPU; no obstante, se generan reflejos brillantes por la luz emitida por cada LED sobre el vidrio, pero, es posible eliminarlos mediante un algoritmo que no representa un elevado costo computacional. Este algoritmo se orienta a identificar los píxeles de valores altos,

que tienen al valor del color blanco (255) y corresponden a los reflejos de luz, sobre una imagen que ha sido convertida de RGB a escala de grises. Se determina un parámetro μ que representa el valor en escala de grises, a partir del cual se considera a un píxel como parte de un reflejo de luz en la imagen;

$$\mu = a + t \tag{13}$$

donde,

$$a = \frac{\sum_{i=0}^{n} v_i}{n} \tag{14}$$

v_i es el valor de un píxel i, comprendido de 0 a 255, de n píxeles definidos por una ROI (en este caso de 200x200 píxeles) que contiene la escala circular del instrumento, y t es un valor de umbral que se fija de forma experimental en un rango de 1 a 20. Se compara cada píxel de la ROI con el valor de μ, y se reemplaza con el valor de a (promedio de los valores de todos los píxeles en la ROI) a los píxeles cuyo valor sea mayor o igual a μ, de ésta manera se obtiene una imagen en escala de grises donde los reflejos son atenuados; éste procedimiento se complementa con la aplicación de un filtro Gaussiano, como se indican en la Figura 50.

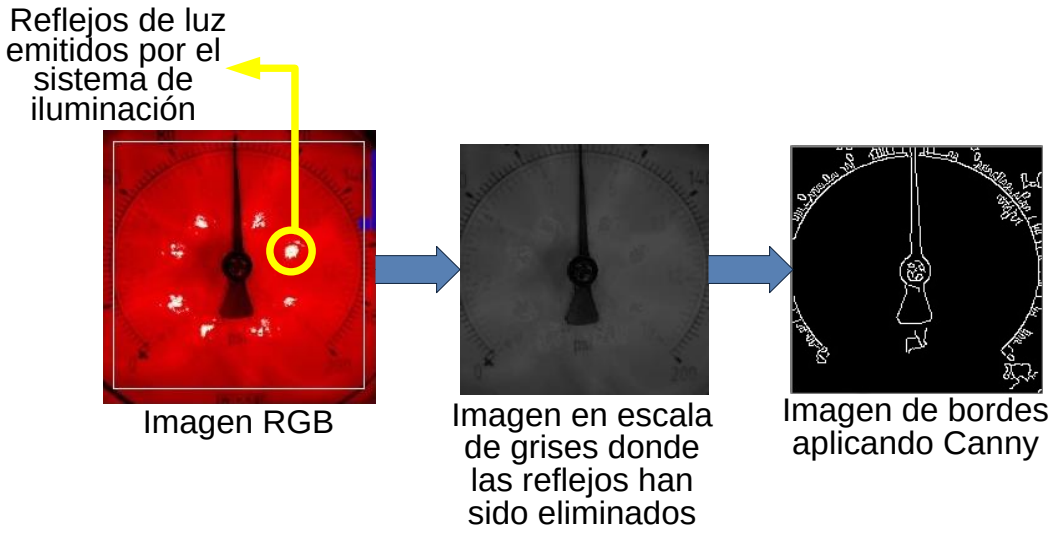


Figura 50: Etapas de procesamiento de imagen del manómetro.

El código que se presenta a continuación corresponde a las etapas de procesamiento que se distinguen en la Figura 50; mediante la función *Rect* se extrae una ROI de 200x200 píxeles de *frame*, con la función *cvtColor* se transforma la imagen RGB a escala de grises; para el algoritmo de eliminación de reflejos es necesario declarar un nuevo objeto Mat *im_newgray* e inicializarlo como una matriz de píxeles de tipo entero de 8 bits y de un solo canal, donde se almacenarán los píxeles en escala de grises modificados durante el proceso de comparación de valores de cada píxel con el parámetro *umbralreflejo,* que se ejecuta luego de calcularse el promedio del valor de los píxeles de la ROI. *GaussianBlur* es una función de un filtro gaussiano de un kernel de 7x7 y desviaciones estándar 1.2; al final se aplica el algoritmo de Canny para obtener los bordes de la imagen (imagen binaria de píxeles blancos y negros) con umbrales bajo y alto de 20 y 40 respectivamente.

```
Mat im_newgray=Mat(200,200,CV_8UC1,cvScalar(125));
int main(){
        //CODIGO DE OTRAS RUTINAS INCLUIDA ADQUISICIÓN DE IMÁGENES
    while(true){
        cap >> frame;
        im_ROI=frame(Rect(x_roi,y_roi,roi_size,roi_size)); //extra la ROI
        cvtColor(im_ROI, img_gray, COLOR_BGR2GRAY);//RGB a escala de grises
        promedio=0;
        for(py=0; py<200; py++){
                for(px=0; px<200; px++){
                        pixgray=img_gray.at<uchar>(px,py);
                        promedio=promedio + pixgray.val[0];
                }
        }
        promedio=promedio/40000;
        a=promedio;
        for(int py=0; py<200; py++){
                for(int px=0; px<200; px++){
                        pixgray=img_gray.at<uchar>(px,py);
                        umbralreflejo=a+20;
                        if(umbralreflejo>255)umbralreflejo=255;
                        if(pixgray.val[0]>umbralreflejo){
                                im_newgray.at<uchar>(px,py)=nvp;
                        }else{im_newgray.at<uchar>(px,py)=pixgray.val[0];}
                }
        }
        GaussianBlur(im_newgray, im_ROI, Size(7,7), 1.2, 1.2);//filtro gaussiano
        Canny(im_ROI, im_ROI, 20, 40, 3);//detector de bordes Canny
        // CONTINUA EL CODIGO DE OTRAS RUTINAS
    }
}
```

b. Rotámetro

El rotámetro es transparente en toda la forma cilíndrica donde se ubica la escala, de modo que está expuesto a la luz ambiental existente a su alrededor; no obstante se encuentran instalados en laboratorios y plantas donde las condiciones de iluminación del ambiente no varían de forma significativa, que generalmente corresponden a procesos químicos; en base a esto se ha considerado no utilizar un sistema de iluminación.

Las etapas de procesamiento de la imagen del rotámetro se distinguen en la Figura 51, de la imagen RGB adquirida por la cámara se extrae una ROI de 200x70 píxeles para convertirla a una imagen de escala de grises; a diferencia del manómetro no es necesario realizar la etapa que elimina los reflejos. Se realiza un etapa de filtrado, que tiene como propósito suavizar los bordes de la imagen, mediante la función de filtro Gaussiano *GaussianBlur* con desviaciones estándar de 1.15 y kernel de 7x7; los bordes de la imagen se obtienen haciendo uso de la función *Canny* con un umbrales 40 y 80.

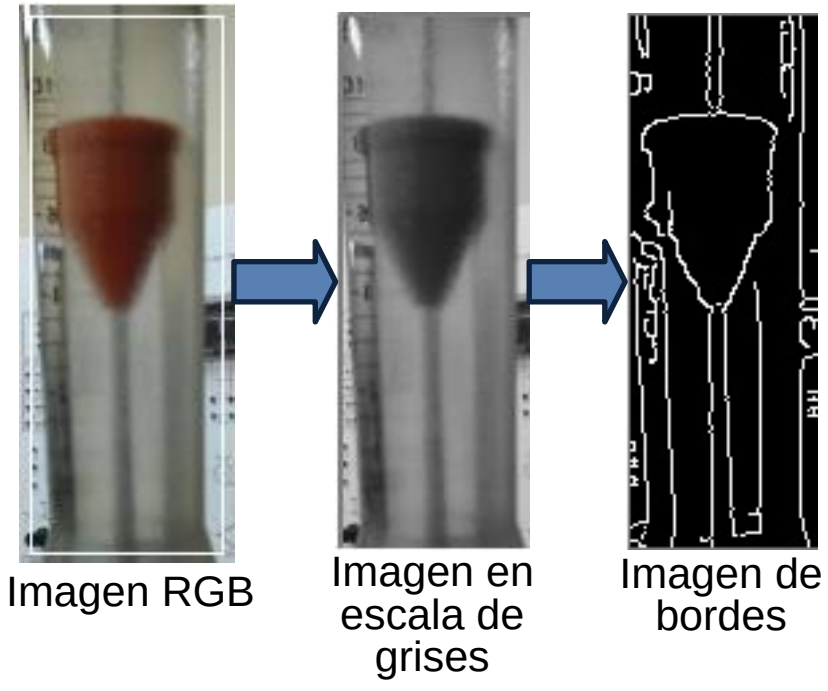


Figura 51: Etapas de procesamiento de imagen del rotámetro.

La rutina para procesar la imagen del rotámetro está implementado luego de la adquisición de la imagen mediante *cap >> frame* dentro del *while(true)* infinito, como se indica en las siguientes líneas de código:

```
int main(){
        //CODIGO DE OTRAS RUTINAS INCLUIDA ADQUISICIÓN DE IMÁGENES
    while(true){
        cap >> frame;
        im_ROI=frame(Rect(x_roi,y_roi,c_columnas,c_filas));//extracción de la ROI
        cvtColor(im_ROI, img_gray, COLOR_BGR2GRAY);//RGB a escala de grises
        GaussianBlur(img_gray, filtro, Size(7,7), 1.5, 1.5);//filtro gaussiano
        Canny(filtro, im_ROI, 20, 40, 3);//detección de bordes
        // CONTINUA EL CODIGO DE OTRAS RUTINAS
    }
}
```

3.3.5. Rutina de centrado de la escala del instrumento

Para que la lectura de la medida se realice correctamente es necesario posicionar la ROI que extrae la región de la escala del instrumento en el centro, ya que posteriormente se harán cálculos de la posición del indicador en la imagen para determinar la magnitud de la variable; en otras palabras mientras la escala no esté adecuadamente posicionada no será posible seguir con la ejecución de las rutinas asociadas a la otra *RNA de reconocimiento del indicador*.

El significado de la activación de las neuronas de la capa de salida de la *RNA de reconocimiento de la escala* se distingue en esta etapa, pues en el vector *n_c_salida.out[pg]* se determinan cuáles son las neuronas ganadoras, en cuanto a la ponderación de activación; luego se ejecuta un desplazamiento de la ROI en la dirección correspondiente, un píxel a la vez, incrementando o disminuyendo los valores de las variables de posición *x_roi* y *y_*roi, contenidas en la función *im_ROI=frame(Rect(x_roi,y_roi,c_columnas,c_filas))* que se explicó en la anterior sección.

En el código se inicia detallando la función *void n_get_ROI(Mat n_insROI)* que se encarga de asignar el valor de los píxeles blancos y negros de la ROI como entradas de la RNA (0.9 y 0.1 respectivamente) en el vector *n_Entrada*; luego de

la ejecución de esta función se ejecutan las funciones *n_calcular_capa_escondida() n_calcular_capa_salida()* propias de la RNA.

Es importante mencionar que ésta rutina de centrado se ejecuta siempre que la bandera *bandera_ajuste* este activa; cuando se ha logrado el propósito de centrar la escala, es decir, cuando las neuronas 1 y 4 de la capa de salida sean las de mayor ponderación de activación (en el código *winx=1* y *winy=4*), ésta bandera se desactiva o se fija en cero conjuntamente con otras variables utilizadas como contadores, éstos últimos intervienen en las rutinas emergentes descritas en la sección 3.3.7. Si la neurona 6 (winy=6) es la de mayor ponderación *x_roi* y *y_*roi se asignan con valores aleatorios para una búsqueda rápida de la escala.

```
void n_get_ROI(Mat n_insROI){
        float temp;
        IplImage* n_inROI=new IplImage(n_insROI);
        CvScalar n_scal;
        int i,j,counter;
        counter=0;
        for(i=0;i<c_filas;i++){
                for(j=0;j<c_columnas;j++){

                        n_scal = cvGet2D(n_inROI,i,j);//función que obtiene el valor del píxel
                        if (n_scal.val[0])temp=0.9;
                        else     temp=0.1;
                        n_Entrada[counter]=temp;
                        counter++;
                }
        }
}
int main(){
        //CODIGO DE OTRAS RUTINAS INCLUIDA ADQUISICIÓN DE IMÁGENES
    while(true){
        //CODIGO DE ADQUISICION Y PROCESAMIENTO DE IMÁGENES
        if (bandera_ajuste==1){
                n_get_ROI(im_ROI);
                n_Gain=0.5; //se define la constante de ganancia de la función sigmoide
                n_calcular_capa_escondida();
                n_Gain=1.5; //se define la constante de ganancia de la función sigmoide
                n_calcular_capa_salida();
                ax=0; ay=0;
                winx=0; winy=0;
                for(int pg=0; pg<7; pg++){
                        if(pg<=2){
                                if(n_c_salida.out[pg]>ax){
                                        ax=n_c_salida.out[pg];
                                        winx=pg;
```

```
                    }
                }else{
                    if(n_c_salida.out[pg]>ay){
                        ay=n_c_salida.out[pg];
                        winy=pg;
                    }
                }
            }
            if(winx==0) x_roi++;
            if(winx==2) x_roi--;
            if(x_roi>120)x_roi=60; if(x_roi<0)x_roi=60;//limita el movimiento de la ROI del píxel 0 al 120 y retorna a 60 en el eje x
            if(winy==3) y_roi++;
            if(winy==5) y_roi--;
            if(winy==6) {x_roi=rand()%121;y_roi=rand()%41;}//define posiciones aleatorias
            if(y_roi>40)y_roi=20; if(y_roi<0)y_roi=20; //limita el movimiento de la ROI del píxel 0 al 40 y retorna a 20 en el eje y
            contador_ajuste++;
            if(winx==1 && winy==4){
                bandera_ajuste=0;
                cont_loss_i=0;
                cont_loss_d=0;
                contador_ajuste=0;
            }
        }
        // CONTINUA EL CODIGO DE OTRAS RUTINAS
    }
}
```

3.3.6. Rutinas que determinan el valor de la variable

Estas rutinas se realizan una vez posicionada la escala del instrumento en la ROI, por lo que se estructuran dentro de un caso contrario (*else*) a la activación de la bandera *bandera_ajuste*; esta etapa está basada en el seguimiento dinámico del indicador del instrumento a través del movimiento de una nueva ROI extraída de la imagen procesada de la escala mediante la función *get_ROI(im_ROI)*, que a la vez alimenta de valores de entrada a la RNA correspondiente al reconocimiento del indicador (como se indicó en las Figuras 48 y 49).

Se ha creado las funciones *seguir_indicador()* y *calcular_variable()* que varían su contenido según la forma geométrica del instrumento al igual que la función *get_ROI(im_ROI)*, como se presenta en las secciones 3.3.6.1 y 3.3.6.2. Los códigos de estas rutinas se estructuran de la siguiente manera en el *main*:

```
int main(){
        //CODIGO DE OTRAS RUTINAS INCLUIDA ADQUISICIÓN DE IMÁGENES
    while(true){
        //CODIGO DE ADQUISICION Y PROCESAMIENTO DE IMÁGENES
        if (bandera_ajuste==1){
                //RUTINA DE CENTRADO DE LA ESCALA DEL INSTRUMENTO
        }else{
                get_ROI(im_ROI);
                Gain=0.5;//se define la constante de ganancia de la función sigmoide
                calcular_capa_escondida();
                Gain=1.5;//se define la constante de ganancia de la función sigmoide
                calcular_capa_salida();
                get_winner(); //función para determinar la neurona de mayor valor
                seguir_indicador();
                calcular_variable();
                 //RUTINA PARA ENVIAR EL DATO DE LA VARIABLE POR WIFI
                master_letter = cvWaitKey(1);//espera 1 ms antes de una nueva ejecución
        }
    }
}
```

a. Manómetro

La forma circular del manómetro involucra que el recuadro que delimita la extracción de una nueva ROI, que contiene una porción del indicador, rote alrededor de un eje como se indica en la Figura 52; se utiliza esta técnica porque ésta porción de imagen, entregada a la RNA para su reconocimiento, corresponde únicamente a 1000 datos de entrada (50x20 píxeles) que son procesados rápidamente en los cálculos de la RNA; de ésta manera se hace posible que el seguimiento del indicador se ejecute en tiempo real.

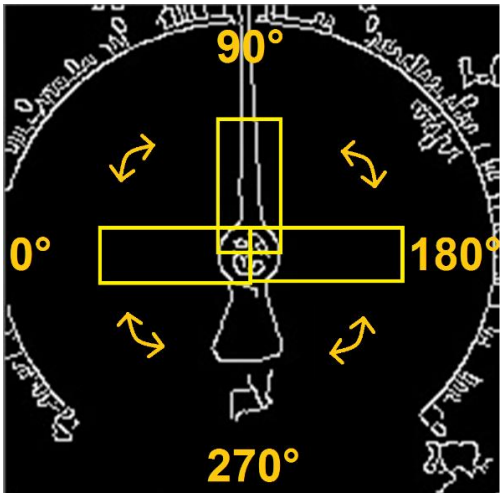


Figura 52: Movimiento de rotación de la ROI que contiene el indicador.

El movimiento de la ROI está basada en la aplicación de una transformación lineal de rotación – descrita y demostrada en (Kolman & Hill, 2006) - definida por,

$$\begin{bmatrix} x_{nm_k} \\ y_{nm_k} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x_{m_k} \\ y_{m_k} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \quad (15)$$

la cual se encarga de mapear uno a uno los píxeles contenidos en la ROI (como se indica en la Figura 53), es decir, se determina las coordenadas x_{nm} y y_{nm} de cada píxel *k* en función de un ángulo ϕ conocido y la coordenada de una posición anterior x_m y y_m en torno a un origen situado en el centro de la imagen (coordenadas $c_x = 100$ y $c_y = 100$). Se establece las coordenas anteriores x_m y y_m como constantes para un $\theta = 0°$, y así, es posible definir el movimiento de la ROI únicamente a través de la variación del ángulo ϕ. Lo descrito se encuentra codificado en la función *void get_ROI(Mat insROI)*.

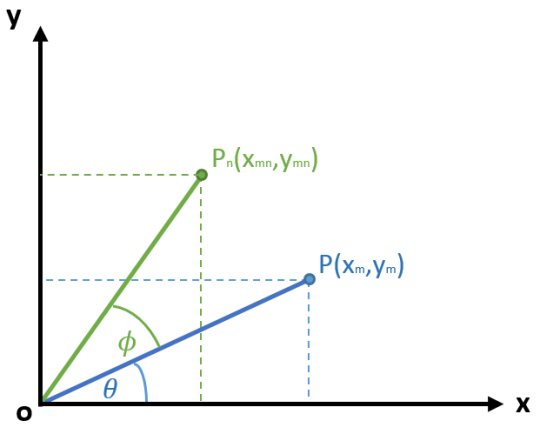


Figura 53: Mapeo de un píxel en un movimiento de rotación.

```
void get_ROI(Mat insROI){
        float temp;
        IplImage *indROI=new IplImage(insROI);
        CvScalar scal;
        int sc, sf, contador, xi, yi, cx, cy, i, j;
        float ang_fi=(fi*3.1416)/180; //se transforma de grados a radianes
        contador=0;
        xi=-50;
        yi=10;
        cx=100;
        cy=100;
```

```
        for(i=0; i<50; i++){
                for(j=0; j<20; j++){
                        sc=((xi+i)*cos(ang_fi)-(yi-j)*sin(ang_fi))+cx;//sc: columnas - eje x
                        sf=((xi+i)*sin(ang_fi)+(yi-j)*cos(ang_fi))+cy;//sf: filas - eje y
                        scal=cvGet2D(indROI, sf, sc);//función que obtiene el valor del píxel
                        if(scal.val[0]) temp=0.9;
                        else temp=0.1;
                        Entrada[contador]=temp;
                        contador++;
                }
        }
}
```

El seguimiento del indicador se produce mediante el incremento o disminución del valor del ángulo declarado por una variable entera *fi*, éste valor depende de la activación de cada neurona de la RNA que reconoce si el indicador se encuentra situado en el centro de la ROI o si está desviado unos cuantos grados a la izquierda o derecha (como se explicó en la sección 3.3.2).

Como se indica en el código presentado a continuación, cuando la ponderación de la neurona 2 es la ganadora (*winner_x=2*) el ángulo *fi* se mantiene, caso contrario se efectúan las variaciones de mínimo 1 grado para generar exactitud a la búsqueda, 5 grados acelerar la búsqueda y 20 grados para una búsqueda mucho más rápida cuando no se encuentra ninguna porción del indicado dentro de la ROI.

```
void seguir_indicador(void){
        if(winner_x==2){
                fi=fi;
        }else{
                if(winner_x==0)fi=fi-5;
                if(winner_x==1)fi=fi-1;
                if(winner_x==3)fi=fi+1;
                if(winner_x==4)fi=fi+5;
                if(winner_x==5)fi=fi+20;
      }
      if(fi>360) fi=fi-360; //limita el valor de fi a un rango de 0 a 360
      if(fi<0) fi=fi+360; //limita el valor de fi a un rango de 0 a 360
}
```

Como ya se ha mencionado la magnitud de la variable que se lee en el instrumento se determina conociendo la ubicación del indicador en la escala, en el caso del manómetro se cuantifica el ángulo de rotación *fi* y se realiza un

escalamiento en función de éste parámetro y el rango de la escala (en este caso corresponde a una escala lineal de 0 a 200 PSI, éstos límites están ubicados en la imagen en 315° y 225° respectivamente) resultando las ecuaciones que se indican en el siguiente código:

```
void calcular_variable(void){
        if(fi>0 && fi<=225) psi=(0.74)*fi+(33.33);
        if(fi>=315 && fi<=360) psi=(0.74)*fi-(233.33);
}
```

b. Rotámetro

En el rotámetro únicamente se emplea un movimiento vertical de una ROI de 30x20 píxeles, que corresponde a 600 datos de entrada de la RNA; éste movimiento está definido por la expresión

$$y_{nr_k} = y_{r_k} + \Delta y \tag{16}$$

donde y_{nr} es la nueva coordenada en el eje de las ordenadas de cada píxel *k* contenido en la ROI, y_r es la coordenada anterior y Δy corresponde a un escalar positivo o negativo ajustable, que genera una variación ascendente o descendente sobre un espacio definido. Δy toma valores de ±1, ±5 según la activación de las neuronas 0, 1, 3 y 4 de la capa de salida de la RNA como se indica en la función *void seguir_indicador(void)*; si la RNA reconoce que el borde del flotador está ubicado en el centro de la ROI no se ejecuta un movimiento, esto es cuando la neurona 2 está activa (*winner=2*).

```
void seguir_indicador(void){
        if(winner==2){
                yr=yr;
        }else{
                if(winner==0)yr=yr-5;
                if(winner==1)yr=yr-1;
                if(winner==3)yr=yr+1;
                if(winner==4)yr=yr+5;
                if(winner==5)yr=yr-10;
        }
}
```

En la función *void get_ROI(Mat insROI)* se captura la ROI que contienen al borde del flotador del rotámetro, píxel a píxel, y se alimenta a la RNA con dicha

porción de imagen; las coordenadas de cada píxel en el eje x permanecen constantes, en cambio en el eje y las coordenadas se establecen en función del valor de la variable de tipo entero *yr* que se ha establecido en la función *void seguir_indicador(void)*.

```
void get_ROI(Mat insROI){
        float temp;
        IplImage *indROI=new IplImage(insROI);
        CvScalar scal;
        int  contador, x, ynr;
        contador=0;
        for(x=21; x<51; x++){
                for(ynr=yr-10; ynr<yr+10; ynr++){
                        scal=cvGet2D(indROI, ynr, x);// ynr: filas – eje y    //x: columnas – eje x
                        if(scal.val[0]) temp=0.9;
                        else temp=0.1;
                        Entrada[contador]=temp;
                        contador++;
                }
        }
}
```

La escala graduada del rotámetro de 1 a 10 GPM es logarítmica, por lo que se utiliza una aproximación polinomial que está en función de la posición del borde del flotador (medida en el recorrido de píxeles en la imagen) y el valor real de flujo que se puede apreciar de forma visual en la escala, como se indica en la Figura 54; se obtiene una ecuación cuadrática definida por:

$$F[LPM] = 0.00049\, yr^2 + 0.14\, yr + 1.97 \qquad (17)$$

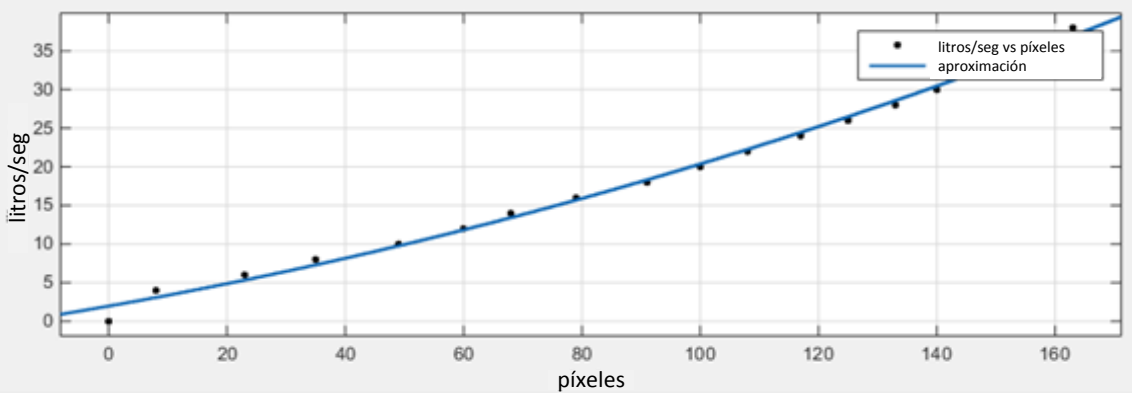


Figura 54: Etapas de procesamiento de imagen del rotámetro.

La ecuación 11 se codifica dentro de la función *void calcular_variable(void)* utilizando el valor de la variable *yr* ya definida anteriormente. Teniendo en cuenta que el sistema que se propone en este trabajo está diseñado para operar con cámaras situadas en una posición fija, esta ecuación es válida para un rango de coordenadas en el eje vertical de 10 a 170 que; éste intervalo corresponde a un de recorrido de píxeles de 0 a 160 (como se indica en el eje x de la Figura 54), y además, a la región en la imagen donde puede situarse el borde del flotador.

```
void calcular_variable(void){
        float x, x2;
        if(yr>=10 && yr<=170){
                x=(yr-170)*(-1);// corresponde a un pre escalamiento para obtener un valor de recorrido que empieza desde 0 y termina en 160
                x2=x*x;
                lpm=0.00049*x2+0.14*x+1.97;
                gpm=0.26*lpm; //se transforma un valor de LPM a GPM

        }
}
```

3.4. Rutinas de la transmisión de datos de forma inalámbrica

El sistema embebido se encuentra en una red WLAN de modo que se incluyen rutinas de comunicación como se indica en el diagrama de flujo de la Figura 47. En en el literal 3.1.3 se justifica el empleo del protocolo UDP (protocolo de la capa de transporte) para lograr una transmisión de datos en tiempo real, para ello es necesario el manejo de sockets de comunicación de tipo UDP que permiten la transmisión bidireccional de datos; un socket corresponde al conjunto formado por una dirección IP y el puerto del proceso origen, más la IP y el puerto de proceso destino; los puertos (asignados con valores de 16 bits) son identificadores de buffers o canales de comunicación en las máquinas de origen y destino.

Mediante el uso de sockets una máquina puede tener varios procesos independientes de transmisión o recepción de paquetes de datos debido a la posibilidad de crear distintos buffers, de este modo les es posible a las tarjetas embebidas (BeagleBone Black y Raspberry Pi3) transmitir información al sistema servidor simultáneamente y en tiempo real.

Es importante identificar que las tarjetas embebidas serán clientes que solicitarán establecer una conexión con el sistema servidor, el cual está creado en LabVIEW y será descrito en la sección 3.6; al utilizar sockets el puerto del cliente es asignado por el sistema operativo de forma automática, mientras que en el servidor se define un puerto para que sea conocido por el cliente.

Se ha definido la dirección IP 192.168.0.102 para el sistema servidor, y en este caso el puerto es 61557 (en el programa de otra tarjeta embebida se asigna un puerto diferente); en el código presentado a continuación se implementa la creación de un socket de tipo UDP, así como se declaran las variables y estructuras necesarias para establecer la comunicación inalámbrica con el servidor; al inicio se indican las librerías necesarias para el manejo de éstos sockets, también se incluye #include <string.h> para el manejo de datos de envío y recepción de tipo string.

```
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <string.h>
#define SERVIDOR "192.168.0.102" //dirección IP del servidor
#define PORT     61557 //puerto del servidor
#define BUFFER   10 //tamaño máximo de los datos a enviar y recibir
int main(){
        char dato_flujo[BUFFER];//vector de caracteres que almacena el dato a enviar
        char mensaje[BUFFER]; // vector de caracteres que almacenará el dato recibido
        int bbb1_s;//variable donde será asignado el socket del dispositivo
        struct sockaddr_in bbb1; //estructura de un socket que corresponde al servidor
        memset((char *) &bbb1, 0, sizeof(bbb1)); //limpia los parámetros del socket bb1
        bbb1.sin_family=AF_INET; //define el uso del protocolo IPv4
        bbb1.sin_port=htons(PORT);//especifica el puerto del servidor
        inet_aton(SERVIDOR, &bbb1.sin_addr); //se especifica la dirección IP del servidor
        socklen_t slen=sizeof(bbb1); //se asigna a slen el tamaño del socket bbb1

        bbb1_s=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP); //crea el socket UDP del
dispositivo local
        while(true){
                //CODIGO DE OTRAS RUTINAS
        }
}
```

3.4.1. Sincronización de comunicación con el sistema servidor

El dispositivo embebido requiere informarle al sistema servidor que está listo para enviar datos, por lo que es necesaria una rutina inicial de sincronización o inicio de sesión, principalmente, para que el servidor conozca el puerto de éste cliente (asignado por el sistema operativo); y además, como un método de confirmación del dispositivo de dónde proviene la información, lo cual aporta seguridad en la comunicación; ésta sincronización se realiza a través del intercambio de caracteres previamente definidos.

En el código que se presenta a continuación (perteneciente al algoritmo para el rotámetro) se envía continuamente la letra *f* con la función *sendto* hasta que se recibe, con la función *recvfrom*, una letra *a* proveniente del sistema servidor. Éste proceso se realiza antes de iniciar la ejecución de las rutinas que determinan el valor de la variable dentro del *while(true)* infinito. También es importante mencionar que si el programa genera un *return 0* se cierra el socket *bbb1_s* al final del código.

```
int main(){
//CREACION DEL SOCKET
memset(dato_flujo,'\0',BUFFER);
        dato_flujo[0]='f';
        while(mensaje[0]!='a'){
                sendto(bbb1_s, dato_flujo, strlen(dato_flujo),0,(struct sockaddr *) &bbb1,slen);
                memset(mensaje, '\0',BUFFER); //limpia el vector mensaje
                recvfrom(bbb1_s,mensaje,BUFFER,0,(struct sockaddr *) &bbb1, &slen);
        }
while(true){
        //CODIGO DE OTRAS RUTINAS
}
close(bbb1_s); //cierra el socket bbb1_s
}
```

3.4.2. Envío del dato correspondiente al valor de la variable

Una vez que se haya determinado el valor de la variable con las técnicas de visión artificial, se envía inalámbricamente este dato mediante la función *sendto*, pero antes es necesario convertir la variable de tipo flotante (*lpm*) a una variable de tipo string mediante la función *sprintf*, éste nuevo dato se almacena en la

variable *dato_flujo* que fue declarada cuando se creó el socket. Debido a que se utiliza el protocolo UDP la función *sendto* se ejecuta y el algoritmo continúa con un nuevo barrido del programa para determinar un nuevo valor de la variable, es decir, no es necesaria una confirmación de que el dato ha llegado por parte del sistema servidor. El código es el siguiente y se situa luego de la función *calcular_variable()*.

```
int main(){
//CREACION DEL SOCKET
//RUTINA DE SINCRONIZACION
while(true){
//DEMAS RUTINAS DEL ALGORITMO
        calcular_variable();
        if(winner_x!=5){
            sprintf(dato_flujo,"%f",lpm);
            sendto(bbb1_s, dato_flujo, strlen(dato_flujo), 0, (struct sockaddr *) &bbb1, slen);
        }
}
}
```

3.4.3. Envío de mensajes de rutinas emergente

En el algoritmo existen rutinas que han sido denominadas como emergentes (se pueden encontrar en las versiones completas de códigos de los Anexos A y B) y básicamente consisten en el conteo de las veces sucesivas que se pierde el indicador durante el proceso de seguimiento o se posiciona fuera del rango que corresponde a la escala; mediante la lectura del valor de éstos contadores se establece si es necesario ejecutar un nuevo ajuste o centrado de la posición de la escala para que la lectura de la variable de proceso no sea falsa.

Si ha sido necesario ejecutar la rutina de centrado de la escala, se debe enviar un mensaje al sistema servidor que se está realizando la acción; además se produce un conteo de las veces que ésta rutina se ejecuta, de modo que si no se puede centrar adecuadamente la escala o la RNA correspondiente no ha podido reconocer ninguna porción de la imagen de la escala durante un cierto número de ejecuciones se debe enviar un mensaje de *error*; éste mensaje tiene como propósito alertar al usuario del sistema de monitoreo que debe acercarse a verificar el estado o la posición de la cámara.

El código correspondiente al envío del mensaje de '*error*' y '*centro*' que está asociado a la lectura del valor de un contador *contador_ajuste* se presenta a continuación, éste se ubica al último dentro de la sección que ejecuta el centrado de la escala - la referencia es if *(bandera_ajuste==1){ }* - dentro del *while(true)* infinito. Es importante mencionar el vector dato_flujo se limpia con la función *memset* antes de que se le asigne los caracteres del mensaje, esto para controlar que no se asignen datos basura en los elementos sobrantes, y de esta forma enviar strings cortos.

```
int main(){
while(true){
//DEMAS RUTINAS DEL ALGORITMO
  if (bandera_ajuste==1){
    //CODIGO DE LA RUTINA DE CENTRADO DE LA ESCALA
    if(contador_ajuste>5){
            if (contador_ajuste>50 && winy==6){
            //--------lazar mensaje de error porque no se encuentra la escala----------
               memset(dato_flujo,'\0',BUFFER);
               dato_flujo[0]='e';dato_flujo[1]='r';dato_flujo[2]='r';dato_flujo[3]='o';dato_flujo[4]='r';
               sendto(bbb1_s, dato_flujo, strlen(dato_flujo),0,(struct sockaddr *) &bbb1,slen);
            }else{
                //----lanzar mensaje de que se está centrando la escala del instrumento--------
                    memset(dato_flujo,'\0',BUFFER);
                    dato_flujo[0]='c';dato_flujo[1]='e';dato_flujo[2]='n';dato_flujo[3]='t';dato_flujo[4]='r';
                    dato_flujo[5]='o';
                   sendto(bbb1_s, dato_flujo, strlen(dato_flujo),0,(struct sockaddr *) &bbb1,slen);
            }
    }
  }
//DEMAS RUTINAS DEL ALGORITMO
}
}
```

3.5. Entrenamiento de las Redes Neuronales Artificiales

El entrenamiento de las RNA se realiza en base al algoritmo de Backpropagation (sección 2.3.3.2); es necesario implementar un programa (ejecutado en una PC por mayor rapidez) que provea a las RNA una secuencia de ejemplos para ajustar los pesos sinápticos de cada neurona, y de esta forma lograr que las salidas se comporten de forma deseada; el programa se encuentra descrito en la Figura 55, la idea base se aplica a los dos tipos de instrumentos analógicos; los códigos completos está en los Anexos C y D.

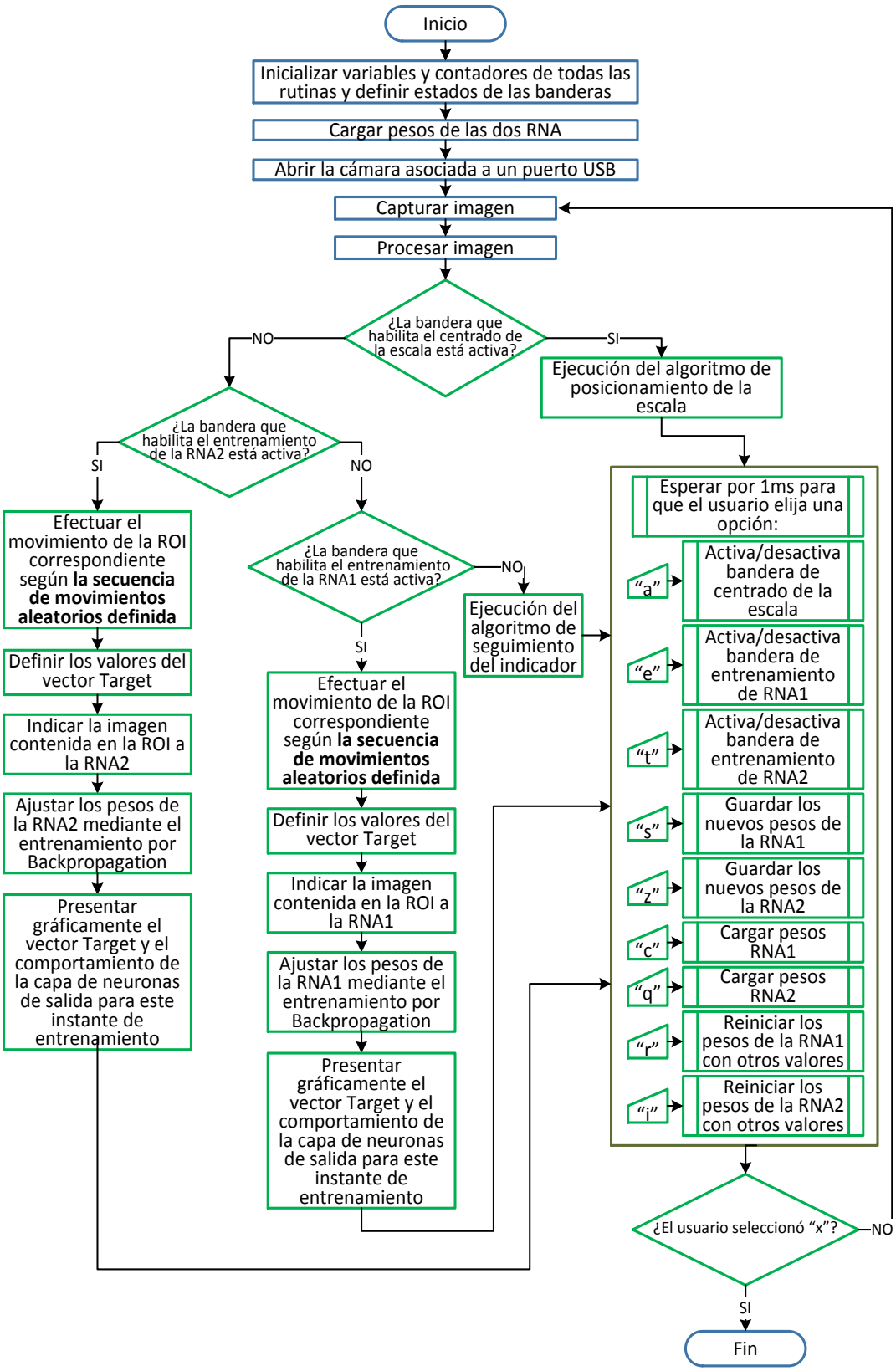


Figura 55: Diagrama de flujo del algoritmo de entrenamiento de las RNA.

Se explicará secciones importantes del código; se inicia con la parte principal que corresponde al ajuste de pesos sinápticos; para éste procedimiento se puede iniciar con valores aleatorios de pesos sinápticos, de modo que al forzar una imagen como ejemplo, primero se evalúa normalmente la respuesta de las neuronas de la capa de salida de la RNA (con las funciones descritas en la sección 3.3.2), luego se comprueba el error entre la salida deseada (target) y la salida que se está produciendo, pero utilizando una gradiente o variación del error para que la corrección de pesos sea paulatina.

A continuación ésta variación de error se propaga hacia las neuronas de la capa escondida con el fin de definir los errores en esa neuronas; para ello se calcula una gradiente de error de la salida de cada neurona escondida, aquí interviene la sumatoria de la multiplicación del error presentado por cada neurona de la capa de salida (obtenido en la etapa anterior) por el valor de cada peso sináptico que conecta a la neurona anterior (de la capa escondida).

Una vez calculados los errores de las neuronas de las dos capas se procede a corregirlos iniciando por la capa de salida; cada peso se corrige sumándole una compensación dada por: la multiplicación de una constante de aprendizaje *n_Eta* por el valor de la variación de error de la neurona actual y por la entrada que conecta a ese peso. En el código que se presenta a continuación se estructura lo mencionado.

```
void n_corregir_pesos_all(void){
	int k,i;
	float sum;
//----------------------------CALCULO DE ERROR PARA LA CAPA DE SALIDA--------------------------
	for (k=0; k<n_N_SAL; k++)
		n_c_salida.error[k]=n_c_salida.out[k]*(1-n_c_salida.out[k])*(n_Target[k]-
n_c_salida.out[k]);
//----------------------------CALCULO DE ERROR PARA LA CAPA ESCONDIDA-------------------------
	for (k=0; k<n_N_HID; k++){
		sum=0;
		for (i=0; i<n_N_SAL; i++)
			sum=sum+n_c_salida.error[i]*n_c_salida.pesos[i][k];
		n_c_escondida.error[k]=n_c_escondida.out[k]*(1-n_c_escondida.out[k])*sum;
	}
//----------------------------CORRECCION DE PESOS PARA LA CAPA DE SALIDA----------
	for (k=0; k<n_N_SAL; k++)
		for (i=0; i<n_N_HID; i++){
```

```
n_c_salida.pesos[k][i]=n_c_salida.pesos[k][i]+n_Eta*n_c_salida.error[k]*n_c_escondida.out[i];
        }
//----------------------------CORRECCION DE PESOS PARA LA CAPA ESCONDIDA----------------
        for (k=0; k<n_N_HID; k++)
                for (i=0; i<n_N_IN; i++){
                        n_c_escondida.pesos[k][i]=
                        n_c_escondida.pesos[k][i]+n_Eta*n_c_escondida.error[k]*n_Entrada[i];
                }

}
```

Esto corresponde a una iteración de Backpropagation, para que la RNA aprenda se debe mostrar el mismo ejemplo hasta disminuir la variación del error a un valor cercano a cero, pero el programa ilustrado en la Figura 55 indicará los ejemplos de forma aleatoria, porque experimentalmente se observó que esta forma de entrenamiento, a diferencia de indicar los ejemplos de forma ordenada, produce un comportamiento más adecuado y estable de las neuronas de la RNA cuando las imágenes mostradas como entradas varían de forma abrupta.

En las siguientes secciones se indicará el procedimiento relacionado a indicar las imágenes ejemplos y definir los valores objetivos (target) para las distintas RNA utilizadas, ya que cada una requiere métodos de aprendizaje diferentes según cómo se desea que se comporten. Es importante señalar que al ser un entrenamiento supervisado, éste termina una vez que se observa que las salidas y los targets se activan al mismo tiempo y poseen valores similares, por ello es importante desplegar una interfaz que indique el proceso de entrenamiento.

3.5.1. RNA de reconocimiento de la escala del instrumento

Se debe seleccionar los objetivos (targets) de manera que cada neurona aprenda a activarse de forma conveniente de acuerdo al desplazamiento de la ROI (como se indicó en la sección 3.3.2); para las RNA del reconocimiento de la escala del instrumento analógico se propone un ciclo de movimientos aleatorios de la ROI dentro de un rango de píxeles, de modo que si el instrumento está perfectamente centrado se fija el vector target en {0, 1, 0, 0, 1, 0, 0} (siendo 1=0.9 y 0=0.1), para una desviación a la derecha el target es {0, 1, 0, 0, 0, 1, 0}, hacia arriba y a la derecha {1, 0, 0, 0, 0 , 1, 0} y de esa forma según corresponda,

existe el caso en el que el target se define en {0, 0, 0, 0, 0, 0, 1} cuando se habilite una fuente interna de ruido, esto para que la RNA pueda identificar la no presencia de la forma de la escala del instrumento.

Una vez definidos los target se corrige los pesos con Backpropagation, es importante mencionar que se definen 200 ciclos de movimientos aleatorios, de los cuales cada 5 veces se asegura la posición de la región central, y 50 ciclos de ruido aleatorio, a todo este conjunto de ciclos se los define como una iteración de entrenamiento. A continuación se presenta el código que define los targets, luego ejecuta Backpropagation con las funciones *n_calcular_capa_escondida(),n_calcular_capa_salida(), corregir_pesos_all()* y finalmente presenta los targets y las salidas.

```
if(b_entrenamiento_2==1){
	aux_noise++;
	if(aux_noise<=200){b_ruido=0;}else{b_ruido=1;}
	if(aux_noise==250){
		b_ruido=0;
		aux_noise=1;
	}
	if(!b_ruido){
		n_Target[6]=0.1;
		if(x_roi==60){
			n_Target[0]=0.1;n_Target[1]=0.9;n_Target[2]=0.1;
		}else{
			if(x_roi<60){
				n_Target[0]=0.9;n_Target[1]=0.1;n_Target[2]=0.1;
			}else{
				n_Target[0]=0.1;n_Target[1]=0.1;n_Target[2]=0.9;
			}
		}
		if(y_roi==20){
			n_Target[3]=0.1;n_Target[4]=0.9;n_Target[5]=0.1;
		}else{
			if(y_roi<20){
				n_Target[3]=0.9;n_Target[4]=0.1;n_Target[5]=0.1;
			}else{
				n_Target[3]=0.1;n_Target[4]=0.1;n_Target[5]=0.9;
			}
		}
		contador_veces++;
		if(contador_veces==5){
			x_roi=60;  y_roi=20;
			contador_veces=0;
		}else{
			mov11x=(rand()%81)-40;
			mov22y=(rand()%31)-15;
```

```
                    x_roi=60+mov11x;
                    y_roi=20+mov22y;
                }

            }else{
                cout << "RUIDO HABILITADO\n";
                n_Target[0]=0.1;n_Target[1]=0.1;n_Target[2]=0.1;n_Target[3]=0.1;
                n_Target[4]=0.1;n_Target[5]=0.1;n_Target[6]=0.9;
                x_roi=60; y_roi=20;
                contador_veces=0;
             }
             n_get_ROI(im_ROI);
             dibujar_ROI();
             n_Gain=0.5;
             n_calcular_capa_escondida();
             n_Gain=1.5;
             n_calcular_capa_salida();
             n_corregir_pesos_all();

cout<<"targuet:"<<n_Target[0]<<n_Target[1]<<n_Target[2]<<""<<n_Target[3]<<n_Target[4]<<n_
Target[5]<<"  "<<n_Target[6]<<"\n";
cout<<"out:"<<n_c_salida.out[0]<<n_c_salida.out[1]<<n_c_salida.out[2]<<""<<n_c_salida.out[3]
<<n_c_salida.out[4]<<n_c_salida.out[5]<<"  "<<n_c_salida.out[6]<<"\n";
            imshow("ROI1", im_ROI1);
            n_plot_targets();
            n_plot_outputs();

}
```

En las Figuras 56 y 57 se indica cómo se produce el movimiento de la ROI que extrae la imagen para entregarla a la RNA, se puede observar el comportamiento de las neuronas (conjunto de líneas de la izquierda) y el estado de los targets (conjunto de líneas azules de la derecha).

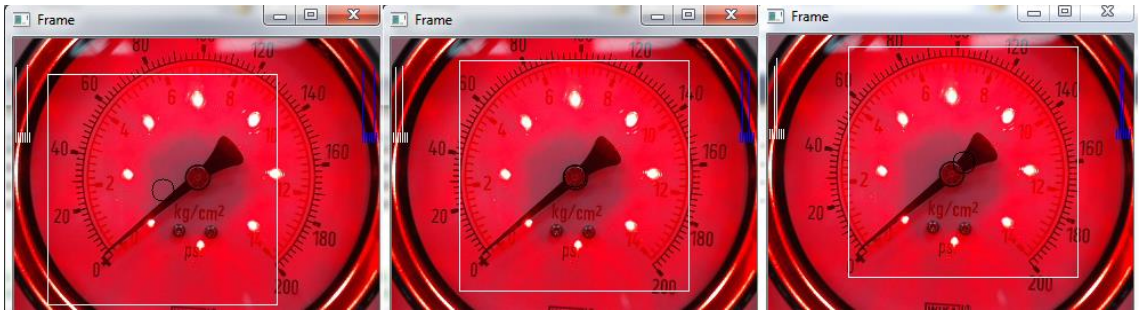


Figura 56: Secuencia de imágenes del entrenamiento de la RNA que reconoce la escala del manómetro.

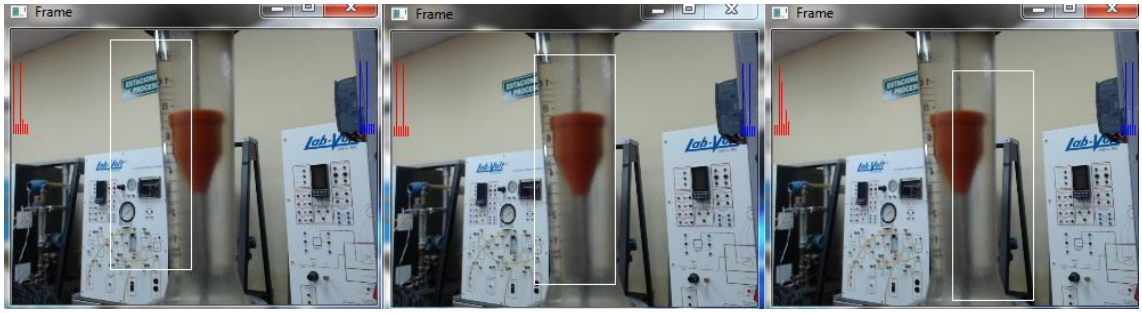


Figura 57: Secuencia de imágenes del entrenamiento de la RNA que reconoce la escala del rotámetro.

3.5.2. RNA de reconocimiento del indicador del manómetro

En este entrenamiento el instrumento debe estar en operación marcando una medida fija, es decir en un ángulo conocido, por lo que la ROI de 50x20 píxeles se mueve para simular el movimiento del indicador cuando existe cambios del valor de la variable. El vector target se fija de acuerdo a la apariencia del movimiento, dicho de otra forma cuando la ROI rote de 1 a 5 grados hacia la derecha, parecerá que el indicador se movió de 1 a 5 grados a la izquierda, entonces el vector target se fija en {0,1,0,0,0,0}, en cambio si la rotación es hacia el lado izquierdo entonces el movimiento aparente será hacia la derecha y el target es {0,0,0,1,0,0}.

Los elementos 0 y 4 del target se guían por la misma regla, solo que el movimiento está comprendido de 6 a 20 grados; el target se fija en {0, 0, 1, 0, 0, 0} cuando la región del indicador del instrumento está justo en el centro de la ROI y se fija en {0, 0, 0, 0, 0, 1} cuando la ROI se haya movido a cualquier lugar en donde no exista ninguna forma relacionada al indicador.

Una iteración de entrenamiento está dada por un ciclo de 20 movimientos aleatorios, en donde en 3 ocasiones se asegura la posición central del indicador y 1 vez en una región lejana al mismo. Como se indica en las siguientes líneas de código:

```
if(b_entrenamiento_1){
        if(contador_veces==10 || contador_veces==15){
```

```
		fi=90;
	}else{
		if(contador_veces==20){
			fi=rand()%308;
			fi=116+fi;
			if(fi>360) fi=fi-360;
			contador_veces=0;
		}
		else{
			fi=(rand()%50)-25;
			fi=90+fi;
		}
	}
	contador_veces++;
	if(fi>=91 && fi<=95){
	  Target[0]=0.1;Target[1]=0.9;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
	}
	if(fi>=96 && fi<=115){
	  Target[0]=0.9;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
	}
	if(fi==90){
	  Target[0]=0.1;Target[1]=0.1;Target[2]=0.9;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
	}
	 if(fi<=89 && fi>=85){
	  Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.9;Target[4]=0.1;Target[5]=0.1;
	}
	if(fi<=84 && fi>=65){
	  Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.9;Target[5]=0.1;
	}
	if(fi>=116 || fi<=64){
	  Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.9;
	}
	fi=fi-108; //se hace un recorrido para definir el punto central (ángulo) en donde se
encuentre el indicador (es decir si el indicador está en 0 grados se recorre 90 grados a la izquierda
fi=fi-90, ya que en el código se considera siempre a fi=90 como el punto de partida)
}
get_ROI(im_ROI);
dibujar_indROI();
Gain=0.5;
calcular_capa_escondida();
Gain=1.5;
calcular_capa_salida();
if(b_entrenamiento_1){corregir_pesos_all();} //habilitar el entrenamiento de la 1ra red
plot_outputs();
plot_targets();
if(!b_entrenamiento_1){
	get_winner();
	seguir_indicador();
	 calcular_variable();
}else{
	cout<<"targuet:"<<Target[0]<<Target[1]<<Target[2]<<Target[3]<<Target[4]<<Target[5]<
<"\n";
	cout<<"out:"<<c_salida.out[0]<<c_salida.out[1]<<c_salida.out[2]<<""<<c_salida.out[3]<<
c_salida.out[4]<<c_salida.out[5]<<"\n";
}
```

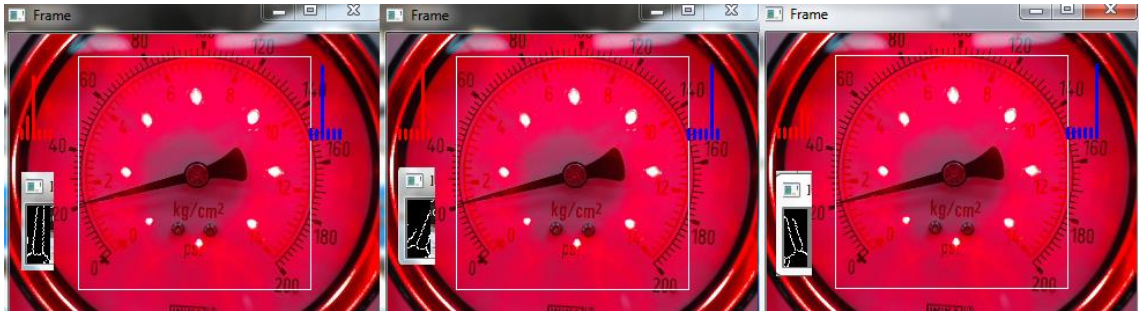
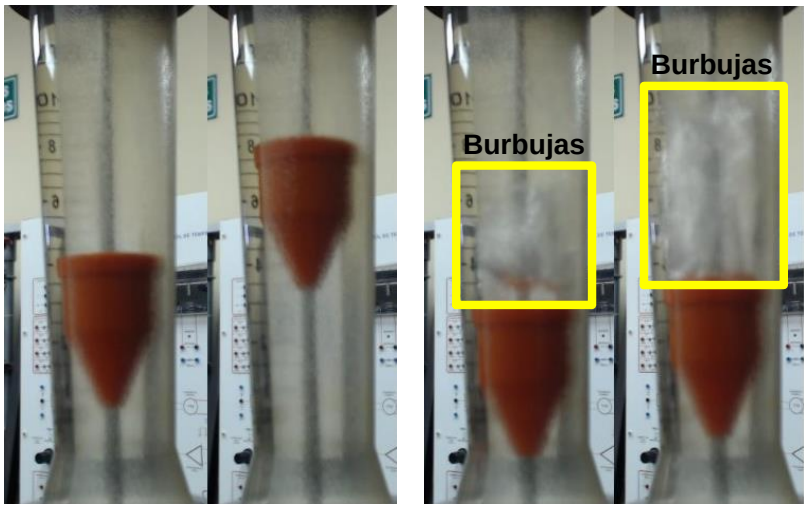


Figura 58: Secuencia de imágenes del entrenamiento de la RNA que reconoce el indicador del manómetro.

3.5.3. RNA de reconocimiento del flotador del rotámetro

De forma semejante al caso del indicador del manómetro, éste proceso de entrenamiento se basa en el movimiento vertical de la ROI para aparentar un movimiento del flotador mientras éste se encuentra en una posición fija, si se mueve verticalmente de 1 a 5 píxeles hacia arriba el vector target se fija en {0,0,0,1,0,0} y {0,1,0,0,0,0} si es hacia abajo, es decir según la apariencia de movimiento; y de esta manera según corresponda en cuanto a la dirección y valor de movimiento.



Flujo estable

Burbujas generadas por la circulación del líquido

Figura 59: Aparición de burbujas en el interior del rotámetro cuando se producen cambios rápidos de la circulación del líquido.

En este caso también se contempla un problema a enfrentar, mismo que trata de la aparición aleatoria de burbujas en el interior del rotámetro generadas por la variación de velocidad de la circulación del líquido, como se indica en la Figura 59, a pesar de que la mayoría del tiempo el flujo permanece relativamente estable, en este trabajo se adapta la RNA para eliminar los efectos de este problema, mediante la creación de un algoritmo de entrenamiento a base de ruido que simula la aparición de formas inesperadas en la imagen.

El entrenamiento está basado en la misma secuencia de ejemplos que se le indica a la RNA como en el anterior caso correspondiente al indicador del manómetro, pero se adiciona 10 ciclos de una secuencia de ejemplos de imágenes de movimiento del flotador invadiendo con ruido al espacio de la ROI que está 3 píxeles por encima y por debajo del borde del flotador (esto se puede distinguir en la Figura 60). Lo mencionado se estructura en el siguiente código.

```
if(b_entrenamiento_1){
        if(aux_noise<=10){b_ruido=0;}else{b_ruido=1;}
        if(aux_noise==20){b_ruido=0; aux_noise=0;}
        if(contador_veces==10 || contador_veces==15){
                fi=100;
        }else{
                if(contador_veces==20){
                        fi=(rand()%41)-20;
                        if(fi<0)fi=89+fi;
                        else fi=111+fi;
                        contador_veces=0;
                        aux_noise++;//aqui para q se cumplan 10 ve
                }else{
                        fi=(rand()%20)-10;
                        fi=100+fi;
                }
        }
        contador_veces++;
        if(fi>=101 && fi<=105){
          Target[0]=0.1; Target[1]=0.9;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
        }
        if(fi>=106 && fi<=110){
          Target[0]=0.9;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
        }
        if(fi==100){
          Target[0]=0.1;Target[1]=0.1;Target[2]=0.9;Target[3]=0.1;Target[4]=0.1;Target[5]=0.1;
        }
        if(fi<=99 && fi>=95){
          Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.9;Target[4]=0.1;Target[5]=0.1;
        }
```

```
            if(fi<=94 && fi>=90){
              Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.9;Target[5]=0.1;
            }
            if(fi>=111 || fi<=89){
              Target[0]=0.1;Target[1]=0.1;Target[2]=0.1;Target[3]=0.1;Target[4]=0.1;Target[5]=0.9;
            }
}
get_ROI(im_ROI);
dibujar_indROI();
Gain=0.5;
calcular_capa_escondida();
Gain=1.5;
calcular_capa_salida();
if(b_entrenamiento_1){corregir_pesos_all();} //habilitar el entrenamiento de la 1ra red
            plot_outputs();
            plot_targets();
            cout << "fi= "<<fi<<"\n";
            cout<<"targuet:"<<Target[0]<<Target[1]<<Target[2]<<""<<Target[3]<<Target[4]<<Target[
5]<<"\n";
            cout<<"out:"<<c_salida.out[0]<<c_salida.out[1]<<c_salida.out[2]<<""<<c_salida.out[3]<<
c_salida.out[4]<<c_salida.out[5]<<"\n";
```

En la Figura 60 se indica una secuencia de imágenes capturadas durante el entrenamiento, se puede distinguir de forma gráfica el comportamiento de las neuronas (líneas de color rojo) y los targets (líneas de color azul), además se puede observar cómo se genera el ruido por encima y por debajo del borde del flotador en la esquina superior derecha de las dos primeras imágenes.

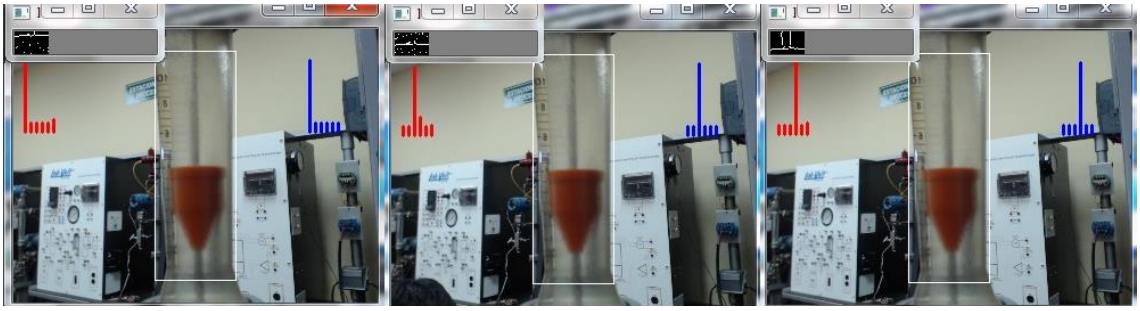


Figura 60: Secuencia de imágenes del entrenamiento de la RNA que reconoce el borde del flotador del rotámetro, se incluye la invasión con ruido por encima y por debajo del borde del flotador.

3.6. Desarrollo del HMI del sistema servidor

Se ha implementado un sistema servidor de la red inalámbrica que adquiere los datos provenientes de las tarjetas embebidas de forma inalámbrica; éstos datos son presentados y almacenados, siguiendo los lineamientos descritos en la sección 3.1.4, se ha elaborado un proyecto en el software LabVIEW que contiene el HMI intuitivo que presenta al usuario el valor de las variables del procesos; y los servicios correspondientes a la supervisión de procesos industriales: tendencias de variables, alarmas, históricos de alarmas y seguridades de usuarios.

En este trabajo se ha logrado que LabVIEW se complemente con diferentes lenguajes de programación para la creación de una página web orientada al servicio de monitoreo remoto (sección 3.8) y con el software Xampp que opera como un servidor local para la creación de una base de datos independiente que se gestiona de forma independiente (sección 3.7). En la Figura 61 se indica cómo está estructurada la programación en LabVIEW para el cumplimiento de las funciones propuestas para el sistema servidor. En las Figuras 62, 63, 64 y 65 se indican las interfaces del HMI.

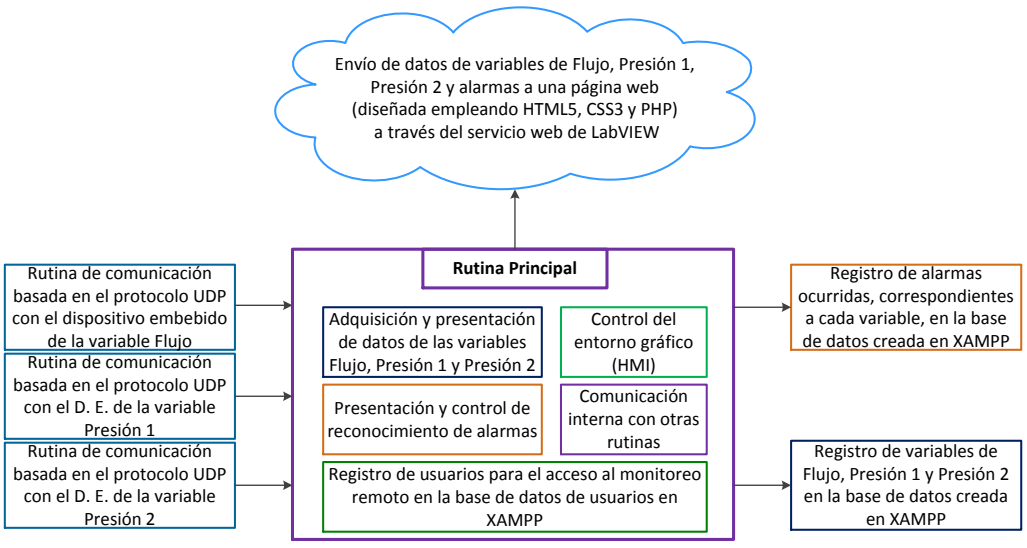


Figura 61: Diagrama de bloques de la programación en LabVIEW para la creación de un sistema servidor.

En la interfaz principal (Figura 63) se presenta, de forma numérica y gráfica, el estado del valor de las variables de los distintos procesos; aquí también se visualiza el estado de la conexión con cada tarjeta embebida a través de objetos indicadores y mensajes; se ha considerado que existan controles de tipo switch para terminar y reiniciar la conexión con la tarjeta embebida, y controles numéricos para establecer el tiempo de registro del valor de la variable en la base de datos; por cuestiones de seguridad éstos controles solo habilitan cuando el usuario administrador se haya identificado.

Figura 62: Interfaz para que el usuario se identifique.

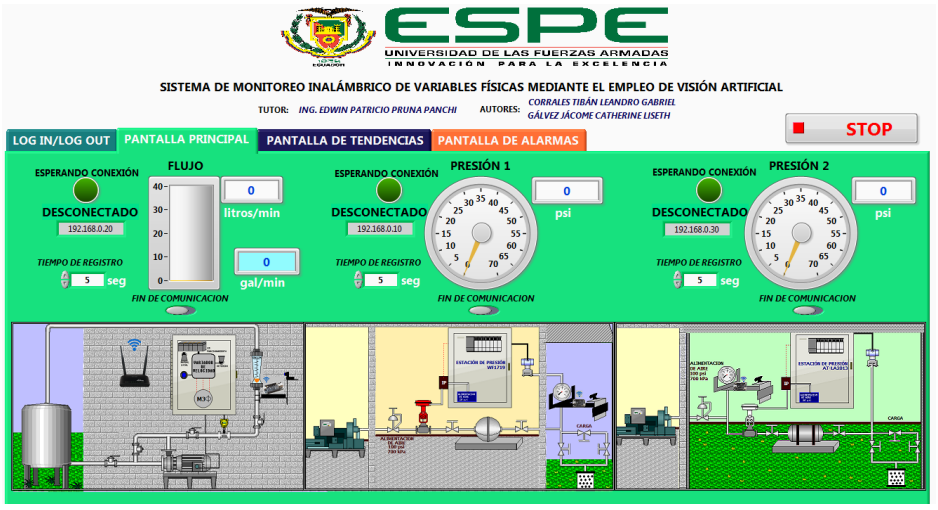


Figura 63: Interfaz principal que contiene el diagrama de cada proceso y presenta el valor de las variables.

Figura 64: Interfaz que contiene cuadros donde se grafican las tendencias de las variables.

Figura 65: Interfaz que presenta el estado de las alarmas de cada variable y sus respectivos históricos.

En la interfaz de la Figura 65 se presenta, para cada variable, indicadores de los cuatro tipos de alarmas: HIGH HIGH, HIGH, LOW, LOW, LOW; además éstas alarmas se van registrando en las tablas de históricos de alarmas para una detección inmediata de los eventos emergentes que han sucedido, éstas alarmas también se registran en la base de datos en XAMPP (sección 3.7). Es importante mencionar que cada vez que se produce una alarma se muestra ésta interfaz para que el usuario pueda reconocer la misma.

A continuación se indican secciones importantes de la programación gráfica en LabVIEW. En la Figura 66 se presenta la programación que brinda seguridad en el manejo del HMI, aquí se produce la comparación que determina si el usuario tiene los permisos para modificar ciertos parámetros, como el tiempo de registro de las variables en la base de datos, la finalización o reinicio de la comunicación con las tarjetas, la adición de nuevos usuarios para el monitoreo a través de la web y la finalización del programa.

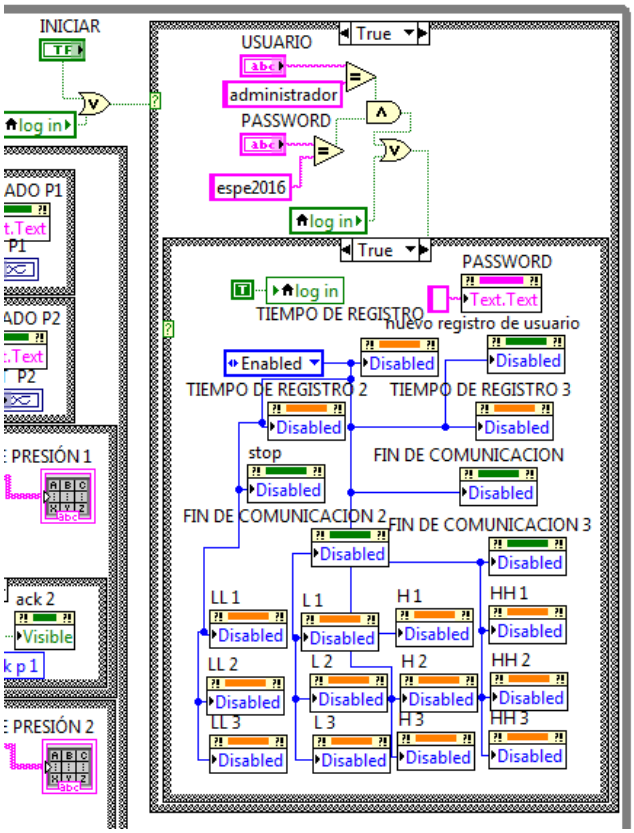


Figura 66: Generación de ruido por encima y por debajo del borde del flotador de rotámetro.

En las Figuras de la 67 a la 69 se indican las rutinas de comunicación con las tarjetas embebidas utilizando funciones del protocolo UDP que contiene LabVIEW; es importante mencionar que la comunicación con cada tarjeta embebida involucra un lazo repetitivo (*while loop*) independiente para que la adquisición de datos de las tres variables se produzca de forma simultánea.

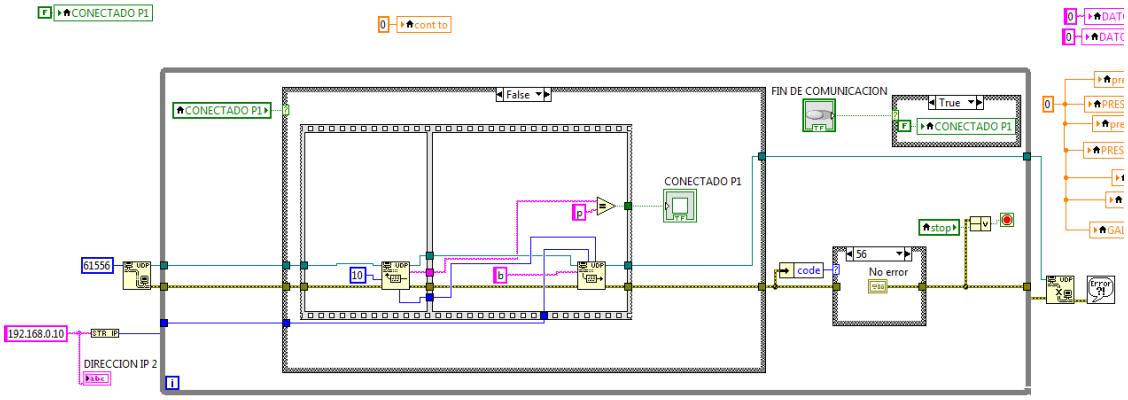


Figura 67: Rutina de comunicación con el dispositivo de dirección IP 192.168.0.10 y puerto 61556. Sincronización de inicio.

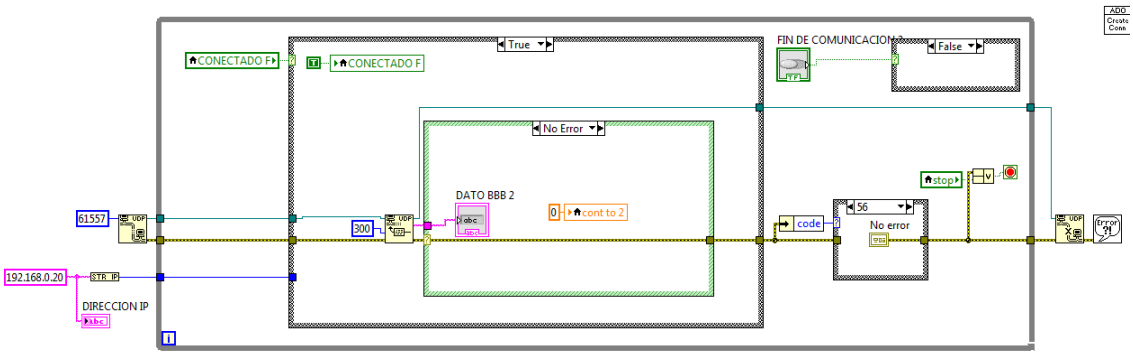


Figura 68: Rutina de comunicación con el dispositivo de dirección IP 192.168.0.20 y puerto 61557. Adquisición del dato en tipo string.

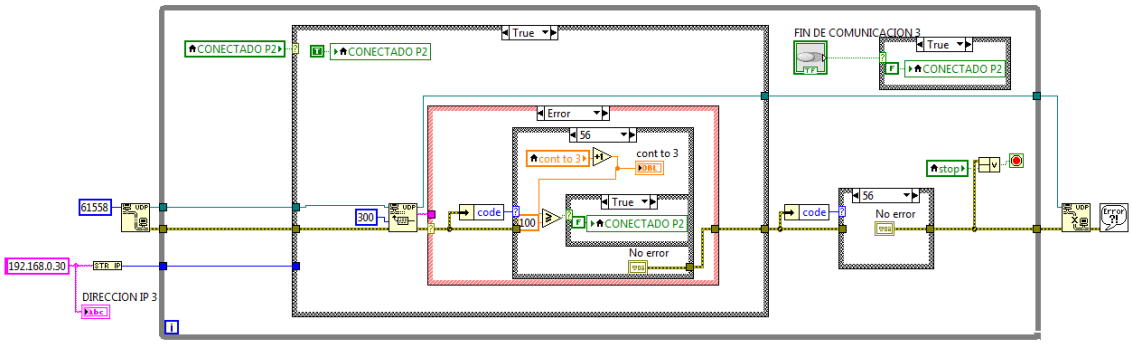


Figura 69: Rutina de comunicación con el dispositivo de dirección IP 192.168.0.30 y puerto 61558. Subrutina de detección de error en la comunicación.

En la Figura 70 se indican cómo se obtienen los datos correspondientes al valor de las variables y se las transforma de tipo string a tipo flotante, esto se ejecuta dentro de una rutina principal contenido en un *while loop* (como se estructura en la Figura 54) al igual que la detección de las alarmas para su registro en las tablas de históricos del mismo HMI que se indica en la Figura 71.

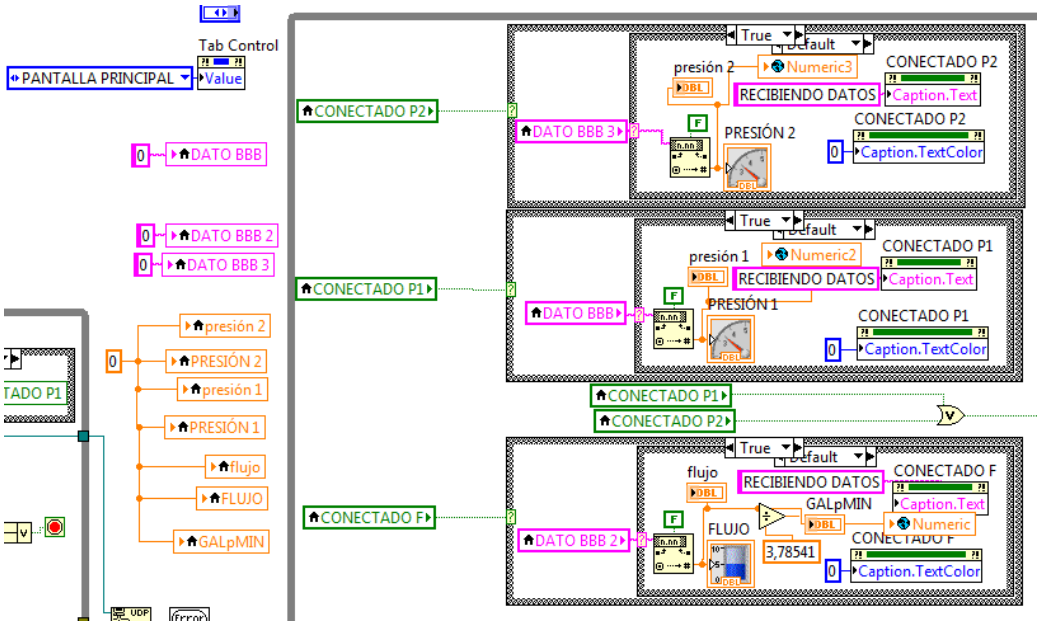


Figura 70: Adquisición y conversión de los datos tipo string, correspondiente al valor de las variables de procesos para ser manejados de forma flotante.

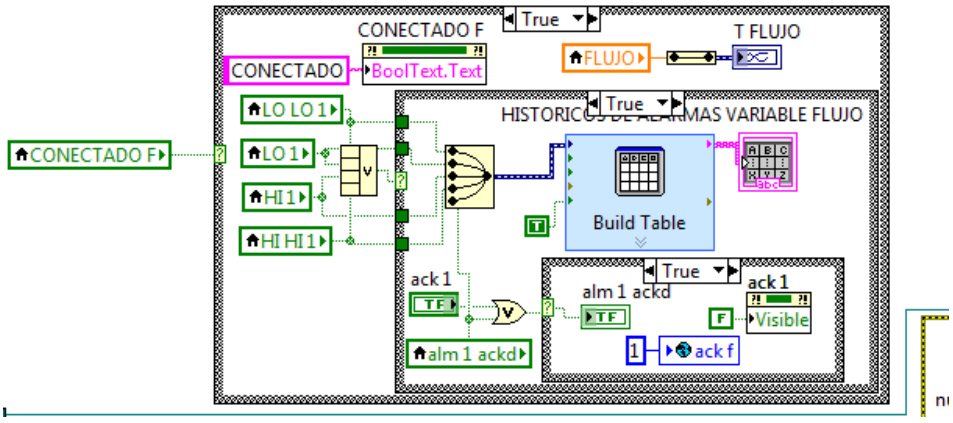


Figura 71: Rutina que detecta el estado de las alarmas y las registra en la tabla de históricos de la interfaz de alarmas

3.7. Desarrollo de la base de datos

3.7.1. Configuración del software XAMPP

Dentro de un sistema de monitoreo industrial es necesario almacenar datos relevantes sobre la planta para la creación de planes de mantenimiento preventivo y toma de decisiones en base a un análisis, por esta razón en la aplicación se realiza la implementación de una base de datos para el registro de información relacionada con la medida de las variables, los estados de las alarmas y los usuarios que tendrán acceso al entorno de monitoreo remoto, con la finalidad de proporcionar seguridad al mismo; para llevar a cabo lo mencionado se procede a la creación de la base de datos MySQL empleando el software XAMPP, cuyas opciones de seguridad deben ser configuradas para trabajar de forma confiable.

Una vez instalado el software XAMPP, es necesario ejecutar el programa como administrador y posteriormente en el cuadro de diálogo llamado panel de control se debe instalar Apache y MySQL, como servicios de Windows (al arrancar el ordenador, los servicios de Windows se inician de forma automática), ya que para el desarrollo de la aplicación es necesario que Apache y MySQL se encuentren iniciados de forma permanente para poder hacer uso del servidor web y la base de datos, a continuación en las Figuras 72 y 73 se indica el procedimiento.

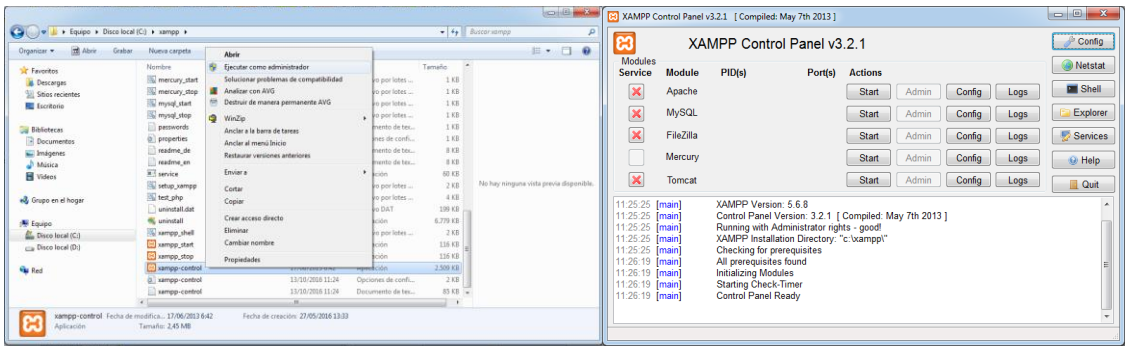


Figura 72: Acceso al panel de control de XAMPP como administrador.

Apache utiliza dos puertos de comunicación, el 80 como puerto principal y 443 como puerto de comunicación seguro mientras que MySQL emplea el puerto de comunicación 3306 como puerto principal. Para que el usuario desde un navegador web pueda acceder a la página web de XAMPP y de phpMyAdmin (página para manejar la administración de MySQL) se utiliza el puerto 80 y el protocolo de presentación HTTP.

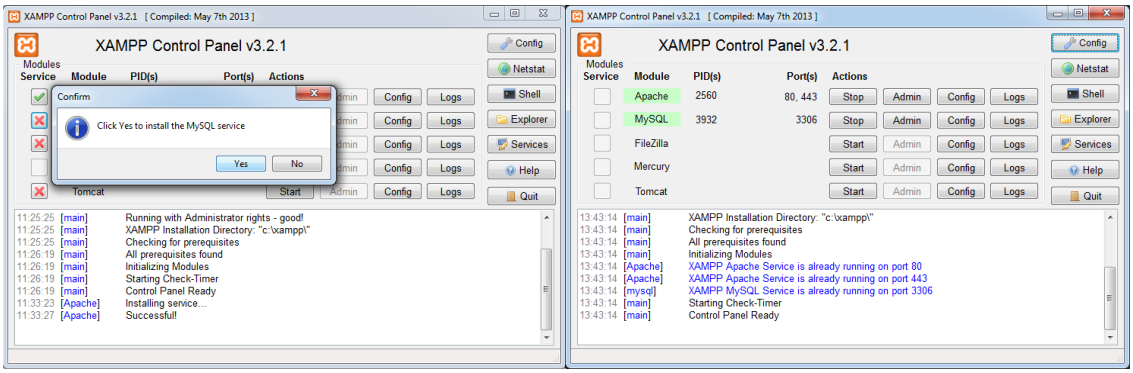


Figura 73: Instalación de Apache y MySQL como servicios de Windows en el panel de control y verificación de inicio correcto al reiniciar el ordenador.

Seguidamente para acceder a las opción de configuración de la seguridad de XAMPP, se debe ingresar al enlace http://localhost/xampp/ en cualquier navegador, dentro de este entorno debe buscase la opción Chequeo de seguridad para acceder en ella, posteriormente se da clic sobre la dirección http://localhost/security/xamppsecurity.php, para proporcionar la clave tanto a la sección MySQL (root) como al directorio XAMPP (.htaccess) y restringir el acceso a la base de datos como a las páginas web de XAMPP, finalmente se verifica en la opción Chequeo de seguridad que las opciones de estado se muestren como seguro, la configuración se puede apreciar en las siguientes Figuras 74 y 75.

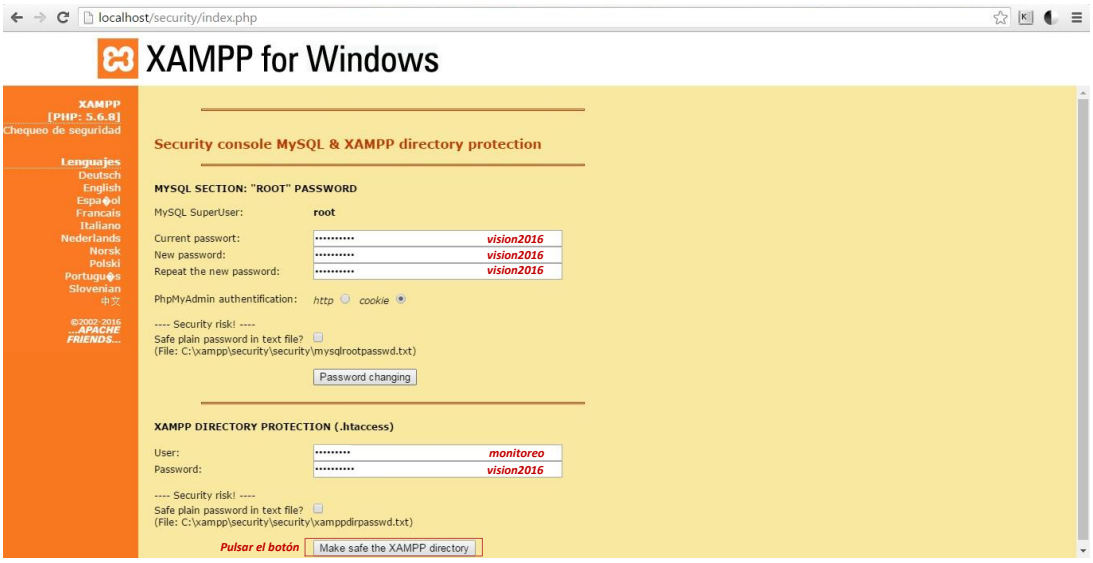


Figura 74: Configuración de las claves de acceso para la sección MySQL y el directorio XAMPP.

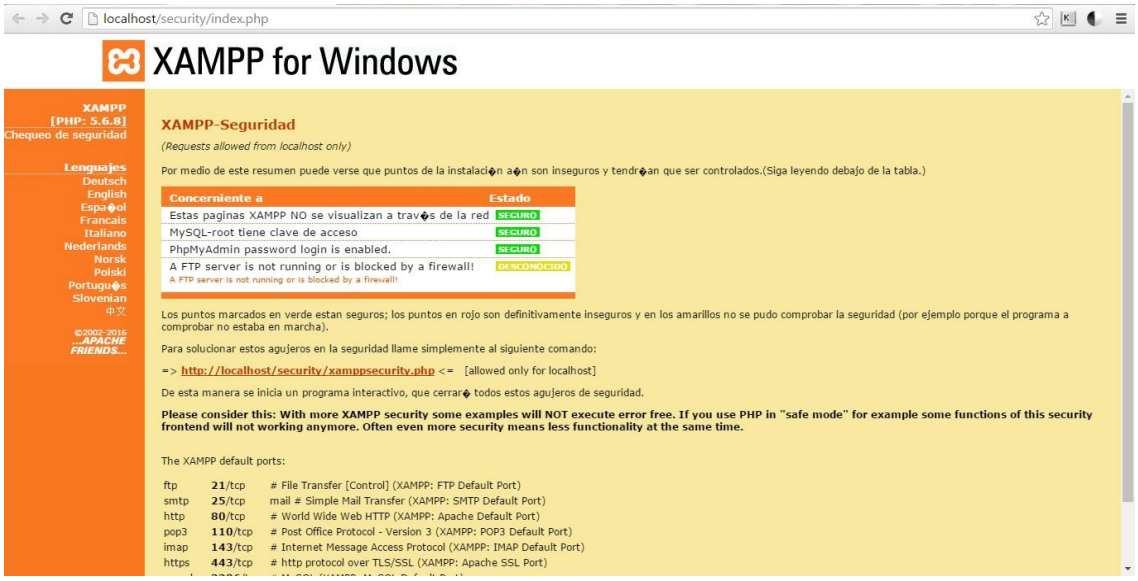


Figura 75: Verificación del estado de la seguridad de XAMPP.

3.7.2. Creación de la base de datos y las distintas tablas en el software XAMPP

A continuación se explica el procedimiento para crear la base de datos necesaria para el desarrollo de la aplicación, la cual se conforma de distintas

tablas destinadas al registro de datos de las diferentes variables, los estados de alarma y los usuarios; las tablas poseen una estructura compuesta de varias columnas que dependen de la información que se desee almacenar.

Para acceder al panel que permite crear una base de datos, se ingresa a la siguiente dirección http://localhost/phpmyadmin/, es importante tomar en cuenta que no es necesario el acceso a internet; la creación de misma se la realiza en el mismo servidor (donde se añade el directorio localhost), sin embargo, si es necesario realizarlo de forma remota desde otro computador, que se encuentre en la misma red, se debe anteponer la dirección IP del servidor local, es decir, del ordenador donde está instalado el software XAMPP; posteriormente se presenta una interfaz donde se debe ingresar el usuario y la clave, que fue configurada en la opción seguridad, siendo respectivamente *root* y *vision2016*, como se muestra en la Figura 76.

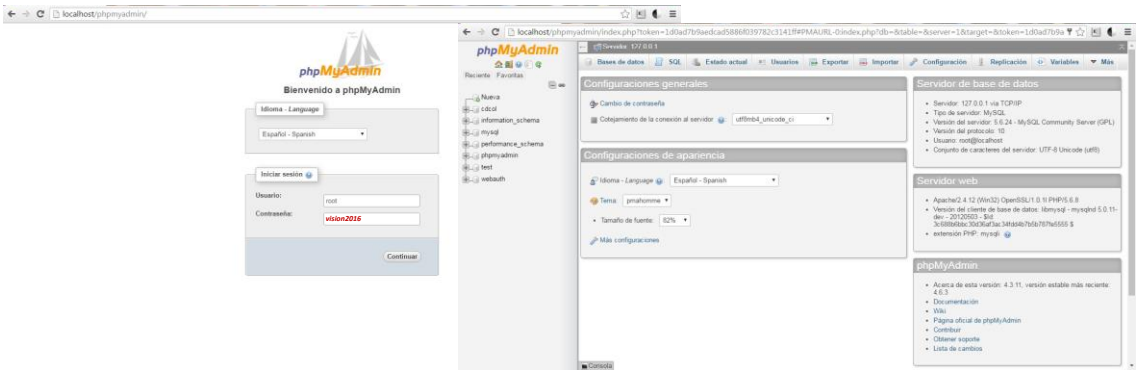


Figura 76: Identificación y acceso al entorno phpMyAdmin.

Una vez dentro del entorno PhpMyAdmin, se selecciona la palabra Nueva en el panel a lado izquierdo y se da un nombre a la base de datos en el recuadro *Crear base de datos*, que para esta aplicación se la conoce como *BD monitoreo por visión*, y se pulsa sobre el botón *Crear*, como se indica en la Figura 77, si se ha realizado correctamente saldrá un mensaje que dice, la base de datos *BD* monitoreo por visión ha sido creada.

Figura 77: Creación de la base de datos.

Para llevar a cabo la creación de tablas dentro de la base de datos, se debe seleccionar en el panel a lado izquierdo el nombre de la base de datos creada, posteriormente se da un nombre a la tabla en el campo *Nombre* dentro del recuadro *Crear tabla* y se especifica el número de columnas, este último depende de la información que se va a registrar, seguidamente al pulsar el botón *Continuar* se muestra la estructura de la tabla, en la cual, se debe dar un nombre a la información que se va a registrar y a su vez configurar los distintos parámetros para evitar errores en el registro de datos, finalmente al pulsar sobre el botón *Guardar* se muestra la tabla creada y en su interior las columnas que se agregaron. A partir de la primera tabla que se agrega dentro de la base de datos las siguientes se crean seleccionando la opción *Nueva*, misma que aparece dentro de la base de datos, igualmente en la estructura de la tabla se proporciona la información y se pulsa sobre el botón *Guardar*.

Se crean cinco tablas para el desarrollo de la aplicación, las cuales son: *datos de flujo, datos de presión 1, datos de presión 2, log in* y *registro de alarmas* cuyo número de columnas depende de la siguiente información; en *datos de flujo* el dato de identificación (id), la fecha, la hora, la lectura de la variable en litros y en galones por minuto; en *datos de presión 1* y *presión 2* el id, la fecha, la hora y la lectura de la presión en PSI; en *log in* el id, el usuario y la clave; y en *registro de alarmas* la fecha, la hora, el tipo de alarma, la lectura y la unidad de la variable. Los id son de tipo INT(11), pues, se desea conocer el número de datos que se están almacenando, y los demás son tipo TEXT, ya que, a la aplicación le interesa almacenar en este formato para poder hacer uso de esta información de forma

conveniente para el usuario, es decir descargar la información en formato PDF o texto plano o una hoja de Excel para proporcionar la facilidad de generar informes y/o reportes, en las Figuras de la 78 a la 82 se indica lo mencionado.

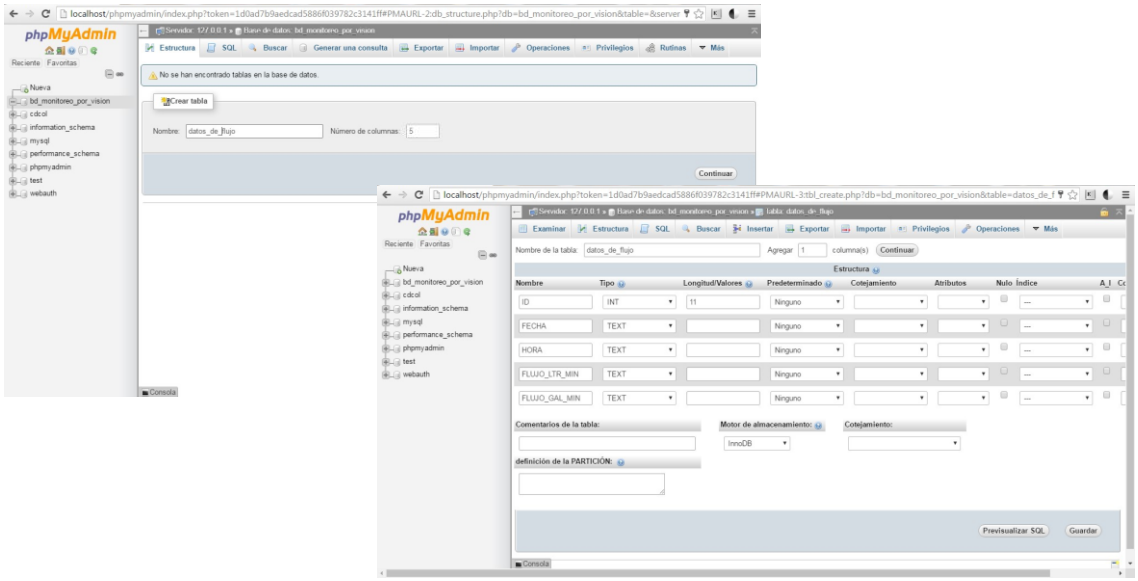


Figura 78: Creación y configuración estructura de la tabla datos de flujo.

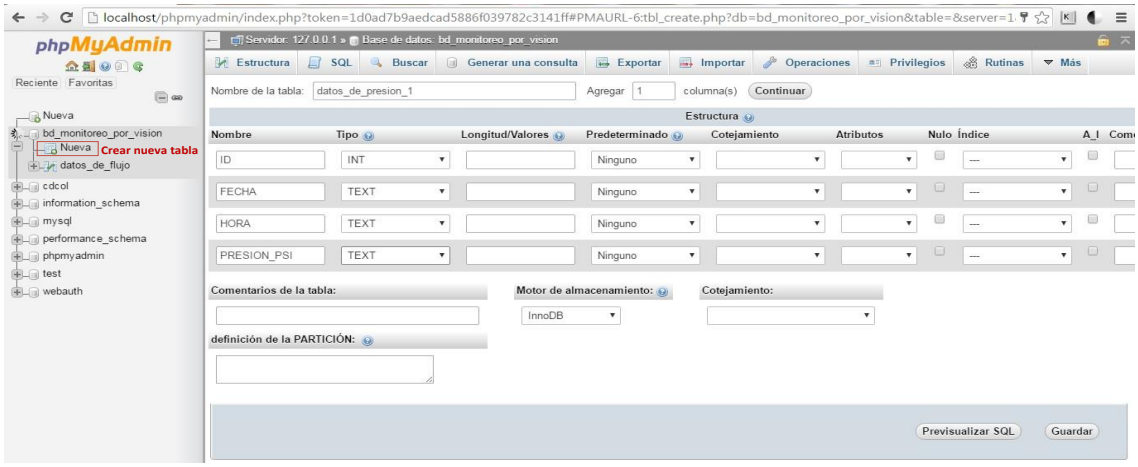


Figura 79: Creación y configuración de la estructura de la tabla datos de presión 1.

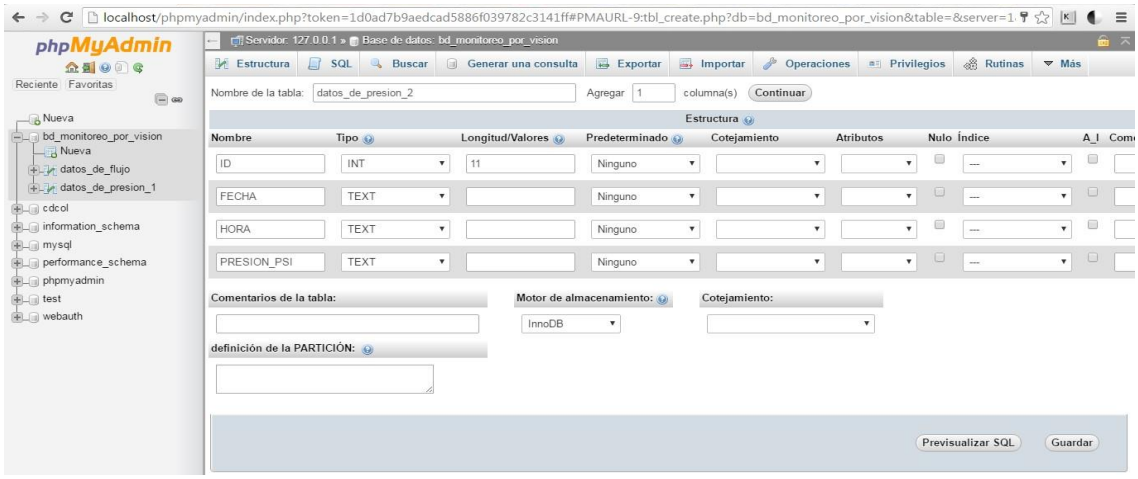


Figura 80: Creación y configuración de la estructura de la tabla datos de presión 2.

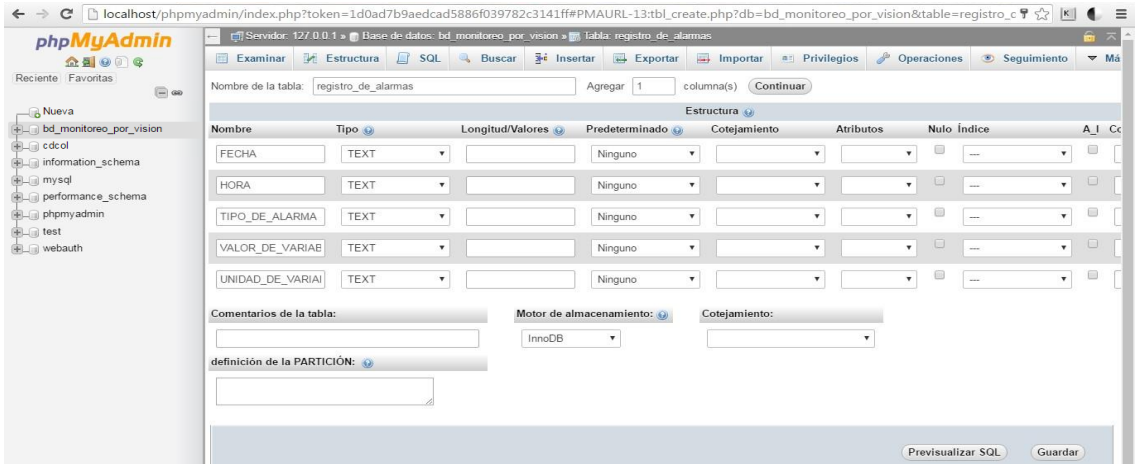


Figura 81: Creación y configuración de la estructura de la tabla registro de alarmas.

En la tabla destinada al registro de los usuarios se activa el campo auto incrementable (A_I), para que el dato de la columna ID se incremente cada vez que se realice un registro con la finalidad de conocer el número de usuarios; al activar el campo aparece una ventana donde se debe agregar un índice que para este caso es PRIMARY, el cual brinda opciones para mostrar la lista de registros en orden accedente o descendente de acuerdo a los datos de la columna ID, además admite hacer uso de varias alternativas (editar, copiar y/o borrar) que se encuentran junto al dato, lo cual es beneficioso, ya que puede ser necesario

modificar o eliminar a un usuario; pero no resulta seguro que los datos de las tablas creadas previamente puedan ser modificados o eliminados, ya que contienen información de las variables (flujo, presión 1 y 2), por lo tanto dichas tablas no poseen índice, y el número de datos registrados se provee desde la aplicación desarrollada en LabVIEW.

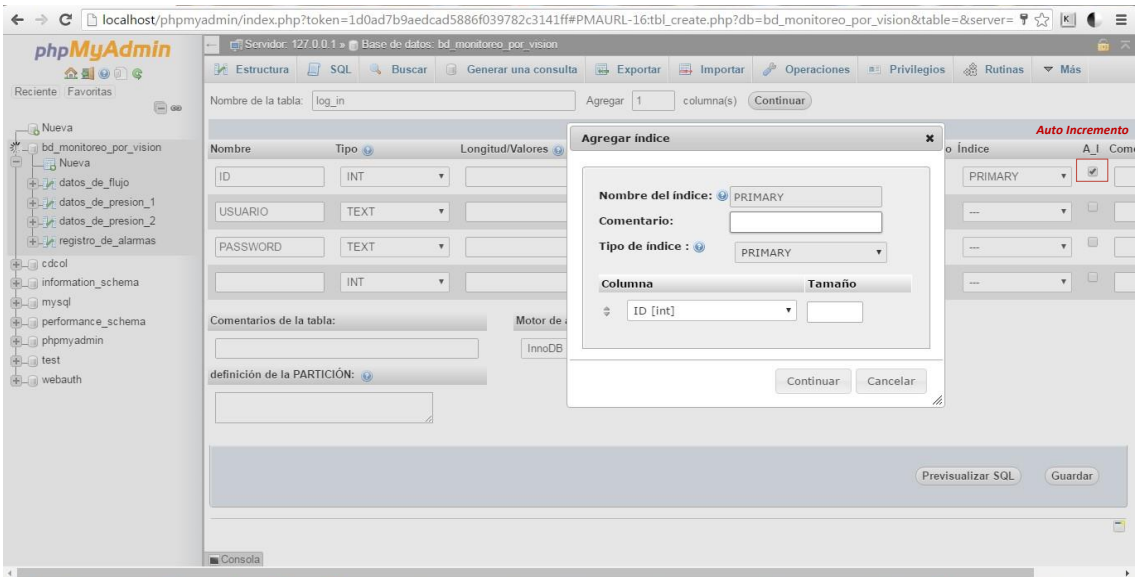


Figura 82: Creación y configuración de la estructura de la tabla log in.

3.7.3. Creación de DSN en el Administrador de orígenes de datos ODBC

La aplicación desarrollada sobre la plataforma LabVIEW, tiene varias tareas entre ellas la de ejecutar instrucciones que permiten registrar información en las diferentes tablas de la base de datos; para dicho propósito es necesario realizar la conexión entre la aplicación y la base de datos, por esta razón se instala un controlador de MySQL Connector / ODBC, el mismo que engloba un conjunto de controladores que permiten el acceso a una base de datos MySQL específica, empleando la interfaz Open Database Connectivity, ODBC; a continuación se agrega un Data Source Name (DSN) en el Administrado de orígenes de datos ODBC en el sistema operativo Windows empleando el controlador MySQL ODBC 5.1 Driver; los DSN en la aplicación desarrollada en la plataforma LabVIEW permiten la comunicación con la base de datos, de forma independiente, en

diferentes instantes de tiempo y sin la necesidad de acceder al programa de la misma, es decir se agregan varios DSN que tienen diferente identificación y se comunican con una misma base de datos; en este trabajo se utiliza el DSN tipo usuario debido a que se está trabajando sobre un ordenador con un usuario individual, como se indica en el diagrama de la Figura 83.

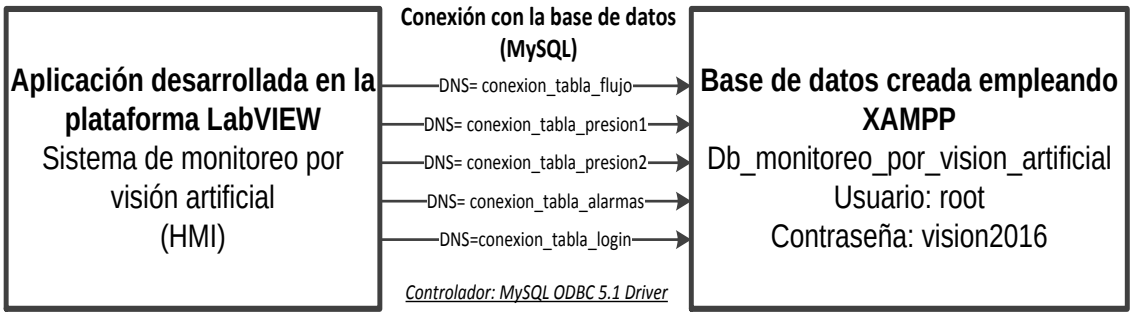


Figura 83: Diagrama de comunicación entre la aplicación desarrollada en LabVIEW y la base de datos en XAMPP empleando los DSN.

En el sistema operativo Windows se accede a la ruta C:\Windows\SysWOW64, se busca la aplicación odbcad32, se ejecuta como administrador, en el cuadro de dialogo llamado Administrador de orígenes de datos ODBC, se selecciona la pestaña DSN usuario, se pulsa sobre el botón Agregar, para crear un DSN de este tipo, en la ventana llamada Crear nuevo origen de datos, se elige MySQL ODBC 5.1 Driver, posteriormente en el cuadro de dialogo llamado MySQL Connector/ODBC Data Source Configuration, se configura los parámetros para la comunicación con una base de datos específica y se pulsa sobre el botón OK para agregar el DSN en la pestaña usuario DSN, en las Figuras 84 y 85 se indica lo descrito.

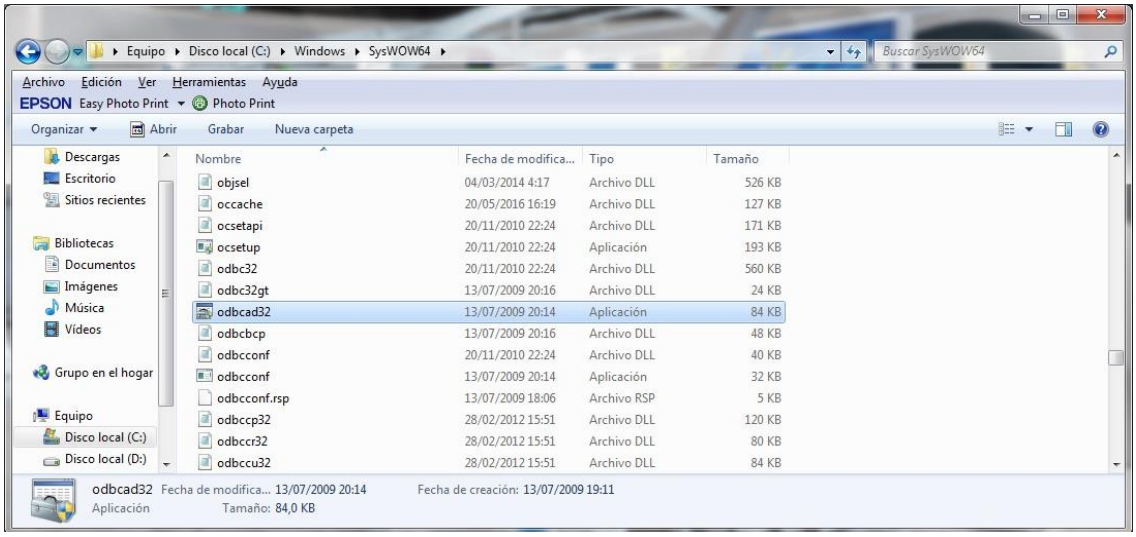


Figura 84: Acceso a la ruta C:\Windows\SysWOW64 y selección de la aplicación odbcad32.

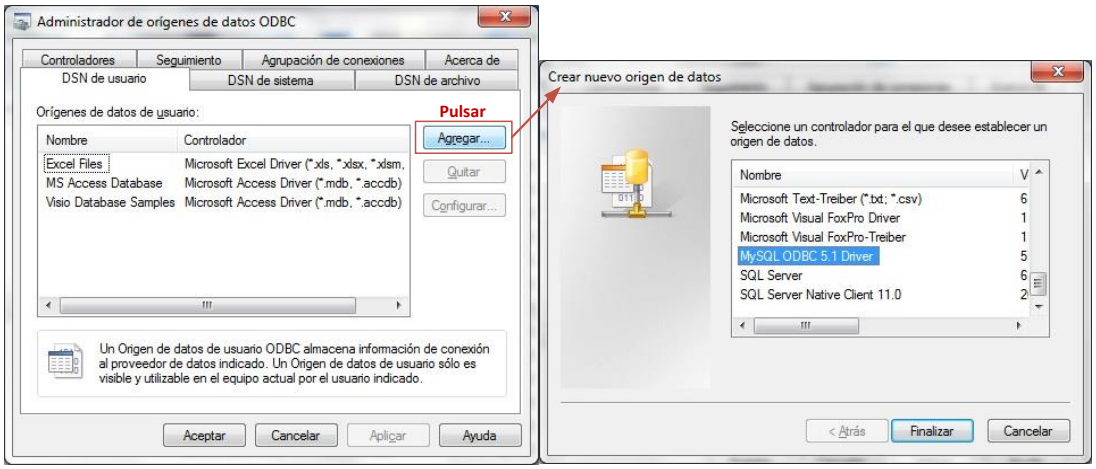


Figura 85: Ventana que permite agregar DSN de usuario (Administrador de orígenes de datos ODBC) y ventana para seleccionar el controlador (Crear nuevo origen de datos).

Para este trabajo se emplean cuatro DSN de usuario cuyos parámetros de conexión de la ventana *MySQL Connector/ODBC Data Source Configuration* son configurados así, en Data Source Name se proporciona un nombre al DSN de usuario, para identificar la conexión; en descripción se detalla algún dato relevante sobre la conexión; en TCP/IP Server se agrega la dirección del

ordenador donde se almacenarán la información (localhost) y el puerto para la transferencia de datos empleando el protocolo de TCP/IP (automático); en User se provee el nombre del usuario que tiene acceso a la base de datos (root); en Password se suministra la clave de dicho usuario (vision2016); en Database se selecciona la base de datos con la que desea establecer una conexión, y se pulsa el botón Test para verificar que la comunicación no tenga errores, como se indica en las Figuras 86 y 87.

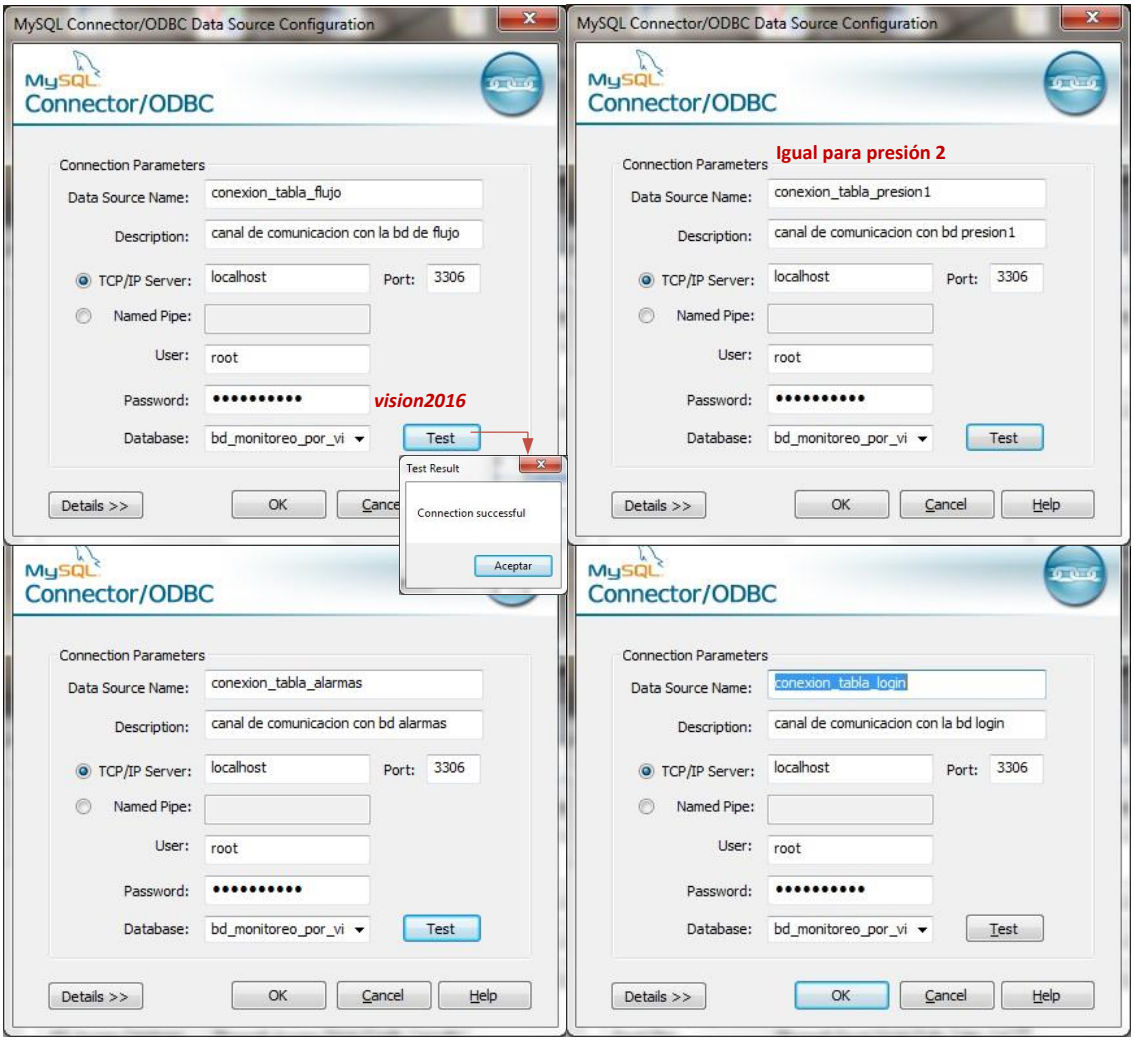


Figura 86: Ventana para configurar el DSN y verificar la conexión con la base de datos (MySQL Connector/ODBC Data Source Configuration).

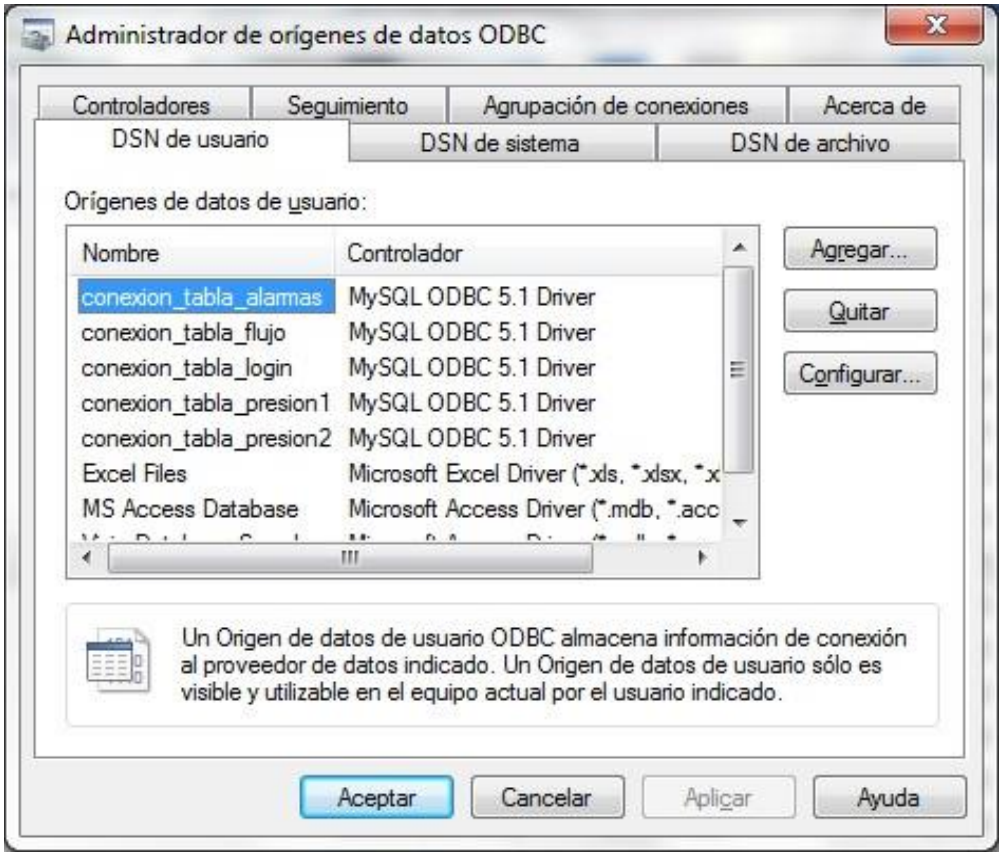


Figura 87: DSN agregados en la ventana Administrador de orígenes de datos ODBC en la pestaña DSN usuario.

3.7.4. Programación en LabVIEW para almacenar información en la base de datos

La programación grafica en LabVIEW para el almacenamiento de la información en la base de datos se organiza en distintos Sub*VIs* que serán llamados desde el programa principal donde se agregara la información correspondiente, para enviar información de las variables, los estados de las alarmas y los datos de los usuarios para el acceso al entorno remoto; se utiliza en esta aplicación la herramienta de código abierto LabSQL para tener acceso desde LabVIEW a la base de datos y registrar la información en las diferentes tablas de la misma, empleando lenguaje SQL, dicha herramienta necesita un

DSN para abrir la conexión con un Data Source especifico y posteriormente ejecutar un comando en la base de datos correspondiente.

Para agregar la herramienta LabSQL a la paleta de funciones de LabVIEW, se descarga la carpeta comprimida LabSQL-1.1a en el enlace http://jeffreytravis.com/lost/download/; se extraen los archivos de la misma; se pega la carpeta LabSQL ADO functions dentro de la carpeta user.lib (en la dirección C:\Program Files (x86)\National Instruments\LabVIEW 2013); se abre una aplicación en LabVIEW, en el Diagrama de Bloques, se selecciona la pestaña View y la opción Function Palette, se pulsa en Customize, se elige la opción Change Visible Palettes, se activa la categoría User Libraries y se pulsa *OK,* al reiniciar LabVIEW las funciones LabSQL se encuentran dentro la categoría activada, en las Figuras 88 y 89 se indican lo descrito anteriormente.

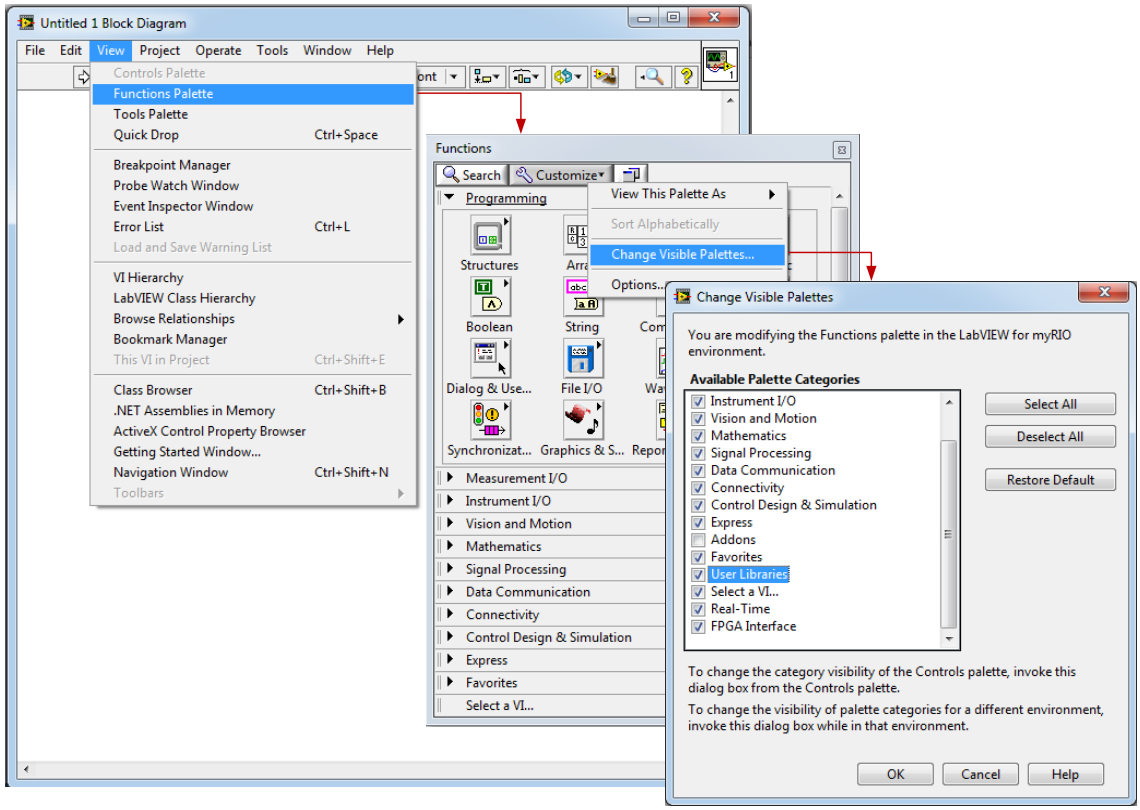


Figura 88: Habilitación de la categoría User Libraries.

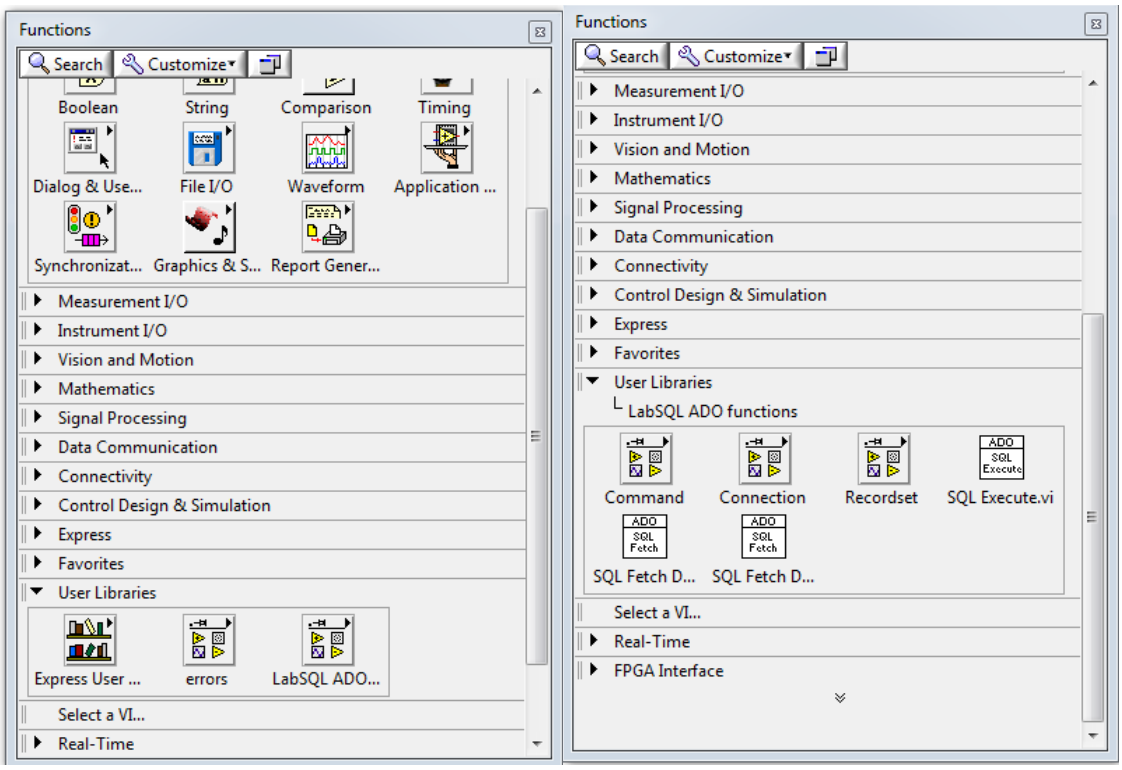


Figura 89: Conjunto de funciones LabSQL dentro de la categoría User Libraries.

3.7.5. Programación para el registro de datos de la variable flujo

El SubVI para el registro de datos en la tabla *datos de flujo*, consta de dos entornos, el Panel Frontal y el Diagrama de Bloques, en el primero se agregan tres controles numéricos, para el registro del número de identificación, la lectura de la variable en litros y galones (id, flujo ltr, flujo gal), como se puede observar en la Figura 90, y en el otro se realiza la programación empleando la función SQL Execute de la herramienta LabSQL, misma que puede ejecutar un comando en la base de datos, en este caso se insertará un registro en la tabla, el icono de SQL Execute consta de varios terminales de los cuales se ocupa, el terminal *Command Text* para asignar la instrucción en lenguaje SQL, la estructura de la instrucción es la siguiente:

INSERT INTO nombre de la tabla (nombre de las columnas de la estructura de la tabla, …, ….) VALUES (‘datos que se desean insertar en las respectivas columnas’,’…’);

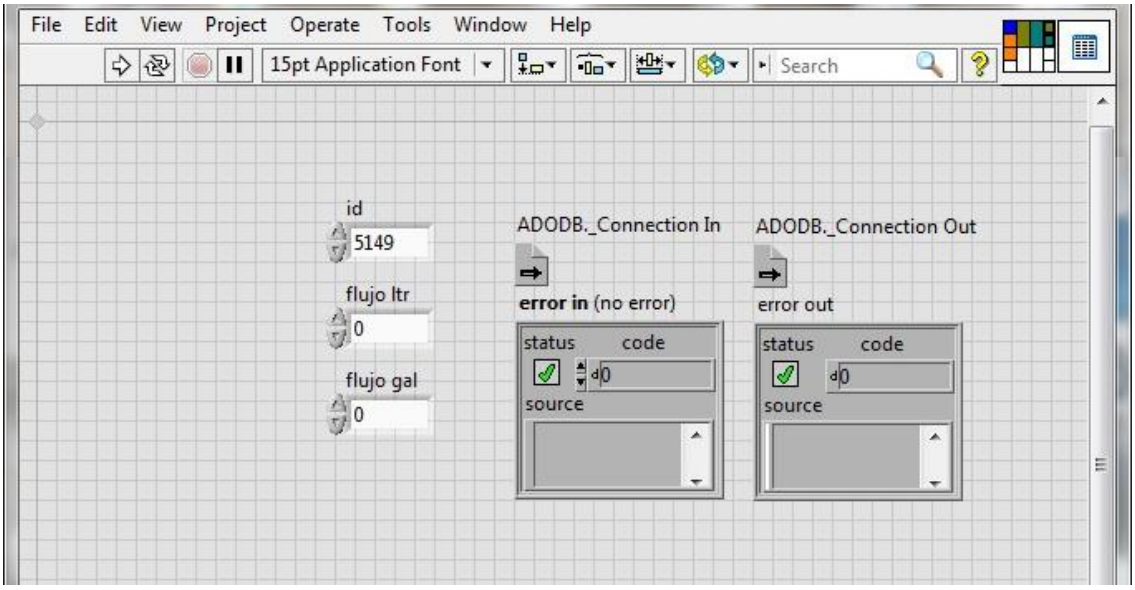


Figura 90: Panel Frontal del SubVI para el registro de datos en la tabla datos de flujo.

En un String Constant se escribe la instrucción hasta VALUES (‘, pues esto se mantiene constante, contrario a los datos del registro que son los que varían, para concatenar los datos actualizados a la cadena de caracteres, se utiliza Concatenate String y se añade los datos manteniendo el formato ‘dato1’,’dato2’,’…’);. Es importante mencionar que los datos son de tipo Double y el Concatenate String solo admite información tipo String por lo que se convierte con el Number To Fractional String, para la fecha y hora se inserta la función Get Data/Time String y en el terminal data format(0) se crea una constante y se elige el short (formato corto), en la Figura 91 se indica la instrucción definida para esta función creada (VI).

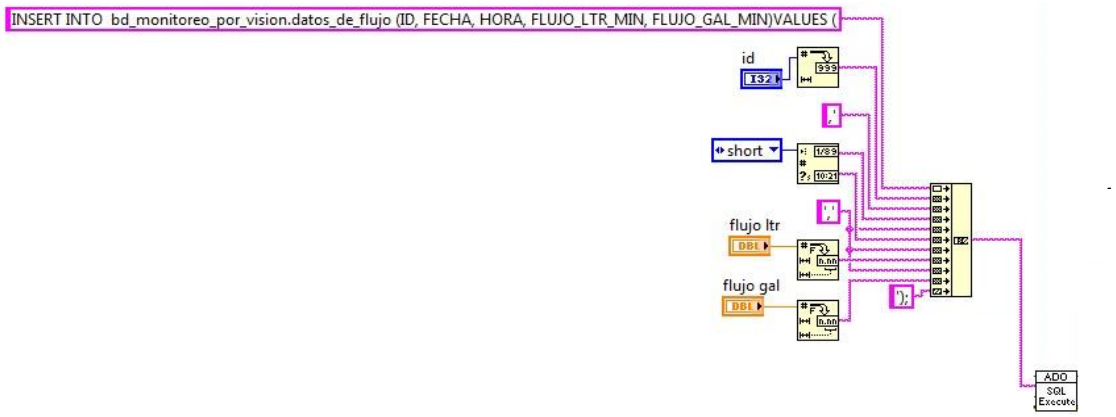


Figura 91: Instrucción para el registro de datos en la tabla datos de flujo.

Los otros terminales que se ocupan son: el Return Data que permite habilita o deshabilitar (TRUE o FALSE) el dato de retorno de la consulta, en esta aplicación se lo considera falso, para el ADODB._Connection in y el error in se crea un control, mientras que para ADODB._Connection out y el error out se crea indicadores, como se indica la Figura 92, se realiza de esta forma, pues el VI será llamado desde el programa principal (SubVI) donde se le unirá con otras funciones LabSQL y se le asignará la información correspondiente.

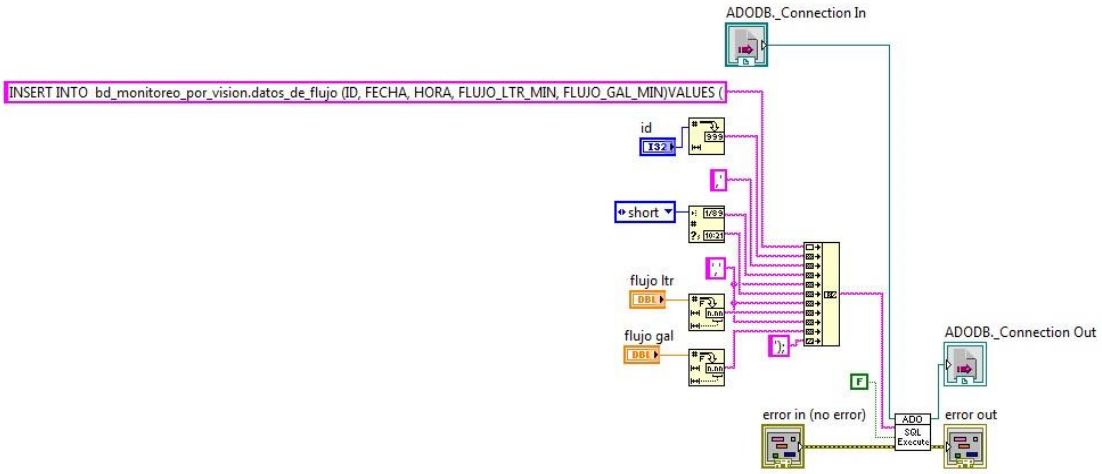


Figura 92: Programación para el registro de datos en la tabla datos de flujo.

Para crear un icono que identifique al SubVI con terminales para agregar la información correspondiente, contenida en el programa principal; se debe asignar

los controles e indicadores en la rejilla de conexión en el Panel Frontal, posteriormente para editar el icono de identificación se da doble clic sobre el recuadro que se ubica en la parte superior derecha junto a la rejilla de conexión, en la ventana que se abre se debe selecciona un gráfico que represente lo que se está realizando, este puede ser modificado con las herramientas del lado derecho, pulsar el botón *OK* para guardar los cambios; en la Figura 90 se indica el resultado.

3.7.6. Programación para el registro de datos de las variables presión

El SubVI para el registro de datos, tanto en las tablas datos de presión 1 como datos de presión 2, consta de dos entornos, el Panel Frontal y el Diagrama de Bloques, en el primero se agregan dos controles numéricos, para registrar el número de identificación y la lectura de la variable (id y presión psi), un control string para especificar la tabla del registro (nombre de la tabla); en el otro se realiza la programación empleando la función SQL Execute, manteniendo las funciones del anterior VI y la estructura de la instrucción con la información correspondiente; seguidamente se agregan los terminales y el icono de identificación siguiendo los pasos previos; se aprecia en las Figuras 93 y 94 el desarrollo del SubVI para el registro de la información.

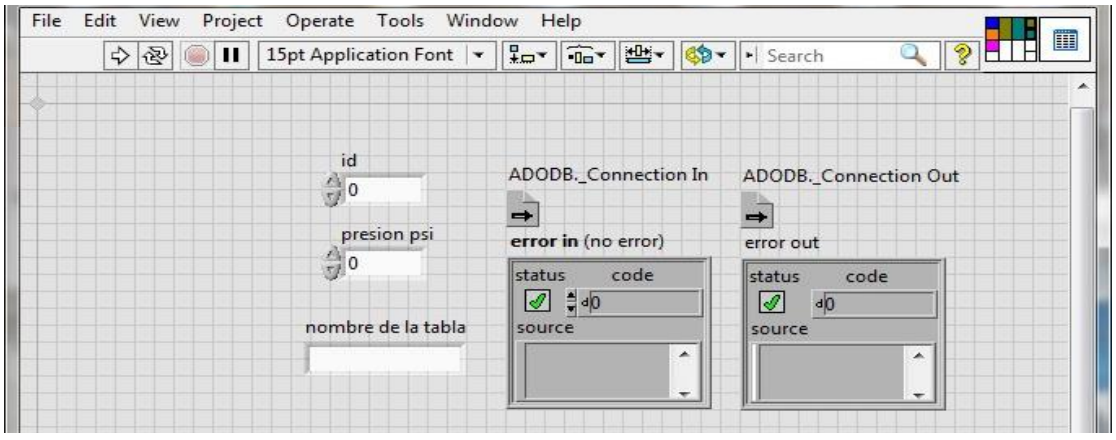


Figura 93: Panel Frontal del SubVI para el registro de datos en la tabla datos de presión 1 o presión 2.

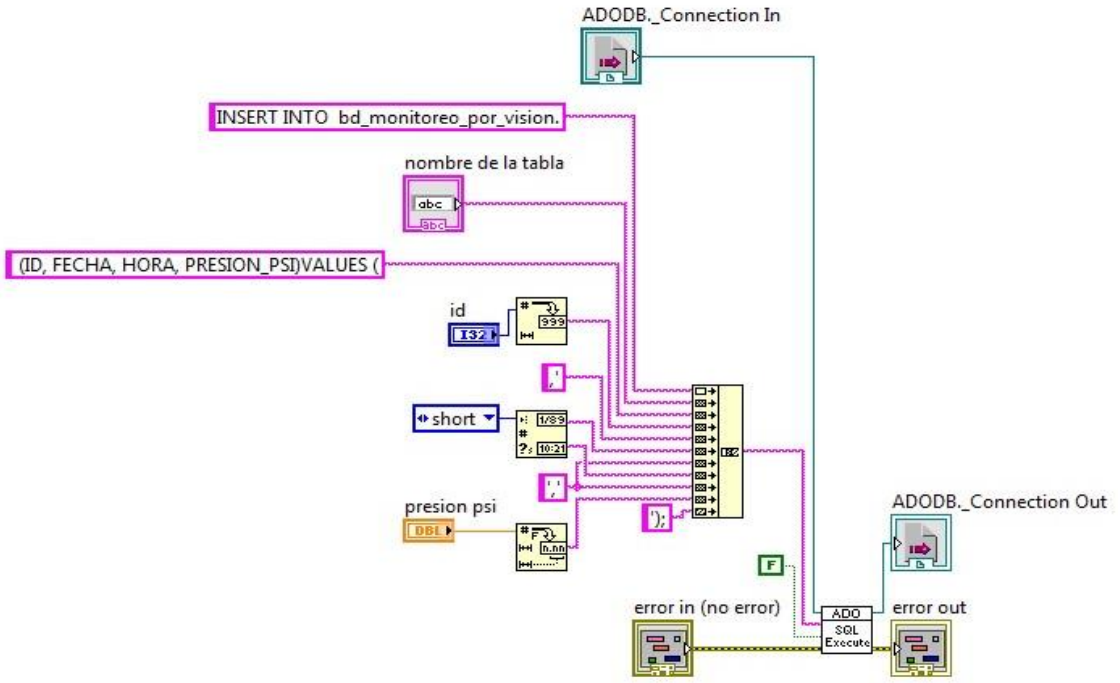


Figura 94: Programación para el registro de datos en la tabla datos de presión 1 o presión 2.

3.7.7. Programación para el registro de datos de los estados de alarma

Para el registro del estado de las alarmas se han creado tres SubVIs, estableciendo uno para el tipo High-High, el siguiente tanto para tipo High como Low y el otro para el tipo Low-Low, es importante mencionar que estos VIs sirven para registrar los estados de alarma de las diferentes variables (flujo, presión 1 y presión 2), por lo tanto, se crean con varios terminales para insertar la información correspondiente a cada variable, desde otro VI superior.

Para el SubVI que registra el estado de alarma High-High se agrega en el Panel Frontal dos controles numéricos, para el registro de la lectura de la variable y el limite High-High establecido por el usuario (Dato, Mayor que); dos controles string, para el registro de la unidad de la variable y la descripción de la alarma (unidad, Mensaje de alarma); un led indicador, para advertir que existe una situación anormal (alarma); en el Diagrama de Bloques se efectúa la programación, empleado la misma lógica de los SubVIs anteriores y la estructura

de la instrucción donde la información corresponde a la tabla registro de alarmas; se inserta Case Structure para que la alarma se registre solo cuando se active; de igual manera se asignan los terminales y el icono de identificación, en las Figuras 95 y 96 se indica lo mencionado.

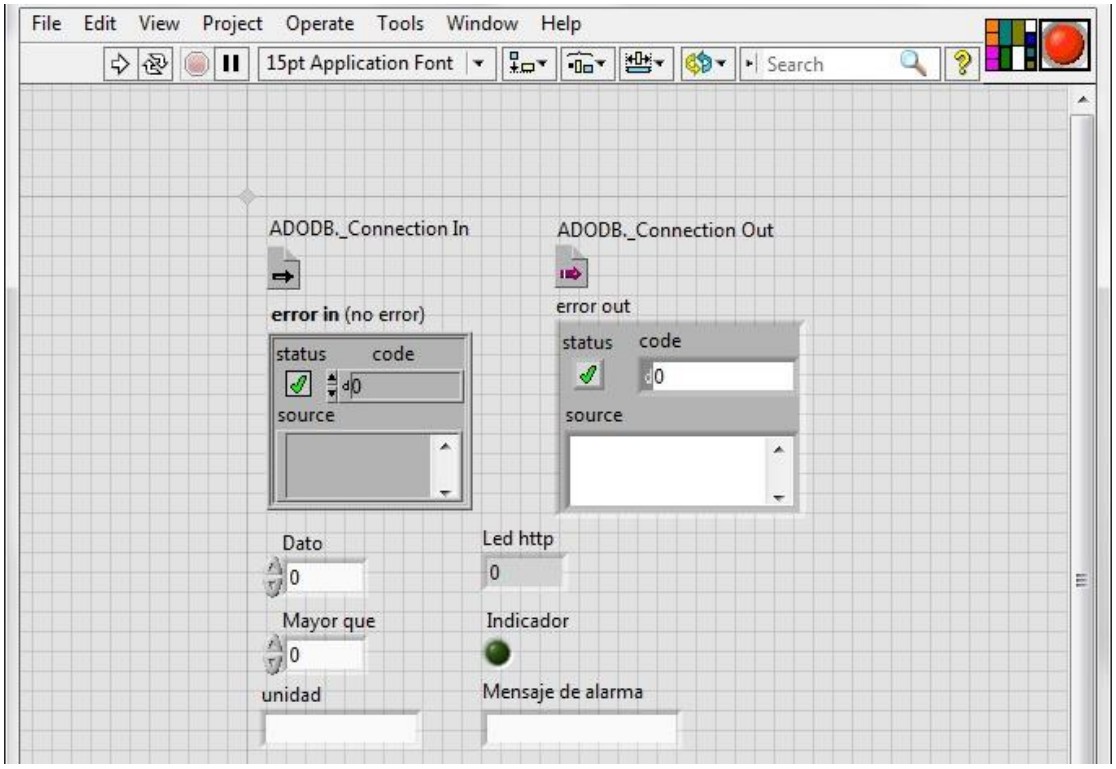


Figura 95: Panel Frontal del SubVI para el registro del estado de alarma High-High.

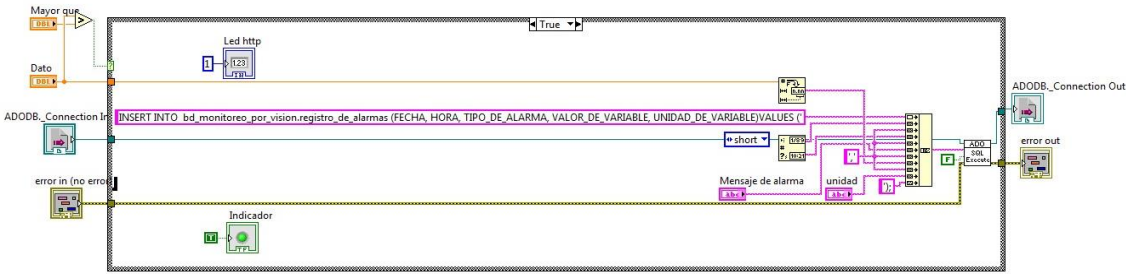


Figura 96: Programación para el registro del estado de alarma High-High.

Para el SubVI que registra el estado de alarma High o Low se agrega en el Panel Frontal tres controles numéricos, para el registro de la lectura de la variable

y para que el usuario establezca el rango en que será High o Low (Dato, Mayor que, Menor que); dos controles string, para el registro de la unidad de la variable y la descripción de la alarma (unidad, Mensaje de alarma); un led indicador, para advertir que existe una situación anormal (alarma); en el Diagrama de Bloques se efectúa la programación, empleando la misma lógica para realizar un registro donde la información para la instrucción sigue siendo la de la tabla registro de alarmas; se inserta Case Structure con el mismo fin que para la alarma High-High; de igual manera se asignan los terminales y el icono de identificación, en las Figuras 97 y 98 se indica lo mencionado.

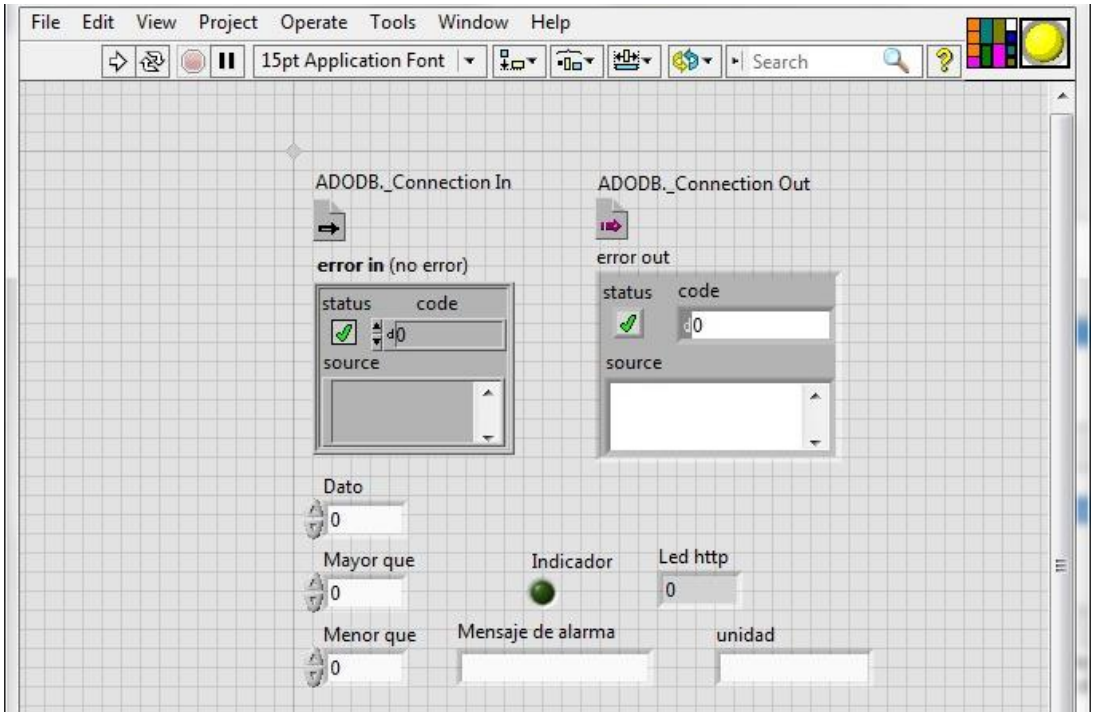


Figura 97: Panel Frontal del SubVI para el registro de los estados de alarma High o Low.

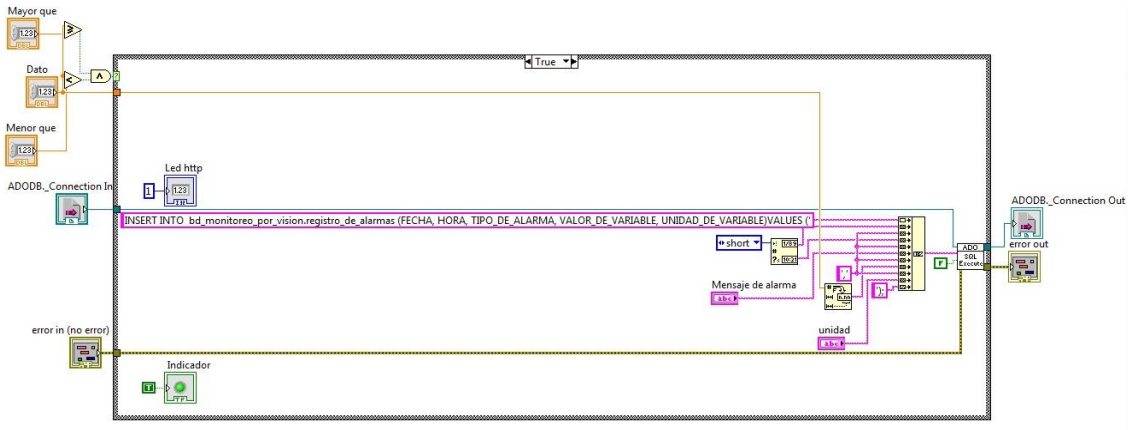


Figura 98: Programación para el registro de los estados de alarma High o Low.

Para el SubVI que registra el estado de alarma Low-Low, en el Panel Frontal se agregan dos controles numéricos, para el registro de la lectura de la variable y para que el usuario establezca el limite Low-Low (Dato, Menor que); dos controles string, para el registro de la unidad de la variable y la descripción de la alarma (unidad, Mensaje de alarma); un led indicador, para advertir que existe una situación anormal (alarma); en el Diagrama de Bloques se efectúa la programación, empleando la misma lógica que en el SubVI anterior donde la información para la instrucción sigue siendo la de la tabla registro de alarmas; se inserta Case Structure con el mismo fin que para la alarma High-High; de igual manera se asignan los terminales y el icono de identificación, en las Figuras 99 y 100 se indica lo mencionado.

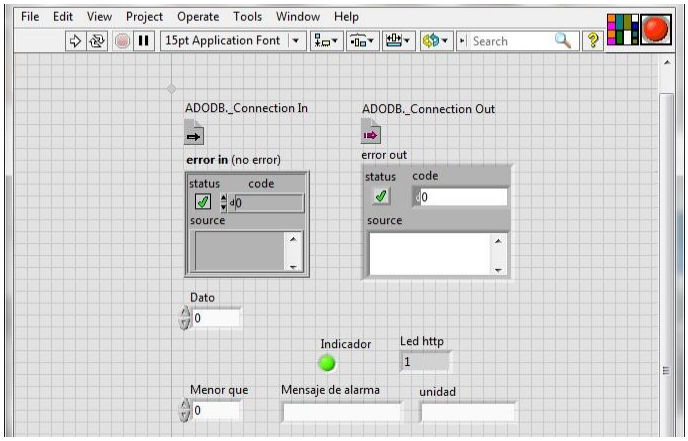


Figura 99: Panel Frontal para el registro de los estados de alarma Low-Low.

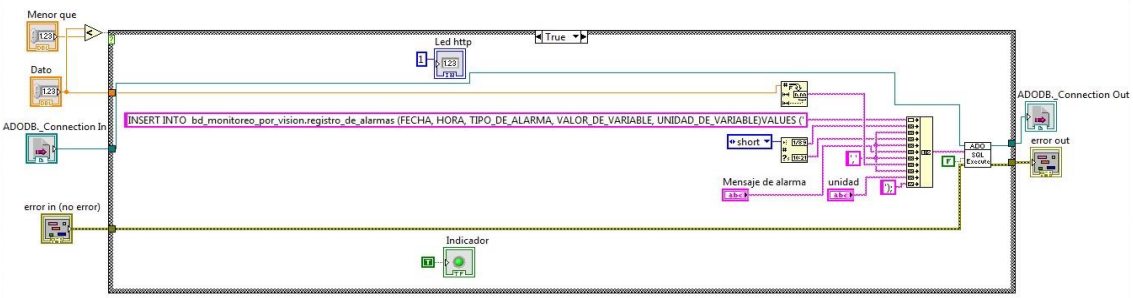


Figura 100: Programación para el registro del estado de alarma Low-Low.

Para que la programación del programa principal no se aglomere, se agrupa en un SubVI los estados de alarma de las diferentes variables, empleando los SubVIs creados para este fin; los terminales que posee el nuevo SubVI para la conexión entre funciones son: el ADODB._Connection y el error, además cuenta con doce terminales de entrada y salida que permiten fijar los límites de los estados de alarma de las diferentes variables y visualizar los mismos desde el HMI, respectivamente; en la programación por cada variable se agregan cuatro SubVIs, estado High-High, estado Low-Low y dos veces estado High o Low, se asigna la información correspondiente a cada SubVI, empleando variables globales para el dato de la variable y los límites de alarma, pues, se emplean en varias rutinas que operan en distintos programas, en las Figuras 101 y 102 se indica lo descrito.

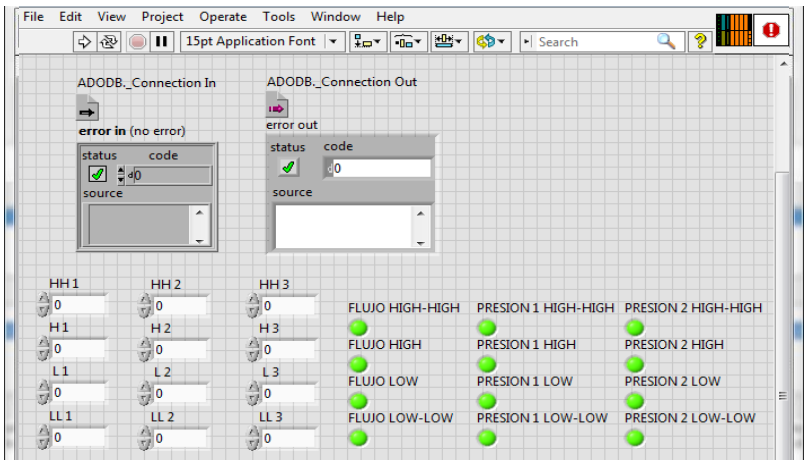


Figura 101: Panel Frontal del SubVI que agrupa los estados de alarma de las diferentes variables.

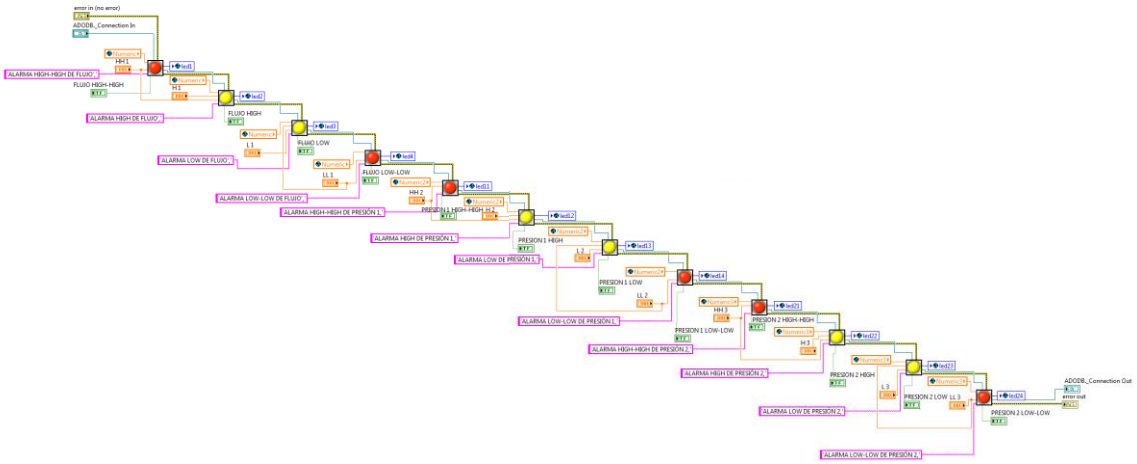


Figura 102: Diagrama de Bloques del SubVI que agrupa los estados de alarma de las diferentes variables.

3.7.8. Programación para el registro de datos de los usuarios

Para el SubVI que realiza el registro de los usuarios en la tabla log in, se agrega en el Panel Frontal dos controles string, para el nombre del usuario y la respectiva clave (usuario y password), como se puede observar en la Figura 103, y en el Diagrama de Bloques se realiza la programación empleando las funciones SQL Execute y Concatenate String, manteniendo la misma estructura de la instrucción empleando la información correspondiente, posteriormente se añaden los terminales y el icono de identificación de igual forma que el anterior, en la Figura 104 se puede observar la programación.

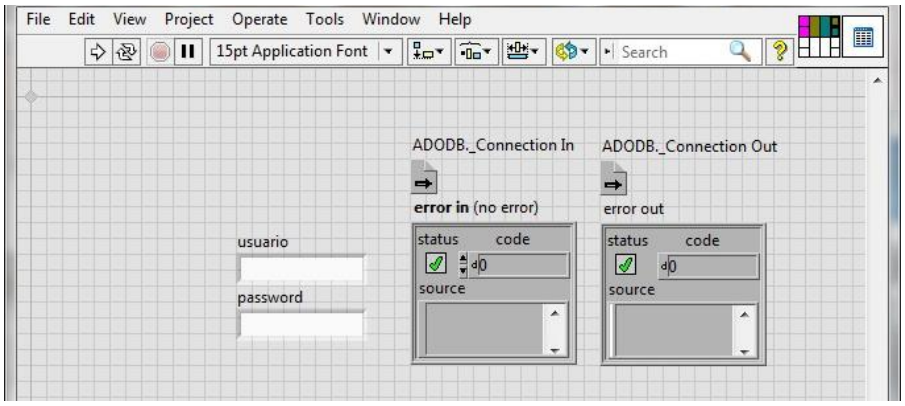


Figura 103: Panel Frontal del SubVI para el registro de datos en la tabla log in.

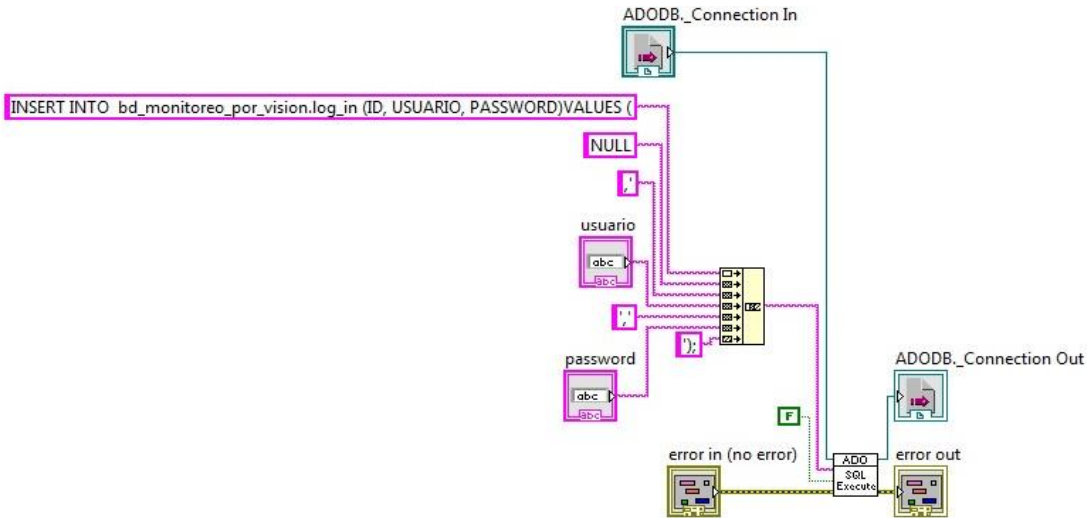


Figura 104: Programación para el registro de datos en la tabla log in.

3.7.9. Programación del VI principal para el registro de información en la base de datos

El registro en la base de datos se ejecuta desde el programa principal, cuya programación emplea, la estructura while-loop, para que la información se registre todo el tiempo hasta que el operador pulse STOP desde el HMI; la función ADO Connection Create, para que devuelva un puntero temporal (ADODB._Conecction) y LabVIEW asigne memoria asociada a dicho puntero; la función ADO Connection Open, para conectarse con el Data Source Name especificado, por lo que se proporciona a la función el nombre del DSN creado; los SubVIs creados con la información necesaria, para registrar la información obtenida del sistema de visión artificial en la base de datos; la función ADO Connection Close, para cerrar la comunicación con el Data Source Name especificado, la función ADO Connection Destroy, para liberar el espacio de memoria y emplearla en otros recursos; la función Time, para establecer desde el HMI cada que tiempo se efectuará un registro en las diferentes tablas de la base de datos.

A continuación se presenta la programación del programa principal, para registrar la información correspondiente a la variable flujo, como se indica en la

Figura 105; para registrar la información de las variables presión, como se indica en la Figura 106; para registrar los estados de alarma de las diferentes variables, como se observa en la Figura 107; para registrar la información de los usuarios, como se observa en la Figura 108.

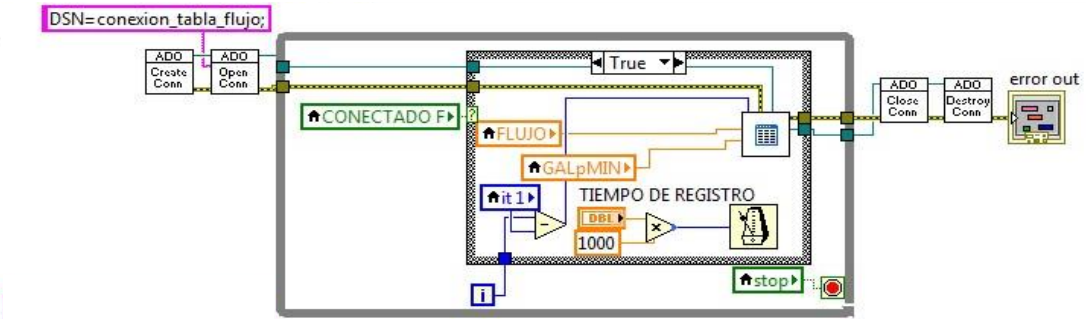


Figura 105: Programación para el registro de información de la variable flujo.

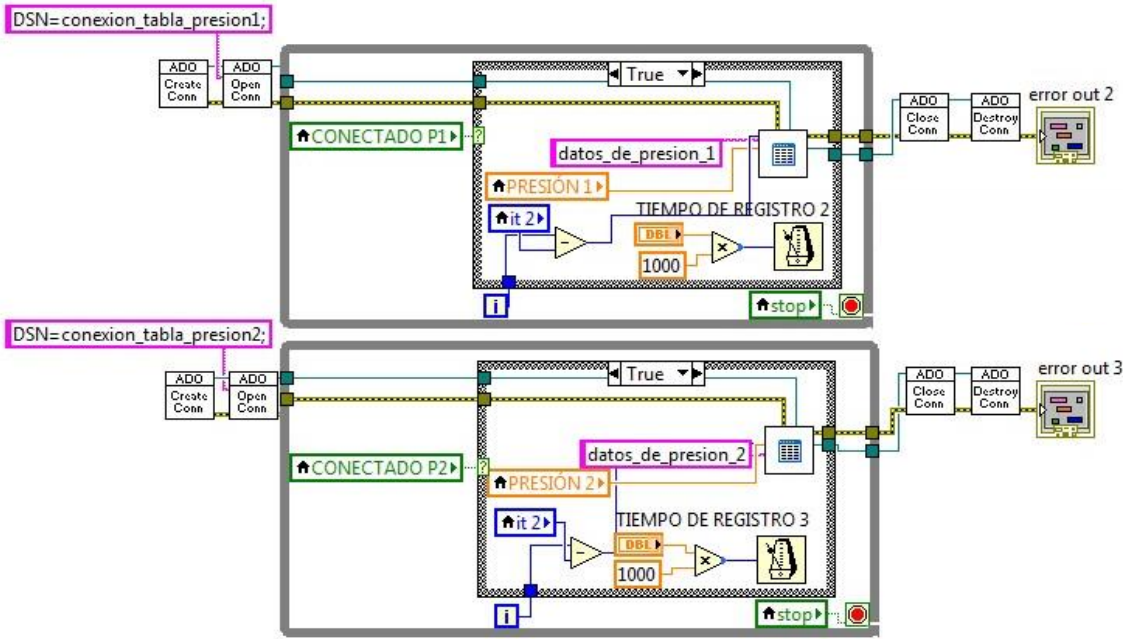


Figura 106: Programación para el registro de información de las variables presión.

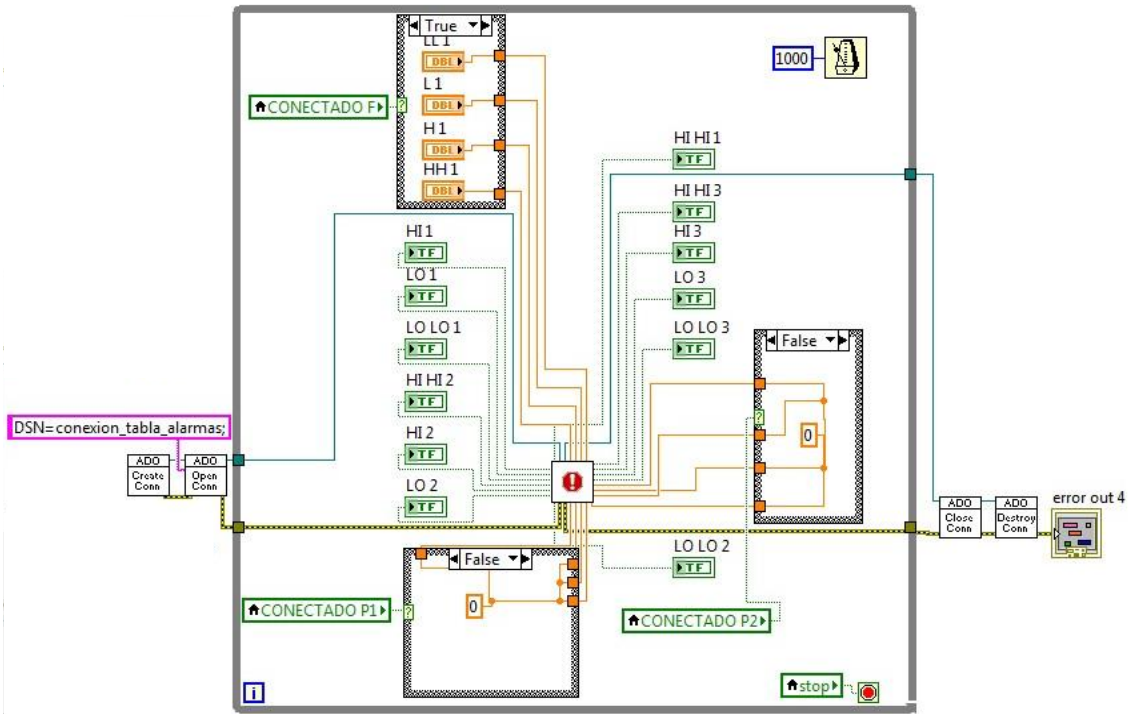


Figura 107: Programación para el registro de los estados de alarma de las diferentes variables.

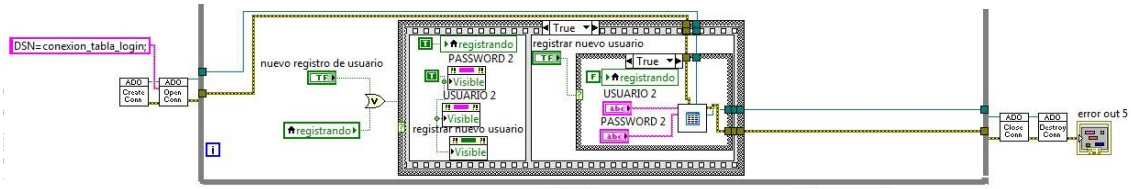


Figura 108: Programación para el registro de información de los usuarios.

3.7.10. Manejo de la información almacenada en la base de datos

Para visualizar los datos almacenados en la base de datos MySQL se accede al entorno phpMyAdmin, se proporciona la información en el cuadro de dialogo de identificación, se selecciona la tabla de la cual se desea visualizar los datos (dicha tabla se encuentra dentro de la base de datos), la información se presentara dentro de la pestaña Examinar y en forma de lista, como se indica en la Figura 109; es importante mencionar que únicamente en la lista de datos de la tabla log in se presentan las opciones para modificar individualmente los

registros, como se indica en la Figura 110, esto se debe a las configuraciones mencionadas al crear la tabla.

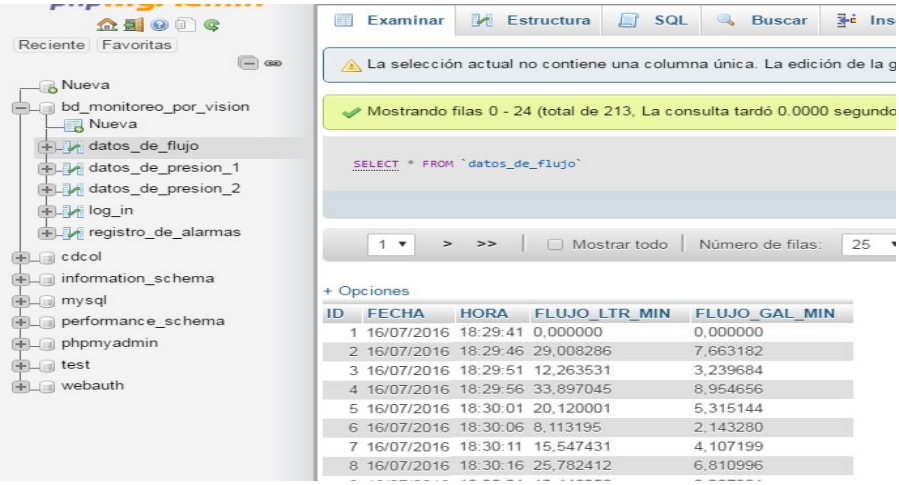


Figura 109: Visualización de los datos registrados en la tabla datos de flujo.

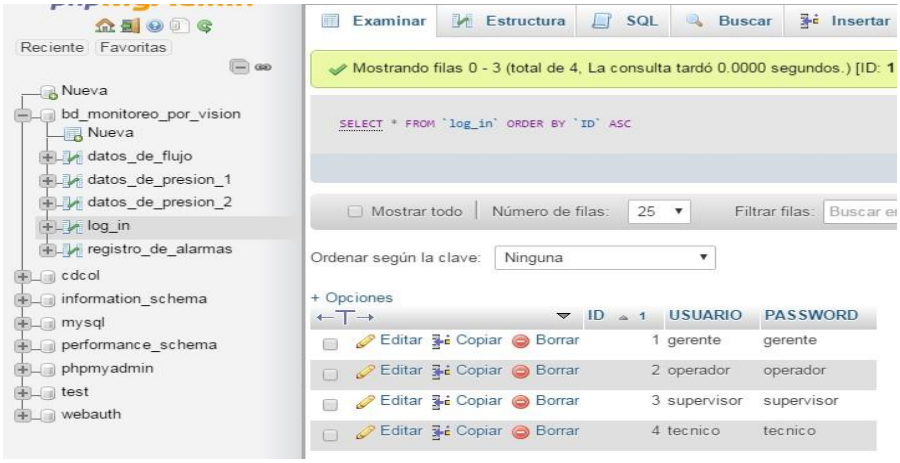


Figura 110: Visualización de los datos registrados en la tabla log in.

Para exportar la lista de datos registrados, se selecciona la tabla de la cual se requiere extraer la información, se da clic izquierdo sobre la pestaña Exportar, posteriormente se elige el modo de exportación que puede ser rápido o personalizado (el personalizado presenta varias opciones configurables para la exportación y el rápido muestra el mínimo de opciones configurables), finalmente se escoge el formato en el que se exportará los datos (PDF, Excel) y se pulsa sobre el botón Continuar, como se indica en la Figura 111.

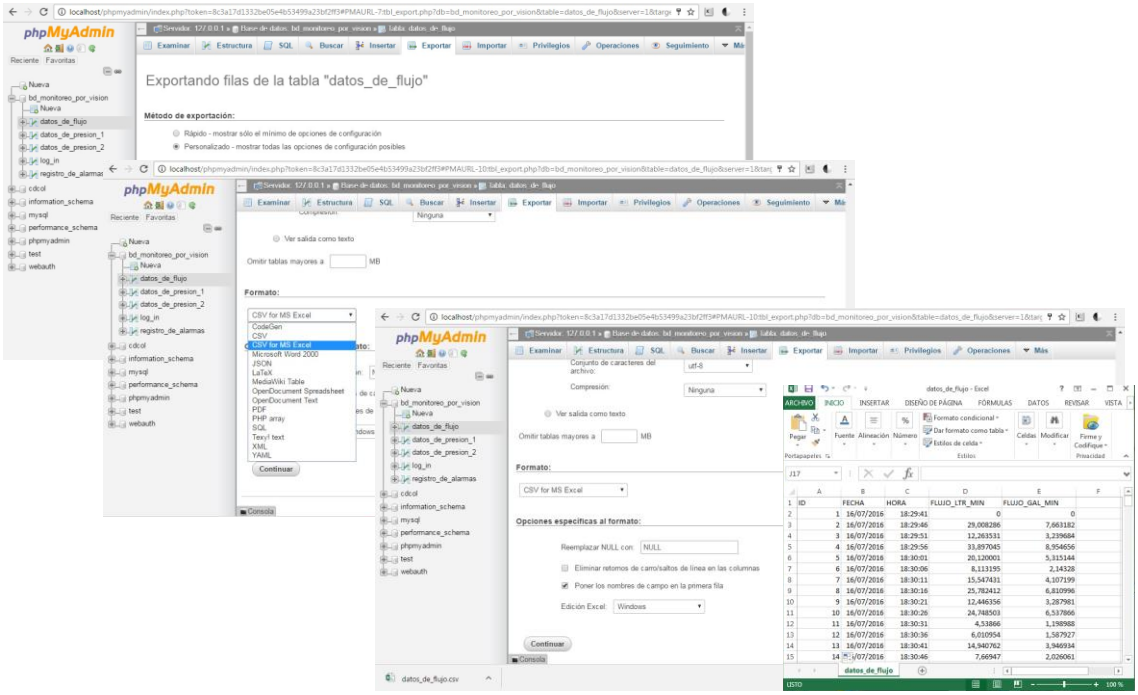


Figura 111: Proceso para exportar los datos registrados en la tabla datos de flujo a un archivo en formato CSV for MS Excel.

Para borrar la lista de información registrada en una tablas de la base de datos, se selecciona la tabla de la cual se requiere vaciar el contenido, se da clic izquierdo sobre la pestaña Operaciones, se escoge la opción Vaciar la tabla (TRUNCATE) y en el cuadro de dialogo se confirma la solicitud, como se indica en la Figura 112.

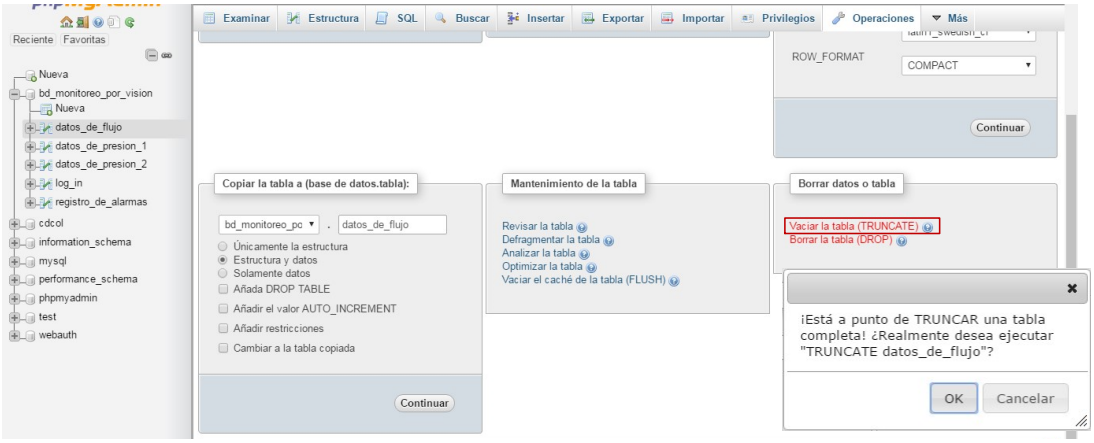


Figura 112: Proceso para borrar los datos registrados en la tabla datos de flujo.

3.8. Desarrollo del sitio web de monitoreo remoto

Como complemento del sistema de monitoreo se desarrolla un sitio web (aplicación visual remota compuesta por páginas web) que consta de una etapa inicial de seguridad de acceso en donde el usuario debe identificarse para visualizar la información correspondiente a las distintas variables; es importante mencionar que varios usuarios podrán acceder simultáneamente al mismo, por lo que se requiere que el equipo se encuentre conectado a la red del sistema de monitoreo (SSID: VISION) y contenga un navegador web, es decir, se puede acceder al sitio web desde un smart phone, una tablet o un ordenador.

Para el desarrollo de las páginas web se emplea la especificación de HTML5; que utiliza el lenguaje de etiquetas HTML para estructurar la página web, la tecnología CSS para diseñar el estilo de presentación al usuario, y el lenguaje de programación Javascript para realizar los efectos dinámicos sobre dicha página web; logrando de esta manera que el entorno que se presente a los distintos usuarios sea amigable e intuitivo. Cabe mencionar que para facilitar el desarrollo del entorno gráfico de las páginas web la codificación se realiza en el software Sublime Text, pues, es un editor de código fuente sofisticado.

3.8.1. Estructuración del entorno gráfico de la página web de monitoreo

Se realiza inicialmente una representación gráfica de la estructura del cuerpo del entorno de monitoreo con el fin de establecer la organización de la información que se presentará al usuario; la estructura se divide en cinco áreas, como se indica en la Figura 113, que satisfacen las necesidades de este trabajo; de la siguiente manera, en el área superior se presentan varios encabezados que rotan secuencialmente de forma automática, junto a esta área se encuentra la barra de navegación que contiene las opciones: refrescar la página, acceder a la base de datos y salir; en el área central lado izquierdo se muestra la información de las variables con el respectivo diagrama del proceso; en el área central lado derecho se visualizan los estados de alarma de las distintas variables; y en el área inferior se indica la información de los autores.

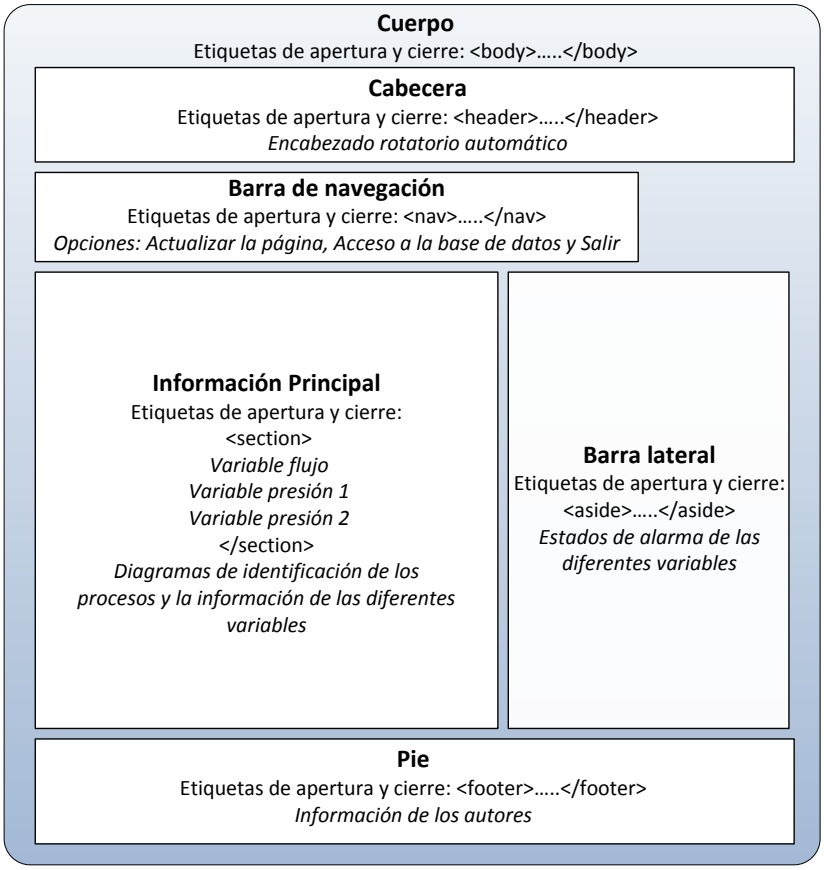


Figura 113: Representación estructural del página web de monitoreo

3.8.2. Creación la página web de monitoreo

Para desarrollar la página web de monitoreo, primeramente dentro del proyecto realizado en LabVIEW se crea una carpeta (contenido_público) que se utilizará para implementar el servicio web, en la sección 3.8.3. Dicha carpeta contendrá los archivos que forman parte de la página web de monitoreo en los cuales se desarrollará la interfaz gráfica de la misma (documento HTML, archivos de estilo, de imágenes y de efectos dinámicos).

Para empezar a codificar se crea un nuevo archivo en la interfaz Sublime Text y se guarda con la extensión .html en la capeta contenido_público, posteriormente se digita la primera línea de código empleando el elemento <DOCTYPE> para declarar el tipo de documento que se está desarrollando (html); se crea la estructura HTML utilizando el elemento <html>; se agrega el

atributo lang dentro de la etiqueta de apertura <html> para asignar el idioma al contenido (español: es); dentro de las etiquetas <html> … </html> se usan los elementos: <head> y <body> para especificar el título de la página web, el mismo que se muestra en la pestaña del navegador, y para desarrollar el entorno de monitoreo a partir de la representación gráfica realizada. A continuación se indica el código para este propósito.

```
<!DOCTYPE html>
<html lang="es"> <!--etiqueta de apertura-->
<head>
  <title>MONITOREO POR VISI&Oacute;N ARTIFICIAL</title> <!--tilde en la O de VISIÓN-->
</head>
<body><!--Estructura del cuerpo del entorno de monitoreo--></body>
</html> <!--etiqueta de cierre-->
```

Para este trabajo se crea un nuevo archivo con la extensión .css en la interfaz Sublime Text dentro de la carpeta contenido_público, el cual contendrá todos los estilos de la página web de monitoreo. Para incorporar el archivo al documento HTML se utiliza el elemento <*link*> con los atributos *rel="stylesheet"* (relación que tiene el documento con el archivo) y *href=" file:///C:/monitoreo/contenido_privado/css/estilos.css"* (dirección del archivo); el valor de la relación (*stylesheet*) indica al navegador que el archivo *estilos.css* contiene el diseño de presentación de la página web. A continuación se presenta el código de lo descrito anteriormente.

```
<head>
  <title>MONITOREO POR VISI&Oacute;N ARTIFICIAL</title>
  <link rel= "stylesheet" href="css/estilos.css">
</head>
```

Las primeras reglas que se declaran en el archivo de estilos (CSS) se muestran en el código que se indica a continuación, estas permiten personalizar el margen tanto interno como externo de todo el entorno gráfico y establecer jerarquías para títulos y subtítulos utilizando los elementos <*h1*>, <*h2*> y <*h3*> de HTML5 que se agregarán dentro de las etiquetas <*body*>; con respecto a los márgenes se requiere utilizar y aprovechar todo el espacio de la pantalla, por lo que el valor de estas propiedades es cero pixeles; y en relación a los elementos

HTML5 de jerarquía se establecen diferentes estilos, asignándoles valores a los parámetros como: el grosor, el tamaño y el tipo de letra.

```
* {margin: 0px;  padding: 0px;} /*personalizar el margen interno y externo */
h1  {font: bold 25px serif;} /*para títulos principales de las diferentes áreas*/
h2  {font: bold 20px serif;} /*para subtítulos en las diferentes áreas*/
h3  {font: bold 12px serif;} /*para títulos e información en el área –pie-*/
```

Continuando con la programación se crea dentro de las etiquetas *<body>* la estructura del entorno de monitoreo basándose en la representación gráfica realizada anteriormente, para lo cual se utilizan varios elementos entre ellos los de jerarquía y los elementos HTML5 que se observó en la Figura 113 (*<header>, <nav>, <section>, <aside> y <footer>).*

Para agregar al documento el estilo del cuerpo se utiliza el método de *referencia por atributo id*, ya que únicamente se añadirá dicho estilo al elemento <body> (el valor del atributo id no debe repetirse en todo el documento). En el archivo de estilos para añadir una regla usando este método se antepone el símbolo numeral al valor del atributo id (*#cuerpo*), posteriormente dentro de la misma se agregan las propiedades para configurar el estilo del cuerpo, como el recuadro que agrupa todas las áreas del entorno de monitoreo, el margen interno, la alineación del texto y el color de fondo. En el documento de la estructura se añade el estilo insertando el atributo id dentro de la etiqueta de apertura *<body>* (*<body id="cuerpo">*). A continuación se presenta el código HTML y CSS de lo antes mencionado.

```
<body id="cuerpo">
  <header> </header>
  <nav> </nav>
  <br> <!--salto de línea-->
  <section>
    <h1>SISTEMA DE MONITOREO A TRAV&Eacute;S DE VISI&Oacute;N ARTIFICIAL</h1>
  </section>
  <br> <br> <br> <br>
  <aside>
    <h1>ALARMAS DEL PROCESO:</h1> <br>
    <h2><u>FLUJO</u></h2> <!--<u> agrega el estilo subrayado al texto-->
    <h2><u>PRESI&Oacute;N 1</u></h2>
    <h2><u>PRESI&Oacute;N 2</u></h2>
  </aside>
```

```
  <footer> </footer>
</body>
```

```
#cuerpo {
  border: 1px solid #000000; /*propiedad para definir el grosor, el estilo y el color del borde*/
  border-radius: 20px; /*propiedad para redondear las esquinas del borde*/
  padding: 10px; /*propiedad para el margen interior (horizontal-vertical)*/
  text-align: center; /*propiedad para centrar el texto del cuerpo*/
  /*propiedad para agregar al cuerpo un color de fondo degradado, en los diferentes navegadores*/
  /*navegador Safari y Chrome*/
  background: -webkit-linear-gradient(360deg, #01DFA5,#01DF74);
  /*navegador Mozilla Firefox*/
  background: -moz-linear-gradient(top, #FFFFFF, #006699);
  /*navegador Explorer*/
  background: -ms-linear-gradient(top, #FFFFFF, #006699);
  /*navegador Opera*/
  background: -o-linear-gradient(top, #01DFA5,#01DF74);
 /*valor del color en hexadecimal #ROJO VERDE AZUL de -00 A FF- para cada color*/
 }
```

Para desarrollar una cabecera que contenga varias imágenes que roten secuencialmente de forma automática, dentro de las etiquetas <*header*> se añaden los elementos para crear una lista de imágenes; posteriormente en el archivo de estilos se declaran reglas utilizando el método mencionado anteriormente (*#cabecera*), y otras empleando aparte del valor cabecera el nombre del elemento, ya que se desea modificar únicamente el estilo de los elementos que se encuentran dentro del elemento <*header*>; seguidamente se añaden dentro de dichas reglas las propiedades para establecer el estilo de la cabecera.

Además se inserta una regla utilizando el método de *referencia por atributo class*, pues, se va aplicar el mismo estilo a varios elementos del documento (el valor del atributo class se puede utilizar en varios elementos del documento), para crear una regla usando este método se antepone un punto al valor del atributo class (*.imagen*), posteriormente se agregan las propiedades dentro de la misma. En el documento de la estructura se añaden los estilos insertando dentro de las etiqueta de apertura <*header*> el atributo id con el valor cabecera y dentro de las etiquetas <*img*> el atributo class con el valor imagen. A continuación se muestra el código de la estructura y el estilo que contiene el detalle de las propiedades añadidas.

```
<header id="cabecera">
   <ul> <!--<ul> <li> elementos para crear una lista con viñetas-->
      <!-- <img src="dirección de la imagen"> elemento para insertar una imagen-->
       <li><img class="imagen" src="img/SELLO.png"></li>
       <li><img class="imagen" src="img/SELLO_ ELEC.png"></li>
       <li><img class="imagen" src="img/SELLO_PROYECTO.png"></li>
   </ul>
</header>
```

```
#cabecera {border: 1px solid #000000;
  overflow: hidden; /*propiedad para recortar y ocultar el contenido que se sobrepasa el borde del cuerpo)*/}
#cabecera ul {width: 300%;
  display: flex; /*propiedad para ubicar los ítems de la lista uno a continuación del otro horizontalmente*/
 /*animación: nombre-de-la-regla duración-de-la-animación función-de-la-animación-en-el-tiempo*/
  animation: rotar 20s infinite alternate linear; /*propiedad de animación para que las imágenes roten
   secuencialmente e indefinidamente*/}
#cabecera li {list-style: none; /*propiedad para quitar el estilo a la lista (sin viñetas)*/}
/*regla para que el navegador considere a las imágenes de la cabecera como bloques, uno debajo de otro sin espacios*/
#cabecera img {display: block;}
/*regla para que las imágenes de los elementos con el atribulo class="imagen" puedan acoplarse a las dimensiones de la
ventana del navegador*/
.imagen {max-width: 100%; height: auto;}
/*regla de animación donde se define que cada imagen permanecerá un tiempo en el entorno gráfico*/
@keyframes rotar {
  /*el porcentaje sirve para definir que irá sucediendo en la animación mientras avanza*/
  0% {margin-left: 0; /*propiedad para establecer el valor del margen externo izquierdo*/}
  30% {margin-left: 0;}
  35% {margin-left: -100%;} 65% {margin-left: -100%;}
  70% {margin-left: -200%;} 100% {margin-left: -200%;}}
```

Para crear la barra de navegación, dentro de las etiquetas <*nav*> se añaden diferentes elementos para crear una lista que contenga hipervínculos, los mismos permitirán actualizar la página, acceder a la base de datos y salir. En el archivo de estilos se declaran varias reglas utilizando el método por atributo id, las mismas permiten establecer el estilo que tendrá la barra de navegación cuando sea manipulada, es decir al pasar el mouse sobre las diferentes opciones de la misma, y cuando no lo sea. A continuación se indica el código para este fin.

```
<nav id="barra_navegacion">
  <ul>
    <!--elemento <a> con el atributo href para crear hipervínculos con otras páginas web-->
    <!--<a href="dirección URL">titulo para insertar el hipervínculo</a>-->
    <li><a href="monitoreo.html">ACTUALIZAR P&Aacute;GINA</a></li>
    <li><a href="http://192.168.0.102:80/phpmyadmin/"  target="top">BASE DE DATOS</a></li>
    <!--atributo target="top" muestra la página web enlazada en toda la ventana del navegador destruyendo el frame
     principal-->
    <li><a href="http://192.168.0.102:80/monitoreo/login.html" target="_parent">SALIR</a></li>
  </ul>
</nav>
```

```
#barra_navegacion {
  background: #0B0B3B; /*propiedad para agregar un color de fondo (azul)*/
  width: 64%; /*propiedad para establecer el ancho de la barra de navegación*/
  text-align: left; /*propiedad para alinear el texto a la izquierda*/
  padding: 5px;}
#barra_navegacion ul {display: flex;}
#barra_navegacion li { list-style:none;}
#barra_navegacion a {
  display: inline-block; /*propiedad para que el navegador considere a cada hipervínculo como un bloque que bordea
   el contenido, y los ubique uno debajo de otro sin espacios*/
  text-decoration: none; /*propiedad para quitar el estilo del hipervínculo*/
  color: #FFFFFF; /*propiedad para establecer el color de la letra*/
  font: bold 14px serif; /*propiedad para configurar el estilo de la letra*/
  padding: 10px;}
/*regla para cambiar el estilo de una opción de la barra de navegación al pasar el mouse sobre la misma*/
#barra_navegacion a :hover {background: orange; border-radius: 3px; color:#0B0B3B; }
```

Para desarrollar la sección de información principal del entorno de monitoreo, dentro de las etiquetas *<section>* se insertan varios elementos, con el propósito de crear tres artículos (subsecciones), cada uno con una tabla que contenga en las diferentes casillas el nombre de la variable monitoreada, el valor de la lectura de la misma (lectura obtenida a través un script que se comunica con el servicio web de LabVIEW), y la imagen que describe el proceso en el que interviene la variable. En el archivo de estilos se declaran diferentes reglas empleando los métodos mencionados, en las cuales se diseña el estilo de la sección principal y el estilo del valor numérico empleando diferentes propiedades para este fin. A continuación se indica el código HTML y CSS que contiene el detalle de las propiedades.

```
<section id="inf_principal">
<h1>SISTEMA DE MONITOREO A TRAV&Eacute;S DE VISI&Oacute;N ARTIFICIAL</h1>
  <article> <!--elemento para crear un artículo-->
    <table width="100%"  border="1"  cellspacing="5"> <!--elemento para crear una tabla-->
      <tr> <!--<tr> <td> elementos para insertar filas y columnas en la tabla-->
        <td><h2>VARIABLE DE FLUJO</h2></td>
        <td id="indicador1" class="diseno_indicador">0</td>
        <td align="center"><h3>ltr/min</h3></td>
      </tr>
      <tr>
        <!--atributo para definir el número de columnas que se desea unir dentro de la fila-->
        <td colspan="3" align="center">
          <figure> <!--elemento que ayuda a identificar que el artículo contiene una ilustración-->
            <img class="imagen" src="img/EST_FLUJO.png">
            <!--<figcaption>elemento para agregar un título descriptivo relacionado con la ilustración -->
            <figcaption><p>DIAGRAMA P&ID DE FLUJO</p></figcaption>
          </figure>
```

```
      </td>
    </tr>
  </table>
 </article>
 <article><!--VARIABLE DE PRESIÓN 1--></article>
 <article><!--VARIABLE DE PRESIÓN 2--></article>
</section>
```

```
#inf_principal {width: 70%;
 float:left; /*propiedad para posicionar que la información principal al lado izquierdo */}
.diseno_indicador {
 width:140px; height:40px; /*propiedad para establecer el ancho y el alto del indicador*/
 padding: 5px; border: 2px solid lightgray; background: lightgray;
 font-size: 18px; /*propiedad para establecer el tamaño de la letra del indicador*/
 box-shadow:inset 2px 2px 5px black; /*propiedad para añadir un efecto de sombra desenfocada sobre recuadro
  del indicador*/}
article {padding: 20px 0px;} /*regla para definir el margen interno de los  elementos <article>*/
```

Para desarrollar la sección de alarmas (barra lateral), dentro de las etiquetas <*aside*> se insertan varios elementos, con el fin de crear tres subsecciones que contengan una tabla con varias casillas en las cuales se muestre, cuatro círculos (mediante el atributo *class="led"*) que representaren a los leds indicadores de alarmas (el led indicador se activa o desactiva dependiendo de la información que obtenga el script que se comunica con el servicio web de LabVIEW), y cuatro nombres que identifiquen a la alarma. En el archivo de estilos se declaran diferentes reglas empleando los métodos mencionados, en las cuales se diseña el estilo de la barra lateral y el estilo del led indicador empleando diferentes propiedades para este fin. A continuación se indica el código HTML y CSS que contiene el detalle de las propiedades.

```
<aside id="barra_lateral">
<h1>ALARMAS DEL PROCESO:</h1><br><h2><u>FLUJO</u></h2>
 <article>
  <table width="100%"  border="0"  cellspacing="10">
   <tr><td align="center"><p id="led1" class="led"></p></td>
    <td><h2><i>HIGH-HIGH</i></h2></td></tr>
   <tr><td align="center"><p id="led2" class="led"></p></td>
    <td><h2><i>HIGH</i></h2></td></tr>
   <tr><td align="center"><p id="led3" class="led"></p></td>
    <td><h2><i>LOW</i></h2></td></tr>
   <tr><td align="center"><p id="led4" class="led"></p></td>
    <td><h2><i>LOW-LOW</i></h2></td></tr>
  </table>
 </article>
```

```
<h2 class="separar"><u>PRESI&Oacute;N 1</u></h2>
  <article>  </article>
<h2 class="separar"><u>PRESI&Oacute;N 2</u></h2>
  <article>   </article>
</aside>
```

```
#barra_lateral {float: right; width: 29%; border: 1px solid #000000;
  background:linear-gradient (30deg, #FE9A2E, #F3E2A9);}
/*regla para establecer el estilo de la línea que separa los estados de alarma de las diferentes variables*/
.separar {padding: 20px 0px 0px 0px; border-top: 2px solid blue;}
/*regla para definir el estilo de los leds gráficos indicadores de alarma */
.led {height:30px; width:30px; border: 2px solid black; border-radius:17px;
  box-shadow:2px 2px 5px black;}
```

Los atributos id de color rojo que se observan en algunos elementos del documento html son para actualizar la información tanto de la lectura de la variable como de los estados de alarma de la misma, esto se realiza mediante un script que se comunica con el servicio web de LabVIEW, el cual se desarrolla en la sección 3.8.8; es importante mencionar que cada variable posee un atributo id para que el script pueda identificar a que elemento debe modificar la información. El código completo del desarrollo de la página web de monitoreo se indica en el Anexo E.

Finalmente para verificar el diseño de la página web de monitoreo se abre el documento creado con extensión html, el cual se ejecuta en un navegador web, en Figura 114 se puede apreciar el entorno de monitoreo conformado por las áreas definidas en la representación gráfica.

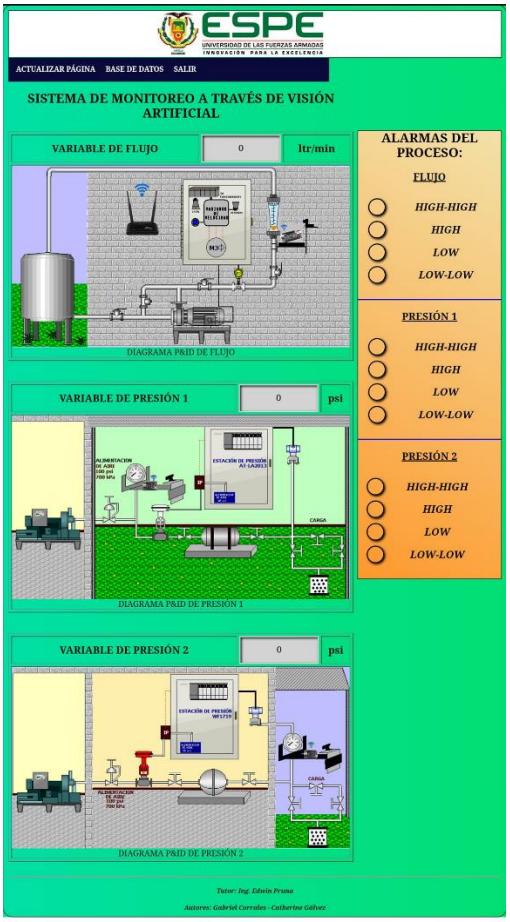


Figura 114: Entorno de la página web de monitoreo.

3.8.3. Creación de un servicio web de LabVIEW

Se desarrolla dentro del proyecto realizado en LabVIEW un servicio web con el fin de publicar la página web desarrollada anteriormente y actualizar la información de las variables para que los usuarios puedan monitorear el sistema en tiempo real de forma remota, para lo cual dentro del servicio web se agrega la carpeta contenido_público, y en el interior de la carpeta Web Resources se añade un nuevo recurso web dentro del mismo se crea un nuevo VI (método HTTP VI) donde se programa la funcionalidad del servicio web (transmitir la información de las variables a varios usuarios que accedan a la página web), al guardarlo se asigna automáticamente el método tipo GET, lo que quiere decir que este VI es el encargado de transmitir la información mediante la solicitud HTTP realizada al

servicio web, en un formato de salida tipo XML. En la Figura 115 se puede observar la programación del VI que contiene la funcionalidad del servicio web.

En este VI se asigna una variable global a cada indicador, de esta forma la información del VI principal puede ser utilizada en el VI de métodos HTTP, por lo tanto el script que se desarrolla a continuación, podrá proveer información actualizada a la página web de monitoreo de este modo los datos que se visualicen de forma local también se mostrarán de forma remota.

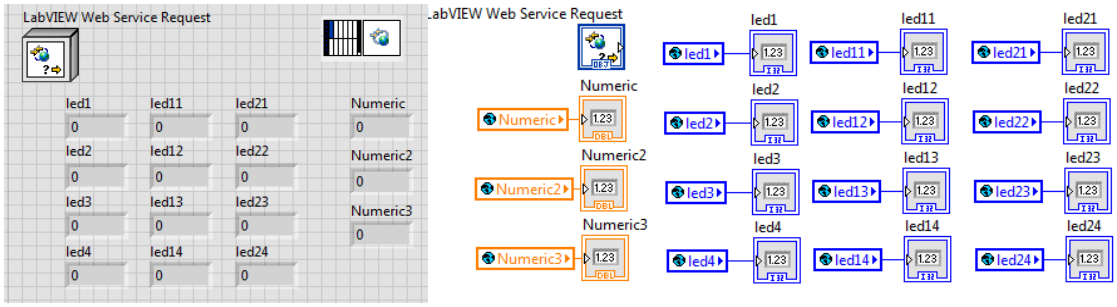


Figura 115: Programación del VI para transmitir información a la página web

Se crea un archivo tipo javascript (script.js) en el editor Sublime Text dentro de la carpeta contenido_público, posteriormente se realiza la programación para que la información de la página web de monitoreo se actualice cada 50 milisegundos, para obtener la información se emplea una instrucción que obtenga la lectura de las variables y los estados de alarma de los diferentes procesos, seguramente se declaran atributos id para presentar la información de cada variable, (atributos que se observan de color rojo en la sección 3.8.2), en base a los cuales se especifica que se modificará el texto del elemento con atributo id="indicador", y la propiedad del color de fondo del elemento con atributo id="led" de esta forma se representa la activación o desactivación de los led gráficos; el código completo se encuentra en el Anexo E y a continuación se muestra la parte esencial del mismo, ya que el código es similar para cada variable.

```
setInterval(function() {
  $.get( "datosweb/escribir", function( data ) {
    $("#indicador1.diseno_indicador").text(data.Numeric);
  if((data.led1)==1)
```

```
    {$("#led1.led").css({"background-color":[200]});}
   else
    {$("#led1.led").css({"background-color":[30]});}
 });
}, 50);
```

Para incluir el script en página web de monitoreo se debe añadir el código que se muestra a continuación antes de la etiqueta de cierre </body>, para que al ejecutar la página web en un navegador el script realice una solicitud tipo GET al servicio web y obtenga la información que se necesite mostrar en la misma.

```
<script>src="js/script.js"></script>
```

Para publicar la página web de monitoreo, es decir que se pueda acceder a la misma desde cualquier dispositivo conectado a la red, se enciende el servicio web pulsando click derecho sobre el mismo y seleccionado la opción start, posteriormente se obtiene la dirección URL pulsando click derecho sobre el archivo HTML y seleccionado la opción Show Public URL (la dirección URL la establece LabVIEW); es importante mencionar que si se realiza alguna modificación dentro de la carpeta contenido_público, los cambios se harán visibles dentro del proyecto en tiempo real. En la Figura 116 se puede observar la estructura del proyecto en LabVIEW.

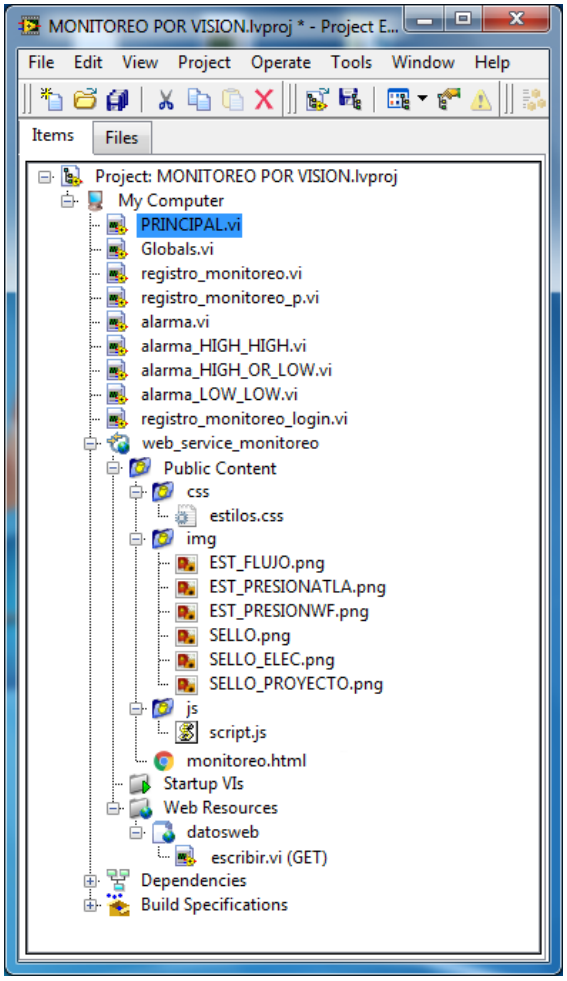


Figura 116: Estructura del proyecto en LabVIEW con el servidor web

3.8.4. Creación de la página web de identificación

Manteniendo los mismos principios del desarrollo de la página web de monitoreo se realiza la página web de identificación en donde los usuarios que deseen acceder al sitio web deberán completar la información solicitada. La publicación de esta página web se realiza utilizando el servidor local, por lo que los archivos (HTML, CSS y PHP) para el desarrollo de la misma se guardan en la carpeta htdocs de XAMPP. El código HTML y CSS correspondiente al diseño de esta página web se encuentra en el Anexo 2, en esta sección únicamente se indicara el código relevante que permite verificar si los usuarios que acceden al sitio web se encuentran registrados.

En el archivo HTML dentro de las etiquetas <body> se agregan varios elementos, para crear un formulario con tres campos de entrada de información, que permitan ingresar el usuario, la contraseña, y la confirmación de acceso (botón ACCEDER), además se añaden atributos al elemento <form> para que al pulsar el botón ACCEDER los datos de identificación se envíen hacia el archivo *monitore_espe.php* utilizando el método HTTP POST (el método POST se utiliza para enviar la información del formulario desde el navegador hacia un archivo determinado, para que dicha información sea procesada). A continuación se indica el código para este propósito.

```
<body id="cuerpo">
<form action="monitoreo_espe.php" method="post">
  <h2>IDENTIFICACI&Oacute;N PARA EL ACCESO</h2>
    <input type="text" placeholder="&#128273; USUARIO" name="Usuario">
    <input type="password" placeholder="&#128273; CONTRASEÑA" name="Contrasena">
    <input type="submit" value="ACCEDER">
</form>
</body>
```

En el archivo llamado *monitore_espe.php* se recibe la información del formulario (usuario y contraseña) y se verifica si el usuario se encuentra registrado en el sistema, para esto se establece la conexión con la base de datos; se ejecuta una instrucción en la misma empleando dicha información, la instrucción selecciona de la lista *login*, usuario y contraseña que sean iguales a los ingresados y envía un resultado, del cual se guarda en una variable el número de coincidencias (filas que coinciden); seguidamente se compara si el valor es igual a cero, lo que indica que no hay coincidencias por lo tanto, se muestra la página web para identificarse nuevamente, caso contario indica que el usuario está registrado por lo que se muestra la página web del monitoreo dentro de un frame principal. A continuación se indica el código para este propósito.

```
<?php
$Usuario=$_POST['Usuario']; $Contrasena=$_POST['Contrasena'];
$conectar=@mysql_connect("localhost", "root", "vision2016", "bd_monitoreo_por_vision") or die
("NO ENCONTRA EL SERVIDOR O LA BASE DE DATOS");
$resultado=mysql_db_query("bd_monitoreo_por_vision", "SELECT * FROM login WHERE
usuario='$Usuario' AND contrasena='$Contrasena'"); $filas=mysql_num_rows($resultado);
if($filas==0){
  header("location:login.html"); mysql_free_result($resultado); mysql_close($conectar);}
```

```
?>
<!DOCTYPE html>
<html lang="es">
<head><title>MONITOREO POR VISI&Oacute;N ARTIFICIAL ESPEL</title></head>
<frameset>
<!--dirección URL asignada por LabVIEW (en la sección 3.8.3 se indicó como obtenerla)-->
<frame src="http://192.168.0.102:8001/web_service_monitoreo/monitoreo.html">
</frameset>
</html>
```

El motivo por el que se emplea un frame para mostrar la página web de monitoreo, es que no se desea mostrar la dirección URL asignada por LabVIEW al usuario, para que él siempre ingrese al sitio web por la página web de identificación, al identificarse correctamente se visualizara la página web de monitoreo pero en la dirección del navegador se mostrará únicamente la dirección del archivo *monitoreo_espe.php*, por lo tanto el usuario solo conocerá dos direcciones para acceder al sitio web, las mismas que siempre le obligaran a identificarse para visualizar el sistema de monitoreo remoto (es una medida de seguridad). De esta manera los usuarios que deseen acceder al sitio de monitoreo únicamente deben acceder a la dirección *http://192.168.0.102/monitoreo/login.html* desde cualquier dispositivo conectado a la red. A continuación se muestra en la Figura 117 el sitio web para el monitoreo.

Figura 117: Apertura del sitio web para el monitoreo remoto desde un navegador de un ordenador conectado a la red (SSID: VISION).

CAPÍTULO IV

4. PRUEBAS Y RESULTADOS

En este capítulo se presentan los resultados obtenidos, a través de pruebas experimentales, con respecto al el entrenamiento de las RNA con los algoritmos propuestos (descritos en la Sección 3.5), los tiempos de ejecución de los algoritmos en las tarjetas embebidas BeagleBone Black y Raspberry Pi 3, y la operación del sistema de monitoreo de variables basado en técnicas de visión artificial que se ha propuesto.

4.1. Entrenamiento de las RNA

Para el análisis de los algoritmos de entrenamiento de las RNA se considera importante mencionar las características relevantes del ordenador donde se ejecutaron, las cuales corresponden a: un procesador Intel Core i7 de 2.30 GHz, una RAM de 8GB y sistema operativo Windows 7. Así también, se considera la capacidad de adquisición de imágenes de la cámara Logitec C920 que corresponde a 30 cuadros por segundo.

4.1.1. Comportamiento de las neuronas de la capa de salida de cada RNA

Se considera calcular el error absoluto resultante de la diferencia de los valores de los targets y las salidas de las neuronas, comprendidos en el rango de 0.1 a 0.9, de modo que se obtienen las curvas del comportamiento de cada neurona de la capa de salida de las cuatro RNA implementadas; las Figuras 118 y 119 corresponden a las RNA empleadas en el algoritmo del manómetro y las Figuras 120 y 121 a las neuronas de las RNA correspondientes al rotámetro. El error se calcula, en cada neurona, en cada iteración de entrenamiento definida individualmente para cada RNA (sección 3.5).

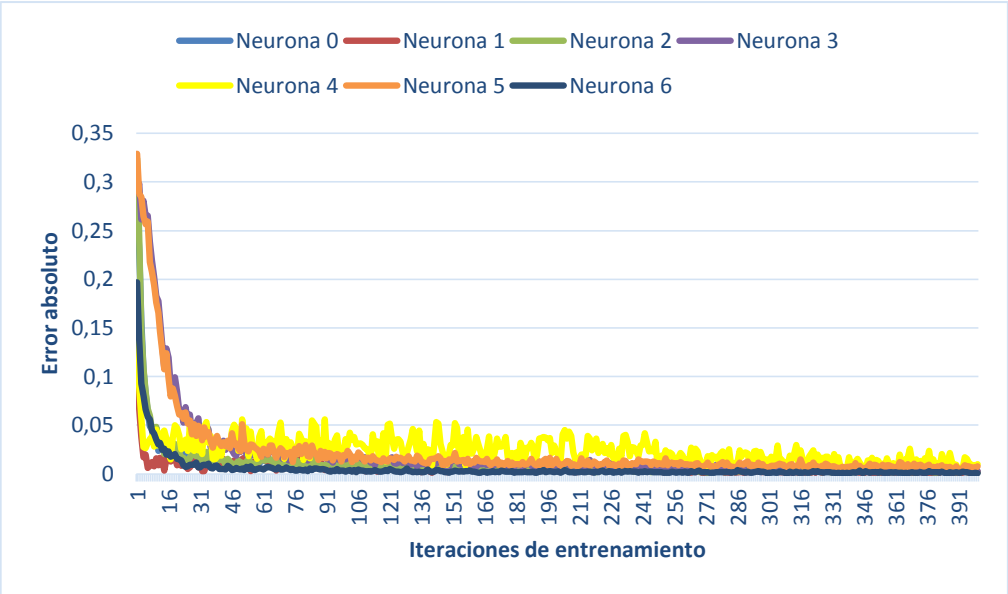


Figura 118: Error que presentan las neuronas de la capa de salida de la RNA de reconocimiento de la escala del manómetro en el entrenamiento.

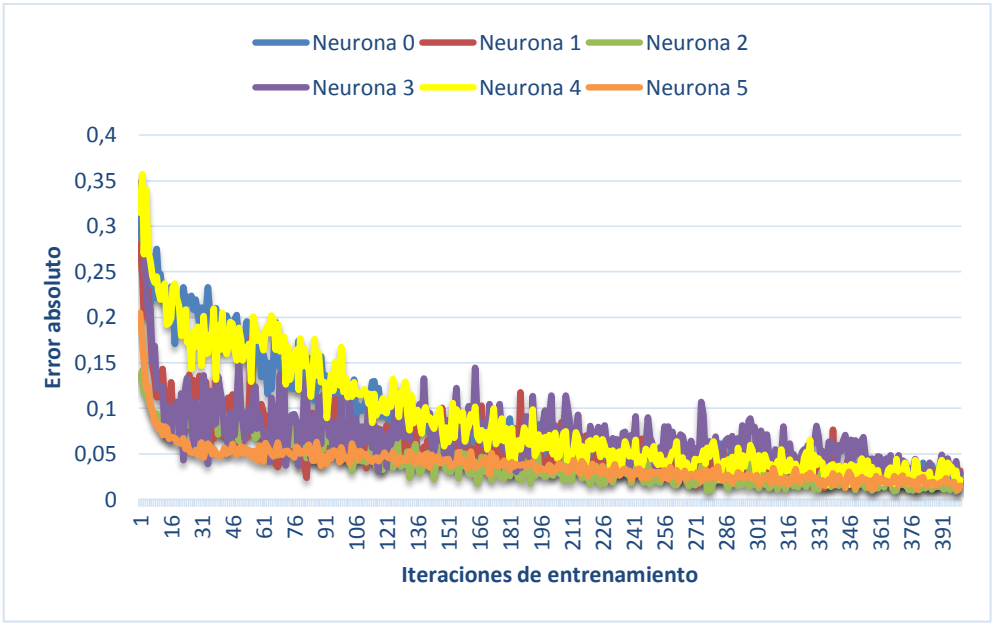


Figura 119: Error que presentan las neuronas de la capa de salida de la RNA de reconocimiento del indicador del manómetro en el entrenamiento.

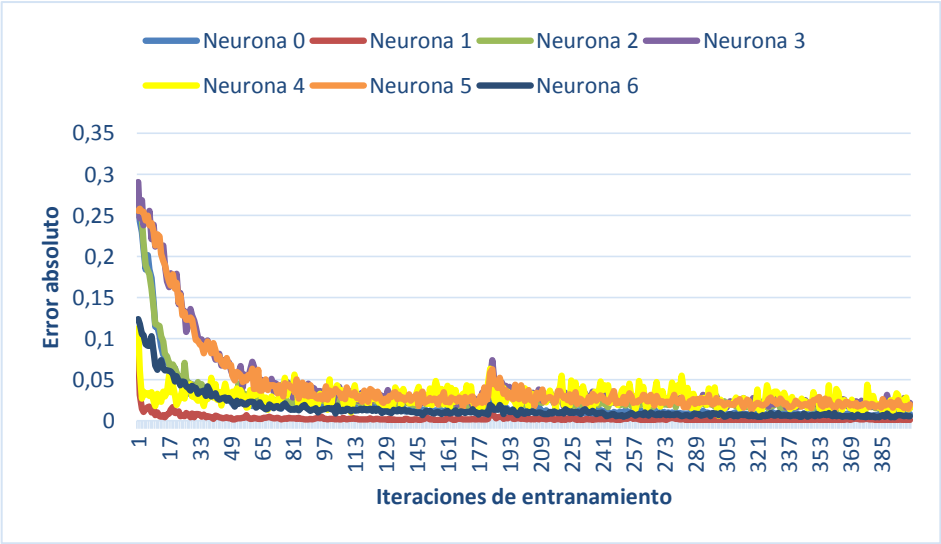


Figura 120: Error que presentan las neuronas de la capa de salida de la RNA de reconocimiento de la escala del rotámetro en el entrenamiento.

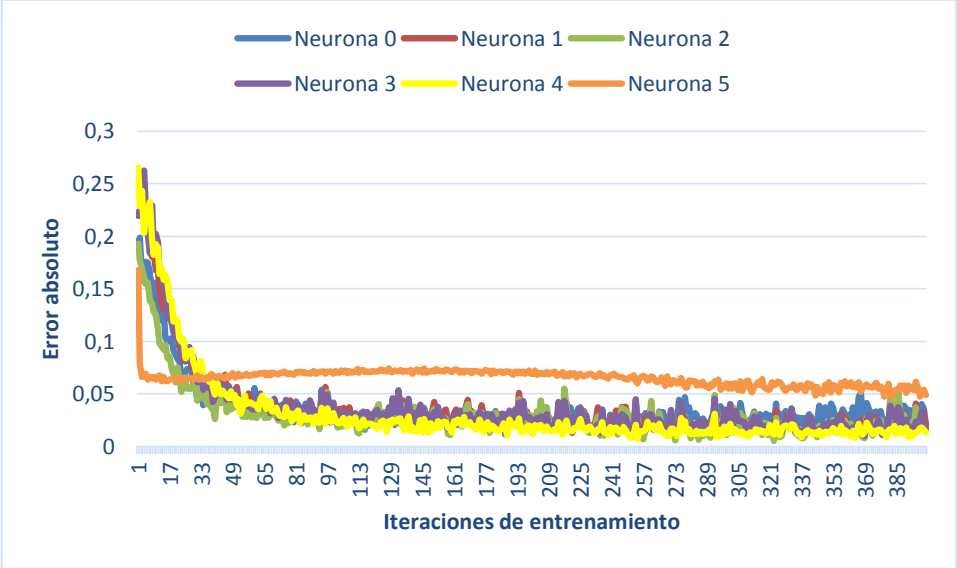


Figura 121: Error que presentan las neuronas de la capa de salida de la RNA de reconocimiento del flotador del rotámetro en el entrenamiento.

De esta manera es posible identificar que la cantidad de neuronas seleccionas en cada RNA, principalmente refiriéndose a las neuronas de la capa oculta, cumplen con el propósito de permitir el correcto aprendizaje de la RNA a través de las secuencias de entrenamiento que se han propuesto, pues el error existente entre las salidas y los targets descienden exponencialmente hasta valores de

errores inferiores a 0.05 en 400 iteraciones de entrenamiento; esto significa que las RNA están en la capacidad de inferir, es decir, de reaccionar adecuadamente ante entradas (imágenes procesadas) que no le han sido enseñadas en el proceso de entrenamiento o le fueron enseñadas pocas veces.

4.1.2. Comparación de los algoritmos de entrenamiento de las RNA

A continuación se presenta, en la Figura 122, la comparación de los algoritmos de entrenamiento de las RNA (de reconocimiento de escala e indicador) que corresponden a los algoritmos de lectura de la medida del manómetro y rotámetro; es importante recalcar que estos resultados se han obtenido utilizando el número de neuronas, de la capa escondida, que han sido establecidos como adecuados para cada RNA que opera en el algoritmo, esto es, 30 neuronas en la RNA de reconocimiento de la escala y 20 en la RNA de reconocimiento del indicador. Se ha considerado obtener el promedio del error absoluto de las curvas que se presentan las Figuras 118 a la 121, que corresponden al error del valor de activación del conjunto de neuronas de la capa de salida de cada RNA comparadas con los valores objetivo o targets.

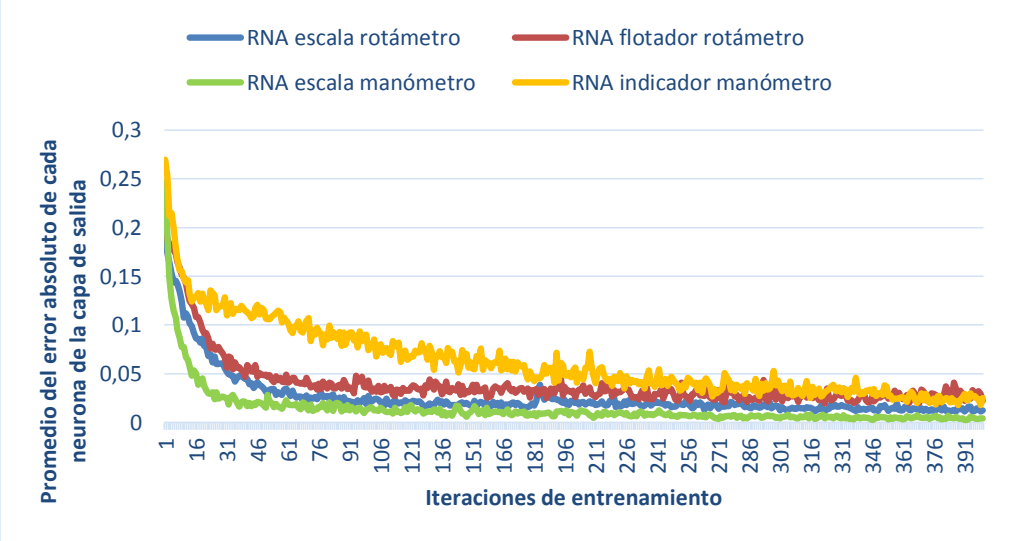


Figura 122: Comparación del entrenamiento de las RNA utilizadas.

Se determina que a partir de la iteración 200 las RNA de reconocimiento de la escala para ambos instrumentos, representa un tiempo de 30 minutos, la RNA de reconocimiento del flotador del rotámetro aprende satisfactoriamente en un tiempo de 40 minutos, y la RNA de reconocimiento del indicador del manómetro requiere 400 iteraciones que representan únicamente 8 minutos.

De esta forma se interpreta en éstos resultados la rapidez y el tiempo requerido para que cada RNA aprenda a reconocer las imágenes que se les son indicadas durante el entrenamiento; ante esto se analiza que el algoritmo propuesto requiere un tiempo de aprendizaje comprendido entre 60 y 90 minutos para ser adaptado correctamente a cualquier instrumento analógico con indicador móvil que puede o no encontrarse en operación. Es importante tener en cuenta que a más tiempo de entrenamiento mayor la probabilidad de que la RNA opere satisfactoriamente. Además se distingue que los diferentes ciclos de entrenamiento, criterios de aprendizaje, y número de neuronas utilizadas en cada capa influyen en la velocidad de aprendizaje.

4.1.3. Entrenamiento de las RNA variando el número de neuronas de la capa oculta

Para este apartado se ha considerado utilizar las dos RNA del algoritmo que corresponde al manómetro, los datos del entrenamiento se obtienen para un diferente número de neuronas definidas en la capa escondida, 10-30-50-70-90 en la RNA de reconocimiento de la escala y 10-20-40-60-80 en la RNA de reconocimiento del indicador como se indican en las Figuras 123 y 124 respectivamente.

Es importante recalcar que los resultados de entrenamiento se presentan para un número de iteraciones en el cual se aprecia de forma visual que el comportamiento de las neuronas de la capa de salida es el adecuado (se presenta un máximo de 400 iteraciones en cada gráfica), esto se logra debido a que se emplea un entrenamiento de tipo supervisado donde se aprecia fácilmente los valores de salidas y los targets.

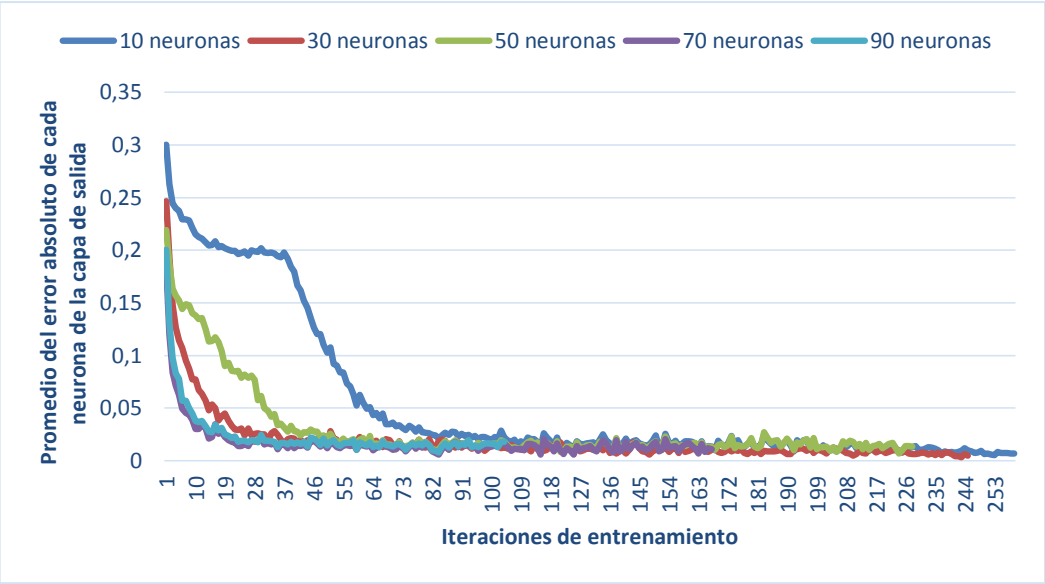


Figura 123: Variación de las iteraciones de entrenamiento requeridas según el número de neuronas de la capa oculta de la RNA de reconocimiento de la escala del manómetro.

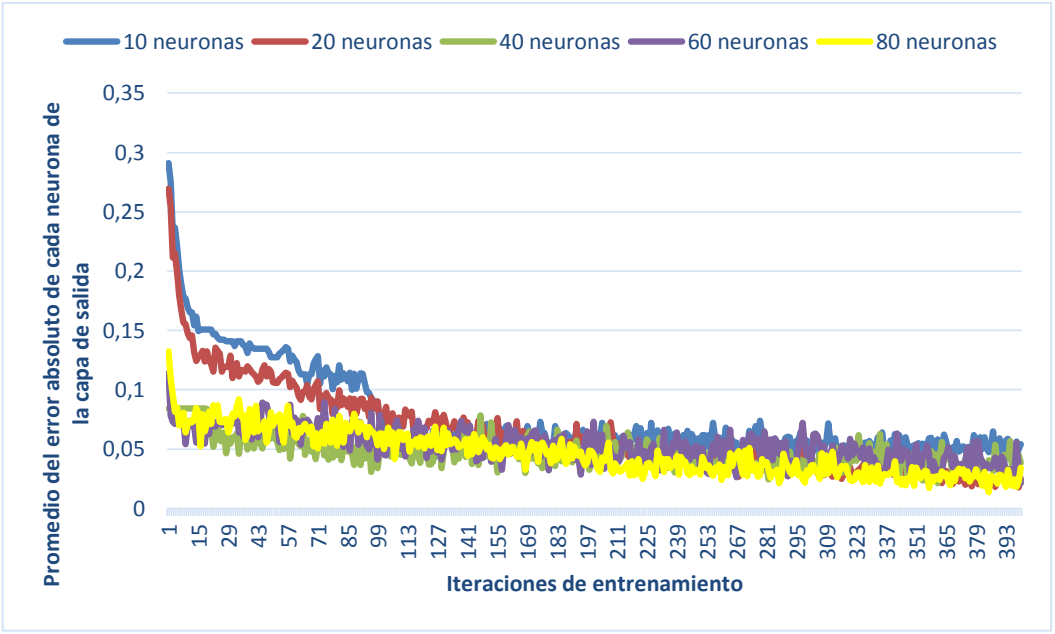


Figura 124: Varias de las iteraciones de entrenamiento requeridas según el número de neuronas de la capa oculta de la RNA de reconocimiento del indicador del manómetro.

En base a estos resultados se analiza que mientras mayor es el número de neuronas de la capa oculta una RNA aprende de forma más rápida y/o requiere menos iteraciones de entrenamiento, debido a que el error disminuye con una pendiente más inclinada, como se aprecia de forma más clara en la gráfica de la Figura 123; también es posible identificar que a partir de un número de neuronas mayor a 50 el tiempo de entrenamiento requerido es prácticamente el mismo.

De esta manera se puede establecer que un número de neuronas de la capa oculta establecido entre 20 y 50 es apropiado para garantizar el aprendizaje de una RNA que reconoce un ROI de 50x20 que contiene la forma del indicador, y 30 a 50 neuronas son apropiadas para el aprendizaje de la RNA de reconocimiento de una ROI de 50x20 que contiene la escala del instrumento. Entonces, también se analiza que ante un mayor número de entradas (reflejadas en el tamaño de las ROI) se requieren más neuronas en la capa escondida para que una RNA logre operar correctamente.

4.2. Tiempos de ejecución del algoritmo en las tarjetas embebidas

Para la obtención de los tiempos de ejecución se considera insertar líneas de código que determinan un tiempo de inicio y un tiempo de finalización dentro del algoritmo propuesto, únicamente en la rutina principal que ejecuta la lectura de la variable empleando la RNA de reconocimiento del indicador, esto debido a que el análisis se orienta a dos propósitos: comparar la velocidad de procesamiento de tarjetas embebidas y determinar la velocidad de adquisición de datos lograda a través de visión artificial.

4.2.1. Variación del tiempo de procesamiento según el número de neuronas empleadas en la RNA

En la Tabla 4 se puede encontrar los tiempos promedio de procesamiento de cada una de las tarjetas utilizadas (BeagleBone Black y Raspberry Pi 3) para diferentes números de neuronas establecidas en la capa oculta de la RNA que

reconoce el indicador del manómetro; estos resultados también se representan de forma más visual en la gráfica de barras de la Figura 125.

Tabla 4

Tiempos de ejecución del algoritmo correspondiente a la lectura de la medida en el manómetro (variable presión)

Tiempos de ejecución manómetro [ms]

Neuronas de la capa oculta de la RNA que reconoce el indicador	***10***	***20***	***40***	***60***	***80***
Tarjeta BeagleBone Black	12,450	14,084	20,533	27,413	35,246
Tarjeta Raspberry Pi 3	4,010	4,343	6,754	8,833	10,370

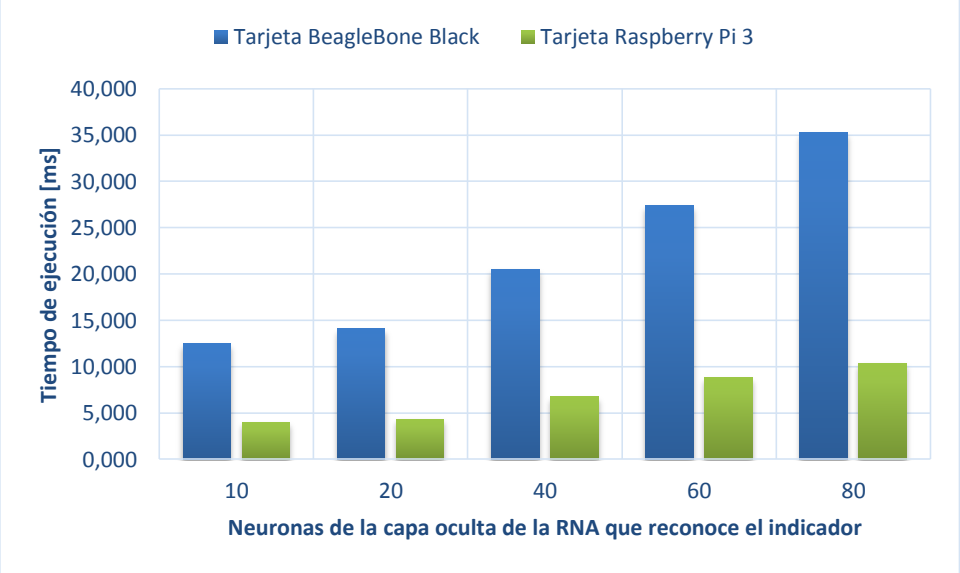


Figura 125: Representación gráfica de los tiempos de ejecución del algoritmo en las tarjetas embebidas para diferentes números de neuronas de la capa oculta de la RNA que reconoce el indicador del manómetro.

De esta forma se puede apreciar la influencia del número de neuronas de la RNA en el incremento del tiempo de procesamiento de las tarjetas embebidas, dicho de otra manera, mientras mayor es el número neuronas empleadas se incrementa el número de iteraciones de cálculos de la RNA, lo que ocasiona que el tiempo necesario para ejecutar un barrido del algoritmo se incremente.

En base a lo anterior se analiza que se adquirirán menor cantidad de muestras de la variable, a través de visión artificial, porque no se puede capturar el movimiento del indicador en una secuencia de imágenes continuas y separadas por intervalos de tiempo pequeño (en el orden de los ms), de tal forma que se puede ir perdiendo el concepto de adquirir y transmitir datos en tiempo real si no se maneja un número apropiado de neuronas.

Además, en estos resultados, se justifica el empleo de 20 neuronas en la RNA de reconocimiento del indicador, debido a que un tiempo inferior a los 20 ms en el dispositivo embebido que presenta los tiempos de ejecución más altos es bastante aceptable para esta aplicación; y también porque durante el desarrollo se evidenció de forma experimental que un número de neuronas inferior a 20 disminuye la efectividad de reconocimiento de la RNA.

4.8.4. Comparación entre BeagleBone Black y Raspberry Pi 3

En las Tablas 5 y 6 se presenta los tiempos promedios que tarda en ejecutarse el algoritmo de lectura de variables por visión artificial para los dos tipos de instrumentos analógicos que se presentan en este trabajo; es necesario tener en cuenta que en ambos casos se utilizan RNA con el mismo número de neuronas de la capa oculta (20) y de la capa de salida (6), pero el tamaño de las ROI es diferente, lo cual implica un mayor número de neuronas de la capa de entrada en el caso del indicador del manómetro; y como ya se indicó en la sección 4.2.1 a más neuronas involucradas en los cálculos mayor es el tiempo requerido para la ejecución.

Es importante mencionar que, para obtener los datos de los tiempos de ejecución, se han seleccionado 50 muestras de forma aleatoria mientras el algoritmo se encontraba en ejecución tanto en la tarjeta BeagleBone Black y la tarjeta Raspberry Pi 3.

Tabla 5

Tiempos de ejecución del algoritmo correspondiente a la lectura de la medida del manómetro (variable presión) en las tarjetas BeagleBone Black y Raspberry Pi 3

Ejecución del algoritmo manómetro

Tarjeta embebida	***Tiempo medio [ms]***	***Desviación estándar [ms]***
BeagleBone Black	**14,084**	2,632
Raspberry Pi 3	**4,343**	0,909

Tabla 6

Tiempos de ejecución del algoritmo correspondiente a la lectura de la medida del rotámetro (variable flujo) en las tarjetas BeagleBone Black y Raspberry Pi 3

Ejecución del algoritmo rotámetro

Tarjeta embebida	***Tiempo medio [ms]***	***Desviación estándar [ms]***
BeagleBone Black	**9,568**	2,787
Raspberry Pi 3	**3,148**	0,144

En los resultados de la Tabla 5 se identifica que para el caso de la BeagleBone Black, si se suma el tiempo promedio de ejecución con el valor de desviación estándar se logra interpretar de forma confiable que se logra adquirir, por visión artificial, una muestra de la variable presión cada 16.716 ms, y en la Raspberry Pi 3 se adquiere cada muestra en intervalos de 5.252 ms.

Es importante mencionar que el algoritmo correspondiente al rotámetro se diseñó desde un principio para ser ejecutado en menor tiempo en comparación

al caso del manómetro, porque como antecedente se ha experimentado que la variable flujo puede llegar a presentar cambios con más velocidad que la presión, de esta manera se justifica que en la Tabla 6 se presenten resultados de tiempo inferiores en comparación a la Tabla 5. De la misma forma, se determina que la BeagleBone Black está en la capacidad de adquirir muestras de la variable flujo cada 12.355 ms y la Raspberry Pi 3 cada 3.292 ms.

Debido a que el tiempo de adquisición se encuentra en el orden de los mili segundos y no alcanza un valor de 20 ms es posible afirmar que el sistema embebido (BeagleBone Black y Raspberry Pi 3) está operando en tiempo real, por lo tanto desde el sistema servidor, donde se presenta la información del estado de cada variable, es factible seleccionar tiempos de adquisición pequeños (que superen los 20 ms) para que la información se actualice de forma frecuente, además al emplear el protocolo UDP los tiempos de transmisión pueden considerarse despreciables, de esta manera se logra un monitoreo remoto en tiempo real.

Al igual que en los resultados que se indicaron en la Figura 125 en la sección 4.2.1, en estos nuevos resultados de las Tablas 5 y 6 se comprueba que la tarjeta Raspberry Pi 3 ejecuta algoritmos tres veces más rápido que la tarjeta BeagleBone Black; entonces, para el caso de éste trabajo, se determina que la tarjeta Raspberry Pi 3 es apta para aplicarse a variables que pueden llegar a cambiar su estado de forma muy rápida y/o son oscilatorias como lo es el flujo de un líquido.

Además de la ventaja en los tiempos de ejecución que presenta la tarjeta Raspberry Pi 3 respecto a su homóloga, queda en evidencia que no es necesario añadir más componentes de hardware debido a que todo se encuentra integrado en la tarjeta, refiriéndose principalmente a los módulos de comunicación inalámbrica WIFI, a diferencia de lo que sucede en la BeagleBone Black; sin embargo los propósitos del proyecto contemplado en este trabajo se pueden cumplir mediante la utilización de cualquiera de las dos tarjetas embebidas.

4.3. Operación del sistema

La operación del sistema propuesto en este proyecto es comparado con equipos especializados que determinan el valor de una variable, específicamente refiriéndose a transmisores industriales de presión absoluta Foxboro IAP20, y de flujo Georg Fischer 8550; éstos transmisores presentan características de medición muy robustas y eficaces, únicamente superadas por equipos de calibración, por ejemplo el Foxboro IAP20 presenta una exactitud de ± 0.05% del rango definido y el Georg Fischer 8550 presenta una precisión de ± 0.7%.

La comparación se realiza para los dos casos tratados en el algoritmo de lectura por visión artificial, la Figura 126 corresponde a la variable presión y la Figura 127 a la variable flujo; para ambos casos se ha considerado realizar las pruebas generando entradas aleatorias y abruptas de tipo escalón.

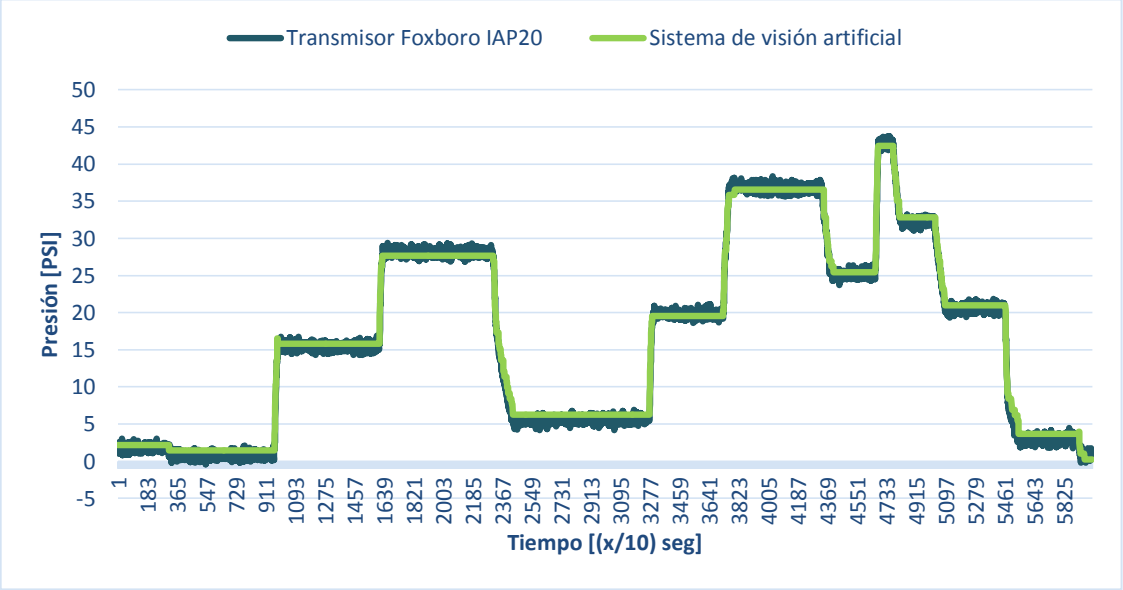


Figura 126: Respuesta del sistema propuesto para la variable presión comparado con el transmisor Foxboro IAP20.

Tabla 7

Resultados de la comparación entre los datos de las curvas presentadas en la Figura 119

Sistema de visión artificial VS transmisor Foxboro IAP20

Datos de la variable presión	***Error absoluto medio [PSI]***	***Desviación estándar [PSI]***
Correspondientes a toda la curva	0,628	0,463
Correspondientes a entradas en estado estable	0,589	0,402

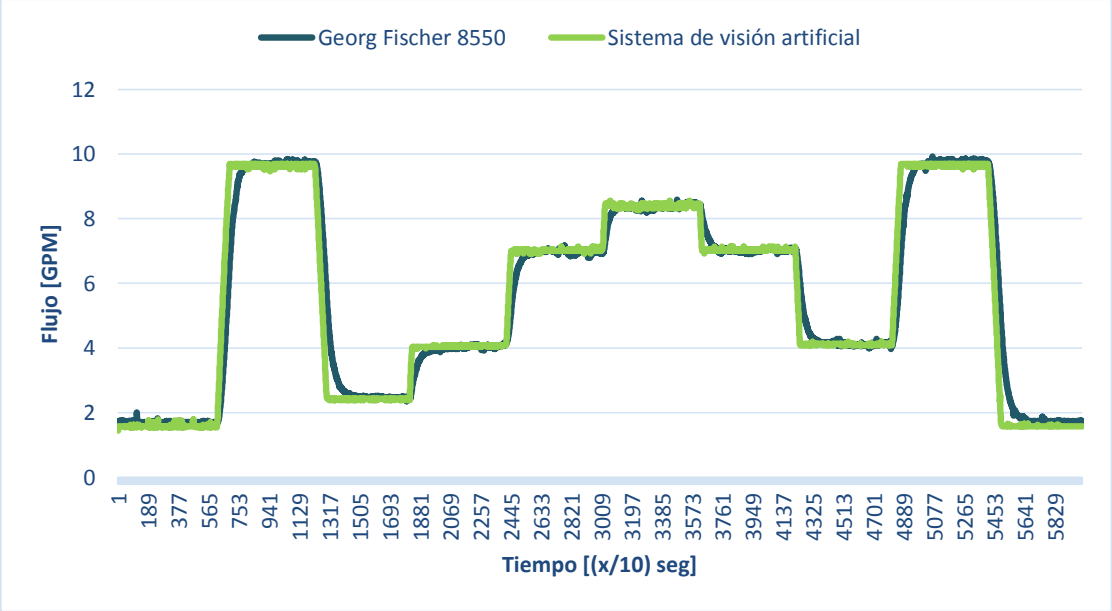


Figura 127: Respuesta del sistema propuesto para la variable presión comparado con el transmisor Georg Fischer 8550.

Tabla 8

Resultados de la comparación entre los datos de las curvas presentadas en la Figura 120

Sistema de visión artificial VS transmisor Georg Fischer 8550

Datos de la variable flujo	***Error absoluto medio [PSI]***	***Desviación estándar [PSI]***
Correspondientes a toda la curva	0,288	0,573
Correspondientes a entradas en estado estable	0,085	0,059

Se realiza una comparación de la respuesta del transmisor industrial y el sistema propuesto para cada variable obteniendo el error absoluto entre cada una de estas muestras, de modo que se presentan las Tablas 7 y 8, las cuales contienen el error absoluto promedio y la desviación estándar éste conjunto de muestras de error absoluto; se considera todos los datos de las curvas y, también, únicamente los datos de las curvas cuando no existe variación de entradas, es decir, cuando las curvas alcanzan una tendencia estable.

Es posible estimar el error de medición presentado por el sistema propuesto, para lo que se considera el promedio del error absoluto correspondiente a entradas en estado estable; en el caso de la variable presión se tiene 0.589 PSI, que sumado a la correspondiente desviación estándar se estima un error máximo que no supera 1 PSI; para el caso del flujo se tiene un error absoluto promedio de 0.085, y mediante la adición del valor de desviación se estima un error absoluto no mayor a 0.150 GPM.

Éstos resultados están en función de la comparación con los transmisores industriales ya mencionados que operan en un rango definido, de tal forma que en el caso de la presión el error máximo estimado representa un 2.5% del rango de la escala comprendido entre 1-40 PSI, y en el flujo se tiene un error del 1.6% del rango de 1-10 GPM, lo cual se considera como aceptable para el monitoreo y registro automático de datos.

Es importante tener en cuenta que al realizar un registro manual a través de la lectura de los instrumentos analógicos un operador no distingue exactamente el valor entre dos graduaciones de la escala, no obstante se tienen como ventaja la opción de variar las ecuaciones de escalamiento en el algoritmo para conseguir respuestas del sistema acordes a la aplicación.

Por otra parte, la tendencia de la respuesta ante cambios bruscos y aleatorios es similar a la operación de los equipos industriales utilizados para ambos variables, esto se evidencia comparando el promedio del error absoluto y la desviación estándar de los datos de toda la curva con los mismos parámetros pero considerando únicamente los datos de las tendencias estables de las curvas; por lo que en el caso de la variable presión no existe mucha diferencia entre cada uno de éstos valores (Tabla 7), entonces es posible afirmar que la tendencia de curva es la misma.

En el caso del flujo sí existe una diferencia más significativa de éstos parámetros (Tabla 8) en comparación a la presión, de modo que se identifica que existe una mínima diferencia en la tendencia, principalmente en las transiciones de ascenso y descenso; esto se debe a que el transmisor Georg Fischer 8550 internamente realiza un conteo rápido de pulsos generados por la rotación de su elemento primario (paletas giratorias) en un periodo de 100 ms para calcular la velocidad de circulación del líquido, originando un retardo en la actualización del valor medido, lo que no sucede con el sistema propuesto que presenta una tendencia de curva más semejante a la entrada de tipo escalón, es decir, la actualización del dato de la variable es más rápida.

En la Figura 126 se observa que el sistema propuesto no alcanza a distinguir las oscilaciones que si distingue el transmisor Foxboro IAP20, sin embargo, el valor presentado puede considerarse como un promedio de las oscilaciones de la curva correspondiente al transmisor; en cambio, en la Figura 127 se puede identificar que el sistema propuesto alcanza a capturar las oscilaciones del flujo, desde luego, esto depende mucho más de la sensibilidad del instrumento analógico.

4.4. Alcances y limitaciones

El proyecto presenta los siguientes alcances:

- La investigación contempla el monitoreo de variables industriales mediante la lectura de instrumentos analógicos basado en visión artificial y redes neuronales para el reconocimiento de formas tanto cilíndrica como circular.
- El presente trabajo emplea dispositivos embebidos en los cuales se ejecutan algoritmos basados en redes neuronales que reconocen tanto la forma del instrumento analógico como el indicador, el cual señala la medición de la variable física, por lo que se procesan operaciones matemáticas para interpretar la posición del indicador; posteriormente se envia la lectura de la medición al sistema servidor mediante comunicación inalámbrica WIFI.
- El proyecto de investigación incluye el monitoreo de variables físicas en tiempo real (orden de los milisegundos), tanto de forma local como remota, así también, el registro de la información de dichas variables en una base de datos en un intervalo definido por el usuario, lo cual resulta beneficioso para un futuro análisis y toma de decisiones en base a estos datos.
- El monitoreo remoto de este sistema está diseñado para ser ejecutado en cualquier dispositivo que contenga un navegador y que se encuentre conectado a la red.

Así también se distinguen las siguientes limitaciones:

- El sistema de monitoreo desarrollado funciona correctamente bajo condiciones de iluminación constantes.
- El trabajo de investigación emplea un algoritmo de visión artificial que depende de que la cámara que interviene en la captura de la imagen mantenga una posición fija.
- La estructura física del sistema de monitoreo no contempla un grado de protección IP, por lo que, para ser implementado en la industria es necesario una estructura especializada que esté acorde al ambiente de operación.

- El router empleado para el sistema de monitoreo no es industrial por lo que la fiabilidad de transmisión de la información no es determinístico.

4.5. Trabajo Futuro

Debido a que éste trabajo emplea técnicas de visión artificial basada en RNA, se requiere realizar un proceso de entrenamiento para que estos elementos de procesamiento aprendan a reconocer distintas imágenes procesadas, por lo que se propone el desarrollo de un software con una interfaz intuitiva que entrene las RNA de forma automática y descargue los pesos sinápticos a los dispositivos embebidos una vez finalizado éste proceso. Así también, que brinde al usuario la posibilidad de variar los parámetros de las funciones utilizadas en el procesamiento de imágenes, como los filtros y el detector de bordes Canny, o las ecuaciones de escalamientos; de esta manera será mucho más sencillo adaptar el algoritmo para cualquier instrumento analógico que posea un indicador móvil sobre una escala.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- La tarjeta Raspberry PI 3 alcanza velocidades de ejecución tres veces más altas en relación a la tarjeta BeagleBone Black para esta aplicación, considerando que se ha utilizado los sistemas operativos recomendados por las respectivas fundaciones.
- Los algoritmos implementados pueden ejecutarse en cualquiera de las dos tarjetas embebidas utilizadas en este trabajo, sin embargo, la tarjeta Raspberry Pi 3 presenta ventajas sobre la BeagleBone Black con respecto a la velocidad de procesamiento y debido a que ya integra un módulo de comunicación inalámbrica.

- La utilización de RNA ha permitido que se diseñe un algoritmo que determine el valor de una variable leída en una escala, además este algoritmo puede ajustarse a cualquier instrumento que base su operación en el movimiento de indicadores sobre una escala graduada.
- A medida que se incrementan el número de neuronas de la capa oculta en una RNA se requiere menos tiempo de entrenamiento, pero, se incrementa el costo computacional del algoritmo que emplea dicha RNA.
- Se debe definir el número de neuronas en cada capa, en especial en la capa oculta ya que el incremento de neuronas reduce la velocidad de procesamiento para la obtención de datos en tiempo real, es decir, es necesario mantener un equilibrio entre el número de neuronas que cumplen el propósito y que no incrementen los cálculos de la red.
- Para un funcionamiento correcto por parte de la RNA se necesita un entrenamiento supervisado, donde se va verificando que las neuronas de salida se comporten igual a los targets establecidos a partir de patrones conocidos, los cuales aparecen en una secuencia aleatoria.
- Para los entrenamientos se debe incluir una rutina de ruido para que los algoritmos de ejecución de la RNA sean robustos, es decir, respondan de una determinada forma ante entradas que no tienen relación con la escala o el indicador del instrumento.
- Realizar un entrenamiento invadiendo con ruido la ROI que contiene al borde del flotador del rotámetro ha resultado beneficioso para que el reconocimiento no se vea alterado cuando surgen formas aleatorias por efecto de la presencia de burbujas o variaciones de iluminación.
- Es importante utilizar un sistema de iluminación para evitar el desarrollo de algoritmos complejos que tienen que ver con la eliminación de sombras y reconstrucción de formas dentro de la imagen, debido a que incrementan el tiempo de procesamiento de la tarjeta embebida.

- El empleo de RNA en aplicaciones de visión artificial permite optimizar el uso de los recursos de hardware ya que, a diferencia de otras técnicas de visión, se utiliza el almacenamiento de datos numéricos que corresponden a los pesos de conexión entre neuronas y no imágenes que actúan como patrones de una aplicación.
- La transmisión de datos desde las tarjetas embebidas al sistema servidor se realiza a través de una red inalámbrica empleando el protocolo UDP, esta técnica de comunicación se asemeja a la empleada por equipos y estándares industriales para la transmisión de datos en tiempo real.
- El tiempo de adquisición de las lecturas correspondientes a las variables físicas es adecuado para un sistema de monitoreo, ya que se permite obtener una curva de respuesta del proceso y establecer el tiempo de registro en el orden de los milisegundos.
- Se puede implementar un control a lazo cerrado, puesto que los resultados de las comparaciones entre las respuestas del transmisor y el sistema propuesto presentan una tendencia similar.
- El software Xampp permite gestionar la base de datos de forma independiente, es decir no a través de la aplicación de LabVIEW, de esta forma el administrador puede descargar fácilmente la información registrada, para utilizarla de forma pertinente.
- Para el monitoreo remoto se ha empleado un servicio web de LabVIEW que permite compartir la información con varios clientes web, los mismos que pueden identificarse e ingresar al mismo tiempo, además, a través de éste servicio es posible desarrollar sitios web personalizados porque se complementa con lenguajes de programación de la especificación HTML5.

5.2. Recomendaciones

- Se recomienda no variar las condiciones de iluminación y la posición de la cámara para que el algoritmo que interpreta la medición del instrumento analógico no envíe lecturas erróneas al sistema servidor.

- Utilizar una cámara que sea compatible con el procesador de la tarjeta embebida para evitar problemas en la adquisición de la imagen.
- Utilizar tarjetas micro-SD clase 10 y con capacidad superior a las 8GB para cargar las imágenes de los sistemas operativos, esto por cuestión de velocidad de manejo de datos y capacidad de almacenamiento de archivos y aplicaciones.
- Descargar y eliminar la información registrada en la base de datos mediante el acceso a la interfaz PHPmyAdmin, con el fin de liberar espacio en la memoria.
- No conectar más de ocho dispositivos a la misma red local, para no alterar la velocidad de transmisión de datos en la red WIFI, esto se debe a la capacidad del enrutador utilizado en este proyecto de investigación.
- Se recomienda verificar el registro de los usuarios que puedan acceder a la página web de monitoreo remoto para que no existan problemas en la identificación de acceso.

REFERENCIAS BIBLIOGRÁFICAS

Adafruit. (2016). *Raspberry Pi 3 - Model B - ARMv8 with 1G*. Recuperado en Agosto de 2016, de https://www.adafruit.com/product/3055

Advanced ilumination. (2016). *RL1424 Bright Field Ring Light*. Recuperado en Agosto de 2016, de http://www.ni.com/pdf/products/us/lighting.pdf

Aguilar Vergara, P. A. (Enero de 2008). Repositorio Universidad de Chile. *Reconocimiento de Bordes en Imágenes Aplicado a Anillos de Árboles*. Santiago de Chile. Recuperado en Agosto de 2016, de http://www.tesis.uchile.cl/tesis/uchile/2008/aguilar_pv/sources/aguilar_pv.pdf

Alfonzo, P., Medina, Y., Quintana, O., & Salazar, L. (2012). *Programación II-Taller de Programación I.* Obtenido de Programación II- Taller de Programación I. Recuperado en Agosto de 2016, de http://exa.unne.edu.ar/informatica/programacion2/TutorialXAMPP.pdf

Alonso, A. D., & Jara, E. (2013). Visión por computadora: identificación, clasificación y seguimiento de objetos. *FPUNE Scientific*. Recuperado en Agosto de 2016, de http://www.une.edu.py:82/fpune_scientific/index.php/fpunescientific/article/view/172/170

Andrade Miranda, G., López Encalada, J., & Chávez Burbano, P. (Marzo de 2009). Repositorio Escuela Politécnica del Litoral. *Sistema de Control Vehicular utilizando Reconocimiento Óptico de Caracteres*. Recuperado en Agosto de 2016, de https://www.dspace.espol.edu.ec/bitstream/123456789/1458/1/2973.pdf

Arenas, A. (Marzo de 2014). *Boletían UNAMente Robótica.* Recuperado en Agosto de 2016, de http://dcb.fi-c.unam.mx/Publicaciones/UNAMenteRobotica/Boletin_7.pdf

BeagleBoard.org Foundation. (2016). *beagleboard.org*. Recuperado en Agosto de 2016, de BeagleBone Black: https://beagleboard.org/black

BeagleBoard.org Foundation. (2016). *BeagleBone: open-hardware expandable computer*. Recuperado en Agosto de 2016, de http://beagleboard.org/support/bone101

BeagleBoard.org Foundation. (2016). *Getting Started with BeagleBone*. Recuperado en Agosto de 2016, de https://beagleboard.org/getting-started

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV* (Primera ed.). Sebastopol: O'REILLY. Recuperado en Agosto de 2016, de http://www-cs.engr.ccny.cuny.edu/~wolberg/capstone/opencv/LearningOpenCV.pdf

Brauer Luna, P. M. (Junio de 2013). Control de un robot humanoide mediante redes neuronales pulsantes para la manipulación de objetos. México, D.F.: Repositorio Instituto Politécnico Nacional. Recuperado en Septiembre de 2016, de http://tesis.ipn.mx/handle/123456789/16050

Caibe Yanzapanta, F. M. (2012). *Repositorio Escuela Superior Politécnica de Chimborazo.* Obtenido de Diseño y construcción de un sistema de transporte de fluidos para la medición de caudales (rotámetro). Recuperado en Agosoto de 2016, de http://dspace.espoch.edu.ec/bitstream/123456789/2513/1/96T00195.pdf

Calderón, J., & Sánchez Montero, Y. (Diciembre de 2004). *Mediciones e Instrumentación Industrial.* Recuperado en Agosto de 2016, de http://www.ancap.com.uy/docs_concursos/ARCHIVOS/2%20LLAMADOS%20FINALIZADOS/2013/REF%2040_2013%20%20%20T%C3%89CNICO%20AYUDANTE%20MANTENIMIENTO%20E%20INGENIER%C3%8DA%20(ELECTROELECTR%C3%93NICO)/MATERIAL%20DE%20ESTUDIO/CONOCIMIENTOS%20ESPEC%C3%8DFICOS/MEDICI

Calvo Tejedor, J. C. (10 de Junio de 2013). Repositorio Universitat de Barcelona. *Control de un Robot Mediante Técnicas de Visión*. Barcelona, España. Recuperado en Agosto de 2016, de http://diposit.ub.edu/dspace/bitstream/2445/47683/2/memoria.pdf

Candelas Herías, F. A., & Pomares Baeza, J. (2009). *Universidad de Alicante.* Recuperado en Agosto de 2016, de Manual de la Práctica 3: Protocolos de Transporte TCP y UDP: https://rua.ua.es/dspace/bitstream/10045/11606/1/Pr3-2009-10.pdf

Cárdenas Vera, M. F., & LLerena Pizarro, O. R. (2012). Repositorio Universidad Politécnica Salesiana. *Automatización de un sistema de centrado de componentes utilizando visión artificial*. Cuenca. Recuperado en Agosto de 2016, de http://dspace.ups.edu.ec/bitstream/123456789/1109/14/UPS-CT002210.pdf

Colegio Oficial de Ingenieros de Telecomunicación. (2004). *La situación de las tecnologías WLAN basadas en el estándar IEEE 802.11 y sus variantes (Wi-Fi).* Recuperado en Agosto de 2016, de http://www.minetur.gob.es/telecomunicaciones/Espectro/NivelesExposicion/Informacin/coitInforme_wifi_2004.pdf

Cotrina Escandón, M. F., & Peña Álvarez, D. d. (2011). Repositorio Escuela Politécnica del Litoral. *Diseño e Implementación de un sistema de calificación para exámenes de opción múltiple mediante la herramienta MATLABR2008B*. Guayaquil. Recuperado en Agosto de 2016, de http://www.dspace.espol.edu.ec/xmlui/bitstream/handle/123456789/19933/DISE%C3%91O%20E%20IMPLEMENTACION%20DE%20UN%20SISTEMA%20DE%20CALIFICACION%20PARA%20EXAMENES%20DE%2

0OPCION%20MULTIPLE%20USANDO%20MATLAB.pdf?sequence=2&isAllowed=y

Creus Solé, A. (2011). *Instrumentación Industrial* (Octava ed.). México, D. F.: Alfaomega.

Cruz Dorado, G. A. (2014). Repositorio Universidad Nacional Autónoma de México. *Mantenimiento y Operación de Redes de Área Local Inalámbrica*. Cuautitlán Izcalli, Estado de México. Recuperado en Agosto de 2016, de http://avalon.cuautitlan2.unam.mx/biblioteca/tesis/848.pdf

De la Fuente López, E., & Trespademe, F. M. (2012). Visión Artificial Industrial Procesamiento de Imágenes para Inspección Automática y Robótica. *Iluminación*. Valladolid: Universidad de Valladolid. Recuperado en Agosto de 2016, de http://www.librovision.eii.uva.es/pdf/cap11.pdf

Debian. (5 de Julio de 2016). *Acerca de Debian*. Recuperado en Agosto de 2016, de https://www.debian.org/intro/about.es.html

Debian Proyects. (2016). *Raspbian*. Recuperado en Agosto de 2016, de Welcome to Raspbian: https://www.raspbian.org/

Diputación de Badajoz. (s.f.). *Las Tecnologías WIFI y WIMAX.* Recuperado en Agosto de 2016, de http://www.dip-badajoz.es/agenda/tablon/jornadaWIFI/doc/tecnologias_wifi_wmax.pdf

Dunn, W. (2005). *Fundamentals of Industrial Instrumentation and Process Control.* McGraw-Hill.

Embedded Linux Wiki. (2016). *Embedded Linux Wiki*. Recuperado en Agosto de 2016, de Beagleboard:BeagleBoneBlack: http://elinux.org/Beagleboard:BeagleBoneBlack#Books

Eng Hee, Y. (2016). *PuTTY; AFree SSH Client.* Recuperado en Agosto de 2016, de http://www.nus.edu.sg/comcen/news/HPC/articles/Putty.pdf

Escolando Ruiz, F., Cazorla, M., Alfonzo, M., Colomina, O., & Lozano, M. (2003). *Inteligencia artificial: Modelos, técnicas y áreas de aplicación.* Madrid: Paraninfo.

Escribano Sánchez, I., & Escardino Malva, A. (2003). *Repositorio Universidad Politécnica de Valencia.* Recuperado en Agosto de 2016, de Desarrollo de un sistema de visión artificial para el control eficiente de pulverizadores de cera en el tratamiento post-cosecha de la fruta: https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjjlNbb4orPAhUHGR4KHSJOAnIQFgglMAA&url=http%3A%2F%2Fcpi.upv.es%2Frs%2F2543%2Fd112d6ad-54ec-438b-9358-

4483f9e98868%2F8e6%2Ffd%2F1%2Ffilename%2Fcasos-cpi-citrosol.pdf&usg=AFQjCNFHq

Esqueda Elizondo, J. J., & Palafox Maestre, L. E. (2005). *Fundamentos de procesamiento de imágenes.* Baja California: Universodad Autónoma de Baja California. Recuperado en Agosto de 2016, de https://books.google.com.ec/books?id=h4Gj8GuwPVkC&pg=PA16&lpg=PA16&dq=ruido+en+procesamiento+de+imagenes&source=bl&ots=Cr1w6dVxcv&sig=BKh0HayUg917YX8fjl1uBQYPUGY&hl=es-419&sa=X&ved=0ahUKEwi37I-q2IXPAhVDph4KHeYrCZAQ6AEIVzAJ#v=onepage&q=ruido%20en%20procesa

Fábregas, E. (21 de Mayo de 2009). *EtherNet/IP.* Recuperado en Agosto de 2016 de Documento de Rockwell Automation: http://automata.cps.unizar.es/conferencias/rockwell/EtherNetIP_210509.pdf

García Moya, A., & Barriga Barros, A. (2012). Curso Práctico de Sistemas Empotrados Basado en Placas de Desarrollo XUPV2P. *IEEE-RITA*, 231-237. Recuperado en Agosto de 2016, de http://rita.det.uvigo.es/201211/uploads/IEEE-RITA.2012.V7.N4.A11.pdf

Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y Javascript.* Barcelona: MARCOMBO.

Gellaboina, M. K., Swaminathan, G., & Venkoparao, V. (2013). Analog dial gauge reader for handheld devices. *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, 1147-1150.

Goldstein, A., Lazaris, L., & Weyl, E. (2011). *Manual imprescindible HTML5 Y CSS3.* España: Grupo ANAYA, S.A.

Hanke, J. C. (s.f.). PHP 5. *PHP 5 Sitios web dinámicos*. Francia: KnowWare E.U.R.L.

Hernández, L. M. (2012). *Diferencia, ventajas y desventajas de instrumentos analógicos y digitales*. Recuperado en Agosto de 2016, de https://todoingenieriaindustrial.files.wordpress.com/2012/10/3-4-diferencia-ventajas-y-desventajas-de-instrumentos-analc3b3gicos-y-digitales.pdf

Higuera, J. A. (s.f.). *Universidad Autonoma de Madrid.* Recuperado en Agosto de 2016, de Instrumentación: https://www.uam.es/personal_pas/patricio/trabajo/segainvex/electronica/proyectos/curso_instrumentacion/instrumentacion1.pdf

Jaffery, Z. A., & Dubey, A. K. (2013). Architecture of noninvasive real time visual monitoring system for dial type measuring instrument. *IEEE Sensors Journal*, 1236-1244.

Jaramillo, M., Fernández, Á., & Martínez de Salazar, E. (2010). Implementación del Detector de Bordes de Canny sobre Redes Neuronales Celulares. *Universidad de Extremadura*. Recuperado en Agosto de 2016, de http://eii.unex.es/profesores/jalvarof/pdf/canny00.pdf

Kolman, B., & Hill, D. (2006). *Algebra Lineal* (Octava ed.). México, D.F.: Pearson Prentice Hall.

Lajara Vizcaíno, J. R., & Pelegrí Sebastiá, J. (2011). *Labview: entorno gráfico de programación.* España: MARCOMBO, S.A.

Larson, C. (21 de Mayo de 2014). *LabVIEW Web Services - Web App*. Recuperado en Septiembre de 2016 de LabVIEW Web Services - Web App: http://chrislarson.me/blog/labview-web-services-web-app.html

Llanos, J. (2012). Método para la Generación de Pérfiles de Demanda en Comunidades Aisladas y Predicción de Demanda de Corto Plazo, para Micro-Redes Basadas en Energías Renovables. Santiago de Chile: Repositorio Universidad de Chile. Recuperado en Septiembre de 2016, de http://repositorio.uchile.cl/handle/2250/102768

Loaiza Quintana, A. F., Manzano Herrera, D. A., & Múnera Salazar, L. E. (2012). Sistema de visión artificial para conteo de objetos en movimiento. *El Hombre y la Máquina*, 87-101. Recuperado en Agosto de 2016, de http://www.redalyc.org/articulo.oa?id=47826850010

Logitech. (2011). *Logitech HD PRO Webcam C920.* Recuperado en Agosto de 2016, de file:///C:/Users/PC/Downloads/robotinocameraproductinfo%20(2).pdf

Logitech. (2016). *Asistencia de Logitech*. Recuperado en Agosto de 2016, de HD Pro Webcam C920: http://support.logitech.com/es_mx/product/hd-pro-webcam-c920

Loja Bravo, C. F. (2015). Repositorio Universidad Politécnica Saleciana. *Inspección de Soldadura MIG/MAG de Piezas Metálicas Utilizando Técnicas de Visión Artificial y Procesamiento de Imágenes*. Cuenca. Recuperado en Agosto de 2016, de http://dspace.ups.edu.ec/bitstream/123456789/8336/1/UPS-CT004928.pdf

López Sandoval, A. E., Mendoza Martínez, C., Reyes Cruz, L. Á., Rivas Araiza, E., Ramos Arreguín, J. M., & Pedraza Ortega, J. C. (Mayo de 2015). Sistema de Autenticación Facial mediante la Implementación del algoritmo

PCA modificado en Sistemas embebidos con arquitectura ARM. *La Mecatrónica en México, 4*, 53-64. Recuperado en Agosto de 2016, de http://www.mecamex.net/revistas/LMEM/revistas/LMM-V04-N02-02.pdf

López Ulloa, C., & Miranda Salvatierra, X. S. (2015). Repositorio Escuela Politécnica del Litoral. *Prototipo de un Sitema Embebido Configurable para la Adquisición y Monitoreo de Datos utilizando una Tarjeta de Desarrollo BeagleBone Black de Texas Instruments aplicado a la Agricultura.* Guayaquil. Recuperado en Agosto de 2016, de https://www.dspace.espol.edu.ec/retrieve/89476/D-84638.pdf

Machado Da Silva, R. P. (2016). Repositorio Univercidad Politécnica de Valencia. *Diseño de una Plataforma Software para el Control de Orientación 3D de un Sistema Quadrotor Basado en Raspberry PI 2*. Valencia, España. Recuperado en Agosto de 2016

Maini, R., & Aggarwal, H. (2009). Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP), 3*, 1-11.

Martín, B., & Sanz, A. (2006). *Redes neuronales y sistemas borrosos.* Madrid: Alfaomega.

Martínez Libreros, M. E., & Potes Blandón, J. (2013). Repositorio Universidad Autónoma de Occidente. *Diseño e Implementación de un Sistema de Visión Artificial para Detectar una Especie de Arácnidos y Nuevos Brotes de Flora en un Hábitat Alterado por un Incendio*. Santiago de Cali. Recuperado en Septiembre de 2016, de https://red.uao.edu.co/bitstream/10614/5274/1/TEK01656.pdf

MathWorks. (2016). *Documentation imgaussfilt*. Recuperado en Agosto de 2016, de http://www.mathworks.com/help/images/ref/imgaussfilt.html

Mc Graw Hill Education. (s.f.). *Protocolo TCP/IP.* Recuperado en Agosto de 2016, de http://assets.mheducation.es/bcv/guide/capitulo/8448199766.pdf

Medrano Chimborazo, L. X., & Ramos Cárdenas, S. E. (Mayo de 2011). Repositorio Escuela Politécnica Nacional. *Diseño e Implementación de un Prototipo de Red Combinada de Video Vigilancia Utilizando Tecnología BPL, ETHERNET para el Laboratorio LTI.* Quito. Recuperado en Agosto de 2016, de http://bibdigital.epn.edu.ec/bitstream/15000/3797/1/CD-3577.pdf

Meister, B. (2007). *PuTTY/Cygwin Tutorial.* Recuperado en Agosto de 2016, de http://www.cs.dartmouth.edu/~campbell/cs50/putty-cygwin-tutorial.pdf

Microsoft. (2016). *Microsoft Store*. Recuperado en Agosto de 2016 de Raspberry Pi 3 Board and 16GB, 10 class, with NOOBS:

https://www.microsoftstore.com/store/msusa/en_US/pdp/Raspberry-Pi-3-Board-and-16GB-10class-with-NOOBS/productID.334851400

Microsoft. (2016). *Windows IoT*. Recuperado en Agosto de 2016 de Raspberry PI 2 & 3 Pin Mappings: https://developer.microsoft.com/en-us/windows/iot/docs/pinmappingsrpi

Mikoluk, K. (27 de Diciembre de 2013). *Udemy, Inc.* Recuperado en Septiembre de 2016, de udemy blog: https://blog.udemy.com/tutorial-de-xampp-como-usar-xampp-para-ejecutar-su-propio-servidor-web/

Molloy, D. (2014). *Exploring BeagleBone.* Recuperado en Agosto de 2016, de The BeagleBone Black: http://exploringbeaglebone.com/wp-content/uploads/2014/12/BeagleBonePoster.pdf

Molloy, D. (2014). *Exploring BeagleBone: Tools and Techniques for Buildong with Embedded Linux.* John Wiley & Sons.

Muñoz Manso, R. (Julio de 2014). Repositorio Universidad de Valladolid. *Sistema de visión artificial para la detección y lectura de matrículas*. Valladolid. Recuperado en Agosto de 2016, de https://uvadoc.uva.es/bitstream/10324/11848/1/TFG-P-165.pdf

National Instruments. (04 de Marzo de 2013). *National Instruments*. Recuperado en Septiembre de 2016, de An Open Invitation -- Open Source Software in LabVIEW: http://www.ni.com/white-paper/3008/en/

National Instruments. (Junio de 2013). *National Instruments*. Recuperado en Septiembre de 2016, de Components of a Web Service (Real-Time, Windows): http://zone.ni.com/reference/en-XX/help/371361K-01/lvconcepts/webservices_components/#static_files_ws

National Instruments. (2013). *Perspectiva de los sistemas embebidos 2013.* Recuperado en Agosto de 2016, de ftp://ftp.ni.com/pub/branches/latam/outlook/12775_ESO_Outlook_IA.pdf

National Instruments. (2016). *National Instruments*. Recuperado en Septiembre de 2016, de National Instruments: http://www.ni.com/labview/

National Instruments. (2016). *NI Smart Cameras for Machine Vision*. Recuperado en Agosto de 2016, de http://www.ni.com/white-paper/6582/en/

NATIONAL INSTRUMENTS. (2016). *Using LabVIEW with TCP/IP and UDP.* Recuperado el Agosto de 2016, de http://zone.ni.com/reference/en-XX/help/371361J-01/lvconcepts/using_labview_with_tcp_ip_and_udp/

ODVA. (2016). *The Common Industrial Protocol (CIP)*. Obtenido de https://www.odva.org/Technology-Standards/Common-Industrial-Protocol-CIP/Overview

OpenCV. (25 de Junio de 2014). *The OpenCV Reference Manual.* Recuperado en Agosto de 2016, de http://docs.opencv.org/3.0-beta/opencv2refman.pdf

OpenCV. (2016). *Canny Edge Detector*. Recuperado en Agosto de 2016, de http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

Pacheco Sánchez, J. A. (Junio de 2011). Repositorio Pontificia Universidad Javeriana. *Sistema de Reconocimiento de Objetos Removidos de una Escena, Utilizando Visión por Computador.* Bogotá. Recuperado en Agosto de 2016, de http://repository.javeriana.edu.co/bitstream/10554/7073/1/tesis535.pdf

Paguay Paguay, A. R., & Urgilés Ortiz, P. R. (Febrero de 2012). Repositorio Universidad Politécnica Salesiana. *Recuperación de Imágenes Mediante Extracción de Blobs Aplicando el Operador Laplaciano de Gauss y el Kernel Gaussiano y Desarrollo de un Prototipo*. Cuenca. Recuperado en Agosto de 2016, de http://dspace.ups.edu.ec/bitstream/123456789/1710/15/UPS-CT002312.pdf

Parra Ramos, C., & Regajo Rodríguez, D. (2006). Repositorio Universidad Carlos III de Madrid. *Reconocimiento Automático de Matrículas*. Madrid, España. Recuperado en Agosto de 2016, de http://www.it.uc3m.es/jvillena/irc/practicas/06-07/14.pdf

PCE Instruments. (2016). *PCE-VS Rotameter*. Recuperado en Agosto de 2016, de http://www.industrial-needs.com/technical-data/rotameter-pce-vs.htm

Ponce, C. P. (2010). *Inteligencia Artificial con Aplicciones a la Ingeniería.* México D.F.: Alfaomega.

Prellezo Gutiérrez, J. M. (Octubre de 2002). *Perl 5.0 Un lenguaje multiuso.* Recuperado en Septiembre de 2016, de Perl 5.0 Un lenguaje multiuso: http://ibon.unizar.es/admin/doc/perl/iespana.es/Perl.pdf

Prieto, C., Febres, J., Cerrolaza, M., & MIquelarena, R. (2010). Sistema de Visión Artificial para el Control de Movimiento de un Asistente Robótico Médico. *Mecánica Computacional, 29*, 6619-6629. Recuperado en Agosto de 2016, de http://www.cimec.org.ar/ojs/index.php/mc/article/viewFile/3473/3390

Pucha Ortíz, R. S. (Abril de 2016). Repositorio Escuela Politécnica Nacional. *Diseño de un Sistema Prototipo para la Transmisión de Imágenes de Estaciones Meteorológicas a través de la Red Celular para Brindar*

Soporte al Personal Técnico de Meteorología del INAMHI. Quito. Recuperado en Agosto de 2016, de https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&ved=0ahUKEwi3_dn3yLDPAhUHrD4KHSUpDXcQFghtMAk&url=http%3A%2F%2Fbibdigital.epn.edu.ec%2Fbitstream%2F15000%2F15179%2F1%2FCD-6954.pdf&usg=AFQjCNFhkUU1ClwNqQ6LpUYT-0neB3t9aQ&sig2=sPknOU60eO8fb

Ramos, M. A. (2015). *Repositorio Universidad Nacional del Nordeste.* Recuperado en Agosto de 2016, de Introducción a los sistemas de tiempo real.: http://exa.unne.edu.ar/informatica/SO/RamosAgustina-TpSO.pdf

Raspberry Pi Foundation. (2016). *Raspberry Pi 3 Model B*. Recuperado en Agosto de 2016, de https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Foundation. (2016). *raspberypi.org*. Recuperado en Agosto de 2016, de Downloads: https://www.raspberrypi.org/downloads/

Rivas, R. R. (2007). *Ergonomía en el diseño y la producción industrial.* Buenos Aires: Nobuko.

Rodríguez Villoria, P. (Mayo de 2013). Repositorio Universidad de Valladolid. *La docencia de visión artificial en el grado en Ingeniería en Electrónica y Automática Industrial.* Valladolid, España. Recuperado en Agosto de 2016, de https://uvadoc.uva.es/bitstream/10324/6284/1/PFC-P-86.pdf

Rodríguez, J. G., & Hernández, M. (2013). *Informática industrial.* Monterrey: Editorial Digital del Tecnológico de Monterrey.

RS. (2016). *Raspberry Pi 3 Model B.* Recuperado en Agosto de 2016, de https://www.inet.se/files/pdf/1974044_0.pdf

Sada Pezonaga, S., & Sanz Delgado, J. A. (25 de Junio de 2015). Repositorio Universidad Pública de Navarra. *Tratamiento de imágenes con ruido impulsivo mediante reglas difusas y algoritmos genéticos*. Pamplona. Recuperado en Agosto de 2016, de Tratamiento de imágenes con ruido impulsivo mediante reglas difusas y algoritmos genéticos: http://academica-e.unavarra.es/bitstream/handle/2454/19033/MemoriaTFG--SergioSada.pdf?sequence=1&isAllowed=y

Sánchez Reinoso, J. S. (2011). Repositorio Universidad Politécnica Salesiana. *Identificación de objetos en tiempo real utilizando técnicas y clasificadores de visión artificial para el reconocimiento de patrones.* Cuenca, Ecuador. Recuperado en Agosto de 2016, de http://dspace.ups.edu.ec/bitstream/123456789/1437/14/UPS-CT002184.pdf

SEEED WIKI. (Julio de 2016). *Raspberry Pi 3 Model B*. Recuperado en Agosto de 2016, de http://wiki.seeedstudio.com/wiki/Raspberry_Pi_3_Model_B

Shah, S. (20 de Mayo de 2014). Real-Time Image Processing on Low Cost Embedded Computers. *Techincal report No. UCB/EECS-2014.* California: University of California, Berkeley College of Engineering. Recuperado en Agosto de 2016, de https://pdfs.semanticscholar.org/80b3/bf9210d55fb2b5086d5f1e39351ee9be5a3c.pdf

SIEMENS. (Abril de 2012). *Industrial Wireless Communication.* Recuperado en Agosto de 2016, de http://w3app.siemens.com/mcms/infocenter/dokumentencenter/sc/ic/Documentsu20Brochures/BR_IWLAN_042012_es_Web.pdf

Sierra Álvarez, S. (2012). Repositorio Universidad EAFIT. *Sistema de Medición de Objetos Basa en Visión Artificial*. Medellín. Recuperado en Agosto de 2016, de https://repository.eafit.edu.co/xmlui/bitstream/handle/10784/4518/Santiago_SierraAlvarez_2012.pdf?sequence=1&isAllowed=y

Sobrado Malpartida, E. A. (2003). Repositorio Pontificia Universidad Católica del Perú. *Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot*. Lima, Perú. Recuperado en Agosto de 2016, de http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/68/SOBRADO_EDDIE_VISION_ARTIFICIAL_BRAZO_ROBOT.pdf;jsessionid=E83BAA44BA7AFE58307EB5B39FFC2928?sequence=2

Torres Faría, D. (11 de Junio de 2009). Universidad Simón BOlívar. *Protocolos de Internet*. Venezuela. Recuperado en Agosto de 2016

Travis, J. (2004). *Jeffrey Travis Studios: software*. Obtenido de Jeffrey Travis Studios: software: http://jeffreytravis.com/lost/

Universida Nacional del Centro de la Provincia de Buenos Aires. (s.f.). *El modelo OSI.* Recuperado en Agosto de 2016, de http://www.exa.unicen.edu.ar/catedras/comdat1/material/ElmodeloOSI.pdf

Universida Técnica del Norte. (s.f.). *Protocolos TCP/IP*. Recuperado en Agosto de 2016, de http://repositorio.utn.edu.ec/bitstream/123456789/1096/5/04%20ISC%20019%20Capitulo%202%20TCP%20IP.pdf

Universidad Nacional de Quilmes. (Agosto de 2005). *Iluminación para las aplicaciones de Visión Artificial.* Recuperado en Agosto de 2016, de Repositorio Universidad Nacional de Quilmes:

http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Tipos%20de%20Iluminaci%C3%B3n.pdf

Universidad Tecnológica de la Mixteca. (s.f.). *Sistemas de Comunicaciones - Redes II.* Recuperado en Agosto de 2016, de Ethernet y Protocolos TCP/IPv4: http://mixteco.utm.mx/~resdi/historial/materias/IPv4.pdf

Vacca, C., Scaglia, G., Serrano, M., Godoy, S., & Mut, V. (2014). Sensado y Control de Caudal por Visión Artificial. *Biennial Congress of Argentina*, 84-89.

Valencia López, J. M., & Abril Cañas, M. (2007). Repositorio Universidad Tecnológica de Pereira. *Registro de transeúntes en tiempo real utilizando un sistema de visión artificial sobre un ambiente controlado*. Pereira. Recuperado en Agosto de 2016, de http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/90/621367V152.pdf?sequence=3&isAllowed=y

Vargas Soria, A. E. (Febrero de 2015). Repositorio Universidad Técnica de Ambato. *Sistema Embebido de Movilización y Posicionamiento para Personas No Videntes mediante Hardware Libre.* Ambato, Ecuador. Recuperado en Agosto de 2016, de http://repo.uta.edu.ec/bitstream/123456789/8653/1/Tesis_t974ec.pdf

WIKA. (2013). *Gauge Failures 8 Most Common Reasons.* Recuperado en Agosto de 2016, de http://www.wika.us/upload/BR_Gauge_Failures_Brochure_en_us_64836.pdf

WIKA. (2016). *Instrumentación mecánica de presión*. Recuperado en Agosto de 2016, de http://www.wika.es/232_34_233_34_es_es.WIKA

WIKA. (2016). *Instrumentación mecánica de temperatura*. Recuperado en Agosto de 2016, de http://www.wika.es/54_es_es.WIKA?ProductGroup=5260

WIKA. (2016). *Measurement Solutions*. Recuperado en Agosto de 2016, de Is a pressure gauge important to everyday operations.: http://www.wika.us/solutions_pressure_gauge_importence_to_operations_en_us.WIKA

WIKA. (2016). *Mechanical temperature measurement*. Recuperado en Agosto de 2016, de http://www.wika.us/service_faq_tm_en_us.WIKA#157

WIKIPEDIA. (Junio de 2016). *PuTTY*. Recuperado en Agosto de 2016, de https://es.wikipedia.org/wiki/PuTTY

WIKIPEDIA. (2016). *Raspberry Pi*. Recuperado en Agosto de 2016, de https://es.wikipedia.org/wiki/Raspberry_Pi

Wikipedia. (2016). *Wikipedia*. Recuperado en Agosto de 2016, de OpenCV: https://en.wikipedia.org/wiki/OpenCV

Yang, Z., Niu, W., Peng, X., Gao, Y., Qiao, Y., & Dai, Y. (2014). An image-based intelligent system for pointer instrument reading. *2014 4th IEEE International Conference on Information Science and Technology*, 780-783.

Zhang, B., Huang, W., Li, J., Zhao, C., Fan, S., Wu, J., & Liu, C. (2014). Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: A review. *Food Research International, 62*, 326-343.

ANEXOS

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor LEANDRO GABRIEL CORRALES TIBÁN y la señorita CATHERINE LISETH GÁLVEZ JÁCOME.

En la ciudad de Latacunga, a los 16 días del mes de enero del 2017

Ing. Edwin Pruna
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO