Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Área Académica de Ingeniería

Desarrollo de una Plataforma para la Definición de Reglas ECA en una Casa Inteligente

T E S I S

PARA OBTENER EL TÍTULO DE
Maestro en Ciencias en Ingeniería de Manufactura
P R E S E N T A :

Ing. Octavio Brindis Pérez

DIRECTOR DE TESIS:
Dr. Joselito Medina Marín

CODIRECTOR DE TESIS:
Dr. Norberto Hernández Romero

Diciembre de 2013

Desarrollo de una Plataforma para la Definición de Reglas ECA en una Casa Inteligente

Resumen

El presente trabajo muestra la implementación de una plataforma computacional para proporcionarle comportamiento reactivo a una casa inteligente, mediante el uso de conceptos de bases de datos activas. El monitoreo de variables de ambiente de una casa o de un espacio habitado, se puede llevar a cabo con la utilización de sensores que muestren el estado en que se encuentran. Las señales detectadas por estos sensores sirven para conocer el estado en que se encuentra el espacio físico, y con base a eso realizar acciones en beneficio de las personas que se encuentren habitando el lugar. Además, si el valor que toma alguna variable de ambiente cae fuera de los rangos normales establecidos, o si alguna combinación de ellas producen un estado no deseado en la casa, entonces la plataforma computacional reacciona y ejecuta las acciones que se le hayan definido para mantener en un estado seguro el interior de la casa.

En este trabajo se desarrolló un circuito electrónico sobre una tarjeta de National Instrument PCI 6024E, y se le montaron sensores para monitorear temperatura, iluminación y humedad. En esta misma tarjeta se configuró un puerto de salidas digitales hacia los dispositivos que modifican las variables de ambiente, para que estos fueran controlados. Se desarrolló un programa en LabView para registrar los valores numéricos enviados por la señal de cada sensor, y esta información se enviaba al motor de reglas activas, desarrollada en Visual C++, mediante datos en archivos de texto. A partir de la información generada en el programa de LabView, el motor de reglas verificaba si alguna de las reglas definidas se disparaba, y en su caso se generaba el disparo de la acción, mediante el envío de una señal a LabView, y consecuentemente al circuito para llevar a cabo la acción física.

Con este trabajo se muestra que la utilización de los conceptos de bases de datos activas puede aplicarse en el desarrollo de casas inteligentes. A partir de esta plataforma, si se deseara incrementar las variables a ser monitoreadas bastaría con agregar los sensores correspondientes, sin necesidad de modificar la plataforma computacional.

Development of a Platform for ECA Rule Definition in a Smart Home

Abstract

This work shows the implementation of a computational platform for providing reactive behaviour in a smart home, using concepts of active databases. The monitoring of environment variables of a house or living space can be carried out through the use of sensors that show the house state. The signals detected by these sensors are used to determine the state of the physical space, and based on it some actions can be performed to help people who are inhabiting the place. In addition, if the value of any environment variable falls outside the established ranges, or any combination of them produce an undesired state in the house, then the computational platform reacts and performs the actions which have been defined to maintain a safe state inside the house. This paper presents an electronic circuit card development on National Instrument PCI 6024E, which was added with sensors to monitor temperature, lighting and humidity. In this card was configured a digital output port to the devices that modify the environment variables, in order to control the environment. A program was developed in LabView to record numerical values sent by the signal from each sensor; this information is sent to the active rules engine, developed in Visual C ++, storing data in text files. From information generated in the LabView interface, the rules engine checks if any of the defined rules is fired; and if it fires any rule, the action part is executed by sending a signal to LabView, and consequently the circuit performs the physical action. This work shows the application of active database concepts in the development of smart homes. In this platform, the addition of environment variables to be monitored would be sufficient by mounting the corresponding sensors and reactive rules, without modifying the computational platform.

Índice

Lista de figuras

Introducción

Antecedentes

Un Sistema de Base de Datos (SBD), tradicional realiza operaciones sobre los datos, a través de comandos o instrucciones definidas por el administrador de la Base de Datos (BD), o los usuarios de la misma, es decir, si un usuario de la BD desea agregar, modificar, eliminar o simplemente hacer una consulta a la BD, entonces debe ejecutar los comandos necesarios para llevar a cabo estas operaciones.

Sin embargo, existen sistemas en el mundo real donde, de acuerdo a lo que está ocurriendo en el universo de discurso, es necesario realizar una acción o procedimiento dependiendo de los cambios que hayan ocurrido, si de antemano se conocen las acciones que se deben ejecutar y el momento en que estas deben ejecutarse, entonces es factible definir una serie de procedimientos para incorporarle a la BD este comportamiento.
Los esquemas tradicionales de SBD no tienen la capacidad de soportar a estos sistemas en los cuales es necesario especificarle a la BD un comportamiento de reacción automática ante los cambios que se susciten dentro del universo de discurso de la BD, ya que de hacerlo un usuario humano es posible que incurra en errores en la detección de eventos.

Por esta razón surgieron los sistemas activos de bases de datos, mejor conocidos como Sistemas de Bases de Datos Activas (SBDA), los cuales deben ser capaces de detectar los eventos que ocurren en el universo de discurso, verificar el estado de la BD durante la ocurrencia de eventos, y además, ejecutar acciones o procedimientos dentro de la BD sin la intervención directa de un usuario humano.

Para proporcionar el comportamiento activo a una BD es necesario definir reglas que describan la reacción automática que debe seguir el manejador de la BD, también conocidas como reglas activas.

La forma más general para lograr la definición de las reglas activas es mediante la aplicación del modelo de regla Evento-Condición-Acción (regla ECA).

- On evento
- If condición
- Then acción

En la cual se define el evento que debemos detectar, la condición que debemos evaluar contra el estado que guarde en esos momentos la BD, así como la acción o grupo de acciones que se ejecutarán si la evaluación de la condición es verdadera. La mayoría de los sistemas comerciales (SYBASE, ORACLE, SQL Server, etc.[3], [4], [5],) ofrecen funciones para darle comportamiento activo a una base de datos, por medio de la definición de "triggers" o "disparadores".

El área de aplicación de un SBDA es muy amplia, desde sistemas de bases de datos administrativos, hasta sistemas de bases de datos de control en aerolíneas y de control de hospitales, así como monitoreo de transacciones financieras, identificación de actividades inusuales en el sistema, cumplimiento de restricciones de integridad, mantenimiento de datos derivados, generación oportuna de reportes, realización de procesos periódicos, entre otros. [2]

Existen diversos manejadores de Base de Datos Activa (BDA) que ofrecen una sintaxis para la definición de reglas ECA. Entre los que podemos mencionar se encuentra el Postgres, Ariel, Starburst, A-RDL, Chimera, HiPAC, Ode y algunos sistemas de BD comerciales como Oracle, Informix, Ingres, Rdb, Allbase/SQL e InterBase. [3]-[20]

Para el desarrollo de este trabajo de tesis, se utilizará esta tendencia de los sistemas de bases de datos, para ser aplicado dentro del área de Inteligencia Ambiental y Casas Inteligentes.

Se han publicado varios libros relacionados con Casas Inteligentes en donde se presentan nuevas tecnologías y aplicaciones de la vida diaria; la más común es a los cuidados de personas de la tercera edad tal como lo explican Augusto, Nugent y Martin (2008), donde se aborda la necesidad de análisis de datos en casas inteligentes para aumentar la autonomía personal y la incorporación de nuevo a la sociedad a las personas mayores con trastornos cognitivos en deterioro.

Posteriormente, los mismos autores escriben sobre la gestión de desastres en el cual este artículo presenta una arquitectura para ayudar a la toma de decisiones proceso de gestión de desastres. Mencionan que un aspecto central de este proceso que es la toma de decisiones en presencia de conflictos. Este problema se ejemplifica con tres escenarios simples relacionados a diversos contextos [21]-[24].

Justificación

La población del mundo está envejeciendo, en el año 2000 había 600 millones de personas con edad de 60 años en adelante; y se estima que habrá 1.2 billones en el año 2025 y 2 billones en el año 2059 [1]. La medicina, así como muchas otras áreas, han sido beneficiadas con el incremento en la disponibilidad de la información y el uso incremental de la tecnología, lo cual ha permitido un mejor manejo de los servicios de salud ofrecidos a las personas de edad avanzada.

Uno de los últimos desarrollos tecnológicos que ayudan a darles mayor independencia a las personas de la tercera edad es el uso de casas inteligentes (Smart Homes), las cuales se encuentran dentro de un área de desarrollo denominada Inteligencia Ambiental (AmI, Ambience Intelligence) [2].

Por otro lado, las bases de datos activas se han implementado para el desarrollo de sistemas reactivos, las cuales ejecutan acciones definidas en reglas que son habilitadas a partir de la ocurrencia de eventos en el propio sistema de BD, y a su vez con la evaluación de la condición, definida en la misma regla, contra el estado que guarde la BD. Esta parte conceptual de BDA puede aprovecharse para suministrarle el comportamiento reactivo necesario a una casa inteligente, la cual reaccionaría ante la ocurrencia de eventos de acuerdo a las variables de ambiente que se deseen monitorear.

Objetivos

Objetivo general

Desarrollar una herramienta computacional que permita la definición y análisis de reglas ECA (Evento-Condición-Acción), para ser utilizada en un prototipo de detección de eventos del ambiente y el encendido y apagado de modificadores de dicho ambiente.

Objetivos específicos

Los objetivos específicos que se realizaron son los siguientes:

1. Aplicar el modelo de reglas ECA, en el desarrollo de base de reglas orientadas a Casas Inteligentes.

2. Desarrollar un circuito que sea capaz de sensar de variables de temperatura, humedad, e iluminación en una casa inteligente.

3. Desarrollar la infraestructura para el detector de eventos que estará monitoreando su ocurrencia.

4. Llevar a cabo la implementación del kernel gestor de las reglas ECA, el cual coordinará la interacción entre el detector de eventos y la evaluación de la condición, así como la ejecución de la acción de la regla.

5. Integrar cada uno de los módulos correspondientes para proveer de comportamiento reactivo a la Casa Inteligente.

Planteamiento del problema

Las casas inteligentes son un tipo de sistema reactivo que se han estudiado para proveer de seguridad y confort a personas de la tercera edad que no pueden vivir de forma independiente sino que requieren del cuidado y ayuda de otra u otras personas para poder llevar a cabo sus actividades diarias.

Tomando en cuenta que el modelo de reglas ECA (evento condición acción) utilizado en el desarrollo de sistemas de BDA se aplica para proporcionar comportamiento reactivo a tales sistemas, es posible trasladarlo al desarrollo de casas inteligentes. En una casa inteligente existen variables de ambiente que se necesitan monitorear, donde el cambio en el estado de cada una de estas variables representaría un evento en el SBDA; la condición de la regla ECA se evaluaría contra el estado en que se encuentre la casa inteligente; finalmente, la acción de la regla ECA puede aprovecharse el envío de señales hacía los dispositivos que puedan modificar el estado de la casa inteligente, sea para encenderlos, apagarlos o regularlos.

En este trabajo de tesis se propone el uso del modelo de reglas ECA, como base para la definición de reglas orientadas al desarrollo de Casas Inteligentes, así como el

enfoque de BDA como mecanismo gestor de los eventos ocurridos al interior de la casa inteligente, de evaluación de las condiciones en el sistema, y ejecutor de las acciones definidas en la base de reglas. La definición de reglas utilizadas en Casas Inteligentes se realiza mediante el modelo de reglas ECA utilizando los sistemas de Base de Datos Activa.

Hipótesis

El desarrollo de Casas Inteligentes ha utilizado tecnología de punta, basada en la integración de equipos con la parte del desarrollo de software. Una Casa inteligente puede verse como un sistema reactivo que ejecuta acciones ante la ocurrencia de eventos.
Por lo tanto, la hipótesis que se plantea es: "Es factible el uso del modelo de reglas ECA y de los fundamentos de Base de Datos Activa en la definición de reglas de Casas Inteligentes.".

Método

Esta tesis se realizó aplicando la siguiente metodología:

1. Revisión de la literatura de las casas inteligentes y de bases de datos activas.

2. Implementación en Visual C++ de la interfaz que incluye el gestor de la base de reglas ECA

3. Desarrollo en PostgreSQL de la base de datos que almacena los registros con información de los valores obtenidos por sensores.

4. Desarrollo del detector de eventos en un circuito electrónico, controlado desde LabView, el cual obtiene información de los sensores.

5. Implementación de un circuito electrónico que ejecute las acciones indicadas por el gestor de reglas ECA.

6. Integración de cada uno de los elementos que conforman el prototipo.

Organización de la tesis

La tesis se desarrolla a lo largo de los siguientes capítulos estructurados de la siguiente manera:

- El capítulo 1 presenta los conceptos básicos dentro del área de Inteligencia Ambiental, así como la descripción de Casas Inteligentes. Además, se describen los elementos típicos que una casa inteligente puede contener.
- En el capítulo 2 se describen a los sistemas de bases de datos activas, y los modelos de conocimiento y ejecución que permiten proporcionar un comportamiento reactivo a los sistemas de bases de datos.
- En el capítulo 3 se desarrolla la explicación de la implementación del gestor de reglas activas, el cual fue desarrollado en Visual C++, y permite la comunicación con el detector de eventos implementado en LabView.
- En el capítulo 4 se describe la infraestructura del detector de eventos, el cual fue implementado en un circuito electrónico, y por medio de una tarjeta de adquisición de datos se manda la señal a Labview, para su correspondiente envío a la interfaz del gestor de reglas activas.

Capítulo 1

Fundamentos de Casas Inteligentes

1.1 Introducción

La tendencia en el desarrollo de la ciencia y la tenología es la integración de diversas áreas para lograr objetivos comunes. El desarrollo de casas inteligentes no es la excepción, ya que en este tipo de desarrollos se requiere de la participación de sistemas electrónicos, sistemas basados en conocimiento, y toma de decisiones basados en técnicas de inteligencia artificial. El estudio de las casas inteligentes forman parte de un área mayor denominada Inteligencia Ambiental.[35]

1.2 Inteligencia Ambiental

Hace algunos años surgío una nueva área de trabajo interdisciplinar, denominada Inteligencia Ambiental (AmI, Ambient Intelligence), también conocida como *entornos inteligentes*. Su gran relevancia reside en los importantes cambios que, a no muy largo plazo, implicarán sus resultados en la vida diaria de las personas. La visión de la Inteligencia Ambiental (AmI) consiste en la creación de espacios donde los usuarios interaccionen de forma natural y sin esfuerzo con los diferentes sistemas, gracias a que las tecnologías de computación y comunicación se convierten, en estos entornos, en invisibles para el usuario, al estar siempre presentes e integradas en los objetos cotidianos del mismo. La AmI ha sido descrita desde distintas perspectivas. Desde un punto de vista psicológico, la AmI puede definirse como el soporte eficaz y transparente para la actividad de los sujetos a través del uso de las tecnologías de la información y las comunicaciones. Otra definición, más tecnológica, la describe como: una inteligencia omnipresente y transparente en un entorno vigilado que soporta las actividades e in-

teracciones de los usuarios. Los fundamentos de la AmI surgen a partir del concepto planteado en 1991 por Weiser , denominado Computación Ubicua, que aprovecha los resultados obtenidos en el desarrollo de muchas otras áreas tecnológicas, entre las cuales caben destacar: las comunicaciones y dispositivos ubicuos, los interfaces multimodales de usuario, los sistemas de agentes artificiales, la domótica y la inteligencia artificial. Sin embargo, las tecnologías que realmente están siendo claves en su desarrollo de la AmI son principalmente tres: la computación ubicua, la comunicación ubicua y los interfaces de usuario inteligentes. [35]

La AmI está creciendo rápidamente como un enfoque multidisciplinario, el cual permite que muchas áreas de investigación tengan una influencia benéfica hacia nuestra sociedad. La idea básica detrás de la AmI es el enriquecimiento de un ambiente con tecnología (principalmente sensores y dispositivos interconectados a través de una red), un sistema puede construirse para tomar decisiones en beneficio de usuarios de ese ambiente basado en la recopilación de información en tiempo real y datos históricos acumulados. AmI tiene una relación decisiva con muchas áreas de ciencias de la computación. Las áreas más relevantes se muestran en la figura 1.1. Las redes de computadoras, sensores, interfaces, la computación ubicua o penetrante, y la Inteligencia Artificial son todas relevantes, pero ninguna de ellas cubre completamente a la AmI. Es decir, la AmI une a todos estos recursos para ofrecer servicios flexibles e inteligentes a usuarios actuando en sus ambientes. [36]

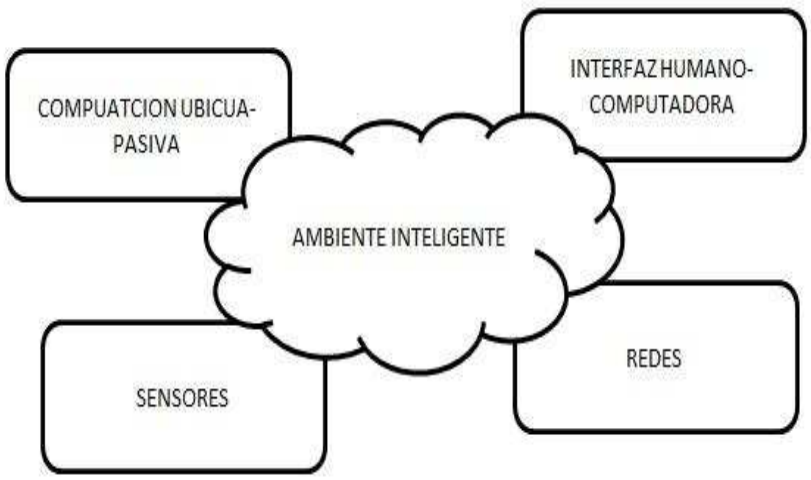

Figura 1.1: Relación entre la AmI y otras áreas

AmI se refiere a los entornos electrónicos que sean sensibles y receptivos a la pre-

sencia de la gente. El concepto de Inteligencia Ambiental fue propuesto en 1998 en una serie de talleres que se organizaron dentro de Philips, y que fueron comisionados por el consejo de la administración. [37]

Como Raffler sucintamente expresó , la AmI puede definirse como: "Un ambiente digital que le da soporte a la gente en su vida diaria de una forma no intrusiva". La AmI está alineada con el concepto de "cómputo desaparecido" ó "cómputo invisible", se menciona que "las tecnologías más profundas son aquellas que desaparecen. Se tejen así mismas en la fábrica de la vida diaria hasta que son indistinguibles". La noción de cómputo invisible está ligada directamente con la noción de cómputo ubicuo o cómputo penetrante , como lo llamó más tarde IBM. Algunos autores igualan los conceptos de cómputo ubicuo y cómputo penetrante con Inteligencia Ambiental, sin embargo, puede argumentarse que los sistemas ubicuos y penetrantes son en sí el conjunto de dispositivos que carecen de algo clave: el requerimiento explícito de "inteligencia".[2]

1.3 Casas Inteligentes

El término de casa inteligente lo podemos entender como una casa equipada con servicios avanzados para sus usuarios. Naturalmente, que tan inteligente debe ser una casa para ser catalogada como casa inteligente es un problema subjetivo. Por ejemplo, una habitación puede tener un sensor para indicar cuando su ocupante está dentro o fuera, y en este contexto mantener las luces encendidas o apagadas. Sin embargo, si los sensores solamente se basan en el movimiento y no en un sensor de posición, digamos por ejemplo que, la puerta puede detectar cuando una persona sale, pero si una persona se encuentra en la habitación leyendo algún libro, y mantiene una posición fija, el sistema puede confundirse y apagar las luces. Si el sistema confunde la ausencia de la persona con la ausencia de movimiento, entonces no lo podemos considerar como un sistema inteligente.

La tecnología disponible en nuestros días es muy rica. Muchos artefactos y artículos de una casa pueden ser enriquecidos con sensores para recolectar información acerca del uso de las instalaciones de la casa, y en algunos casos, actuar independientemente de la interacción humana. Algunos ejemplos de estos dispositivos son los aparatos electrodomésticos (estufas, refrigeradores), artículos del hogar (tapetes, camas, sofá), y dispositivos para adecuar la temperatura (aire acondicionado, calentadores). Los beneficios esperados de esta tecnología pueden ser: (a) incremento en la seguridad de la

casa (por ejemplo, monitoreando los patrones en el estilo de vida o las actividades más recientes de los usuarios y proporcionando asistencia cuando una posible situación de peligro se está presentando, (b) confort (por ejemplo, a justando la temperatura automáticamente), y (c) en la economía (por ejemplo, controlando el uso de las luces).

Existe una amplia gama de sensores y actuadores, desde aquellos que funcionan de manera independiente (por ejemplo sensores de humo), hasta aquellos que se encuentran embebidos en otros dispositivos (por ejemplo, en hornos de microondas, en camas), y también aquellos que pueden ser usados como prendas (por ejemplo, camisas que monitorean los latidos del corazón). En recientes aplicaciones se incluye el uso de Casas Inteligentes para proporcionar ambientes seguros a personas con necesidades especiales, y éstas puedan tener una mejor calidad de vida. Por ejemplo, en el caso de las personas que se encuentran en las primeras etapas de demencia senil (el caso más frecuente de los adultos mayores que sufren de la enfermedad de Alzheimer), el sistema puede ser hecho a la medida para reducir riesgos y garantizar cuidados apropiados en momentos críticos, mediante acciones de monitoreo, diagnosticando situaciones de interés y avisando a las personas que se encuentren a cargo.

Para definir el comportamiento de una Casa Inteligente, es factible la utilización del modelo de reglas ECA (evento-condición-acción), ampliamente utilizado en el desarrollo de bases de reglas para BDA.

El concepto de casas inteligentes juega un papel importante en la planificación del futuro de modelos de vivienda basadas en el cuidado. Más y más grupos de investigación están trabajando en este dominio, tal es el caso de MIT, Siemens, Cisco, IBM, Xerox, Microsoft, entre otros. De acuerdo a una casa inteligente debe contener tres elementos importantes como red interna, control inteligente y la casa automática. La red interna es la parte básica de la casa inteligente y puede ser alámbrica e inalámbrica. El control inteligente significa puertas de acceso a gestión de los sistemas, la casa automática significa productos dentro de la casa y enlaces a servicios y sistemas fuera de la casa.

Las Casas inteligentes, también conocidas como casas automatizadas, edificios inteligentes, casas con sistemas integrados o domótica, es un desarrollo de diseño reciente. Las Casas Inteligentes incorporan dispositivos comunes que cuentan con el control de la casa. Originalmente, el sistema de casa inteligente se utilizo para controlar los sistemas ambientales como la iluminación y la calefacción, pero recientemente el uso de tecnología inteligente se ha desarrollado de manera que casi cualquier componente eléctrico dentro

de la casa puede ser incluidos en el sistema.[23]

1.4 Elementos de una casa inteligente

El layout típico de una casa inteligente contiene sensores que ofrecen información acerca del estado que guarda la casa inteligente. En la figura 1.2 se muestra distribución de sensores en una casa inteligente.

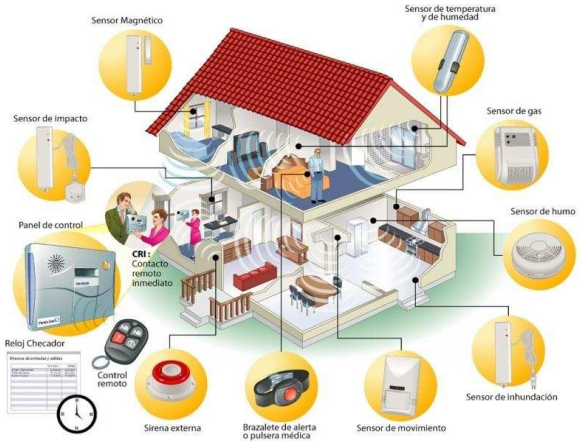

Figura 1.2: Distribución de los elementos de una casa inteligente.

Los elementos que mas comúnmente contiene una casa inteligente son los siguientes:

Sensores de ubicación: El ambiente tiene sensores de movimiento, los cuales tienen la habilidad de identificar el paradero de una persona (cocina, sala, sanitario, recámara, etc.)
Se asume que la persona debe usar una etiqueta electrónica que se comunique con los sensores ubicados en las puertas de cada cuarto o espacio, y poder monitorear su ubicación.
Alarmas de humo: El ambiente de la casa puede tener alarmas de humo en todos los cuartos, y detectar en el momento oportuno la presencia de un incendio.

Sensores / dispositivos: La casa inteligente debe controlar o monitorear el estado de los siguientes aparatos electrodomésticos que son comunes en los hogares:

1. Estufa

2. T.V.
3. Refrigerador
4. Llave de agua en la cocina
5. Llave de agua en lavabo de baño
6. Regadera
7. Aire acondicionado
8. Timbre de la puerta
9. Teléfono
10. Sensores de temperatura
11. Etc.

Todos estos dispositivos pueden estar en un estado de encendido o apagado. Sin embargo, existen lámparas que pueden regular la iluminación, o sistemas de aire acondicionado que permitan regular la temperatura ambiente.

1.5 Estado del Arte en el desarrollo de Casas Inteligentes

En [38] se propone el uso de una casa inteligente para darle asistencia a personas de la tercera edad durante la noche, para reducir la ocurrencia, y posibles efectos colaterales, de accidentes a este grupo de personas. Esta solución tecnológica incluye la visión nocturna, que proporciona información que es evaluada por un algoritmo basado en la noción de causalidad y razonamiento espacio-tiempo. El sistema alerta, explica, y asesora a los asistentes, evalúa situaciones, se anticipa a los problemas, y puede mejorarse para considerar el suministro de víveres a la casa.

Los autores de [39] proponen el uso de razonamiento temporal y el manejo de eventos complejos en la implementación de casas inteligentes. Mencionan que el manejo de razonamiento temporal es un punto clave en el desarrollo de estos sistemas inteligentes, y reportan la experiencia obtenida en el uso de BDA y razonamiento temporal.

El trabajo realizado en [40], muestra un sistema de casa inteligente aplicada al monitoreo de personas de la tercera edad con deficiencias cognitivas. En este trabajo, la casa contiene un sistema de monitoreo en red, el cual puede ser revisado y ajustado con base a la secuencia de actividades rutinarias que el o los habitantes de la casa realizan. Además, los asistentes en el cuidado de las personas de la tercera pueden ajustar los parámetros de las condiciones ambientales que deben existir al interior de la casa, y en consecuencia se ven involucrados en el manejo del sistema de monitoreo.

En [41] los autores presentan un trabajo acerca de la programación de sistemas de cuidado para el hogar, que pueda soportar la personalización, adaptabilidad a través del tiempo, y confiabilidad. Aplicaron un enfoque basado en políticas para construir su sistema, y muestran los detalles de la implementación que desarrollaron.

Por otro lado, en [42] se presenta el uso de la Inteligencia Ambiental en el desarrollo de una arquitectura para ayudar en el proceso de toma de decisiones a los administradores de desastres. Esta arquitectura consiste de una red de sensores y un procesador de información recopilada por los sensores. La red de sensores intenta recopilar la mayor cantidad de información como le sea posible, obteniendo señales de diferentes modalidades y complejidades. Estas señales pueden ser tan simples como señales de sonido o temperatura, o tan complicadas como secuencias de video e imágenes escaneadas. El procesador de información ayuda a recopilar estos datos y los filtra de los ruidos que pudieran contener en la señal, y que la información sea adecuada para una toma de decisiones correcta en el manejo del desastre.

En el trabajo [43] se hace énfasis en que el desarrollo de casas inteligentes requiere de la aplicación de conceptos como ingeniería de conocimiento y desarrollo de software. En este trabajo se desarrolla el análisis de los datos suministrados por una institución de cuidado de personas, y se implementó una interfaz computacional para simular el comportamiento de la casa inteligente, dependiendo de la ubicación de la persona en la casa y el estado de los dispositivos que pudiera encender y apagar como aparatos electrónicos, electrodomésticos, estufas, entre otros.

1.6 Conclusiones

En este capítulo se presentan los conceptos fundamentales que han sido desarrollados en la implementación de casas inteligentes. Se describe el concepto de la Inteligencia Ambiental (AmI), en la cual se mezclas diferentes áreas de la inteligencia artificial y

la interacción que existe entre ellas para formar la AmI. Además, como una de las aplicaciones de la AmI se encuentran las Casas Inteligentes, en las cuales se involucran los sensores, computación ubicua, redes de comunicaciones, interfaces con el usuario, y técnicas de inteligencia artificial.

Capítulo 2

Bases de datos activas

2.1 Introducción

Los esquemas tradicionales de sistemas de bases de datos fueron creados para almacenar información vital dentro de una organización. De esta manera, se representa una parte del mundo real que es importante conservar. Así, si existe algún cambio en la parte del mundo real que se tiene almacenada en el sistema de base de datos, éste también se refleja dentro del sistema de base de datos.

Sin embargo, algunas situaciones no pueden modelarse siguiendo este patrón. Por ejemplo, imaginemos el sistema que se tiene en la Central de Autobuses, en los días de vacaciones y días festivos existe mucha afluencia de personas para abordar autobuses y llegar a distintos destinos; por lo que es necesario agregar salidas de autobuses adicionales, de acuerdo a la cantidad de usuarios que arriben a la terminal. Si el número de usuarios sobrepasa un cierto umbral, entonces se programan más salidas de autobuses. El problema lo podemos atacar de dos maneras:

- Instalar en cada terminal de trabajo un módulo que verifique el número de usuarios; pero tendría que instalarse el mismo módulo en cada terminal, propiciando duplicidad en los programas.

- Otra manera sería hacer consultas periódicas a la BD, sin embargo, si se realiza una verificación del estado de la BD por períodos muy cortos se estaría consumiendo mucha potencia de cómputo. Por el otro lado, si se realiza una verificación del estado de la BD por períodos largos, podría ser demasiado tarde para llevar a cabo una acción, pueden perderse eventos que ocurran dentro del intervalo de

búsqueda y nuestro mecanismo de verificación no podrá detectarlos, además de que el orden de los eventos no es posible determinarlos.

En los enfoques descritos se presentan problemas que afectan completamente el desempeño de nuestro sistema, por tal motivo, es necesario implementar mecanismos que reaccionen automáticamente en tiempo y forma ante la ocurrencia de eventos en el interior y exterior de la BD, de ahí el surgimiento de los sistemas de bases de datos activas. Un sistema de Base de Datos Activa (SBDA), debe de ofrecer la funcionalidad de un SBD tradicional, además de proporcionar un comportamiento activo al SBD, es decir, además de definir el modelo de datos y el conjunto de programas que lo manipularán (esquema tradicional de BD), es necesario definir una base de reglas activas donde se especifique el comportamiento que esperamos sea realizado por el SBDA. A continuación se describen los conceptos fundamentales de un SBD, útiles para la comprensión de los conceptos de SBDA que más adelante se describen.

2.2 Sistema de base de datos

Un sistema de Base de Datos (SBD) se forma a partir de la unión de una base de datos (BD) ó repositorio de datos, y de un conjunto de programas que se encargarán de manipular a los datos almacenados en la BD. En la figura 2.1 se muestra el esquema de un SBD. Los SBDs fueron diseñados para manejar grandes volúmenes de información.

El manejo de datos involucra la definición de las estructuras donde serán almacenados los datos, así como de proveer mecanismos que manipulen eficaz y adecuadamente a éstos datos. Una BD no es más que el conjunto datos que tienen relación entre sí y mantienen un significado implícito. Como se mencionó anteriormente, en la BD se refleja una parte del mundo real, solo aquella parte que es de nuestro interes, llamada minimundo o universo de discurso. Si el estado del mundo real es modificado, tales modificaciones también son realizadas dentro de la BD, para mantener una coherencia de información entre el mundo real y el minimundo.

Los mecanismos de manipulación de datos, en lo que se refiere a la creación de la BD, las modificaciones, agregaciones, eliminaciones y acceso a los datos, se realizan mediante un programa o un conjunto de programas conocido como Sistema Manejador de Bases de Datos (SMBD).

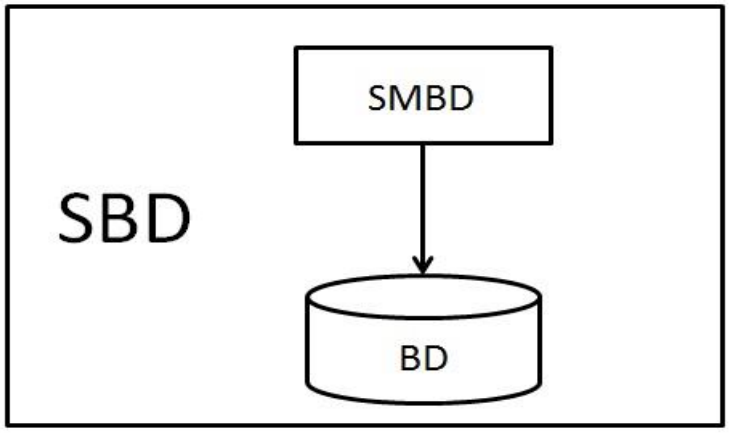

Figura 2.1: Esquema tradicional de un SBD.

2.3 Comportamiento Activo

Un SDBA es una extensión de un SBD "pasivo" con la posibilidad de especificar un comportamiento "reactivo". Parte fundamental en la especificación del comportamiento reactivo es la definición de una base de reglas activas, que proporcionen los mecanismos de reacción al SBD. En la figura 2.2 se presenta un esquema de la arquitectura general que un SBDA debe contener. Algunas de las aplicaciones de SBDA son [Paton]:

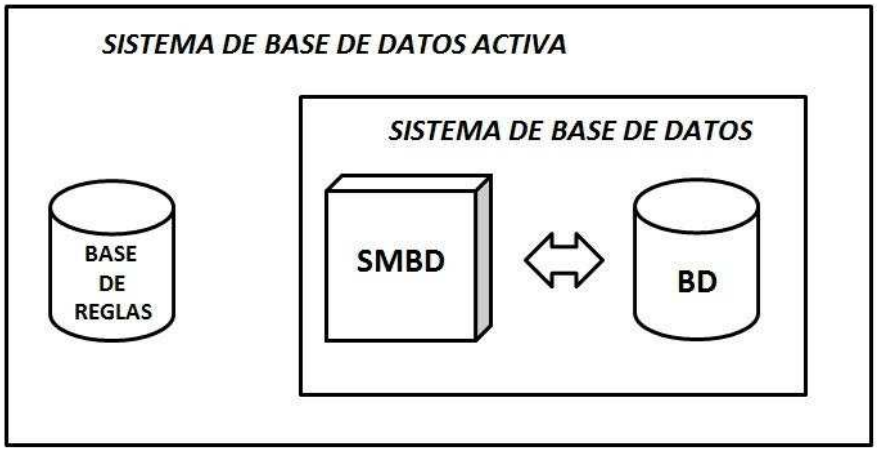

Figura 2.2: Arquitectura general de un SBDA.

- monitoreo de transacciones financieras,
- identificación de actividades inusuales en el sistema,
- cumplimiento de restricciones de integridad,
- mantenimiento de datos derivados,
- generación oportuna de reportes,
- realización de procesos periódicos,
- entre otras.

En vista de lo anterior, un SDBA debe ofrecer un *modelo de conocimiento* y un *modelo de ejecución* en el manejo de las reglas activas.

2.3.1 Modelo de conocimiento

El modelo de conocimiento del SBDA [5] indica al sistema la manera en que éste debe de comportarse ante la presencia de ciertos eventos relevantes y el estado que guarde la BD en el momento en que ocurran estos eventos [45]. El enfoque más utilizado para el modelo de conocimiento del SBDA es la definición de reglas Evento-Condición-Acción ó simplemente reglas ECA [44].

Regla ECA

La definición de la base de reglas ECA es la que provee de la funcionalidad reactiva al SBDA y está estrechamente ligada a la sintaxis del lenguaje de reglas que tenga el SBDA donde se deseen implementar. [45] Una regla ECA está formada por tres elementos: el evento, la condición y la acción. La forma general para representar a una regla ECA es la siguiente:

- on evento
- if condición
- then acción

El evento de la regla describe un suceso al cual la regla debe de ser capaz de responder. La condición de la regla examina el contexto de la base de datos en el cual el evento ocurrió, y finalmente, la acción de la regla describe la tarea que será llevada a cabo por la regla si el evento correspondiente ha tomado lugar y la evaluación de la parte condicional de la regla resulta verdadera. [44]

La mayoría de los SBDA soportan reglas ECA con sus tres componentes. En algunas propuestas de SBDA la parte de evento o de condición pueden ser omitidas o estar implícitas en los otros elementos de la regla. Si no se proporciona el evento de la regla, entonces estaríamos manejando una regla del tipo condición-acción, mejor conocida como regla de producción. Si la condición de la regla no es especificada, entonces se trata de una regla evento-acción, donde a partir de la ocurrencia del evento de la regla se ejecuta la acción especificada.

Evento
Un evento es algo que ocurre en un punto del tiempo. Por lo que la especificación de un evento involucra la descripción de un suceso que va a ser monitoreado. La naturaleza de la descripción y la manera en que el evento puede detectarse, depende en gran medida de la fuente ó generador de eventos.

Condición
La parte condicional de una regla es una expresión lógica la cual determina si la acción de la regla será ejecutada. La condición se evalúa contra el estado que guarde el estado de la BD en el momento en que el evento de la regla a ocurrido, si la evaluación de dicha expresión resulta verdadera entonces la acción ó conjunto de acciones especificadas en la acción de la regla serán ejecutadas.

En algunos sistemas, la especificación de la condición es opcional, formando de esta manera reglas de tipo Evento-Acción, donde a partir de la ocurrencia del evento se dispara la regla y la acción es ejecutada [44].

Acción
La acción de una regla ECA describe el conjunto de tareas que serán realizadas si la evaluación de la parte condicional de la regla resulta verdadera.

La acción de la regla puede ser la modificación de la estructura en la BD (operaciones de modificación, inserción ó eliminación de datos), el envío de una señal a un dispositivo externo a la BD, informar a un usuario o al administrador del sistema sobre

alguna situación que esté ocurriendo en el estado de la BD, abortar una transacción o bien, tomar alguna acción alternativa utilizando el comando *do-instead*.

La acción de la regla puede a su vez ser el evento que active alguna regla ECA o formar parte de un evento compuesto que active a una regla ECA.

Además de especificar en un SBA el modelo de conocimiento correspondiente a los elementos necesarios para especificar correctamente a cada uno de ellos, es de vital importancia saber la manera en que serán manejados en tiempo de ejecución, por lo cual se hace necesaria la definición de un modelo de ejecución.

2.3.2 Modelo de ejecución

Además de un modelo para la definición de cada una de los atributos y características con que deben contar los elementos de la regla ECA en el modelo de conocimiento, también se cuenta con un modelo de ejecución para el SBDA.

El modelo de ejecución es la manera en que la base de reglas ECA serán manejadas en tiempo de ejecución, es decir, como se detectarán los eventos para activar reglas, realizar la verificación de la parte condicional de las reglas activadas, así como la ejecución de las acciones dentro o fuera de la BD.

El modelo de ejecución está muy relacionado con cada SBDA, pero de manera general en todos los casos siguen las mismas fases en la ejecución de las reglas ECA [44].

La primera fase se refiere a la aparición de los eventos que dispararán a las reglas (*signaling*). En la segunda fase se toman los eventos encontrados en la fase anterior y dispara a aquellas reglas cuyo evento de la regla haya ocurrido (*triggering*). La tercera fase realiza la evaluación de la parte condicional de la regla (*evaluation*), formando un conjunto con aquellas reglas cuya evaluación de la condición haya resultado verdadera. En la cuarta fase (*scheduling*) se toma el conjunto de reglas formado en la tercera fase y se decide el orden en que serán procesadas. Finalmente, en la última fase (*execution*) se llevan a cabo las acciones correspondientes a las reglas disparadas. Figura 2.3. Las acciones ejecutadas en la última fase, pueden a su vez generar eventos a ser considerados dentro de la primera etapa, produciendo un disparo de reglas en cadena.

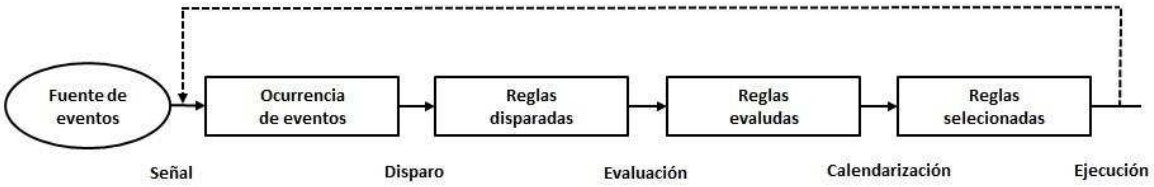

Figura 2.3: Secuencia de pasos del modelo de ejecución de reglas activas.

2.4 Conclusiones del capítulo

En este capítulo se abordaron los conceptos relacionados a los sistemas de bases de datos, a partir de su funcionalidad como repositorios de información. Además, se presentó un panorama acerca de la funcionalidad de los sistemas de bases de datos activos, los cuales pueden ser utilizados en situaciones donde se requiera de una reacción por parte del sistema de base de datos, cuando un evento, o una serie de eventos, de importancia ocurran en el mismo.

Capítulo 3

Implementación del Gestor de Reglas Activas

3.1 Introducción

Para poder realizar la definición de las reglas ECA, es necesario contar con un gestor de reglas activas, el cual se encarga de administrar el disparo de las reglas, tomando en cuenta la ocurrencia de eventos, la habilitación de las reglas de acuerdo a los eventos que sucedan, evaluar las reglas, y, en su caso, ejecutar la acción que tenga definida la regla. En este capítulo se describe la estructura de la implementación del gestor de reglas ECA utilizado en este trabajo de tesis.

3.2 Estructura de la plataforma computacional.

Se llevó a cabo la implementación de una interfaz de usuario, donde es posible definir reglas de tipo ECA para estipular el comportamiento que la casa inteligente debiera tener, ante la ocurrencia de ciertos eventos. La plataforma de esta interfaz se muestra en la figura 3.1.

Los eventos que ocurren en el medio ambiente son monitoreados mediante sensores. Los sensores envían la señal detectada a través de un circuito electrónico, y se recibe en la interfaz implementada en *LabView*. A su vez, *LabView* genera esta información en formato numérico para que la interfaz desarrollada en Visual C++ interprete estos datos, y los compare con los rangos establecidos en la base de reglas ECA. Si alguno de estos datos de entrada provoca el disparo de alguna de las reglas, la interfaz en Visual

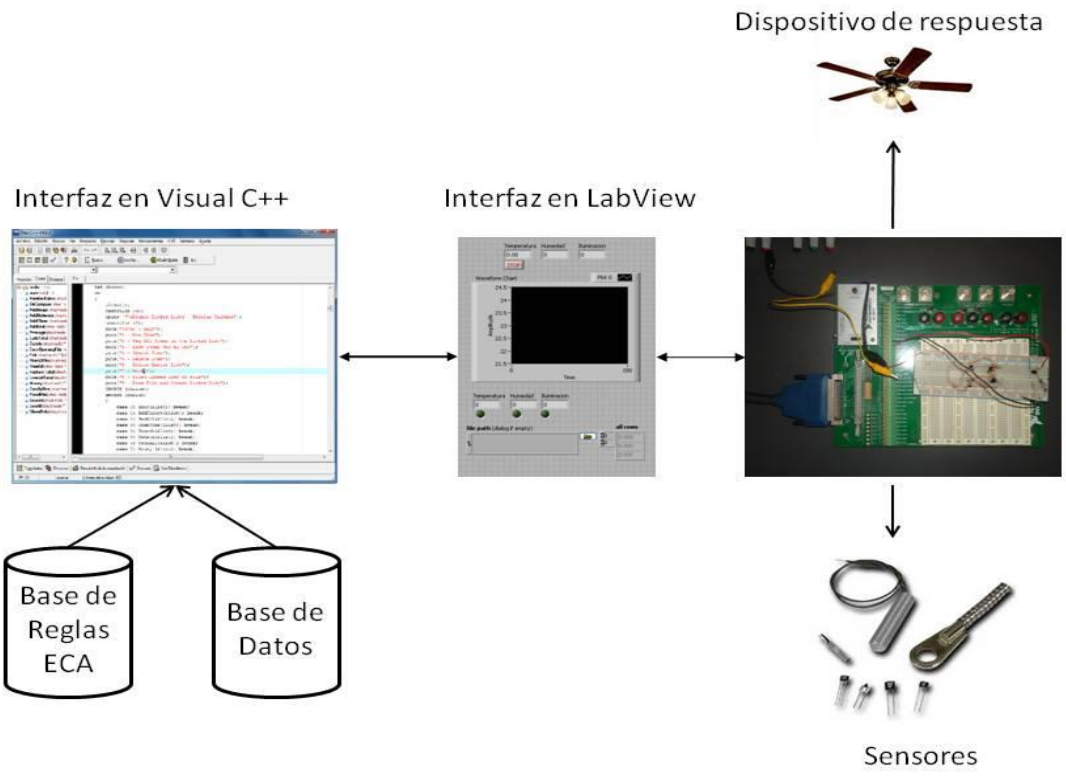

Figura 3.1: Plataforma para proporcionar comportamiento reactivo a una casa inteligente.

C++ envía la orden a la interfaz en LabView para que se ejecute la acción correspondiente. Esta interfaz genera la señal correspondiente para que se encienda (o apague) el dispositivo que se indicó de acuerdo a la regla ECA.

3.3 Interfaz en Visual C++

Esta interfaz se utiliza para establecer comunicación con la base de datos desarollada en *Postgresql*, y con la base de reglas ECA. Además de mantener comunicación con el programa de Labview en cuanto a la lectura de los valores obtenidos por los sensores.

Esta interfaz fue programada con base a la programación orientada a objetos, tomando para ello el lenguaje de programación de C++, en el ambiente de Visual Studio 2005. A continuación se describen las clases que se programaron en la interfaz gráfica y de administración de las reglas activas.

3.3.1 Clase: CondicionECA

Con esta clase se crean instancias para manipular la parte condicional de la regla ECA.

Atributos:

Operador: en él se almacena el operador que será utilizado para evaluar la condición de la regla ECA.

Valor: es utilizado, en conjunto con el operador, para llevar a cabo la evaluación de la regla, y determinar si la condición se cumple o no.
Constructores:

*CondicionECA(char *O, int valor)*: Crea un objeto de la clase a partir de un operador y un valor.

Métodos:

*char *devuelveOperador()*: Retorna el valor almacenado en el atributo Operador.

int devueveValor():Retorna el valor almacenado en el atributo Valor.

3.3.2 Clase: conectaBD

Esta clase es la aplicación principal por donde inicia el código, estableciendo conexión con la base de datos para el envío de los valores de variables, la revisión del estado de los sensores así como la ejecución del motor de reglas ECA.

3.3.3 Clase GeneraEntrada

En esta clase se crea una entrada del archivo .CSV, el cual es creado por LabView, el cual contiene los valores de las variables Temperatura, Humedad e Iluminación.

Atributos:

T: en el se almacena los datos obtenidos por LabView, que corresponden al sensor de temperatura almacenados en el archivo csv.

H: en el se almacena los datos obtenidos por LabView, que corresponden al sensor de humedad almacenados en el archivo csv.

I: en el se almacena los datos obtenidos por LabView, que corresponden al sensor de iluminación almacenados en el archivo csv.

Constructores:

GeneraEntrada(int T, int H, int I): Crea un archivo en formato csv, para identificar el estado en el que se encuentran las variables para temperatura (T), humedad (H) e iluminación (I).

3.3.4 Clase: Leer

Clase dedicada leer los datos de las variables de temperatura (t), humedad (h) e iluminación (i), los cuales son actualizados mediante la ejecución de LabView.

Atributos:

archivo: en él se almacena el nombre del archivo generado por LabView

t: en él se almacenan los datos de temperatura obtenidos de LabView con el sensor de temperatura.

h: en él se almacenan los datos de humedad obtenidos de LabView con el sensor de humedad.

i: en él se almacenan los datos de iluminación obtenidos de LabView con el sensor de iluminación.

Constructores:

*Leer(char*a)*: toma el nombre del archivo donde queda almacenado.

Métodos:

void Revisa(void): revisa los valores obtenidos del archivo de LabView.

int obtenerT(void): obtiene el valor de la temperatura almacenado en T.

int obtenerH(void): obtiene el valor de la humedad almacenado en H.

int obtenerI(void): obtiene el valor de la iluminación almacenado en I.

3.3.5 Clase LeeReglas

Esta clase se utiliza para leer el archivo de reglas ECA las cuales se encuentran almacenadas en un archivo de texto, además lleva a cabo la evaluación de las reglas de acuerdo a la ocurrencia de eventos.

Atributos:

*ReglasECA *Reglas[n]*: en este atributo se representa la base de reglas ECA, almacenadas como un arreglo de objetos de la clase ReglasECA; las cuales fueron leídas a partir de un archivo de texto. Además, se utiliza para evaluar las reglas disparadas.

Constructor:

*LeeReglas(char *a)*: crea una instancia de la clase LeeReglas, y recibe como argumento el nombre del archivo donde se encuentran almacenadas las reglas en formato de texto.

Métodos:

*char *evaluaReglas(char *e,int v)*: este metodo se ejecuta con base a la ocurrencia de eventos, y evalua el valor del evento contra las reglas ECA almacenadas en el atributo Reglas[n]. si la evaluación de la regla regresa un valor verdadero , el metodo devuelve la accion que debe ejecutarse.

3.3.6 Clase: ObtConBD

Establece la conexión con el sistema de base de datos, lo cual permite el envío de datos hacia la base de datos, así como la recuperación de información almacenada en ella.

Atributos:

SQLHENV henv: utilizado para almacenar el estado que guarda el sistema

SQLHDBC hdbc: se utiliza para establecer la conexión con el sistema de base de datos indicando el servidor de base de datos, usuario y el nombre de repositorio de datos.

SQLHSTMT hstmt: se utiliza para almacenar la cadena de texto que contendra la instrucción de SQL que se ejecuta en el sistema de base de datos.

SQLRETURN retcode: utilizado para almacenar el valor que devuelven las funciones ejecutadas en la conexiń con la base de datos.

Constructor:

ObtConBD(int l, char IPServer[ ], char cadsql[ ]): crea una instancia de la clase ObtConBD, a partir de los argumentos del servidor de base de datos (IPServer), y la cadena SQL que sera ejecutada en la base de datos (cadsql).

3.3.7 Clase: registro

Esta clase se encarga de registrar los valores de temperatura, humedad e iluminación en la base de datos; los valores que registra los recibe como argumentos en el constructor.

Atributos:

*char *nomTabla*: contiene el nombre de la tabla donde se almacena los valores de temperatura, humedad e iluminación.

*char *Tiempo*: contiene el valor de la hora en la que se envían los valores a la base de datos.

int Temperatura: contiene el valor para la temperatura que se envía a la base de datos.

int Iluminacion: contiene el valor para la iluminación que se envía a la base de datos.

intHumedad: contiene el valor para la iluminación que se envía a la base de datos.

Constructores:

registro(void): crea una instancia de la clase sin recibir argumentos, asignando valores por default a los atributos de la clase.

*registro(char *nomTabla, char *Tiempo, int Temperatura, int Iluminacion, int Humedad)*: crea una instancia de la clase a partir de los valores dados por los parametros nomTabla1, Tiempo1, Temperatura1, Humedad1 e Iluminacion1, asignados a los atributos nomTabla, Tiempo, Temperatura, Humedad e Iluminacion, respectivamente.

Métodos:

void enviaBD(void): con este metodo se genera la cadena de texto con la instrucción de SQL, utilizando los valores de los atributos, para insertar el registro en la tabla correspondiente en la base de datos.

3.3.8 Clase: ReglaECA

Atributos:

*char *Evento*: contiene la descripción del evento definido en la regla ECA. Los valores utilizados son para el cambio reportado en los valores de temperatura, humedad e iluminación.

*char *Condicion*: contiene la parte condicional de la regla donde se almacena la variable y el valor de comparación.

*char *Accion*: en este atributo se describe la acción que se ejecuta si la parte condicional de la regla se cumple. Tales acciones podran encender o apagar los dispositivos que regulen las variables consideradas.

*CondicionECA *Con*: este atributo es una instancia u objeto de la clase CondicionECA, y es utilizada para realizar la evaluación de la parte condicional.

Constructores:

ReglaECA(void): crea una instancia de la clase utilizando valores por default para los atributos que componen a la regla ECA.

*ReglaECA(char *e, char *c, char *a)*: crea una instancia de la clase a partir de los valores para evento condición acción en forma de cadena de texto.

*ReglaECA(char *e, CondicionECA *c, char *a)*: crea una instancia de la clase tomando el evento y la acción como cadenas de texto, y la parte condicional se forma aprtir de un objeto de la clase CondicionECA.

Métodos:

*void modificaEvento(char *e)*: se utiliza para modificar el evento de la regla ECA.

*void modificaCondicion(char *c)*: se utiliza para modificar la condicion de la regla ECA.

*void modificaAccion(char *a)*: se utiliza para modificar la acción de la regla ECA.

*char *obtieneEvento(void)*: devuelve el valor del evento para el objeto de la clase ReglaECA instanciado.

*char *obtieneCondicion(void)*: devuelve el valor de la condición para el objeto de la clase ReglaECA instanciado.

*char *obtieneAccion(void)*: devuelve el valor de la acción para el objeto de la clase ReglaECA instanciado.

*CondicionECA *obtieneCon(void)*: devuelve el valor de la parte condicional de la regla como un objeto de la clase CondicionECA.

3.3.9 Clase: MainFrm

Atributos:

int T: almacena el valor del estado de la temperatura, el número 0 indica que el dispositivo para regular la temperatura esta apagado y un 1 indica que esta encendido.

Int H: almacena el valor del estado de la humedad, el número 0 indica que el dispositivo para regular la humedad esta apagado y un 1 indica que esta encendido.

Int I: almacena el valor del estado de la iluminación, el número 0 indica que el dispositivo para regular la iluminación esta apagado y un 1 indica que esta encendido.

int prevT: este atributo es utilizado para ir almacenando el valor previo de la temperatura, para poder compararlo con el valor actual y determinar si se dio un cambio en el valor de la temperatura.

intprevH: este atributo es utilizado para ir almacenando el valor previo de la humedad, para poder compararlo con el valor actual y determinar si se dio un cambio en el valor de la humedad.

intprevI: este atributo es utilizado para ir almacenando el valor previo de la iluminación, para poder compararlo con el valor actual y determinar si se dio un cambio en el valor de la iluminación.

*LeeReglas *lr*: almacena la base de reglas ECA como una instancia de la clase LeeRglas, la cual es monitoreada para identificar las reglas disparadas a partir de la ocurrencia de eventos.

Metodos:

void OnTimer(UINT nIDEvent): este método es empleado para dar un lapso de tiempo en la actualización de los datos.

void OnBdConecta(): es utilizado para hacer la conexión a la base de datos con los valores actualizados.

void OnBdTermina(): es uilizado para finalizar la conexión con la base de datos.

El diagrama de clases se muestra en la figura 3.2.

3.4 Desarrollo de la BD

La información obtenida por los sensores se fue registrando en una base de datos desarrollada en *Postgresql*. A esta base de datos se le denominó *casa*, y se introducían datos desde la interfaz de Visual C++ mediante el código siguiente:

```
strcpy_s(instSQL,"insert into ");$
strcat_s(instSQL, nomTabla);$
strcat_s(instSQL," values (\'");$
strcat_s(instSQL, Tiempo);$
strcat_s(instSQL,"\',");$
sprintf(cad, "%d",Temperatura);
```

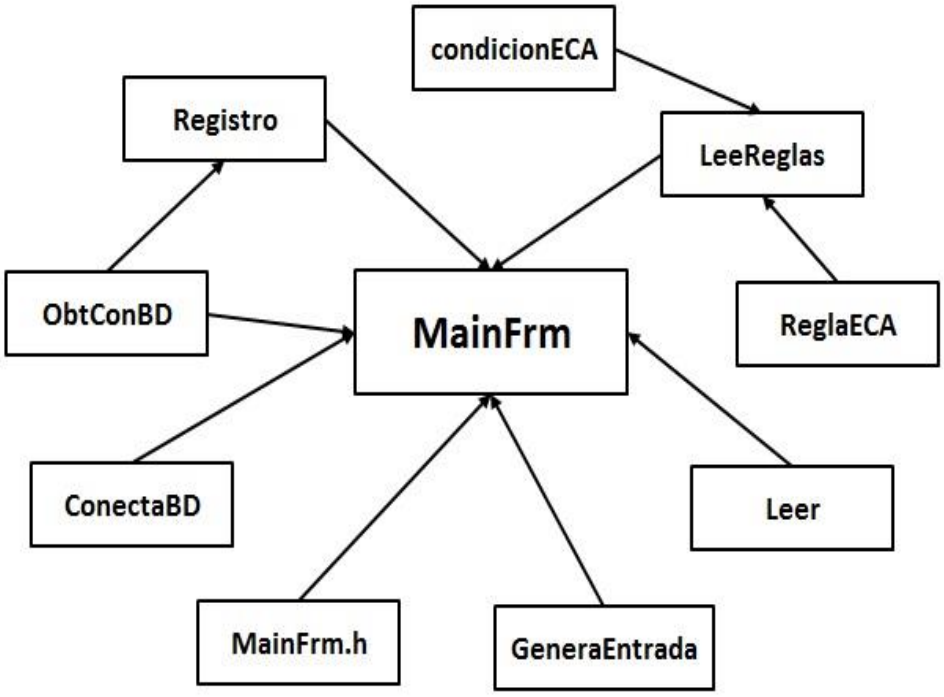

Figura 3.2: Diagrama de clases

```
strcat_s(instSQL, cad);
strcat_s(instSQL,",");
sprintf(cad, "%d",Iluminacion);
strcat_s(instSQL, cad);
strcat_s(instSQL,",");
sprintf(cad, "%d",Humedad);
strcat_s(instSQL, cad);
strcat_s(instSQL,");");
printf("instSQLccc %s\n",instSQL);
```

La estructura de la tabla datos que forma parte de la base de datos casa se muestra en la figura 3.3.

Los campos que se definieron fueron los siguientes:

- Tiempo
- Temperatura

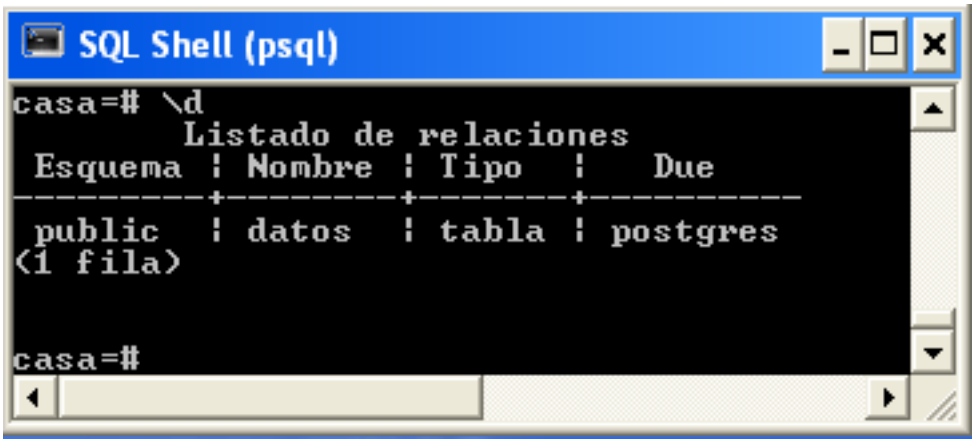

Figura 3.3: Estructura de la Base de Datos

- Iluminación
- Humedad

3.5 Comunicación con el detector de eventos

El gestor de reglas activas mantiene comunicación con el detector de eventos a partir de archivos de datos, los cuales son generados dentro del programa desarrollado en LabView. Estos archivos permiten al gestor de las reglas conocer el estado que guardan las variables que están siendo monitoreadas.

3.6 Conclusiones del capítulo

En este capítulo se muestra la estructura general que contiene el gestor de reglas activas. Se presenta un descripción de cada una de las partes que lo componen, partiendo de la base de reglas ECA, el motor que realiza la habilitación de las reglas, la BD que almacena los cambios en los estados de las variables, el detector de eventos implementado en LabView, y en el mismo los dispositivos que se encienden o se apagan, dependiendo de la acción que esté definida en la regla.

Capítulo 4

Infraestructura del detector de eventos

4.1 Introducción

El detector de eventos es parte esencial del gestor de reglas activas, ya que es el elemento que proporciona información sobre el valor o el estado que tendrán las variables que están siendo monitoreadas en la casa inteligente. El detector de eventos se describe en un capítulo separado, dado de que gran parte del trabajo de desarrollo de este trabajo de tesis se dedicó a la implementación del detector de eventos.

4.2 Descripción del circuito

El circuito está diseñado para sensar la temperatura, humedad e iluminación interna de la casa, pero también podría aplicarse en la industria, transporte público, hospitales, entre otras. En el desarrollo de este trabajo, el sensor de temperatura fue utilizado con otros componentes, tales como resistencias y diodos.

4.2.1 Circuito electrónico sensor de temperatura

Se compone de un sensor de temperatura LM35, cuya salida es linealmente proporcional a la temperatura en grados Celsius. Este sensor tiene la ventaja sobre los sensores calibrados en grados Kelvin ya que el usuario no requiere de restar una constante muy grande de voltaje de salida para obtener la escala en grados Celsius, el LM35

no requiere de ninguna calibración o recorte externo para dar una precisión de 1/4°C y a temperatura ambiente y 3/4°C sobre un rango de temperatura de -55 a 150°C, utiliza menos de 60mA de corriente, genera baja temperatura de funcionamiento 0.08°C en el aire, es de baja impedancia de salida 0.1Ω para 1mA de carga, el LM35 está herméticamente encapsulado para mejores resultado en la temperatura. Conectado a la fuente de 5V con una resistencia de 10K, de la misma forma se conecta la segunda terminal con dos diodos 1N914 conectados en serie hacia voltaje negativo, de igual forma esa es la salida de los datos que van conectados a la entrada de la tarjeta de National Instrument PCI 6024E de LabView, la tercera terminal va directo a voltaje negativo con una resistencia de 18K, como se muestra en la figura 4.1.

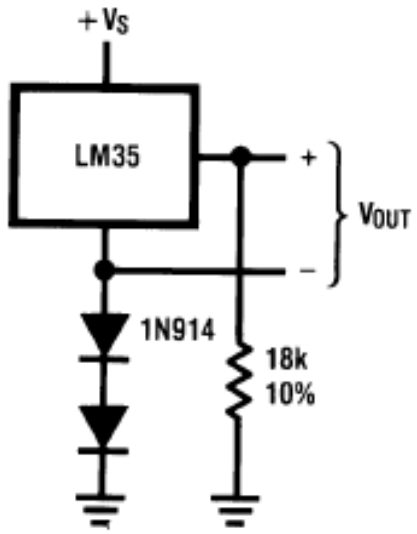

Figura 4.1: Conexión de los sensores

4.2.2 Circuito electrónico sensor de iluminación

La función de la fotorresistencia es detectar la luminosidad del medio, y de esta forma abrir o cerrar el circuito por medio de su resistencia, que varía dependiendo de la cantidad de luz. Este circuito es muy apreciable y utilizado en muchas aplicaciones, pero la más común y la mas ejemplar es la de el alumbrado público en las calles; cabe recordar que estas lámparas se alimentan por medio de la corriente de la red de suministro eléctrico, por lo que sería imposible controlar cada lámpara por medio de un simple

interruptor, además de que no siempre alguien se diera la tarea de encender todas las lámparas. Dependiendo si es día o noche, la lámpara automáticamente encenderá cuando comience a anochecer o en su defecto cuando amanezca se apagará. Esto tiene como consecuencia que se deje de desperdiciar energía por el día y así se evite estar pagando por algo que no se está utilizando.

Por tal motivo se usó la fotorresistencia, la cual se conectó a la alimentación de +5V directamente, en serie hacia -5V con una resistencia de 100K. La salida de los datos que se toman de la fotorresistencia son la entrada de la tarjeta de National Instrument PCI 6024E de LabView, para ser administrados por el programa que se desarrolló en LabView; según la luminosidad del ambiente esta cerrará o abrirá el circuito para dar la orden de encendido o apagado de la lámpara de la habitación, en la figura 4.2 se muestran detalles de conexión de la fotorresistencia.

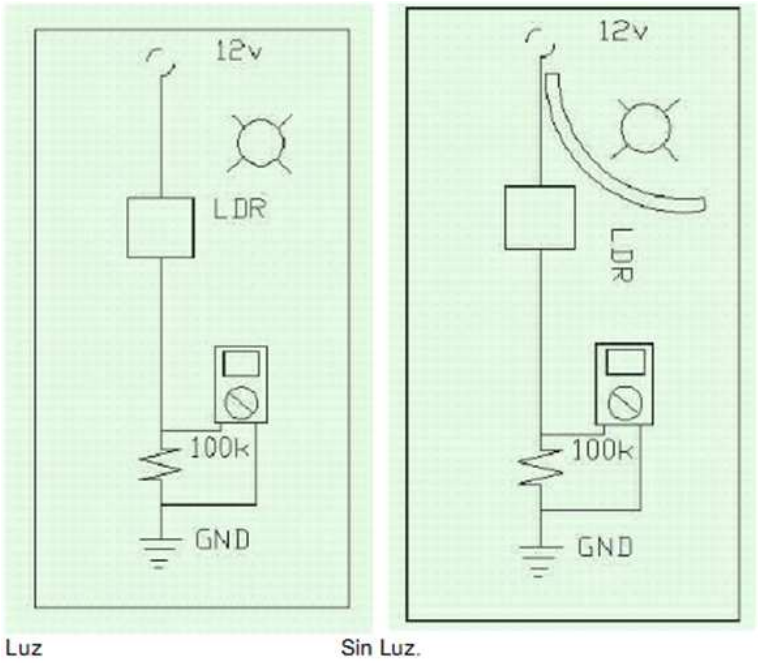

Figura 4.2: Conexión de las fotorresistencias

4.2.3 Circuito electrónico sensor de humedad

Este circuito visto en la figura 4.3, está diseñado para tomar lecturas de la humedad relativa del ambiente interno de las habitaciones de la casa, también es usado en industrias donde se requiera controlar la humedad, instrumentación meteorológica, instrumentos de medición de humedad relativa, etc.

Para el desarrollo de este trabajo de tesis, se utilizó el sensor 808H5V5, el cual es de bajo costo en comparación con otros, su estabilidad es a largo plazo, alta precisión (4RH para 25,30-80%HR,cuando el voltaje es de 5VCC), consta de tres patillas: la primera es el polo negativo (-), la segunda es la salida del dato y la tercera es el polo positivo (+). De esta forma se conectó polarizando con la primera y tercera patilla a una fuente de 5V con su respectiva resistencia de 10K, la segunda patilla se conectó a la entrada de la tarjeta de National Instrument PCI 6024E de LabView, para ser administrados por el programa que se desarrolló en LabView.

Basado en un voltaje de 5.0V DC, a 25°C en el ambiente.

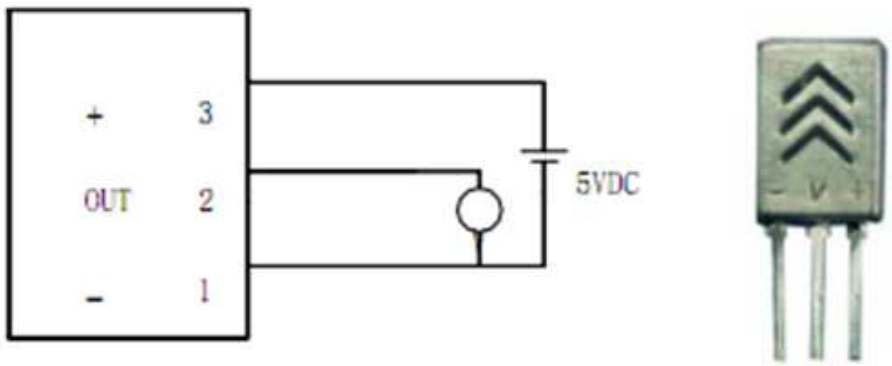

Figura 4.3: Conexión de sensor de humedad

4.3 Interfaz en LabView

En la figura 4.4 se muestra la interfaz que se implentó en Labview, cuya función es la de adquirir la información generada por los sensores descritos en la sección anterior. Además, muestra gráficamente el comportamiento de las variables sensadas. A partir

de la ejecución de reglas ECA, esta interfaz envía la señal para activar o desactivar los actuadores indicados en la regla ECA.

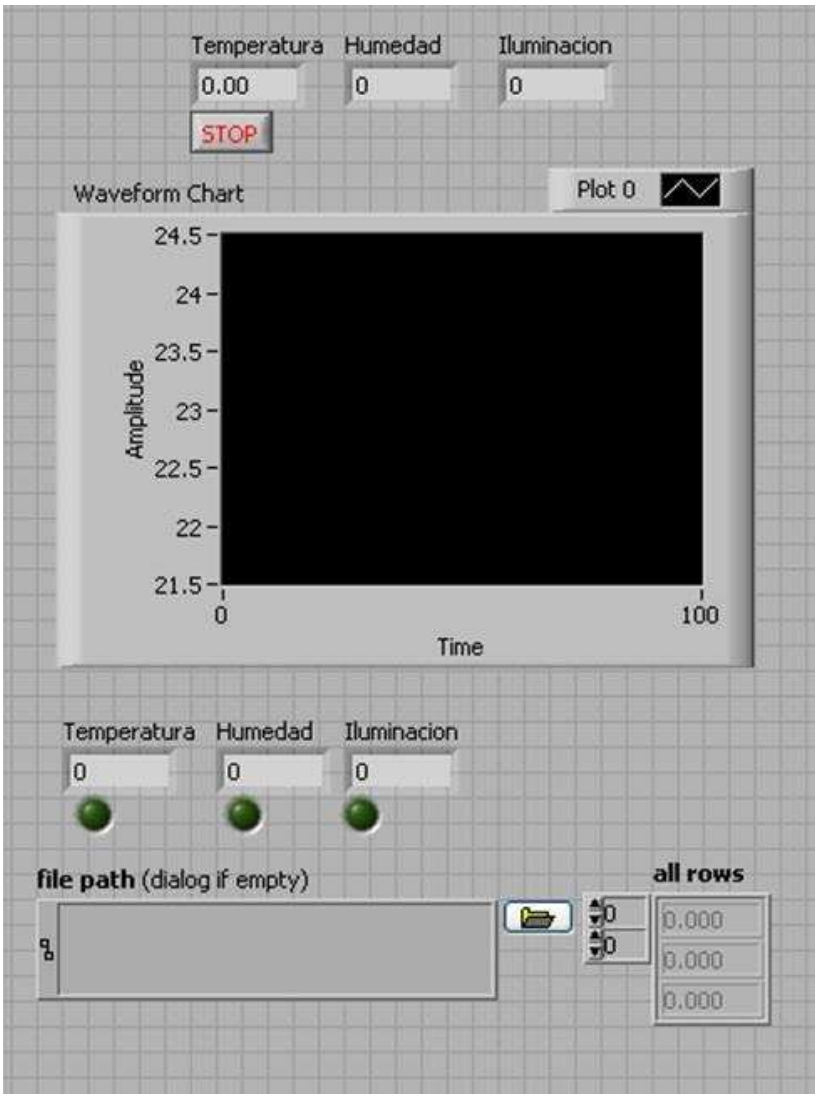

Figura 4.4: Interfaz de ejecución en Labview para el detector de eventos.

En la parte superior de la interfaz se incluyeron tres indicadores para mostrar los valores enviados por los sensores correspondientes a Temperatura, Humedad e Iluminación. En la señal generada por el sensor de Temperatura, se realizó una conversión

para ajustar los valores sensados a valores en grados centígrados. Los valores para Humedad e Iluminación se toman directamente de los que los sensores proporcionan.

También se incluyó un objeto para graficar el estado de las variables de Temperatura, Humedad e Iluminación, donde se puede observar el cambio que sufren estas variables con respecto a las condiciones del ambiente.

La parte inferior contiene indicadores que funcionan como alarmas, para indicar el momento en que la condición de una regla ECA se cumple. Cuando sucede que el valor de una variable cumple con la condición establecida, el led correspondiente se enciende.

También se incluye en la parte inferior un elemento que indica la ruta del archivo donde se almacenan los valores de los sensores.

Finalmente, en la parte inferior derecha se incluye un arreglo de tres elementos para indicar cuando un actuador se encienda (1), o se apague (0). Si la condición de una regla se cumple, y la acción de la misma indica que un actuador debe encenderse, entonces se asigna un valor de encendido (1), en el elemento correspondiente del arreglo, y este a su vez ejecuta la acción en el actuador.

4.4 Descripción del diagrama de bloques

En el circuito electrónico es ahí donde los sensores de temperatura, humedad e iluminación registran los datos que serán ocupados por el gestor de reglas mismos que LabView recolecta con la *DAQ*, que hace la interface para seguir el proceso de los datos los cuales son tres y ocupamos un divisor de señales para separarlas y poder trabajar cada una por separado, ya que los datos numéricos están en decimales, para esto ocupamos un multiplicador y una constante adecuada para obtener el valor real en los indicadores del panel frontal, también la gráfica estará llevando registro de las variables así como también se estará creando un archivo donde estarán todos los datos. Este archivo con extensión .lvm, será administrado por el algoritmo de C++, en el se encuentran las reglas ECA, las cuales con lo establecido por el usuario se crearán los parámetros que desee, otro archivo que *LabView* administrará con un *File Path* y otros elementos que harán posible las acciones de las reglas ECA.

Con el *File Path* se administra las acciones pues en conjunto con tres *Index Array*,

tomarán valores de 0 y 1 que el algoritmo de C++ ya habrá asignado según las condiciones del ambiente se encenderán o apagarán los leds de acuerdo al algoritmo de C++ para que con la segunda *DAQ* de salida haga la interfaz con los aparatos electrónicos tales como ventilador, humidificador o lámparas según sea el propósito.

Todo este diagrama estará gobernado por un *While Loop* con un rango de 5 min por evento, esto es que solo se obtendrán lecturas cada 5 min para ser administradas por los programas y obtener resultados, todo esto se resume en el diagrama de bloques ya concluido, la figura 4.5 lo muestra.

Esta aplicación no solo se emplea para casas, si no también se puede emplear para hospitales, transporte público, industria, entre otras, pues este es la base de la plataforma de reglas ECA, las cuales se pueden manipular según el propósito al que fuera asignado.

4.5 Conclusiones

En este capítulo se presentó una descripción de los elementos utilizados en la implementación del detector de eventos. Para llevar a cabo la detección de eventos de variables de ambiente es necesario contar con los sensores adecuados para cada una de ellas, y a su vez establecer la comunicación mediante una tarjeta de adquisición de datos. A partir de los datos recopilados se puede tener un panorama acerca del estado en que se encuentra la casa inteligente, y de esta forma estar en posibilidad de efectuar el disparo de las reglas ECA, dependiendo del estado que se presente.

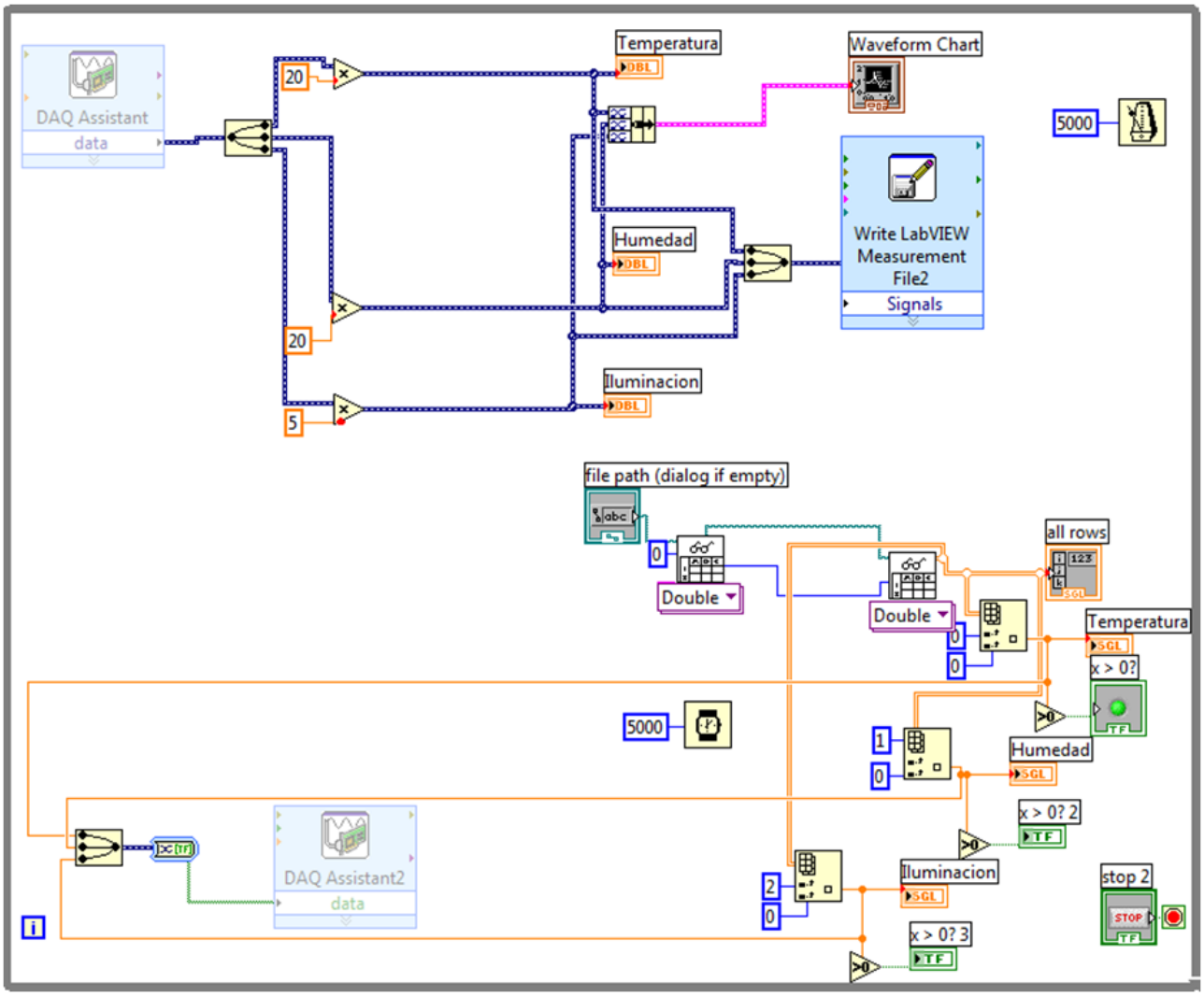

Figura 4.5: Diagrama de bloques terminado

Conclusiones

El desarrollo de casas inteligentes ha tomado importancia debido a la necesidad de independencia en la vida de las personas de la tercera edad o de aquellas que tengan alguna discapacidad física, y que no puedan vivir de forma independiente. En estos casos se requiere de la ayuda profesional de enfermeras, o de familiares que se ofrecen a cuidarlos.

Las Casas inteligentes, como parte de la inteligencia ambiental, toma en cuenta los avances que hay en el desarrollo de componentes electrónicos como sensores, redes de comunicación de datos, computación ubicua, y técnicas de inteligencia artificial para sistemas basados en conocimientos. Como parte de los sistemas basados en conocimientos se encuentran las BDA, en las cuales es posible definir comportamientos reactivos ante la ocurrencia de eventos que provocan la ejecución de acciones dentro del sistema de bases de datos.

La utilización de reglas ECA y de conceptos de BDA es factible para el desarrollo de casas inteligentes, ya que con esta estructura se puede tener un mejor control de los dispositivos que se encuentran involucrados en todo el sistema. Además, a partir de los sistemas de comunicación, la ejecución de la acción de una regla ECA se puede ver reflejada en el encendido o apagado de los dispositivos electrónicos que ayuden a controlar el ambiente de las habitaciones al interior de la casa.

En este trabajo de tesis, se llegó a la implementación de la plataforma sobre la cual se pueden definir reglas ECA, que puedan monitorear constantemente el estado en que se encuentre la casa y al mismo tiempo, poder realizar acciones que ayuden a la independencia en la vida de los ocupantes de la misma.

Se partió desde el desarrollo del circuito que contiene los sensores, los cuales monitorean el ambiente de la casa. Para este trabajo solamente se consideraron sensores de temperatura, iluminación y humedad. Sin embargo, si se desean agregar sensores para

otro tipo de variables de ambiente, no implicaría un trabajo mayor el hecho de agregarlo.

De la misma manera se implementó en el mismo circuito las salidas para el encendido y/ apagado de los dispositivos que pudieran controlar estas variables, tales como los focos de la casa, o un ventilador que mejore las condiciones de clima.

Mediante un programa desarrollado en LabView se procesan las señales obtenidas desde el circuito, y estas a su vez se analizan en el gestor de las reglas ECA, el cual podría indicar si las acciones, definidas en las reglas habilitadas, se disparaban o no.

El resultado de este trabajo de tesis puede aplicarse para poder hacer una implementación real de una casa inteligente, a partir de la definición de un conjunto de reglas apropiadas para el control y monitoreo de la casa, así como el incremento de los sensores necesarios en la instalación de la casa.

Trabajo futuro

Como parte del trabajo futuro se pretende llevar a cabo el análisis de conjuntos de reglas ECA aplicadas en casas inteligentes, para asegurar que su ejecución no provoque estados inconsistentes, así como evitar ciclos en el disparo de las reglas.

Por otro lado, tomando como base los resultados obtenidos en este trabajo de tesis, se pretende el desarrollo de un prototipo de casa inteligente, que involucre otras variables del medio ambiente como la ubicación de los ocupantes, o conocer el estado de encendido/apagado de los aparatos electrónicos y electrodomésticos que se encuentran en la casa.

Apéndice A

Programación del detector de eventos en LabView

A.1 Implementación en LabView

LabVIEW es un entorno de programación gráfica usado por miles de ingenieros e investigadores para desarrollar sistemas sofisticados de medida, pruebas y control usando íconos gráficos e intuitivos y cables que parecen un diagrama de flujo. Ofrece una integración con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos, todo para crear instrumentación virtual. La plataforma LabVIEW es escalable a través de múltiples objetivos y sistemas operativos, desde su introducción en 1986 se ha vuelto un líder en la industria.

LabVIEW es un lenguaje de programación gráfica que utiliza íconos en lugar de líneas de texto para crear aplicaciones. En contraste con la programación basada en texto y lenguajes de instrucciones para determinar la ejecución del programa, LabVIEW utiliza la programación de flujo de datos, en donde se determina la ejecución. Se construye una interfaz de usuario mediante un conjunto de herramientas y objetos, a esta interfaz del usuario se le conoce como panel frontal. Los objetos del panel frontal se controlan mediante código, utilizando representaciones gráficas de funciones. El diagrama de bloques contiene el código en su interior, y de alguna manera, el diagrama de bloques se asemeja a un diagrama de flujo.

LabVIEW se integra plenamente para la comunicación con el hardware, tales como GPIB, VXI, PXI, RS-232, RS-485, y el plug-in de dispositivos DAQ. LabVIEW también

tiene funciones integradas para la conexión de la aplicación para la Web con las normas de LabVIEW y el software de servidor Web, tales como TCP / IP la creación de redes y ActiveX.

En LabVIEW los programas son llamados instrumentos virtuales (VIs, en ingles). Los VI contienen tres principales componentes: el panel frontal, el diagrama de bloques, los iconos y los conectores del panel. El panel frontal se puede construir con controles e indicadores los cuales son terminales de entrada y salida. Los controles son perillas, push buttons, selectores y otros elementos de entrada. Los indicadores son gráficas, leds y otros displays. Los controles simulan dispositivos de entrada y proporcionan datos al diagrama de bloques del VI. Los indicadores simulan dispositivos de salida y muestran los datos que adquiere o genera el diagrama de bloques. El diagrama de bloques incluye terminales, subVIs, funciones, constantes, estructuras y redes las cuales transmiten datos al diagrama de boques. Después de construir el panel frontal y el diagrama de bloques, se ejecuta el VI y se verifica la existencia de errores.

Los programas de LabView son llamados Instrumentos Virtuales (Virtual Instruments) o VIs, porque su apariencia y operación imita a los instrumentos físicos, como Osciloscopios y Multímetros. Cada instrumento virtual usa funciones que manipulan entradas desde la interfaz de usuario o de otras fuentes y muestran esta información o la mueven hacia otros archivos u otras computadoras.

Un instrumento virtual contiene los siguientes componentes:

- Panel Frontal.- *(Front Panel)* Sirve como interfaz de usuario
- Diagrama de Bloques.- *(Block Diagram)* Contiene el código fuente gráfico del instrumento que define su funcionalidad.
- Iconos y herramientas de conexión.- Identifica al VI para que se pueda usar en otro VI. Un VI dentro de otro VI se denomina SubVI. Un SubVI corresponde a una subrutina en lenguaje de programación basado en texto.

Panel frontal.

Primero se abre un VI en la figura A.1 se muestra, esta opción se da al abrir el programa, aparecerá el panel frontal y el menú de controles, que es donde pondremos los indicadores y controles tales como los indicadores numéricos, la gráfica, el *file path*, el *all row* y el botón de *stop*.

En la figura A.2 se muestran los indicadores de temperatura, humedad e iluminación, los cuales reciben el dato real que se está sensando, va directamente conectado al ícono de la *DAQ Assistant*.

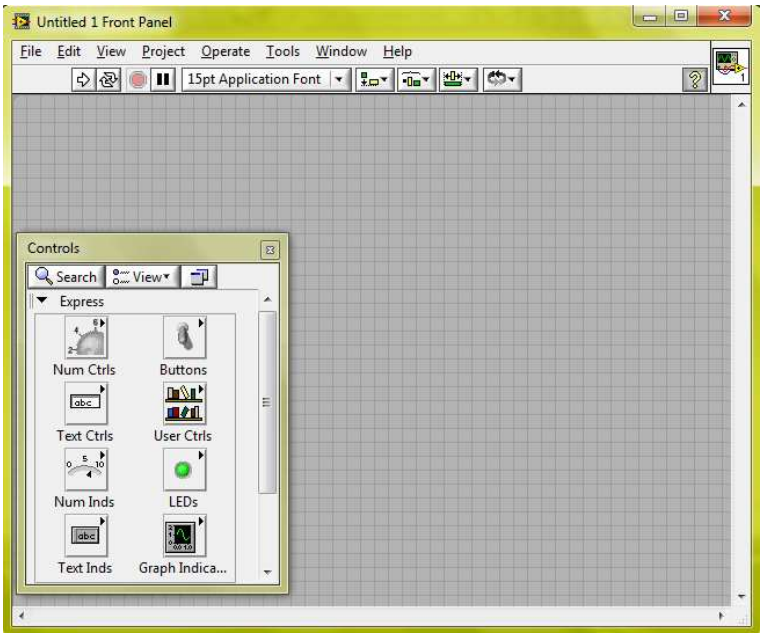

Figura A.1: Panel Frontal

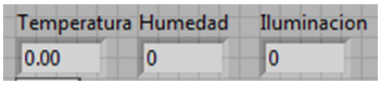

Figura A.2: indicadores

En la figura A.3 se muestra un graficador que presenta la frecuencia de cómo cambian los indicadores, a esta se le conectan los indicadores anteriores.

La figura A.4 muestra los indicadores con *led*, su función es la de recibir un 0 ó 1 según la decisión del algoritmo de C++ que es quien administra los datos recibidos de la tarjeta de adquisición de datos de *LabView*, su conexión se explica en el diagrama de bloques ya que van acompañados de otros iconos que no se muestran en el panel frontal.

La figura A.5 muestra el indicador de la ruta del archivo de texto el cual es generado por C++ con informacion correspondiente a las condiciones de los sensores, su conexión se explica en el diagrama de bloques ya que van acompañados de otros iconos que no se muestran en el panel frontal.

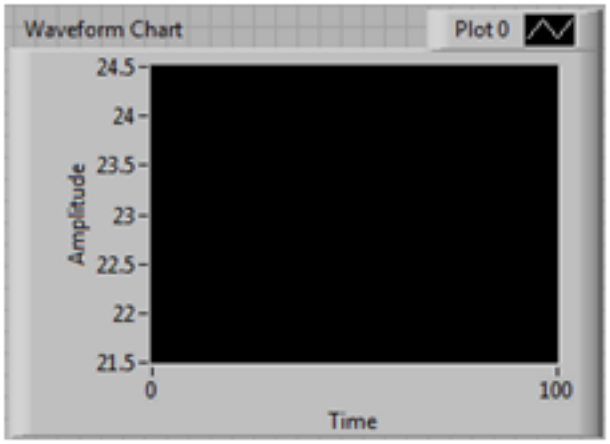

Figura A.3: Gráfica

Figura A.4: Leds indicadores

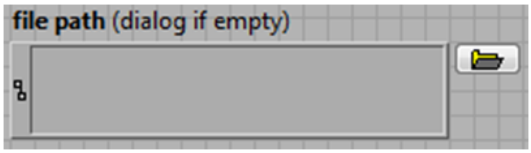

Figura A.5: Instrumento para proporcionar ruta de archivos

La función de el ícono de la figura A.6 es la de mostrar los ceros y unos que según los sensores y el programa en C++ generaran en un archivo en Excel con la extensión .cvs, esto es solo para cerciorarse de que cada programa haga su función, su conexión se explica en el diagrama de bloques ya que van acompañados de otros íconos que no se muestran en el panel frontal.

El botón que se muestra en la figura A.7 nos sirve para detener el programa de

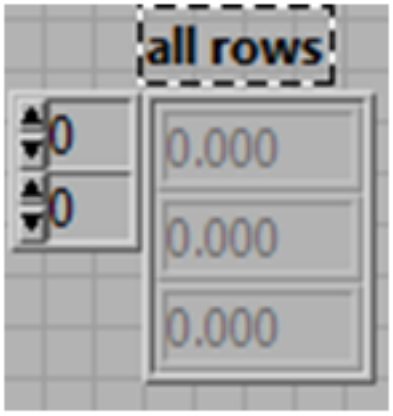

Figura A.6: Tabla indicadora

LabView.

Figura A.7: Botón de paro

En conjunto de todos estos iconos se resume en el panel frontal como se muestra en la figura A.8.

Diagrama de bloques.

Una vez abierto el panel frontal mostrado en la figura A.9, con las teclas *ctrl+T* se abre el diagrama de bloques y aparece por defecto el menú de funciones, en donde se hará toda la programación de *LabView* con diferentes funciones, para esto se emplean funciones como *DAQ Assistant*, *While Loop*, multiplicación, entre otros que a continuación se describen.

El *While Loop (WL)* figura A.10, se toma del menú de funciones y se arrastra hasta el diagrama, puede expandirse a la medida que se requiera dado que en su interior lleva todas las funciones para realizar el programa, este no lleva hilos de conexión solo el botón de *stop* y el botón de número de ejecuciones.

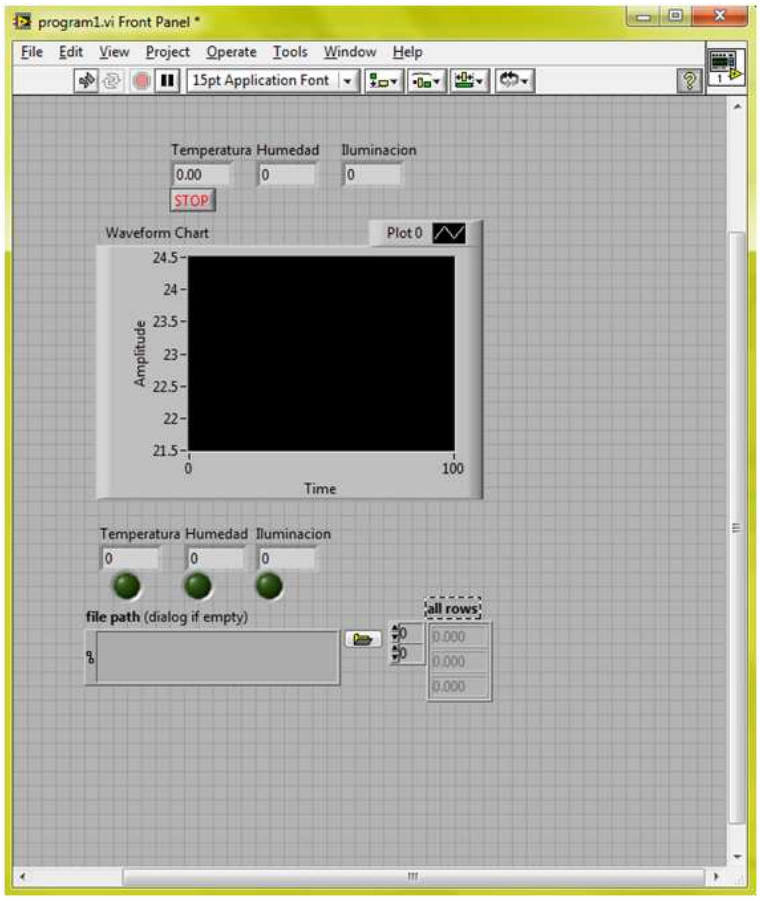

Figura A.8: Panel Frontal

El *DAQ Assistant* mostrado en la figura A.11 se encarga de hacer la digitalización entre los sensores del circuito y el ambiente de *LabView*, es la parte muy importante del programa y está localizado en el menú de funciones de programación, se arrastra hasta el interior del *WL*, se clickea dos veces para configurar la tarjeta de adquisición de datos. Al dar doble clic en el ícono de *DAQ Assistant* se abrirá automáticamente una página de la configuración como en la figura A.12, ahí se elige *Analog Input o Digital I/O*, según sea el caso (entrada analógica o salida digital), para este proyecto solo se utilizan esas dos opciones de las cinco que ofrece esta configuración.

Con la opción de *Analog Input*, se despliega un menú de medidas a elegir (figura A.13), en este caso se seleccionó voltaje posteriormente se selecciona el canal de la tar-

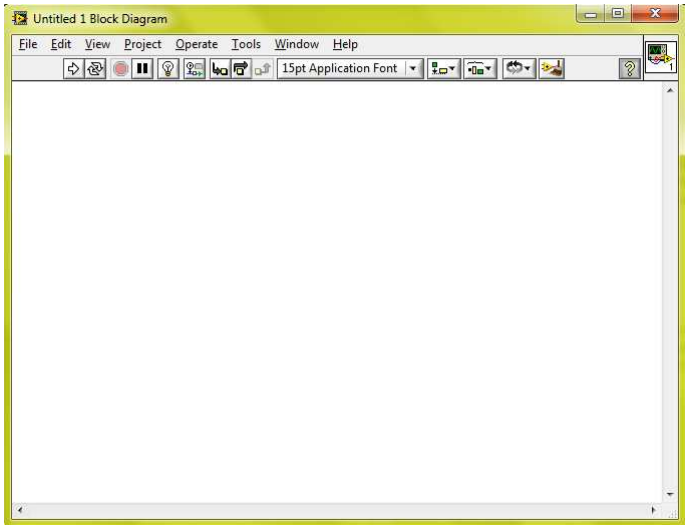

Figura A.9: Diagrama de Bloques

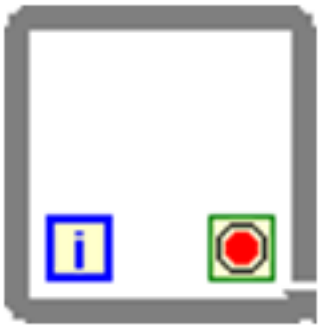

Figura A.10: While

jeta de datos.

Elegido el canal se oprime el botón de Finish, para dar por terminada la configuración y dar paso a la prueba de los canales y el voltaje.

En la figura A.14 se muestran los últimas configuraciones para el voltaje y los canales el número de repeticiones así como el tiempo de utilización de las pruebas, en la figura 4.17 es la prueba por medio de una grafica se muestra el voltaje que circula por los canales que se ocuparán.

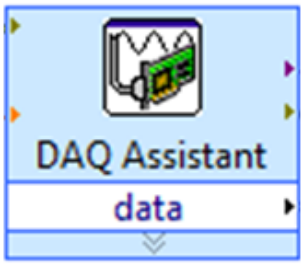

Figura A.11: DAQ

Figura A.12: Create New

La figura A.15 contiene un divisor de señales se encuentra en el submenú de *Signal Manipulation*, se usa para dividir las tres señales de los sensores de temperatura, humedad e iluminación, que provienen de la *DAQ* y las salidas van a un multiplicador.

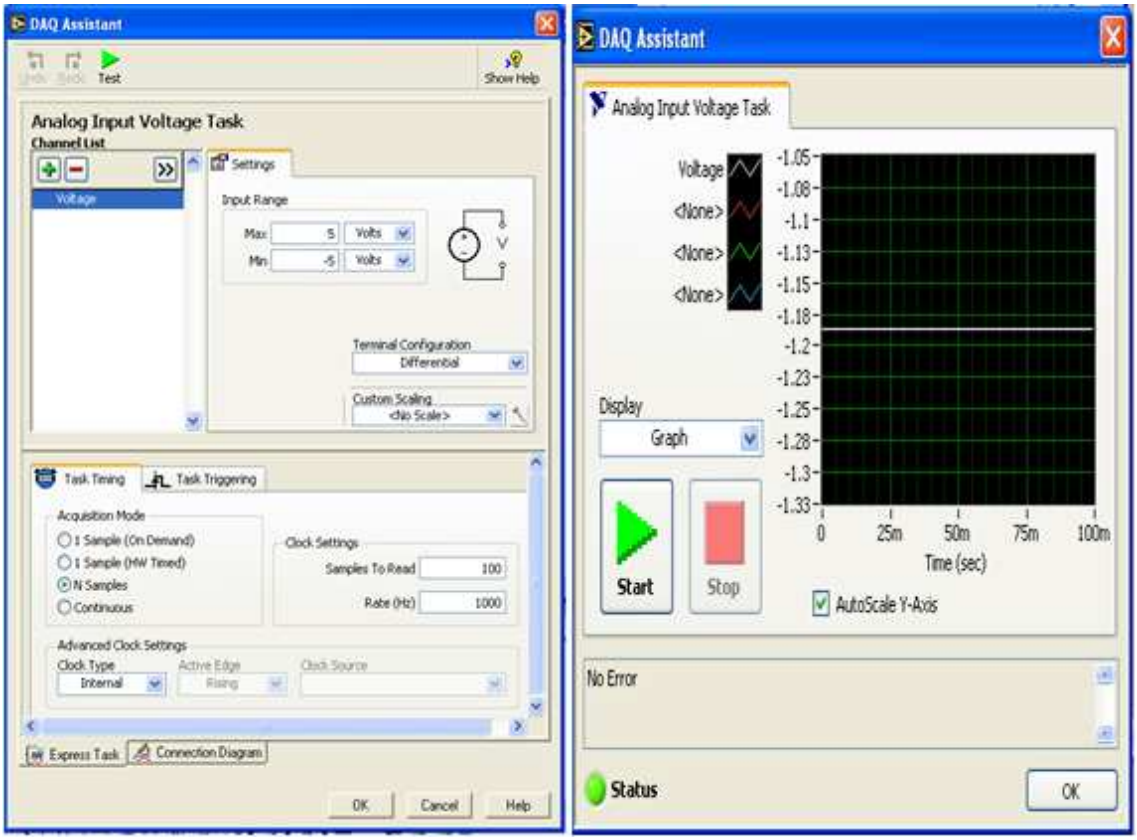

Figura A.13: Prueba de canales

Con el multiplicador de la figura A.16 y una constante le da un aumento al resultado de la señal para que sea real y visible, lo cual se aplica para las tres variables, este ícono se encuentra en el menú de *Mathematics* submenú *Numeric*.

la figura A.17 muestra un tramo de hilo de conexión, todos los iconos deben estar conectados, ya que si se encuentra uno perdido el programa no se ejecutará y este enviará una señal de error que a su vez lo muestra en una caja de diálogo.

La figura A.18 Muestra un indicador, este se obtine en el panel frontal, y en el diagrama de bloques se conecta a la salida del multiplicador, este indicador solo es de tipo entrada, así también para las tres variables este icono se encuentra en el menú del panel frontal.

Figura A.14: Configuración de DAQ

Figura A.15: Divisor de señales

Figura A.16: Operacional y constante

Figura A.17: Hilo de conexión

La figura A.19 contiene un *bundle*, el cual sirve para reordenar un grupo de datos, que estos son los que han obtenido los sensores, los datos ya ordenados en una sola

Figura A.18: Indicador de temperatura

señal van directamente a un *waveform chart*, que es donde estarán representados por una gráfica que nos mostrará la fluctuación de las señales, es solo de carácter informativo, este icono se encuentra en el menú de *programming*, submenú *Clusters, Class and Variant*.

Figura A.19: Bundle

El *Waveform Chart* (gráfica), de la figura A.20 nos muestra las seãles de temperatura, humedad e iluminación, es solo informativa, nos sirve para ver cómo se comportan las variables, este icono se encuentra en el menú del panel frontal en *indicators*.

Figura A.20: WaveForm Chart

La figura A.21 muestra un *Write LabView Measurement*, el cual tiene la función de ir generando un archivo de texto con los datos obtenidos a partir de las lecturas

de las tres variables consideradas en el proyecto.Este archivo es de extensión .lvm, y es utilizado para que el algoritmo de gestor de reglas ECA para verificar el dispar de las reglas. La configuracón de este elemento se lleva a cabo en el cuadro de diálogo de Propiedades. Para el propósito de este proyecto se configuro para que se guarde en un solo archivo, con el nombre temp.lvm y sobrescribirlo si ya existe uno con el mismo nombre. Los datos que se generaron también fueron guardados en la base de datos de *Postgres* para tener un historial.Este ícono se encuentra en el menú de *Express*, submenú *Output*.

Figura A.21: Write LabView Measurement

El *File Path* figura A.22, es parte del panel frontal, pero su conexión se realiza en el diagrama de bloques, puesto que es aquí donde aparece en forma de ícono. Este elemento se encuentra en el menú de *File I/O*, submenú *Path*; su función es la de direccionar el archivo va a leer el programa del proyecto, ya que puede haber varios programas, para este caso se va a elegir el archivo que está generando C++, los datos de este archivo se verán en la pantalla de en el panel frontal.

El *Read From Spreadsheet File (RFSF)* que se muestra en la figura A.23, se encuentra en el menú de *File I/O*, submenú *Path*,y se encarga de leer los datos de un archivo generado en *Excel*, lleva la constante de cero para que solo lea la fila cero (esto porque siempre interesan la primera fila ya que constantemente se genera información), su entrada está conectada al file path y su salida va conectada a otro *read from spreadsheet file*, el cual se encarga de organizar los datos en una sola columna, este segundo *RFSF* va conectado a un *All Rows*.

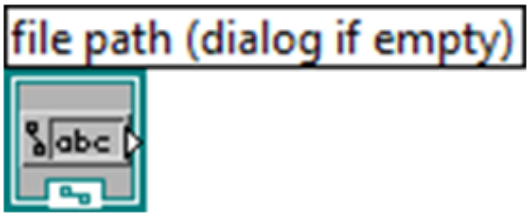

Figura A.22: File Path

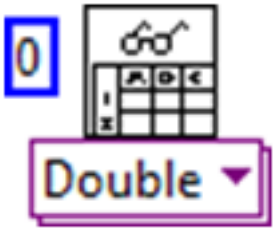

Figura A.23: Read From Spreadsheet File

En la figura A.24 se muestra un *All Rows* va conectado al *RFSF*; se visualiza en el panel frontal, pues es aquí donde se muestran los 1 ó 0's según las reglas ECA que el programa de C++ asignará a los valores de las variables.

Figura A.24: All Rows

Se muestra en figura A.25 un *Index Array*, el está ubicado en el menú de *Programming*, submenú *Array*, su función es la de posicionar el valor con dos constantes (0,0), de la tabla de valores que se están obteniendo de las variables, así mismo para las otras dos variables, con sus respectivas constantes (1,0), (2,0), y así tener los tres valores para ser mostrados en los indicadores correspondientes, se se visualizan en el panel frontal.

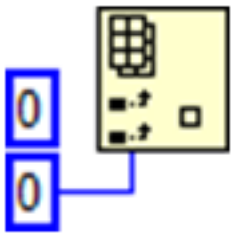

Figura A.25: Index Array

Este es un *Boolean Propierties* figura A.26, este va conectado después del *Index Array*, y antes del indicador de las variables en uso, este *led* se apagará o encenderá según el algoritmo de C++, pues 1 se enciende, 0 se apaga, esto sirve como alerta visual de que éste funcionando correctamente el programa, este icono se encuentra en el menú de *Express*, submenú *Controls Tex* en el panel frontal, esto es para las tres variables.

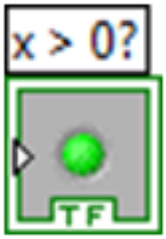

Figura A.26: Boolean Propierties

Este *Greater Than*, va conectado antes del *Boolean Propierties*, pues este hace la función de elegir entre 1 y 0 para encender el *led*, se localiza en el menú de *programming*, submenú *comparison*, este también lo llevan las tres variables, figura A.27.

Figura A.27: Operacional Greater Than

El *Merge Signals* figura A.28, sirve para reunir las tres señales de las variables para

sacar a una sola señal, la cual va a la *DAQ*, para tener las señales de salida y así poder controlarlas.

Figura A.28: Merge Signal

El *DAQ Assistant2* es utilizado para enviar señales digitalizadas ya procesadas a dispositivos analógicos como un ventilador, humificador o una lámpara, su configuración es similar a la de entrada de datos la diferencia es que ahora se ocupará la opción de *Digital I/O*, figura A.29.

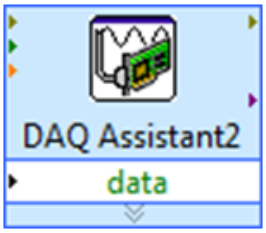

Figura A.29: DAQ Assistant 2

Bibliografía

[1] WHO. The world is fast ageing - have we noticed, April 2006. URL: www.who.int/ageing/en/.

[2] C. D. Nugent and J. C. Augusto, editors. Proceedings of the 4th International Conference on Smart Homes and Health Telematic (ICOST2006), volume 19 of Assistive TechnologyResearch, Belfast, UK, June 2006. IOS Press.

[3] Ye-In-Chan, Fwo-Long-Chen, (1997), RBE: a rule-by-example active database system, Software -Practice and experience, Vol. 27, No. 4; pp. 365-394

[4] http://www.cse.ohio-state.edu/ gurari/course/cis670/sybaseonly. html

[5] Paton N.W., Diaz O.,(1999), Active Database Systems, ACM Computing Surveys,Vol. 31, No. 1, pp. 64-103.

[6] McGoveran D., Date. C.J.,(1992), A guide to SYBASE and SQL Server : a users guide to the SYBASE product, Sybase, Inc.

[7] Lacy-Thompson T.,(1990), INFORMIX-SQL, A tutorial and reference, ISBN-0-13-465121-9,Ed. Prentice Hall.

[8] Hursh C.J., Hursch J.L.,(1991), Oracle SQL Developers Guide, ISBN-0-8306-2529-1, Ed. McGraw-Hill.

[9] Gonzlez-Pérez A.,(1999), SQL Server, Programación y administración, ISBN 970-15-0376-7, Ed. Alfaomega ra-ma.

[10] Hanson E.N.,(1996), The Design and Implementation of the Ariel Active Database Rule System, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 1.

[11] http://dns1.mor.itesm.mx/ emorales/Cursos/RdeC/node70.html

[12] Dayal U., Blaustein B., Buchmann A., Chakravarthy U., Hsu M., Ledin R., McCarthy Rosenthal A., Sarin S., Cary M.J., Livny M. and Jauhari R.,(1998), The HiPAC Project: combining active database and timing constraints, SIGMOD Record, 17(1), pp. 51-70.

[13] Widom J.,(1996), The Starburst Active Database Rule System, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 4.

[14] StonebrakerM., Kemmintz G.,(1991), The POSTGRES Next-Generation Database Management System, Communications of the ACM, Vol. 34, No. 10.

[15] http://www.ca.postgresql.org

[16] Potamianos S., StonebrakerM.,(1996), The POSTGRES Rule System, Active Database Systems: Triggers and Rules for Advanced Database Processing, Ed. JenniferWidom and Stefano Ceri, pages 43-61.

[17] Hanson E.N.,(1996), The Ariel Project,Active Database Systems: Triggers and Rules for Advanced Database Processing, Ed. Jennifer Widom and Stefano Ceri, pages 63-86.

[18] Miranker D.P., TREAT: A better match algorithm for AI production systems, in Proceedings of the AAAI Conference on Artificial Inteligence, pages 42-47, August 1987.

[19] Widom J.,(1996), The Starburst Rule System, Active Database Systems: Triggers and Rules for Advanced Database Processing, Ed. JenniferWidom and Stefano Ceri, pages 87-109.

[20] Augusto J.C., Nugent C.D. (2006). Smart Homes Can Be Smarter, Lecture Notes in Computer Science, 1-15.

[21] Augusto J.C., (2007). Ambient Intelligence: Basic Concepts and Applications, Communications in Computer and Information Science, 16-26.

[22] Augusto J.C., McCullagh P., McClellan V.,Walken J.A.,(2006). Enhanced Healthcare Provision Through Assisted Decision-Making in a Smart Home Environment, Communications in Computer and Information Science, 45-47.

[23] Augusto J.C., Nugent C.D., Martin S., Olphert C.,(2005). Software and Knowledge Engineering Aspects of Smart Homes Applied to Health, Personalised Health

Management Systems: The Integration of Innovative Sensing, Textile, Information and Communication Technologies, 164-171.

[24] William Mann, Sumi Helal, (2002). Smart Phones for the Elders: Boosting the Intelligence of Smart Homes, AAAI Technical Report WS-02-02.

[25] IST Advisory Group. The european union report, scenarios for ambient intelligence in 2010, 2001.

[26] J.C. Augusto and D. Cook. Ambient Intelligence: applications in society and opportunities for AI. IJCAI, Hyderabad, India, January 2007. Lecture Notes for the tutorial given during 20th Int. Joint Conference on Artificial Intelligence (IJCAI 2007).

[27] Rayer. Other perspectives on ambient intelligence, 2006.

[28] M. Weiser. The computer for the twenty-first century. Scientific American, 165:94104, 1991.

[29] Norbert Streitz and Paddy Nixon. Special issue on the disappearing computer. In Communications of the ACM, V 48, N 3, pages 3235. ACM Press, March 2005.

[30] M. Weiser. Hot topics: Ubiquitous computing. IEEE Computer, 26(10):7172, 1993.

[31] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. IEEE Computer, 36(3):2531, March 2003.

[32] JC. Augusto and CD. Nugent. Designing Smart Homes: the role of Artificial Intelligence. Springer, 2006.

[33] Elsmari R., Navathe S. B., Sistemas de Bases de Datos, Conceptos fundamentales. Segunda Edición. Editorial Addison-Wesley. ISBN 0-201-65370-2.

[34] L. Warshaw and L. Obermeyer and D. Miranker and S. Matzner, (1999) VenusIDS: An Active Database Component for Intrusion Detection, Applied Research Laboratories, University of Texas, Austin.

[35] N. Carreto, A. B. Bermejo,(2005), Inteligencia Ambiental. Centro de Difusión de Tecnologías;

[36] J. Medina,(2010), Aplicación de Bases de Datos Activas en Inteligencia Ambiental. Universidad Autónoma del Estado de Hidalgo.

[37] B. de Ruyter, E. Aarts,(2004), Ambient Intelligence: visualizing the future; AVI '04 Proceedings of the working conference on advanced visual interfaces.

[38] J.C. Augusto. P. McCullagh, V. McClelland, and J.A. Walkden. Enhanced Healtcare Provision Throug Assited Decision-Making in a Smart Home Environment. Proceedings of the 2nd Workshop on Artificial Intelligence Techniques for Ambient Intelligence - AITAmI'07, IJCAI, pages 27-32.

[39] J. C. Augusto, and C. D. Nugent. A New Architecture for Smart Homes Based on ADB and Temporal Reasoning. Proceedings of 2nd International Conference On Smart homes and health Telematic, ICOST2004), Assistive Technology Research Series.

[40] J. C. Augusto. Towards personalization of services and an integrated service model for smart homes applied to elderly, Proc. Int. Conf. on Smart Homes and Health Telematics, pp. 151-158, Sherbrooke, Canada, Jul. 2005.

[41] F. Wang, and K. J. Turner. Towards Personalised Home Care Systems In: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments 2008, Athens, Greece July 16 - 18, 2008. ACM International Conference Proceeding Series; 2008, Vol. 282.

[42] J.C. Augusto, J. Liu, and L. Chen. Using Ambient Intelligence for Disaster Management. Proceeding KES'06 Proceedings of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part II, Pages 171-178. ISBN:3-540-46537-5 978-3-540-46537-9

[43] J.C. Augusto, C.D. Nugent, S. Martin, C. Olphert. Software and knowledge engineering aspects of smart homes applied to health. Stud Health Technol Inform. 2005;117:164-71.

[44] Paton N.W., Diaz O., Active Database Systems, ACM Computing Surveys, Vol. 31, No. 1,1999, pp. 64-103.

[45] Dittrich K., Gatziu S., Geppert A., The Active Database Management System Manifesto: A Rulebase of ADBMS Features. A Join Report by the ACT-NET Consortium.