Un modelo para extender capacidades de robot basado en el concepto de computación en la nube

Sarli, Juan
Andini, Armando
Gutiérrez, Milagros

CIDISI, Universidad Tecnológica Nacional, Facultad Regional Santa Fe

Abstract

En los últimos años el concepto de computación en la nube como una extensión de la arquitectura orientada a servicios (SOA), está teniendo un impacto sin precedentes en la forma en que los robots tradicionales se comportan. Los requerimientos de ejecutar numerosas aplicaciones concebidas a-priori, como también las que van surgiendo en lo inmediato, encuentran en la computación en la nube una herramienta eficiente en costo y expeditiva en la solución. En este trabajo se presenta un modelo que permite extender las capacidades del robot sobre-demanda, proponiendo para ello una arquitectura de computación en la nube basada en el modelo software como servicio (SaaS).

Palabras Clave

Computación en la nube, Robot, Inteligencia Artificial, Software como Servicio, Arquitectura Orientada a Servicios.

Introducción

El nuevo paradigma de computación en la nube, o cloud computing en inglés, hace posible moverse desde la computación tradicional (basada en escritorio) a la computación basada en la web. De esta manera, computación en la nube hace referencia a aplicaciones basadas completamente en la web: los desarrolladores utilizan una plataforma web a través de un navegador web, desarrollan software y almacenan datos en la web, configuran anchos de banda, memoria, poder de cómputo y ejecutan aplicaciones en la web.

Todo esto tiene un impacto significativo en la programación de robots, dado que es posible aumentar sus capacidades sin necesidad de modificar su arquitectura física. La idea de conectar un robot a un computador externo no es novedosa, en los años ’90, investigadores de la universidad de Tokio, propusieron un “cerebro remoto” con el cual lograban dotar de inteligencia a robots desde dispositivos externos a ellos [1-2]. Un ejemplo muy conocido de este tipo de aplicaciones son las cirugías médicas remotas, donde un operador humano (médico) controla un robot a distancia enviándole comandos.

Debido a la insuficiente capacidad de cómputo, las restricciones de memoria y las limitaciones en cuanto al tiempo de autonomía que presentan los distintos robots, resulta inviable procesar grandes volúmenes de información sobre sus procesadores. Por ello, hoy en día los trabajos se orientan al concepto de “cloud robotic”, el cual permite crear robots capaces de aprender nuevas habilidades obteniendo información directamente desde la nube (cloud) [3-5]. De este modo, los robots podrán actuar de forma independiente recibiendo las instrucciones necesarias desde aplicaciones que se ejecutan en la nube. Así, aplicando el modelo de computación en la nube no sólo se logra superar estos inconvenientes sino que también se consigue robots más inteligentes, pequeños y baratos.

En esta área emergente se identificaron una serie de desafíos a abordar, los cuales son: necesidad de conectividad de los robots, adecuada infraestructura de hardware y software para dar soporte a la ejecución del servicio en la nube, necesidad de estandarización de los robots para compartir aplicaciones en forma eficiente, entre otras.

El objetivo de este trabajo es el desarrollo de una infraestructura informática y de comunicaciones que permita extender las

capacidades del robot sobre-demanda, proponiendo para ello una arquitectura de computación en la nube basada en el modelo de software como servicio (SaaS). En las siguientes secciones se presentan los conceptos usados para el desarrollo del trabajo, las herramientas y la metodología utilizada. El modelo propuesto, sus componentes principales, la transferencia de instrucciones al robot y el acceso a las capacidades del servidor en la nube. Finalmente se presentan los resultados y las conclusiones del trabajo.

Elementos del Trabajo y Metodología

1. Computación en la Nube

Para llevar adelante el trabajo, se comenzó por investigar sobre los nuevos conceptos emergentes de computación en la nube y cómo ellos debían implementarse en la infraestructura que se tenía a disposición: los robots LEGO® MINDSTORMS® NXT 2.0[1] del laboratorio de Inteligencia Artificial (IA).

Hoy en día, los paradigmas de computación en la nube y las arquitecturas orientadas a servicios (SOA) están conquistando la forma en que las organizaciones realizan sus cómputos [6-7]. Por un lado, SOA considera un sistema de software conformado por una colección de servicios débilmente acoplados los cuales se comunican unos con otros a través del uso de protocolos de intercambio de mensajes e interfaces estándares [8]. Por otro lado, computación en la nube representa un nuevo modelo de tecnologías de información para consumir y entregar servicios sobre Internet. Extiende el alcance de SOA incluyendo plataformas de desarrollo, capacidad de almacenamiento y capacidad de cómputo [9]. Computación en la nube se define como "un modelo para permitir que recursos informáticos configurables compartidos puedan ser accedidos desde todas partes, a pedido y en forma conveniente" [10]. Generalmente se reconocen tres tipos de servicios ofrecidos por la nube: SaaS, plataforma como servicio (PaaS) e infraestructura como servicio (IaaS). Sin embargo, cualquier otra virtualización de un servicio puede referenciarse como "X como servicio", por ejemplo conectividad como servicio, robot como servicio [11-12].

2. Robot Lego NXT

Un robot Lego nxt está compuesto por su "cerebro" que es el ladrillo nxt, motores, diferentes sensores, puerto USB, puertos para sensores y motores, módulo de conectividad bluetooth y una diversidad de piezas que permiten crear distintos modelos de robot (Figura 1). Esta plataforma de hardware permite desarrollar aplicaciones que se ejecuten sobre la misma (con sus restricciones de autonomía y capacidad de procesamiento), como así también soporta la ejecución de programas remotos (mediante el módulo bluetooth) removiendo las restricciones antes mencionadas.

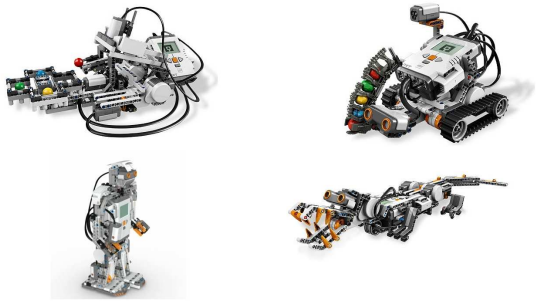

Figura 1: Distintas configuraciones del robot Lego Mindstorm

Este tipo de robots posee un software propietario (firmware) el cual se utiliza para el diseño de aplicaciones sobre los dispositivos nxt. El mismo se denomina NI LabVIEW®[2], admite el desarrollo de programas mediante mínimos conocimientos de programación, utilizando una paleta de acciones (o bloques) predefinidas por el software y que el desarrollador puede utilizar para realizar sus programas (Figura 2). De este modo, no es necesario conocer un lenguaje de programación específico o poseer un historial de desarrollos previos para

[1] http://mindstorms.lego.com/en-us/default.aspx

[2] http://www.ni.com/academic/mindstorms/esa/

iniciarse en el mundo de las aplicaciones para robots nxt.

Figura 2: Vista de LabView

Dadas las características de este software, las posibilidades definición de comportamientos para los dispositivos nxt se encuentran limitadas a las combinaciones de los bloques preexistentes. Por lo tanto, para implementar técnicas de inteligencia artificial sobre este tipo de robots, se optó por un software libre denominado LeJOS[3] que permite extender su comportamiento logrando que las aplicaciones se desarrollen con la granularidad deseada.

3. API LeJOS

LeJOS es una máquina virtual de Java[4] pequeña, la cual fue portada al bloque nxt. Dentro de las características más relevantes de LeJOS se encuentran las siguientes: permite trabajar sobre un IDE, facilita la transferencia de los programas desarrollados en la computadora al dispositivo nxt a través de herramientas bien definidas, posee threads (hilos) interrumpibles, contiene arreglos multidimensionales, implementa recursión, sincronización, excepciones, una buena documentación de la misma [13].

Debido a la experiencia previa de los participantes en el desarrollo y a que LeJOS se basa en Java, su elección como API de base permitió que se adquiriera un rápido conocimiento de la funcionalidad que ésta brindaba.

Para dar soporte a los comportamientos que pueden realizar los robots, LeJOS posee la clase *Behavior*, la cual consta de tres métodos que deben ser redefinidos cada vez que se desea implementar una nueva conducta. Estos son: *takeControl* retorna un valor booleano el cual indica si el comportamiento se vuelve activo; *action* define cual es la acción llevada a cabo por el robot, una vez que *takeControl* retorna verdadero; *supress* determina que acciones realizar cuando el comportamiento ha finalizado (su ejecución finaliza inmediatamente el método *action*).

Luego de definidos todos los comportamientos que se desea que el robot posea, se continúa definiendo una instancia de la clase *Arbitrator* la cual recibe como parámetro un arreglo de los *behavior* implementados. Luego, la misma invoca al método *start* el cual recorre el arreglo de comportamientos, verificando cuál de ellos se torna activo, es decir toma el control del robot. Utilizando esta herramienta fue posible encarar el desarrollo de agentes inteligentes que controlen el comportamiento del robot físico. Se considera agente inteligente a un sistema de software que actúa en un ambiente recibiendo percepciones y actuando en dicho ambiente a través de actuadores [14].

4. Arquitectura de Software Propuesta

Una vez definidos los conceptos y herramientas con las que se trabajó, se presenta el diseño de la arquitectura para adaptar el concepto de computación en la nube (descripto en la sección 1) a la plataforma de hardware (descripta en la sección 2) disponible para extender sus capacidades.

El concepto de computación en la nube, necesita indefectiblemente de conectividad a Internet, éste fue el primer inconveniente encontrado al momento de diseñar la arquitectura dado que los robots Lego nxt, no cuentan con dicha conectividad. Es por ello que fue necesaria la incorporación de un dispositivo intermedio, el cual dote de esta capacidad al robot para, de este modo,

[3] http://lejos.sourceforge.net/index.php
[4] www.java.com/es/download

permitirle comunicarse con el servidor en la nube.
Las siguientes subsecciones detallan los distintos aspectos que se contemplaron al momento de diseñar la arquitectura propuesta.

4.1 Elección Dispositivo Intermedio

Como los robots Lego nxt cuentan con conectividad bluetooth para mantener una comunicación remota, el dispositivo elegido debía contar con dicha característica y además brindar conectividad hacia internet. Se analizaron dos posibilidades: utilizar un smartphone o una PC como se muestra en la figura 3.

Figura 3: Conectividad del robot – dispositivos intermediarios utilizados

Se realizaron pruebas para analizar las ventajas y desventajas de ambos métodos de comunicación.
Para las pruebas se utilizó un programa de navegación que le permite al robot navegar sobre un laberinto diseñado en el laboratorio de IA. Cuando se realizaron los testeos con la PC, si bien tenía la ventaja de tener un tiempo de respuesta pequeño y aceptable, a medida que el robot comenzaba a alejarse de la PC, la señal se atenuaba gradualmente hasta perderse totalmente. El alcance de la señal, fue medido experimentalmente y se determinó que el mismo era de aproximadamente 10 metros. Debido a este problema y a la incapacidad de movilizar la PC según hacia donde navegara el robot, se optó por realizar las pruebas sobre el otro dispositivo.
Para testear el smartphone fue necesario incorporar a LeJOS un conjunto de bibliotecas que permitieran desarrollar una aplicación Android. Sorteado el problema, se procedió a realizar un programa de control remoto, el cual por medio del bluetooth enviaba las instrucciones al robot. Si bien el dispositivo móvil puede movilizarse y evitar de esta forma la perdida de la señal bluetooth, la solución planteada fue diferente; se decidió montar el teléfono celular sobre el robot, logrando de esta forma solucionar el problema de la pérdida de señal debido a la distancia (Figura 4).

Figura 4: Robot con conexión wi-fi

Finalizados los ensayos con ambos dispositivos, se eligió trabajar con el smartphone como dispositivo intermedio para comunicar al robot con el servidor en la nube.
Luego de seleccionado el dispositivo intermedio, que actúa como enlace de comunicación, se comprendió como transferir instrucciones entre una aplicación remota y el robot vía bluetooth, lo cual es tan solo un subproblema del problema planteado.

4.2 Diseño de la Arquitectura

El siguiente paso fue diseñar una arquitectura que permita comunicar los dos extremos del modelo: el robot físico encargado de percibir y actuar y la aplicación en la nube encargada de tomar la decisión de que acción emprender de acuerdo a las percepciones y al estado actual del robot. Para ello es necesario tanto, enviar las percepciones desde el robot físico al componente que se encuentra en la nube como también enviar la información de la acción a emprender desde la aplicación en la nube al robot físico. Además esta comunicación estará mediada por aplicaciones que corren en el smartphone que actúan como intermediarios.

Lo expuesto en la sección 4.1, es la solución propuesta al problema de comunicación entre el robot y el smartphone, es necesario resolver el intercambio de información entre este último y la aplicación en la nube. Para ello se programó un servicio web que brinda el acceso a las aplicaciones del robot en la nube. La arquitectura de cloud-robotic que se propone en este trabajo es una arquitectura basada en el esquema Cliente/Servidor el cual se enmarca en el estilo arquitectónico de procesos comunicantes [15]. Dicha arquitectura, es un modelo de aplicación distribuida en el que las tareas se reparten entre los que permiten el acceso al servicio, llamados servidores; y los que demandan estos servicios, llamados clientes [16]. En la Figura 5, se esquematiza la arquitectura descripta en función de módulos, componentes y mecanismos de comunicación.

El modelo planteado se diseñó para trabajar con una carga de procesamiento asimétrica: el servidor es el que posee mayor poder de procesamiento de los tres elementos del modelo, es por ello que sobre él recae la mayor carga; debido a que el smartphone tiene una capacidad media y su funcionalidad es la de actuar como intermediario, la carga de procesamiento que maneja no es tan grande como la del servidor, pero es mayor a la del robot nxt. Por otro lado, como el robot es el que menor capacidad posee, realiza el procesamiento mínimo e indispensable. Teniendo esto en mente, se establecieron las tareas que cada uno de los componentes mencionados realiza.

Las tareas que se le encomendaron al servidor fueron: establecer/cerrar las conexiones para acceder al servicio web, lo cual se resolvió por medio del módulo *Administrador de Conexiones*; gestionar la forma en que se accede a las aplicaciones, problema que se solucionó mediante el módulo *Administrador de Servicio*; almacenar los servicios solicitados y ejecutarlos, cuestión resuelta por el componente *Cuerpo del Servicio*.

Por otro lado, al smartphone se le asignaron las siguientes funciones: establecer/cerrar las conexiones para acceder al servicio web y para comunicarse con el robot Lego nxt, esto se resolvió por medio del módulo *Administrador de Conexiones*; transferir la información entre el servidor y el robot, funcionalidad realizada por el módulo *Resolvedor de Protocolo*.

Finalmente, al último componente del modelo, es decir el dispositivo nxt, se le encargo una única labor: procesar las instrucciones que reciba y otorgar una respuesta en el caso que sea necesario. Dicha tarea se resolvió mediante el componente de *Sistema Embebido* que para el caso planteado fue LeJOS.

La utilización de la arquitectura propuesta ofrece una serie de ventajas, entre las cuales se pueden destacar como las más importantes a las siguientes: Otorga escalabilidad, centraliza el control en un punto, favorece la mantenibilidad y además permite que se trabaje con clientes con baja capacidad de procesamiento debido a que la ejecución de las aplicaciones se realiza en el servidor.

El esquema descripto con anterioridad, posee un buen nivel de generalización, esto se debe a que el módulo *Cuerpo del Servicio* se diseñó para que las aplicaciones cargadas al servidor (Cloud Robotic) posean la menor cantidad de modificaciones, con respecto a las aplicaciones que se desarrollan para ejecutarse en el robot o mediante bluetooth.

Definida la arquitectura, establecidas sus bondades y asignadas las tareas a cada una de las componentes del modelo, se comenzó a trabajar sobre el desarrollo de la aplicación para el smartphone e implementar el Cloud Robotic. A continuación se describe como se realizaron cada uno de estos pasos.

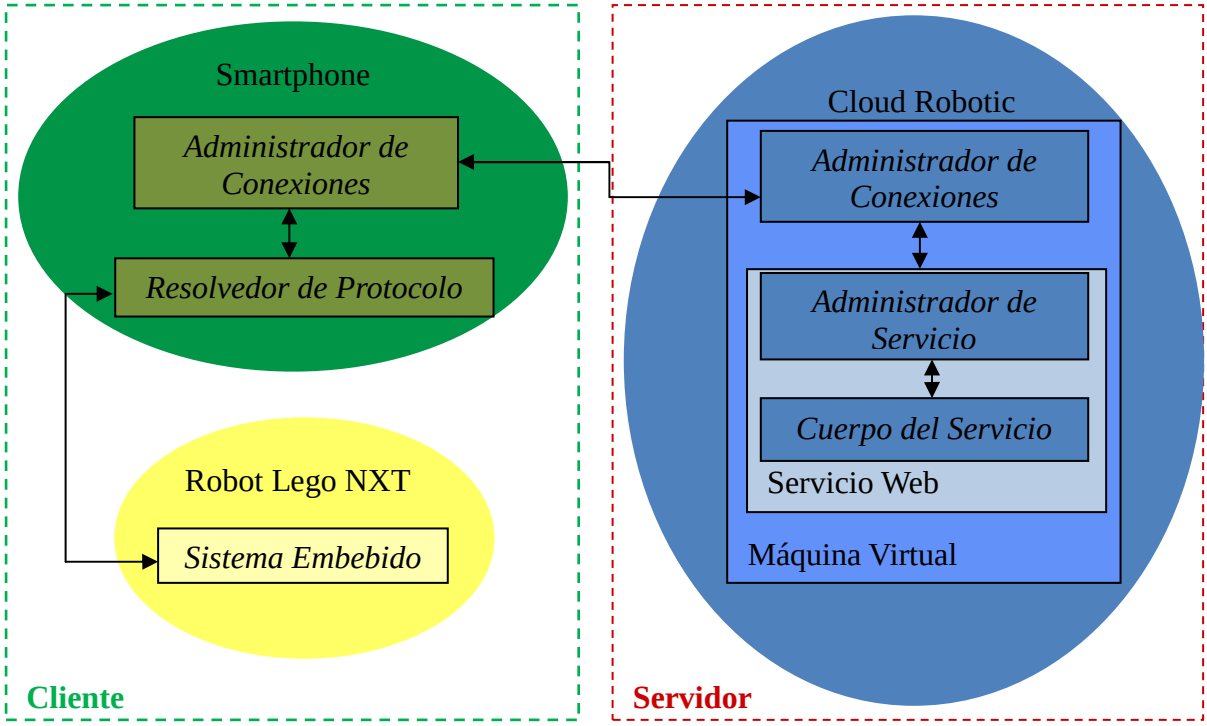

Figura 5: Arquitectura Cloud-Robotic

4.3 Desarrollo Aplicación Smartphone

Para llevar a cabo esta actividad, se utilizó el IDE Eclipse[5] con el lenguaje de programación Java.
A su vez se utilizaron bibliotecas Android[6] para el diseño de la aplicación en el teléfono móvil y la plataforma LeJOS para que esta última pueda mantener una comunicación con el robot Lego nxt, mediante el uso del conector bluetooth.
El aplicativo desarrollado cuenta con una interfaz que permite seleccionar entre las opciones de ejecutar un programa, subir su aplicativo al cloud robotic, ayuda e información acerca del mismo.
El software implementado para el smartphone contiene los módulos *Administrador de Conexiones* y *Resolvedor de Protocolo* cuyas funcionalidades son:

- Establecer/cerrar conexiones con el robot Lego nxt mediante la clase *NXTConnector* que es el adaptador de LeJOS para la comunicación bluetooh.
- Establecer conexión con el servidor: la comunicación con el servidor se realizó por medio de socket.
- Recepción de mensajes: se desarrolló la clase *SelectorPaquetes* la cual determina el tipo, el destinatario y la necesidad de respuesta del mensaje, por medio de cierta información del protocolo que se encuentra en el paquete recibido.

Esta última funcionalidad fue de vital importancia, debido a la necesidad de implementar un protocolo de intercambio de mensajes para lograr la comunicación entre el robot y el servidor. Para llevar a cabo el envío y recepción de notificaciones se definieron los mensajes: *iniciarAplicacion*, *detenerAplicacion*, *enviarRespuestaInstruccion* y *enviarInstruccion* con el formato de paquetes descripto en la figura 6, el cual incluye los campos: *longitud* contiene la cantidad de bytes del paquete; *cabecera* determina el tipo de mensaje enviado (00 *iniciarAplicación*, 01 *detenerAplicación*, etc); *información* posee los datos enviados por LeJOS entre los cuales se destaca el destinatario y la necesidad de respuesta.

[5] http://eclipse.org/
[6] http://developer.android.com/sdk/index.html

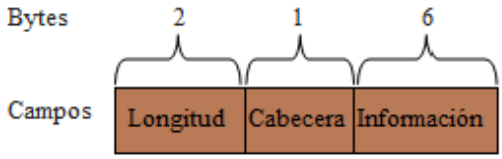

Figura 6: Formato de mensajes del protocolo de comunicación

4.3 Implementación Cloud Robotic

El cloud robotic fue pensado como un conjunto de servicios que pueden ser de interés para un robot Lego Mindstorm tales como navegar un laberinto, generar un mapa de un laberinto o mover un brazo de robot entre otras muchas aplicaciones. Para que un robot pueda acceder a estas aplicaciones se desarrolló un servicio web por cada una de ellas. La figura 7 muestra el diseño del cloud robotic. Cuando el servicio es invocado (método init()), el mismo ejecuta la aplicación correspondiente (run()) y establece la conexión entre el robot y la aplicación solicitada.

Al momento de realizar el servidor, se analizaron las características necesarias que debía poseer para brindar el acceso a los servicios. Dentro de estos aspectos, se determinó que debía poseer todos los privilegios para modificar ancho de banda, utilización de puertos, creación y gestión de hilos, entre otros. Es por ello que se optó por desarrollar el servicio en una PC dedicada para poder contar con todas las características mencionadas.

A continuación se describe el estado actual del mismo.

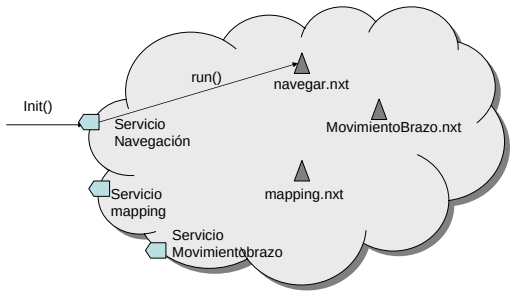

Figura 7: Servicios y aplicaciones en el Cloud-robotic

Para el desarrollo del servidor se utilizó Java en conjunto al IDE Eclipse, por lo que el entorno de máquina virtual sobre el que se encuentra el mismo hace referencia a la máquina virtual de java.

Por otro lado, para implementar la funcionalidad del módulo *administrador de conexiones* del servidor, se utilizó la clase *ServerSocket* para que se escuche en un puerto bien conocido. A su vez por cada petición que reciba, se crea un hilo, el cual es el encargado de responder a las solicitudes de servicio que posea el cliente. Este hilo es de la clase *HiloServidor* y es el encargado de implementar la funcionalidad del módulo *Administrador de Servicio*, para lo cual emplea las siguientes clases:

- *EjecutorAplicaciones*: Se encarga de interactuar con los servicios solicitados y ejecutarlos.
- *ProcesadorPaquetes*: Define el tipo de mensaje recibido y a partir del mismo selecciona la acción a llevar a cabo.
- *NXTCommInternet*: Realiza el envío de las instrucciones del servicio mediante el empleo de un socket.

Esta última clase implementa la interface *NXTCommRequest* de LeJOS que es la utilizada por la API para realizar las comunicaciones remotas. Este hecho permitió que la transferencia de instrucciones se realice de forma transparente, ya que a las aplicaciones subidas al servidor no hubo que modificarlas.

Para implementar el módulo *Cuerpo del Servicio* se diseñó la clase *AplicacionAbstracta*, la cual le otorga el formato a todas las aplicaciones que se encuentran en el cloud robotic. Esta clase, es abstracta y posee un método *run()* el cual se debe redefinir en cada uno de los servicios. En este método se debe incluir el código de la aplicación que vaya a ser accedida remotamente, la única restricción con que se cuenta, es que debe eliminarse cualquier conector (de LeJOS) que se utilice para comunicar la aplicación con el robot de forma remota. Esta restricción se utiliza para que todos los servicios disponibles se accedan de la misma forma, logrando transparencia para el usuario. Otra

de las razones por la cual se impuso esta restricción es que en la arquitectura planteada, la clase *EjecutorAplicaciones* posee un conector, por lo que, si existiera otro adaptador para realizar la misma tarea, la plataforma LeJOS no podría determinar cuál de los dos utilizar.
En la Figura 8 se muestra un diagrama de clases identificando las relaciones que poseen las clases de los módulos *administrador de servicio* y *de conexiones*.

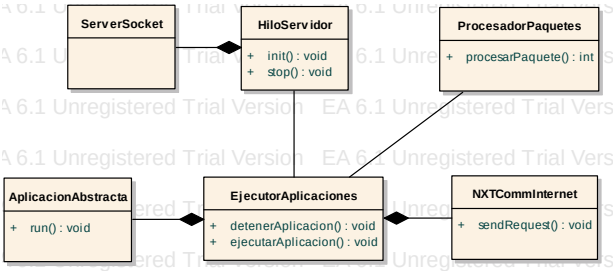

Figura 8: Diagrama de clases

Resultados

Con el objetivo de realizar la arquitectura planteada, se desarrollaron una serie de pruebas intermedias para cumplir con los distintos pasos a lo largo del desarrollo. A continuación se detalla cada uno de los resultados obtenidos.
La primera tarea desempeñada fue desarrollar aplicaciones que funcionen sobre el robot, por lo cual se implementaron dos aplicativos: *Seguir la Linea* y *No Te Caigas*. El primer programa, permite al robot navegar siguiendo una línea negra (que percibe a través de un sensor de color) y variar su velocidad según si el color detectado es azul (acelerar) o amarillo (frenar). El segundo programa, brinda al robot de la capacidad de navegar sin caer al suelo, la misma se logró utilizando un sensor infrarrojo el cual mide la distancia al suelo, superado un cierto umbral de distancia, el robot detectaba que estaba por caer y cambiaba su rumbo.
Para probar la conectividad remota (vía bluetooth), fue necesario construir una aplicación en android que pudiera enviar comandos al dispositivo nxt. Este control puede también usarse utilizando el giróscopo provisto por el smartphone.
Luego, se seleccionó un modelo de arquitectura que de soporte a la ejecución de aplicaciones remotas y a la interconexión necesaria entre los diferentes artefactos que intervienen. Se tuvo en cuenta en esta selección que se trataba de aplicaciones distribuidas. Se desarrolló un diseño preliminar para representar la arquitectura que permite alcanzar los requerimientos planteados. La figura 5 muestra la arquitectura definida para dar soporte al conjunto de servicios para robots Lego basada en el concepto de cloud. El cliente en este caso va a ser un robot que tiene un sistema embebido, es decir aquel encargado de administrar sensores y motores. A su vez, el cliente también está conformado por el smartphone, el cual posee el módulo administrador de conexiones que es el encargado de establecer las conexiones para acceder al servicio web y de enviar las instrucciones al robot, mientras que el módulo resolvedor de protocolo se encarga de resolver la transferencia de mensajes utilizados por el robot y el servicio que se accede.
En cuanto al módulo Cloud Robotic esta integrado por una máquina virtual que asegura la ejecución del servicio para robots (dentro de la plataforma Java) y un administrador de conexiones, que permite comunicar al servidor con el cliente. Por otro lado, el servicio web por su parte esta formado por dos módulos: administrador de servicio el cual se encarga de administrar las características del mismo (ancho de banda, velocidad de procesamiento, uso de CPU, entre otros), y el cuerpo del servicio que es la aplicación del robot que se ejecuta en la nube.

Discusión

El trabajo realizado es un modelo conceptual de base, para en un futuro trabajar con una plataforma totalmente implementada que permita el acceso concurrente de múltiples robots (y/o agentes que trabajen con la plataforma

LeJOS y Java) a distintos servicios web. Si bien en la presentación realizada, los servicios que se encuentran en el cloud robotic son muy simples, se espera como trabajo futuro contar con aplicaciones que permitan controlar dispositivos hogareños a distancia (dotándolos de esta forma, de cierta inteligencia).

Los resultados obtenidos hasta el momento son satisfactorios aunque aún son preliminares y parciales, debe notarse que para lograr una arquitectura que cumpla con todos los requisitos planteados es importante que se tenga total autoridad en la máquina que aloje al servicio web, dado que de lo contrario podrían surgir diferentes inconvenientes en cuanto a la administración de las conexiones y la calidad de servicio prestada. Hasta el momento se desarrollo este servidor en una máquina interna del laboratorio de IA accediendo a través de una LAN.

Conclusión

En este trabajo se presenta un modelo conceptual de una aplicación distribuida que es el punto de partida para el desarrollo de la misma. A partir de este modelo fue posible comenzar a desarrollar y probar módulos considerados como fundamentales dado que de ellos depende la viabilidad del proyecto. Se logró con éxito dotar de capacidades de conexión al robot con tiempos de latencias aceptables. Por otro lado se trabajó sobre el envío de órdenes a los motores del robot como así también la obtención de datos de los sensores. Se desarrollaron aplicaciones del robot las cuales fueron accedidas a través de la invocación de servicios web, dotando de la funcionalidad de carga dinámica de aplicaciones al robot. Se espera en lo inmediato avanzar sobre la estandarización de mensajes y las instrucciones del robot.

Agradecimientos

Los autores quieren agradecer a la Facultad Regional Santa Fe de la Universidad Tecnológica Nacional que hace posible el financiamiento de estas investigaciones a través del proyecto PID 25/O128.

Referencias

[1] Fumio, K., Ikuo, M., Kotaro, K., Youhei K., Masayuki I., Hirochika, I. Development of a remote-brained humanoid for research on whole body action. En Proceeding of IEEE international conference on robotics and automation. Leuven, Belgium. 1998.

[2] Masayuki, I., Fumio, K., Satoshi K., Hirochika I. Vision-equipped apelike robot based on the remote-brained approach. En proceeding of IEEE international conference on robotics and automation. 1995.

[3] Guoqiang, H., Wee, P., Yonggang, W. Cloud robotics: architecture, challenges and applications". Network IEEE vol 26(4). 2012.

[4] Widyawardana, A., Adrianto R. Service oriented architecture in robotic as a platform for cloud robotic (Case study: human gesture based teleoperation for upper part of humanoid robot). En Proceeding of IEEE international conference on cloud computing and social networking. Pg. 1-4. 2012.

[5] Google. Google cloud robotics. Disponible en web: http://googlemonthly.com/google-directions/google-ia-2011-cloud-robotics.html. Mayo 2012.

[6] Armbrust, M., Fox, A., Griffith, R., Joseph,A., Katz, D., Konwinski, A., LEE, G., Patterson, D., Rabkin, A., Stoica, I. and zahariaa, M. (2010) View of cloud computing. Magazine Communications of the ACM, Volume 53 Issue 4, Pp 50-58

[7] Miller, M. (2009). Cloud computing. Web-based applications that change the way you work and collaborate on line. Que publisher. ISBN: 978-0-7897-3803-5

[8] Papazoglou, Michael. Web services: Principles and technology. Prentice Hall. ISBN: 9780321155559. 2008.

[9] Savu, Laura. "Cloud Computing. Deployment models, delivery models, risks and research challenges". Proceeding IEEE International conference on computer and management. 2011, mayo, Pg. 1-4. 10.1109/CAMAN.2011.5778816

[10] Mell, P., Grance, T. The NIST definition of cloud computing. Recommendation of the national Institute of Standards and technology. Special publication 800-145 NIST. 2011.

[11] Yong, N., Chang L., Xing, K., Zhang. "Connectivity as a service: Outsourcing enterprise connectivity over cloud computing environment". Proceeding 2011 international conference on computer and management. 2011, mayo. Pp. 1-7 10.1109/CAMAN.2011.5778899

[12] Chen, Yinong, Du, Zhihui, García Acosta, Marcos. "Robot as a service in cloud computing". Proceeding Fifth IEEE International sympsium on service oriented system engineering. 2010, junio. Pg 151-158. 10.1109/SOSE.2010.44.

[13] LeJOS, Java for LEGO Mindstorms. Url: http://lejos.sourceforge.net/nxj.php

[14] Russell, S.; Norvig, P. ”Artificial inteligent: a modern approach. Third edition” Ed. Prentis Hall. (2010)
[15] Kai Qian, Chongwei Xu, Xiang Fu, Lixin Tao:, Connectivity over cloud computing environment”. Software Architecture and Design Illuminated. Jones and Bartlett Publishers Proceeding 2011 international conference on computer and management. 2011, mayo. Pp. 1-7 10.1109/CAMAN.2011.5778899. 2010.
[16] Shaw, M.; Garlan, D.: Software Architecture, Perspectives on an Emerging Discipline. Prentice-Hall(1996).

Datos de Contacto:

Juan Leonardo Sarli. Universidad Tecnológica Nacional Facultad Regional Santa Fe. Centro de I+D CIDISI. Lavaise 610. Santa Fe. Argentina. juanleonardosarli@gmail.com.
Armando Andini, Universidad Tecnológica Nacional Facultad Regional Santa Fe. Centro de I+D CIDISI. Lavaise 610. Santa Fe. Argentina. ArmandoAndini@gmail.com
Ma. de los Milagros Gutierrez. Universidad Tecnológica Nacional Facultad Regional Santa Fe. Centro de I+D CIDISI. Lavaise 610. Santa Fe. Argentina. mmgutier@frsf.utn.edu.ar