Universidad de las Ciencias Informáticas

Título: Sistema de monitoreo basado en microcontroladores AVR.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores:

Antuanett Barrios Frometa

Yordan Carlos Pérez Attanaff

Tutores:

Ing. Julio Alberto Leyva Durán

Ing. Ormel Chávez Polo

“Año 57 de la Revolución”

La Habana, Cuba

Junio, 2015

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del _____.

_____

Antuanett Barrios Frometa

Firma del autor

_____

Yordan Carlos Pérez Attanaff

Firma del autor

_____

Ing. Julio Alberto Leyva Durán

Firma del tutor

_____

Ing. Ormel Chávez Polo

Firma del tutor

Dedicatoria

A la memoria de mis abuelos Alejandro Benitez y a Maria del Carmen Solana Jones por nunca dudar de mi capacidad para lograr mis metas, por siempre apoyarme e influenciar tanto en el desarrollo de mi aprendizaje. Por tantos consejos y por enseñarme otra manera de ver la vida. A ellos, le dedico estos años de tanto estudio.

Antuanett Barrios Frometa

A mis padres por todo el apoyo y cariño, por la guía y el ejemplo.

Yordan Carlos Pérez Attanaff

Agradecimientos

A mis padres, Leticia y Roberto por darme la vida, por ser los mejores padres del mundo. Por estar junto a mí siempre y por impulsarme a seguir adelante.

A mi hermano David por darme motivos para crecerme en la vida, por tanto amor y paciencia.

A mi abuelita Mauca por el inmenso cariño y apoyo que desde pequeña me ha ofrecido y por confiar ciegamente en mí.

A mi familia en general por ser tan especial.

A mis amigos (Mare, Tani, Anna, Dayma, Leiser, Nelson, Yugi, José y Jorge Luis) por enseñarme la parte divertida de la universidad. En especial, Jorge Luis por ser más que un amigo incondicional al que nunca quisiera perder.

A mis tutores por tanta paciencia y dedicación.

A todos los profesores que me ayudaron durante la carrera, a todos muchísimas gracias sin ustedes no hubiera podido lograr tanto.

Antuanett Barrios Frometa

A todos mis amigos, a David, Lian, Yordanis, Angel, Edo, Camilo, el Cuenca, el Jabao por ser mis hermanos, a los que aún están aquí, y a los que no, gracias por soportarme.

A mi familia, por su apoyo, a mi tía Yohandra, gracias por tanto esfuerzo.

A mis profesores, que no tengo palabras para agradecer tanta paciencia y dedicación.

A Lety, a Robert, a Mary, gracias por abrirme las puertas y por ser una familia.

A mi tutor, por confiar en mí y por todas esas ideas.

A la gente del Kenpo, al Joe, a la gente de MoonLight, del café, a la gente de humanOS, sin ustedes la UCI se queda vacía.

Y a todos los que de alguna manera tuvieron algo que ver con estos 5 años, gracias.

Yordan Carlos Pérez Attanaff

Resumen

El desarrollo en la aplicación de la domótica a nivel mundial ha aumentado debido a las ventajas y servicios que esta ofrece. El ahorro de energía, el confort y la seguridad son aspectos esenciales provistos en las viviendas inteligentes así como el control y monitoreo de las funciones que intervienen en estas. Estas tareas se llevan a cabo gracias al empleo de sistemas embebidos, los que son ampliamente utilizados en cualquier ámbito de la sociedad por la flexibilidad que ofrecen

El presente trabajo investigativo expone el desarrollo de un sistema de monitoreo basado en dos microcontroladores AVR ATMega8, integrado por un sistema embebido y una aplicación de monitoreo. La solución es capaz de generar alertas a partir de las especificaciones del usuario sobre un valor determinado obtenido mediante sensores analógicos, así como modificar valores en los microcontroladores, lo que permite la interacción con actuadores. El firmware de los microcontroladores se desarrolla en el lenguaje C y la aplicación de monitoreo en C++ utilizando el framework Qt y la biblioteca libmodbus. Ambos subsistemas se comunican a través del protocolo Modbus sobre el estándar RS-232.

Palabras clave: AVR, domótica, microcontrolador, Modbus, monitoreo, sistemas embebidos.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Introducción

La comodidad, el ahorro energético, la seguridad, el control y monitoreo de los bienes tanto patrimoniales como personales son algunos aspectos que siempre el hombre ha deseado tener en lugares donde se reside como en donde se produce. Estos aspectos le han dado auge al desarrollo y a la aplicación de la domótica, de esta misma manera las tecnologías de automatización, de control y de la informática son incluidas en el hogar. La domótica junto a las tecnologías antes planteadas ofrecen servicios para el ahorro de energía, la seguridad y el confort no solo en las viviendas sino también en locales como oficinas, bibliotecas, laboratorios, bancos, entre otros. Su empleo permite la disminución de costos y la optimización de los recursos como consecuencia de la automatización de tareas, el incremento de la seguridad, una mayor eficiencia en los servicios de mantenimiento, mejor calidad de vida mediante el control de climatización e iluminación en tiempo real etc.

Los hogares, viviendas o locales que cuentan con los servicios de la domótica son llamados inteligentes; tienen instalado sistemas de medición, monitoreo, control y regulación de las funciones que intervienen en ellos, mediante distintos equipos como controladores [1], sensores o microcontroladores (MCU) que están interconectados a través de un BUS[2] de comunicación. Estos equipos se encargan de recibir la información y procesarla de acuerdo a la petición que haga el usuario[3] de encender o apagar un dispositivo en la vivienda y a través de ellos se puede detectar incendios, fugas de gas, escapes de agua, además mediante ellos la seguridad y el control del hogar es mayor. Principalmente la seguridad del hogar ha jugado un rol importante, esta combina varias funciones como: la anti-intrusión, la detección de humo, de gas, de fuego, de inundaciones, la alerta médica, la teleasistencia.

En el contexto global, la seguridad del hogar ha sido un tema de mucha importancia para el desarrollo de las nuevas tecnologías. Es así que forma parte de las características de los sistemas domóticos junto a otras como: la flexibilidad, el ahorro energético y la comunicación con el usuario final, para de esa manera cumplir con los estándares mundiales existentes hoy en día. Actualmente, existen diferentes organizaciones internacionales que se encargan de publicar normas y recomendaciones para la correcta aplicación de los sistemas domóticos en general, incluyendo a los de seguridad como ISO[4], IEC[5], UIT[6], entre otras. Sin embargo, dichas normas se aplican

1 Son los dispositivos que gestionan el sistema según la programación y la información que tienen o que reciben.

2 Puerto estándar de comunicación, consiste en la transferencia entre dos dispositivos.

3 Individuo que utiliza una computadora, sistema operativo, servicio o cualquier sistema.

4 Organización Internacional de Normalización.

5 Comisión Electrotécnica Internacional.

6 Unión Internacional de Telecomunicaciones.

según el criterio voluntario de las empresas que proveen estas soluciones, lo cual no permite una regulación obligatoria que asegure el correcto funcionamiento de los sistemas (Iberico, 2015).

La aplicación de los sistemas domóticos en los hogares de países desarrollados como Alemania, Italia, Francia, Inglaterra, Canadá, Japón y Estados Unidos tienen porcentajes más altos (Sociales, 2015). Al contrario de los países pocos desarrollados, donde la actual aplicación de la domótica no está al alcance de la mayoría.

Cuba aún no presenta el mismo nivel de aplicación de la domótica en comparación a países desarrollados y actualmente la mayoría de los locales no cuentan con sistemas de seguridad debido a que representan unidades con bajos niveles de eficiencia e integración, siendo el factor humano el componente principal, el cual se encarga de vigilar las instalaciones y que, en la mayoría de las veces, no cuenta con las herramientas necesarias para prevenir y alertar un incidente. Estos sistemas no han sido ampliamente adoptados, principalmente, debido al costo de implementación y mantenimiento que involucran. Además no ofrecen una flexibilidad adecuada a la infraestructura existente (Iberico, 2015).

A pesar de lo antes expuesto y de Cuba tener un bloqueo económico, comercial y financiero impuesto por Estados Unidos que impide a la isla el acceso al mercado y la obliga a invertir mucho al tener que recurrir a otros muy distantes, ha contado siempre con la innovación e invención, usando los recursos disponibles, ahorrando los materiales y recursos financieros para el desarrollo de la sociedad. Es así como actualmente se avanza en la informatización del país y cuenta con la ayuda de universidades como la de Cienfuegos, en específico la facultad de Ciencias Económicas y Empresariales y la de las Ciencias Informáticas (UCI), en particular el Centro de Informática Industrial (CEDIN). Estas universidades han realizado estudios, proyectos e innovaciones referentes al tema de la domótica contando con la ayuda de centros de investigación y desarrollo. Ejemplo del avance en la esfera de la domótica en el país es la consideración de la universidad de Cienfuegos en proponer como prioridad construir edificios sostenibles, e incorporarles tecnologías energéticas amigables con el entorno, además de los sistemas automatizados de control, para que las potencialidades que los edificios inteligentes presentan permitan obtener beneficios económicos, larga vida útil y una importante reducción del impacto ambiental, también de llevar a cabo algunos proyectos demostrativos donde se utilicen varias fuentes renovables de energía que permitan evaluar los aspectos asociados a las condiciones económicas, laborales y ambientales (Felipe, 2015). Mientras que la UCI prestigia con el posicionamiento en el mercado de productos de alta confiabilidad y calidad, realiza proyectos demostrativos en esta esfera. Particularmente el CEDIN en el Departamento de Aplicaciones que se encuentra actualmente desarrollando un producto orientado a un determinado local que cuenta con activos de alto valor para el negocio donde los equipos operan las 24 horas del día, dígase un nodo de servidores, estación de bombeo utilizando MCU AVR de Atmel, de tipo ATMega8. Este local se encuentra aislado de la entidad (empresa,

institución) a la que pertenece y el control de acceso al mismo se hace manual, además no cuenta con un vigilante que pueda avisar o prevenir en caso de ocurrir un incidente ni con un mecanismo para monitorear las variables incidentes del mismo y poder responder a cualquier problema que pueda interferir en el buen funcionamiento de alguno de los medios que contiene.

Para darle solución se plantea como problema: ¿Cómo monitorear las variables incidentes de un local aislado desde una entidad? de este se deriva que el objeto de estudio son los sistemas de monitoreo. El objetivo para resolver el problema antes mencionado es desarrollar un sistema de monitoreo basado en MCU AVR de Atmel tipo ATMega8 de las variables incidentes en un local. Siendo el campo de acción: los sistemas de monitoreo basados en MCU AVR.

Para responder a lo antes planteado se definen las siguientes tareas de investigación:

- Revisión de bibliografías con temas relacionados con el objeto de estudio para solucionar el problema planteado.
- Estudio de las características técnicas y del lenguaje de programación del microcontrolador AVR ATMega8 para la implementación del *firmware*[7].
- Estudio de la comunicación por el protocolo Modbus mediante la interfaz RS-232 desde el microcontrolador AVR ATMega8 y la computadora para el desarrollo del sistema de monitoreo.
- Selección de las herramientas, tecnologías y metodologías para el desarrollo del sistema.
- Análisis, diseño y modelado del sistema a desarrollar, para lograr menor cantidad de errores y mayor calidad del producto.
- Implementación de la aplicación de monitoreo y del sistema embebido para lograr un producto terminado.
- Realización de pruebas funcionales al sistema para garantizar la calidad.

A continuación se reseña el uso dado a diferentes métodos en el desarrollo de la investigación:

Métodos teóricos:

- Mediante el método Análisis-Síntesis se analizan los contenidos de las informaciones obtenidas a partir de las teorías y documentos relacionados con la domótica y en particular en uno de sus objetivos: la seguridad, la relación de los microcontroladores y la computadora, permite buscar rasgos para elaborar y fundamentar la propuesta de solución

7 Conjunto de instrucciones de un programa informático que se encuentra registrado en una memoria ROM, flash o similar.

al problema del presente trabajo. La evaluación de todas las fuentes consultadas, los resultados del diagnóstico de la realidad objeto de estudio y los criterios del autor como premisas fundamentales para el diseño de la propuesta.

- El método Análisis Histórico – Lógico se emplea en la búsqueda y análisis de bibliografías de trabajos investigativos, artículos científicos, revistas sobre el campo de acción y el objeto de estudio publicados para definir que herramientas, tecnologías y metodologías a emplear en el transcurso de todo el trabajo investigativo.
- La modelación es utilizada para crear algunos artefactos de ingeniería de software para modelar los eventos y la solución propuesta.

Métodos empíricos:

- Método experimental: Es aplicado durante todo el proceso de construcción de la solución analizando el correcto funcionamiento del firmware embebido en el microcontrolador, realizando pruebas intermedias que permitan y aseguren el avance de la solución.

El contenido del presente documento se encuentra estructurado en 4 capítulos. En el **CAPÍTULO 1: Fundamentación Teórica** se muestran las principales definiciones utilizadas en la investigación. Se hace referencia a la metodología de desarrollo, las herramientas y lenguajes de programación que apoyan el desarrollo de la solución. En el **CAPÍTULO 2: Características del sistema** se describe la propuesta de solución, se muestra las fases iniciales de la metodología de desarrollo seleccionada, en la que se plantean las Historias de Usuarios (HU), se especifican los requerimientos y las características del sistema y la propuesta de solución. Dentro del **CAPÍTULO 3: Diseño e Implementación** se describen la arquitectura sistema y cómo se enmarca dentro de la propuesta de solución, los patrones arquitectónicos y los de diseño que se utilizan en el proceso de desarrollo de la solución. Se crean las tarjetas CRC y las tareas de la ingeniería para desglosar las actividades comprendidas en cada Historia de Usuario. También se definen los estándares de codificación utilizados en la implementación del sistema. En el **CAPÍTULO 4: Pruebas y resultados** se realizan las pruebas necesarias para comprobar que el sistema desarrollado tiene la calidad requerida, cumple con el objetivo y los requisitos planteados por el cliente.

CAPÍTULO 1: Fundamentación Teórica

En el presente capítulo se abordan los elementos teóricos necesarios para darle solución a la situación problemática a resolver. Se plantean conceptos asociados a la domótica, los sistemas embebidos, el firmware, y a los dispositivos usados en los sistemas domóticos. También se hace referencia a la metodología de desarrollo, las herramientas y lenguajes de programación a emplear durante todo el desarrollo de la solución.

1.1 Sistemas Domóticos

En los últimos años se han notado algunos cambios muy significativos e interesantes relacionados con lugares donde se residen, se produce y se trabaja que han modificado, de manera positiva, la comodidad y la seguridad. Los mismos han permitido la permanente comunicación, la automatización, el control remoto de múltiples equipos y el ahorro energético. En específico, la domótica engloba todas las ventajas que ha traído consigo estos cambios.

“La tecnología aplicada al hogar”, conocida como Domótica, integra automatización, informática y nuevas tecnologías de comunicación; todas ellas dirigidas a mejorar la comodidad, la seguridad y, en definitiva, el bienestar dentro de los hogares (Domótica, 2007). En algunas bibliografías consultadas el término domótica varía, algunos de ellos son los siguientes:

- Es la adopción, integración y aplicación de las nuevas tecnologías informáticas y comunicacionales al hogar. Incluye principalmente el uso de electricidad, equipos y dispositivos eléctricos y electrónicos, sistemas informáticos y diferentes equipos de telecomunicaciones, así como la incorporación de la telefonía móvil e Internet (Felipe, 2015).
- Se refiere a la automatización y control de los sistemas domésticos como la iluminación, climatización, persianas y toldos, puertas y ventanas, cerraduras, riego, electrodomésticos, suministro de agua, suministro de gas, suministro de electricidad, etc. El sistema permite el encendido, apagado y regulación de todos estos sistemas de manera automática, basándose en parámetros que el usuario puede determinar (Domótica, 2007).

A modo de resumen, la domótica puede definirse como adopción, integración y aplicación de las nuevas tecnologías informáticas y comunicacionales al hogar ofreciendo servicios aplicados al ámbito doméstico.

La domótica se remota a la década de los setenta, cuando tras muchas investigaciones aparecieron los primeros dispositivos de automatización de edificios basados en la aún exitosa tecnología X-10 (protocolo de comunicaciones para el control remoto de dispositivos eléctricos para transmitir señales de control entre equipos de automatización del hogar) (Domótica, 2007).

Las primeras edificaciones inteligentes que surgieron empleaban un consumo de energía mínimo para operar, y con el paso del tiempo se les fueron incorporando servicios que optimizaron su funcionalidad (Domótica, 2007). Actualmente, el concepto de edificio inteligente se ha adaptado a los avances tecnológicos sucesivos, de manera que este concepto se aplica tanto para construcciones de oficinas, como a hospitales, hoteles, bancos, museos, casas, etc. (Tiscornia, 2015), donde la automatización y el control de objetos de la casa son capaces de ser monitoreados desde una computadora, teléfono inteligente o cualquier otro dispositivo.

En la mayor parte de la bibliografía consultada para referirse a los servicios que ofrece la domótica se utiliza el término: “pilares" y aunque algunos autores incluyen también “las comunicaciones” y “la telegestión y accesibilidad”; lo más generalizado es considerar tres pilares que son: gestión energética, confort y seguridad.

La gestión energética está dada por la posibilidad del sistema domótico de encargase de gestionar el consumo de energía, mediante actuadores y sensores que controlan el apagado y encendido de luces y equipos electrodomésticos en función de las condiciones de presencia, temperatura, horarios y otras, de un modo automático. Con respecto al confort, la domótica proporciona una serie de comodidades relacionadas con el control automático de servicios como: la climatización, la iluminación, la apertura y cierre de puertas y ventanas entre otros. Relacionado con la seguridad se debe destacar que la que proporciona un sistema domótico es más amplia que la que puede proporcionar cualquier otro sistema, pues integra tres campos de la seguridad que normalmente están controlados por sistemas distintos que son los siguientes:

- Seguridad de los bienes: en esta dirección se gestiona el control de acceso y de presencia y es posible simular mediante un sistema domótico la presencia de personas en el inmueble.
- Seguridad de las personas: especialmente para las personas mayores, personas minusválidas y enfermas. Se puede tener acceso automático mediante un nodo telefónico, por ejemplo a la policía.
- Incidentes y averías: mediante sensores, se pueden detectar los incendios y las fugas de gas y agua, la ocurrencia de tormentas o descargas eléctricas y tomar las medidas necesarias que pueden ser entre otras la activación de una alarma, la llamada a los bomberos o a la policía, la apertura o cierre de ventanas o la desconexión de equipos electrodomésticos.

Los sistemas domóticos en la actualidad tienen características generales entre las que se destacan: integración, interrelación, facilidad de uso, control remoto, fiabilidad y actualización, a continuación se presenta una breve explicación de cada una de ellas.

- Integración: El sistema integra bajo su control el funcionamiento de los múltiples sistemas que se decida incluirle, de este modo, los usuarios no tienen que estar pendientes de los diversos equipos autónomos, con su propia programación, indicadores situados en diferentes lugares, dificultades de interconexión entre equipos de distintos fabricantes; sino que es el propio sistema quien lo hace centralizadamente. Esta es la principal virtud que debe tener un sistema domótico, dado que permite el control del inmueble y por tanto deberá integrar todos sus sistemas.

- Interrelación: Una de las principales características que debe ofrecer un sistema domótico es la capacidad para relacionar diferentes elementos y obtener una gran versatilidad y variedad en la toma de decisiones. Así, por ejemplo, es sencillo relacionar el funcionamiento del aire acondicionado con el de otros electrodomésticos, con la apertura de ventanas, con que la habitación esté ocupada o con la temperatura existente.

- Facilidad de uso: Con sólo mirar las interfaces de control se puede tener una completa información del estado de todos los equipos o sistemas del inmueble y desde allí se puede realizar cualquier modificación de modo sencillo, incluida la incorporación o eliminación de elementos (partes del inmueble, equipos electrodomésticos, etc.) del control del sistema. Por lo general el uso de una interfaz gráfica propicia una comunicación fácil con el usuario y le permite rápidamente obtener información sobre aspectos como: la temperatura, humedad, los equipos que están funcionando, si hay alguien en las proximidades, o el nivel de agua que tiene el tanque.

- Fiabilidad: Está avalada por la potencia, rapidez y fiabilidad que tienen las computadoras actuales, unida a la utilización de un sistema de alimentación ininterrumpida que garantice el funcionamiento de las prestaciones básicas del sistema.

- Actualización: Mantener actualizado el sistema es tan sencillo como la simple instalación de nuevas versiones o actualizaciones del programa informático en la computadora, lo cual constituye un proceso elemental y con consumo de tiempo mínimo. Esto es posible dado que la lógica del funcionamiento completo del sistema recae sobre el software[8] y no en los equipos que se encuentren instalados al sistema, que serán automáticamente reconocidos por la nueva versión instalada.

- Control remoto: Esta es una potencialidad de los sistemas domóticos surgida con el desarrollo experimentado por las comunicaciones y está dada por el uso desde cualquier

8 Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

lugar del mundo, de las mismas posibilidades de supervisión y control disponibles localmente, mediante conexión telefónica desde otra computadora.

1.2 Sistemas embebidos o empotrados

Con el avance tecnológico en el sector de la electrónica y microelectrónica se encuentran los llamados sistemas embebidos (SE) presentes en muchos lugares que prácticamente pasan por desapercibidos, definidos por algunos autores como:

- Sistema informático que se encuentra físicamente incluido en un sistema de ingeniería más amplio al que supervisa o controla. Los sistemas embebidos se encuentran en multitud de aplicaciones, desde la electrónica de consumo hasta el control de complejos procesos industriales. Están presentes en prácticamente todos los aspectos de la sociedad como, teléfonos móviles, automóviles, control de tráfico, ingenios espaciales, procesos automáticos de fabricación, producción de energía, aeronaves, etc. Además, el auge de los sistemas embebidos está en constante aumento, ya que cada vez más máquinas se fabrican incluyendo un número mayor de sistemas controlados por computador. Un ejemplo cercano es la industria del automóvil, ya que un turismo actual de gama media incluye alrededor de dos docenas de estos automatismos (ABS, *airbag*, etc.). Otro ejemplo cotidiano son los electrodomésticos de nueva generación, que incluyen sistemas embebidos para su control y temporización. Hoy día son tantas las aplicaciones de estos sistemas que son mucho más numerosos que los sistemas informáticos "convencionales" o de propósito general. La tendencia es que estos computadores estarán en "todas partes" dando lugar a lo que se llama computación ubicua, *pervasive* ("impregnada") *computing* o inteligencia ambiental (Zamorano, 2014).
- Los SE están construidos para controlar una función o serie de funciones y no están diseñados para ser programados por el usuario final de la misma manera en que lo es una PC (Olivia, 2012).

Los SE cumplen con características tales como (Computing, 2010):

- Compatibles con una amplia gama de procesadores y arquitecturas de procesador.
- Sensibles económicamente.
- Tienen restricciones de tiempo real.
- Las implicaciones de un fallo de software es mucho más grave que en los sistemas de escritorio.
- Suelen tener limitaciones en el consumo de energía.

- Tienden a tener menos recursos del hardware que los sistemas de escritorio.
- Suelen almacenar todo su código objeto en la memoria ROM.
- Requieren para su desarrollo de herramientas y técnicas especializadas junto a métodos de diseños simples y eficientes.
- A menudo deben operar bajo condiciones ambientales extremas.

1.3 Firmware

Llamado como “el nivel más bajo de las aplicaciones empotradas”, los conceptos de *firmware* son breves y pocas veces abarcan toda la dimensión de lo que realmente significa *firmware*. Seguidamente se expone el concepto incluido en el Diccionario FOLDOC de Computación: “Software guardado en memoria ROM o ROM programable”. Ciertamente la palabra surge en 1968 por la unión del adjetivo “*firm*” (firme, rígido, estable) y la terminación de software “*ware*”, (Computing, 2010).

- El *firmware* tiene la responsabilidad de “la conducta de un sistema cuando se enciende por primera vez”, (Computing, 2010).

En resumen, el *firmware* es un bloque de instrucciones bajo algún lenguaje de programación integrado al hardware. Se comporta como una interfaz para que los SE hagan uso de los recursos del HW.

El desarrollo de *firmware* es un proceso diferente al desarrollo de aplicaciones para computadoras personales. El código debe de ser minimalista, eficiente, en tiempo real, estable, fácil de leer, etc. Estas características están estrechamente relacionadas con las particularidades de los sistemas embebidos enunciadas anteriormente.

1.4 Dispositivos usados en sistemas domóticos y embebidos

Con el desarrollo experimentado por la electrónica y la informática a escala mundial, han surgido múltiples dispositivos y medios de comunicación que son utilizados por los sistemas domóticos, todos ellos se pueden clasificar en uno de los cinco grupos siguientes:

- Controladores: Los controladores son los dispositivos que gestionan el sistema según la programación y la información que tienen o que reciben. Puede haber uno o varios controladores distribuidos por el sistema.
- Actuadores: Los actuadores son los dispositivos electrónicos o electromecánicos capaces de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (apagar/encender, subir/bajar, apertura/cierre, etc.), modificando el entorno.

- Sensores: Los sensores son los dispositivos electrónicos que se encargan de acondicionar diferentes tipos de información a un formato reconocido por los elementos de procesamiento. Un sensor puede ser un botón o un reconocedor de huella digital, pero si los elementos de procesamiento son digitales, en ambos casos la salida va ser codificada en 1's y 0's. Con los sensores se pueden monitorear distintos parámetros como: temperatura, humedad, velocidad, etc.
- Bus: Es el medio de transmisión que transporta la información entre los distintos dispositivos y puede ser por un cableado propio, por las redes de otros sistemas (red eléctrica, red telefónica, red de datos) o de forma inalámbrica.

Es preciso aclarar que todos los dispositivos del sistema de domótica no tienen que estar físicamente separados, sino varias funcionalidades pueden estar combinadas en un dispositivo, por ejemplo un equipo puede ser compuesto por un controlador, actuadores, sensores y varias interfaces.

A continuación se hace una breve descripción de algunos de los dispositivos empleados en los sistemas domóticos y embebidos.

1.4.1 Controladores

Existen diversos controladores como tipos de periféricos, y es común encontrar más de un controlador posible para un mismo dispositivo, algunos de los más usados son Controlador Lógico Programable (PLC), Controlador de Alta Disponibilidad (PAC), computadora y Tarjetas de Adquisición de Datos (DAQ, siglas en inglés).

- **Controlador Lógico Programable (PLC)**

Un autómata programable industrial (API) o PLC, es un equipo electrónico, programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente de tipo industrial, procesos secuenciales.; trabaja en base a la información recibida por los captadores (Automation, 2015). Se emplean en instalaciones donde es necesario un proceso de maniobra, control, señalización. Su aplicación abarca desde procesos de fabricación industriales de cualquier tipo a transformaciones industriales, control de instalaciones, etc. Las funciones básicas son:

- Detección: lectura de la señal de los captadores distribuidos por el sistema de fabricación.
- Mando: elaboración y envío de las acciones al sistema mediante los accionadores y preaccionadores.
- Diálogo hombre máquina: mantenimiento de un diálogo con los operarios de producción, obedeciendo sus consignas e informando del estado del proceso.

- Programación: Introducción, elaboración y cambio en el programa de aplicación del autómata. El diálogo de programación debe permitir modificar el programa incluso con el autómata controlando la máquina.

Los PLC son ampliables y escalables, tienen la posibilidad de gobernar varias máquinas con un mismo autómata, tienen un menor tiempo empleado en la elaboración de proyectos debido a que no es necesario dibujar el esquema de contactos ni simplificar las ecuaciones lógicas, cuenta con un mínimo espacio de ocupación; algunos de los inconvenientes que presentan son el precio de los componentes, la centralización del sistema, por lo que si ocurre algún fallo en la Unidad Central de Procesamiento (CPU, siglas en inglés) falla todo el sistema, es muy laborioso a nivel de programación y montaje, debe hacerse todo desde cero.

- **Controlador de Alta Disponibilidad (PAC)**

Los PAC son utilizados para el control de procesos, adquisición de datos, monitoreo remoto, visión artificial y control de movimiento. Son capaces de funcionar como un nodo de computación completa y compleja en una red distribuida (Alicante, 2015).

Los PAC son fáciles de usar, instalar, mantener, ampliar, están a un bajo precio pero en comparación a los PLC son altos, garantizan la máxima disponibilidad de los procesos sin embargo la precisión de los procesos analógicos es baja, la redundancia de reguladores es automática, la anti-interferencia que presenta es débil, debido a que maneja frecuencias altas y no resiste a las interferencias de dispositivos con las mismas frecuencias.

- **Computadora**

La computadora incluye microprocesador (CPU), memoria y dispositivos de entrada/salida, junto a los buses que permiten la comunicación entre ellos; la distingue de otros dispositivos similares, la realización de tareas muy diversas cargando distintos programas en la memoria para que los ejecute el procesador. Realiza funciones con un índice menor de errores, procesa y almacena la información pero representan una fuerte inversión, ya que los equipos son costosos y requieren el acondicionamiento del área laboral además de un cambio constante de la tecnología.

- **Tarjetas de Adquisición de Datos (DAQ, siglas en inglés)**

Las Tarjetas de Adquisición de Datos se encargan de las conversiones de señales desde analógica a digital y la comunicación con la computadora. Sus características más relevantes son: el número de canales analógicos, la velocidad de muestro, la resolución, el rango de entrada, la capacidad de temporización y la forma de comunicarse con el computador.

El rango de entrada indica los márgenes entre los que debe estar la señal de entrada para que pueda ser convertida. Las tarjetas de adquisición de datos suelen dar varias posibilidades que se pueden seleccionar por hardware o por software (Santiago, 2013).

Estas 4 características vienen a determinar la capacidad y la precisión de la tarjeta de adquisición:

- A mayor número de canales, mayor capacidad.
- A mayor velocidad de muestreo, mayor capacidad.
- A mayor resolución, mayor precisión.
- A menor rango de entrada, mayor precisión.

La forma en que se comunica la tarjeta con la computadora es mediante entrada - salida por interrupción, lo normal o mediante acceso directo a memoria (DMA, siglas en inglés). En aquellos casos en los que el flujo de datos puede ser elevado.

1.4.2 Procesador Digital de Señal (DSP, siglas en inglés)

Los DSP son microcontroladores o microprocesadores diseñados específicamente, tanto en arquitectura hardware como conjunto de instrucciones, para realizar tareas típicas de procesamiento digital de señales en tiempo real. Se fabrican bajo un formato semejante al de un microcontrolador (CPU+Memoria+Periféricos de Entrada y Salida en un solo chip) pero con una arquitectura especialmente diseñada para realizar la tareas mas habituales en procesamiento digital de señales, de forma rápida (Miñarro, 2009)).

Algunas características de los DSP son (Miñarro, 2009):

- Uso de CPUs con arquitecturas Harvard modificadas de 16 y 32 bits. Versiones de coma fija[9] y coma flotante[10].
- Gran cantidad de periféricos de Entrada/Salida integrados, especializados en la transferencia en tiempo real de grandes volúmenes de datos.
- Incluyen gran cantidad de memoria dentro del chip.
- No empleo de sistema operativo.
- Sistemas de desarrollo típicos basados en ANSIC.
- Coste del chip medio.
- La placa base en la que se incorporan las placas de circuito impreso (PCB) suele tener una complejidad media y a veces forma parte de un sistema superior, como por ejemplo, la tarjeta de sonido de un PC.

1.4.3 Microprocesadores

Los microprocesadores según varias bibliografías consultadas los consideran como:

[9]Notación científica que consiste en destinar una cantidad fija de dígitos para la parte entera y otra para la parte fraccionaria.

[10] Notación científica usada en los microprocesadores con la cual se pueden representar números racionales extremadamente grandes y pequeños de una manera muy eficiente y compacta, y con la que se pueden realizar operaciones aritméticas.

- Un chip que incluye básicamente la CPU y circuitería relacionadas con los buses de datos y memoria. Para poder realizar su tarea se necesitan otros chips adicionales (Sistema mínimo) tales como memoria, circuitos de entrada salida Entrada/Salida (I/O) y reloj. (Harper, 2010).
- Una pastilla de circuito integrado que contiene todos los elementos necesarios para realizar los complejos cálculos numéricos y lógicos que se ejecutan en una computadora.

En la tabla # 2 se describen algunas características de los microprocesadores.

1.4.4 Microcontroladores

Los MCU son según diferentes bibliografías:

- Un Circuito Integrado con una escala de integración muy grande que internamente consiste en una Unidad Central de Procesamiento (CPU*, Central Processing Unit),* memoria para datos, memoria para código, temporizadores, fuentes de interrupción y otros recursos necesarios para el desarrollo de aplicaciones, generalmente de propósito general. A pesar del MCU incluir prácticamente los elementos suficientes y necesarios para ser considerado una computadora en un circuito integrado, frecuentemente no es tratado como tal, ya que su uso típico consiste en el desempeño de funciones de "control" interactuando con el "mundo real" para monitorear condiciones (a través de sensores) y en respuesta de ello, encender o apagar equipos (por medio de actuadores) (Cruz, 2009).
- Un circuito integrado que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada. Debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna (Cohen Manrique, 2005)
- Un computador completo, aunque de limitadas prestaciones, que está contenido en un chip de un circuito integrado (Crips, 2004).
- Un sistema de control completo basado en un microprocesador pero construido en un solo chip (Angulo, 2003).
- Un circuito integrado programable que contiene todos los componentes de un procesador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de "controlador incrustado" .El MCU es un computador dedicado. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar, y todos los recursos complementarios disponibles tienen como única finalidad: atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada (Olivia, 2012).

A modo de resumen un microcontrolador puede definirse como un circuito programable que contiene todos los componentes de un computador y controla el funcionamiento de una determinada tarea, es decir, solamente puede ejecutar una actividad específica.

Los microcontroladores dado el amplio peso que tienen en las aplicaciones de carácter industrial y consumo, es posible manejar un sistema de desarrollo y la posibilidad de abordar proyectos prácticos no muy complejos en poco tiempo (Miñarro, 2009). Están enfocados en sistemas de propósito específico, sistemas que se crean con una funcionalidad única, la cual no va a cambiar durante su tiempo de vida útil. Por ejemplo: cajas registradoras, videojuegos, máquinas de escribir, hornos de microondas, etc. Se clasifican de acuerdo a la arquitectura de su Unidad Central de Procesamiento (CPU), la cual puede ser Repertorio de Instrucciones Complejo (RISC, siglas en inglés) o un Repertorio de Instrucciones Reducido (CISC, siglas en inglés). La filosofía RISC describe la simplicidad en el hardware[11] (HW), mientras que CISC plantea que el programador escriba menos, prácticamente todos los nuevos microcontroladores son RISC. Otro elemento para la clasificación es el tamaño de datos que pude ser de 4, 8, 16 y 32 bits. Con base en el acceso a las memorias de código y datos, se tienen 2 modelos Arquitectura Von Neumann (requiere de un solo espacio de memoria para almacenar instrucciones y datos) y Arquitectura Harvard (tiene un espacio para el almacenamiento de código físicamente separado del espacio de almacenamiento de datos). Por su capacidad de expansión: Un MCU puede tener una Arquitectura Abierta (acondicionado para el uso de memoria externa) o Cerrada (Cohen Manrique, 2005). Los MCU populares son de 8 bits y dentro de sus instrucciones incluyen algunas que permiten evaluar o modificar los bits individuales. Disponen de 8 Kbyte para instrucciones y 1Kbyte para datos. Trabajan a frecuencias máximas de 20 MHz. Incluyen 2 recursos que les permite obtener información generada por dispositivos analógicos: el convertidor analógico digital (ADC, siglas en inglés) y el comparador analógico (AC, siglas en inglés).

Un ADC recibe una muestra de una señal analógica y a partir de esta genera un número (valor digital), mientras que un AC es un recurso que indica la relación existente entre 2 señales analógicas externas. Es muy útil para aplicaciones donde no precisa conocer el valor digital de una señal analógica, sino que es suficiente con determinar si esta es mayor o menor que alguna referencia (Cruz, 2009).

Existe una variada serie de periféricos usados por los fabricantes de microcontroladores (Cortez, 2009). A continuación se hace referencia a algunos de ellos.

11 Conjunto de los componentes que conforman la parte material (física) de una computadora.

- Puertos entrada/salida: Los puertos de entrada/salida permiten al microcontrolador establecer una comunicación y controlar, a través de interfaces u otros dispositivos, a relés, interruptores, teclados, pantallas LCD (*Liquid Crystal Display*), sensores, etc. Pueden ser unidireccionales o bidireccionales.

  Las funciones especiales que estos realizan pueden ser mapeados y se configuran a través de los registros de funciones especiales. Estos puertos son la principal utilidad de los pines de un microprocesador.

- Puerto serie: Este periférico existe en los microcontroladores en el modo UART *(Universal Asynchronous Receiver Transmitter)* o USART (*Universal Synchronous Synchronous Receiver Transmitter)* dependiendo de si permiten o no el modo sincrónico de comunicación. El objetivo de este periférico es la comunicación con otro microcontrolador o con una computadora. Se implementa a través de las interfaces RS-232, RS-485, RS-422.

- Ethernet: La interfaz de comunicación Ethernet se utiliza mayormente para la comunicación del microcontrolador con otro o con una computadora. Es muy útil en la integración de los sistemas de control basado en microcontroladores para realizar tareas de monitoreo y control remoto de los sistemas a través de una HMI (*Human Machine Interface*).

- Memoria no volátil: La memoria del microcontrolador es un lugar donde se almacenan las instrucciones del programa y los datos que manipula. En los microcontroladores el programa normalmente se almacena en una memoria no volátil ROM *(Read Only Memory).* Es conocida generalmente como la memoria de programa pues contiene las instrucciones a ejecutar por el CPU. Para soportar esta función existen varios tipos de memorias no volátiles. Las más utilizadas actualmente para la programación de microcontroladores son: EEPROM, FLASH: La EEPROM *(Electrica Erasable Programmable Read Only Memory*) constituye una memoria de sólo lectura, programable y borrable eléctricamente. Tanto el grabado como el borrado, se realizan eléctricamente desde una computadora a través de una interfaz. El número de veces que pueden realizarse ambas operaciones en una memoria EEPROM es finito, por lo que no es aconsejable una reprogramación continua. Flash: Se trata de una memoria no volátil de bajo consumo energético, que se puede escribir y borrar. Funciona como una ROM, pero consume menos energía y es más pequeña. Es más rápida y de mayor densidad que la EEPROM. La alternativa flash está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado.

- Memoria volátil: El almacenamiento de datos y de variables se realiza en la memoria volátil RAM (*Random Access Memory*). Este tipo de memoria pierde la información almacenada cuando falta la energía (Cohen Manrique, 2005).

Las partes principales de un microcontrolador son:

1. Procesador
2. Memoria no volátil para contener el programa.
3. Memoria de lectura y escritura para guardar los datos.
4. Líneas de E/S para los controladores de periféricos:
    a) Comunicación paralelo
    b) Comunicación serie
    c) Diversas puertas de comunicación (bus I2C[12], USB, etc.)
5. Recursos auxiliares:
    a) Circuito de reloj
    b) Temporizadores
    c) Perro Guardián (*watchdog*)
    d) Conversores Analógico – Digital (AD) y Digital-Analógico (DA)
    e) Comparadores analógicos
    f) Protección ante fallos de la alimentación
    g) Estado de reposo o de bajo consumo

En años recientes el mundo de la electrónica y en especial el diseño de sistemas embebidos basados en microcontroladores, ha sido el escenario de una batalla entre dos grupos de diseñadores de estos sistemas, un grupo está formado por aquellos entusiastas en electrónica que usan los microcontroladores de la familia PIC del fabricante MICROCHIP y otro grupo lo forman las personas que prefieren diseñar sistemas basados en los microcontroladores de la familia AVR de ATMEL (academia.edu, 2015).

A continuación se presenta una tabla de comparación entre algunas características de los AVR y los PIC, haciendo referencia al Boletín “BATALLA DE MICROCONTROLADORES ¿AVR o PIC?” de Omar Otoniel Flores Cortez publicado en septiembre, 2009.

Microcontroladores más usados

[12] B*us* de comunicaciones en serie

Características	PIC	AVR
Descripción	Son una familia de microcontroladores de 8 bits, cuentan con un CPU RISC y memoria FLASH para el almacenamiento del Firmware.	Son una familia de microcontroladores de 8 bits cuentan con una CPU RISC y su memoria de programa viene implementada en FLASH.
Incluye periféricos	Puertos Digitales E/S, Conversores Análogo digital, Contadores, Temporizadores, Módulos PWM, entre otros. Disponen de un puerto para programación, usualmente un puerto serial. La programación se realiza a alto voltaje, mayor que 5Vdc, lo que hace necesario el uso de circuitos externos que realizan esta conversión de niveles y por lo tanto incrementan la complejidad del circuito programador.	Puertos Digitales E/S, Conversores Análogo digital, Contadores, Temporizadores, Módulos PWM, entre otros. Dispone de un periférico específico para la programación de su memoria, el puerto ISP, el cual es un puerto serial formado por 3 pines del microcontrolador, son estos 3 pines los que se conectan a un programador y este a un puerto del PC, para realizar el grabado o "quemado" del chip.
Simplicidad	Menos simple en comparación a los AVR.	Más simples en comparación a los PIC.
Potencia o consumo de Energía.	Alto consumo de energía.	Reducido consumo tanto en voltaje como en corriente, lo que permite el desarrollo de aplicaciones que funcionen a baterías.
Inclusión o no de Circuito Interno de reloj	Se debe de conectar un cristal externo que hará la función de oscilador de reloj.	Incluye un bloque oscilador formado por un arreglo RC para proveer la señal, también puede conectarse un

	cristal externo y no usar el oscilador interno

Tabla 1 Comparación entre los MCU AVR y PIC.

Hace unos años existía una diferencia nítida entre el concepto de microprocesador y de microcontrolador, en la actualidad, los sistemas fabricados en un solo chip han adquirido tal grado de complejidad que es muy difícil separar ambos conceptos (Miñarro, 2009).

La diferencia entre ellos es mejor vista a partir de la siguiente tabla resumen (Miñarro, 2009).

Comparación entre los microcontroladores y microprocesadores			
Dispositivo	**Definición**	**Características**	**Representación**
Microcontrolador	Es la implementación, dentro de un único chip, del microprocesador, la memoria y subsistemas de Entrada y Salida.	• Uso de CPUs con arquitecturas Harvard de 2, 4, 8, 16 o 32 bits. • Gran cantidad de periféricos de Entrada/Salida integrados. • Necesidad de poca memoria y generalmente no posibilidad de manejar memoria externa. • No empleo de sistema operativo. • Sistemas de desarrollo típicos basados en ANSIC. • Bajo coste del chip. • La placa (PCB) en la que se insertan suele ser sencilla en comparación con un formato PC, pues a	

		veces solo contiene este único chip.	
Microprocesador	Es la implementación en forma de circuito integrado (IC) de una Unidad Central de Proceso (CPU) junto con los buses de interconexión (el bus de control, el bus de direcciones y el bus de datos).	• Uso de CPUs con arquitecturas de 8, 16, 32 y 64 bits. • Gran cantidad de periféricos de E/S integrados. • Incluyen memoria dentro del chip y necesidad de manejo de gran cantidad de memoria externa. • Empleo de sistema operativo. • Sistemas de desarrollo basados en leguajes de alto nivel tal como el C. • Amplia gama de chips y gran variación en el coste. • Se insertan en placas base, de tamaño cada vez mas compacto, que incorporan una amplia gama de chips dedicados a las distintas funciones previstas.	

Tabla 2 Comparación entre microcontrolador y microprocesador

1.5 Tecnologías y herramientas utilizadas

Con el objetivo de realizar el firmware y la aplicación de monitoreo, se realiza un estudio sobre las diferentes herramientas y tecnologías a utilizar, definidas por el departamento de aplicaciones del CEDIN. A continuación se realiza una breve descripción de las mismas.

1.5.1 Metodología de Desarrollo

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo han evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo (Rayma Krishna, 2011).

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos (*software*). Se clasifican en dos tipos, las tradicionales, centradas específicamente en el control riguroso de los procesos de desarrollo de un software y las metodologías ágiles, quedan mayor valor a los usuarios, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Entre las metodologías de desarrollo ágiles más conocidas y usadas se encuentra la Programación Extrema (XP, siglas en inglés) que básicamente plantea que se trabaje directamente con el cliente haciendo pequeñas iteraciones y como resultado mini entregas cada dos semanas, donde no existe más documentación que el propio código (Solias, 2003).

XP es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Esta metodología se basa en la idea de que existen cuatro variables que guían el desarrollo de sistemas: Costo, Tiempo, Calidad y Alcance. La manera de encarar los desarrollos avalados por este modelo de desarrollo es permitir a las fuerzas externas (gerencia, clientes) manejar hasta tres de estas variables, quedando el control de la restante en manos del equipo de desarrollo (Informáticos, 2013). Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software (Rodríguez, 2013). Es requisito para XP definir un estándar en el tipo de codificación, esto hace que los programadores tengan definido ya el estilo de programación y no que cada uno programe a su estilo. Las pruebas

en cada iteración son más que importante; de eso se trata este paradigma de programación, corregir mientras se programa. De esta forma se van cubriendo todos los baches que cada versión padezca. El código no es de nadie, todo el equipo puede manipular el código que existe, de esta forma cada pareja puede mejorar cada sección de código que utiliza, esto requiere de una prueba del mismo y la re-implementación en el sistema general. Cada dos semanas se entrega una versión al cliente, que lo verifica, realiza la retroalimentación y se continúa el desarrollo; este ciclo continúa hasta que el sistema cumpla con las expectativas del cliente, acto que concluirá el proyecto (Barr, 2006)

Para seleccionar la metodología que se ajuste a las características del presente trabajo, se tuvo en cuenta lo planteado por Pressman en su 7ma edición: “Ingeniería de Software. Un enfoque práctico”, donde expresa que independientemente de la metodología que se decida utilizar, en el desarrollo de software, se deben tener en cuenta dos factores fundamentales: los técnicos y los medioambientales, dentro de estos últimos inciden directamente las personas. Además se tuvo presente que la aplicación de esta metodología al desarrollo de sistemas embebidos se debe a que este proceso tiene entre sus características principales que la implementación de software y la del hardware se realizan de forma paralela. Esto implica que muchas veces no se cuente con la solución de *hard* donde se debe empotrar el *soft* impidiéndole al equipo de programación comenzar el desarrollo desde el inicio del proyecto (Debian, 2015). Otras razones por las cuales se decide utilizar XP es porque el presente trabajo se realiza a corto plazo, se tiene solo 2 integrantes lo que facilita que la programación sea par, el plazo de entrega es corto, se tiene una constante comunicación con el cliente y se emplea la reutilización del código.Hardware a utilizar

1.5.2 Interfaces para la comunicación serial

Los MCU ATMega8 incluyen 3 recursos para comunicarse de manera serial con dispositivos externos, empleando interfaces estándares. Los recursos son: Un Transmisor Receptor Universal Sincrónico Asincrónico (USART), una Interfaz para Periféricos Seriales (SPI, siglas en inglés) y una Interfaz Serial de Dos Hilos (TWI, siglas en inglés). Las interfaces estándares empleada a lo largo de la investigación son USART y SPI.

USART incluye recursos independientes para transmisión y recepción, lo que es posible enviar y recibir datos en forma simultánea, se utiliza en el presente trabajo para establecer la comunicación entre el sistema embebido y la aplicación de monitoreo. Mientras que la SPI es usada para transferir paquetes de información de 8bits entre circuitos integrados. Esta interfaz se emplea debido a que se necesita más periféricos conectados y un solo microcontrolador no cuenta con esa determinada cantidad de pines para permitir la conexión, por lo que se tiene un MCU configurado como esclavo y otro como maestro.

El sistema embebido se comunica de manera serial con la computadora a través de la interfaz RS-232, se hace uso de los pines que portan datos: RXD y TXD. Dicha interfaz permite el empleo de

Modbus, uno de los protocolos de comunicaciones más populares en sistemas de automatización, telecontrol, monitorización y control; es un protocolo de transmisión para sistemas de control y supervisión de procesos (SCADA[13], siglas en inglés) con control centralizado, puede comunicarse con una o varias Estaciones Remotas (RTU[14], siglas en inglés) con la finalidad de obtener datos de campo para la supervisión y control de un proceso. Con el uso de la interfaz RS-232 se transmite información entre diferentes dispositivos electrónicos conectados al mismo bus, donde existe un solo dispositivo maestro y varios esclavos conectados.

Modbus específicamente se usa para monitorear desde la aplicación (cliente) los datos que envía el sistema embebido (servidor), para recolectar los valores captados por los sensores.

1.5.3 Bibliotecas de soporte para el desarrollo del firmware

- LCD: Este fichero permite el trabajo con el display 16x2, la configuración de los pines de datos y de control; posibilita leer la hoja de datos del display.
- USART: Es usado para la comunicación serial, configura el modo de transmisión y recepción serie del microcontrolador.
- keypad: Este fichero posibilita la configuración del puerto al que está conectado el teclado matricial.

1.5.4 Bibliotecas de soporte para el desarrollo de la aplicación de monitoreo

- QT: Es empleada esta biblioteca debido que con ella se puede desarrollar aplicaciones utilizando el lenguaje C++, las cuales pueden ser con o sin interfaz gráfica, puede funcionar en varias plataformas: Microsoft Windows, Unix/X11, Mac OS X, se distribuye bajo ediciones comerciales o bien Open Source, incluye una extensa librería de clases C++ y utilidades para construir las aplicaciones de forma rápida y sencilla.
- Libmodbus: Es una biblioteca de software libre para enviar / recibir datos de acuerdo con el protocolo Modbus. Esta biblioteca está escrita en C y soporta RTU (de serie) y comunicaciones (Ethernet). Se emplea para enviar acciones al microcontrolador y a la aplicación, para gestionar los sensores a conectar al microcontrolador AVR y para generar las alertas en la aplicación de monitoreo.

13 Es un software para ordenadores que permite controlar y supervisar procesos industriales a distancia.

14 Define a un dispositivo basado en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto donde se procese.

1.5.5 Herramientas

- AVRDUDE v6.1: Se utiliza para descargar, subir y manipular el contenido de la ROM y EEPROM de microcontroladores AVR utilizando la técnica de programación ISP (*in-system programming)*.
- AVR-GCC v4.8.1: Software libre bajo la licencia GNU, disponible para Windows, Linux y MacOs. Se emplea para compilar el firmware embebido en el microcontrolador.
- Visual Paradigm v8.0: Herramienta CASE que da soporte al modelado visual con UML. Se empleó para generar artefactos que apoyen la propuesta de solución.

1.5.6 Lenguajes de programación

Para el desarrollo del firmware se emplea el lenguaje de programación C debido a que "se ha convertido en el lenguaje de los programadores de sistemas embebidos" (Alcoy, 2015). Algunas de las ventajas que posee relacionadas directamente con los sistemas embebidos son (Alcoy, 2015)

- Es pequeño y bastante sencillo de aprender, los compiladores están disponibles para casi todos los procesadores en uso hoy en día, lo que garantiza portabilidad.
- Tiene la ventaja de ser independiente del procesador, lo que permite a los programadores concentrarse en algoritmos y aplicaciones en lugar de los detalles particulares de la arquitectura del procesador.
- Posibilita la producción de código compacto, eficaz para casi todos los procesadores.

Para el desarrollo de la aplicación se decide emplear el lenguaje de programación C++ por ser un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. C++ mantiene las ventajas de C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Se usa C++ también por el Entorno de desarrollo integrado que se utiliza en el Departamento de Aplicaciones.

1.5.7 Lenguaje de modelado

- UML: es un lenguaje para visualizar, especificar, construir y documentar los elementos que componen un sistema basado en la programación orientada a objetos. Se emplea para modelar algunos artefactos como: el diagrama de estados y las tarjetas CRC.

1.5.8 Entorno de desarrollo integrado (IDE, siglas en inglés)

- QtCreator: ofrece una multiplataforma, entorno de desarrollo integrado (IDE) para los desarrolladores de aplicaciones para crear aplicaciones para múltiples plataformas de escritorio y dispositivos móviles, como Android y iOS. Está disponible para Linux, OS X y sistemas operativos de Windows.

1.5.9 Plataforma

Para el desarrollo del sistema se utiliza la distribución Debian de GNU/Linux, debido a que brinda a través de su repositorio, las herramientas necesarias, cuenta con un sistema de seguimiento de errores público, es estable, rápido y ligero (Sanchez, 2014).

1.5.10 Conclusiones

En este capítulo se analizaron los conceptos asociados al problema, detallando características de lo sistemas domóticos y todo lo referente a ellos. Se determinó el empleo de microcontroladores AVR en comparación a otros dispositivos usados en los sistemas domóticos debido a que incluyen una CPU, temporizadores, interrupciones y otros recursos útiles para el desarrollo de aplicaciones. Se plantean las herramientas y metodologías seleccionadas para la realización de la aplicación de monitoreo y el firmware. Se elige la metodología de desarrollo XP debido a las facilidades que esta ofrece para la gestión de sistemas embebidos. Para el desarrollo del firmware y de la aplicación de monitoreo se emplea C y C++ como lenguajes de programación, respectivamente. Se programa en C++ con la ayuda del IDE QtCreator, el mismo proporciona muchas bibliotecas del propio C++. Para la realización de diagramas y modelos para un mayor entendimiento se usa el lenguaje UML. Se emplea libmodbus para hacer uso del protocolo de comunicación Modbus a través de la interfaz de capa física configurada en RS-232 para la transmisión de datos entre la aplicación de monitoreo y el sistema embebido. Se decide trabajar con la plataforma Debian debido a sus múltiples ventajas por ser un Sistema Operativo Libre, con este también se emplea el compilador AVR-GCC y AVRdude para programar el MCU.

CAPITULO 2: Características del sistema

En el presente capítulo se exponen los elementos que forman parte de las características del sistema como: los requerimientos y la propuesta del sistema, las características del hardware. Se platean y explican los artefactos generados a través de la aplicación de la metodología XP tal como las historias de usuario y el plan de entrega de cada una de ellas.

2.1 Propuesta del sistema

El sistema propuesto está dirigido para un local como por ejemplo: un almacén, una oficina, un cuarto de corrientes débiles, un nodo central de servidores, un laboratorio, entre otros que cumplan con las siguientes características:

1. Superficie no mayor de 100 metros cuadrado.
2. Totalmente cerrado.
3. Climatizado.
4. Aislado de la entidad a la que pertenece.
5. Puede contar hasta 6 sensores analógicos.

La solución propuesta consta de dos partes, el sistema embebido en dos microcontroladores y la aplicación de monitoreo, estos se comunican a través del protocolo Modbus haciendo uso de la interfaz RS-232 para la transferencia de información. La aplicación trabaja como cliente mientras que el sistema embebido como servidor. El diálogo que se establece entre ellos a través de Modbus es: “Encuesta-Respuesta” esta operación permite que el maestro interrogue a un esclavo con una única dirección en la red y espera por la respuesta de este último (Larman). La aplicación debe permitir que el usuario desde la entidad conozca el estado del local a través de notificaciones que serán generadas desde el sistema embebido a través de Modbus y en caso de algún evento que viole la seguridad el usuario sabe que decisión tomar, además puede controlar los sensores que desee (6 sensores analógicos, como máximo) tener conectados al microcontrolador y gestionar los usuarios que tienen permiso para trabajar con la aplicación.

El sistema embebido debe permitir que el usuario introduzca un PIN para acceder al local, para esto se cuenta con un teclado matricial y con un display. El sistema propuesto solo monitoreará las variables incidentes, de acuerdo a los sensores que tenga instalados. La aplicación permitirá que el administrador conozca el estado del local para así tomar las medidas necesarias mediante las alertas generadas.

2.2 Requerimientos del sistema

1. Comunicar el sistema embebido con la computadora.
2. Controlar el acceso al local.

3. Controlar el acceso a la aplicación de monitoreo.
4. Gestionar usuario.
5. Controlar Sensores.
6. Gestionar Notificaciones.
7. Mostrar alertas.
8. Monitorear las variables incidentes de un local.
9. Registrar las acciones que ocurren en la aplicación de monitoreo.
10. Mostrar las acciones ejecutadas.

2.2.1 Características del Hardware

Los MCU AVR ATMega8 tienen las siguientes características principales:

- Arquitectura RISC.
- Arquitectura Harvard.
- Ejecución de una instrucción en 1 ciclo de reloj.
- 1 MIPS por 16 MHz.
- Arquitectura del tipo Registro-Registro, con 32 Registros de propósito general de 8 bits, habilitados para un acceso rápido.
- Memorias: FLASH[15], EEPROM[16] y SRAM[17].
- Memoria de código: 8Kbyte de memoria FLASH.
- Memoria de datos: 1Kbyte de SRAM y 512 Mbyte de EEPROM.
- Terminales para entrada/salida: 23.
- Temporizadores: 2 de 8bits y 1 de 16 bits.
- Canales PWM: 3.
- Fuentes de interrupción: 19.
- Interrupciones externas: 2.

15 Memoria en la que se almacenan los datos que no cambian durante la ejecución normal de un programa.

16 Memoria en la que se almacenan las variables en las que se requiera conservar su contenido, en ausencia de energía.

17 Memoria en la que se almacena los datos que van hacer leídos y escritos continuamente.

- Operación en un rango de voltaje de 2.7 a 5.5 V.
- 6 registros pueden ser usados como apuntadores de 16 bits para direccionamiento indirecto en el espacio de datos. Para ello, estos registros se denominan como X, Y y Z.
- El registro Z también puede usarse como apuntador a la memoria de programa.
- Arquitectura Optimizada para ejecutar código C compilado.
- Tecnología de memoria sobresaliente, 3 tecnologías diferentes en el mismo.
- Canales de Conversión Analógico/Digital: 8 de 10 bits.
- Reloj de tiempo real.
- Interfaz SPI[18] Maestro/Esclavo.
- Transmisor/Receptor Universal Síncrono/Asíncrono (USART).
- Interfaz serial de dos hilos.
- Oscilador interno configurable.
- *Watchdogtimer*[19].
- Son dispositivos de alto rendimiento.

Microcontrolador AVR ATMega8		
No.	Posee	Emplea
1	Una comunicación por SPI (MOSI, MISO, SCK, SS)	4 pines
2	Una conexión con la pantalla (display LCD 16x2)	7 pines
3	Una conexión con un teclado matricial 4x4	8 pines
4	Sensores analógicos (ADC0, ADC1, ADC2, ADC3, ADC4, ADC5)	6 pines
5	Una Interfaz de comunicación RS-232	2 pines para la comunicación serial (RXD y TXD)

18 Protocolo estándar de comunicaciones, usado para transferir paquetes de información de 8 bits entre circuitos integrados.

19 Temporizador, se compone de un registro de n-bits.

Tabla 3 Características del Hardware

Atendiendo que se tiene dos MCU por las razones antes expuestas (Véase Interfaces para la comunicación serial), el configurado como esclavo cuenta con el teclado, el *display*, los pines para la comunicación por SPI, para la comunicación con el *display* y para la conexión del teclado mientras que el configurado como maestro tiene los restantes recursos (puerto serial y sensores).

A continuación se muestra en la figura un circuito de control con los dos MCU y los recursos conectados a ellos.

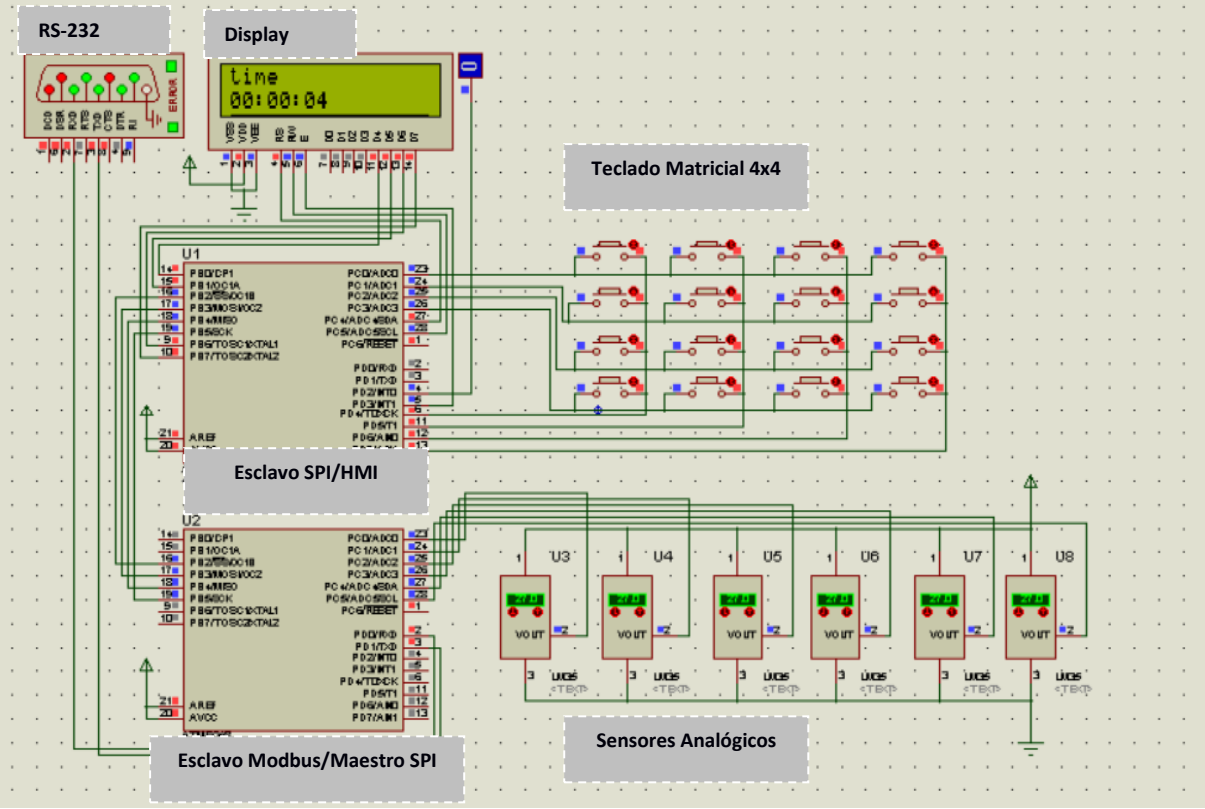

Figura 1 Circuito de Control

2.3 Requisitos no funcionales (RNF)

2.3.1 RNF de rendimiento

- El sistema debe garantizar una respuesta rápida en la comunicación entre el microcontrolador y la computadora.

2.3.2 RFN de espacio

- El binario resultante de la compilación debe de ajustarse a la cantidad de memoria FLASH disponible.
- El consumo de memoria RAM debe ajustarse a cantidad de memoria disponible.

2.3.3 RFN de fiabilidad

- En caso de algún fallo en la comunicación con el MCU o con algún dispositivo, el sistema debe efectuar un número determinado de intentos para restablecer dicha comunicación e informar del fallo.

2.3.. RFN de implementación

- El código resultante de la implementación debe de estar regido por un Estándar de Codificación, permitiendo ser legible con el objetivo de facilitar las mejores posteriores del firmware. El Estándar de Código debe de establecerse y describirse en el presente documento.

2.3.5 RFN de usabilidad

- Es necesaria una preparación previa para operar con el sistema.

2.3.6 RFN de Hardware del cliente

- Computadora de 512 MB de RAM o superior.
- Espacio en disco de 512MB.

2.4 Personas relacionadas con el sistema

Las personas relacionadas con el sistema son: el administrador y los técnicos. El administrador se encarga de tomar decisiones en caso de incidentes y los técnicos están al tanto del estado del local. La siguiente tabla muestra dicha información:

Usuarios	Responsabilidad
Administrador	Encargado de gestionar a los usuarios, las notificaciones, de controlar los sensores, el acceso al local y a la aplicación de monitoreo. Además de monitorear las variables incidentes del local, ver las alertas generadas y los registros de acciones ejecutadas en la aplicación de monitoreo.
Técnico	Encargado de gestionar las notificaciones, de controlar el acceso al local y a la aplicación de monitoreo. Además de monitorear las variables incidentes del local y ver las alertas generadas.

Tabla 4 Personas relacionadas con el sistema

2.5. Planeación

La primera fase definida por la metodología XP según lo planteado por Pressman en su 6ta Edición: “Ingeniería de Software. Un enfoque práctico” es la Planeación. Comienza por la creación de una serie de historias, llamadas Historias de Usuario (HU) las cuales son definidas por el cliente, asigna

un valor o prioridad a cada HU respecto a la función o característica que tiene en el negocio. Los miembros del equipo XP evalúan cada historia y le asignan puntos de estimación, el cual se mide en semanas de desarrollo. En caso de necesitarse HU nuevas, estas pueden escribirse en cualquier momento.

Los clientes y el equipo de XP trabajan juntos para decidir cómo agrupar las historias hacia el próximo lanzamiento para que el equipo las desarrolle (Labaut, 2012). Una vez establecido el compromiso básico, para un lanzamiento el equipo puede ordenar las HU que se desarrollarán de tres maneras: todas las historias serán implementadas de manera inmediata (dentro de pocas semanas), las historias con valor más alto se moverán en el programa y se implementarán al principio, las historias más rigurosas se moverán dentro del programa y se implementarán al principio según Pressman; en el presente trabajo se decide implementar todas las historias de un modo inmediato (dentro de pocas semanas).

2.5.1 Historia de Usuario

Para el presente trabajo las HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. Su tratamiento es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más específicas o generales, añadirse nuevas o ser modificadas, con esto se comprueba la flexibilidad de la metodología. Como resultado se definió las HU siguientes, las cuales se derivan en una lista de tareas de programación. Si bien las HU tienen como punto de vista lo que el usuario necesita, las tareas de programación definen que se debe hacer para satisfacer esa necesidad.

Historia de Usuario	
Número: 1	**Nombre historia: Comunicar el SE con la computadora.**
Usuario:	**Iteración Asignada:** 1
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación de monitoreo debe permitir seleccionar la opción para conectarse al sistema embebido y de esta forma establecer la comunicación introduciendo los siguientes parámetros: Puerto, Tasa de Baudio, Paridad, Cantidad de bits de datos y Bit de parada y el ID del MCU esclavo.	

Observaciones:

Tabla 5 HU #1: Comunicar el SE con la computadora.

Historia de Usuario	
Número: 2	**Nombre historia: Controlar el acceso al local.**
Usuario: Administrador, Técnico.	**Iteración Asignada:** 1
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: El sistema embebido debe permitir el control de acceso al local verificando el PIN entrado por el usuario. Antes de entrar al local, la pantalla solo muestra la hora. Además debe existir una opción en la aplicación de monitoreo para cambiar el PIN del acceso al local.	
Observaciones: Tanto el administrador como el técnico tienen permitido el acceso al local.	

Tabla 6 HU #2: Controlar el acceso al local.

Historia de Usuario	
Número: 3	**Nombre historia: Controlar el acceso a la aplicación de monitoreo.**
Usuario: Administrador, Técnico.	**Iteración Asignada:** 1
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación debe tomar como datos de entrada: el nombre del usuario y la contraseña, verificar los datos entrados por el usuario y permitirle el empleo de la misma.	
Observaciones: Tanto el Administrador como el Técnico pueden acceder a la aplicación de monitoreo.	

Tabla 7 HU #3: Controlar el acceso a la aplicación de monitoreo.

Historia de Usuario	
Número : 4	**Nombre historia: Gestionar usuario.**
Usuario: Administrador	**Iteración Asignada:** 2

Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación de monitoreo debe permitir la gestión de usuarios, de esta forma puede insertar, modificar o eliminar a uno de ellos. Para la inserción de un nuevo usuario es necesario el nombre, el usuario y la contraseña del mismo. La aplicación debe mostrar una lista con los usuarios registrados y permitir seleccionar alguno de ellos para modificar los datos o para eliminarlos.	
Observaciones:	

Tabla 8 HU #4: Gestionar usuario.

Historia de Usuario	
Número: 5	**Nombre historia: Controlar Sensores.**
Usuario:	**Iteración Asignada:**2
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 4	
Descripción: La aplicación debe permitir que se muestre un listado con los sensores conectados y al seleccionar alguno de ellos se modifique el nombre y el tipo de sensor y se guarde los valores insertados en la base de datos.	
Observaciones:	

Tabla 9 HU #5: Control sensores.

Historia de Usuario	
Número: 6	**Nombre historia: Gestionar notificaciones.**
Usuario: Administrador, Técnico.	**Iteración Asignada:**2
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Medio
Estimación de esfuerzo: 3	
Descripción: La aplicación debe permitir que se muestre todas las notificaciones registradas y las opciones de crearlas, modificarlas o eliminarlas. Para crear una notificación se necesita los siguientes parámetros: objetivo, condición y limite, igual que para modificarlas; mientras que	

para eliminar solo se necesita seleccionar una de las notificaciones registradas.
Observaciones:

Tabla 10 HU #6: Gestionar notificaciones.

Historia de Usuario	
Número: 7	**Nombre historia: Mostrar alertas**
Usuario: Administrador, Técnico.	**Iteración Asignada:**3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Medio
Estimación de esfuerzo: 2	
Descripción: La aplicación debe permitir que se muestren las alertas asociadas a las notificaciones configuradas.	
Observaciones: Tanto el administrador como el técnico pueden ver las alertas generadas.	

Tabla 11 HU #7: Mostrar alertas.

Historia de Usuario	
Número: 8	**Nombre historia: Monitorear las variables incidentes de un local.**
Usuario: Administrador, Técnico.	**Iteración Asignada:** 3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación muestra la información captada por los sensores y no se guarda en la base datos.	
Observaciones:	

Tabla 12 HU #7: Monitorear las variables incidentes de un local.

Historia de Usuario	
Número: 9	**Nombre historia: Registrar las acciones que ocurren en la aplicación de monitoreo.**

Usuario:	**Iteración Asignada:** 3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación debe permitir que las acciones realizadas como: Gestionar usuario, Modificar un sensores, entre otras sean guardadas en la base datos.	
Observaciones:	

Tabla 13 HU #9: Registrar las acciones que ocurren en la aplicación de monitoreo en la base de datos.

Historia de Usuario	
Número: 10	**Nombre historia: Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.**
Usuario:	**Iteración Asignada:** 3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: La aplicación debe permitir que se muestre un listado con los registros de las acciones ejecutadas.	
Observaciones:	

Tabla 14 HU #10:Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.

2.5.2 Plan de duración de las iteraciones

Este plan consiste en mostrar las HU en el orden en que se desarrollaron, así como la iteración y la duración estimada en semanas y días.

Iteración	Orden de la HU a implementar	Duración (semanas, días)
1	HU # 1: Comunicar el SE con la computadora. HU # 2: Controlar el acceso al local.	3 semanas

	HU # 3: Controlar el acceso a la aplicación de monitoreo.	
2	HU # 4: Gestionar Usuario. HU # 5: Controlar Sensores. HU # 6: Gestionar Notificaciones.	3 semanas
3	HU # 7: Mostrar alertas. HU # 8: Monitorear las variables incidentes de un local. HU # 9: Registrar las acciones que ocurren en la aplicación de monitoreo. HU # 10: Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.	3 semanas

Tabla 15 Plan de duración de las iteraciones.

2.5.3 Plan de entregas

El plan de entrega (PE) específica que HU serán implementadas en cada entrega del producto y sus prioridades; de este modo permite conocer con exactitud qué HU serán desarrolladas en la próxima liberación. El PE debe ser negociado y elaborado con el cliente y el equipo de desarrollo durante las reuniones de planificación de entregas.

Cada HU se dividió en una iteración en correspondencia con las características de su implementación. La siguiente tabla muestra 10 entregas y el orden de las mismas, así como la fecha en que culmina cada iteración.

Plan de Entregas	
Sistema de monitoreo basado en microcontroladores AVR.	
Fecha de Reunión de Planificación:	27/01/2015
Nombre del Documentador:	Antuanett Barrios Frometa, Yordan C. Perez Attanaff.
Entrega No.	10
Historias de usuario a implementar en la entrega.	

No. HU	Título	Prioridad	Fecha de entrega	Liberación en la que se incluirá
1	Comunicar el SE con la computadora.	Alta	03/03/2015	1
2	Controlar el acceso al local.	Alta	03/03/2015	1
3	Controlar el acceso a la aplicación de monitoreo.	Alta	03/03/2015	1
4	Gestionar Usuario.	Alta	25/03/2015	2
5	Controlar Sensores.	Alta	25/03/2015	2
6	Gestionar Notificaciones.	Alta	25/03/2015	2
7	Mostrar alertas.	Alta	20/04/2015	3
8	Monitorear las variables incidentes de un local.	Alta	20/04/2015	3
9	Registrar las acciones que ocurren en la aplicación de monitoreo.	Alta	20/04/2015	3
10	Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.	Alta	20/04/2015	3

Tabla 16 Plan de entregas.

2.6 Conclusiones

En este capítulo se describió a que determinado local esta destinado el sistema. Se planteó como propuesta de solución una aplicación de monitoreo y un sistema embebido. Se especificaron: los técnicos y el administrador como las únicas personas con acceso al sistema, además se determinaron los requerimientos funcionales y los no funcionales, las características del MCU AVR ATMega8. Se determinó llevar a cabo las historias de usuarios en ordenen de modo inmediato (dentro de pocas semanas). Se confeccionó el plan de entrega y el plan de duración de las iteraciones donde se muestra en que iteración se irán liberando las HU y cuando se entregarán para lograr la satisfacción del cliente.

CAPÍTULO 3: Diseño e Implementación

En este capítulo se presenta y se describe la arquitectura, los patrones de diseño y los estándares de codificación utilizados. Se diseñan las clases necesarias para la realización de sistema. Se realizan las tarjetas CRC (Clase – Responsabilidad – Colaborador) y las tareas de ingeniería, definidas para desglosar las actividades que se realizan en cada HU.

3.1 Arquitectura del sistema

Crain Larman plantea en el libro: “UML y Patrones”, 2da Edición que una arquitectura es un conjunto de decisiones significativas sobre la organización del software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios y el estilo de arquitectura que guía esta organización, estos elementos y sus interfaces, sus colaboraciones y su composición.

3.1.1 Patrón arquitectónico

Consiste en una estructura que plantea la forma en que se van a organizar los componentes de un sistema de software y las responsabilidades de cada uno. A continuación se muestran los patrones arquitectónicos que interviene en el proceso de desarrollo de la aplicación y del firmware.

3.1.1.1 Arquitectura de la aplicación de monitoreo

Para el desarrollo de la aplicación de monitoreo se decide el uso de la arquitectura por capas debido a: la organización de la estructura lógica en capas separadas de responsabilidades distintas y relacionadas, la separación clara y cohesiva de intereses, la colaboración y el acoplamiento desde las capas más altas hacia las más bajas; evitando el acoplamiento de las capas más bajas a las más altas. La arquitectura está compuesta por 4 capas: Presentación, Monitoreo, Servicios Técnicos y Comunicación; las mismas permiten contar con las funcionalidades necesarias para el desarrollo de la aplicación. La primera capa es el encargada de mostrar todos los componentes visuales de la aplicación, mientras que la segunda gestiona las peticiones de la capa de presentación, se comunica tanto con Presentación como con Servicios Técnicos. Esta última está conformada por los paquetes de Persistencia (DatabaseAdapter) y de Seguridad (Security) y se comunica con Monitoreo y con Comunicación. La última capa permite establecer las conexiones entre el sistema embebido y la aplicación de monitoreo, además posibilita hacer uso del protocolo Modbus RTU.

Para representar la arquitectura se modela en diagramas de paquetes, estos permiten centrarse en ilustrar el acoplamiento paquete-paquete (Anisbel Clara Hernández, 2014) con la ayuda de las líneas de dependencia. A continuación se muestra la relación de las capas y de los paquetes, un

ejemplo es la relación que tienen los paquetes NewNotificacion y DataProvider (clases de la aplicación de monitoreo) pertenecientes a la capa Monitoreo, con DatabaseAdapter de la capa Servicios técnicos y con la capa de Presentación.

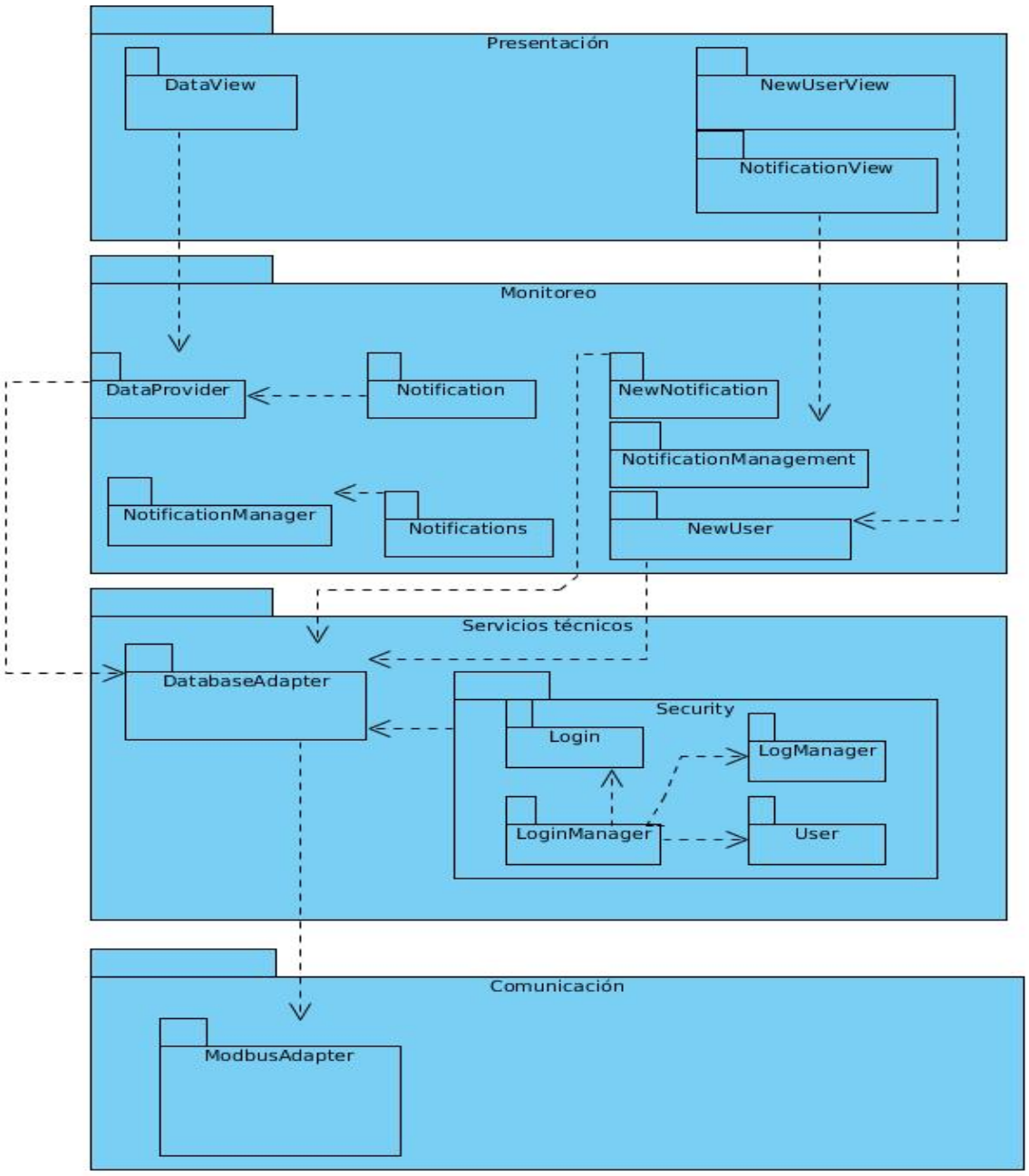

Figura 2 Arquitectura de la aplicación de monitoreo

3.1.1.2 Arquitectura del Sistema Embebido

La arquitectura propuesta para la realización del sistema embebido es Cliente-Servidor, este constituye un patrón para *software* distribuido, el cual define dos tipos de componentes principales: el Esclavo o Maestro y el Servidor o Cliente. Igualmente se define el modo de interacción entre

ellos: la comunicación basada en un protocolo a través del mecanismo de encuesta-respuesta (Joskowicz, 2008). El uso de esta arquitectura se atribuye a la existencia de dos microcontroladores que se comunican a través de la interfaz SPI, comportándose uno como esclavo y otro como maestro.

Esclavo

- Constituye el componente activo, inicia la comunicación con el servidor enviando las peticiones o encuestas.
- Debe de obtener una respuesta a partir de la petición.

Maestro

- Constituye el componente pasivo, espera por las peticiones de los esclavos.
- Procesa las peticiones al recibirlas y ofrece una respuesta.

3.1.1.3 Arquitectura del Sistema

El protocolo Modbus se basa en la arquitectura cliente-servidor, su empleo permite establecer una comunicación entre el sistema embebido y la aplicación de monitoreo. Este último se comporta como cliente mientras que el sistema embebido como servidor. Visto de una manera individual poseen arquitecturas diferentes pero cuando se trata como sistema se mantiene la misma arquitectura del protocolo Modbus.

3.2 Patrones de diseño

Los patrones de diseño software son los que nos permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos (Rumbaugh, 1998). Para esta investigación se hizo uso de los patrones que a continuación se presentan:

3.2.1 Patrones Grasp (General Responsibility Assignment Software Pattern)

Patrones generales de software para asignar responsabilidades, es el acrónimo de *General Reponsability Assignament Software Pattern*. Se consideran una serie de „‟Buenas Prácticas„‟ para el diseño de software (academia.edu, 2015). El objetivo fundamental de su utilización es asignar responsabilidades a las clases dependientes.

Las descripciones de cada patrón fueron tomadas del libro UML y Patrones de Craig Larman.

Experto: Se emplea para asignar una responsabilidad a la clase que tiene la información necesaria para realizar la responsabilidad. Se evidencia en las clases: SettingsManager, ModbusAdapter.

Creador: Se usa para crear clases descendientes o instancias a partir de una clase con alta jerarquía. Se evidencia en las clases: NotificationManagement, NotificationManager, Notifications.

Bajo acoplamiento: Se emplea con el objetivo de que las clases que dependen entre sí, estén lo menos ligadas posible, de forma tal que se pueda entender la mayor parte de sus estructuras y sus funciones. En caso de que se produzca una modificación en alguna de ellas, tuviese la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Este patrón se refleja en la organización de las clases NotificationManagement y NotificationManager.

Alta Cohesión: Permite asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. Dichas funcionalidades están repartidas en cada una de las clases dependientes, tales como: DataProvider, DataView, ModbusAdapter.

Polimorfismo: Se emplea cuando las alternativas o comportamientos relacionados varían según el tipo (clase); el objetivo es asignarle la responsabilidad del comportamiento utilizando operaciones polimórficas a los tipos para los que varía el comportamiento. Este patrón está presente en la clase databaseAdapter.

3.2.2 Patrones Gof (*Gang of Four*)

Singleton: Permitió garantizar que una clase sólo tenga una única instancia, proporcionando un punto de acceso global a la misma. Este patrón se ve reflejado en las clases: DatabaseAdapter, NotificationManager.

3.3 Diagrama de estados

Los diagramas de estados, como se muestran a continuación representan los eventos y estados interesantes de un objeto, y el comportamiento de un objeto como reacción a un evento. Las transiciones se representan con flechas, etiquetadas con sus eventos. Los estados se representan en rectángulos de esquinas redondeadas (Rumbaugh, 1998). En los sistemas embebidos suelen ser modelado mediante una máquina de estados. Según (Uruguay, 2015)una máquina de estados es un comportamiento que especifica las secuencias de estados por las que pasa un objeto durante su vida, junto con las respuestas a determinados eventos.

El diagrama de estados de UML proporciona una representación gráfica para los tres elementos que se definen a continuación (Uruguay, 2015):

- Estado: Condición o situación en la vida de un objeto durante la cual satisface alguna condición, se realiza alguna actividad o se espera algún evento.

- Evento: es la especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y en el espacio. En el contexto de las máquinas de estados, un evento es la aparición de un estímulo que puede disparar una transición de estados.
- Transición: Es una relación entre dos estados que indica que un objeto que esté en el primer estado realizará ciertas acciones y entrará en el segundo estado cuando ocurra un evento especificado y se satisfagan unas condiciones especificadas.

A continuación se muestran el diagrama de estados perteneciente al Control de acceso al local del sistema embebido.

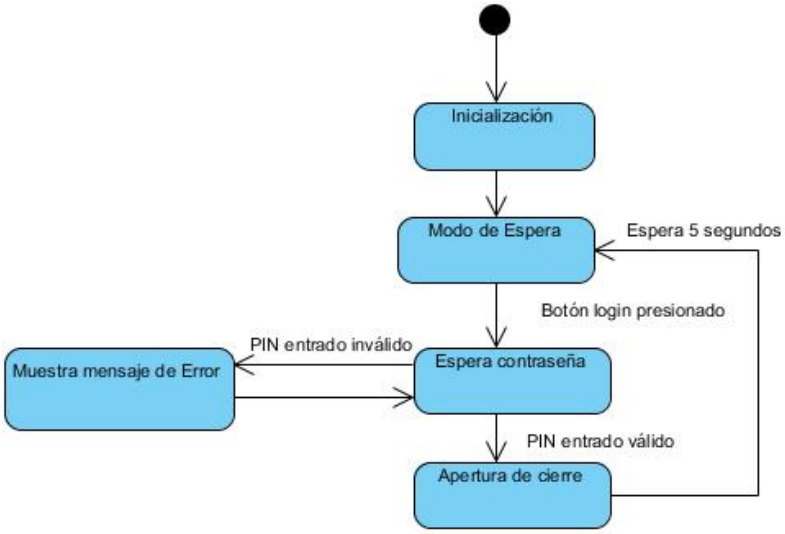

Figura 3 DE Control de acceso al local

Para el sistema embebido se han diseñado 5 estados: Inicialización, Modo de espera, Espera contraseña, Apertura de cierre y Muestra mensaje de error. A continuación se describirán cada uno de estos estados.

Elemento	Clasificación	Descripción
Inicialización	Estado	El *firmware* inicia su ejecución. Es el estado donde se inicializan las interfaces de hardware, las estructuras de datos internas así como las variables globales.

Modo de Espera	Estado	Se espera por que el usuario ejecute una acción.
Botón *login* presionado	Evento	Si el botón *login* es presionado, pide la contraseña a través de la pantalla.
Espera contraseña	Estado	Se espera por que el usuario introduzca la contraseña.
PIN entrado inválido	Evento	Se verifica que el PIN entrado no sea inválido.
PIN entrado válido	Evento	Se verifica que el PIN entrado sea válido.
Apertura de cierre	Estado	Si el PIN entrado es válido, se abren las aperturas de cierre.
Espera 5 segundos	Evento	Espera 5 segundos antes de volver al Modo de espera.
Muestra mensaje de error	Estado	Si el PIN entrado es inválido se regresa al evento: “Espera contraseña”.

Tabla 17 Descripción del diagrama de estados

3.4 Diagrama de Despliegue

Los diagramas de despliegue muestran cómo se configuran las instancias de los componentes y los procesos para la ejecución *run-time*(tiempo en ejecución)en las instancias de los nodos de proceso (algo con memoria y servicios de proceso) (Larman).

A continuación se muestra el diagrama de despliegue del sistema propuesto, integrado por un sistema embebido y una aplicación de monitoreo. El sistema embebido está compuesto de dos MCU AVR con dos firmwares integrados. Ambos MCU se comunican a través de la interfaz SPI; mientras que el sistema embebido establece relación con la aplicación de monitoreo a través del protocolo Modbus.

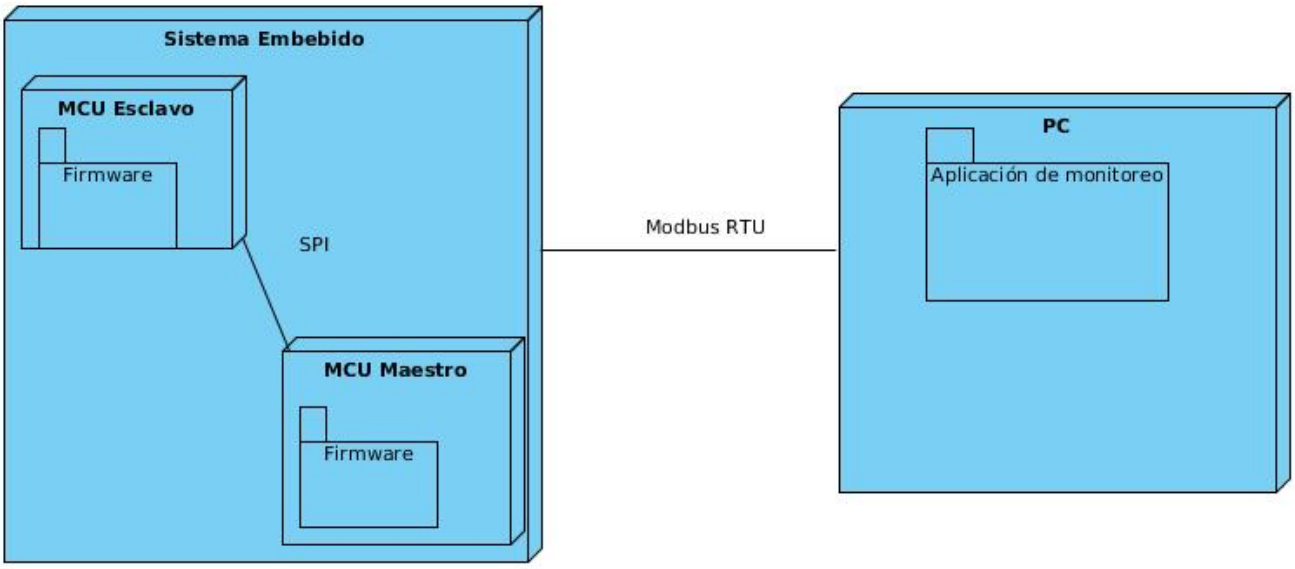

Figura 4 Diagrama de Despliegue

3.5 Tareas de Ingeniería

La metodología XP, propone que una vez que se realice una HU es necesario probar y mostrar al cliente para comprobar si cumple con las especificaciones hechas por el mismo. Para esto, en el proceso de implementación, las HU fueron desglosadas en tareas de ingeniería (TI). A continuaciones algunas tareas de ingenierías planteadas, el resto se encuentran en el anexo.

Tarea de ingeniería	
Número de la tarea: 1	**HU: Comunicar el SE con la computadora.**
Nombre de la tarea: Establecer comunicación entre elSE y la PC.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27/02/2015	**Fecha fin: 03/03/2015**
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff

Descripción: La comunicación entre el sistema embebido y la computadora se hace a través del protocolo Modbus Serial o RTU, con el empleo de la biblioteca libmodbus mediante a interfaz USART para la transmisión de datos. Para iniciar la conexión es preciso la entrada de los siguientes valores: el puerto, bit de datos, bit de parada, tasa de baudios y el ID del MCU esclavo; mientras que para desconectar solo se presiona un botón y se limpia el *buffer* o bufer(espacio de la memoria en un disco*)*.

Tabla 18 TI #1: Establecer comunicación con el MCU y la PC.

Tarea de ingeniería	
Número de la tarea: 2	**HU: Comunicar el SE con la computadora.**
Nombre de la tarea: Establecer comunicación entre los MCU.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 03/03/2015	**Fecha fin:** 05/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: La comunicación entre los MCU se establece mediante la interfaz SPI. Se configuran los MCU uno como esclavo y otro como maestro**.**	

Tabla 19 TI #2: Establecer comunicación entre los MCU.

Tarea de ingeniería	
Número de la tarea: 6	**HU: Controlar el acceso al local.**
Nombre de la tarea**: Controlar el acceso al local.**	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13/03/2015	**Fecha fin:** 13/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: En el firmware se configura un arreglo (conjunto de datos) para 1 PIN. Se ingresa el PIN y se verifica que este correcto, de esta manera se accede al local.	

Tabla 20 TI #6: Controlar al acceso al local.

Tarea de ingeniería	
Número de la tarea: 8	**HU: Controlar el acceso a la**

	aplicación de monitoreo.
Nombre de la tarea: Registrar el usuario y la contraseña.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 17/03/2015	**Fecha fin:** 17/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se registra el nombre del usuario, el tipo y la contraseña. En la base datos se guarda un *hash* MD5 de la contraseña del usuario.	

Tabla 21TI #7: Registrar el usuario y la contraseña.

Tarea de ingeniería	
Número de la tarea: 9	**HU: Controlar el acceso a la aplicación de monitoreo.**
Nombre de la tarea: Controlar al acceso a la aplicación de monitoreo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 18/03/2015	**Fecha fin:** 19/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se verifica el usuario y la contraseña entrada con la guardada en la base datos y se genera un registro o *log con* nombre del usuario y la hora en que accedió a la aplicación.	

Tabla 22 TI #9: Controlar al acceso a la aplicación de monitoreo.

Tarea de ingeniería	
Número de la tarea: 10	**HU: Controlar el acceso a la aplicación de monitoreo.**
Nombre de la tarea: Cambiar el PIN desde la aplicación de monitoreo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 18/03/2015	Fecha fin: 19/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff

Descripción: Se modifica el valor del registro *holding*0 de Modbus por el PIN introducido por el usuario.

Tabla 23 TI #10: Cambiar el PIN desde la aplicación de monitoreo.

Tarea de ingeniería	
Número de la tarea: 7	**HU: Controlar sensores.**
Nombre de la tarea: Modificar sensores.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13/03/2015	**Fecha fin:** 13/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: El alias y el tipo del sensor se guardan en la base datos y se modifica el registro *InputRegister*de Modbus. El tipo de sensor esta predefinido.	

Tabla 24 TI #7: Controlar sensores.

Tarea de ingeniería	
Número de la tarea: 13.	**HU: Mostrar alertas.**
Nombre de la tarea: Mostrar alertas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 10/04/2015	**Fecha fin:** 13/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: La aplicación muestra un listado con las alertas que se generan a partir del cumplimiento de las condiciones configuradas en las notificaciones.	

Tabla 25 TI #7: Mostrar alertas.

Tarea de ingeniería	
Número de la tarea: 14.	**HU: Monitorear las variables incidentes del local.**
Nombre de la tarea: Monitorear las variables incidentes del local.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3

Fecha inicio: 14/04/2015	**Fecha fin:** 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: La aplicación permite mostrar los valores de los sensores. Mediante Modbus se obtiene los valores de los sensores, estos se actualizan cada un tiempo antes configurado y se muestran en la aplicación, no se guardan en la base datos.	

Tabla 26 TI #14: Monitorear las variables incidentes del local.

3.6 Tarjetas CRC (Clase – Responsabilidad – Colaborador)

Las tarjetas CRC son fichas, una por cada clase, en las que se escriben brevemente las responsabilidades de la clase, y una lista de los objetos con los que colabora para llevar a cabo esas responsabilidades. Se desarrollan normalmente en una sesión de trabajo en grupo pequeño (Pressman).

DataProvider	
Descripción: Representación de un registro Modbus.	
Responsabilidades:	
Nombre	Collaborator
Devuelve los datos de los registros Modbus.	ModbusAdapter

Tabla 27 Tarjeta 2: Clase DataProvider

DatabaseAdapter	
Descripción: Permite establecer la comunicación con la base datos SQlite y registrar, guardar y borrar las notificaciones de la base datos,	
Responsabilidades:	
Nombre	Collaborator
Muestra los registros de Modbus por tipo.	DataProvider
Salva, elimina y muestra todas las notificaciones registradas.	Notification

Tabla 28 Tarjeta 1: Clase DatabaseAdapter

NotificationManager	
Descripción: Actualiza los valores de Modbus y chequea las notificaciones que se cumplen y las muestra,	
Responsabilidades:	
Nombre	Collaborator
Permite tomar el nombre y el valor de la notificación.	Notification
Muestra una lista con todas las notificaciones registradas en la base datos,	DatabaseAdapter
Establece la conexión con el servidor (sisteman embebido) y actualiza los datos de todos los registros.	ModbusAdapter

Tabla 29 Tarjeta 3: Clase NotificationManager

LogManager	
Descripción: Permite mostrar los registros de las acciones ejecutadas por el usuario.	
Responsabilidades:	
Nombre	Collaborator
Permite mostrar los registros guardados en la base datos.	DatabaseAdapter

Tabla 30 Tarjeta 4: Clase LogManager

Management	
Descripción: Permite gestionar las notificaciones y los usuarios.	
Responsabilidades:	
Nombre	Collaborator
Permite guardar las nuevas notifiaciones en la base datos.	DatabaseAdapter
Permite crear una nueva notificación.	NewNotification
Permite crear un nuevo usuario.	NewUser

Tabla 31 Tarjeta 5: Clase Management

Login	
Descripción: Permite autenticar un usuario para acceder a la aplicación de monitoreo.	
Responsabilidades:	
Nombre	Collaborator
Permite al usuario acceder a la aplicación.	User

Tabla 32 Tarjeta 6: Clase Login

ModbusAdapter	
Descripción: Se encarga de leer los valores de los registros.	
Responsabilidades:	
Nombre	Collaborator
Establece la comunicación con el servidor (sistema embebido)	Modbus

Tabla 33 Tarjeta 7: Clase ModbusAdapter

ValueEditor	
Descripción: Permite registrar el sensor conectado al MCU con el alias insertado por el usuario.	
Responsabilidades:	
Nombre	Collaborator
Permite tomar los valores de los registros de Modbus.	DataProvider
Permite acceder a la base de datos.	DatabaseAdapter
Permite hacer uso de Modbus.	ModbusAdapter
Permite el empleo con las notificaciones.	NotificationManager

Tabla 34 Tarjeta 8: Clase ValueEditor

3.6 Estándares de Codificación

Un estándar de codificación agrupa los aspectos relacionados con la generación del código. Los programadores lo realizan para trabajar de forma coordinada dentro del proyecto, de acuerdo con el tipo de lenguaje que utilizan y las normas que propone el mismo, tratando de lograr un código claro y legible tanto para él como para otro programador (Computing, 2010).

A continuación se muestran algunos estándares de codificación que se tuvieron en cuenta para la implementación de la aplicación de monitoreo y el firmware.

Longitud de línea: Se evitó la utilización de más de 80 caracteres en una línea.

Comentarios: Se utilizaron los comentarios limitados por /* <Comentario> */ para mostrar información que comprende más de una línea. También se utilizaron los comentarios de la siguiente forma: //-----<Comentario>----, para separar un conjunto de métodos por una sola línea.

Declaraciones: Los métodos y variables se declararon con nombres asociados a la función por la cual fueron creados. En caso de nombres compuestos, el primero se definió en minúscula y los demás comenzando con mayúscula, mientras que en las clases se definen los dos con mayuscula, ejemplo: NotificationManagement. En caso de separarlos no se utilizó espacios, sino guiones bajos (<nombre1>_<nombre2>).

Sentencias if, switch, foreach, for: se utilizó llaves para abrir y cerrar el bloque de acción (es) que se ejecutan dada (s) su (s) condición (es).

Variables: Todas las instancias y variables de clase o método comenzaron con minúscula. Las palabras internas que lo forman empezaron con su primera letra en mayúsculas. Los nombres de variables no debían empezar con un subguión "_".

3.7 Conclusiones

En este capítulo se determinó que patrón arquitectónico emplear en el SE de acuerdo a los dos microcontroladores que lo compone y al rol que ocupa cada uno empleando la interfaz de comunicación SPI. También se definió la arquitectura por capas en la aplicación de monitoreo y como arquitectura del sistema se plantea la misma que usa el protocolo Modbus (Cliente-Servidor).Se especificaron los patrones de diseño y arquitectónicos que se emplean en la elaboración del sistema. Se describió el flujo de eventos que se activan para el control de acceso al local. Además se representó el diagrama de despliegue, con el fin de modelar la solución final y se definieron las tarjetas CRC como artefacto de la metodología XP para modelar y ver la relación entre las clases, las tareas de ingeniería y los estándares de codificación para el desarrollo del sistema.

CAPÍTULO 4: Pruebas y resultados

La prueba es un conjunto de actividades que se plantean con anticipación y se realizan de manera sistemática. Una estrategia para la prueba de software debe incluir pruebas de bajo nivel (necesarias para confirmar la correcta implementación de un pequeño segmento de código fuente) y de alto nivel (que validen las principales funciones del sistema a partir de los requisitos del cliente) (Pressman).

4.1 Estrategia de Prueba

Para la realización de las pruebas se creó una estrategia, en la que se tuvo en cuenta los niveles de las mismas, brindando la posibilidad de seleccionar las pruebas más factibles para comprobar las funciones desarrolladas como parte de la propuesta de solución. Entre los niveles se encuentran (Pressman):

- Prueba del sistema (Ingeniería del sistema).
- Prueba de validación (Requisitos).
- Prueba de integración (Diseño).
- Prueba de unidad (Código).

La metodología XP propone realizar tantas pruebas como sea posible con el objetivo de disminuir los errores que puedan afectar los resultados. Se encuentran divididas en dos grupos, pruebas unitarias y pruebas de aceptación, también llamadas pruebas del cliente. También se puede hacer uso de otras pruebas que los desarrolladores crean necesario utilizar con el objetivo de garantizar el correcto funcionamiento de la propuesta funcional de la solución.

Pruebas Unitarias: Son consideradas una de las etapas imprescindibles de XP. Este tipo de prueba se debe realizar antes de escribir el código, de forma tal que se le aplica a todos los métodos que se implementen una vez terminados y antes de ser publicados. Que todo el código realizado pase correctamente las pruebas unitarias es lo que garantiza que funcionen en conjunto con los demás. En este sentido, las pruebas deben ser guardadas en conjunto con el sistema, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (Uruguay, 2015). Se considera necesario probar el flujo de datos de la interfaz del módulo antes de iniciar cualquier otra prueba. Si los datos no entran ni salen apropiadamente, todas las demás pruebas resultarán inútiles. Además, durante la prueba de unidad deben recorrerse las estructuras de datos locales y evaluarse (si es posible) el impacto local sobre los datos globales (Labaut, 2012).

Pruebas de integración: Se utilizan con el objetivo de tomar componentes probados en unidad y estructurar un programa, de forma tal que interactúen correctamente y estén de acuerdo con el

programa diseñado. La integración puede ser descendente si se integran los módulos desde el control o programa principal, o bien, ascendente, si la verificación del diseño empieza desde los módulos más bajos y de allí al principal. La selección de una estrategia de integración depende de las características del software (Pressman).

Prueba de validación: El objetivo de estas pruebas es obtener información útil para la validación de la implementación de los algoritmos estudiados. Se asume para esta parte que el software ha cumplido la etapa de verificación, por lo tanto está libre de errores de tiempo de ejecución, lo que no significa que esté libre de errores lógicos (diferencias entre la estrategia propuesta y la implementada) (Computing, 2010).

Prueba del sistema: Son similares a las pruebas de caja negra, solo que éstas buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema.

4.2. Pruebas de Aceptación

Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como Pruebas de Caja Negra[20]. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (Uruguay, 2015).

Los casos de prueba derivados de las HU fueron elaborados por el equipo de desarrollo y probados por el cliente. Este se enfocó en las características generales y las funcionalidades del sistema, creando diversos escenarios para comprobar que una historia de usuario fue desarrollada correctamente.

Los casos de pruebas se especifican a continuación, las restantes se encuentran en el anexo.

Caso de Prueba de Aceptación	
Código: HU1_CP1	**Historia de Usuario: Comunicar el SE con la computadora.**
Nombre: Introducir los parámetros requeridos por la aplicación de monitoreo para establecer la	

20 Permiten obtener un conjunto de condiciones de entrada que ejerciten todos los requisitos funcionales de un Software.

comunicación.
Descripción: Se desea probar que el sistema permite establecer la comunicación entre la aplicación y el sistema embebido.
Condiciones de Ejecución: Todos los campos de los parámetros deben de estar llenos.
Entrada/Pasos de Ejecución: • El usuario introduce los parámetros que son pedidos por la aplicación de monitoreo. • Presiona el botón: “Connect”.
Resultado Esperado: El sistema establece la comunicación entre la aplicación y el sistema embebido.
Evaluación de la Prueba: Satisfactoria.

Tabla 35 CP1: Introducir los parámetros requeridos por la aplicación de monitoreo para establecer la comunicación.

Caso de Prueba de Aceptación	
Código: HU2_CP2	**Historia de Usuario: Controlar el acceso al local.**
Nombre: Ingresar el valor del PIN por el teclado matricial.	
Descripción: Se desea probar que el sistema permite introducir al usuario el valor del PIN para acceder al local.	
Condiciones de Ejecución: El valor del PIN debe ser de 4 dígitos.	
Entrada/Pasos de Ejecución: • El usuario presiona el botón”*login*”. • El usuario introduce el PIN. • El usuario tiene acceso al local.	
Resultado Esperado: El sistema permite el acceso al local.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 36 CP2: Ingresar el valor del PIN por el teclado matricial.

Caso de Prueba de Aceptación	
Código: HU2_CP3	**Historia de Usuario: Controlar el acceso al local.**

Nombre: Ingresar el valor del PIN incorrecto por el teclado matricial.
Descripción: Se desea probar que el sistema no permite al usuario introducir el valor del PIN incorrecto para acceder al local.
Condiciones de Ejecución: El valor del PIN debe ser de 4 dígitos.
Entrada/Pasos de Ejecución: • El usuario presiona el botón"*login*". • El usuario introduce el PIN. • El *display* muestra un mensaje de Error. • El usuario no tiene acceso al local.
Resultado Esperado: El sistema no permite el acceso al local y espera por el correcto valor del PIN.
Evaluación de la Prueba: No satisfactoria.

Tabla 37 CP3: Ingresar el valor del PIN incorrecto por el teclado matricial.

Caso de Prueba de Aceptación	
Código: HU3_CP4	**Historia de Usuario: Controlar el acceso a la aplicación de monitoreo.**
Nombre: Ingresar los parámetros para acceder a la aplicación de monitoreo.	
Descripción: Se desea probar que el sistema permite al usuario introducir los parámetros (usuario, contraseña) para acceder a la aplicación.	
Condiciones de Ejecución:	
Entrada/Pasos de Ejecución: • El usuario introduce la contraseña y el usuario. • El usuario presiona el botón "*login*".	
Resultado Esperado: El sistema no permite el acceso al local y espera por el correcto valor del PIN.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 38 CP4: Ingresar los parámetros para acceder a la aplicación de monitoreo.

Caso de Prueba de Aceptación

Código: HU3_CP5	**Historia de Usuario: Controlar el acceso a la aplicación de monitoreo.**
Nombre: Cambiar el PIN desde la aplicación de monitoreo.	
Descripción: Se desea probar que el sistema permite al usuario cambiar el PIN con el que se accede al local desde la aplicación de monitoreo.	
Condiciones de Ejecución: El usuario que estar autenticado como Administrador. Debe de haber presionado el link "*security*".	
Entrada/Pasos de Ejecución: • El usuario presiona el link "*users*". • El usuario presiona el boton "door pin". • El usuario inserta el nuevo PIN.	
Resultado Esperado: El sistema permite al usuario cambiar el PIN con el que se accede al local desde la aplicación de monitoreo.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 39 CP5: Cambiar el PIN desde la aplicación de monitoreo.

Caso de Prueba de Aceptación	
Código: HU4_CP6	**Historia de Usuario: Gestionar usuario.**
Nombre: Insertar usuario.	
Descripción: Se desea probar que el sistema inserta usuarios.	
Condiciones de Ejecución: El usuario Administrador es el responsable de insertar un nuevo usuario. Debe de haber presionado el link "*security*".	
Entrada/Pasos de Ejecución: • El usuario presiona el link *"user".* • El usuario selecciona la opción: "new". • El usuario ingresa los parámetros (*name, username, password, type*) requeridos por la aplicación. • El usuario presiona el botón "save".	
Resultado Esperado: El sistema inserta un nuevo usuario.	

Evaluación de la Prueba: Satisfactoria.

Tabla 40 CP6: Insertar usuario.

Caso de Prueba de Aceptación	
Código: HU4_CP7	**Historia de Usuario: Gestionar usuario.**
Nombre: Modificar usuario.	
Descripción: Se desea probar que el sistema permite modificar unusuario.	
Condiciones de Ejecución: El usuario Administrador es el único con permisos de gestionar usuario. Debe de haber presionado el link "*security*".	
Entrada/Pasos de Ejecución: • El usuario presiona el link *"user".* • El usuario selecciona la opción:"edit". • El usuario ingresa los parámetros (*name, username, password, type*) requeridos por la aplicación. • El usuario presiona el botón "save"*.*	
Resultado Esperado: El sistema modifica un usuario.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 41 CP7: Modificar usuario.

Caso de Prueba de Aceptación	
Código: HU4_CP8	**Historia de Usuario: Gestionar usuario.**
Nombre: Eliminar usuario.	
Descripción: Se desea probar que el sistema permite eliminar un usuario.	
Condiciones de Ejecución: El usuario Administrador es el único con permisos de gestionar usuario.. Debe de haber presionado el link "*security*" y después el link *"user".*	
Entrada/Pasos de Ejecución: • El usuario selecciona la opción:"delete". • El sistema borra el usuario.	
Resultado Esperado: El sistema elimina un usuario.	

Evaluación de la Prueba: No satisfactoria.

Tabla 42 CP8: Eliminar usuario.

Caso de Prueba de Aceptación	
Código: HU4_CP9	**Historia de Usuario: Gestionar usuario.**
Nombre: Mostrar usuario.	
Descripción: Se desea probar que el sistema permite mostrar unusuario.	
Condiciones de Ejecución: El usuario Administrador es el único con permisos de gestionar usuario. Debe de haber presionado el link "*security*" y después el link *"user".*	
Entrada/Pasos de Ejecución: • El sistema muestra los usuarios registrados.	
Resultado Esperado: El sistema muestra al usuario.	
Evaluación de la Prueba: No satisfactoria.	

Tabla 43 CP9: Mostrar usuario.

Caso de Prueba de Aceptación	
Código: HU5_CP10	**Historia de Usuario: Controlar sensores.**
Nombre: Modificar sensores.	
Descripción: Se desea probar que el sistema reconozca los sensores conectados al MCU.	
Condiciones de Ejecución: Debe estar conectado el sensor al MCU y el usuario seleccionar el link "*remote*" y después el link "*sensors*".	
Entrada/Pasos de Ejecución: • El sistema muestra un listado con los sensores, los valores captados y el tipo de sensor. • El usuario selecciona la opción:"*type*" y elige el tipo de sensor que sea el conectado. • El usuario selecciona un sensor. • El sistema muestra un formulario con los espacios en blanco para el parámetro a insertar. • El usuario inserta el parámetro (*alias*). • El usuario presiona el botón "*save*".	
Resultado Esperado: El sistema modifica el nombre y el tipo de los sensores conectados.	

Evaluación de la Prueba: Satisfactoria.

Tabla 44 CP10: Modificar sensores.

Caso de Prueba de Aceptación	
Código: HU5_CP11	**Historia de Usuario: Controlar sensores.**
Nombre: Mostrar sensores.	
Descripción: Se desea probar que el sistema no reconoce sensores digitales.	
Condiciones de Ejecución: Debe estar conectado el sensor.	
Entrada/Pasos de Ejecución: • El usuario presiona el link *"sensors".* • El sistema muestra un listado de los sensores instalados con valores no esperados.	
Resultado Esperado: El sistema no reconoce los sensores conectados.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 45 CP11: Mostrar sensores.

Caso de Prueba de Aceptación	
Código: HU6_CP12	**Historia de Usuario: Gestionar Notificaciones.**
Nombre: Crear notificación.	
Descripción: Se desea probar que el sistema permite crear una notificación.	
Condiciones de Ejecución: Haber presionado el link "*notifications"*.	
Entrada/Pasos de Ejecución: • El usuario presiona el link "manager". • El usuario presiona el botón"*new"*. • El usuario introduce los parámetros. • El usuario presiona el botón"*save"*.	
Resultado Esperado: El sistema permite crear una nueva notificación.	

Evaluación de la Prueba: Satisfactoria.

Tabla 46 CP12: Crear notificación

Caso de Prueba de Aceptación	
Código: HU6_CP13	**Historia de Usuario: Gestionar Notificaciones.**
Nombre: Modificar notificación.	
Descripción: Se desea probar que el sistema permite modificar una notificación.	
Condiciones de Ejecución: Tener notificaciones registradas y el usuario haber presionado el link "*notifications"*.	
Entrada/Pasos de Ejecución: • El usuario presiona el link "*manage*". • El usuario selecciona una notificación. • El usuario presiona el botón "*edit"*. • El usuario introduce los parámetros. • El usuario presiona el botón "*save"*.	
Resultado Esperado: El sistema permite modificar una notificación.	
Evaluación de la Prueba: No satisfactoria.	

Tabla 47 CP13: Modificar Notificación

Caso de Prueba de Aceptación	
Código: HU6_CP14	**Historia de Usuario: Gestionar Notificaciones.**
Nombre: Eliminar notificación.	
Descripción: Se desea probar que el sistema permite eliminar una notificación.	
Condiciones de Ejecución: Tener notificaciones registradas y el usuario haber presionado el link "*notifications"*.	
Entrada/Pasos de Ejecución: • El usuario presiona el link "manage". • El usuario selecciona una notificación.	

• El usuario presiona el botón “*delete”*. • El sistema borra la notificación.
Resultado Esperado: El sistema permite eliminar una notificación.
Evaluación de la Prueba: Satisfactoria.

Tabla 48 CP14: Eliminar notificación

Caso de Prueba de Aceptación	
Código: HU6_CP15	**Historia de Usuario: Gestionar Notificaciones.**
Nombre: Mostrar notificación.	
Descripción: Se desea probar que el sistema permite mostrar una notificación.	
Condiciones de Ejecución: Tener notificaciones registradas y el usuario haber presionado el link “*notifications”*.	
Entrada/Pasos de Ejecución: • El sistema muestra la notificación.	
Resultado Esperado: El sistema permite mostrar una notificación.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 49 CP15: Mostrar Notificación

4.3. Ejecución de los casos de pruebas de aceptación

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos encontrados en la anterior, para garantizar que al final del proceso el producto quede libre de la mayor cantidad de errores posible y listo para entregar al cliente. Durante la ejecución de los casos de pruebas de aceptación algunas no arrojaron los resultados esperados. A continuación se listan los casos de prueba de aceptación que arrojaron alguna no conformidad durante la ejecución de la primera iteración de pruebas:

1. Ingresar el valor del PIN incorrecto por el teclado matricial.
2. Eliminar usuario.
3. Mostrar usuario.
4. Modificar Notificación.

La plantilla de no conformidades constituye un registro de los defectos y fallos encontrados en el transcurso de las pruebas, cuyo principal objetivo es verificar en un futuro que estos errores fueron erradicados en posteriores iteraciones. La siguiente tabla muestra un resumen del registro de defectos y dificultades encontradas durante la primera iteración de pruebas:

No.	No Conformidad	Aspecto Correspondiente	Etapa de detección	Respuesta del desarrollador
1	Se intentó que el sistema no permitiera el acceso al local con un PIN incorrecto y que mostrara el mensaje de advertencia deseado.	Botón *login* del teclado matricial	Prueba de Aceptación	Resuelta una vez terminada la primera iteración de prueba.
2	El sistema no permitió que seleccionado un usuario se eliminara el mismo de la base de datos mediante el botón *delete*.	Botón *delete*	Prueba de Aceptación	Resuelta una vez terminada la primera iteración de prueba.
3	El sistema no muestra los usuarios registrados.	Menú desplegable *security*	Prueba de Aceptación	Resuelta una vez terminada la primera iteración de prueba.
4	El sistema no permitió que seleccionada una notificación se modificara a través del botón *edit*.	Botón *edit*	Prueba de Aceptación	Resuelta una vez terminada la primera iteración de prueba.

Tabla 50 No conformidades por HU.

4.4 Resultados

Para que un proceso de pruebas tenga éxito se requiere de un análisis final de los resultados arrojados, es decir, la evaluación del producto que se está probando de acuerdo a todos los defectos y fallos del sistema encontrados a lo largo del proceso. Las pruebas de aceptación se realizaron en dos iteraciones y se utilizaron los 19 casos de pruebas diseñados, para verificar que las funcionalidades implementadas fueron las acordadas en las historias de usuario y que respondían correctamente a sus necesidades. A continuación se muestra un cuadro resumen que muestra los resultados obtenidos en las pruebas ejecutadas durante las dos iteraciones realizadas:

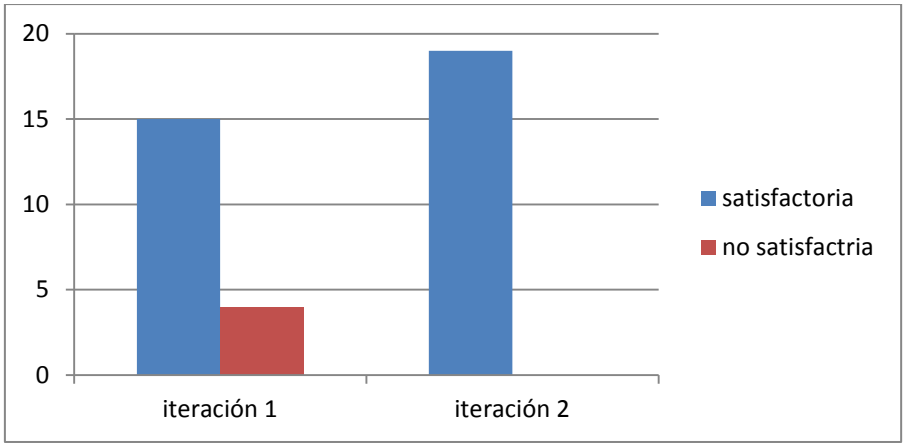

Tabla 51 Resultados de las pruebas

Como se muestra en la Figura 6, de los 19 casos de pruebas diseñados 15 resultaron exitosos, existiendo 4 que lanzaron una no conformidad. Por tal motivo, una vez resueltos los defectos detectados se pasó a una segunda iteración de pruebas, donde se volvieron a realizar todos los casos de prueba diseñados, para verificar que fueron resueltas las no conformidades de la primera iteración y que además no habían sufrido cambios los casos de pruebas exitosos de la primera iteración. Al concluir la segunda iteración, todos los casos de prueba arrojaron resultados satisfactorios.

4.5 Conclusiones

En este capitulo se realizó un estudio de las estrategias de pruebas que se hacen a la solución. Se decide validar los requisitos del cliente a través de las pruebas de aceptación. Recomendadas a efectuar por la metodología XP para verificar el cumplimiento de las funcionalidades del sistema. El equipo de desarrollo definió 19 casos de pruebas y el cliente las verificó. Como resultado se obtuvieron en la primera iteración 15 casos de pruebas satisfactorias y 4 no satisfactorias. Para la segunda iteración se corrigieron las 4 no conformidades y se obtuvieron como resultado todos los casos de pruebas satisfactorias.

Conclusiones Generales

En la presente investigación se hizo un estudio de los sistemas domóticos, de los sistemas embebidos y de los dispositivos que se emplean, en específico los MCU. El análisis fue realizado sobre un sistema embebido en MCU AVR y una aplicación de monitoreo. Ambos integran un sistema de monitoreo capaz de generar alertas de acuerdo a las notificaciones que se configuren, de controlar sensores analógicos, de monitorear las variables incidentes de un local según el sensor que este conectado al MCU, de brindar facilidades al usuario de conocer el estado en que se encuentra un local aislado desde la entidad y de prevenir de cualquier incidente.

Para el desarrollo del sistema se aplicó la metodología XP, que ha sido utilizada con éxito en proyectos de desarrollo de sistemas embebidos. Dicha metodología permitió definir junto al cliente los requisitos funcionales y no funcionales con los que debía cumplir el sistema de monitoreo; determinándose la arquitectura, las herramientas y tecnologías para dar solución al problema planteado. El desarrollo fue guiado a través de diez historias de usuarios que contienen un conjunto de tareas que tributaron a obtener prototipos funcionales en cada iteración que fueron probados y aceptados por el cliente, hasta obtener la solución final.

El uso del sistema operativo GNU/Linux con la distribución Debian permitió el empleo de herramientas para la compilación y programación del código para el firmware. Además posibilitó economizar los recursos financieros evitando el pago de licencias de software. También el empleo de la biblioteca libmodbus permitió la comunicación entre el sistema embebido y la aplicación de monitoreo, mientras que la interfaz SPI facilitó la comunicación entre los microcontroladores que compone al sistema embebido y la interfaz USART posibilitó el envío y recepción de datos a través del puerto serial.

A través de las pruebas realizadas durante el proceso de desarrollo se validó la solución propuesta obteniéndose como resultado un sistema de monitoreo basado en microcontroladores AVR para monitorear las variables incidentes de un local aislado a la entidad a la que pertenece. Además se obtiene mejor seguridad mediante el control de acceso y el firmware embebido en el MCU puede ser ejecutado en cualquier otro miembro de la familia de los AVR.

Recomendaciones

1. Para tener comunicación entre un local y la entidad a mayor distancia se recomienda hacer uso de un adaptador Ethernet.
2. Emplear un MCU con mayores recursos.
3. Cambiar el driver de Modbus RTU a TCP.

Bibliografía

Domótica. Energía espacial, en la Tierra. **NeoTeo. 2007.** 2007.

academia.edu. 2015. [En línea] 2015. http://www.academia.edu/9795641/INGENIER%C3%8DA_DEL_SOFTWARE_METODOLOG%C3%8DAS_Y_CICLOS_DE_VIDA_Laboratorio_Nacional_de_Calidad_del_Software.

AIE. 2015. Asociación de la Industria Eléctrica-Electrónica de Chile. [En línea] 13 de Febrero de 2015. http://www.aie.ci.

Alcoy, Escuela Politécnica Superior de. 2015. [En línea] Abril de 2015. server-die.alc.upv.es.

Alicante, Universidad de. 2015. Repositorio Intitusional de la Universidad de Alicante. [En línea] 2015. http://rua.ua.es.

Angulo, J. 2003. *Microcontroladores PIC Diseño practico de aplicaciones.* Madrid : s.n., 2003.

Anisbel Clara Hernández, Delís Ise Morales. 2014. *Trabajo de disploma para optar por el título de Ingeniero en Ciencias Informáticas: Integración de herramientas de pruebas de seguridad para aplicaciones web en XILEMA-PlatSI.* La Habana : Universidad de las Ciencias Informáticas, 2014.

Arias, Danae Pérez. 2011. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Servicio de Facturación para Elastix. Universidad de Las Ciencias Informáticas. La Habana.* 2011.

Automation, Control &. 2015. [En línea] 2015. http://control.sdindustrial.com.mx.

Barr, M. 2006. *Programming Embedded Systems. O'Reilly.* 2006.

Berger, Arnold S. 2002. *Embedded Systems Design: An Introduction to Processes, Tools and Techniques.* 2002.

Cohen Manrique, C. 2005. *Diferencias entre microcontroladores y microprocesadores.* 2005.

Computing, Imperial College Department of. 2010. FOLDOC Computing Dictionary. [En línea] 2010. http://foldoc.org/firmware..

Conocimiento, Comunidad Digital del. 2012. Las pruebas de integración de software. [En línea] 2012. http://www.academica.mx/blogs/las-pruebas-integraci%C3%B3n-software..

Cortez, O. O. 2009. *Boletín Escuela de Informática.* El Salvador. : s.n., 2009.

Crips, J. 2004. *Introdution to Microprocessaors and Microcontrollers.O.* 2004.

Cruz, C. F. 2009. *Comunicación entre microcontroladores bajo el protocolo ZIGBEE. Tesis para optar por el título de Doctor en Ciencias de la Computación.* Madrid : Universidad Politécnica de Madrid., 2009.

Debian. 2015. [En línea] 2015. https://www.debian.org.

Drobka, J. D. 2004. *Piloting XP onFourMission-CriticalProjects. IEEE Software.* 2004.

Felipe, Percy Viego. 2015. *Cubasolar.* [En línea] Mayo de 2015. http://www.cubasolar.cu.

Flores, Omar Otoniel. 2009. *Boletin "BATALLA DE MICROCONTROLADORES ¿AVR o PIC?".* 2009.

Harper, D. 2010. Etymology Dictionary. 2010.

Heath, Steve. 2003. *Embedded Systems Design.* s.l. : 2nd Ed. Oxford : Newnes, 2003.

Iberico, Roberto Acosta. 2015. Escuela Superior de Ingenieros. Grupo de Robótica, Visión y Control. [En línea] 2015. http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/523/IBERICO_ACOSTA_ROBERTO_SISTEMA_SEGURIDAD_SENSOR_ZIGBEE.pdf?sequence=1.

Improving Mobility in eXtreme Programming Methods through. **RIDI, F. 2012.** s.l. : International Journal of Conputer Science and Technology, 2012.

Informáticos, S. 2013. Dpto. Sistemas Informáticos y Computación:UPV. [En línea] 2013. http://www.upv.es/entidades/DSIC/index.html..

Joskowicz, Ing. José. 2008. *Reglas y Prácticas en eXtreme Programming.* 2008.

Labaut, M. A. 2012. *Trabajo de disploma para optar por el título de Ingeniero en Ciencias Informáticas: Firmware para tarjeta de adquisición basada en un microcontrolador STM32.* La Habana : Universidad de Las Ciencias Informáticas, 2012.

Larman, Crain. *UML y Patrones.*

Miñarro, B. Ú. 2009. Apuntes de: Sistemas embebidos. 2009.

Olivia, M. 2012. *Firmware para tarjeta de adquisición basada en un microncontrolador STM32. Tesis para optar por el título de Ing. En Ciencias Informáticas. La Habana: Universidad de las Ciencias Informáticas.* La Habana : s.n., 2012.

Pierce, D. 2004. *Extreme Programming without Fear. Embedded Systems Programming.* 2004.

PIERCE, D. 2004. *Extreme Programming without Fear. Embedded Systems Programming.* 2004.

Pressman. *Ingeniería del Software. Un enfoque práctico.* s.l. : 6ta Edición.

Rayma Krishna, C. 2011. *Survey on Extreme Programming in Software Engineering. International Journal of Computer Trends and Technology.* 2011.

Ridi, F. 2012. *Improving Mobility in eXtreme Programming Methods through. International Journal of Conputer Science and Technology.* 2012.

Rodríguez, A. G. 2013. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Aplicación de Configuración Web para Conversores de Protocolos. Universidad de Las Ciencias Informáticas.* La Habana. : s.n., 2013.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 1998. *UML. Manual de referencia.* s.l. : Addison Wesley, 1998.

Sanchez, M. 2014. *Metodologías de Desarrollo de Software.* [En línea] Febrero de 2014. http://www.informatizate.net.

Santiago, F. 2013. *Los microcontroladores AVR de Atmel.* 2013.

2015. Seguridad domestica. [En línea] Mayo de 2015. http://www.livemodern.org/teoria-de-la-domotica/sensores-en-domotica/#sthash.pIh7Eg8S.dpuf.

Sociales, S. N. 2015. *Universitat de Barcelona.* [En línea] Mayo de 2015. http://www.ub.edu/geocrit/sn/sn-146%28136%29.htm.

Solias, M. 2003. Una explicación de la programacion extrema (XP). . [En línea] 2003. http://ww.apolosoftware.com.

Tiscornia, E. 2015. Domótica. La Vivienda Inteligente. . *Universidad de Palermo.* [En línea] 2015. http://www.palermo.edu/ingenieria/downloads/CyT3/CYT308.pdf.

Uruguay, Universidad de la Republica. 2015. Facultad de Ingeniería. . [En línea] Mayo de 2015. Obtenido de http://www.fing.edu.uy.

Vasco, Universidad del pais. 2015. *Campus Gipuzkoa: .* [En línea] 2015. http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/PLC/plc.htm.

WEG. 2015. [En línea] 13 de Febrero de 2015. http://www.weg.net.

WEQ. 2012. *Manual de la Comunicación MODBUS-RTU.* 2012.

Yera, Anier Sánchez. 2013. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Conversor de protocolo Modbus TCP a Modbus RTU basado en el microcontrolador STM32F107VCT.Universidad de Las Ciencias Informáticas.* La Habana : s.n., 2013.

Zamorano, J. 2014. Sistemas embebidos y ubicuos. *Master Universitario en Ingenieria Informatica.* [En línea] 2014. http://www.datsi.fi.upm/docen.

Glosario de términos

ASIC: Circuito Integrado de aplicación específica.

Circuito de control: Es la parte más delicada de la controladora, se encarga de controlar las entradas (Puerto LPT, Entradas Analógicas, Entradas Digitales y circuito de potencia) y las salidas (Salidas Digitales).

Circuito Integrado: Circuito hecho a la medida para un uso en particular, en vez de ser concebido para propósitos de uso general.

MISO: Señal de entrada al Maestro y salida en el Esclavo.

MOSI: Señal de salida del Maestro y entrada en el Esclavo.

PIN: Número de identificación personal.

PWM: Modulación por ancho de pulsos.

RXD: Terminal destinada para la recepción de datos.

SCK: Señal del reloj para sincronizar la comunicación. Es generada por el Maestro.

SS: Señal útil para el manejo de múltiples Esclavos.

TXD: Terminal destinada para la transmisión de datos.

Vdc : Unidad de Voltaje en Corriente Continua (CC).

Anexos

Tarjetas CRC

DataView	
Descripción: Vista encargada de mostrar los valores de un tipo de registro de Modbus.	
Responsabilidades:	
Nombre	Collaborator
Muestra todos los registros Modbus por tipo.	DatabaseAdapter
Establece la conexión con el servidor (sistema embebido).	ModbusAdapter

SettingsManager	
Description: Permite la conexión entre el Sistema Embebido y la computadora.	
Responsibilities:	
Name	Collaborator

NewNotification	
Description: Se encarga de crear una nueva notificación.	
Responsibilities:	
Name	Collaborator
Salva la notificación creada en la base datos.	DatabaseAdapter
Crea una nueva notificación.	Notification

NewUser	
Description: Permite registrar un nuevo usuario.	
Responsibilities:	
Name	Collaborator
Permite que se guarde un nuevo usuario en la base datos,	DatabaseAdapter

Notification	
Description: Representa una entidad..	
Responsibilities:	
Name	Collaborator

NotificationManagement	
Description: Se encarga gestionar la notificación.	
Responsibilities:	
Name	Collaborator
Muestra, registra y elimina las notificaciones.	DatabaseAdapter
Crea,modifica,salva y muestra en una lista las notificaciones creadas.	NewNotification

LoginManager	
Descripción: Permite mostrar el formulario con los valores que se necesitan insertar para acceder a la aplicación de monitoreo.	
Responsabilidades:	
	Collaborator
Muestra el formulario.	Login
Verifica que el usuario autenticado sea el registrado en la base datos,	User
Permite acceder a la base datos.	DatabaseAdapter
Guarda en la base datos.	LogManager

Notifications	
Description: Se encarga de mostrar las notificicaciones que están ocurriendo.	
Responsibilities:	
Name	Collaborator
Muestra una lista con todas las notificaciones que están ocurriendo en ese momento.	NotificationManager

Tareas de Ingenieria

Tarea de ingeniería	
Número de la tarea: 3	Número de HU: 2.
Nombre de la tarea: Ingresar el valor del PIN por el teclado matricial.	
Tipo de tarea: Desarrollo	Puntos estimados: 0,5
Fecha inicio: 06/03/2015	Fecha fin: 09/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: El PIN es ingresado desde el teclado matricial, cuenta solo con 4 dígitos, este se guarda en el registro *holding* de Modbus, dicho registro por defecto tiene ingresado un PIN con valor 0 el cual no será válido para la entrada al local.	

Tabla #1 : TI 1 : Modificar el valor del PIN.

Tarea de ingeniería	
Número de la tarea: 4	Número de HU: 2.
Nombre de la tarea: Modificar el valor del PIN.	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio: 10/03/2015	Fecha fin: 11/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Desde la aplicación de monitoreo a través de la opción "*user pins"* es modificado el PIN de esta manera se modifica el registro holding de Modbus.	

Tabla #2 : TI 4 : Modificar el valor del PIN.

Tarea de ingeniería	
Número de la tarea: 5	Número de HU: 2.
Nombre de la tarea: Verificar el valor del PIN.	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio: 11/03/2015	Fecha fin: 12/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff

Descripción: Se verifica el PIN guardado en el holding (Registro de salida), con el PIN entrado desde el teclado matricial.

Tabla #3 : T1 5: Verificar el valor del PIN.

Tarea de ingeniería	
Número de la tarea: 7	Número de HU: 3.
Nombre de la tarea: Verificar el usuario y la contraseña.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 16/03/2015	Fecha fin: 16/03/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se verifica el usuario y la contraseña desde la Base Datos.	

Tabla # : TI 7: Verificar el usuario y la contraseña.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Insertar notificación.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #5 : TI : Insertar notificación.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Modificar notificación.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff

Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.

Tabla #6 : TI 10 : Modificar notificación.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Eliminar notificación.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #7 : TI 10 : Eliminar notificación.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Mostrar notificación.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #8 : TI 10 : Mostrar notificación.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Insertar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015

Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #9 : TI 10 : Insertar usuario.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Modificar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #10 : TI 10 : Modificar usuario.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Eliminar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #11 : TI 10 : Eliminar usuario.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Mostrar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 3

Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #12 : TI 10 : Mostrar usuario.

Tarea de ingeniería	
Número de la tarea: 15.	Número de HU: 9.
Nombre de la tarea: Registrar las acciones que ocurren en la aplicación de monitoreo.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se guardan en la base datos todas las acciones que son ejecutadas en la aplicación de monitoreo.	

Tabla #13 : TI 15: Monitorear de variables incidentes sobre un local.

Tarea de ingeniería	
Número de la tarea: 16.	Número de HU: 10.
Nombre de la tarea: Mostrar las acciones ejecutadas.	
Tipo de tarea: Desarrollo	Puntos estimados: 3
Fecha inicio: 14/04/2015	Fecha fin: 17/04/2015
Programador Responsable:	Antuanett B. Frometa, Yordan C. Perez Atttanaff
Descripción: Se muestra una lista con todas las acciones ejecutadas en la aplicación de monitoreo, antes registradas en la base datos.	

Tabla #14 : TI 16: Monitorear de variables incidentes sobre un local.

Casos de Prueba

Caso de Prueba de Aceptación	
Código: HU10_CP18	**Historia de Usuario: Mostrar las acciones ejecutadas en la aplicación de monitoreo.**
Nombre: Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.	
Descripción: Se desea probar que el sistema muestra las acciones que ocurren en la aplicación de monitoreo.	
Condiciones de Ejecución: El usuario tiene que estar autenticado como Administrador.	
Entrada/Pasos de Ejecución: • El usuario selecciona la opción"*logs".* • La aplicación muestra los registros de las acciones ejecutadas por el usuario.	
Resultado Esperado: El sistema muestra los registros de las acciones ejecutadas por el usuario en la aplicación de monitoreo.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 15 CP19: Mostrar los registros de las acciones ejecutadas en la aplicación de monitoreo.

Caso de Prueba de Aceptación	
Código: HU7_CP16	**Historia de Usuario: Mostrar alertas.**
Nombre: Mostrar alertas.	
Descripción: Se desea probar que el sistema muestre las alertas generadas.	
Condiciones de Ejecución: Haber configurado alguna notificación.	
Entrada/Pasos de Ejecución: • El usuario presiona el link "notifications". • El sistema muestra un listado de las alertas generadas.	
Resultado Esperado: El sistema muestra las alertas generadas de acuerdo a las notificaciones configuradas.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 16 CP16: Mostrar alertas.

Caso de Prueba de Aceptación

Código: HU8_CP17	**Historia de Usuario: Monitorear las variables incidentes del local.**
Nombre: Monitorear las variables incidentes del local.	
Descripción: Se desea probar que el sistema muestra el estado de las variables	
Condiciones de Ejecución: Tener conectado al menos un sensor analógico.	
Entrada/Pasos de Ejecución: • El usuario presiona el link "*remote"*. • El usuario selecciona la opción (*coils, sensors*) que desee de acuerdo a la variable que quiera conocer. • El sistema muestra un listado con los valores.	
Resultado Esperado: El sistema muestra los variables incidentes de seguridad del local.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 17 CP17: Monitorear las variables incidentes del local.

Caso de Prueba de Aceptación	
Código: HU9_CP18	**Historia de Usuario: Registrar las acciones que ocurren en la aplicación de monitoreo**
Nombre: Registrar las acciones que ocurren en la aplicación de monitoreo.	
Descripción: Se desea probar que el sistema registra las acciones que ocurren en la aplicación de monitoreo en la base datos.	
Condiciones de Ejecución:	
Entrada/Pasos de Ejecución: • El usuario interactúa con la aplicación.	
Resultado Esperado: El sistema registra las acciones en la base datos.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 18 CP18: Registrar las acciones que ocurren en la aplicación de monitoreo.