Universidad de Matanzas
Facultad de Ciencias Técnicas

SISTEMA DE MONITOREO DE BAJO COSTO PARA PROCESOS Y SISTEMAS MECÁNICOS.

Tesis Presentada como Requisito Parcial
para la Obtención del Título de
Máster en Ingeniería Asistida por Computadora

Autor: Ing. Alberto Villalonga Jaén

Tutores: Dr.C. Rodolfo E. Haber Guerra

M.Sc. Gerardo Beruvides López

Matanzas, 2017

DECLARACIÓN DE AUTORIDAD Y NOTA LEGAL

Yo, Alberto Villalonga Jaén, declaro que soy el único autor de la siguiente tesis, titulada Sistema de monitoreo de bajo costo para procesos y sistemas mecánicos y, en virtud de tal, cedo el derecho de copia de la misma a la Universidad de Matanzas, bajo la licencia *Creative Commons* de tipo *Reconocimiento No Comercial Sin Obra Derivada*, con lo cual se permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento de sus autores, no haga uso comercial de la obra y no realice ninguna modificación de ella.

Matanzas, 3 de enero de 2017.

Alberto Villalonga Jaén

RESUMEN

El diseño e implementación de sistemas de monitoreo es de vital importancia para la economía puesto que estos permiten explotar de una manera más eficiente cualquier proceso productivo. Esta importancia se ve comprometida de cierta manera por los altos precios que poseen los elementos necesarios para la implementación de dichos sistemas. En este trabajo se brinda un alternativa a estos elevados precios pues se presenta un sistema de monitoreo flexible, específicamente para procesos mecánicos, de bajo costo basado en una Raspberry Pi 2 modelo B, configurable, con comunicación Ethernet, que permite la conexión de sensores con los estándares del mercado y con el uso de una aplicación (multiplataforma) para ordenador desarrollada en Qt que actúa como servidor y permite la monitorización del proceso, así como el almacenamiento de datos, con persistencia, para el posterior análisis de los mismos.

Palabras claves: Sistemas de Monitoreo, Raspberry Pi, Arquitectura Flexible.

ABSTRACT

Designing and implementing monitoring systems are very important for obtaining economical profits because they allow to exploit in a more efficiently way any productive process. Sometimes, the high prices of the necessary components of the monitoring systems make difficult their implementation. This work proposes, as an alternative for those high prices, a low cost and flexible monitoring systems for mechanical processes and systems based on a Raspberry Pi 2 model B, which allows to connect standards sensors, with Ethernet communication through client/server architecture and a Qt application (multiplatform) used as server for process monitoring in real time and data storage to realize later analysis.

.**Keyworks**: Monitoring Systems, Raspberry Pi, Flexible Architecture.

TABLA DE CONTENIDO

Introducción 1

Capítulo 1. Generalidades 4

1.1 Antecedentes y estado del arte de los sistemas de monitoreo 4

1.1.1 Antecedentes. Principales conceptos 4

1.1.2 Estado del arte de los sistemas de monitoreo. 7

1.2 Elección del ordenador de placa reducida y el sistema operativo. 11

1.2.1 Elección del sistema operativo para Raspberry-Pi 13

1.3 Elección del lenguaje de programación y del entorno de desarrollo. 14

1.3.1 Evolución histórica de los lenguajes de programación. 14

1.3.2 Criterios de comparación 17

1.3.3 Elección del lenguaje a utilizar 23

1.3.4 Elección del entorno de programación. 24

1.4 Conclusiones parciales del capítulo 25

Capítulo 2 Arquitectura del sistema 26

2.1 Arquitectura de *hardware* 26

2.1.1 Sistema de acondicionamiento de señales 26

2.1.2 Raspberry-Pi 28

2.1.3 Servidor 29

2.2 Arquitectura de software 30

2.2.1 Requerimientos del software 30

2.2.2 Características del sistema 30

2.2.3 Contexto del sistema 32

2.2.4 Cliente 32
2.2.5 Servidor 34
2.3 Protocolo de comunicación 36
2.4 Bases de datos 37
2.5 Conclusiones parciales del capítulo 39
Capítulo 3 Configuración, monitoreo y pruebas del sistema 40
3.1 Aplicación Servidor 40
3.1.1 Configuración del sistema 42
3.1.2 Visualización de las variables de proceso. 44
3.1.4 Visualización de alarmas. 46
3.1.4 Visualización de históricos. 46
3.2 Aplicación Cliente 47
3.2.1 Configuración del sistema 49
3.2.2 Visualización de las variables de proceso. 50
3.2.3 Visualización de alarmas. 52
3.3 Validación del sistema 52
3.3.1 Pruebas de caja negra 54
3.4 Conclusiones parciales del capítulo 55
Conclusiones 57
Recomendaciones 58
Referencias Bibliográficas 59

INTRODUCCIÓN

La realización un proceso industrial requiere no sólo administrar la mano de obra, la materia prima y la maquinaria de la planta, sino también disponer de la información necesaria para la toma de decisiones. Por eso es importante poseer una herramienta que permita visualizar, almacenar y manejar la información del proceso industrial en cuestión.

Las computadoras son herramientas capaces de almacenar, procesar y presentar información en forma atractiva y confiable, por lo que se ha fomentado como tendencia en las industrias modernas el asociar sus procesos automatizados a programas que posean un ambiente en el cual el usuario pueda tener acceso para monitorear y modificar los distintos elementos que conforman su sistema de control.

El correcto aprovechamiento del conocimiento e información disponible sobre el proceso permitirá la evaluación automatizada, continuada y en línea, del proceso de una forma objetiva. Se garantiza de esta manera una uniformidad en la decisión, independiente de las apreciaciones subjetivas de los diferentes especialistas, facilitando la detección de situaciones anómalas y su diagnóstico a través de un seguimiento continuo de las variables de proceso.

Los sistemas de monitoreo son de vital importancia para el funcionamiento de cualquier proceso productivo pues nos brinda la posibilidad de comprender de una manera detallada su dinámica de trabajo lo que posibilita que se realicen correcciones sobre sus principales parámetros de ajuste con la meta de lograr un funcionamiento más eficiente.

En Cuba la implementación de estos sistemas ha posibilitado alcanzar nuevos niveles de calidad y producción, proporcionado grandes avances en la industria cubana. Aunque a veces su uso se ve limitado debido a que casi en su totalidad, los sistemas de monitoreo que se montan en nuestro país son foráneos, lo que trae consigo una gran inversión debido a sus alto precios en el mercado más los costos de importación, lo que a veces condiciona que algunas empresas que no consten con el presupuesto necesario y prescindan de su utilización.

Una de las principales líneas que ha seguido Cuba para garantizar su desarrollo industrial y tecnológico, y hacer frente a situaciones como esta, ha sido el desarrollo de tecnologías propias basadas en su mayoría en la utilización de software y hardware libre. Siguiendo esta línea de investigación este trabajo está encaminado al diseño de un sistema de monitoreo, que posea como base un ordenador de placa reducida (SBC por sus siglas en ingles), de distribución libre, así como la programación de las aplicaciones, basadas en software libre, que se ejecutaran sobre el SBC y el servidor del sistema y que contenga las funcionalidades necesarias para su implementación en el monitoreo de sistemas y procesos mecánicos. Para ello se debe realizar una búsqueda bibliográfica sobre los sistemas de monitoreo de bajo costo basados en ordenadores de placa reducida, lo cual permite llevar a cabo un correcto análisis y diseño de la arquitectura de hardware y software, así como la programación de las funcionalidades del sistema, el cual se debe probar y poner a punta.

El trabajo se encuentra dividido en 3 capítulos, a través de los cuales se expone el proceso de análisis, diseño e implementación del hardware y el software para el sistema de monitoreo. El capítulo 1 hace un análisis bibliográfico con el fin de exponer los antecedentes y estado del arte de los sistemas de monitoreo y la utilización en estos de los

sistemas embebidos principalmente de los ordenadores de placa reducida. Se realiza una comparación entre los ordenadores de placa reducida más populares y los lenguajes de programación más utilizados en la actualidad y a partir de los resultados obtenidos en ambas se selecciona un lenguaje, un entorno de desarrollo, un ordenador de placa reducida y el sistema operativo sobre el cual correrán las aplicaciones del sistema. El capítulo 2 aborda el proceso de análisis y diseño de software, donde se definen los requisitos del sistema, se describe el protocolo de comunicación utilizado para el intercambio de aplicación entre el SBC y el servidor. Además, se expone el diseño de las bases de datos del sistema. En el capítulo 3 se muestra la implementación de las distintas funcionalidades del software y se comentan los resultados de las pruebas realizadas al software y al sistema en su conjunto.

CAPÍTULO 1. GENERALIDADES

En este capítulo se abordan los antecedentes y el estado actual de los sistemas de monitoreo, haciéndose énfasis en los sistemas de monitoreo basados en sistemas embebidos. Se presentan las principales características de los ordenadores de placa reducida más populares en el mercado y se justifica la selección del utilizado este trabajo, así como el sistema operativo sobre el cual se ejecutarán las aplicaciones. Se hace una comparación entre los lenguajes de programación orientado a objetos más populares y entre distintos entornos de desarrollo integrado (IDE) y como resultado se justifica la selección de ambos para el desarrollo de las aplicaciones.

1.1 Antecedentes y estado del arte de los sistemas de monitoreo

1.1.1 Antecedentes. Principales conceptos

Un proceso se define como una secuencia, u orden definido, de actividades químicas, físicas o biológicas que se llevan a cabo para la conversión, transporte o almacenamiento de material o energía. Para la obtención de un producto deseado y de acuerdo con las especificaciones exigidas se hace necesario que el proceso siga un conjunto de ordenes la cuales se encuentran supeditadas a la selección de las materias primas y los parámetros del proceso. Las exigencias que se imponen a los procesos productivos en cuestión de rendimiento, calidad y flexibilidad hacen necesario la introducción de tecnologías que permitan la vigilancia y el control de éstos, para suplir estas necesidades se crearon los sistemas de monitorización.

La monitorización es la determinación de las condiciones de funcionamiento de un sistema en tiempo real. Permite detectar las posibles interferencias o fallas que pudieran presentarse en el curso de alguna acción y lo que posibilita corregir el procedimiento antes de obtener un resultado final.

Los sistemas de monitoreo son aquellos que permiten la vigilancia del proceso, dotando al operario de los mecanismos necesarios para su alerta, así como la interacción amigable con el proceso y el registro de su evolución.

Los primeros sistemas de monitoreo eran algo rústicos y se basaban en la creación de sinópticos del proceso realizados en maquetas. Para la visualización de los parámetros del proceso se utilizaban lámpara y *displays*, por lo que se necesitaba de un conocimiento profundo del proceso para su entendimiento.

Posteriormente lo avances experimentados en el campo de la electrónica tales como el desarrollo de los microprocesadores y microcontroladores permitió el monitoreo a través de tarjetas inteligentes la cuales podían comunicarse a una mayor velocidad con los elementos del campo, así como que la presentación de los datos y la interacción con el usuario fuera de una manera más amigable a través de la utilización de teclados y pantallas alfanuméricas y gráficas.

Luego con la evolución de los sistemas de cómputo y las comunicaciones se crearon sistemas de monitoreo basados en *interfaces* hombre maquina (HMI por sus siglas en inglés) los cuales son robustos y de fácil configuración. También surgen los sistemas de monitoreo basados en ordenadores los cuales posibilitan la utilización de herramientas de software de sencilla configuración, con prestaciones muy variadas y que permiten la

comunicación con el proceso a través de una amplia gama de buses y redes. En la Fig. 1.1 se muestra un ejemplo de pantalla de un sistema de monitoreo basado en ordenador.

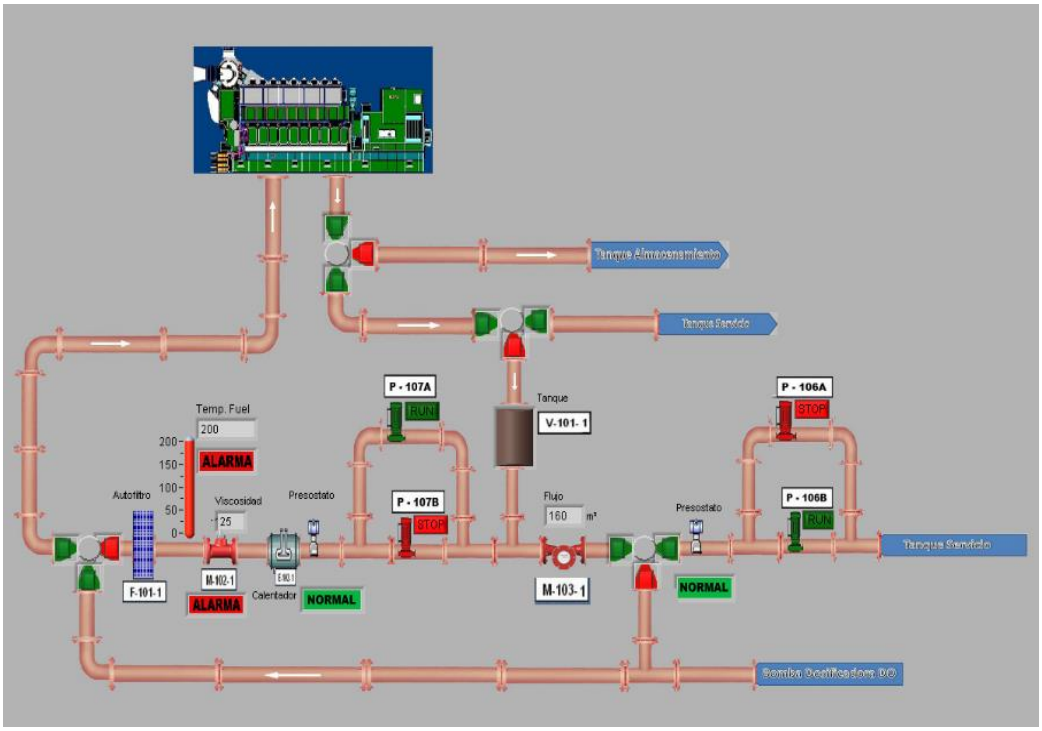

Figura 1.1 Pantalla de un sistema de monitoreo basado en ordenador.

En la actualidad los sistemas de monitoreo poseen dos funcionalidades básicas:

- Adquisición y registro de datos.
- Representación del Proceso mediante:
  - Creación de Sinópticos
  - Representación de Alarmas
  - Gráficas y Tendencias

- Históricos y Bases de Datos

1.1.2 Estado del arte de los sistemas de monitoreo.

El creciente desarrollo industrial que ha experimentado el mundo en nuestros días ha despertado un gran interés en el estudio de los sistemas de monitoreo ya que estos permiten un aumento gradual en los niveles de producción y los estándares de calidad, además de garantizar una mayor seguridad para los operadores de procesos. El crecimiento de estos ha venido aparejado también con el vertiginoso desarrollo de la electrónica y los sistemas de cómputo con los cuales posee una estrecha relación (Villalonga, *et al.*, 2016).

Incorporado a ello, el creciente impulso experimentado por las tecnologías de la informática y las comunicaciones y el incremento en las investigaciones en áreas como las redes de comunicaciones industriales, los sensores, la teoría de control, los softwares de ingeniería y la ciencia de la computación, ha propiciado la aparición de nuevos retos de investigación tales como la abstracción en sistema de tiempo real, robustez, modelación y control de sistemas híbridos, control sobre redes, redes de sensores y actuadores entre otros (Villalonga, *et al.*, 2016), que han permitido que los sistemas de monitoreo alcancen nuevos niveles de calidad.

Desde sus inicios las tecnologías utilizadas para el diseño e implementación de sistemas de monitoreo han estado dominadas por grandes transnacionales las cuales proponen equipamientos y soluciones *hardware* robustos y dedicados, sobre todo a ambientes industriales, garantizando así altos estándares de calidad y seguridad, aunque con elevados

costos. Además, casi todos los softwares que se comercializan son licenciados lo que añade un gran costo adicional a la implementación de dichos sistemas.

En la actualidad el creciente desarrollo experimentado por los sistemas embebidos, así como la robustez que se ha alcanzado en sus diseños ha permitido que se presenten como una alternativa para el diseño de sistemas de monitoreo en aplicaciones donde la seguridad y la robustez no sean vitales, debido a sus bajos precios, así como sus elevadas capacidades de cómputo y bondades para incorporarse a redes de datos ya sea alámbricas o inalámbricas. Entre los principales sistemas embebidos que se utilizan para el monitoreo encontramos los ordenadores de placa reducida, los sistemas basados en microcontroladores y los arreglos de compuertas programables (FPGA por sus siglas en inglés).

En la industria actual se puede constatar la aplicación de los sistemas embebidos sobre todo en procesos donde la utilización de redes inalámbricas de sensores es necesario, debido a las difíciles condiciones de acceso en el proceso que dificulta el cableado, mediante el uso de *WiFi*, explotando así sus potencialidades de conexión (Priyanka y Mahesh, 2016; Singh, *et al.*, 2016; Jadhav y Hambarde, 2016)(Rawat, *et al.*, 2016). También se utilizan en sistemas donde la trasmisión inalámbrica se realiza a través del protocolo *ZigBee* (Abinath, *et al.*, 2015; Peng y Wan, 2013; Wen-Tsai y Yao-Chi, 2011), demostrando asi sus grandes potencialidades que le permiten acolparse a estas redes sin la utilizacion de interfaces que aumentarian el costo del sistema. La utilización de la combinación de ambos protocolos es muy frecuente y debido al uso de sistemas embebidos su integración no es complicada, (Darshini y Esakki, 2016) proponen una solución en la cual utiliza ambos sistemas de transmisión para la construcción de un robot que monitoree las condiciones ambientales en

minas para poder predecir posibles accidentes y así salvar vidas humanas y (Bharathi y Mubeen, 2016) utilizan su combinación para el diseño de un sistema de monitoreo de nivel y temperatura, a través del uso de un ordenador de placa reducida, logrando altos nivel de confiabilidad. En otras aplicaciones inalámbricas que utilizan como base para la trasmisión la radio frecuencia es también muy común el uso de estos sistemas, (Raguvaran y Thiyagarajan, 2015; Sachan, 2012; Calvo, *et al.*, 2016) asi lo evidencian.

En la industria no solo se explotan las posibilidades de integración de estos en las redes inalámbricas sino también en sistemas con redes alámbricas. (Kumar y Rameshkumar, 2013; Patel, *et al.*, 2015; Zhanitta y Sathya, 2015) hacen uso de sistemas embebidos para el monitoreo utilizando como estándar de conexión *ethernet* y protocolo de comunicación TCP-IP haciendo uso de servidores web para la presentación de la información, otra de sus grandes ventajas.

En otras aplicaciones simplemente se benefician de su portabilidad, capacidad de cómputo y bajos costos (Garcia, 2016; Suresh, *et al.*, 2014; Swaroop, *et al.*, 2015)(Rahim, *et al.*, 2014). Además, se emplean también para combinar la supervisión y el control (Dixi, 2014; Ejiofor y Oladipo, 2013; Hankare, *et al.*, 2015; Suresh, *et al.*, 2014).

El desarrollo en la actualidad de la cuarta revolución industrial, impulsada entre otros factores por el internet de las cosas (*IoT* por sus siglas en inglés)y lo sistemas ciberfísicos (Almada-Lobo, 2015), o sea la nueva generación de sistemas embebidos (Isikdag, 2015), ha permitido alcanzar nuevos parámetros de calidad en la producción debido a la nueva precisión y el desempeño alcanzado por los sistemas de monitoreo en su interacción con los operarios (Deshpande, *et al.*, 2016; Karankumar, *et al.*, 2014). (Rodríguez Molano, *et*

al., 2016) presentan una arquitectura para monitoreo industrial basado en un ordenador de placa reducida haciendo uso del IoT demostrando así las potencialidades de esta tecnología.

Es importante destacar que su uso para sistemas de monitoreo no solo se restringe a ambientes industriales, sino que los podemos encontrar en muchas áreas de la vida cotidiana como son la salud, nuestros hogares, empresas, reportando variables meteorológicas o entre otras muchas áreas.

(Ferdoush y Li, 2014; Meetali, 2016) hacen uso de ordenadores de placa reducida para la medición de variables ambientales en lugares de difícil acceso, y (Tomar y Bhatia, 2015) (Schumann-Bölsche y Schön, 2015) los utilizan en zona de desastres para ayuda humanitaria, confirmando que pueden ser una solución estable y de gran confiabilidad.

Las grandes potencialidades y bondades que presentan para el monitoreo de los distintos sistemas implantadas nuestros hogares, van desde el control de los sistemas de seguridad (Bhagyalakshmi, *et al.*, 2015; Sankaranarayanana, *et al.*, 2014), hasta permitir un control total sobre todos de manera segura y cómoda desde un *smart phone* (Ganesh y A, 2015; Rao y Uma, 2015; Suradkar, et al., 2016; Vujović y Maksimović, 2015).

La medicina es otra área donde los sistemas de monitoreo sin de vital importancia pues permiten medir la evolución de los signos vitales de los pacientes (Biswas, *et al.*, 2016). (Kasundra y Shirsat, 2015) desarrollan un sistema de monitoreo para pacientes basados en *Raspberry-Pi* y microcontroladores que permite obtener a los médicos notificaciones del estado de los pacientes a través de un aplicación en sus *smart phones* o simplemente dejarlas registradas en la base de datos de sus servidores centrales.

Se pueden encontrar también en aplicaciones no tan convencionales de los sistemas de monitoreo como son el monitoreo de trafico de paquetes redes de computadora (Michal, *et al.*, 2014; Turk, *et al.*, 2014), en sistemas de monitoreo y control de tráfico para evitar accidente (Aburva y Jeevabharathi, 2015) o en el cobro de peajes (Jahan, *et al.*, 2013).

1.2 Elección del ordenador de placa reducida y el sistema operativo.

1.2.1 Elección del ordenador de placa reducida

Los ordenadores de placas reducidas son sistemas embebidos, en los cuales en un sólo circuito integrado se incorporan todas las potencialidades de una computadora completa. Su diseño se centra en un sólo microprocesador con la RAM, los dispositivos de entrada-salida y todas las demás características de un computador funcional en una sola tarjeta, que suele ser de pequeño tamaño, y que contienen todo lo que necesita en la placa base.

En la actualidad estos pueden ser agrupados en dos grandes categorías: propietarios y *open source.* Los SBC propietarios son diseñados en su mayoría para aplicaciones específicas, lo que les aporta mayor robustez y calidad. Por otra parte, los *open source* como su nombre lo indica permiten el acceso libre tanto al *hardware* como al *software* que lo compone lo que permiten alcanzar grandes potencialidades a los usuarios con conocimiento avanzados en estas áreas, aunque no se encuentran diseñados para una aplicación específica.

En general utilizan variadas arquitecturas de *hardware* basados en procesadores de computadoras personales convencionales (Intel, AMD) u otros como los ARM los cuales tradicionalmente han sido utilizados en aplicaciones industriales y más recientemente en sistemas móviles.

Tabla 1.1 Características principales de los SBC más populares del mercado

	Raspberry Pi 2 model B	*Beagle Bone Black*	*OLINUXINO A13*	*RIoTBOARD*	*ODROID XU3*
Procesador	ARM Cortex-A7. 4 núcleos	ARM Cortex-A8. 1 núcleo	ARM Cortex-A8. 1 núcleo	ARM Cortex-A9. 1 núcleo	ARM Cortex-A15. 8 núcleos
Frecuencia procesador	900 MHz	1 GHz	1 GHz	1 GHz	2 GHz
Memoria RAM	1 GB DDR2	512 MB DDR3	512 MB DDR3	1 GB DDR3	2 GB DDR3
USB	4	1	4	4	4
Ethernet	10/100	10/100	10/100	10/100	10/100
WiFi Bluetooth	No	No	No	No	No
GPIO	26	66	142	10	30
Almacenamiento on board	-	4 GB eMMC	-	4 GB Flash	4 GB eMMC
Expanion almacenamiento	microSD	microSD	microSD	microSD y SD	microSD
Precio (€)	37	100	65	120	144

En la Tabla 1.1 se muestra un comparativa entre los SBC más populares del mercado. Como se puede apreciar, todos poseen características similares, algunos con mayor capacidad de procesamiento o GIPO, pero sin duda la Raspberry-Pi 2 model B desataca con respecto a los demás debido a su bajo costo lo que permite que la característica rendimiento-costo sea mejor que la de sus competidores, por lo que es el SBC seleccionado en este trabajo.

1.2.2 Elección del sistema operativo para Raspberry-Pi

Existen varios sistemas operativos que pueden funcionar sobre la arquitectura de Raspberry-Pi, entre los que desatacan varias distribuciones de Linux y Windows 10. Es importante tener en cuenta en la selección del sistema operativo la aplicación o uso específico se desea para el mismo. Entre los requerimientos fundamentales a considerar para el diseño del sistema propuesto se tienen: la ejecución de tareas en tiempo real, la explotación optima de los recursos de hardware y la posibilidad de realizar una presentación amigable de la información al usuario. A continuación, se presenta en la tabla 1.2 un resumen con las distribuciones más utilizadas.

Tabla 1.2

Sistema Operativo	*Descripción*
Raspbian	Distribución del sistema operativo *GNU/Linux* basada en *Debian Whezzy*. Utiliza LXDE como ambiente de escritorio. está optimizada para el hardware de *Raspberry-Pi*, presenta un ambiente de usuario amigable es reconocida como la distribución oficial para *Raspberry-Pi*.
Arch Linux	Distribución de muy buen rendimiento debido a su minimalismo que ofrece control total sobre todos los elementos del sistema operativo lo que permite poder personalizarla, aunque debido a esto se necesitan conocimientos expertos para poder usarla, no presentando un interface amigable al usuario.
RISC OS	Distribución ligera que permite exportar de manera eficiente los recursos de hardware de Raspberry-Pi, aunque no presenta una interfaz amigable para el usuario.

Basándose en los tres criterios anteriormente expuestos para la selección del sistema operativos fue seleccionado para la confección de este trabajo *Raspbian* puesto que a pesar de no poseer un control sobre el hardware tan preciso como las otras distribuciones se

encuentran optimizado para el hardware de *Raspberry-Pi*, además posee una interfaz de usuario amigable y al ser presentada como la distribución oficial posee gran soporte de información.

1.3 Elección del lenguaje de programación y del entorno de desarrollo.

Para la creación del software de este proyecto se debe realizar un estudio de las numerosas alternativas de desarrollo existentes. Desde usar un único lenguaje de alta eficiencia como C++ y basarse en librerías intermedias para lograr portabilidad, hasta la utilización de lenguajes que no dependen en absoluto de la máquina de ejecución como Java. Sin embargo, no se puede perder de vista que el objetivo de la aplicación es obtener alta eficiencia y precisión en los cálculos y en la trasmisión de información. Hoy en día los lenguajes de programación más populares, para el desarrollo de este tipo de aplicaciones, son Java, C++ y C#.

1.3.1 Evolución histórica de los lenguajes de programación.

Lenguaje C++.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. Debido a la particularidad de su origen es calificado como un lenguaje hibrido atendiendo al punto de vista de los lenguajes orientados a objetos

En 1985 Stroustrup publica la primera edición del libro "*The C++ Programming Language*" que sirvió de estándar informal y texto de referencia. Posteriormente el lenguaje fue estandarizado (ISO C++) y paralelamente son publicadas la segunda y tercera ediciones de modo tal que reflejan estos cambios (Stroustrup, 1997).

Desde sus inicios, C++ intentó ser un lenguaje que incluye completamente al lenguaje C (quizá el 99% del código escrito en C es válido en C++) pero al mismo tiempo incorpora muchas características sofisticadas no incluidas en aquel, tales como: POO, excepciones, sobrecarga de operadores, *templates* o plantillas (Deitel y Deitel, 2013).

Lenguaje Java.

Java originalmente fue denominado Oak. Sus inicios datan de 1991 cuando James Gosling (en Sun Microsystems) encabezó un proyecto cuyo objetivo original era implementar una máquina virtual ampliamente portable y un lenguaje de programación, ambos orientados a dispositivos embebidos, procesadores incorporados en diversos dispositivos de consumo masivo como VCRs, tostadoras, PDAs, teléfonos móviles, entre otros. La sintaxis del lenguaje heredó características de C y C++, explícitamente eliminando aquellas que para muchos programadores (según los diseñadores) resultan excesivamente complejas e inseguras (Schildt, 2014).

Con el auge de Internet, parece natural aprovechar este lenguaje para desarrollar aplicaciones distribuidas y portables. La primera implementación de Java data de 1995 y pronto los navegadores web incorporan soporte Java para la ejecución de pequeñas aplicaciones interactivas (Applets.) En la actualidad su uso es promovido para el desarrollo

de aplicaciones empresariales del lado del servidor, especialmente a través del estándar J2EE, así como en dispositivos móviles (a través del estándar J2ME).

En realidad, Java hace referencia a un conjunto de tecnologías entre las cuales el lenguaje Java es sólo una de ellas. Por tal motivo muchas veces se habla de la plataforma Java, la cual es inseparable del lenguaje.

Sun controla los estándares de Java a través de un mecanismo de apertura parcial denominado el Java Community Process (JCP) (Rivero y Lopez, 2009).

Lenguaje C#.

C# es un lenguaje de programación orientado a objetos creado por Microsoft como parte de su plataforma .NET, con el objetivo de proporcionar a los programadores un soporte completo para el desarrollo de aplicaciones (Sharp, 2013).

C# es una versión avanzada de C y C++ y se ha diseñado especialmente para el entorno .NET. C# es un nuevo lenguaje orientado a objetos, empleado por programadores de todo el mundo para desarrollar aplicaciones que se ejecutan en la plataforma .NET. De todas formas, C# no es parte del entorno .NET. C# es parte de Microsoft Visual Studio. NET 7.0. Es un paso muy importante en la evolución de los lenguajes de programación y es una solución ideal para las aplicaciones de alto nivel. Con C# se puede desarrollar todo tipo de proyectos de aplicaciones cliente / servidor.

C# amplía las capacidades de C, C++, Visual Basic y Java para proporcionar un completo entorno de desarrollo en el que crear aplicaciones. C# mezcla la potencia de C, las

capacidades de orientación a objetos de C++ y la interfaz gráfica de Visual Basic (Dimitrov, *et al.*, 2013).

1.3.2 Criterios de comparación

Comparar lenguajes de programación es una tarea que nunca ha sido sencilla ni objetiva. Teniendo en cuenta los requerimientos del software para el sistema que se desea desarrollar se establecen los siguientes criterios de comparación:

- Portabilidad.
- Matemática de precisión compleja.
- Gestión de memoria.
- Velocidad de ejecución.
- Licencia.
- Eficiencia.

Portabilidad.

El lenguaje C++, aunque no es un lenguaje automáticamente distribuido en los sistemas UNIX, prácticamente todos lo pueden ejecutar ya sea en una variante comercial o mediante el popular GNU GCC/G++ con lo que la disponibilidad está asegurada. En cuanto a su portabilidad, el único inconveniente notorio radica en ciertos problemas (cada vez menos frecuentes) en las implementaciones de la Standard Template Library (STL).

No obstante, lo indicado, el C++ presenta importantes dificultades de portabilidad, particularmente en cuanto a los siguientes aspectos (Rivero y Lopez, 2009):

- Características dependientes de la implementación: Lo que permite realizar fuertes optimizaciones en distintas arquitecturas, resulta con frecuencia una pesadilla para la portabilidad. Muchos detalles importantes son dejados a criterio de quien escribe el compilador, tales como los tamaños de diversos tipos de datos, juegos de caracteres, comportamiento ante ciertos errores.

- Acceso a librerías del sistema operativo: Las interfaces y librerías principales no han seguido un proceso de estandarización tan riguroso como el lenguaje, lo que ha traído como consecuencia diversas soluciones incompatibles para los mismos problemas. Estrictamente este no es un problema del lenguaje, sino más bien de la plataforma utilizada (por ejemplo, las variantes de UNIX).

En ese sentido Java introdujo un enfoque radical (aunque predecible) al diseñar un lenguaje prácticamente sin características dependientes del implementador (potencialmente algo menos eficiente) y con una extensa librería utilitaria cuya interfaz de programación está muy fuertemente estandarizada.

En el caso de C#, al separar los pasos de compilación a código intermedio de ejecución, se tiene la posibilidad de crear entornos de ejecución propios de las diferentes arquitecturas. La plataforma ha tomado la orientación de suministrar un entorno de ejecución (run-time) y un entorno de desarrollo (kit de desarrollo).

Con todo, la portabilidad alcanzada es cualitativamente superior a la que se puede obtener con el lenguaje C/C++ y se consigue de manera automática por cualquier desarrollador.

Matemática de precisión compleja.

En este punto el mejor lenguaje es FORTRAN con diferencia; de hecho, es el único lenguaje con soporte nativo para números complejos. Además, permite usar tipos de datos de enorme precisión (hasta 8 bytes).

Le sigue a cierta distancia C++, mucho más estructurado y moderno, pero menos especializado en este ámbito. Es muy frecuente usar bibliotecas que internamente hacen uso de código FORTRAN.

Java posee una máquina virtual muy limitada en este aspecto y casi todo el peso recae sobre código software, por eso es muy deficitario. Aunque existen bastantes utilidades y bibliotecas de clases para suplir esta característica, dejan bastante que desear.

En el caso de C# el nivel de desarrollo alcanzado es pobre, pero la estructura de la plataforma de programación (.NET), al no definir una máquina virtual como hace java, permite mayores optimizaciones (Rivero y Lopez, 2009).

Gestión de memoria.

C++ es el más eficiente ya que tiene control absoluto de la memoria. Se pueden usar todas las técnicas de manejos de punteros y conteo de referencias.

Java y C# tiene recolectores automáticos de memoria que facilitan enormemente la programación. No es necesario reservarla ni liberarla. Pero esta sencillez de utilización no

es crítica ya que, con un buen diseño y metodología en C++, pueden crearse estructuras complejas con respecto al uso de memoria, pero muy fáciles de usar en el resto del programa. Además, los recolectores automáticos generan una carga de tiempo adicional al procesador afectando el tiempo de ejecución de la aplicación (Rivero y Lopez, 2009).

Velocidad de ejecución.

C++ es bastante eficiente, sobre todo si se trata de un compilador capaz de realizar optimizaciones sobre el código.

Java y C# no generan código nativo, sino para una capa intermedia que necesita ser interpretada/compilada. El enfoque de Java es definir una arquitectura hardware virtual. Una máquina con sus instrucciones y niveles de pila.

C# define un lenguaje intermedio semánticamente mucho más rico, pero con igual capacidad teórica de resolver problemas, lo permite definir instrucciones de más alto nivel. En ambos enfoques durante la ejecución se transforma ese nivel intermedio en código máquina, pero en C#, los resultados experimentales dan mejor rendimiento. No obstante, el tener que traducir el código en esa capa intermedia, una vez más, agrega una carga de tiempo en la ejecución del código, afectando el tiempo global de ejecución de la aplicación (Rivero y Lopez, 2009).

Licencia.

Las implementaciones de Java pueden adquirirse de varias compañías (en lugar de una única como es .NET). Java tiene un largo camino andado en relación al desarrollo de su

arquitectura sobre diferentes plataformas. La tecnología Java es abierta y se basa en gran medida en estándares de organizaciones de normalización.

En el caso del C++ existen variantes tanto propietarias como abiertas (Rivero y Lopez, 2009).

Eficiencia.

Prácticamente todos los computadores ejecutan los programas mediante una o más unidades centrales de procesamiento (CPU) las cuales (dependiendo de la marca y el modelo) sólo comprenden el llamado lenguaje máquina o código máquina, el cual consiste de una serie de operaciones relativamente elementales o de muy bajo nivel tales como escribir “bytes” en memoria, sumar un par de números o leer “bytes” de un dispositivo externo.

Por lo tanto, todos los lenguajes de programación deben ser traducidos en algún momento a lenguaje máquina para que los programas sean ejecutados; simplificando, a este proceso se le suele denominar compilación y tanto el lenguaje C como el lenguaje C++ siguen este esquema de ser compilados al lenguaje máquina del procesador en el que se van a utilizar.

Java fue creado desde el inicio para ser ejecutado en cualquier clase de dispositivo o CPU, y uno de sus aspectos más interesantes es que no se compila directamente en el lenguaje máquina del CPU en uso, sino en un pseudo lenguaje máquina denominado “bytecode”. Este Java compilado en bytecode puede ser transportado a cualquier computador en el cual se dispone de un programa especial encargado de la traducción del bytecode al verdadero lenguaje máquina del CPU en uso.

En otras palabras, este programa especial “interpreta” el “bytecode”, efectivamente ejecutando la aplicación Java original. Este programa intérprete se conoce (simplificando un poco) como Java Virtual Machine (JVM) (Máquina Virtual de Java) o Java RuntimeEnviroment (Ambiente de Ejecución de Java).

Un programa compilado en *bytecode* en tanto debe ser además traducido (interpretado) en lenguaje máquina, en general resulta algo más lento que un programa ya traducido al lenguaje máquina del CPU donde este paso adicional ya no se requiere.

Un segundo inconveniente, particularmente en aplicaciones relativamente pequeñas, radica en los recursos de memoria que típicamente utiliza el Java Virtual Machine; si bien esto suele ser configurable, dichos ajustes no suelen ser sencillos ni bien documentados (Rivero y Lopez, 2009).

Un tercer inconveniente para ciertas clases de aplicaciones se encuentra en lo impredecible que es el recolector de basura, el cual en muchas ocasiones no realiza su trabajo en el momento más apropiado y suele consumir mucho tiempo de CPU en su análisis, contribuyendo a la lentitud.

Afortunadamente los implementadores de las JVM han optimizado mucho la inteligencia del recolector de basura al punto que en la actualidad esto sólo es un problema en casos excepcionales. Un programa en Java suele ser notoriamente más lento si la tarea principal consiste en operaciones lógico-matemáticas, la eficiencia suele ser ligeramente inferior a la correspondiente a C / C++ para aplicaciones que hacen uso de muchos otros componentes y librerías auxiliares. C# en este aspecto es muy similar a Java (Rivero y Lopez, 2009).

1.3.3 Elección del lenguaje a utilizar

Para tomar una decisión en la elección del lenguaje, se realiza una tabla resumen (ver Tabla 1.3) donde se les asignan valores a los diferentes criterios de selección en función de su importancia para el desarrollo de la aplicación (Gámez, *et al.*, 2009).

Tabla.1.3 Comparación entre los lenguajes de programación.

Criterio de Selección	*Peso*	*C++*	*C#*	*Java*
Portabilidad	**5**	**3**	**4**	**5**
Capacidad 2D/3D	**5**	**3**	**2**	**3**
Matemática de precisión	**5**	**3**	**2**	**2**
Gestión memoria	**5**	**5**	**3**	**3**
Velocidad	**10**	**10**	**8**	**6**
Licencia	**10**	**10**	**0**	**10**
Eficiencia	**10**	**10**	**6**	**6**
Total	**50**	**44**	**25**	**35**

Teniendo en cuenta estos resultados se opta por escoger como lenguaje de programación el C++ para aprovechar su velocidad de ejecución, eficiencia y todas las potencialidades que ofrece de manera general.

1.3.4 Elección del entorno de programación.

Para la elección del IDE se tuvo en cuenta principalmente que fuera una aplicación de software libre, por las ventajas que estas brindan entre las que se destacan: evitar la dependencia tecnológica de empresas foráneas, el ahorro por concepto de pagos de licencias, además de su compatibilidad con el lenguaje seleccionado y posibilidad de trabajo multiplataforma.

Entre los IDE software libre más populares del mercado se encuentran:

NetBeans: Es una plataforma para el desarrollo de aplicaciones de escritorio principalmente para Java, aunque soporta otros lenguajes como C/C++, php entre otros. SunMicroSystems funda el proyecto de código abierto en junio 2000. Sus limitantes fundamentales se encuentran en su entorno gráfico que no es muy amigable. Cabe destacar su soporte multiplataforma.

Eclipse: Es un IDE multiplataforma compuesto por un conjunto de herramientas de programación de código abierto. Pese a que está escrito en su mayor parte en Java (salvo el núcleo), se ejecuta sobre máquina virtual de esta y su uso más popular es como un IDE para Java, aunque es neutral y adaptable a cualquier tipo de lenguaje, como C/C++, C# o XML. Sus mayores ventajas radican en su gran comunidad de desarrollo.

Qt Creator: Es un IDE completamente integrado para el desarrollo de proyectos basados en las librerías Qt mediante el lenguaje C++. Está disponible para plataformas Windows, Mac OS X y Linux. Provee mediante el uso de dichas librerías portabilidad para el código en las diferentes plataformas, es decir, el mismo código puede ser compilado sin realizar

cambios para los diferentes sistemas operativos, siempre y cuando no se utilicen librerías o características propias de alguno de estos sistemas operativos. Brinda gran cantidad de librerías además de los estándares de C++ y una novedosa concepción en la programación guiada por eventos en ambientes visuales.

Finalmente se seleccionó Qt Creator por sus novedosas concepciones como es el caso de la comunicación entre módulos para la programación en ambientes gráficos, mediante el mecanismo que lo hace diferir de la mayoría de las plataformas los “signals” y los “slots”. Otro de las características particulares fundamentales de este IDE es la portabilidad del código escrito a cualquier compilador Qt de cualquier plataforma (Thelin, 2007).

1.4 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido concluir:

1. Los sistemas embebidos poseen gran aplicación en el diseño de sistemas de monitoreo, pues se presentan como alternativas a los elevados precios en el mercado de los sistemas convencionales.

2. Los ordenadores de placa reducida una de los principales sistemas embebidos utilizados en los sistemas de monitoreo entre los que se destaca la Raspberry Pi 2 model B, el cual es el SCB base utilizado en el sistema propuesto en este trabajo.

3. De los lenguajes de programación más utilizados en la actualidad el que permite el desarrollo de aplicaciones que cumplan con los requisitos del sistema propuesto es C++.

CAPÍTULO 2 ARQUITECTURA DEL SISTEMA

En el presente capítulo se expone el proceso de desarrollo del hardware y software para el sistema de monitoreo. Se analiza la arquitectura básica del mismo y se muestran los elementos que conforman su análisis y diseño.

2.1 Arquitectura de *hardware*

Para el diseño del sistema se hace necesario una arquitectura sencilla y a la vez eficiente que permita explotar todas las bondades que brindan los elementos que la componen además de ser capaz de sobreponerse en corto tiempo a las posibles fallas que puedan ocurrir.

Se propone para interconexión de los elementos del sistema una arquitectura flexible, la cual es capaz de adaptarse a cambios en cualquiera de los elementos que la componen sin necesidad de realizar grandes cambios en la estructura o la parametrización de los restantes componentes.

En la figura 2.1 se presenta la arquitectura la cual está compuesta por un sistema de acondicionamiento de señales, el ordenador de placas reducidas (Raspberry-Pi 2 *model* B) y el servidor. Los usuarios que pueden interactuar con el sistema a través de las respectivas interfaces tanto con el SBC como con el servidor.

2.1.1 Sistema de acondicionamiento de señales

El sistema de adquisición de datos está compuesto por dos etapas fundamentales, la primera etapa se encarga del acondicionamiento de señal y la segunda la conversión de la señal

analógica de entrada a digital. La primera etapa como indica su nombre es la encargada de modificar la señal de entrada hasta llevarla al rango de valores que acepta el conversor analógico-digital (0…5) V, esta consta tres de versiones, una versión para la conexión de sensores con salidas digitales (sensores que conmutan su salida entre uno y cero donde los voltajes pueden variar hasta 24V máximo), otra para sensores con salidas analógicas de corriente 4…20mA y una última para sensores con salidas de voltaje de 0…10V(Villalonga, *et al.*, 2016).

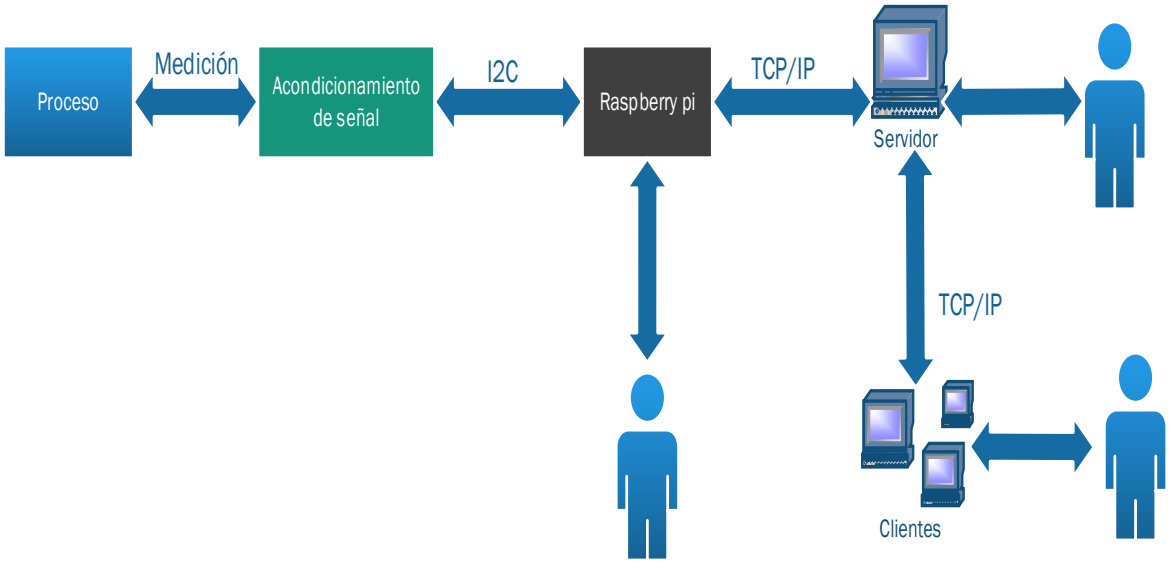

Figura 2.1 Arquitectura hardware del sistema

La segunda etapa del sistema de adquisición de datos se encarga de la conversión de las señales analógicas a digitales, para lo que se utiliza un ADC-Pi Plus. El ADC-Pi Plus (mostrado en la figura 2) es un circuito integrado diseñado para trabajar en el rango de voltajes de 0 … 5.06V y está compuesto por dos adc MCP3424 de Microchip configurados para trabajar de manera diferencial lo que brinda un total de ocho canales (cuatro canales diferenciales por cada uno), que se comunican a través del bus i2c, y un convertidor de

nivel que permite la comunicación por i2c de componentes que funcionan a 5V sin dañar el bus i2c de la Raspberry Pi que trabaja a 3.3V. Además, posee la opción de programar la ganancia de la salida. Este se conecta a la Raspberry Pi a través del puerto GPIO (entradas salidas de propósito general, por sus siglas en ingles). Cada MCP3424 soporta ocho direcciones i2c y cada placa consta de dos por lo que permite como máximo la conexión de cuatro ADC-Pi Plus brindándole un número máximo de 32 entradas al sistema.

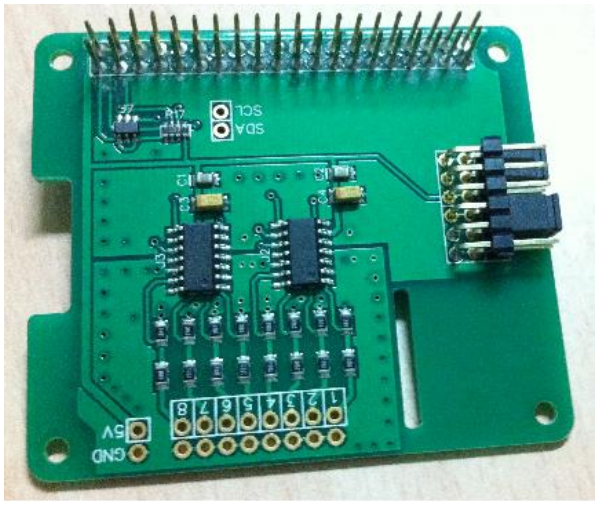

Figura 2.2 ADC-Pi Plus

2.1.2 Raspberry-Pi

Raspberry-Pi es un ordenador de placa reducida de bajo costo, del tamaño de una tarjeta de crédito, diseñado con fines educacionales que permite de una manera sencilla explorar el mundo de la computación y aprender a programar en lenguajes como Scratch y Python con solo conectarla a un monitor o televisor y además conectarle teclado y ratón.

La Raspberry Pi 2 modelo B (mostrada en la figura), está basada en el chip Broadcom BCM2836 que incluye un procesador central quadcore a 900MHz de 32-bit, arquitectura

ARM Cortex-A7 y 1GB de RAM DDR2 por lo que es de 4 a 6 veces más potente que sus predecesores.

Figura 2.3 Raspberry-Pi B model 2

2.1.3 Servidor

El servidor se encarga de la recepción de los datos y su almacenamiento para su posterior visualización y análisis. El objetivo de esta arquitectura es monitorear el proceso a distancia, lo cual se puede lograr a través de la conexión en red de todos los elementos, (Raspbery Pi, servidor y clientes) la cual se realiza a través del estándar Ethernet.

Ethernet es un estándar de redes de área local para computadores, con acceso al medio por detección de la onda portadora y con detección de colisiones (CSMA/CD) (Peterson y Davie, 2012). Se puede clasificar en Ethernet clásico o conmutado. La diferencia fundamental entre ambos es la velocidad de acceso al medio, el ethernet clásico permite

velocidades hasta 10 Mbps mientras el conmutado hasta 10000 Mbps (Tanenbaum y Wetherall, 2011), la arquitectura propuesta trabaja a 100 Mbps por lo que se utiliza Ethernet conmutado.

2.2 Arquitectura de software

2.2.1 Requerimientos del software

El proceso de análisis y diseño de software según los patrones planteados por el RUP, debe ser guiado por casos de uso, centrado en la arquitectura, iterativo e incremental (Jacobson, *et al.*, 2000). Para ello se deben obtener los requisitos del software, los cuales permiten determinar un modelo inicial de casos de uso, aunque no sea definitivo, brinda la posibilidad de incluir futuros requerimientos y además sirve de punto de partida en el análisis y diseño de la arquitectura del sistema. Este proceso se complementa con la confección de un plan de iteraciones a partir del modelo de casos de uso que permita llevar a cabo el proyecto de una manera eficiente.

2.2.2 Características del sistema

En las tablas 2.1-2.5 se exponen las principales funciones que debe cumplir el sistema, en donde se incluyen además los sus valores, en cuanto a riesgo, prioridad y categoría. Estas tablas permiten poder decidir en qué iteración del proceso de desarrollo ubicar cada función en virtud de la característica propia de la metodología RUP.

Tabla2.1 Funciones relacionadas con la comunicación

Ref.	*Característica*	*Riesgo*	*Prioridad*	*Categoría*
R1.1	Establecer conexión entre el servidor y la Paspberry-Pi.	Crítico	Crítico	Evidente
R1.2	Garantizar una comunicación estable.	Crítico	Crítico	Evidente

Tabla 2.2 Funciones relacionadas con la supervisión.

Ref.	*Característica*	*Riesgo*	*Prioridad*	*Categoría*
R2.1	Visualizar las variables del proceso.	Ordinario	Importante	Evidente
R2.2	Adquirir variables las variables del proceso	Ordinario	Importante	Evidente
R2.3	Obtener reportes de históricos almacenados.	Ordinario	Importante	Evidente

Tabla 2.3 Funciones asociadas con la actualización de las bases de datos.

Ref.	*Característica*	*Riesgo*	*Prioridad*	*Categoría*
R4.1	Mantener actualizadas y registrar las variables del proceso	Crítico	Importante	Evidente
R4.2	Hacer el registro de las alarmas en bases de datos.	Crítico	Importante	Evidente

Tabla 2.4 Funciones asociadas a la configuración.

Ref.	*Característica*	*Riesgo*	*Prioridad*	*Categoría*
R5.1	Configurar los usuarios del sistema.	Crítico	Importante	Evidente
R5.2	Configurar los privilegios por tipos de usuarios.	Crítico	Importante	Evidente
R5.3	Configurar alarmas.	Crítico	Importante	Evidente
R5.4	Configurar variables.	Crítico	Importante	Evidente

Tabla 2.5 Otras funciones.

Ref.	*Característica*	*Riesgo*	*Prioridad*	*Categoría*
R7.1	Obtener ayuda del sistema.	Ordinario	Secundario	Evidente

2.2.3 Contexto del sistema

Para el correcto funcionamiento del sistema los softwares que se diseñaron, sobre todo los que trabajan sobre la Raspberry-Pi, se deben cumplir los requerimientos de tiempo necesarios para que no ocurran perdida de datos , además de garantizar también una correcta configuración de la comunicación de red para evitar la pérdida de paquetes que se traducen en demoras de tiempo ya que estos paquetes, como parte de la política de comprobación de errores de la red, deben ser reenviados cuando ocurre la perdida de los mismos. Para poder dar cumplimiento a estos requisitos se diseñó el software tomando como base la arquitectura que se presenta en la figura 2.4. Esta se basa en una arquitectura cliente-servidor en la cual las Raspberry-Pi actúa como cliente y la aplicación que se ejecuta en la PC como servidor.

2.2.4 Cliente

La aplicación cliente se ejecuta sobre la Raspberry-Pi y como se puede apreciar en la figura consta de dos módulos fundamentales adquisición y trasmisión de datos y visualización y configuración. En orden de garantizar que la aplicación posea un procesamiento eficiente de la información y nos incurra en la perdida de datos se utiliza programación por hilos para lograr un paralelismo entre ambos módulos lo que nos permite cumplir con las líneas de tiempo del proceso que se esté monitoreando.

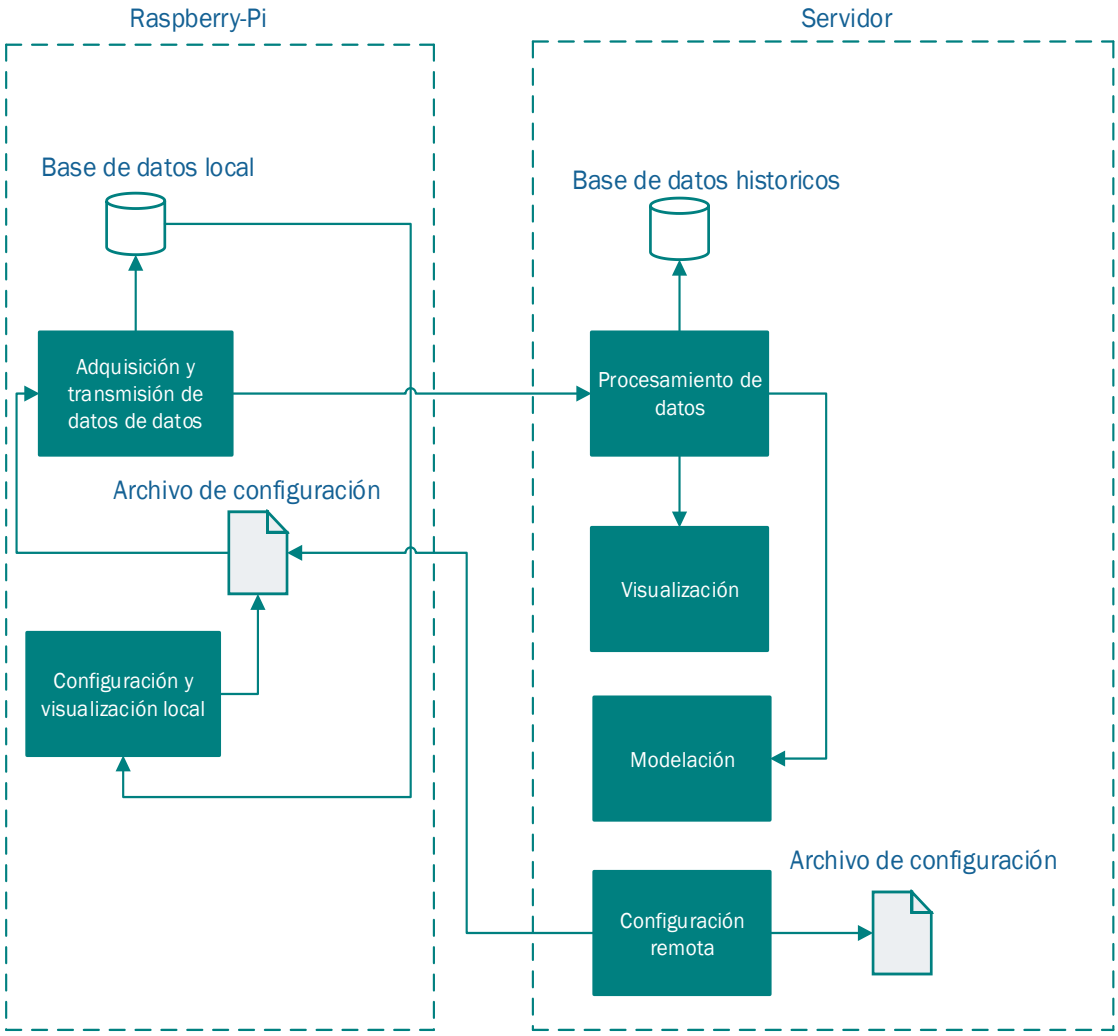

Figura 2.4 Arquitectura de software.

El módulo de adquisición y trasmisión de datos a su vez consta de dos *interfaces* una encargada de gestionar la comunicación con la tarjeta de adquisición de datos y la segunda se ocupa del establecimiento de la comunicación y el intercambio de datos con el servidor a través del protocolo TCP-IP. Cada interface se encuentra implementada en hilos separados por lo que ambas funcionan en paralelo.

La *interface* de adquisición de datos se comunica con el ADC-Pi Plus a través los pines del GPIO destinados a la comunicación I2C, para ello a uso de una clase creada basándose en

las librerías en C que proporciona el fabricante del ADC-Pi Plus. Además de la adquisición de los datos este módulo se encarga de la configuración de los parámetros del *hardware* como son la ganancia por canal y las direcciones I2C de los mismos.

La interface de red es la encargada de gestionar todo lo referente al protocolo de red implementado, o sea la recepción de comandos enviados por el servidor y su procesamiento, así como el envío hacia el mismo de los datos recolectados del campo.

2.2.5 Servidor

La aplicación servidor que se ejecuta sobre el ordenador que actúa como servidor en el sistema y es la encargada esencialmente de recibir la información recolectada del campo por el ordenador de placa reducida, almacenarla, y presentársela al usuario a través de gráficos y tablas. Además, brinda un paquete de herramientas matemáticas, el cual es configurado por los usuarios del sistema, que permita realizar el procesamiento de los datos y la obtención de modelos matemáticos que describan la dinámica del proceso que se está monitoreando.

La aplicación servidor está compuesta por cuatro módulos fundamentales: procesamiento de datos, visualización, modelación y configuración remota. Cada módulo posee una función específica y se ejecutan en hilos de procesos diferentes, aunque entre ellos pueden intercambiar información.

El módulo de visualización es el encargado de presentar la información al usuario, para ello hace uso de una interface gráfica que permite representar la información a través de gráficos de históricos y tendencias o mediante tablas de variables. Para la creación de los

gráficos se utilizó la biblioteca *qwt (Qt Widgets for Technical Applications)* versión 5.2.1, biblioteca de código abierto distribuida bajo la Licencia Pública General de *GNU.*

El módulo de procesamiento de datos se ocupa de la recepción de los datos enviados por la Raspberry-Pi, su procesamiento y posterior almacenamiento en la base de datos. La recepción de los datos se realiza a través de la *interface* de red. Para el procesamiento de datos ofrecen una serie de filtros digitales, posibilita también la aplicación de transformadas como Fourier y *wavelet* y de una serie de estadígrafos como kourtosis, rms (raíz cuadrática media por sus siglas en ingles), desviación estándar, pico máximo, entre otros. Para la implementación de los estadígrafos y las transformadas se utilizó la biblioteca de código abierto *gsl* (*GNU Scientific Library*) 2.1 la cual es distribuida bajo la Licencia Pública General de *GNU.*

El módulo de modelación permite la obtención de modelos matemáticos que describan la dinámica del proceso monitoreado. La modelación se realiza fundamentalmente a través del método de los mínimos cuadrados ordinarios, para la implementación de este método se hizo uso de la biblioteca *gsl*. Este módulo brinda también la opción al usuario de seleccionar de las variables que están siendo monitoreadas cuales deseada usar como entradas y cuales como salidas para la obtención de los modelos.

El módulo de configuración consta de una interface gráfica que permite parametrizar todos los elementos que componen el sistema. A través de este se define los usuarios del sistema con sus respectivas jerarquías, todas las características de las variables del proceso que se van a monitorear (rango de medición, señal de salida del sensor que las mide, alarmas y sus respectivos valores), la configuración de los distintos canales del sistema de adquisición

de datos y de la red. Esta configuración es almacenada en un archivo con formato *XML* y trasmitida a la *Raspberry-Pi* a través de la *interface* de red. La información de configuración trasmitida es almacenada en la Raspberry-Pi en un archivo de configuración en formato *XML*, como información redúndate que permite al sistema recuperarse de un fallo de comunicación sin perder su parametrización.

2.3 Protocolo de comunicación

La comunicación entre la Raspberry Pi, el servidor y los clientes se efectúa a través de protocolo TCP/IP, el cual permite que se puedan comunicar diferentes equipos mediante una interfaz de red sin importar las características de los mismos o el sistema operativo sobre el cual se trabajan.

El uso del protocolo TCP/IP garantiza el establecimiento de conexiones y el intercambio confiable de datos. Sin embargo, TCP/IP pertenece a las capas de transporte (TCP) y red (IP) del modelo OSI de la ISO (Fall y Stevens, 2012) y no es responsable de darle el formato deseado por una determinada aplicación a su campo de datos ni de la lógica en el intercambio de los mismos, para ello se necesita la creación de un protocolo sobre la capa de aplicación de dicho modelo que se adapte a las necesidades del sistema. Por lo que se decidió crear un protocolo basado en las necesidades del sistema, mostrado en la figura 2.5.

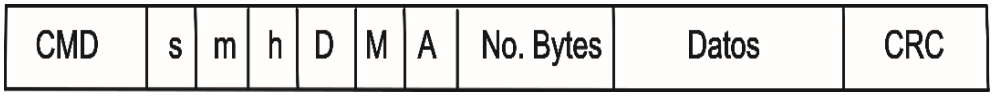

Figura 2.5 Trama de datos.

Seguidamente se explican los campos de la trama de datos:

CMD: Indica la acción a realizar con los datos recibidos. Este campo puede ser un comando o una respuesta a un comando enviado. Por convenio se decidió otorgarles a los comandos valores pares y a las respuestas valores impares. (1 byte).

Los campos s, m, h, D, M, A indican: segundo, minuto, hora, día, mes y año respectivamente en que se envió el dato. (1 byte cada uno).

NO. BYTES: Número de bytes que tiene el campo DATOS. (1 byte).

DATOS: Este campo contiene la información específica para cada comando o respuesta. Este campo es opcional ya que hay comandos que no llevan datos. (longitud variable).

CRC: Código de Redundancia Cíclica (1 byte).

En este protocolo se incluyen elementos para la detección de errores. Se inserta en la trama el campo CRC con el cual se chequea si los datos restantes están arribaron correctamente o han sido corrompidos. La comprobación se realiza a través de la comparación del resultado de la función lógica OR exclusivo de todos los bytes de la trama con el CRC.

2.4 Bases de datos

Se determinó que era necesario la utilización de bases de datos para el almacenamiento de información debido a la persistencia que debía presentar cierta información, como la configuración de las variables del proceso, además del gran volumen de información que se adquiere en el procesamiento en tiempo real y la necesidad de disponer de reportes históricos de las incidencias del sistema.

Se decidió la utilización de bases de datos relacionales para cumplir con estos requerimientos por las ventajas que estas ofrecen (Churcher, 2007):

- Integridad de datos a múltiples niveles: A nivel de campo para asegurar que los datos son correctos (tipo, valor), a nivel de tabla para asegurarse de que no hay registros repetidos y a nivel de relaciones para asegurarse que la relación entre dos tablas es válida.

- Fácil obtención de datos: Los datos pueden ser obtenidos de una tabla en particular o de un grupo determinado de tablas relacionadas dentro de la base de datos.

En la figura 2.6 se muestra el diseño de las tablas que participan en el almacenamiento de información. La tabla *Usuarios* contiene toda la información referente a los usuarios del sistema (nombre de usuario, contraseña y privilegios). La tabla *Variables* almacena la información de la configuración de todas las variables del sistema. Las tablas *Variables_log* y *Alarmas* contienen un registro del valor de todas las variables medidas y las alarmas ocurridas.

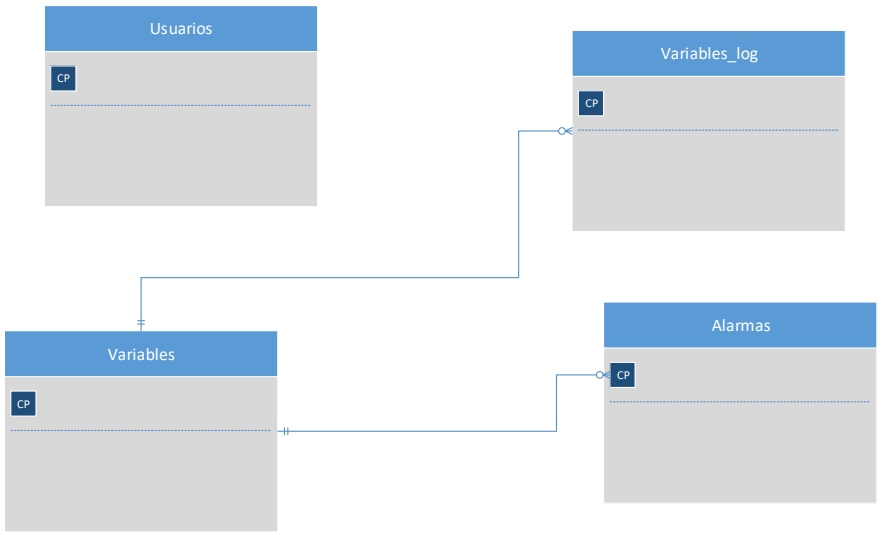

Figura 2.6 Diagrama de entidad-relación de la base de datos.

2.5 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. Se realizó el diseño de la arquitectura de hardware del sistema, la cual está basada en una arquitectura flexible de tipo cliente servidor.

2. Se realizó el análisis, diseño e implementación de las funcionalidades del sistema necesarias para un correcto funcionamiento de los softwares que lo componen.

CAPÍTULO 3 CONFIGURACIÓN, MONITOREO Y PRUEBAS DEL SISTEMA

En este capítulo se muestra como realizar algunas de las acciones de configuración y monitoreo sobre el sistema, mediante el uso de la Aplicación de Servidor y la Aplicación Cliente. Se exponen las interfaces gráficas de ambas mientras son usadas en la realización de las acciones, para una mejor comprensión de su utilización. Se exponen las pruebas realizadas al software. Entre las diferentes variantes disponibles se aplican las que se consideran ideales para la aplicación y que al mismo tiempo concuerdan con el sistema de programación que se utiliza en el desarrollo.

3.1 Aplicación Servidor

La Aplicación Servidor permite la supervisión del proceso, así como la configuración y administración del sistema. Mediante esta aplicación podemos visualizar, tendencia, históricos y los valores de las variables, así como configurar los parámetros del sistema y los usuarios y sus privilegios. En la figura 3.1 se muestra la ventana inicial de la aplicación.

La aplicación no puede ser utilizada apenas se ejecuta, sino que es necesario la autenticación en la misma. Al accionar la opción de autenticación en la barra de herramientas, se despliega un cuadro de diálogo (figura 3.2) con las opciones de introducir el nombre de usuario y la contraseña. Una vez hecha la autenticación, el usuario tiene acceso a las funcionalidades del sistema que corresponden con su nivel de jerarquía.

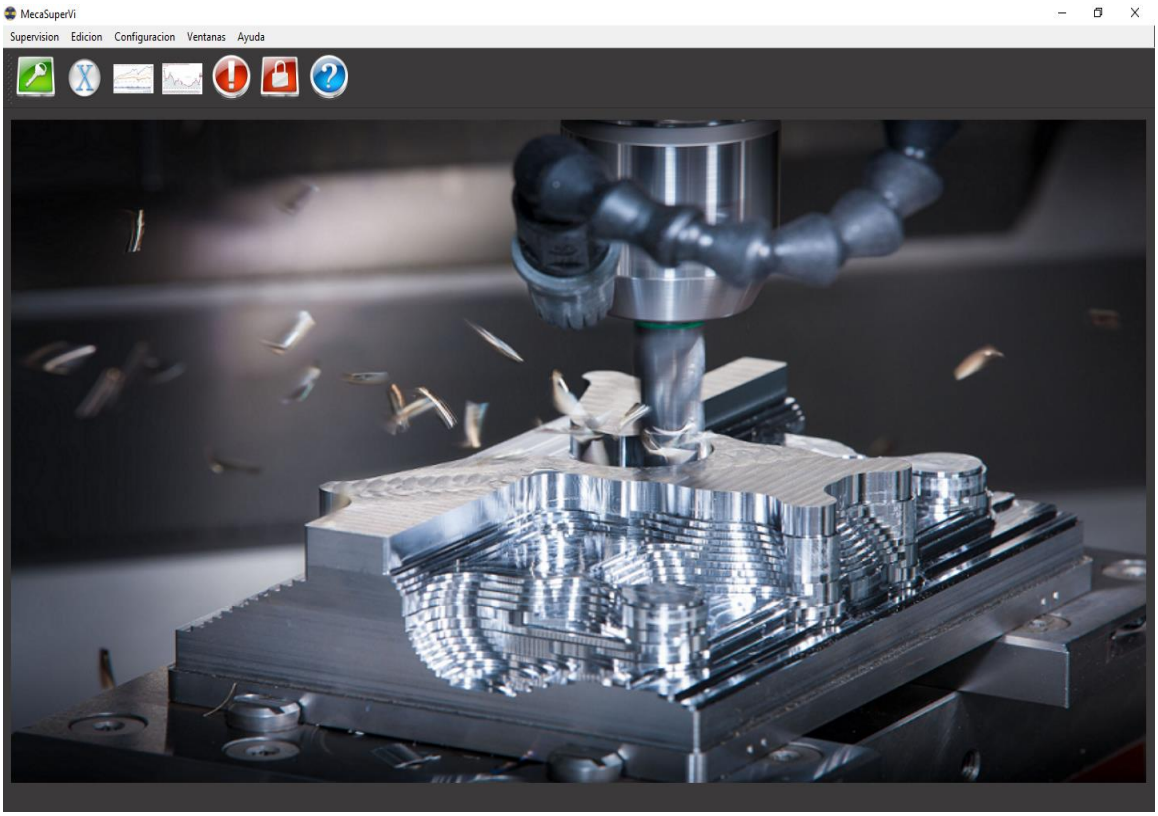

Figura 3.1. Ventana inicial de la aplicación Server.

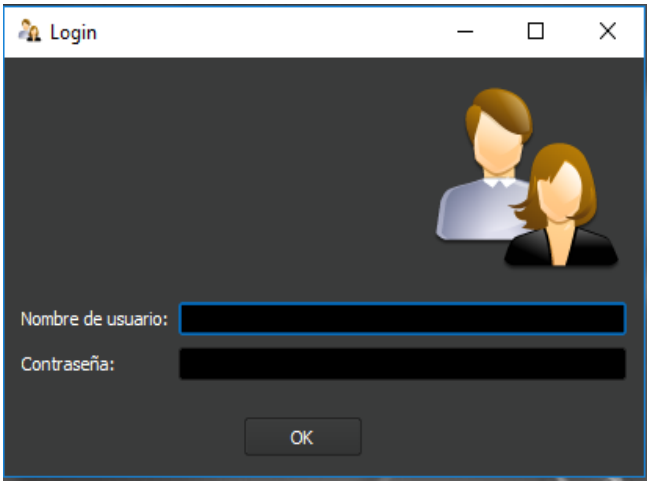

Figura 3.2. Cuadro de dialogo para la autenticación, aplicación Server.

3.1.1 Configuración del sistema.

Esta sección permite administrar los parámetros relacionados con las variables y los usuarios del sistema. La Ventana Editor de Variables permite insertar una nueva variable, modificar parámetros de las existentes y eliminar las que se deseen. Para crear una nueva variable se debe rellenar los campos que se encuentran en la parte inferior de la ventana, estos son: nombre, señal de salida, rango de medición y alarmas de proceso. En cuanto los dos ultimo campos se debe aclarar que el primero se inhabilita cuando se selecciona como señal de salida la opción digital, y que el campo alarma de proceso se activa solamente en las variables que lo requieran definiéndose si será por valor mínimo, máximo o ambos. A esta se puede acceder a través del menú configuración.

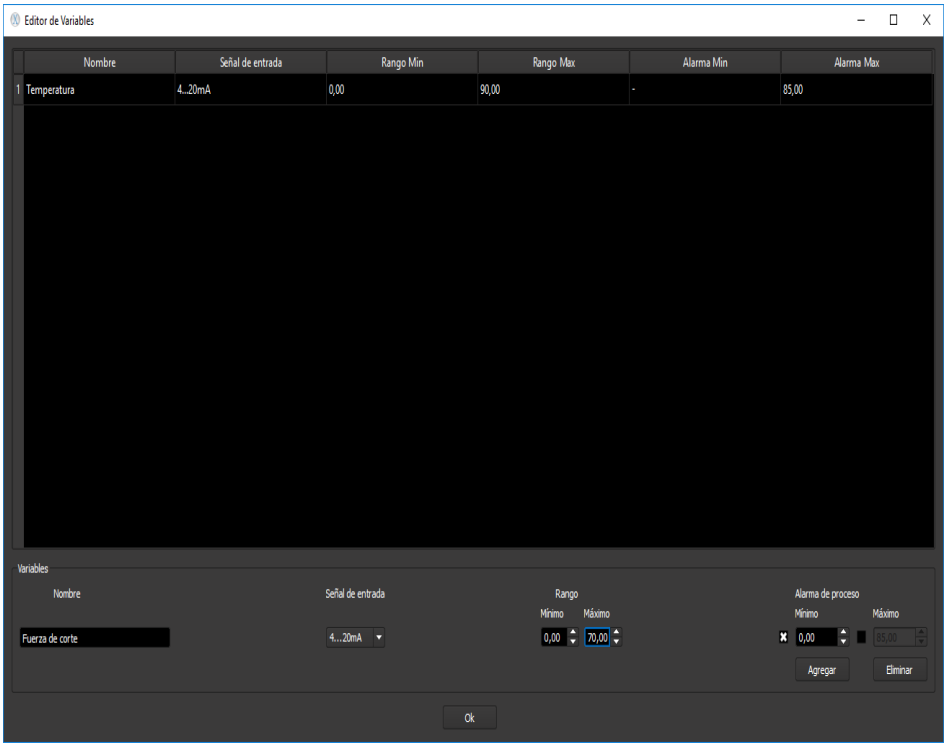

Figura 3.3. Ventana de configuración de variables.

La Ventana de administración de usuarios se puede encontrar en el menú de configuración de la aplicación. En esta se pueden insertar nuevos usuarios al sistema, modificar parámetros de los existentes o eliminar del sistema a algún usuario. En ella se muestra un listado de todos los usuarios en forma de tabla, mediante el cual se pueden observar sus parámetros (excepto la contraseña) y seleccionar los usuarios a modificar o eliminar. En la figura 3.4 se muestra la sección de usuarios de la aplicación. Cuando se desea añadir o modificar un usuario aparece el cuadro de dialogo añadir/modificar usuario en el cual se encuentran los campos nombre de usuario, contraseña y privilegios.

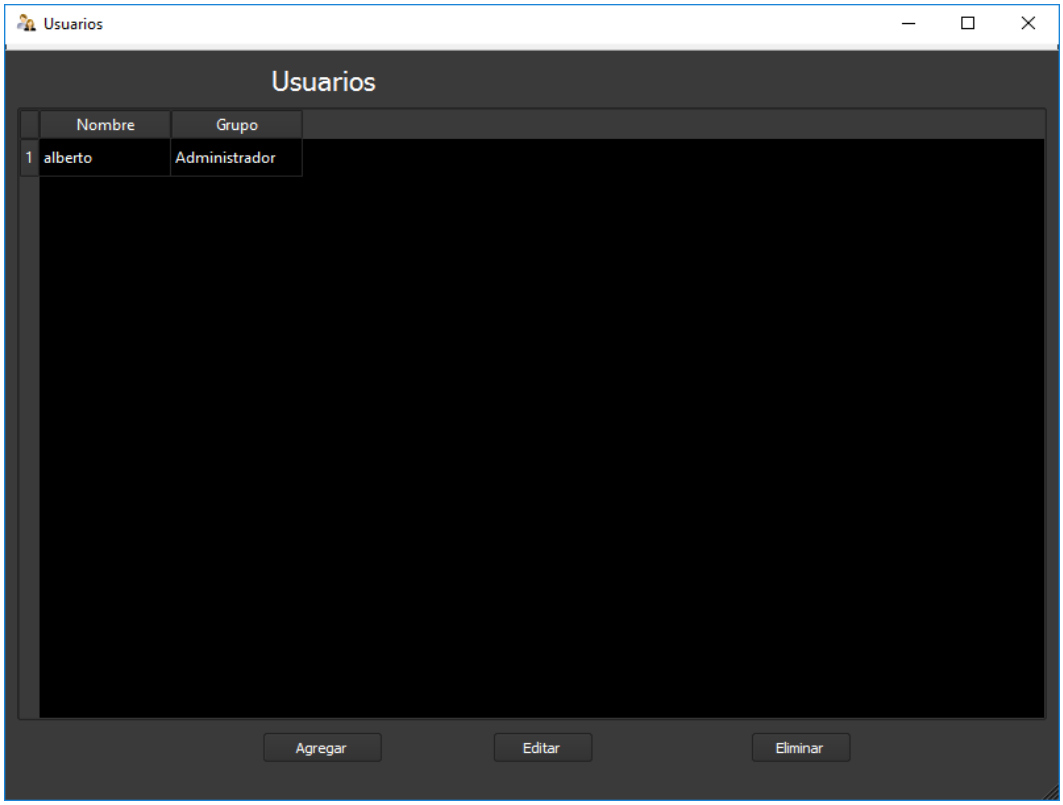

Figura 3.4. Ventana de configuración de usuarios.

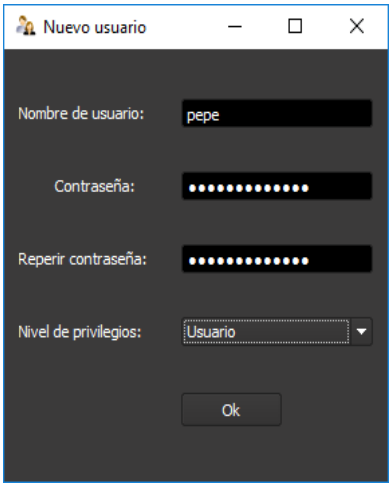

Figura 3.5. Cuadro de dialogo agregar/modificar usuario.

3.1.2 Visualización de las variables de proceso.

La aplicación consta con dos modos fundamentales para la presentación de las variables: la ventana de tendencias y la ventana tabla de variables.

La ventana de tendencias permite a los usuarios visualizar un gráfico en el cual se puede apreciar las variables monitoreadas en el tiempo, cuenta con un menú para realizar operaciones referentes al estado de la ventana y al gráfico que se obtiene. Se accede a ella a través del menú Supervisión o de la barra de herramientas. En la figura 3.6 se muestra una captura de la ventana de tendencias.

La ventana Tabla de Variables (figura 3.7) permite la visualización de los valores actuales de las variables que se están supervisando. La misma está compuesta por una tabla con dos columnas, la primera contiene el nombre de la variable y la segunda su valor actual.

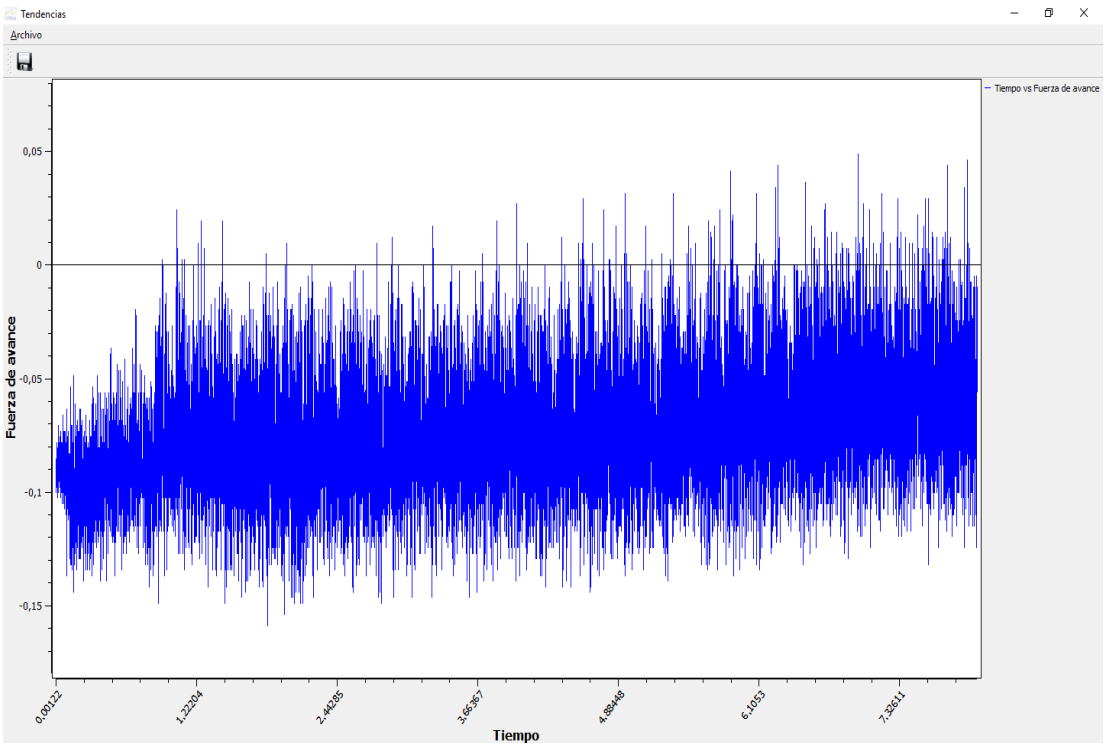

Figura 3.6. Ventana de Tendencias.

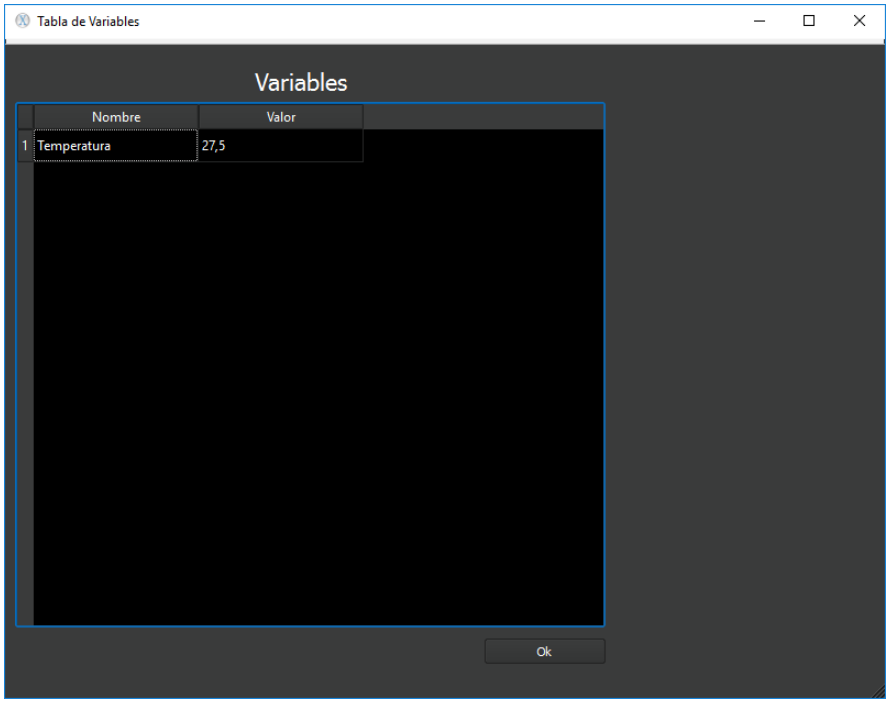

Figura 3.7. Ventana tabla de variables.

3.1.4 Visualización de alarmas.

La ventana de alarmas (Fig 2.3) permite visualizar todas las alarmas ocurridas durante el funcionamiento del sistema que se esté monitoreando. La misma consta de una tabla la cual a su vez está compuesta por tres columnas que indican la hora, la fecha de ocurrencia y el texto informativo de la alarma.

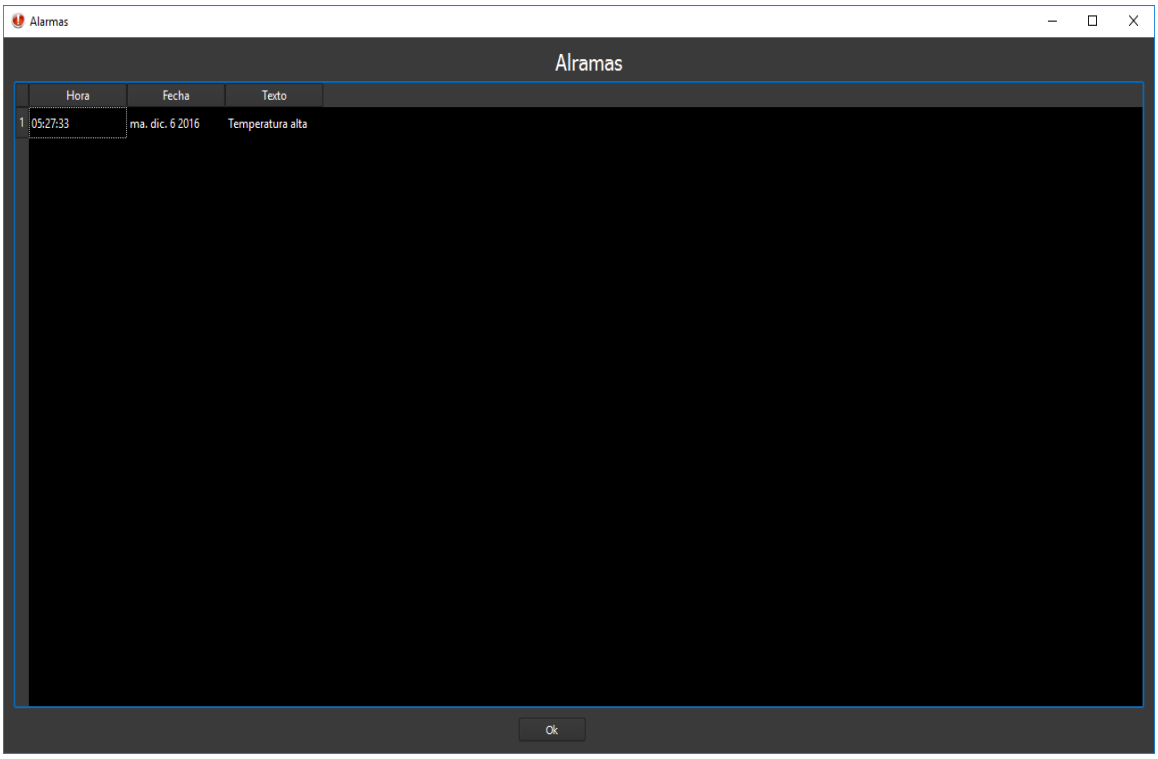

Figura 3.8. Ventana tabla de alarmas.

3.1.4 Visualización de históricos.

La ventana históricos (figura 3.9) permite observar los valores históricos de las variables del sistema que se esté monitoreando. Cuenta con un menú para realizar operaciones referentes al estado de la ventana y al grafico que se obtiene. En su costado derecho posee

dos botones que son los que permiten moverse por el grafico en el tiempo y el cuadro de texto que aparece encima de dichos botones muestra el valor sobre el cual se encuentra el punto de visualización.

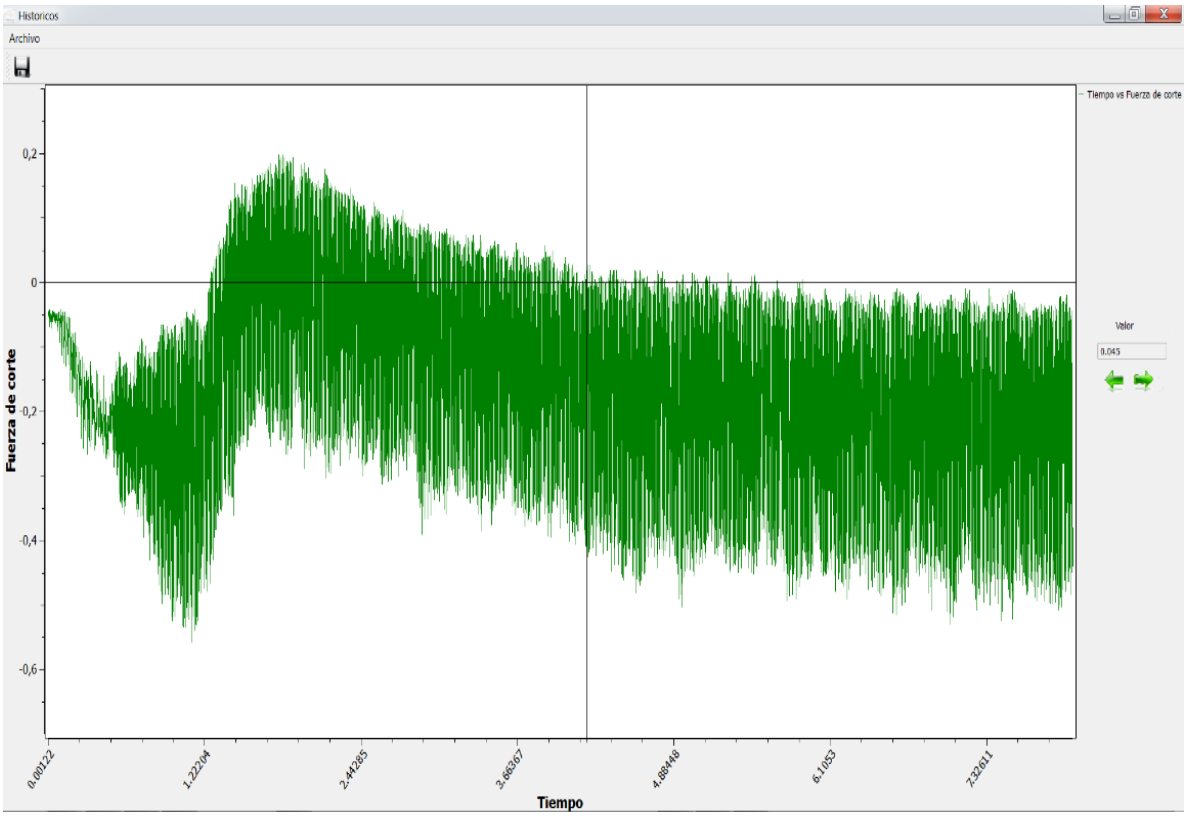

Figura 3.9. Ventana de históricos.

3.2 Aplicación Cliente

Esta aplicación permite realizar la parametrización de los elementos del sistema que se encuentran jerárquicamente por debajo del SBC (ADC-Pi) y del mismo. Ademas permite también poder monitorear el sistema aunque con la utilización de recursos limitados ya que la principal función de esta es la recolección y envió de los datos registrados en el campo.

Inicialmente solo se tiene acceso a las opciones de autenticación y de ayuda. Para acceder al sistema es necesario autenticarse en el mismo mediante la opción de autenticación, la cual despliega un cuadro de dialogo (ver figura) en el cual se introducen: la dirección IP de la aplicación servidor, el nombre de usuario y la contraseña. Si la autenticación es correcta, es posible acceder a los servicios de la aplicación que son permitidos por los privilegios que tenga el tipo de usuario que utiliza la misma.

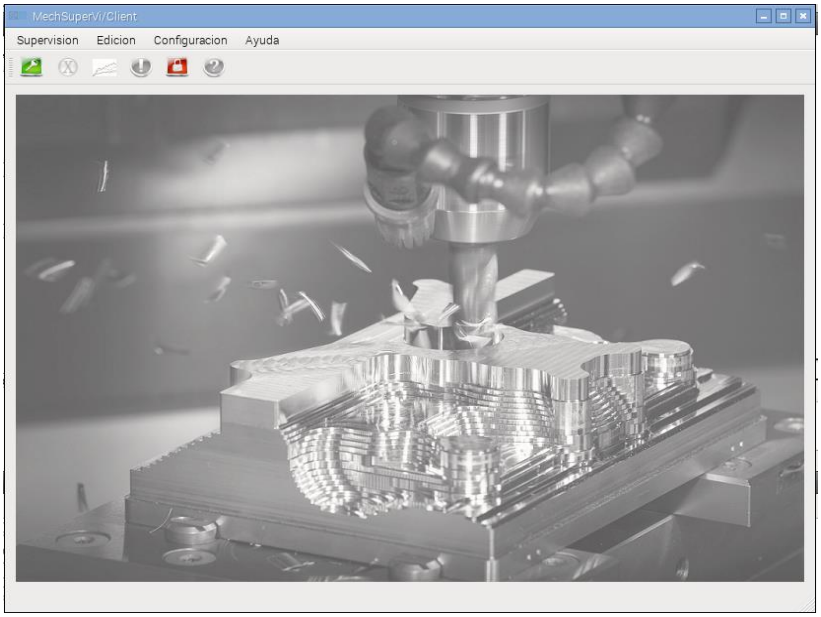

Figura 3.10. Ventana principal aplicación cliente.

Figura 3.11. Cuadro de dialogo para autenticarse aplicación cliente.

3.2.1 Configuración del sistema.

En esta sección podemos configurar los elementos que componen el sistema ya sean las variables, los usuarios u otros parámetros. La ventana Variables (figura 3.12) permite agregar, editar y eliminar las variables del sistema que se desean monitorear. Está formada por una tabla con cinco columnas en donde se muestran las variables ya configuradas. Además, cuenta en su parte inferior con controles de edición para introducir las nuevas variables, editar o eliminar las ya existentes. A esta se puede acceder a través del menú configuración.

La ventana Usuarios (figura 3.13) muestra los usuarios que tienen acceso al sistema y sus privilegios. La misma está compuesta por una tabla con dos columnas, la primera contiene el nombre del usuario y la segunda el grupo al que pertenece con sus respectivos privilegios (usuarios o administradores) y nos permite crear o modificar usuarios a través del cuadro de diálogo agregar/modificar usuarios(figura) así como eliminar los ya existentes

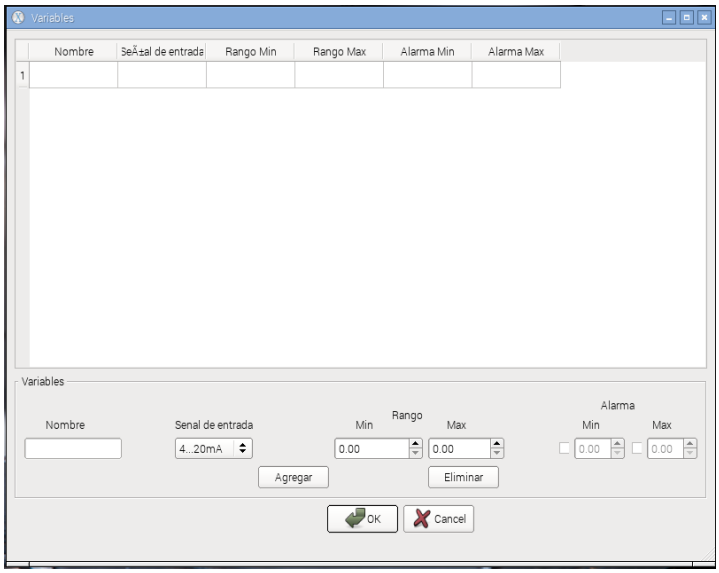

Figura 3.12. Ventana variables aplicación cliente

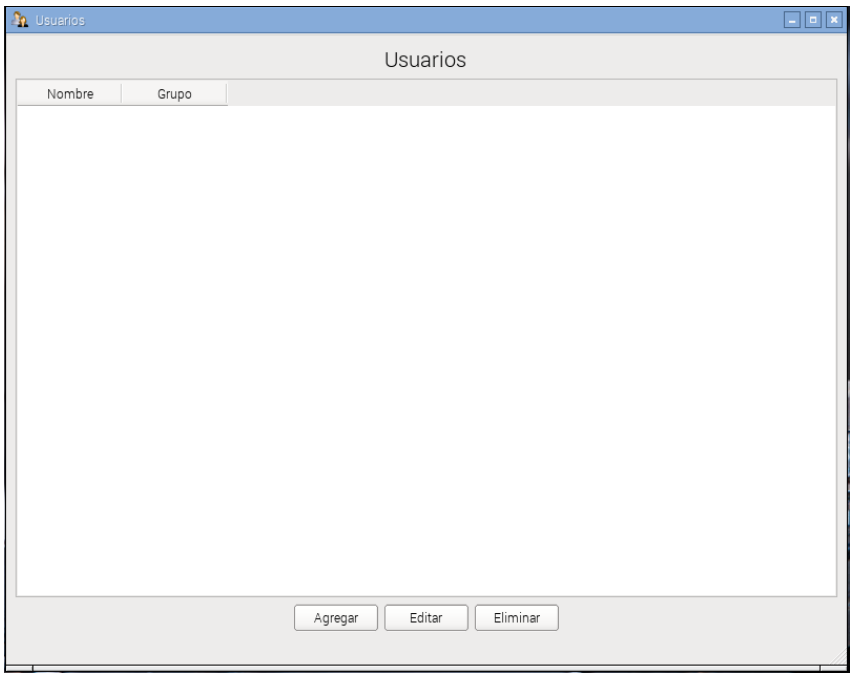

Figura 3.13. Ventana Usuarios aplicación cliente

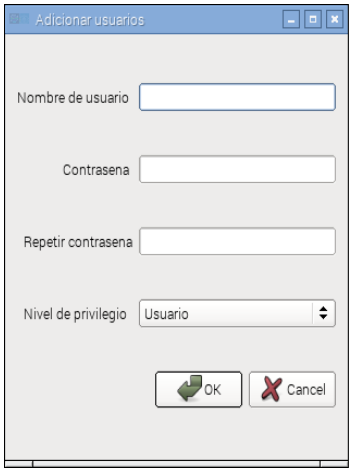

Figura 3.14. Cuadro de dialogo añadir/modificar usuario

3.2.2 Visualización de las variables de proceso.

La aplicación consta con dos modos fundamentales para la visualización de las variables: la ventana de tendencias y la ventana tabla de variables.

La ventana de tendencias permite a los usuarios visualizar un gráfico en el cual se puede apreciar las variables monitoreadas en el tiempo, cuenta con un menú para realizar operaciones referentes al estado de la ventana y al grafico que se obtiene. Se accede a ella a través del menú Supervisión o de la barra de herramientas. En la figura 3.15 se muestra una captura de la ventana de tendencias.

Al igual que en la aplicación servidor la ventana tabla de variables (figura 3.16) permite la visualización de los valores actuales de las variables que se están supervisando. La misma está compuesta por una tabla con dos columnas, la primera contiene el nombre de la variable y la segunda su valor actual.

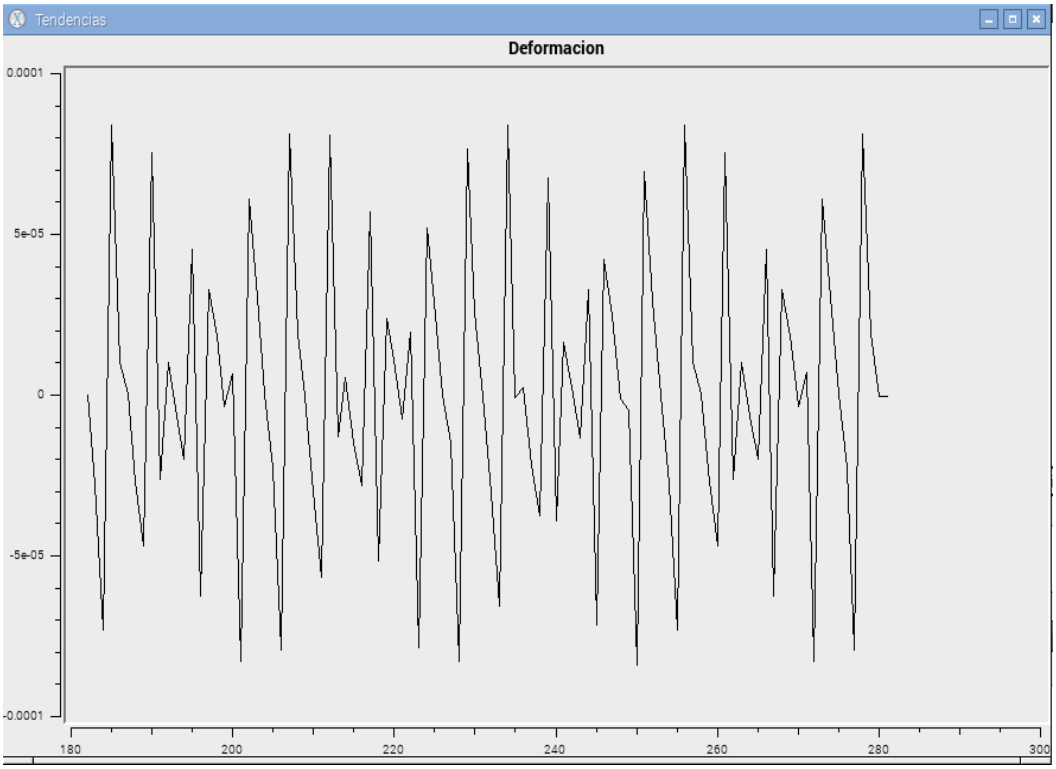

Figura 3.15. Ventana tendencias aplicación cliente.

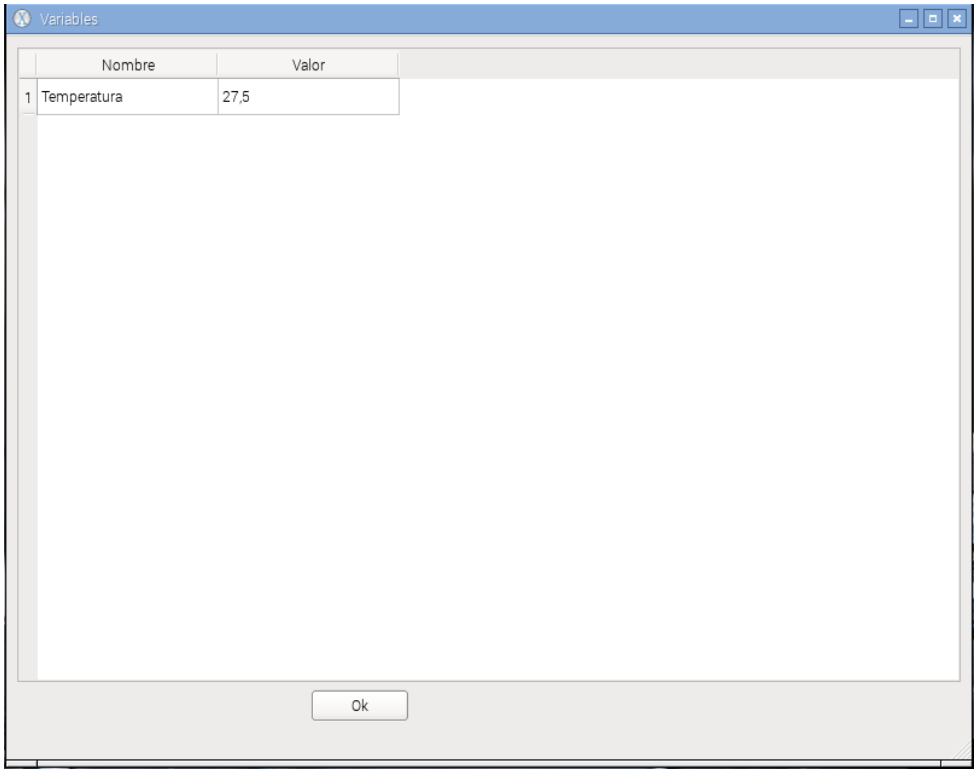

Figura 3.16. Ventana tabla de variables aplicación cliente

3.2.3 Visualización de alarmas.

La ventana de alarmas (figura 3.17) permite visualizar todas las alarmas ocurridas durante el funcionamiento del sistema que se esté monitoreando. La misma consta de una tabla la cual a su vez está compuesta, al igual que la diseñada para la aplicación servidor, por tres columnas que indican la hora, la fecha de ocurrencia y el texto informativo de la alarma.

3.3 Validación del sistema

Las pruebas de unidad forman parte de un proceso para comprobar los módulos, clases o métodos en un programa. Resulta conveniente realizar primero la prueba de los bloques más pequeños del programa, que inicialmente probar el software en su totalidad. Este grupo

de pruebas está íntimamente ligado con el proceso de programación. La prueba de una unidad facilita la tarea de eliminar errores, pues, cuando se encuentra un error, se conoce que existe en un módulo, clase o método particular. Las pruebas de unidad realizadas dieron buenos resultados. En la tabla 3.1 se muestran algunas de las clases probadas.

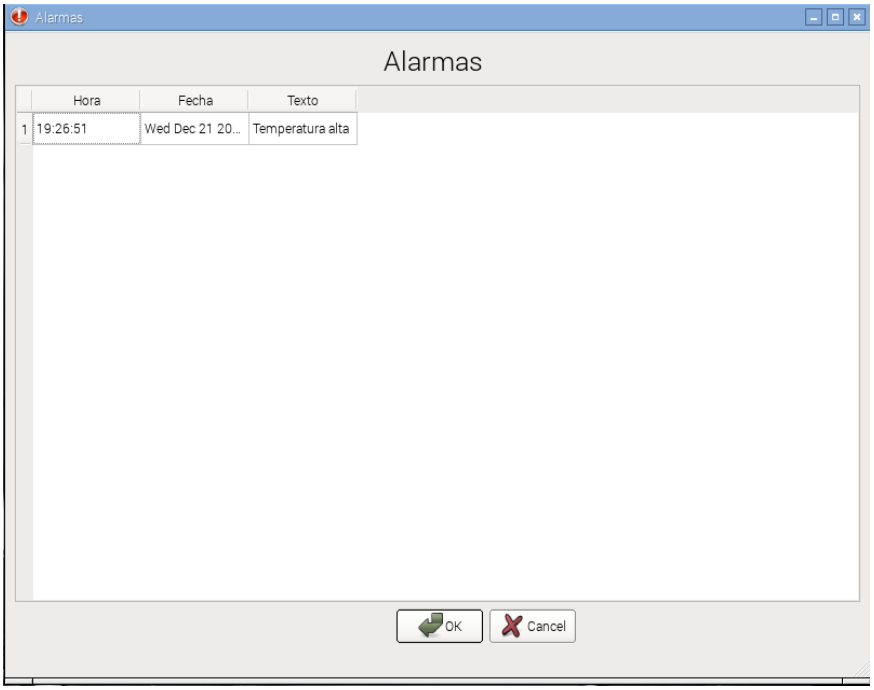

Figura 3.17 Ventana de alarmas aplicación cliente

Tabla 3.1 Algunas clases probadas

Nombre	*Aplicación*	*Breve descripción*	*Error*
Net_thread	Cliente.	Encapsula en un hilo de ejecución todas las funcionalidades relacionadas con la conectividad e intercambio de datos con la red.	No
Adc_interface	Cliente	Encapsula en un hilo de ejecución todas la comunicación con el ADC-Pi Plus realizando la recolección de los datos.	No

Continúa…

Tabla 3.1 Algunas clases probadas (continuación)

System_control	Cliente	Encapsula en un hilo el control de todos los subsistemas de la aplicación cliente.	No
graphics	Cliente	Se encarga de la visualización de la ventana de tendencias y la creación de los gráficos.	No
Db_thread	Server	Encapsula en un hilos las funcionalidades dedicadas a la conexión y el intercambio de información con la base de datos	No
Control_thread	Server	Encapsula en un hilo el control de todos los subsistemas de la aplicación servidor.	No
Data_process	Server	Contiene todas las funciones matemáticas y los filtro digitales que se utilizan para el tratamiento de la información recibida del campo	No
H_plotter	Server	Se encarga de graficar los datos históricos de las variables del sistema.	No

3.3.1 Pruebas de caja negra

Las pruebas de cajas negras son aquellas que se le realizan al software, partiendo de los requisitos funcionales del mismo, sin tener en cuenta su estructura interna. Las pruebas consisten en aplicar al sistema un conjunto de juegos de datos bajo determinadas condiciones y observar las salidas que se obtienen, para determinar si la funcionalidad que se prueba opera correctamente. Las pruebas de caja negra no son una alternativa a las técnicas de prueba de caja blanca, son un enfoque complementario.

Luego de realizadas las pruebas de caja negra de manera satisfactoria se pasó a probar el sistema a través de su conexión a un proceso real. Para ello se diseñó la maqueta mostrada

en la figura en la que se observar al sistema muestreando la señal de salida de un pequeño panel solar. Se le realizaron variaciones a la salida del panel a través del uso de un potenciómetro, evidenciándose que se obtenían buenas respuestas y con gran rapidez demostrándose así la robustez del sistema.

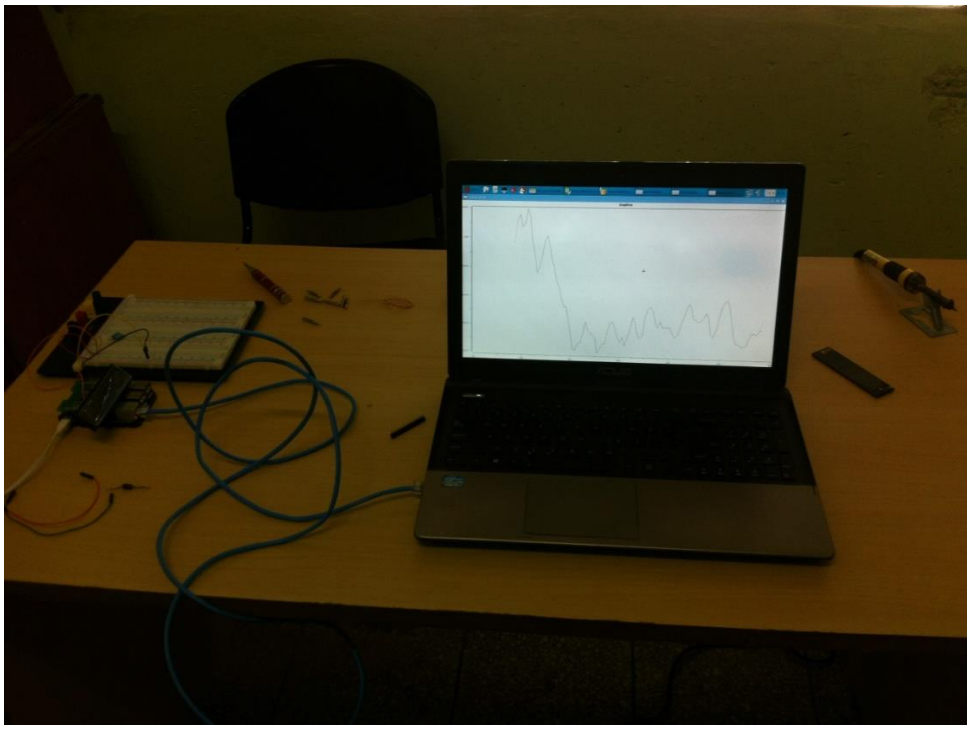

Figura 3.18. Maqueta de prueba del sistema.

3.4 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. Se realizó la descripción general de las funcionalidades de la Aplicación de Cliente y la Aplicación Servidor, exponiéndose las principales accione a llevarse a cabo para ejecutar la configuración y el monitoreo.

2. Se presentaron las principales pruebas que se le realizaron al sistema para su validación, demostrándose en cada una de ella la robustez y confiabilidad del mismo.

CONCLUSIONES

Una vez concluido este trabajo se puede llegar a las siguientes conclusiones:

1. Se diseñó una arquitectura de hardware que permite al sistema poder reponerse a fallas o cambios de cualquiera de los elementos que lo componen sin tener que someterse a restructuración o una nueva parametrización, demostrándose así su robustez.

2. Se analizaron las principales funcionalidades del software para el correcto funcionamiento el sistema y se diseñaron dos aplicaciones que permiten un desempeño eficiente del mismo.

3. Se realizaron pruebas que permitieron demostrar la robustez y eficiencia del sistema diseñado.

RECOMENDACIONES

Para aumentar la versatilidad y las bondades de este sistema, se realizan las siguientes recomendaciones:

1. Implementar un servidor web que se conecte con la base de datos del sistema contenida en el servidor, para permitir la supervisión.

2. El desarrollo de nuevas interfaces que permitan la conexión al sistema de sensores con otros estándares de salida menos utilizados en el mercado.

3. El desarrollo de módulos de detección de fallos y modelación a través de herramientas de inteligencia artificial que le aporten al sistema mayor exactitud y capacidad de adaptación a distintos procesos.

REFERENCIAS BIBLIOGRÁFICAS

Abinath, T.R.; Sudhakar, V. ; Sasikala, S., 2015. "Remote Host Process Control and Monitoring of Industry Appliances". *International Journal of Engineering Research and General Science*, 3 (2), pp. 960-965; DOI: 10.14257/astl.2015.109.08.

Aburva, A. ; Jeevabharathi, M., 2015. "Vehicular Monitoring and Tracking Using RASPBERRY PI". *International Journal of Innovative Research in Science, Engineering and Technology*, 4 (2), pp. 485-490; DOI: 10.15662/ijareeie.2014.0307062.

Almada-Lobo, F., 2015. "The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES)". *Journal of Innovation Management*, 3 (4), pp. 16-21.

Bhagyalakshmi, P.; Divya, G. ; Aravinda, N.L., 2015. "Raspberry PI And Wifi Based Home Automation". *International Journal of Engineering Research and Applications*, pp. 57-60; DOI: 10.1016/S0141-9331(02)00039-X CrossRef.

Bharathi, M. ; Mubeen, S., 2016. "Wireless Industrial Parameter Monitoring Using Raspberry PI 3". *International Journal of Innovative Technology and Research*, 4 (4), pp. 3315 – 3318.

Biswas, M.; Landge, R.; Mahajan, B. ; Kore, S., 2016. "Raspberry Pi Based Patient Monitoring System using Wireless Sensor Nodes". *International Research Journal of Engineering and Technology*, 3 (4), pp. 1693-1696.

Calvo, I.; Gil-García, J.M.; Recio, I.; López, A. ; Quesada, J., 2016. "Building IoT Applications with Raspberry Pi and Low Power IQRF Communication Modules". *Electronics*, 5 (54),pp. 2-17; DOI: 10.3390/electronics5030054.

Churcher, C., 2007. *Beginning Database Design.* Apress ISBN 978-1-59059-769-9.

Deitel, P. ; Deitel, H., 2013. *C how to program.* Pearson Education, ISBN 978-0-13-299044-8.

Darshini, B. ; Esakki, E., 2016. "Industrial Process Monitoring and Control using Raspberry Pi". *Journal of Engineering and Applied Sciences*, 11 (2), pp. 1384-1387.

Deshpande, A.; Pitale, P. ; Sangita, S., 2016. "Industrial Automation using Internet of Things". *International Journal of Advanced Research in Computer Engineering & Technology*, 5 (2), pp. 266-269.

Dimitrov, D.; Germanov, H.; Murdanliev, I.; Stoynov, M.; Valkov, M.; Bivas, M.; Kostov, N.; Nedyalkov, N.; Vasilev, N.; Donchev, P.; Hadjieva, P.; Ivanov, R.; Kirilov, R.; Todorov, R.; Zlatinov, S.; Staev, S.; Nakov, S.; Bozhikov, T.; Stoev, T.; Konov, T.; Georgiev, V.; Kolev, V.; Pavlov, Y. ; Yosifov, Y., 2013. *Fundamentals of Computer Programming with C#.* Svetlin Nakov & Co, ISBN 978-954-400-773-7.

Dixi, S., 2014. “Enhancement of Industrial Automation Using Single Board Computer Network”. *International Journal of Emerging Technologies and Engineering*, pp. 69-71.

Ejiofor, V. ; Oladipo, F., 2013. “Microcontroller based Automatic Water level Control System”. *International Journal of Innovative Research in Computer and Communication Engineering*, 1 (6), pp. 1390-1396.

Fall, K.R. ; Stevens, W.R., 2012. *TCP/IP illustrated.* Addison Weasly, ISBN 978-0-321-33631-6.

Ferdoush, S. ; Li, X., 2014. “Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications”. *Procedia Computer Science*, 34, pp. 103-110, DOI: 10.1016/j.procs.2014.07.059.

Gámez, Y.; Moreno, V. ; Martínez, Y., 2009 of Conference. “Herramienta interactiva de simulación de procesos para la enseñanza de ingeniería Automática: Nodo Virtual de Procesos”. En: (ed.)^(eds.), *Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology*. vol., pp 1-10.

Ganesh, U.M. ; A, K.R., 2015. “Raspberry Pi Home Automation Based on Internet of Things (IoT)”. *International Journal of Advanced Research in Computer and Communication Engineering*, 4 (12), pp. 301-304; DOI: 10.17148/IJARCCE.2015.41269.301.

Garcia, E., 2016. *Desarrollo de un controlador PID industrial de bajo coste mediante raspberry pi para control de temperatura*. Universidad Politécnica de Valencia.

Hankare, P.T.; Gujarathi, A.; Oza, S. ; Dangodara, K., 2015. “Raspberry Pi Based Sequential Switching”. *International Journal of Engineering and Technical Research* 3(1), pp. 209-211.

Isikdag, U., 2015. “Internet of Things: Single-Board Computers”. *Enhanced Building Information Models: Using IoT Services and Integration Patterns*. Cham: Springer International Publishing, ISBN 978-3-319-21825-0, pp. 43-53.

Jacobson, I.; Booch, G. ; Rumbaugh, J., 2000. *El proceso unificado de desarrollo de software*. Pearson Education, ISBN 84-7829-0362.

Jadhav, S. ; Hambarde, S., 2016. “Android based Automated Irrigation System using Raspberry Pi”. *International Journal of Science and Research*, 5 (6), pp. 2345-2351; DOI: 10.21275/v5i6.NOV164836.

Jahan, K.; Saifur, M.; Kaium, M. ; Mahfuz, S., 2013. “Raspberry Pi Image Processing Based Economical Automated Toll System”. *Global Journal of Researches in Engineering Electrical and Electronics Engineering*, 13 (13), pp. 35-41.

Karankumar, M.; Shreya, M. ; Kapil, R., 2014. “Analysis of TOI (Things of Internet) Industrial Monitoring System on Raspberry pi Platform”. *International Journal of Computer Science and Mobile Applications*, 2 (11), pp. 33-40.

Kasundra, C.T. ; Shirsat, A.S., 2015. “Raspberry-Pi Based Health Monitoring System”. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2015 (8), pp. 7147-7154; DOI: 10.15662/ijareeie.2015.0408083.

Kumar, S.R. ; Rameshkumar, S., 2013. “Industrial Temperature Monitoring and Control System through Ethernet LAN”. *International Journal Of Engineering And Computer Science*, 2 (6), pp. 1988-1991.

Meetali, V.R., 2016. “Monitoring of weather parameters Raspberry Pi”. *International Journal of Research In Science & Engineering*, pp. 611-615.

Michal, K.; Michal, H.; Cechovic, L.; Kapitulík, J. ; Jurecka, M., 2014 of Conference. “WSN for Traffic Monitoring using Raspberry Pi Board”. En *Federated Conference on Computer Science and Information Systems*. vol. 2, pp. 1023–1026; DOI: 10.15439/2014F310.

Patel, T.K.; Wadekar, U.; Wabale, A. ; Datkhore, S., 2015. “Appliances Control Using Ethernet and Raspberry PI”. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5 (3), pp. 494-497.

Peng, D. ; Wan, S., 2013. “Industrial Temperature Monitoring System Design Based on ZigBee and Infrared Temperature Sensing”. *Optics and Photonics Journal*, pp 277-280; DOI: 10.4236/opj.2013.32B065

Peterson, L. ; Davie, B., 2012. *Computer networks : a systems approach.* Elsevier, ISBN 978-0-12-385059-1.

Priyanka, L. ; Mahesh, K., 2016. “A Raspberry Pi Based Global Industrial Process Monitoring through Wireless Communication”. *International Journal of Advanced Research in Computer and Communication Engineering*, 5 (9), pp. 468-471; DOI: 10.17148/IJARCCE.2016.59105.

Raguvaran, K. ; Thiyagarajan, J., 2015 of Conference. “Raspberry PI Based Global Industrial Process Monitoring Through Wireless Communication”. En: *International Conference on Robotics, Automation, Control and Embedded Systems*. vol., DOI: 10.1109/RACE.2015.7097298.

Rahim, R.; Zainudin, M.; Ismail, M. ; Othman, M., 2014. “Image-based Solar Tracker Using Raspberry Pi”. *Journal of Multidisciplinary Engineering Science and Technology*, 1 (5), pp. 369-373;

Rao, P. ; Uma, S.K., 2015. “Raspberry Pi Home Automation with Wireless Sensors using Smart Phone”. *International Journal of Computer Science and Mobile Computing*, 4 (5), pp. 797-803.

Rawat, S.; Kumar, P. ; Jain, G., 2016. “Implementation of the Principle of Jamming for Hulk Gripper Remotely Controlled by Raspberry Pi”. En: Pant, M.; Deep, K.; Bansal, J.C.; Nagar, A. ; Das, K.N. (eds.), *Proceedings of Fifth International Conference on Soft Computing for Problem Solving: SocProS 2015, Volume 1*. Singapore: Springer Singapore, ISBN 978-981-10-0448-3, pp. 199-208.

Rivero, A. ; Lopez, E., 2009. *Programación y desarrollo de un nodo virtual de procesos*. Habana: Universidad Tecnológica de la Habana.

Rodríguez Molano, J.I.; Medina, V.H. ; Moncada Sánchez, J.F., 2016. “Industrial Internet of Things: An Architecture Prototype for Monitoring in Confined Spaces Using a Raspberry Pi”. En: Tan, Y. ; Shi, Y. (eds.), Data Mining and Big Data: First International Conference, DMBD 2016, Bali, Indonesia, June 25-30, 2016. Proceedings. Cham: Springer International Publishing, ISBN 978-3-319-40973-3, pp. 521-528.

Sachan, A., 2012. “Microcontroller Based Substation Monitoring and Control System with Gsm Modem”. *Journal of Electrical and Electronics Engineering*, 1 (6), pp. 13-21; DOI: 10.9790/1676-0161321.

Sankaranarayanana, S.; Wanb, A.T. ; Pusa, A.H., 2014. “Smart Home Monitoring using Android and Wireless Sensors”. Internationa Journal Engineering and Manufacturing, DOI: 10.5815/ijem.2014.02.02 .

Schildt, H., 2014. *Java The Complete Reference* McGraw-Hill Education, ISBN 978-0-07-180856-9.

Schumann-Bölsche, D. ; Schön, A.-M., 2015. “A Raspberry in Sub-Saharan Africa? Chances and Challenges of Raspberry Pi and Sensor Networking in Humanitarian Logistics”. *Procedia Engineering*, 107, pp. 263-272, DOI: 10.1016/j.proeng.2015.06.081.

Sharp, J., 2013. *Microsoft Visual C# 2013 Step by Step.* O’Reilly Media, ISBN 978-0-7356-8183-5.

Singh, K.; Lingaiah, J. ; Satish, B.P., 2016. “Raspberry Pi Based Global Industrial Process Monitoring Through Wireless Communication”. *International Journal of Innovative Technology*, 4 (13), pp. 2413-2422.

Stroustrup, B., 1997. *The C++ Programming Language* Addison Wesley, ISBN 0-201-88954-4.

Suradkar, P.; Singh, A.V.; Gokhale, A.; Jadhav, P. ; Khanuja, M., 2016. “Automated System Using Raspberry Pi for Remotely Controlling Devices Using Android”. *Imperial Journal of Interdisciplinary Research*, 2 (5), pp. 1570-1573.

Suresh, N.; Balaji, E.; Jeffry, K. ; Jenith, J., 2014. “Raspberry Pi Based Liquid Flow Monitoring and Control”. *International Journal of Research in Engineering and Technology*, 3 (7), pp. 122-125; DOI: 10.15623/ijret.2014.0307020 .

Swaroop, P.; Sheshank Reddy, Y.; Syed Saif, E. ; Sasikala, S., 2015. "The Real Time Temperature Sensing using Raspberry PI". *International Journal for Innovative Research in Science & Technology*, 1 (12), pp. 232-237.

Tanenbaum, A. ; Wetherall, D., 2011. *Computer Networks* USA: PRENTICE HALL, ISBN 978-0-13-212695-3.

Thelin, J., 2007. *Foundations of Qt Development.* ISBN 978-1-59059-831-3.

Tomar, V.S. ; Bhatia, V., 2015. "Low Cost and Power Software Defined Radio Using Raspberry Pi for Disaster Effected Regions". *Procedia Computer Science*, 58, pp. 401-407, DOI: 10.1016/j.procs.2015.08.047.

Turk, Y.; Demir, O. ; Gören, S., 2014. "Real Time Wireless Packet Monitoring with Raspberry Pi Sniffer". En: Czachórski, T.; Gelenbe, E. ; Lent, R. (eds.), *Information Sciences and Systems 2014: Proceedings of the 29th International Symposium on Computer and Information Sciences*. Cham: Springer International Publishing, ISBN 978-3-319-09465-6, pp. 185-192.

Villalonga, A.; Quiza, R.; Rodriguez, G. ; Haber, R., 2016 of Conference. "Arquitectura flexible y de bajo costo para sistema de monitoreo de procesos mecánicos". En: *18 Conferencia Cientifica de Ingeniería y Arquitectura* Habana, Cuba, noviembre 2016, vol., pp.

Vujović, V. ; Maksimović, M., 2015. "Raspberry Pi as a Sensor Web node for home automation". *Computers & Electrical Engineering*, 44, pp. 153-171, DOI: 10.1016/j.compeleceng.2015.01.019.

Wen-Tsai, S. ; Yao-Chi, H., 2011. "Designing an industrial real-time measurement and monitoring system based on embedded system and ZigBee". *Experments systems with applications*, pp. 4522–4529; DOI: 10.1016/j.eswa.2010.09.126.

Zhanitta, J. ; Sathya, B., 2015. "Industrial Application Monitoring and Control using Raspberry pi and TCP/IP Protocol". *International Journal of Advanced Research Trends in Engineering and Technology*, 2 (13), pp. 110-115.