ARQUITECTURA HÍBRIDA DE NAVEGACIÓN PARA ROBOT PIONEER PD3X

[*]Saúl Piña [**]Jorge Parra

Octubre 2014

Resumen

En el presente informe se da a conocer de manera detallada el diseño, desarrollo e implementación de una arquitectura de navegación híbrida basada en la arquitectura AuRA para el robot Pionner P3DX, con el objetivo de planificar rutas entre dos puntos cualesquiera en un ambiente conocido. Para ello se usó el Algoritmo de Dijkstra como método para la determinación del camino más corto de un punto a otro. Por otra parte, se segmentó el mapa con la Triangulación de Delaunay, para poder usar el diagrama de Voronoi asociado. Además, se explica el funcionamiento e instalación de la aplicación de software desarrollada. Se exponen los resultados y conclusiones del conjunto de pruebas realizadas para validar la arquitectura.

Palabras clave: Robótica, ARIA, AuRA, Arquitectura Híbrida, Pioneer P3DX.

[*] *Decanato de Ciencias y Tecnología, Universidad Centroccidental Lisandro Alvarado, Barquisimeto, Venezuela, sauljabin@gmail.com*

[**] *Decanato de Ciencias y Tecnología, Universidad Centroccidental Lisandro Alvarado, Barquisimeto, Venezuela, thejorgemylio@gmail.com*

Introducción

La robótica móvil es un campo de investigación que supone la integración de numerosas disciplinas entre las que se encuentran la electrónica, ingeniería mecánica, inteligencia artificial, estadística, entre otros. Tiene como objetivo construir una ruta de navegación sobre un entorno conocido o desconocido, a través de un sistema que integra la percepción del entorno mediante sensores, toma de decisiones y acciones.

La arquitectura AuRA propone un enfoque donde se fusiona los paradigmas deliberativo y reactivo, lo que le permite al robot responder a eventos inesperados y a la vez cumplir con los objetivos de la misión.

Pioneer P3DX

El Pionner P3DX (Figura 1) es un robot ligero ideal para ser usado en el interior de laboratorios o aulas de clases. El robot cuenta con una rueda y un motor ubicados a cada lado del mismo. Además, tiene una rueda en la parte posterior que le ayuda a mantener el equilibrio. Igualmente, el Pionner P3DX posee sensores de ultrasonido o sonares, empleados para detectar obstáculos en el ambiente. Es uno de los robots más popularmente usados en las áreas de la educación e investigación. Puedes ser fácilmente personalizado y mejorado, integrandolo con accesorios de diversa índole. Por otra parte, posee un conjunto completo de aplicaciones y librerías que sustentan el desarrollo de proyectos de investigación.

Especificaciones del Pioneer P3DX

Construcción

Dimensiones: 45,5 cm de largo por 38,1 cm de ancho (Figura 2).
Cuerpo: 1,6 mm de aluminio.
Neumáticos: de caucho relleno de goma.

Operación

Peso del Robot: 9 kg.

Figura 1: Robot Pioneer P3DX.

Carga operativa: 17 kg.

Potencia

Tiempo de ejecución: 8-10 horas con 3 baterías.
Tiempo de carga: 12 horas o 2,4 horas.
Baterías: Soporta hasta 3 a la vez.
Tensión: 12 V.
Capacidad: 7,2 Ah.

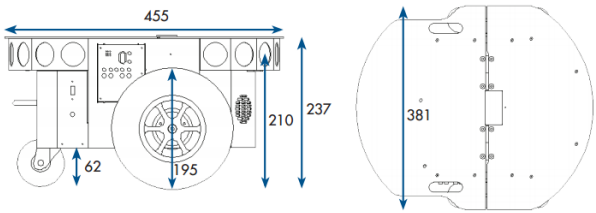

Figura 2: Dimensiones (milímetros) Robot Pioneer P3DX.

Arquitectura AuRA

La arquitectura AuRA [1] (Autonomous Robot Architecture) introducida por Ronald Arkin, plantea un enfoque híbrido de navegación para robots móviles. Está constituida principalmente por dos capas, una deliberativa basada en la inteligencia artificial tradicional y otra reactiva basada en teoría de esquemas. En la Figura 3 se puede

observar la representación del AuRA, la capa deliberativa la componen el *planificador de misión*, *razonador espacial* y el *secuenciador del plan*. En el nivel más alto se encuentra el *planificador de misión*, representa la interfaz con el humano y permite establecer los objetivos, las restricciones y los parámetros en los que debe operar el robot. El *razonador espacial* puede ser referido como navegador, este usa la información cartográfica almacenada para construir la ruta deseada que el robot debe seguir para completar la misión. El *secuenciador del plan* puede ser comparado con un piloto, convierte la ruta generada por el *razonador espacial* al conjunto de comportamientos (esquemas) que debe ejecutar la capa reactiva. En la capa reactiva, el *control de esquema* es el responsable de llevar a cabo y monitorear en tiempo real el conjunto de comportamientos que el secuenciador ordena ejecutar, cada comportamiento esta asociado a un esquema perceptual, en otras palabras, cada estimulo genera una respuesta.

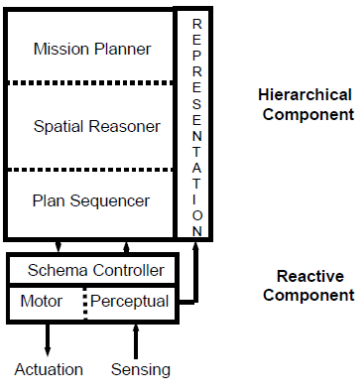

Figura 3: Representación de Arquitectura AuRA.

Fortalezas de la Arquitectura AuRA

Modularidad, flexibilidad, generalización e hibridación, son los características resaltantes de esta arquitectura.

Modularidad, es altamente modular debido a su diseño en capas, lo que permite intercambiar o mejorar los componentes sin afectar el

funcionamiento general.

Flexibilidad, esta arquitectura permite integrar diferentes modelos de inteligencia artificial para la construcción de rutas, aprendizaje y adaptación.

Generalización, puede ser empleado en distintos dominios como la manufactura, navegación, entre otros.

Hibridación, fusiona los paradigmas deliberativo y reactivo, lo que le permite al robot interactuar con eventos inesperados mientras intenta satisfacer sus metas.

Entorno de Desarrollo

Para llevar a cabo la presente investigación se desarrollo una aplicación cliente con las siguientes herramientas de software:

ARIA (Advanced Robot Interface for Applications) [2], es una librería desarrollada en el lenguaje de programación C++ para todas las plataformas MobileRobots/ActivMedia. Provee el conjunto de herramientas necesarias para controlar y monitorear de forma dinámica al robot y sus elementos. Ademas, esta librería incluye implementaciones llamadas *wrapper* en los lenguajes de programación Python y Java, siendo este último el lenguaje de desarrollo para este investigación.

MobileSim [3], es un software para la simulación de plataformas MobileRobots/ActivMedia. Permite usar la librería ARIA y sus wrappers. Es un ambiente ideal para realizar experimentación y pruebas de los algoritmos desarrollados antes de implementarlos en los robots reales.

Eclipse IDE [4], es un entorno de desarrollo integrado para múltiples lenguajes de programación en los que destacan Java y C++. Provee las herramientas necesarias para el desarrollo, depuración y compilación de aplicaciones.

Java [5], es un lenguaje de programación de propósito general, orientado a objetos y de alto nivel.

Configuración del Entorno

Antes de configurar el entorno de desarrollo es necesario descargar el código fuente del proyecto, este está disponible al público con licencia de software **MIT** en la siguiente dirección:
`https://bitbucket.org/sauljabin/ariajava-p3dx`.

Requisitos

1. Sistema operativo de 32bits.
2. Java 7 o mayor.
3. Eclipse 4 o mayor.

Configuración del Entorno Sobre Windows

1. Instalar `MobileSim-0.7.2-1.exe`, disponible en:
   `http://robots.mobilerobots.com/wiki/MobileSim`.
2. Instalar `ARIA-2.7.6.exe`, disponible en:
   `http://robots.mobilerobots.com/wiki/ARIA`.
3. Agregar a la variable de entorno `PATH` la ruta:
   `C:\Program Files\MobileRobots\ARIA\bin\`.
4. Dirigirse a la ruta `C:\Program Files\MobileRobots\ARIA\bin\`, cambiar el nombre del archivo `AriaVC10.dll` a `Aria.dll`.
5. Importar proyecto a eclipse.
6. Agregar al `Java Build Path` el wrapper Java de ARIA que se encuentra en la ruta:
   `C:\Program Files\MobileRobots\ARIA\java\Aria.jar`.
7. Verificar que la variable `PATH_LIBARIA` del archivo `CONFIG.props` sea igual a: `C\:\\Program Files\\MobileRobots\\ARIA\\bin\\`.

Configuración del Entorno Sobre Debian/Ubuntu

1. Descargar el archivo `MobileSim-0.7.3+gcc4.6.tgz`, disponible en: `http://robots.mobilerobots.com/wiki/MobileSim`.

2. Ejecutar los siguientes comandos por consola para instalar MobileSim:

```
$ tar xzvf MobileSim-0.7.3+gcc4.6.tgz
# mv -f MobileSim-0.7.3 /usr/local/MobileSim
# ln -s -f /usr/local/MobileSim/MobileSim /usr/bin/mobilesim
```

3. Descargar el archivo `ARIA-2.8.1+gcc4.6.tgz`, disponible en: `http://robots.mobilerobots.com/wiki/ARIA`.

4. Ejecutar los siguientes comandos por consola para instalar ARIA:

```
$ tar xzvf ARIA-2.8.1+gcc4.6.tgz
# mv -f Aria-2.8.1 /usr/local/Aria
# echo '/usr/local/Aria/lib' >/etc/ld.so.conf.d/aria.conf
# ldconfig
```

5. Importar proyecto a eclipse.

6. Agregar al `Java Build Path` el wrapper Java de ARIA que se encuentra en la ruta: `/usr/local/Aria/java/Aria.jar`

7. Verificar que la variable `PATH_LIBARIA` del archivo `CONFIG.props` sea igual a: `/usr/local/Aria/lib/`

Implementación de la Arquitectura

Capa Deliberativa

Planificador de misión

Con el fin de planificar la misión, se desarrolló una interfaz gráfica intuitiva que permite configurar los parámetros generales, cargar la información cartográfica y además presentar la ruta óptima que deberá recorrer el robot. Como se observa en la Figura 4 se puede ingresar la posición inicial y final deseada.

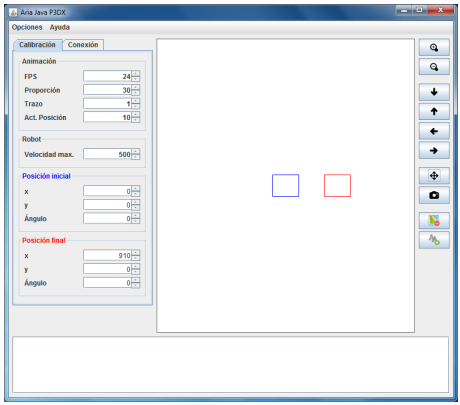

Figura 4: Planificador de misión.

Por otra parte es posible cargar mapas a través de archivos de texto plano con extensión *.map*, este formato de archivo es compatible con MobileSim y puede ser generado o editado por Mapper3 [6].

En el Listado 1 se puede apreciar un ejemplo de como está estructurado un archivo *.map*, cabe destacar que en este formato las longitudes son medidas en milímetros.

Listado 1: Ejemplo archivo mapa.

```
1  2D-Map
2  Cairn: RobotHome -5000 -5000 0 "" ICON "Home"
3  Cairn: Goal 5000 5000 0 "" ICON "Goal"
4  LineMinPos: -6000 -6000
5  LineMaxPos: 6000 6000
6  NumLines: 4
7  LINES
8  -6000 6000 6000 6000
9  -6000 -6000 6000 -6000
10 -6000 6000 -6000 -6000
11 6000 6000 6000 -6000
```

A continuación se describen las líneas que conforman el mapa:
Línea 1: `2D-Map`, indica el tipo de mapa.
Línea 2: `Cairn: RobotHome -5000 -5000 0 "" ICON "Home"`, representa la posición inicial del robot.

Línea 3: `Cairn: Goal 5000 5000 0 "" ICON "Goal"`, representa la posición objetivo.
Línea 4: `LineMinPos: -6000 -6000`, indica el punto de inicio del mapa.
Línea 5: `LineMaxPos: 6000 6000`, indica el punto fin del mapa.
Línea 6: `NumLines: 4`, indica la cantidad de líneas (obstáculos) que contiene el mapa.
Línea 7: `LINES`, indica el inicio de la sección de obstáculos.
A partir de la línea 8: sección de obstáculos.

La Figura 5 muestra el resultado del archivo *.map* descrito en el Listado 1.

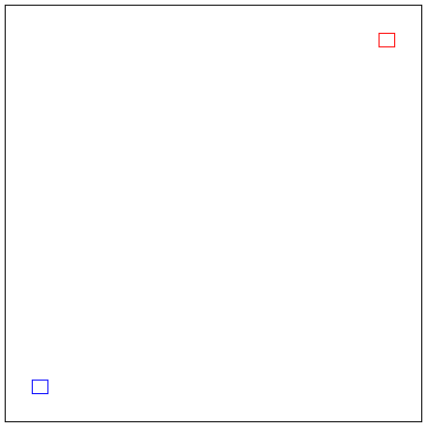

Figura 5: Resultado de ejemplo archivo mapa Listado 1.

Razonador Espacial

Secuenciador del Plan

Capa Reactiva

Herramienta de Software Desarrollada

Experimentación

Conclusión

AL FINALIZAR

Referencias

[1] Arkin, R. y Balch, T. (1997). AuRA: Principles and practice in review. Journal of Experimental & Theoretical Artificial Intelligence. Taylor & Francis. Vol. 9. No. 2-3. Pp. 175-189.

[2] MobileRobots. (2014). ARIA. Disponible en: `http://robots.mobilerobots.com/wiki/ARIA`.

[3] MobileRobots. (2014). MobileSim. Disponible en: `http://robots.mobilerobots.com/wiki/MobileSim`.

[4] Eclipse. (2014). Eclipse IDE. Disponible en: `https://www.eclipse.org/ide/`.

[5] Oracle. (2014). Java. Disponible en: `http://www.oracle.com/technetwork/java/index.html`.

[6] MobileRobots. (2014). Mapper3. Disponible en: `http://robots.mobilerobots.com/wiki/Mapper3`.