Desarrollo de una arquitectura de software para el robot móvil Lázaró

Development of a software architecture for mobile robot Lázaró

Jesús M. García^{1*} Ángel E. Gil¹ Ender A. Sánchez¹

Recibido 23 de agosto de 2016, aceptado 26 de junio de 2017

Received: August 23, 2016 Accepted: June 26, 2017

RESUMEN

Este artículo describe el desarrollo de una arquitectura de software para el robot móvil Lázaró, desarrollado en el Laboratorio de Prototipos de la Universidad Nacional Experimental del Táchira. La arquitectura propuesta posee tres niveles, un nivel base que gestiona el uso de los distintos elementos del robot a bajo nivel. Un segundo nivel de desarrollo formado por un conjunto de librerías que permiten generar aplicaciones para el control del robot en conjunto con el nivel base. Por último, un nivel de interfaz que permite el control y programación de Lázaró a alto nivel. En esta capa se le brinda al usuario una interfaz que consta de un panel de control directo y un simulador 3D, donde se pueden observar las acciones que ejecutan los actuadores, además se puede monitorear la información sensorial y se pueden programar las instrucciones a ser ejecutadas por el robot móvil. Finalmente, se presentan varios casos de prueba para verificar las funcionalidades y capacidades de esta arquitectura desarrollada en C#.

Palabras clave: Arquitectura de software, robot móvil, comportamiento, intérprete, simulador 3D.

ABSTRACT

This article describes the development of software architecture for the mobile robot Lázaró, developed at the Laboratorio de Prototipos of Universidad Nacional Experimental del Táchira. The designed architecture has three levels: a basic level that manages the use of the various elements of a robot at a low level. A second level of development consisting of a set of libraries that allows generating robot control applications together with the base level. Finally, interface level that allows control and programming of Lázaró at a high level. This layer provides the user with an interface consisting of a direct control panel and a 3D simulator, where the user can observe the actions that running the actuators; also the user can monitor sensory information and you can program instructions to be executed by the mobile robot. Finally, several test cases are presented to verify the functionality and capabilities of this architecture developed in C#.

Keywords: Software architecture, mobile robot, behaviour, interpreter, 3D simulator.

INTRODUCCIÓN

Una arquitectura de software comprende la estructura, el funcionamiento e interacción entre

las diversas partes del software y como tal, estas pueden aplicarse en el desarrollo de sistemas de diversas áreas [1]. Específicamente, para el caso de los robots existen fundamentalmente tres

¹ Laboratorio de Prototipos. Decanato de Investigación. Universidad Nacional Experimental del Táchira. Av. Universidad, sector Paramillo, San Cristóbal, Estado Táchira, Venezuela. E-mail: jmgarcia@unet.edu.ve; agil@unet.edu.ve; enderabelardo@gmail.com

* Autor de correspondencia.

grupos de arquitecturas que engloban el universo de posibilidades: la arquitectura deliberativa o jerárquica, en la que todas las funciones y módulos actúan de manera secuencial [2-3]; en este caso, es común observar tres módulos: percepción, planeación y acción. Posteriormente aparece la arquitectura reactiva o basada en comportamientos [4] esto propone métodos reactivos de respuesta que son especialmente útiles cuando no se conoce completamente el entorno donde actúa un robot. Estos dos grupos de arquitecturas poseen ciertas ventajas: en el caso de la arquitectura deliberativa, permite al robot una rápida actuación en un entorno ya estudiado, mientras que la arquitectura reactiva, permite al robot responder frente a imprevistos que puedan surgir durante la ejecución de sus tareas [5, 6]. Adicionalmente, aparecen arquitecturas híbridas, que toman características de los dos grupos anteriores para lograr una mejor respuesta en el robot al momento de ejecutar las tareas [7, 8].

Adicionalmente al paradigma adoptado al desarrollar una arquitectura, se han implementado herramientas de inteligencia artificial con el propósito de hacer más eficiente al robot frente a las tareas a efectuar. En el área de la robótica móvil, el problema de la navegación en entornos no estructurados presenta continuas mejoras al hacer uso de estas herramientas. En [9] se presenta una arquitectura reactiva que utiliza redes neuronales y un controlador difuso para reconocer el medio ambiente y reaccionar a problemas que se presenten al robot (Ejemplo: presencia de obstáculos). En [7] se presenta una arquitectura híbrida que utiliza un controlador difuso de velocidad cuyas entradas son la posición y distancia respecto a los obstáculos ubicados alrededor del robot. En [10] también se presenta una arquitectura híbrida con un controlador difuso que maneja un comportamiento destinado al seguimiento de paredes.

La selección de un tipo particular de arquitectura estará asociada a las características físicas y funcionales del robot donde se van a implementar. Por ejemplo, en el desarrollo del robot social *Maggie*, se implementaron un conjunto de herramientas para conformar una arquitectura de control que permitiese lograr el control efectivo y eficiente de los componentes lógicos y físicos que conforman el robot mediante la implementación de un paradigma automático deliberativo [11]. Otra

aplicación relacionada es *ERBPI* (*Easy Robot Behavior Programming Interface*), una interfaz de programación para robots móviles que no requiere del usuario ninguna experiencia en programación ni en robótica. Para lograr esto, se adoptó un enfoque basado en comportamientos. De esta forma, la nueva aplicación hizo uso del paradigma conexionista, donde los comportamientos se definen estableciendo conexiones configurables entre los sensores y actuadores del robot [12].

Para ser más efectiva, una arquitectura debería poder ser utilizada por distintos usuarios para diferentes aplicaciones, para una o más plataformas. Además, debe ser factible la incorporación de subsistemas, proporcionando así un desarrollo evolutivo sin afectar la funcionalidad ya lograda [13]. Por ello, una tendencia reciente apunta al diseño de arquitecturas abiertas [14] y desarrolladas de acuerdo a estándares de programación que permitan su reutilización. Este es el caso de [15] donde se desarrolla una arquitectura usando el *Lenguaje Unificado de Modelado (UML)* y la *Metodología de Desarrollo de Arquitecturas Software para Robots (MDASR)*. En [16] se diseña una arquitectura para un robot móvil que usa el enfoque *Siemens 4* para el diseño de arquitecturas de sistemas de software.

A partir de estas experiencias se afirma que, una arquitectura de software para robots debe satisfacer los requerimientos de los usuarios y también cumplir con la integración de los componentes físicos y lógicos de un robot. Bajo estas premisas, se generó esta investigación dirigida a desarrollar una arquitectura de software para el control del robot móvil Lázaro disponible en el Laboratorio de Prototipos de la Universidad Nacional Experimental del Táchira.

De manera que, este artículo describe una arquitectura de software para el control de los diversos sensores y actuadores de este robot, completamente escalable y adaptable, ya que permite crear y adicionar módulos jerárquicos y/o comportamientos reactivos al sistema robot de manera sencilla utilizando un lenguaje de programación estándar de amplia difusión. Además, posee una interfaz de usuario que permite modificar el modo de operación del robot (teleoperación o autonomía de algún actuador) y probar algún comportamiento o acción en un simulador 3D antes de la ejecución real de la tarea programada.

DESARROLLO

Caso de estudio: el robot Lázaro

Lázaro, es un robot móvil tipo *Skid Steer* [17], diseñado para exploración sobre terrenos de superficie irregular (ver Figura 1). Cuenta con cuatro ruedas traccionadas por dos motores DC (cada motor mueve las 2 ruedas del mismo lado). Además, cada rueda posee amortiguación pasiva independiente para amoldarse a las irregularidades del terreno [18]. Las dimensiones del vehículo son 425 mm de ancho, 468 mm de longitud y 252 mm de altura; su masa es 26 Kg. Cuenta con una Unidad de Medición Inercial (IMU) por medio de esta se pueden medir los ángulos *roll* (α) y *pitch* (φ) respecto al plano horizontal. La velocidad máxima de avance que puede generar el vehículo es 0,28 m/s obtenida a partir de codificadores angulares instalados en los motores de tracción [19]. Adicionalmente, cuenta con un brazo que posee 2 articulaciones: una rotacional para el giro del primer eslabón del brazo cuyo ángulo de rotación (θ_1) es medido a partir de un codificador absoluto, y una lineal para obtener el desplazamiento vertical (d_2) del segundo eslabón del brazo el cual es medido por un potenciómetro instalado en un actuador lineal que forma el segundo eslabón (ver

Figura 2). Como efector final, Lázaro posee una rueda pivotante que permite establecer un contacto rodante entre el brazo y el suelo a medida que el robot se desplaza o se mueve el brazo.

Figura 1. Robot Lázaro.

Adicionalmente, posee un telémetro láser que está ubicado al frente de chasis robot y realiza el barrido láser de manera vertical que permite después de un análisis, detectar obstáculos ubicados frente al robot, estimar su posición y dimensionar el obstáculo. Esta

Figura 2. Diagrama de Lázaro mostrando la ubicación de los sensores y las variables articulares del brazo θ_1 y d_2 .

información puede complementarse por medio de cuatro sonares auxiliares (dos apuntando al frente y dos hacia abajo, ver Figura 2) que también se utilizan para detectar obstáculos y depresiones del terreno, además de un sensor infrarrojo ubicado al frente del brazo y apuntado hacia abajo que cumple la misma función. Finalmente, el robot tiene un sensor de fuerza alojado en la proximidad del efector lineal del brazo, el que mide la fuerza de contacto entre la rueda pivotante y el suelo.

En cuanto al control del robot, este se ejecuta a distancia desde un computador remoto ya sea de manera autónoma o teleoperada mediante un Joystick. Por tanto, la comunicación entre el robot y este computador se lleva a efecto por medio de módulos XBee®. Adicionalmente, en el robot se instalaron dos microcontroladores (PIC): uno destinados a recolectar y codificar la información percibida por los sensores (salvo el telémetro láser), y el otro destinado a recibir las consignas de actuación provenientes del computador y codificarlas para ser enviadas a tres tarjetas controladoras de los actuadores (TRex). Todos estos periféricos están organizados conforme a la configuración de hardware mostrada

en la Figura 3. En el caso del telémetro, se encontró que el volumen de datos que este entrega es muy grande, por lo que se acopló un ordenador portátil sobre el robot, el que tiene como función, recibir la información del telémetro, procesarla, y entregar, como salida, información relacionada a algún tipo de obstáculo encontrado frente al robot (si lo hay).

Descripción de la arquitectura propuesta

Para el desarrollo de la arquitectura de software del robot móvil Lázaro se seleccionó el entorno de desarrollo *Visual Studio 2012* con el lenguaje de programación C#, que ofrece una compleja variedad de librerías como son: Puerto serial (*SerialPortClass*), Componentes gráficos (*WindowsForms*) y entornos 3D *Microsoft Xna* ideales para este trabajo. Además el IDE de desarrollo posee un conjunto de herramientas que permiten la documentación de código y la utilización de diagramas del lenguaje unificado de modelo UML. Utilizando estas herramientas se desarrolló una arquitectura de software con tres niveles: nivel base, nivel desarrollo y una interfaz gráfica de control (ver Figura 4). Estos niveles se explican a continuación.

Figura 3. Configuración de hardware.

Figura 4. Arquitectura de software desarrollada para LázarO.

Nivel base

Es el nivel de comunicación entre el robot móvil y el computador, se encarga de gestionar la comunicación de los componentes del hardware con el software, así mismo posee un conjunto de clases con métodos para el control y monitoreo de los diversos dispositivos de LázarO (Actuadores y Sensores). Esta capa sigue el patrón y principios de la arquitectura de software Modelo – Vista – Controlador.

El nivel base tiene un flujo de datos representados en la Figura 5, su controlador principal (Control LázarO) gestiona las instrucciones de control (Actuadores) y la obtención de datos (Sensores) mediante los módulos de comunicación serial, conectados a los dispositivos XBee® del computador los que están enlazados con los XBee® del robot móvil. Esta comunicación permite controlar directamente los motores del robot mediante el PIC de control y el TRex, como asimismo obtener los datos de los sensores y de la unidad inercial.

El flujo de datos desde el robot al ordenador remoto se realiza or medio de dos tramas de datos: la primera (entre el XBee® Control LázarO y XBee® Control Pc; ver Figura 5) está estructurada por 22 bytes

Figura 5. Diagrama de flujo de datos. Nivel base.

que incluyen el conteo de pulsos de los motores izquierdo y derecho (tracción), sentido de giro de estos motores, lecturas de los sonares ubicados en diferentes posiciones del robot, potenciómetro y sensor de fuerza ubicados en el brazo; y lecturas de tres codificadores dedicados a medir la posición angular del brazo y de dos remolques adicionales (el robot está diseñado para la conexión de dos remolques).

La segunda trama (entre el XBee® IMU Lázaro y XBee® IMU Pc; ver Figura 5) está estructurada por 16 bytes utilizados para enviar datos obtenidos a partir de la IMU: aceleración en dirección “x”, “y” y “z”, además de los ángulos de cabeceo (*pitch*), alabeo (*roll*) y guiñada (*yaw*). Desde el ordenador remoto se envía información al robot por medio de una única trama de 7 bytes (entre el XBEE Control Pc al XBEE Control Lázaro, ver Figura 5) que permite enviar de manera codificada, las órdenes para controlar los motores izquierdo-derecho (tracción) y los dos actuadores del brazo.

Adicionalmente se diseñó para esta capa, un conjunto de clases para el control y manejo de datos de los diversos sensores, actuadores y dispositivos del robot móvil Lázaro, de manera que se dispone de una librería de clases (.dll) lo cual incrementa la robustez de la capa. A continuación se listan las clases desarrolladas:

- Clase *SerialComunicacion*: Es una clase de comunicación encargada de gestionar la entrada y salida de datos en una conexión bidireccional (dúplex) o unidireccional entre los dispositivos XBEE y la arquitectura de software. En su estructura se incluye el método *EnviarDatos* que gestiona la transmisión desde el computador al robot utilizando tramas que se envían cada 150 milisegundos si cumple las siguientes condiciones: que el puerto serial este abierto, si el envío de datos está activo y si el canal de envío se encuentra disponible. A su vez, el método *RecibirDatos* controla la recepción de datos en la comunicación serial.
- Clase *Evento*: Es una clase de comunicación encargada de gestionar el flujo de datos e indicar cuándo se ejecuta una acción o ocurre un hecho notificando entre las diversas clases que componen la arquitectura de software.
- Clase *UnidadInercial*: Clase encargada de almacenar en memoria los datos generados por la IMU en cada tiempo de respuesta. Específicamente, es responsable de gestionar la variación de aceleración en el espacio y los ángulos de inclinación del robot móvil.
- Clase *Sensor*: Clase genérica encargada de asignar los datos de los sensores que responden 2 bytes (16 bits) o menos. Esta clase gestiona la información acerca de los sonares, sensores de presión, codificadores absolutos, entre otros.
- Clase *Encoder*: encargada de almacenar los pulsos de los codificadores incrementales localizados en los motores de tracción. También posee atributos que le permiten indicar hacia donde están girando los motores revelando los estados hacia adelante, atrás o detenido.
- Clase *Odometría*: encargada de calcular la posición y orientación del robot en el plano por medio de los codificadores incrementales de tracción.
- Clase *Acción*: Es una clase de comunicación e interpretación de instrucciones que gestiona la trama enviada al robot móvil, controlando de esta manera directamente a los actuadores, por medio de instrucciones formadas por un conjunto de datos que se envían con una secuencia estructurada o específica a la unidad de control de los actuadores, la que interpreta y ejecuta las órdenes. En esta clase se produce el envío de datos tanto de una manera secuencial o creándose hilos de ejecución controlables.
- Clase *ControlActuadores*: Clase que se encarga de gestionar los procesos de ejecución de acciones de los diversos actuadores con los que cuenta Lázaro. La clase posee un constructor y contiene 17 métodos públicos de control que al ser llamados emiten las instrucciones para la gestión del actuador (ver Tabla 1). Los métodos según los parámetros que reciban pueden realizar la gestión de instrucciones secuencialmente o multiproceso. Por ejemplo, el método *BrazoGiroDer* (80, 10.000, 0), de acuerdo con los atributos especificados permite el giro de la primera articulación del brazo hacia la derecha con una potencia de 80%, una duración

de 10.000 ms y en una ejecución secuencial (ver Tabla 1). Las funciones que actúan en esta clase al ser invocadas en conjunto permiten ejecutar las rutinas y comportamientos de movimiento que posee Lázaró.

- Clase *ControlSensores*: Clase para el control de los sensores, encargada de almacenar los datos de los diversos sensores del robot móvil. Se asocia con las clases *Sensor*, *Encoder* y *UnidadInercial*.
- Clase *ControlLázaro*: Clase principal de la librería Lázaró encargada de gestionar la capa base de la arquitectura de software. Contiene los objetos de las clases de comunicación, control actuadores y control de sensores en el diagrama de clase.

Nivel desarrollo

Es el encargado de la creación e incorporación de posibles comportamientos, movimientos, destrezas y rutinas, que podrán ser programados con las clases especificadas en la capa base. Este nivel intermedio es el gestor y responsable de las rutinas creadas para Lázaró por medio de una clase llamada *ComportamientoLázaro* la que posee atributos y métodos encargados de gestionar acciones, interpretación de instrucciones y de ejecutar comportamientos con el uso de las diversas funcionalidades que posee el nivel o capa base.

Para facilitar la creación de rutinas, el nivel de desarrollo posee un intérprete el que es un

método capaz de analizar y ejecutar instrucciones programadas por el usuario. Adicionalmente, reconoce el lenguaje propio de la arquitectura de software (palabras reservadas que fueron creadas con expresiones regulares en el código). El método intérprete ejecuta un flujo de datos como se observa en la Figura 6, recibe un *String* (instrucción programada por el usuario) donde pasa por un análisis léxico mediante el uso de expresiones regulares e indica si el código o conjunto de instrucciones son válidos o no. Al reconocer las instrucciones válidas selecciona el actuador que ejecutará la instrucción y se analiza la potencia, el tiempo y la acción que el mismo realizará. Después de analizar la instrucción y validarla, el intérprete traduce e invoca al método respectivo en el nivel base, según el actuador.

Interfaz gráfica

La interfaz gráfica tiene como principal uso, proporcionar un entorno visual al usuario para la comunicación e interacción con las capas inferiores de la arquitectura [20] (ver Figura 7). Al respecto, se desarrolló una GUI accesible y de fácil uso la que se fundamenta en el uso de paneles de control y monitoreo. Específicamente, está formada por una pantalla principal (ver Figura 8) con cuatro paneles (Control directo, Intérprete, Simulador, Sensores y Consola), estos se describen a continuación.

Panel Simulador 3D (ver Figura 8, rectángulo superior izquierdo): El simulador tiene como objetivo principal mostrar los posibles movimientos que ejecutará el robot móvil por medio de las instrucciones dadas por los usuarios, brindando una imagen de estas

Tabla 1. Algunos métodos de la clase *Control Actuadores*.

Método	Parámetros del método			Actuador
	Potencia	Tiempo	Ejecución	
BrazoGiroDer	Magnitud de potencia en el giro brazo hacia la derecha. Entre (1-100).	ms	0 - 1	Brazo
LinealArriba	Magnitud de potencia en el actuador lineal hacia arriba. Entre (1-100).	ms	0 - 1	Actuador lineal
MotorIzqAdel	Magnitud de potencia en el motor izquierdo de tracción hacia adelante. Entre (1-100).	ms	0 - 1	Motor izq.
MotorDerAtras	Magnitud de potencia en el motor derecho de tracción hacia atrás Entre (1-100).	ms	0 - 1	Motor der.
MoverAdelante	Magnitud de potencia en ambos motores de tracción hacia adelante. Entre (1-100).	ms	0 - 1	Motor izq. Motor der.

La ejecución toma el valor de 0 si se espera que sea secuencial y 1 si la acción será multiproceso.

Figura 6. Diagrama de flujo de datos en el intérprete.

acciones mediante múltiples vistas. La construcción del modelo 3D, se realizó utilizando el software Solid Edge®. Posteriormente, se exportó el modelo 3D a Blender donde, a partir de las transformaciones de escala, rotación y traslación se programó el movimiento simulado del robot de acuerdo con las instrucciones emitidas por el usuario.

Panel Sensorial (ver Figura 8, rectángulo superior derecho): Este panel organiza en un conjunto de tablas, las lecturas tomadas a partir de todos los sensores que posee el robot móvil Lázaró. La gestión del panel sensorial depende de la invocación de los eventos *Imu* y *Sensores* localizados en la capa base, encargados de extraer y mostrar en la interfaz los datos de las lecturas de los sensores almacenadas en el nivel base de la arquitectura y, a su vez, actualizarlos cada vez que llega un nuevo dato por medio de las comunicación establecida con el robot utilizando los XBee®.

Panel Intérprete-Consola (ver Figura 8, rectángulo inferior izquierdo): El panel Intérprete-Consola incorpora dos secciones que son el intérprete y la consola. El intérprete está compuesto por un editor donde el programador codificará las instrucciones que se ejecutarán tanto secuencialmente o de manera multiproceso. El intérprete, mediante la capa desarrollo, utiliza el método llamado *IntérpreteTexto*

Figura 7. Flujo de datos en la interfaz gráfica.

que permite interpretar y ejecutar las instrucciones dadas por el usuario. Adicionalmente, la consola tiene como función monitorear el estado en que se encuentra cualquier método de las capas de la arquitectura durante la ejecución de un programa.

Panel Control Directo (ver Figura 8, rectángulo inferior derecho): permite la teleoperación del robot a por medio de cuatro botones encargados de realizar acciones múltiples combinando movimientos de tracción, giro del brazo (1era articulación) y desplazamiento del actuador lineal (2da articulación).

RESULTADOS Y DISCUSIÓN

Se llevaron a cabo pruebas de implementación de la arquitectura diseñada para observar la funcionalidad de todos los niveles. Se especifican los casos de prueba empleados:

- *Prueba 1. Implementación de programas de prueba en la capa base:* utilizando el constructor del nivel base se crearon pequeños programas para realizar la conexión con el robot móvil LázarO y ejecutar movimientos simples de acuerdo al programa.
- *Prueba 2: Implementación de programas de prueba en la capa de desarrollo:* esta capa

depende de la capa base para su funcionamiento. Se creó una aplicación que permite enlazar solo estas dos capas. Posteriormente, se crearon librerías que al ejecutarse desde la capa de desarrollo, permiten la activación de comportamientos sencillos. Ejemplo: *Comportamiento Hola* (el robot ejecuta un conjunto de movimientos brindando la sensación de un saludo cuando detecta la presencia de movimiento a partir de sus sonares).

- *Prueba 3: Funcionamiento de la interfaz gráfica:* esta interfaz incorpora las funcionalidades de las capas inferiores y posee cuatro paneles que se probaron individualmente.

En la prueba del panel de control directo (teleoperación) se indicaron diversas acciones de control a ser ejecutadas por LázarO utilizando para ello los trackbar y botones destinados para tal fin después de haber activado la opción de teleoperación. La Figura 9 muestra tres secuencias de prueba: en la primera, el robot está en reposo. En la segunda, se ordena al robot el avance hacia adelante, mientras que, en la tercera se ordena el giro de la primera articulación del brazo.

Asimismo, se hicieron pruebas del panel intérprete al ejecutar métodos programados (Figura 10).

Figura 8. Interfaz de usuario.

Figura 9. Prueba del panel de control directo.

El objetivo de estas pruebas fue verificar que el intérprete traduzca las instrucciones programadas por el usuario y realice las acciones. Además, en estas pruebas se ejecutaron instrucciones programadas por el usuario controlando el robot móvil tanto en el simulador como físicamente (ver Figura 11). Se buscó corroborar la exactitud del simulador frente a tareas programadas, que permita luego estudiar los movimientos a ejecutar antes que el robot los realice de manera real. Se encontró que el simulador sigue con gran exactitud el movimiento del robot real, en especial el movimiento de brazo acoplado. En el caso del

desplazamiento del robot, la correspondencia no es del todo efectiva pues el modelo cinemático que rige al simulador no toma en cuenta el efecto del deslizamiento en un robot *skid steer* [17]. También se ejecutaron pruebas del panel sensorial, tomando lecturas de las variables mediadas con los sensores instalados y se compararon con las mediciones correspondientes tomadas físicamente con otros instrumentos de medición. Se corroboró una exactitud mayor al 85% en el caso de sensores externos (sonares, IMU) y 90% en el caso de sensores internos (potenciómetro y codificador absoluto).

Figura 10. Prueba del panel intérprete.

- *Prueba 4: Conexión entre el computador remoto y el robot móvil Lázaró:* se encontró que el tiempo de comunicación promedio entre el usuario y el robot, fue de 106.608 ms. Adicionalmente se encontró que la ejecución de comandos por parte del robot

tiene un retardo de 600 ms. Esta información será vital para el desarrollo de algoritmos de control de las articulaciones y para el diseño de comportamientos, en especial aquellos asociados a la navegación reactiva que requieren una rápida respuesta del robot.

Figura 11. Prueba de correspondencia entre el simulador y el robot real.

Las pruebas realizadas sobre la arquitectura desarrollada permitieron, en primer lugar, cuantificar el tiempo de respuesta del robot y también verificar la funcionalidad y usabilidad de los tres niveles de la arquitectura propuesta para lograr una respuesta efectiva del robot ante las órdenes emitidas por el usuario.

Si se requiere evaluar la arquitectura propuesta de manera cuantitativa se deben definir métricas que permitan valorar algunas características importantes. Al respecto, la creación de métricas no ha tenido la mayor relevancia [21], gran parte de las métricas definidas se han hecho en función de algunos parámetros específicos para

alguna tarea en particular, por ejemplo: variables cuantificables en la navegación. En el caso de la evaluación general de una arquitectura es frecuente el uso de la valoración cualitativa. Al respecto, en [8, 22-23] se listan algunos parámetros de evaluación que se utilizarán a continuación para valorar la arquitectura propuesta. La evaluación se hizo en función de opiniones de dos tipos de usuario: expertos y nóveles. Los resultados se listan en la Tabla 2 y se utiliza una escala de cinco valores (muy malo, malo, regular, bueno, muy bueno) (ver Tabla 2).

En general, se observa una arquitectura de fácil utilización y con posibilidad de crecimiento

Tabla 2. Evaluación cualitativa de la arquitectura propuesta.

Característica	Evaluación
Orientación a objetivos definidos.	Bueno
Flexibilidad.	Muy bueno
Facilidad de implementación.	Bueno
Reactividad.	Bueno
Facilidad de planeación.	Bueno
Operación óptima.	Bueno
Aprendizaje de tareas.	Malo
Robustez.	Bueno
Eficiencia en la ejecución de tareas.	Bueno
Extensibilidad o escalabilidad.	Muy bueno
Reestructuración.	Bueno
Tiempo de respuesta.	Regular
Modularidad.	Muy bueno
Acoplamiento a distintas plataformas.	Regular
Mantenibilidad.	Bueno
Facilidad de diagnóstico.	Bueno
Aceptación.	Bueno
Acoplamiento al hardware del robot.	Muy bueno
Propósito general.	Bueno
Control directo de actuadores.	Muy bueno

(escalabilidad) a medida que se implementen nuevos comportamientos y estructuras. Claro está que para programar utilizando la sintaxis desarrollada se requiere un conocimiento del tema, pero aparte de este hecho, la arquitectura se percibe amigable al usuario que puede teleoperar, percibir el estado del robot mediante las mediciones obtenidas con sus sensores, simular el funcionamiento del robot y finalmente programar después de un entrenamiento específico del lenguaje utilizado. Estas características permiten que la arquitectura diseñada sea eficaz para suplir las necesidades de operación requeridas para este robot.

CONCLUSIONES

Se desarrolló una arquitectura de software para el robot móvil Lázaró que permite el control de los actuadores y monitoreo de los sensores mediante un conjunto de librerías implementadas en lenguaje C#. Para su concepción se analizaron los requerimientos de los usuarios, el sistema de control anterior del robot y el estado del arte de diversas arquitecturas de software para robots, posteriormente se propuso una arquitectura de tres niveles para el control del robot móvil Lázaró.

Se construyó una librería de gestión conocida como nivel base, que permite controlar a los actuadores del robot móvil mediante procesos secuenciales o multiproceso, además permite el monitoreo de los sensores disponibles en el hardware del robot móvil Lázaró. Luego, la capa de desarrollo implementada permite la inclusión de nuevos algoritmos o comportamientos de control, lo que facilita el crecimiento del sistema en el futuro. Además, incorpora un intérprete de comandos que reconoce los métodos programados por el usuario mediante un analizador léxico. Se desarrolló una GUI que incorporó las funciones del nivel base y desarrollo, donde el usuario posee paneles de control y programación de instrucciones para el robot, un simulador 3D, monitoreo de información sensorial, códigos de demostración de comportamientos y un panel de control directo que actúa como un joystick virtual para controlar los actuadores del robot móvil Lázaró. Finalmente, se realizaron pruebas en los tres niveles de la arquitectura de software, comprobándose su funcionalidad y una apropiada velocidad de comunicación entre el robot y el ordenador remoto donde se ejecuta el control.

Como trabajos futuros está la creación de más comportamientos en la capa de desarrollo que permitan generar respuestas preestablecidas del robot ante algunos estímulos percibidos desde el entorno. Además está pendiente la incorporación de las clases y métodos que permitan implementar dentro de esta arquitectura, la obtención de la localización más precisa de Lázaró utilizando para ello un telémetro láser y un pequeño ordenador a bordo del robot.

AGRADECIMIENTOS

Este trabajo fue financiado por el Decanato de Investigación de la Universidad Nacional Experimental del Táchira por medio de los proyectos 01-020-2010 y 01-024-2013.

REFERENCIAS

- [1] L. Bass, P. Clements y R. Kazman. "Software Architecture in practice". 3rd ed. Addison-Wesley Professional. 2013.
- [2] D. Nakhaeini, S.H. Tang, S.B. Mohd Noor and O. Motlagh. "A review of control architectures for autonomous navigation of mobile robots".

- International Journal of Physical Sciences. Vol. 6 N° 2, pp. 169-174. 2011.
- [3] A. Ollero, A. Mandow, V. Muñoz and J. Gómez de Gabriel. "Control architecture for mobile robot operation and navigation". *Robotics & Computer-Integrated Manufacturing*. Vol. 11 N° 4, pp. 259-269. 1994.
- [4] R.C. Arkin. "Behavior-based Robot Navigation in Extended Domains". *Journal of Adaptive Behavior*. Vol. 1 N° 2, pp. 201-225. 1992.
- [5] M. Hank and M. Haddad. "Hybrid control architecture for mobile robots navigation in partially known environments". 11th International Conference on Informatics in Control, Automation and Robotics. Vienna, pp. 513-521. 2014.
- [6] N. Olmedo, H. Zhang and M. Lipsett. "Mobile robot system architecture for people tracking and following applications". *IEEE International Conference on Robotics and Biomimetics*. Bali, pp. 825-830. 2014.
- [7] D. Nakhaeinia, P. Payeur, T.S. Hong and B. Karasfi. "A hybrid control architecture for autonomous mobile robot navigation in unknown dynamic environment". *IEEE International Conference on Automation Science and Engineering*, Gothenburg, pp. 1274-1281. 2015.
- [8] A. R. Da Silva and A. Machado. "Control of Mobile Robots with Amorphic Architecture". *IEEE Latin America Transactions*. Vol. 14 N° 7, pp. 3093-3101. 2016.
- [9] S. Nurmaini and B. Tutuko. "A New Control Architecture in Mobile Robot Navigation based on IT2Neuro-Fuzzy Controller". *International Conference on Electrical Engineering and Informatics*, Bandung, pp. 1-7. 2011.
- [10] G. Acosta, J. Gallardo y R. Pérez. "Arquitectura de control reactiva para la navegación autónoma de robots móviles". *Ingeniare. Revista chilena de ingeniería*. Vol. 24 N° 1, pp. 173-181. 2016.
- [11] R. Rivas. "Diseño software de una arquitectura de control de robots autónomos inteligentes: aplicación a un robot social". Tesis para optar al grado de doctor. Universidad Carlos III de Madrid. 2010. URL: <http://e-archivo.uc3m.es/handle/10016/9472>.
- [12] J. Caccavelli. "Nueva interfaz de programación de robots móviles para talleres de Robótica Educativa". Tesis de Licenciatura, Universidad de Buenos Aires. 2012. URL: <http://robotica.dc.uba.ar/index.php/tesis/>
- [13] E. Michalczewsky y P. R. Fillottrani. "Arquitecturas de Software para el diseño de Robots Móviles Autónomos". *III Workshop de Investigadores en Ciencias de la Computación*, Argentina, pp. 1-4. 2001. URL: <http://sedici.unlp.edu.ar/handle/10915/21720>.
- [14] G. Pérez, R. Araguás, D. Gaydou, G. Steiner, y L. Canali. "RoMAA-II, an Open Architecture Mobile Robot". *IEEE Latin America Transactions*, vol. 12, N° 5, pp. 915-921, 2014.
- [15] N. Londoño. "Arquitectura software para robots móviles aplicando la metodología MDASR". *Avances en Sistemas e Informática*. Vol. 6 N° 3, pp. 133-144. 2009.
- [16] A. Muzaffar, S. Mir, M. Latif, W. Butt, y F. Azam. "Software Architecture of a Mobile Robot". *International Conference on Computational Science and Computational Intelligence*, Las Vegas, pp. 102-107. 2015.
- [17] A. Mandow, J. L. Martínez, J. Morales, J. Blanco, A. García-Cerezo y J. González. "Experimental kinematics for wheeled skid-steer mobile robots". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, pp. 1222-1227. 2007.
- [18] C. Porras, J. Porras, J. García y J.L. Martínez. "Diseño de la estructura mecánica de un robot móvil con un brazo para el contacto con el suelo". *V Congreso Internacional de Ingeniería Mecánica y III de Ingeniería Mecatrónica*, Bogotá, pp. 1-10. 2011.
- [19] J. M. García, J. L. Martínez, A. Mandow and A. García-Cerezo. "Caster-leg aided maneuver for negotiating surface discontinuities with a wheeled skid-steer mobile robot". *Robotics and Autonomous Systems*. Vol. 91, pp. 25-37. 2017.
- [20] R. Tabile, E. Godoy, R. Pereira, G. Tangerino, A. Porto and R. Inamasu. "Design and Development of the Architecture of an Agricultural Mobile Robot". *Engenharia Agrícola*. Vol. 31 N° 1, pp. 130-142. 2011.
- [21] F. Correa, J. Gallardo, N. Muñoz y R. Pérez. "Estudio comparativo basado en métricas para diferentes arquitecturas de navegación reactiva". *Ingeniare. Revista chilena de ingeniería*. Vol. 24 N° 1, pp. 46-54. 2016.

- [22] M. Matamoros, J. Savage and J. Ortega-Arjona. "A comparison of two Software Architectures for General Purpose Mobile Service Robots". IEEE International Conference on Autonomous Robot Systems and Competitions, Vila Real, pp. 131-136. 2015.
- [23] U. Sheikh, M. Jamil and Y. Ayaz. "A Comparison of Various Robotic Control Architectures for Autonomous Navigation of Mobile Robots". International Conference on Robotics and Emerging Allied Technologies in Engineering, Islamabad, pp. 239-243. 2014.