IX Jornadas de Ciencia y Tecnología
de Facultades de Ingeniería del NOA
Santiago del Estero, 3 y 4 de Octubre de 2013

# Aplicación de middleware DDS en una red domótica

Daniel Villagran[1] ; Paola Beltramini[1]; Sergio H. Gallina[1]

*(1) Departamento de Electrónica, Facultad de Tecnología y Ciencias Aplicadas, Universidad Nacional de Catamarca.*
*ldvillagran@hotmail.com ; sgallina@tecno.unca.edu.ar; pbeltramini@tecno.una.edu.ar*

RESUMEN: En el marco del proyecto denominado "Desarrollo de una Plataforma de Control de Entornos Residenciales" nos propusimos el desarrollo de una red domótica compuesta por nodos inteligentes, estandarizados y con capacidad para identificarse dentro de la red existente sin necesidad de un administrador central y complejas configuraciones. Cada nodo debe poseer capacidad de comunicación, de procesamiento de la información obtenida y fundamentalmente, cada uno de los nodos debe ser capaz de promover el establecimiento de una red de datos eficiente. Para ello se hace necesario definir las características de los nodos, su homogenización basada en estándares internacionales y posteriormente, desarrollar el sistema operativo y las aplicaciones necesarias. En este trabajo se analiza la problemática asociada al diseño e implementación de middleware (o software de intermediación) para distribución de datos orientado a sistemas distribuidos de tiempo real según el modelo de publicación-suscripción.

Palabras clave: Domótica, Nodos inteligentes, DDS, tiempo real, publicación-suscripción.

## 1 INTRODUCCION

La tendencia actual en Domótica es integrar mediante software simple una red de dispositivos inteligentes inalámbricos, que puedan trabajar juntos de manera óptima y al mismo tiempo mantener un cierto grado de flexibilidad.
En Argentina, la oferta domótica actual está centrada principalmente en los aspectos de seguridad y control automático, y su aplicación se ha orientado a viviendas nuevas por las características de complejidad que se supone respecto de las instalaciones a realizar.
En Octubre de 2011, la Comisión Domótica del CIEC (Colegio de Ingenieros Especialistas de Córdoba) presentó la última actualización de la "Guía de contenidos mínimos para la Elaboración de un proyecto de domótica", donde se detallan los aspectos básicos que se deben tener en cuenta al momento de encarar una instalación domótica (arquitectura, topología, tipo de enlaces, protocolos). Entre los aspectos generales destacados esta comisión aconseja:
* El proyecto debe respetar los estándares internacionales de domótica.
* Los productos realizados por diferentes fabricantes deben poder ser combinados entre si.
* Se debe poder garantizar el mantenimiento y las ampliaciones futuras con la instalación de nuevos equipos.
* Se recomienda utilizar protocolos estandarizados a nivel mundial, de esta manera se evita cualquier tipo de incompatibilidad entre productos de diferentes fabricantes.

A partir de estas consideraciones, y en el marco del Proyecto "Desarrollo de una Plataforma de Control de Entornos Residenciales", el cual propone el desarrollo de una instalación domótica basada en nodos autónomos que no requieren una unidad central de control, se describe la implementación del middlware con el desarrollo DDS de OMG en la red de nodos inteligente tomando con base que los nodos están construido con los estándares IEEE 1451.

## 2 RED DOMOTICA

### 2.1 *Consideraciones generales*

Para el diseño de la red Domótica que cumpla con las premisas de la CIEC, se ha encarado el desarrollo de "nodos inteligentes" autónomos, fiables, identificables y adaptables a una variada topología y medios de transmisión, sin perder de vista las compatibilidades logradas. Cada uno de estos nodos se ha planteado con capacidad para:
* Eliminar interfaces mediante una representación única para todos los subsistemas.
* Eliminar el control maestro o central de operaciones.

* Software embebido libre de mantenimiento.
* Posibilidad de monitoreo mediante PC, Tablet o Smartphone.
* Protocolo de comunicación conocido por los fabricantes de equipos del hogar y de bajo costo en el uso de los recursos de la red.
* Estandarización del diseño con independencia de marcas y equipos.

## 3 NODOS DE LA RED

### 3.1 *Nodos de la red*

Se ha encarado el desarrollo de un "nodo inteligente", en el cual se encuentran integrados un sensor digital o analógico, un elemento actuador, una unidad de procesamiento y una interfaz de comunicación. Se busca que estos nodos posean las siguientes características:
• Que al conectar un nuevo nodo a la red este pueda auto identificarse.
• Que los componentes del nodo sean plug & play.
• Cada una de las placas para transductores debe atender las necesidades de un artefacto del hogar con una interfaz estandarizada.
En función de estas especificaciones, el diseño de cada nodo se realiza en base al estándar IEEE 1451, respondiendo a una estructura como la que se muestra en la fig.1. En ella se puede ver que cada nodo está compuesto por una NCAP (Network Capable Application Processor), una TIM (Transducer Interface Module), conectados mediante una conexión cableada de 10 líneas (IEEE 1451.2). Cada NCAP debe controlar un único TIM y a cada TIM se pueden conectar entre 1 hasta un máximo de 255 transductores mediante protocolo $I^2C$.

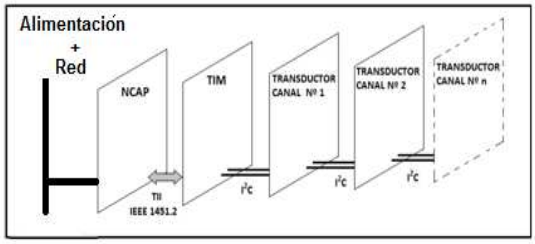


Figura 1. Estructura del Nodo

### 3.2 *Funciones del nodo*

El NCAP tiene tres funciones bien definidas:
* Realizar el networking a través, en nuestro caso, de Internet 0;
* Atender las comunicaciones con el TIM y los transductores; y
* Ejecutar la aplicación específica del nodo.

#### 3.2.1 Comunicación NCAP-NCAP

El estándar IEEE 1451 no establece condiciones para la comunicación NCAP-NCAP dentro de la red, la implementación en nuestro caso la hemos realizado utilizando el modelo de comunicación INTERNET 0 (Protocolo de red simple que propone una reducción de las capas del modelo OSI), sobre una línea de alimentación de nodos de 24 V DC.

#### 3.2.2 Comunicación NCAP-TIM

El TIM se comunica con un NCAP usando el modelo TII, definido en el estándar IEEE 1451.2, donde un único TIM se conecta a un NCAP. Esta interfaz es compatible con los diferentes comandos que pueden ser enviados a la TIM, y también proporcionará la capacidad de las interrupciones en el protocolo.

#### 3.2.3 Comunicación TIM-Canal transductor

Cada transductor se conecta al TIM mediante un bus I2C, esto permite la identificación univoca del canal; cada canal al conectarse debe identificarse frente al TIM quien actualizara sus TEDS (Transducer Electronic Data Sheet.) u hoja de datos electrónicos de un transductor.

### 3.3 *Topología*

La topología adoptada, es decir la forma en que está diseñada la red, (físicamente o bien lógicamente) se basa en un bus implementado sobre la línea de alimentación de CC (24V), con protocolo Internet 0 y codificación manchester.
Esta topología, mostrada en la fig.2, presenta las ventajas de posibilitar la implementación sobre pequeños microcontroladores y de identificación de los nodos basados en direcciones IP.

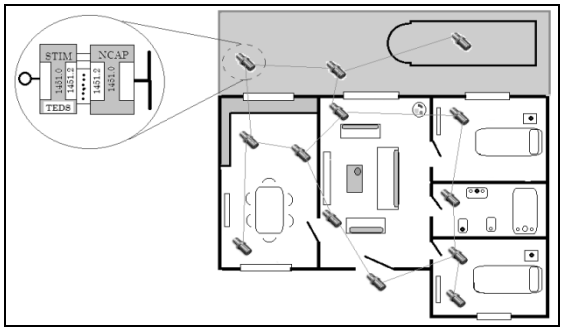


Figura 2: Configuración de una red domotica

## 4 MECANISMOS DE COMUNICACIÓN ENTRE NODOS

La arquitectura de comunicaciones entre los nodos tiene una gran importancia en la red, principalmente en el rendimiento, la facilidad para llevar a cabo transacciones, la forma de detectar errores y la robustez frente a diferentes situaciones erróneas.

Hay una amplia variedad de paradigmas y modelos de comunicación. Los mecanismos habituales se basan en alguno de los siguientes paradigmas:

- modelo punto a punto,
- modelo cliente-servidor,
- modelo publicación-suscripción o editor-suscriptor.

En nuestra aplicación, y luego de analizar diferentes modelos para el intercambio de información en sistemas distribuidos, se optó por aplicar un modelo de comunicación *data-oriented publicación-subscripción* en lugar de que los datos entren y salgan de un servidor central (modelo cliente servidor), o de un canal exclusivo entre nodos (modelo punto a punto).

En el modelo publicación-suscripción se distinguen dos tipos de nodos: Los que producen información (*editores o publicadores*) y los que reciben la información (*suscriptores*). Una aplicación puede ser un editor de datos, un suscriptor de los datos, o ambos, un editor y el suscriptor. Aplicar un modelo publicación-suscripción, los recursos dejan de estar centralizados en un servidor, y pasan a formar parte de un conjunto de datos de interés, conocido como espacio global de datos o *data-space* (espacio de datos).

El middleware DDS (*Data Distribution Service*), es un estándar de comunicaciones de datos abierto de la OMG *(Object Management Group)* que utiliza el paradigma publicación-suscripción, apoyado en el protocolo de conexión Real-Time Publish Subscribe (RTPS). DDS define las relaciones de comunicación entre publicadores y suscriptores, las cuales están desacopladas en espacio, tiempo y flujo puesto que los nodos no necesitan una ubicación específica y pueden publicar cuando lo deseen sin más restricciones temporales que la que el desarrollador decida imponer con el ancho de banda dedicado.

DDS posee alto rendimiento, predictibilidad y determinismo, altas posibilidades de configuración, y por tanto es una buena elección para sistemas distribuidos de tiempo real crítico.

DDS usa el concepto de espacio de datos global (Fig.3). DDS define el concepto de "tópico" (posiciones de distintos objetos, alarmas, mediciones de sensores, temperatura, presión, localización, etc.), el cual es un canal lógico para el intercambio de información entre los publicadores y subscriptores. Es posible definir uno o varios campos en el topic, que sirvan de clave o *key*, DDS usará esta clave para ordenar los datos, posibilitando que una aplicación pueda pedir al middleware los datos que coincidan con una clave determinada, para tratarlos de forma independiente al resto. Por ejemplo, el topic puede ser el tipo de sensor, y la clave la ubicación del identificador del vehículo, de forma que no es necesario crear un topic para cada vehículo.

La estructura de los datos de los topics se define usando un lenguaje de definición de datos estándar del OMG, denominado IDL, cuya sintaxis es muy similar a la definición de datos en C++.

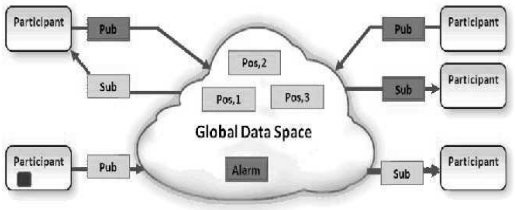


Figura 3. Modelo simplificado de comunicaciones DDS

Entre las principales ventajas de este estándar de comunicación podemos mencionar:

- Nodos Plug & play: los publicadores no necesitan conocer la localización de los suscriptores y viceversa, es decir, no se necesitan configurar las direcciones de los nodos. El descubrimiento de los distintos nodos (o participantes) es automático.
- La recepción de los datos es asíncrona, no es necesario que el suscriptor realice una petición por cada dato que quiere recibir, sino que se le avisa cuando hay datos disponibles.
- Redundancia: Es posible que se suscriban a los mismos datos diferentes suscriptores, así como que se escriban desde distintos publicadores.
- DDS dispone de un rico conjunto de calidades de servicio (QoS), que permiten regular, priorizar y dar forma a los flujos de datos en una red, haciendo sencillo resolver los problemas más comunes de diseño que se presentan en un sistema distribuido.
- Permite acortar el tiempo de desarrollo de aplicaciones de tiempo real, ya que facilita una capa adicional que desacopla a la aplicación de las dificultades inherentes en la comunicación.
- DDS dispone de un protocolo de interoperabilidad, también estándar del OMG, el RTPS (Real-Time Publish Subscribe), de

forma que incluso cada lado de la comunicación puede usar una implementación diferente del estándar.

## 5 PROTOCOLO DE TIEMPO REAL PUBLICACIÓN-SUSCRIPCIÓN (RTPS)

### 5.1 *Conceptos básico y su aplicación en protocolo IP/UDP*

Con la explosión de Internet, el conjunto de protocolos TCP / UDP / IP se ha convertido en el marco básico en el que se construyen todas las comunicaciones basadas en Internet. Sin embargo, estos protocolos tienen nivel de transporte demasiado bajo para ser utilizados directamente por cualquier de las aplicaciones más sencillas, por lo que han surgido protocolos de alto nivel como HTTP, FTP, DHCP, DCE, RTP, DCOM y CORBA, cada uno de los cuales proporciona óptima funcionalidad para fines específicos o dominios de aplicación.
El protocolo RTPS define el protocolo de interoperabilidad de DDS y está diseñado para ejecutarse en un mecanismo de transporte poco fiable, tal como UDP / IP, (en nuestro caso INTERNET 0). Su objetivo y el alcance es asegurar que las aplicaciones basadas en implementaciones de DDS diferentes proveedores pueden interoperar. Se aprovecha de las funciones de multidifusión del mecanismo de transporte, donde un mensaje de un remitente puede llegar a múltiples receptores.
Las principales características del protocolo RTPS incluyen:
• Propiedades de rendimiento y calidad de servicio para permitir comunicaciones fiables de aplicaciones en tiempo real a través de redes IP estándar.
• Conectividad Plug&play para que las nuevas aplicaciones y servicios se detecten automáticamente y las aplicaciones puedan unirse y dejar la red en cualquier momento sin necesidad de reconfiguración.
• Tolerancia a fallos creando redes sin puntos únicos de fallo.
• Extensibilidad para permitir que el protocolo pueda ser ampliado y mejorado a nuevos servicios sin romper la compatibilidad y la interoperabilidad. • Configurabilidad para permitir equilibrar los requisitos de fiabilidad y puntualidad de cada entrega de datos.
• Modularidad para permitir que los dispositivos simples puedan implementar subconjuntos del protocolo y aún participen en la red.
• Escalabilidad para permitir el potencial crecimiento de la red sin afectar su funcionalidad ni rendimiento.
• Seguridad para proteger a la red de posibles errores de programación.
Estas especificaciones definen los formatos de los mensajes, la interpretación, y escenarios de uso que subyacen a todos los mensajes intercambiados por las aplicaciones que utilizan el protocolo RTPS.

### 5.2 *Implementación del protocolo en el desarrollo de los nodos*

Nuestro sistema implementado de publicación-suscripción permiten una distribución asíncrona de información y está formada (Fig.4) por: Productor de información que publica esta información sin tener que saber quién está interesado en recibirla; Consumidor de información que suscribe a los canales que diseminan la información que le interesa; Mediador (broker) que recibe la información de los productores y peticiones de suscripción de los consumidores y también se encarga de encaminar la información publicada a los destinatarios suscritos al canal y el Canal que son los conectores (lógicos) entre los productores y los consumidores de información. Los canales modelan una relación de uno a muchos entre productores y consumidores.

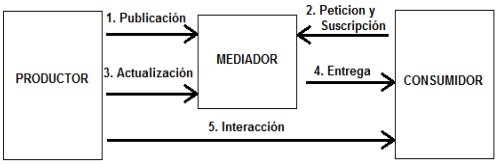


Figura 4: Arquitectura del modelo DDS

El protocolo se divide en tres partes debido a la funcionalidad que persigue cada una de ellas.
A). Modelo independiente de la plataforma (*PIM*)
* El RTPS PIM contiene cuatro módulos que definen cómo se realizan las implementaciones del protocolo para que sea independiente de la plataforma.
* El módulo de estructura: Todas las entidades RTPS se asocian a un dominio RTPS lo que crea un plano de comunicaciones separado que contiene una serie de participantes. Cada participante tiene puntos finales que pueden ser lectores, escritores o ambos a la vez. Estos puntos finales son los que suministran y extraen información de la red.
* El módulo de mensajes: Define el contenido de información que se intercambian entre lectores y escritores. Cada mensaje tiene una cabecera y en su interior submensajes con sus propias

cabeceras. A su vez cada submensaje está compuesto por elementos. Gracias a esta estructura decimos que DDS y RTPS son centrados en los datos ya que cada elemento es un dato.

* El módulo de comportamiento: Describe las secuencias permitidas de mensajes que pueden ser intercambiados entre lectores y escritores así como los cambios que produce cada mensaje.

* El módulo de descubrimiento: describe la parte del protocolo que permite a los participantes obtener información sobre

* La existencia y atributos del resto de participantes del dominio. Se construye un metatráfico que informa de escritores, lectores, altas y bajas.

B). Modelo específico de la plataforma (*PSM*):

Es el encargado de mapear el PIM a cada plataforma. Se definen las representaciones de bits y bytes de todos los tipos de mensajes RTPS y las equivalencias con los tipos nativos. Además, pueden ser soportados varios PSM's pero todas las implementaciones de DDS deben, como mínimo, implementar PSM sobre UDP/IP.

C). El Modelo de Transporte:

RTPS da soporte a una gran cantidad de transportes y calidades de servicio. Se ha diseñado para que sea capaz de ejecutar multicast sobre transporte best effort como sería UDP. Lo mínimo necesario que se puede ofrecer es el servicio sin conexión. Es decir, no hace falta que garantice entrega ordenada.

Nuestro sistema tiene comunicación centrada en los datos (Data-Centric) por lo cual nos permite que la computación se hace en nodo (local) y son los datos los que se van desplazando a través de la red. Es decir, que los datos son los que viajan de un nodo a otro para ser procesados localmente garantizando coherencia.

El DDS nos está proporcionando la habilidad de especificar parámetros como frecuencia de publicación de datos, tiempo de validez, etc. que en definitiva nos permite establecer una calidad de servicio (QoS). Así este entorno nos permite desarrollar una comunicación más controlada por el programador y más específica de cada situación. La herramienta fundamental para llevar a cabo la implantación es el lenguaje IDL, lenguaje para descripción de interfaces (Interface description language), es un lenguaje de programación utilizado para describir los componentes de software de una interfaz. Describe una interfaz en un lenguaje neutral, lo cual nos permite la comunicación entre componentes de software que no comparten el mismo lenguaje como por ejemplo, entre los componentes escritos en C++ y otros escritos en Java. Permitiendo al sistema distribuido utilizar lenguajes y sistemas operativos diferentes. IDL ofrece un puente entre dos sistemas diferentes.

### 5.3 *Los mensajes lógicos básicos*

El modelo de comunicación de publicación-suscripción proporciona un mecanismo de acoplamiento flexible para las comunicaciones de red entre los objetos donde el objeto emisor, Publisher, no tiene por qué estar al tanto de los objetos que reciben, Suscriptores.

En base a las normas IEEE 1451.1 que utilizaremos en nuestro diseño el modelo de comunicación publicación-suscripción se apoya en dos operaciones.

*Publish* es un Objeto de Puertos Publicadores

*AddSubscriber* y *Callback* son Objetos de los Puertos Suscriptores

La Figura 5 muestra la organización objeto que genera la comunicación publish-subscribe.

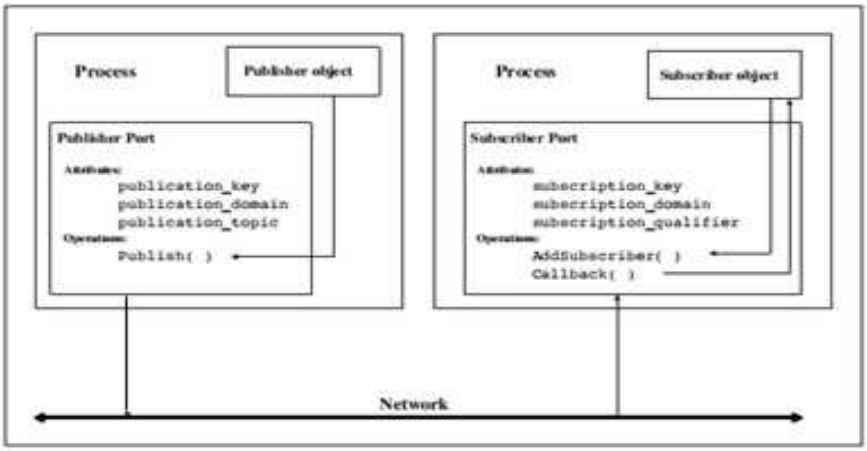


Figura 5: componentes de comunicación del Publish-Subscribe

Para describir la estructura y funcionalidad de las clases de objetos utilizaremos diagramas UML.

De este modo, se define el encabezado de la clase como los atributos de la clase, las operaciones visibles de la red como pública, y las operaciones locales como protegidas.-
La clase Publisher Port (se muestra en la Fig.6) proporciona la operación de publicación.

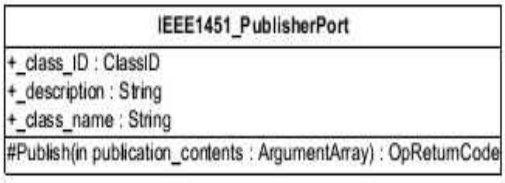


Figura6: Diagrama de clase UML Publisher Port

La clase Subscriber Port proporciona objetos con un mecanismo para suscribirse a las publicaciones. La estructura de esta clase se muestra en la fig.7.

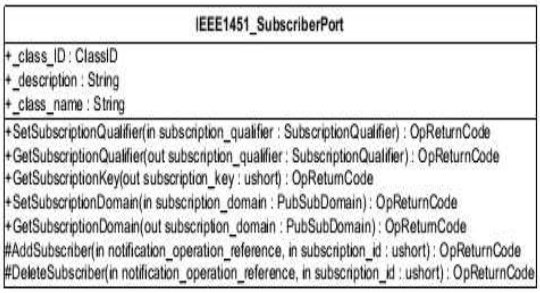


Figura 7: Diagrama de clase UML Subscriber Port

Las operaciones que se definen por clase del *Subscriber Port* se pueden resumir de la siguiente manera:
La operación *SetSubscriptionQualifier* se utiliza para inicializar o modificar el valor actual de clasificado de la suscripción del puerto. El calificador de suscripción se utiliza para determinar qué publicaciones serán aceptados por el puerto
La operación *GetSubscriptionQualifier* se utiliza para obtener el valor actual de clasificado de la suscripción del puerto.
La operación *GetSubscriptionKey* se utiliza para obtener el valor actual de la clave de suscripción del puerto. Esta clave se utiliza para determinar el tipo de publicación que el puerto está suscribiéndose.
La operación *GetSubscriptionDomain* se utiliza para obtener el valor actual de la suscripción de dominio que define un servicio de publicaciones candidatos para ser aceptado por el puerto.
Las operaciones *AddSubscriber* y *DeleteSubscriber* son opcionales.
Las operaciones anteriormente definidas se utilizan para las comunicaciones de red publish-subscribe.
La interacción entre las operaciones y objetos puede ser explicado de la siguiente manera:
El objeto Publisher invoca la operación publicará (Publish) en su correspondiente *Publisher Port* local enviando como argumento de entrada el contenido de la publicación.
Utilizando la infraestructura de red, la invocación de los resultados de operación publicará en la entrega de la publicación a todos los *Subscriber Ports* en el Dominio de la publicación.
La recepción de cada *Subscriber Port* usa los valores de los atributos de *Subscription Key*, *Subscription Domain*, y *Subscription Qualifier* para filtrar la publicación entrante.
Si la publicación pasa por el filtro de un puerto de suscriptor, el Puerto invoca todas las operaciones de devolución de llamada a sus suscriptores registrados, que proporciona como argumento de entrada el contenido de la publicación.
En base a lo desarrollado en UML y con la utilización del software *YAKINDU SCT* permite el diseño del sistema en lenguaje C++ que se implementara en cada *Publisher Port* y *Subscriber Port*

### 5.4 *Implementación de calidades de servicio (QoS) con RTPS*

Uno de los aspectos importantes a considerar en el DDS es la calidad de servicio (QoS); debido a que nuestro desarrollo es una red Domotica tenemos que considerar que van a coexistir datos que requieran repuesta más rápidas que otras es decir el diseño debe responder a las necesidades de aplicaciones de tiempo real con datos críticos, para ello debemos proporcionar una serie de mecanismos estandarizados conocidos como políticas de calidad de servicio, que permiten configurar cómo se produce la comunicación, para así limitar los recursos utilizados por el middleware con el objeto de detectar que a nuestra aplicación le permite especificar ciertos parámetros relacionados con la forma de un servicio se entrega. Los QoS proporcionan la capacidad de controlar y limitar el uso de recursos como el ancho de banda de red, memoria, fiabilidad, puntualidad, y la persistencia de datos, entre otros. La especificación DDS propone 22 políticas de QoS que cubren todos los aspectos de la gestión de las comunicaciones: aspectos

temporales de los mensajes, flujo de mensajes y meta datos.
A continuación se enumeran las principales políticas de calidad de servicio incorporadas a nuestra aplicación: Plazo (*Deadline*): Indica la velocidad mínima a la que el escritor enviará los datos. Filtro temporal (*Time-Based Filter*): limita el número de muestras que se entregan a la aplicación por segundo, estableciendo una separación temporal mínima entre ellas. Modo de vida (*Liveliness*): Especifica como la infraestructura DDS determina si una entidad está viva o no. Tiempo de vida (*Lifespan*): Establece la máxima duración de la validez de cualquier mensaje individual. El propósito de esta política es evitar la entrega de muestras con un retardo asociado excesivo. Dueño (*Ownership*): Especifica si diferentes escritores pueden actualizar la instancia de un objeto. Importancia del dueño (*Ownership strength*): Parámetro que se adjudica a los escritores y les establecen un número de cuán importante es su dato. Esto permite que unos escritores tengan prioridad para escribir datos sobre otros. Partición (*Partition*): Es un modo de separar tópicos dentro de un dominio Modo Publicador (*Publish_Mode*) permite la implementación de mecanismos de control de flujo de forma sencilla.Datos del tópico(*Topic_Data*)/ Datos de usuario (*User_Data*) Estas políticas de calidad de servicio permiten la entrega de información adicional durante el descubrimiento de entidades DDS.

## 6 CONCLUSIONES

La utilización de middleware DDS efectivamente acorta los tiempos de desarrollo de aplicaciones que requieren la distribución de datos en tiempo real. Además, DDS facilita la implementación domotica como la notificación de presencia de datos que ingresan en sensores y que permiten a actuadores realizar las distintas tareas como así también permite desarrollos a futuro en base a un conjunto extenso de políticas de calidad de servicio.
Se ha comprobado que aplicación del modelo de publicación-subscripción para nuestro sistema domotico es viable y además adecuado, siendo además que al no requerir de un servidor centralizado para tareas de intercambio de datos, es altamente escalable. Dada la arquitectura modular del sistema desarrollado, el código generado es altamente reutilizable, por lo que se pueden integrar con un mínimo esfuerzo los servicios de mensajería, y también intercambio de audio o vídeo implementados en nuevas aplicaciones que se desarrollen. Por último la implementación sobre IP/UDP proporciona una interfaz que se pueden aplicar sobre plataformas que son totalmente compatibles con el desarrollo del Internet.

## 7 REFERENCIAS


Pucheta J.A., S.H. Gallina, L.D Villagrán, P.I. Beltramini, G. Peretti & S.F. Felissia, *Desarrollo de una Plataforma de Control de Entornos Residenciales*, WICC, 2012.

Pucheta J.A., S.H. Gallina, L.D Villagrán, P.I. Beltramini, G. Peretti & S.F. Felissia, *Internet cero: Domótica con nodos inteligentes*, Revista Producción científica de la F.T. y C.A., 2012

OMG: Data-Distribution Service for Real-Time Systems (DDS) v1.2, http://www.omg.org/cgi-bin/doc?formal/07-01-01.pdf, 2012.

OMG: Object Management Group, http://www.omg.org/, 2012.

OMG: The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification" Version 2.1 OMG Document Number: formal/2009-01-05

López G.E., *System-On-A-Chip Solution For Plug And Play Networked Smart Transducers*, University of Pittsburgh, 2002.

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Network Capable Application Processor (NCAP) Information Model, IEEE1451.1-1999, June 26, 1999

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE std 1451.2-1997, 2012.

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems. IEEE std 1451.3-2003, March 31, 2004

Lopez Vega JM, J. Sanchez Monedero, J. Povedano Molina & J.M. Lopez Soler, *QoS Policies for Audio/Video Distribution Over DDS Middleware*, Workshop on Distributed Object Computing for Real-time and Embedded Systems; 2008, http://www.omg.org/news/meetings/-workshops/rt_embedded_2008.htm, 2012

Rodríguez López I & M. García Valls, *Desarrollo de Software Distribuido de Tiempo Real basado en DDS*, Proyecto de fin de carrera ingeniería técnica de telecomunicación: telemática, 2009.