DISEÑO E IMPLEMENTACIÓN DE LA TELEOPERACIÓN DE UN ROBOT MÓVIL

JAVIER ANTONIO CONSTAIN ARROYAVE

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2012

DISEÑO E IMPLEMENTACIÓN DE LA TELEOPERACIÓN DE UN ROBOT MÓVIL

JAVIER ANTONIO CONSTAIN ARROYAVE

Proyecto de grado para optar al título de
Ingeniero Mecatrónico

Director
Dr. JESÚS ALFONSO LÓPEZ
Ingeniero Electricista
Magister en Automática
Doctor en Ingeniería

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2012

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Mecatrónico

ING JUAN CARLOS MENA.

Jurado

ING. JUAN CARLOS PERAFÁN

Jurado

Santiago de Cali, 12 de Abril de 2012

A mi familia, mis amigos, a todos los creyeron en mí y en especial a los que no. Porque todos ellos son los que me motivan a salir adelante.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

En este proyecto se describe el proceso de diseño e implementación de un modelo prototipo de teleoperación de un robot móvil capaz de brindar una herramienta que permita soportar nuevas tendencias de migración a educación en entornos virtuales y remotos.

La universidad se encuentra actualmente en un proceso de generar nuevas opciones y alternativas de enseñanza, entre ellos se encuentran los cursos virtuales y a distancia, que le permitirán al estudiante poder aprovechar y aplicar sus conocimientos con mayor dinamismo. La área de la Ingeniería está desarrollando aplicaciones en las que se muestran opciones para el acceso a los laboratorios de manera virtual y remota, empleando distintas herramientas y metodologías creadas y adecuadas en este proceso.

Para el desarrollo de este trabajo se siguió la metodología del diseño concurrente, buscando lograr el mayor desempeño, adaptándose lo mejor posible a los requerimientos de la universidad y haciendo uso de cada una de las herramientas que la misma institución dispone. De este modo se obtuvo un diseño sólido basado en técnicas web cliente - servidor como html, javascript y php, para la interacción herramienta - usuario. A través del lenguaje C se pudo establecer el lazo servidor – robot con el cual se logró establecer exitosamente el vínculo o enlace para correr algoritmos propios de la robótica móvil, como la teleoperación o el seguimiento de trayectorias en uno de los robots e-puck disponibles.

La interfaz diseñada permite al usuario una interacción dinámica y visual, que logra su total atención y admiración, buscando la mayor participación posible del alumnado y permitiendo la mayor interacción posible, mientras que a su vez retroalimenta la mayor cantidad de información del estado del robot. Y gracias al diseño modular del sistema permite la adaptación de distintas interfaces o la interacción con nuevos robots o plantas del laboratorio para ampliar su funcionalidad en cualquiera de las partes que le componen.

Palabras clave: Actuadores, cinemática, diseño, móvil, robótica, robot, sensores, teleoperación.

INTRODUCCIÓN

El presente proyecto presenta el proceso de desarrollo de una plataforma para la teleoperación de un robot móvil que facilita el desarrollo de prácticas y estudios en el área de la robótica móvil para los estudiantes, docentes e investigadores del programa de ingeniería mecatrónica de la universidad autónoma de occidente.

Partiendo de los avances que han obtenido las tecnologías de la información, ha comenzado una revolución silenciosa en el modo que se imparte la educación, por ello se deben desarrollar distintas herramientas que permitan el desarrollo de las actividades académicas incluso, por fuera de la planta física de la academia.

Durante el proceso de ejecución se tienen en cuenta los contenidos de los cursos en robótica móvil, para así poder establecer la capacidad de funcionamiento que puede llegar a tener el entorno remoto. Se realiza el estudio de las herramientas y robots móviles existentes en la universidad para que la conectividad entre las diferentes partes del sistema pueda ser satisfactoria.

Se diseña el entorno de trabajo remoto y las rutinas que permiten la interacción de los usuarios con el sistema, teniendo el cuenta la posibilidad de conexión desde cualquier parte. Siendo posible así, realizar las pruebas necesarias que comprueban su correcto funcionamiento.

La característica principal que debe permitir la plataforma es la conectividad de muchos usuarios dentro un área local o a través de la Internet, además de un diseño que hiciera posible, en un futuro, adaptar nuevos equipos de los laboratorios. Requerimientos que conllevan a un serio estudio de posibilidades y alcance tecnológico, siendo también necesario la aplicación de una herramienta metodológica avanzada de diseño, como lo es el diseño concurrente.

La importancia de este estudio recae en la posibilidad que brinda a los estudiantes de acceder a los recursos físicos de la universidad desde cualquier parte, brindando más comodidades para el desarrollo de sus proyectos de clase. También ayuda a las instituciones a ofrecer sus programas a distancia ofreciendo un factor diferenciador a la competencia, siendo pocas instituciones con el acceso remoto a sus plantas y equipos.

1.OBJETIVOS

1.1 OBJETIVO GENERAL

Diseñar e implementar la teleoperación de robot móvil como apoyo a los procesos educativos

1.2 OBJETIVOS ESPECÍFICOS

- Estudiar los problemas que se desarrollan en los cursos de robótica móvil
- Seleccionar los robots móviles para implementar el sistema remoto
- Definir el protocolo de transmisión de datos para la comunicación remota
- Definir las tareas a realizar de manera remota por los robots móviles e implementar los algoritmos que las realicen en los mismos.
- Diseñar el entorno de trabajo para la teleoperación.
- Realizar pruebas de funcionamiento y analizar resultados

2.MARCO TEÓRICO

2.1 ROBÓTICA MÓVIL

La Asociación Robótica Industrial Japonesa (JIRA, Siglas en inglés para: *Japanese Industrial Robot Association*) ha creado una lista que clasifica y busca abarcar a todos los tipos de robots existentes.

- Manipuladores operados manualmente: Máquinas que deben ser operadas directamente por una persona.

- Manipuladores secuenciales: Dispositivos que realizan una serie de tareas en una misma secuencia cada vez que son activados.

- Manipuladores programables: Un brazo robótico industrial.

- Robots controlados numéricamente (*Playback* robot): Robots que son instruidos para desarrollar tareas a través de la información recibida en secuencias y posiciones en forma de datos numéricos.

- Robots *sensate*: Robots que incorporan cualquier tipo de sensores, para realizar retroalimentación a través de ellos.

- Robots adaptativos: Robots que ajustan su comportamiento a los cambios en su entorno.

- Robots inteligentes: Se consideran que son robots de alta tecnología que poseen inteligencia artificial.

- Sistemas mecatrónicos inteligentes: Computadores o sistemas embebidos que controlan una flota de robots o dispositivos robóticos.[1]

Así, un robot móvil puede definirse como un dispositivo electromecánico, no fijo en una posición, que se puede desplazar en un entorno determinado. Se busca que

1 BRANWYN, Gareth, Absolute Beguinner's Guide to Building Robots. Estados Unidos, Que Publishing, 2003. 384 p.

este entorno pueda incluso llegar a ser desconocido y posiblemente cambiante, además de realizar sus tareas sin supervisión.

2.1.1 Aplicaciones de los robots móviles. Los robots móviles generalmente se desenvuelven en entornos que son típicamente interiores, como edificios, oficinas o laboratorios, y exteriores como el campo abierto.[2]

Entre las diversas aplicaciones de los robots móviles están los limpia pisos, patrulleros detectores de intrusos, asistentes de hospital, guías en museos o centros comerciales, exploradores espaciales, rastreadores en zonas desconocidas, de alta peligrosidad o donde el acceso de personas es restringido, también, son usados para desactivar o detectar bombas, la movilización de troncos recién talados, entre otros. [3]

Además, existen una gran cantidad de robots diseñados para tareas de investigación, batallas de robots y en general todo tipo de aplicación a la que las personas les puedan encontrar uso, ya sea personal, investigación o comercial.

2.1.2 Anatomía de los robots móviles. La robótica se puede considerar como un campo interdisciplinar donde se fusionan diversas ramas del saber como la electrónica, la mecánica y la informática. Es por esto que se pueden encontrar diversos tipos de componentes y sistemas dentro de un robot.

2.1.2.1 Sensores. Los sensores son los elementos que permiten al robot obtener la información de su entorno y saber lo que sucede en este.

- Sensores de contacto. Se encuentran entre los sensores más básicos, operan por simple contacto con una estructura sólida.

♦ **Bumpers.** Suelen ser dispositivos pasivos, generalmente compuestos por un pulsador que envía un pulso al hacer contacto.

♦ **Whisker o bigotes.** Consiste básicamente en que cuando tocan un objeto, causan una vibración que puede ser detectada, enviando una señal.[4]

2 CAÑAS, José María. Jerarquía dinámica de esquemas para la generación de comportamiento autónomo. Doctor en telecomunicaciones. Madrid. Universidad Politécnica de Madrid. 2003.

3 SIEGWART, Roland y NOURBAKSHS, Illah. Introduction to autonomous mobile robots. Londres, MIT Press, 2004. 321 p.

4 GIBILISCO, Stan. Concise Encyclopedia of Robotics. Estados unidos, McGraw Hill, 2003. 365 p.

- Sensores de luz. Son aquellos que detectan la presencia de rayos de luz.

♦ **Celdas fotoeléctricas.** Son resistencias que varían su valor de resistencia dependiendo de la cantidad de luz que reciben. Su funcionamiento puede ser comparable con el de un potenciómetro, con la diferencia que funciona con la intensidad de luz.

♦ **Detectores de proximidad infrarrojos.** Está configurado por una pareja de diodos infrarrojos emisor y receptor. No están diseñados para decir la distancia exacta de un objeto, sin embargo, permiten generar una señal cuando detectan la presencia de un objeto.

♦ **Sensores de rango infrarrojos.** Permiten calcular el rango a un objeto por medio de triangulación, llegando a ser prácticamente insensibles al color o la textura del objeto al que apunten.

♦ **Sensores piroeléctricos.** Son sensores cuya salida varía cuando ocurren pequeños cambios en la temperatura del sensor. Generalmente el elemento del sensor es un cristal de litio-tantalio. Son sensores usados en su mayoría para las alarmas detectoras de intrusos.

♦ **Sensores ultravioleta.** Son sensores que son sensibles a las radiaciones en el rango de los 185 – 260 nanómetros, pero insensibles al rango de la luz visible. Una fuente común de este rango de radiación es el fuego.

- Sensores de sonido. Le permiten percibir los sonidos del ambiente al robot.

♦ **Micrófonos.** Le permiten al robot realizar ciertos movimientos dependiendo del patrón del sonido que les llegue. Tienen la ventaja que permiten establecer fácilmente una interfaz con un microprocesador.

♦ **Sonares.** Permiten conocer la presencia de un objeto, sin embargo es posible calcular la distancia del mismo midiendo el tiempo que duró la onda en ir y volver.[5]

- Sistemas de visión. Son básicamente cámaras de video que se pueden conectar a los robots, brindándoles la posibilidad de tener visión.

5 JONES, Joseph L; FLYNN, Anita y SEIGER, Bruce. Mobile Robots. Inspiration to implementation. 2 ed. Estados Unidos, A K Peter, 1998. 486 p.

- Sensores internos. Permiten al robot conocer su estado interno. Entre ellos encontramos: medidores de nivel de batería, *encoders* incrementales, *encoders* absolutos, giroscopios, brújulas y temperatura.

2.1.2.2 Motores. Son los elementos que permiten el movimiento del robot. En el mercado existen dos tipos de motores, AC y DC. Sin embargo en robótica móvil, por requerimientos de energía y tamaño se usan más los motores de DC.

- Motor DC. Son la esencia de la mayoría de los robots móviles. Sin embargo, su velocidad de giro es muy rápida para ser aprovechada por el robot móvil y su torque generalmente no es suficiente para mover el robot, por ello suele ser necesario adaptarle ciertos mecanismos mecánicos como engranajes, piñones y correas para aprovechar al máximo sus capacidades.

- Servo motores. Son motores DC que ya vienen adaptados con una caja reductora y el controlador del motor. Permiten realizar el control del motor mucho más fácil. Sin embargo la rotación de estos suele venir limitada (entre 130 – 180 grados),por lo que es necesario modificar el hardware para lograr que funcione correctamente en los robots.

- Motores paso a paso. Son motores cuyo alambrado está organizado de manera diferente a los motores DC. De esta manera el rotor del motor puede moverse de manera discreta de un cableado de cobre a otro en el estator. Estos motores pueden brindar controles de velocidad más precisos que los motores DC o los servomotores, además que no es necesario usar cajas reductoras.

2.1.2.3 Energía. Para poder desplazarse en el entorno o realizar las tareas, los robots necesitan de una fuente de poder. Es necesario también, que esta fuente de poder entregue un voltaje constante para que los circuitos trabajen adecuadamente. Entre los diversos sistemas de alimentación que se pueden encontrar se encuentran: baterías, celdas solares, motores a combustión, etc.

2.1.2.4Procesamiento. Es el corazón del robot, donde se mandan las órdenes a los actuadores y reciben los datos o información de los sensores. Son básicamente los procesadores, en ellos se almacenan los programas con los que va a funcionar el robot y así poder cumplir con sus tareas.

Estos procesadores, pueden ir desde un simple chip de microprocesador hasta grandes computadores personales.

2.1.3 Locomoción. La Locomoción hace referencia al movimiento del robot, aquello que le permite desplazarse en el entorno que se encuentra. Existen diversas formas de desplazarse: caminar, saltar, correr, deslizarse, patinar, nadar, volar y rodar.

2.1.3.1 Sistemas de piernas. Se caracterizan por ejercer una serie de puntos de contacto entre el robot y la tierra.

Entre las ventajas de estos sistemas se encuentran la adaptabilidad y maniobrabilidad en terrenos irregulares, pueden atravesar huecos o abismos sin problemas mientras que la longitud de las piernas lo permita y ofrecen el potencial de manipular objetos en el ambiente con gran habilidad. Entre sus desventajas están la complejidad de los mecanismos y la potencia, el gran número de grados de libertad, además que deben ser capaces de sostener (y en ocasiones elevar y bajar) el robot, y la maniobrabilidad depende que las piernas tengan el número suficiente de grados de libertad para impartir fuerzas en el terreno en diferentes direcciones.[6]

2.1.3.2 Sistemas de ruedas. En los robots móviles el uso de ruedas es el más popular por varios motivos. Simplifican la mecánica y la construcción del robot, ofrecen buena relación peso – mecanismo, esto debido a que, a diferencia de sistemas de piernas y orugas, requiere de mecanismos menos complejos y pesados; finalmente, existen muchos juguetes, que vienen con ruedas, que pueden ser modificados para uso en robótica. Sin embargo en ambientes donde el terreno es muy irregular, las ruedas se desempeñan mal.

Existen cuatro diseños de ruedas básicos para los robots móviles (Figura 1), estos varían en la cinemática que ofrecen. Entre ellos se encuentran:

- Rueda estándar: Dos grados de libertad, rotación alrededor del eje de la rueda y el punto de contacto.

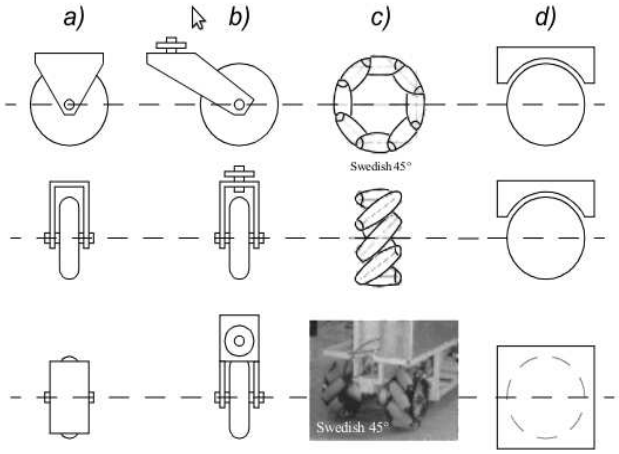
- Rueda castor: Dos grados de libertad, rotación alrededor de una articulación con gobierno con offset.

- Rueda sueca: Tres grados de libertad, rotación alrededor del eje de la rueda, alrededor de las sub ruedas y alrededor de los puntos de contacto.

6 SIEGWART, Roland y NOURBAKSHS, Illah. Introduction to autonomous mobile robots. Londres, MIT Press, 2004. 321 p.

• Bola o rueda esférica: Difícil de realizar descripción técnica, aunque es posible describirla como una rueda con tres grados de libertad.

Figura 1. Los cuatro tipos de ruedas básicas. a)Rueda estándar b) Rueda Castor c) Rueda suiza d) Rueda esférica.



Fuente: Introduction to Autonomous Robots. p. 31.

2.1.4 Desarrollos más comunes. Existen una serie de proyectos que pueden ser desarrollados como herramientas de introducción al mundo de la robótica, que pueden llegar a ser herramientas muy útiles para desarrollos más avanzados.

2.1.4.1 Robot deambulador. El objetivo es avanzar y evitando quedar estancado. Debe mantenerse en movimiento, posiblemente, explorando el territorio. El robot es liberado en un ambiente completamente desconocido para este, comienza a andar y por medio de sensores detecta la presencia de los obstáculos y los evita para no quedarse bloqueado.

2.1.4.2 Seguidor de luz. El robot debe perseguir o evitar una fuente de luz, encontrándose en un ambiente obscuro. Generalmente la fuente de luz puede ser una linterna.

2.1.4.3 Seguidores de línea. Consiste en una línea blanca o negra que se pone, en el mayor de los casos, sobre el piso con un color que contrasta la línea, formando una ruta la cual el robot debe seguir.

Este tipo de técnica suele ser empleada en algunas industrias, donde el robot, siguiendo la ruta, inspecciona el entorno en busca de anomalías.

2.1.4.4 Teleoperación. Es el término técnico empleado al control de robots autónomos por control remoto.

En un sistema robótico teleoperado, una persona puede controlar la velocidad, dirección y otros movimientos del robot a cierta distancia. Ciertas señales van hacia el robot indicándole las órdenes de control, mientras que ciertas señales vienen desde este indicando que el robot sigue las instrucciones y diversos estados en que se encuentre.

2.1.5 Los robots autónomos. Entre los principales desafíos de la robótica móvil es la generación del comportamiento autónomo. El robot, por tanto, conociendo su objetivo o lugar de destino, debe tomar la mejor decisión y alcanzarlo.

Este concepto, ubicado entre los principales temas de investigación en robótica móvil, es conocido como navegación. Sin embargo la navegación no es posible sin tener en cuenta cuatro bases principales: percepción, localización, planificación y mapeo.

2.1.5.1 Percepción. Consiste en la información que obtiene el robot acerca de su entorno, para eso emplea y manipula las señales de los sensores que dispone.

2.1.5.2 Localización. Hace referencia a la posición del robot en el plano. Este debe conocer la posición de todos los puntos que forman su estructura en el plano, de modo que ninguna de sus partes del robot tenga contacto con el entorno.

El problema de la localización sería fácilmente resuelto si la posición del robot la brindara un sistema GPS. Sin embargo, la precisión de estos dispositivos tiene un error de unos cuantos metros, además, casi siempre sería indispensable que el robot se encontrara en un campo abierto donde no se pierda la señal del dispositivo GPS.

Además si el robot interactúa con personas, el robot debe poder identificarlos y definir su posición relativa con ellos.

2.1.5.3 Mapeo. Hace referencia a los datos del plano o mapa sobre el cual se desplaza el robot, almacenado previamente en su memoria.

Esto permite al robot localizarse y realizar mejor la planificación de la trayectoria, evitando con mayor facilidad obstáculos y determinando la configuración del espacio.

2.1.5.4 Planificación. Para llegar a un punto determinado, partiendo de la posición actual del robot. Este debe escoger la ruta más eficiente para llegar al punto, evitando obstáculos o quedarse estancado, y respetando las restricciones físicas del mismo robot o el entorno.

Así, con un plano y el conocimiento del punto de llegada, el planteamiento de la trayectoria involucra identificar la ruta que hará que el robot llegue al punto de llegada cuando la orden sea ejecutada.[7]

2.1.5.5 SLAM. Hace referencia a la pregunta ¿Qué pasa si el robot está en un ambiente desconocido donde no conoce su localización y no dispone del mapa?

En la navegación tradicional el robot dispone de un mapa a partir del cual puede localizarse, o conoce su localización, a partir de la cual puede ir generando su mapa. SLAM representa entonces un paradigma como el del huevo o la gallina ¿Cuál fue primero? [8]

2.2 CINEMÁTICA DE ROBOTS MÓVILES

La cinemática en los robots, les permite conocer su posición en todo instante de tiempo. Principalmente existen dos tipos de cinemáticas, directa e inversa.

Debido a que los robots móviles de la universidad presentan en su mayoría configuraciones diferenciales, se mostrará el modelo basado en ellos. Estos robots tienen modelos no holónimos, lo que representa que aunque se puedan mover hace adelante y hacia atrás no puede desplazarse lateralmente sin que exista un derrape.

7 CHOSET, Howie; LYNCH, Kevin; SETH, Hutchinson; KANTOR, George; BURGARD, Wolfram; KAVRAKI Lydia y THRUN, Sebastian. Principles of robot motion. Theory, algorithms and implementation. Londres, MIT Press, 2005. 603 p.

8 Radish: The Robotics Data Set Repository [en línea]. HOWARD, Andrew. ROY, Nick [Consultado Ago 2010]. Disponible en Internet: http://radish.sourceforge.net

Por todo lo anterior y para facilitar el modelado de las cinemáticas se manejan ciertas condiciones ideales:

- El robot se mueve en una superficie plana.

- Los ejes guías son perpendiculares al suelo.

- Durante el movimiento la rueda mantiene un contacto permanente con el suelo.

- El robot no posee partes flexibles.

- El cambio de posición se produce alrededor de un arco de circunferencia.

- El comportamiento del robot es análogo al de un sólido rígido.

2.2.1 El CCI. En la robótica móvil se emplea el término CCI, o centro de curvatura instantánea, para hacer referencia al círculo imaginario de radio "R" que se forma cuando el robot móvil se desplaza con las velocidades diferentes en ambas ruedas.

2.2.2 Cinemática directa. Esta permite conocer la posición y orientación del robot en todo momento a partir de la velocidad del robot y sus demás parámetros geométricos.

De este modo se puede calcular la pose del robot en cada instante t a partir de:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\ \Delta t) & -sen(\omega\,\Delta t) & 0 \\ sen(\omega\ \Delta t) & \cos(\omega\ \Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - CCI_x \\ y - CCI_y \\ \theta \end{bmatrix} + \begin{bmatrix} CCI_x \\ CCI_y \\ \omega\ \delta t \end{bmatrix} \quad (1)$$

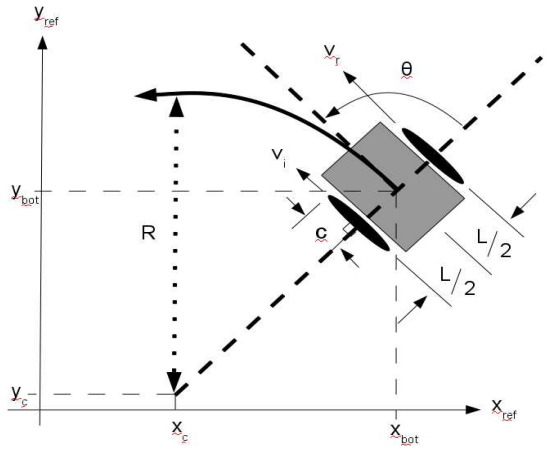
Integrando la ecuación anterior se obtiene:

$$x(t) = \frac{1}{2}\int_0^t (v_r(t)+v_l(t))\cos(\omega(t))\delta t \quad (2)$$

$$y(t) = \frac{1}{2}\int_0^t (v_r(t)+v_l(t))\,sen(\omega(t))\delta t \quad (3)$$

$$\theta(t) = \frac{1}{d}\int_0^t (v_r(t)+v_l(t))\delta t \quad (4)$$

Figura 2. Formación del CCI en la rotación de un robot móvil



2.2.3 Cinemática inversa. Por medio de la cinemática inversa es posible, a partir del tiempo de desplazamiento dado y sus parámetros geométricos, determinar las velocidades de los motores que le permitirán llegar a la posición final en ese tiempo dado.

Así, si el modelo directo puede definirse cómo.

$$\dot{p} = J(p)\dot{q} \quad (5)$$

Donde p hace referencia a las coordenadas generalizadas, q a las variables de actuación y J(P) es la matriz jacobiana del movimiento. Entonces el modelo inverso se puede definir como:

$$\dot{q} = J^{*}(p)\ \dot{p} \tag{6}$$

Donde:

$$J^{*}(p)=(J^{T}(p)\ \cdot J(p))^{-1}\ J^{T}(p) \tag{7}$$

2.3 PROTOCOLOS DE COMUNICACIÓN Y HERRAMIENTAS DE PROGRAMACIÓN

2.3.1 Bluetooth. *Bluetooth* es creado como un medio de baja potencia por el cual los dispositivos puedan comunicarse en cortas distancias. Su función es la de englobar, entre sus especificaciones, un conjunto de sub normas, como las frecuencias de radio, y el modo de modular y demodular señales, que permiten llevar a un modo inalámbrico otras funciones que de lo contrario se tendrían que realizar de modo cableado.

Para establecer una conexión *bluetooth* es necesario definir qué equipo la va a realizar de modo entrante o saliente. Así, el dispositivo con la conexión saliente debe localizar el dispositivo objetivo y el protocolo de transporte; y aquellos con conexiones salientes deben definir el protocolo de transporte y estar pendientes de conexiones entrantes.

Así como los equipos con conectividad a internet (Computadores, celulares) que disponen de una dirección MAC, los dispositivos con conectividad *bluetooth* poseen un código único de identificación de 48 bits conocido como la “dirección del dispositivo”. De esta manera, los equipos con conexiones salientes pueden encontrar, identificar y comunicarse con otros que se encuentren con conexiones entrantes, sin importar la capa del proceso aplicativo *bluetooth*.

Sin embargo, para la mayoría de los usuarios, es muy incómodo y difícil recordar la dirección de los dispositivos *bluetooth* con lo que se conecta usualmente, por lo cual es posible darles un “nombre” a esos equipos, que posteriormente, el

programa se encargará de convertirlo a la dirección numérica.

Existen diversos protocolos de transporte dentro del estándar *bluetooth*, esto debido a la variedad de aplicaciones posibles y los diversos requisitos de cada una. Cada una distinguida, principalmente por sus garantías de transporte y recepción, y las semánticas.

Después de realizar un análisis de protocolos existentes, se realiza una investigación de los sistemas operativos mejor soportados por los entornos de desarrollo de aplicaciones *bluetooth* existentes y para qué lenguajes de programación están disponibles. Una descripción de estos se puede ver en la tabla 1. [9]

Cuadro 1. Diferentes tipos de API's *bluetooth* y sus protocolos soportados.

API (SO)	**RFCOMM**	**L2CAP**	**SCO**
PyBlueZ (GNU/Linux)	Si	Si	Si
PyBlueZ (GNU/Linux)	Si	No	No
BlueZ (GNU/Linux)	Si	Si	Si
Microsoft (WinXp)	Si	No	No
Widcomm (WinXp, Win7)	Si	Si	Si
Java- JSR82 (multi plataforma)	Si	Si	No
Series 60 Python (Symbian OS)	Si	No	No
Series 60 C++ (Symbian OS)	Si	Si	No
Mac OS X	Si	Si	No

Fuente: HUANG, Albert. RUDOLPH, Larry. Bluetooth Essentials for programmes. p. 38.

9 HUANG, Albert y RUDOLPH, Larry. Bluetooth Essentials for Programmers. Nueva York, Cambridge University Press, 2007. 198 p.

2.3.2 Lenguajes de programación.

2.3.2.1 Java. Es un lenguaje de programación orientado a objetos actualmente desarrollado por *Oracle*, muy aplicado en el desarrollo de aplicaciones de escritorio y web. Sigue la filosofía de "Escríbelo una vez, córrelo en todas partes" gracias a la implementación de una Máquina virtual que corre los programas compilados estando en un nivel de adaptación más bajo. Es además uno de los lenguajes de programación más usados en la actualidad.[10]

Entre las características que destacan a Java y lo convierten en uno de los lenguajes más utilizados en la actualidad se encuentran:

- Multiplataforma. Gracias a la filosofía ya mencionada y la implementación de la máquina virtual que debe estar instalada en cada máquina, esto es posible.

- Orientado a objetos. Cumple plenamente con las características de herencia, encapsulamiento, polimorfismo y enlaces dinámicos.

- Robustez. Gracias a un mecanismo de ubicación de memoria y de recolección automática de basuras. Buen manejo de excepciones y revisiones por parte de compilador para evitar errores de programa y por parte del interpretador para los errores de ejecución aseguran al sistema para evitar colapsos en este.

- Distribuido. Implementa protocolos HTTP y FTP, así los programadores pueden invocar funciones que accedan a los archivos desde cualquier sistema remoto por internet.

- Dinámico. Permite adquirir archivos dinámicamente desde un disco local o en cualquier computador conectado por internet.

- Seguro. No usa directamente los apuntadores de memoria, todos los programas corren en un espacio conocido como la "caja de arena". Además implementa un sistema de encriptación de llaves que permite a la aplicación a transmitir por internet de un modo seguro.

- Desempeño. Aunque es un lenguaje interpretado utiliza procesos llamados hilos, los cuales, con una técnica de compilación adaptativa y "justo a tiempo" en

10 HORTON, Ivor, Beginning Java 2. JDK. 5 edición. Indianapolis, Wiley Publishing, 2005. 1470 p.

la máquina virtual, ayudan a incrementar el desempeño.

- Interpretado. Esto le permite aumentar su capacidad multi plataforma, además que incrementar su calidad y facilita la depuración de errores.

2.3.2.2 C. Es un lenguaje de nivel medio el cual combina el control de estructuras de un lenguaje de alto nivel con la habilidad de manipular bits, bytes y apuntadores o direcciones; dando al programador el control total sobre su computador.

Es compatible con C++, también es un lenguaje estandarizado por la ANSI e ISO, su última versión estándar es la C99, sin embargo esta versión no es completamente compatible con los estándares de C++.[11]

Posee la ventaja de ser usado no solo para programar programas para computadores, también se usa para firmware de micro controladores, sistemas operativos y programación gráfica. Sigue también una filosofía análoga al "escríbelo una vez, compílalo donde quieras", lo que permite que un programa con ligeras modificaciones (generalmente en los entornos gráficos), pueda migrar a cualquier dispositivo.

En su esencia dispone de pocas funcionalidades, aún así posee la característica de poder incorporar varias librerías que han sido diseñadas para suplir estas carencias, por lo que es posible encontrar una librería para cada tarea que se necesite realizar, o desarrollar la librería que supla la necesidad, de ser necesario.

Resumiendo otras características:

- Reubicación de objetos.

- Creación de muchos archivos hex para chips con memoria externa.

- Manejo más flexible de datos constantes.

- Capacidad de crear direcciones de memoria específicas en cualquier tipo de dispositivo de memoria.

11 SCHILDT, Herbert. C/C++ Programmer's Reference. 3 ed. California, McGraw Hill, 2003. 358 p.

- Sobrecarga de funciones.

- Parámetros por defecto y de cantidad variable.

- Capacidad de generar distintas opciones de compilación según sea necesario en un mismo programa.[12]

2.3.3 Lenguajes de la Internet.

2.3.3.1 HTML. *Hypertext Markup Languaje* o Lenguaje de Hiper texto Descriptivo sirve para describir la estructura, por medio de marcadores, de las páginas web lo que permite:

- Publicar documentos, tablas, fotos, encabezados, entre otros, en línea.

- Recuperar información entre páginas por medio de enlaces de hiper texto, con el clic de un botón

- Diseñar formularios para conducir transacciones con servicios remotos, buscar información, ordenar productos, entre otros.

- Incluir presentaciones, videos, música y otras aplicaciones directamente en los documentos.[13]

Con el tiempo se han desarrollado diferentes versiones de html, las cuales han marcado una diferencia en la forma en que el usuario experimenta la web:

- HTML 1.0 nunca fue considerado como un estándar aprobado y era más lo que no brindaba, que lo que se podía hacer. No se podía especificar ningún tipo de fondo para las páginas ni para las imágenes, no habían tablas ni frames, no se podían cambiar las fuentes, las imágenes solo podían ser GIF y no había forma de llenar formularios. Sin embargo, aunque todas las páginas parecían tener un mismo formato, fue este el principio de la navegación web.

12 What are the features of 'C' Language [en línea]. Nueva York [Consultado en Oct 2010]. Disponible en internet [http://wiki.answers.com/Q/What_are_the_features_fo_'C'_language]

13 HTML & CSS. [en línea]. World Wide Web Consortium [Consultado Oct 2010] Disponible en línea: http://www.w3.org/standards/webdesign/htmlcss

- HTML 2.0 implicó un gran avance ya que permitió que las limitantes anteriormente mencionadas fuesen posibles.

- HTML 3.2 incluyó el soporte para el primer nivel de CSS, aunque no todos los navegadores web brindaban ese soporte. De esta manera fue posible para los diseñadores web, dar apariencias más personalizadas a sus páginas.

- HTML 4.0 se lanzó como un proceso de redefinir la función de HTML, abolió la tradición de dar un formato a un elemento dentro del mismo archivo html, otorgando esta función a CSS. Esto principalmente porque el mantenimiento de las páginas web se había convertido en un trabajo muy complejo, por no tener en cuenta esta separación, y así permitir un mayor desarrollo de la web.

- HTML 5 es el nuevo estándar en desarrollo el cual espera brindar características como: elementos de lienzo para dibujos, reproducción de audio y video, mejor soporte para almacenamiento fuera de línea, nuevos elementos para contenidos específicos (como artículos, piés de notas, encabezados, secciones), y nuevos controles de formularios (calendarios, fechas, tiempo, correo, url, búsquedas). [14]

2.3.3.2 CSS. *Cascading Style Sheets* ó Capas de Estilo en Cascada, es un lenguaje que describe el modo en que las páginas web son mostradas. De esta forma se puede adaptar la presentación de las páginas a diferentes tipos de dispositivos.[15]

Con esta herramienta es posible dar formato y estilo visual a los textos y demás elementos (colores, fondos, fuentes, etc.) que conforman la página como si fueran vistos desde un programa tipo Microsoft Word. Gracias a la definición de estructuras, al momento de cambiar el formato de una serie de instancias con características similares no es necesario cambiar cada una de ellas como con HTML, sino que se definen previamente como parte de la estructura, siendo necesario únicamente cambiar el contenido de la estructura para lograr el mismo resultado.[16]

2.3.3.3 JavaScript. Es el lenguaje de programación orientado a objetos e

14 HTML 5 Introduction [en línea]. Refsnes Data [Consultado Oct 2010]. Disponible en línea: http://w3schools.com/html5/html5_intro.asp

15 HTML & CSS. [en línea]. World Wide Web Consortium [Consultado Oct 2010] Disponible en línea: http://www.w3.org/standards/webdesign/htmlcss

16 PFAFFENBERGER, Brian, SCHAFER, Steven; et al. HTML, XHTML and CSS Bible. 3 ed. Indianápolis, Wiley Publishing, 2004. 790 p.

interpretado de internet, posee la característica que puede ser embebido y ejecutado en las páginas web para lograr que estas sean dinámicas, de modo que el código correrá una vez la página sea descargada por el usuario o por un evento causado por la misma.

Es también posible hacer que las páginas interactúen con el dispositivo sobre el que se esté ejecutando, logrando que la página muestre y actúe dependiendo de la información de quien la visita, como el nombre, ubicación geográfica o, sistema operativo. Así es posible decir que se vuelve posible que se ejecuta un programa directamente en la página.[17]

Entre las tareas que se pueden realizar con JavaScript se encuentran:

- Lograr que la página web responda directamente a las interacciones del usuario con eventos de formularios (campos de entrada, áreas de texto, botones, listas de selección) y enlaces de híper texto.

- Distribuir pequeñas colecciones de información tipo bases de datos y proveer, para esa información, una interfaz amigable con el usuario.

- Controlar la navegación cuando ocurre en varias *frames*, complementos o *applets* de Java basados en la selección del usuario dentro del documento HTML.

- Pre procesar información que será posteriormente enviada al servidor.

- Cambiar el contenido y estilo de los navegadores de modo dinámico e instantáneo como respuesta a una interacción del usuario.

Aún así no es posible:

- Establecer o reconocer las preferencias de los navegadores, las características visuales de la ventana principal, botones de acción o impresoras.

- Correr aplicaciones en el computador cliente.

17 Scripting and ajax [en línea]. World Wide Web Consortium [Consultado Oct 2010] Disponible en línea: http://www.w3.org/standards/webdesign/script

- Leer o escribir archivos o directorios en cualquiera de los computadores, clientes o servidores.

- Capturar la información transmitida por un servidor para retransmitirla.

- Enviar correos electrónicos secretos desde quienes visitan la página hacia uno.

Lo que vuelve a JavaScript una herramienta sumamente importante, es el hecho que se ha vuelto un estándar sobre otros lenguajes interpretados de la web, por lo tanto es mejor soportado por los distintos navegadores, sistemas operativos y dispositivos existentes.

2.3.4 LAMP. Hace referencia a la combinación de un grupo de programas o API's de código abierto los cuales, en combinación, forman una plataforma muy robusta para la la implementación y el desarrollo web. Estos programas son Apache, MySQL y PHP.

Su robustez radica en que por medio del libre compartir de información, programadores de todas partes contribuyen a crear piezas de programas muy poderosas y eficientes, que son a su vez disponibles para todo el mundo, quienes por medio del reporte de errores y otras sugerencias, otorgan características que mejoraran y evolucionan el producto continuamente.

2.3.4.1 Apache. Actúa como el servidor Web, siendo su principal trabajo analizar la información requerida por un explorador web y mostrar los resultados correctos de acuerdo al código de ese archivo. Entre otras características se encuentran:

- Protección por contraseñas de las páginas para múltiples usuarios,

- Páginas de error personalizadas,

- Mostrar código en varios niveles de HTML, y ser capaz de determinar hasta qué niveles puede aceptar el explorador,

- Uso y registro de errores en formatos diversos y a la medida.

- Almacenamiento virtual para direcciones IP diferentes y ubicadas en el mismo servidor.

- Directivas de clasificación de directorios para varios archivos.

- Re escritura de URL's sin límite específico.

2.3.4.2 PHP. (PHP: *Hypertext Preprocessor* ó PHP: preprocesador de hipertexto) es un lenguaje de programación interpretado que corre en el servidor, el cual convierte un sitio de internet en dinámico gracias al procesamiento de información en el servidor. Muy popular por su rápida curva de aprendizaje y cantidad de funciones que en ocasiones superan a sus pares con licencias comerciales.

2.3.4.3 MySQL. (SQL refiere a *Structured Query Language* o Lenguaje de consulta estructurado) Es la aplicación constructora de bases de datos, que complementa estas dos ya mencionadas en el proceso de almacenar y mostrar grandes cantidades de información y seguir siendo transparente para el usuario.

2.3.4.4 L. El último elemento de LAMP, la “L” hace referencia a los sistemas operativos basados en Linux, los cuales se prefieren por:

- Multi arquitectura. Pueden correr en sistemas X86, X64, Itanium, SPARC, Alpha, PowerPC, basados en 68k, ARM; y básicamente en cualquier sistema con un micro procesador incorporado.

- Son eficientes, rápidos y usan efectivamente el hardware.

- A diferencia de otros sistemas, no poseen licencias ni acuerdos de uso, que restringen acceso a los procesos que se ejecutan, por lo que permiten conocer, modificar y personalizar todo lo que ocurre en cualquier nivel del sistema.

- Su robustez, pueden permanecer encendidos (incluso meses) sin presentar ninguna falla y dispone de una gran cantidad de aplicaciones, también libres, que pueden complacer a cualquier tipo de usuario.

- Soportan todas las características de un sistema moderno: memoria virtual, hilos, múltiples procesadores, redes avanzadas, varios lenguajes de

programación, interfaces de usuarios gráficas, entre tantos.

- Pueden coexistir con cualquier sistema Windows, Mac OSX, BSD en el mismo computador.

- Pueden acceder a los archivos de Windows y trabajar con estos sistemas operativos u otros conectados en la red sin presentar ningún problema.

- Existe una importante línea de soporte y documentación, bibliografía, numerosos grupos y organizaciones, los cuales brindan ayuda y herramientas para que toda persona, principiante o experto, pueda colaborar en el desarrollo y aprendizaje de este sistema.[18]

2.3.5 Visual Studio (.NET). Visual Studio es el entorno de desarrollo de Microsoft de aplicaciones y entornos web, permite crear aplicaciones rápida y fácilmente a través de técnicas como agarrar – arrastrar, buscadores, y otros controles y herramientas amigables al usuario.

Es una herramienta que puede conseguirse únicamente para sistemas Microsoft Windows, en cualquiera de sus versiones, principalmente en dos modos: versiones gratuitas básicas, específicas para cada lenguaje, y unas más comerciales, junto a herramientas de depuración y de apoyo al desarrollo.[19]

El *Visual Studio framework* provee una interfaz de programación para los servicios y API's de Windows e, integra un número de tecnologías que han emergido y evolucionado por Microsoft desde los 90's. Esta plataforma consiste de cuatro grupos de productos separados:

- Herramientas de desarrollo y librerías: El conjunto de lenguajes, entre ellos C#, J# y VisualBasic.NET; un grupo de herramientas de desarrollo, incluyendo Visual Studio .NET; una librería de clases completa para crear servicios y aplicaciones web y para Windows; y el CLR (*Common Lenguage Runtime*).

- Servicios web: Se ofrecen servicios web comerciales, específicamente la iniciativa de Servicios .NET; que por un pago, los desarrolladores pueden usar estos servicios para construir las aplicaciones que lo requieran.

18 WELSH, Matt; et al. Running Linux. 4 ed. California, O'Reilly, 2002. 692p.

19 Visual Studio Home [en línea], Microsoft Corporation [Consultado Oct 2010]. Disponible en Internet: http://www.microsoft.com/visualstudio/en-us/

- Servidores especializados: Un conjunto de servidores habilitados para Visual Studio, incluyendo el Servidor SQL, Exchange Server, Biztalk Server, ente otros. Proveen una funcionalidad especializada para el almacenamiento racionalizado de información, email, y comercio B2B.

- Dispositivos: Habilitar Visual Studio para dispositivos diferentes a un computador, como celulares o consolas de video.

La plataforma Visual Studio fue desarrollada pensando en las siguientes necesidades de la industria:

- Computación distribuida: Provee servicios remotos y de arquitecturas web que explotan los estándares web como HTTP, XML, SOAP y WSOL, lo que permite el desarrollo de aplicaciones cliente/servidor multiusuario robustas.

- Componentización: Simplifica la integración de componentes de software desarrollado por distintos vendedores y soporta el desarrollo de aplicaciones distribuidas.

- Servicios empresariales: Permiten el desarrollo de aplicaciones a la escala de la industria sin necesidad de escribir código para manejar transacciones, seguridad o pulir.

- Cambiar paradigmas web: Busca permitir el intercambio de funcionalidades a través de diferentes plataformas web, dispositivos y lenguajes de programación.

- Madurez de la industria de las TI: Soporta la interoperabilidad, escala, seguridad y administración de las aplicaciones web que necesitan las empresas de gran escala.[20]

20 LAM, Hoang y THAI, Thuan L. NET Framework Essentials. 3 ed. Estados Unidos, O'Reilly, 2003. 284p.

3.DISEÑO METODOLÓGICO

3.1 CARACTERIZACIÓN DE LOS ROBOTS MÓVILES DISPONIBLES EN LA UAO.

La Universidad Autónoma de Occidente cuenta con una serie de robots móviles con distintas capacidades y servicios. Es necesario, entonces, realizar una caracterización de cada uno de estos para poder seleccionar con mejor criterio el o los robots útiles para el montaje del laboratorio así como planear las posibles aplicaciones que se puedan realizar con ellos.

3.1.1 LEGO MINDSTORM NXT. Un kit de robótica desarrollado por LEGO. basado en bloques de construcción, permite al usuario crear diferentes configuraciones de robots para diferentes aplicaciones.

Figura 3. Lego Nxt armado como un robot móvil.

Cuenta con un sistema de procesamiento principal, el ladrillo inteligente NXT, a este llegan todos los datos de los sensores y se mandan las señales a los

actuadores.

3.1.1.1 Microprocesador. Cuenta con un procesador principal Atmel ARM de 32 bits, el AT91SAM7S256 a 48MHz, el cual cuenta con 256Kb de memoria FLASH y 64 Kb de memoria RAM.

Adicional trae un coprocesador Atmel AVR de 8 bits, el ATmega48 a 8 MHz, con 4Kb de memoria FLASH y 512 bytes de memoria RAM.

3.1.1.2 Sensores. El ladrillo inteligente puede recibir información hasta de cuatro sensores diferentes, entre ellos se pueden encontrar. Un sensor de toque, sensor de luz infrarrojo, sensor de sonido, sensor de temperatura, sensor ultrasónico, receptor infrarrojo, brújula, sensor de color, acelerómetro o un giroscopio.

El ladrillo incorpora cuatro botones con los cuales es posible desplazarse entre los menús de firmware o realizar ciertas acciones cuando un programa esté corriendo.

Figura 4. Lego Nxt armado como un robot manipulador.

3.1.1.3 Actuadores y alimentación. El ladrillo inteligente permite conectar hasta tres servomotores.

Posee un altavoz el cual permite generar sonidos con resolución de 8 bits a un rango de frecuencias de 2 a 16 KHz. Además tiene una pantalla de 24 x 40.6 mm en blanco y negro con 100 x 64 píxeles.

El ladrillo inteligente requiere de seis baterías AA para su funcionamiento, es posible también, adquirir una batería recargable.

3.1.1.4 Comunicaciones. El NXT dispone de comunicación vía *bluetooth* por medio de un chip *CSR BlueCore* 4 V2. Por este medio puede enlazarse con con un máximo de tres dispositivos, aunque se comunica con un dispositivo a la vez a una distancia máxima de diez metros.

El cuarto puerto de sensores del ladrillo inteligente puede configurarse para trabajar como un puerto RS485 bidireccional de alta velocidad (921.6 bits/s).

3.1.1.5 Programación. LEGO en colaboración con **National Instruments** han desarrollado una aplicación que permite desarrollar unos programas basados en bloques. Esta aplicación ha sido recientemente liberada.

Existen además proyectos de terceros, creados con el propósito de aumentar las capacidades y funcionalidades del robot. Por ejemplo el proyecto **Lejo**s o una librería de trabajo en **MatLab**.

3.1.2 VEX. Es un sistema de diseño de robots, que permite a los estudiantes aprender los conceptos de robótica y diseñar sus robots adaptándole los sensores que requiera el diseño.

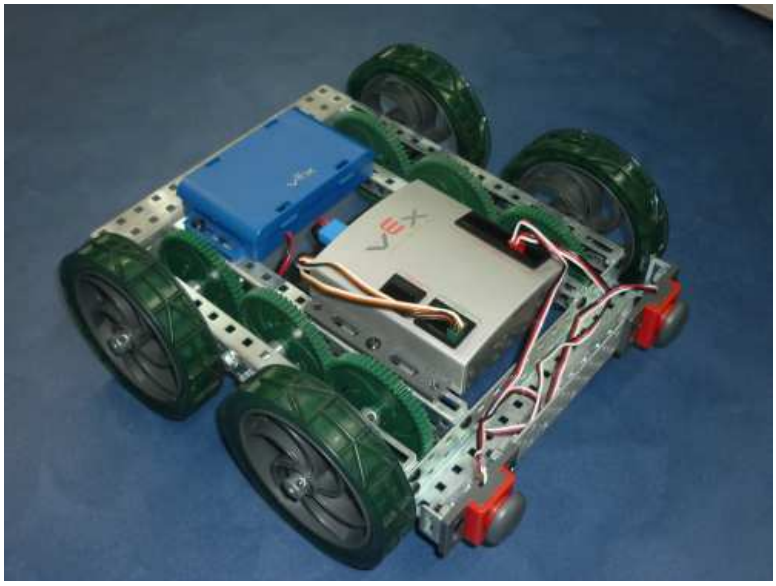
La idea principal del proyecto fue crear un robot que se adaptara a las necesidades de FIRST (*For Inspiration and Recognition of Science and Technology*. Por la inspiración y el reconocimiento de la ciencia y la tecnología), una organización que diseña programas innovadores y accesibles que permitan a nuevas personas adentrarse al mundo de la ciencia, tecnología, ingeniería y la matemática. Mientras adquieren confianza, conocimiento y habilidades para la vida.

3.1.2.1 Microprocesador. El VEX incluye dos posibles sistemas de procesamiento central. Uno basado en PIC y otro en Cortex. La UAO dispone actualmente de robots VEX únicamente con unidades PIC.

La unidad PIC es un microcontrolador PICmicro PIC18F8520 con 1800 bytes + 1024 bytes EE2 de memoria RAM y 32 Kb de memoria Flash.

3.1.2.2 Sensores. La unidad PIC cuenta con 16 puertos que pueden ser configurados como pines de entrada / salida digital o entradas análogas. Si el pin es configurado como entrada digital, la frecuencia de entrada puede ser hasta de 50 KHz; por otro lado, como entradas análogas ofrece una resolución de 10 bits y tiempo de acceso de cada 10 microsegundos.

Figura 5. Robot Vex ensamblado como un robot móvil.



Cuenta también con seis entradas de interrupciones, las cuales pueden ser usadas para medir los cambios en las entradas por medio de las interrupciones.

Entre los sensores que se pueden acoplar se encuentran: finales de carrera, *bumpers*, ultrasonidos, *encoder*, sensor de luz, potenciómetros, acelerómetro análogo y seguidores de línea infrarrojos.

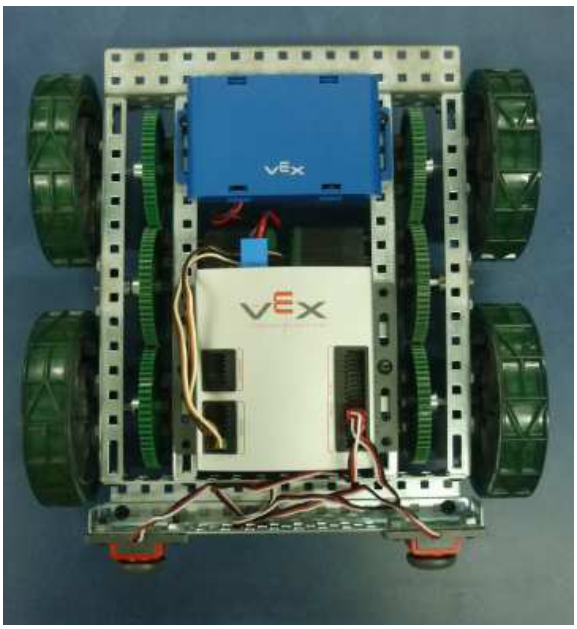
3.1.2.3 Actuadores y alimentación. La unidad PIC brinda la posibilidad de conectar hasta ocho servomotores los cuales incluyen juegos de piñones y engranes adicionales para poder realizar conversiones velocidad/torque según la necesidad.

La alimentación requiere de seis baterías AA o es posible emplear un módulo de alimentación que se puede adquirir para el robot.

3.1.2.4 Comunicaciones. Los VEX disponen de un módulo de comunicación inalámbrica por medio de radio frecuencia, sin embargo están diseñados para recibir órdenes de un control remoto.

Disponen de un puerto de entrada y un puerto de salida serial a 115Kb/s.

Figura 6. Robot Vex armado como un robot móvil. Visto desde arriba.



3.1.2.5 Programación. La programación del robot es posible en lenguaje C y Assembler. El enlace para subir los programas al robot se realiza por medio de un cable que se puede conectar al puerto serial o al USB por medio de un adaptador USB – Serial que incluyen, y al robot por medio de un puerto específico con conexión directa a la unidad PIC.

La programación del robot es recomendada en programas como Mplab o RobotC.

3.1.2.6 Expansiones. El VEX está diseñado como un sistema modular, por lo tanto, es posible agregar nuevas piezas a la estructura, modificar la unidad central, crear y adaptar circuitos propios y de terceros.

3.1.3 E-puck. Un robot móvil de plataforma abierta cuyo objetivo de desarrollo fue alcanzar los siguientes criterios: tamaño de escritorio, amplio rango de posibilidades desde el punto de vista de un ingeniero hasta un estudiante, amigable con el usuario, bajo costo, ideología abierta. Priorizando la flexibilidad y el tamaño de escritorio.

3.1.3.1 Microcontrolador. Dispone de un microcontrolador dsPIC 30F6014A de la empresa microchip. Este embebe un procesador de 16 bits con archivo de registro de 16 entradas y un procesador digital de señales. Corre a una velocidad de 64MHz y provee 16 MIPS picos de poder de procesamiento. Memoria RAM de 8Kb y Flash de 144Kb.

Figura 7. Robot e-puck.

3.1.3.2 Sensores. Ocho sensores infrarrojos de proximidad, ubicados alrededor de la carrocería del robot.

Un acelerómetro 3D.

Tres micrófonos para captura de sonidos por triangulación.

Cámara a color CMOS con 640 x 480 píxeles de resolución integrada. Sin embargo, debido a la capacidad de memoria del microprocesador la captura a color se realiza a una resolución de 40 x 40 píxeles a 4 *frames* por segundo, pudiendo duplicarse la cantidad de capturas si la captura de la imagen es en

blanco y negro.

3.1.3.3 Actuadores. Dos motores paso a paso con una resolución de 1000 pasos por revolución.

Un altavoz conectado a un códec de audio.

Ocho LED rojos con la posibilidad de modular su intensidad lumínica.

Un grupo de LED verdes.

Un LED rojo ubicado detrás de la cámara.

3.1.3.4 Comunicaciones. Conector con interfaz para un *in – circuit debugger*.

Receptor infrarrojo para control remoto.

Interfaz RS232

Conexión *Bluetooth*

3.1.3.5 Interfaz con el usuario. Dos LED muestran el estado de la batería.

Botón de r*eset*.

Interruptor giratorio de 16 posiciones para especificar un número de 4 bits.

3.1.3.6 Programación. El e - puck incluye librerías para manipular el hardware en lenguaje C. Para la programación del microcontrolador emplea un *bootloader* a través de Bluetooth. Además incluye la herramienta BTcom para poder manipularlo desde un computador.

Es posible encontrar entornos de simulación como webots, player/stage y Enki. Siendo estos dos últimos entornos de código abierto.

3.1.3.7 Extensiones. Un escáner rotatorio, equipado con sensores de triangulación infrarrojos.

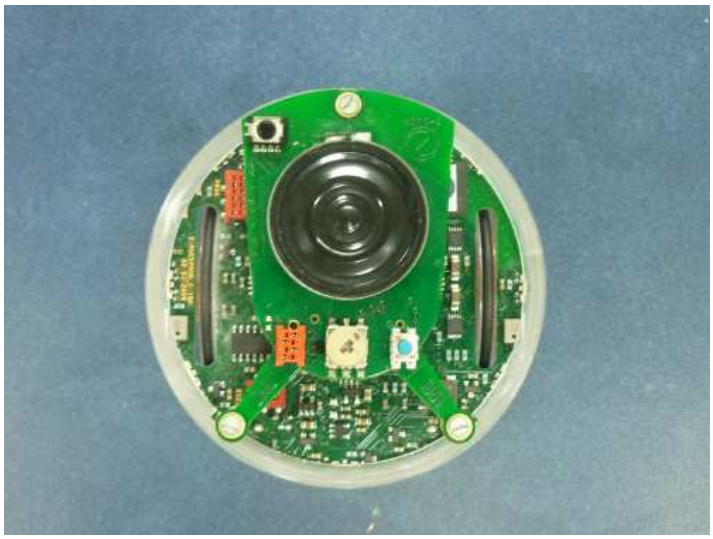
Una torre con tres cámaras lineales.

Interfaz de radio de comunicaciones *ZigBee*.

Un anillo de LED RGB.

Torre de visión omnidireccional.

Figura 8. Robot e-puck. Vista superior.



3.1.4 Moway. Son robots desarrollados para la docencia e investigación principalmente en temas de micro robótica y robótica autónoma. Así se pretende que pueda ser utilizado por principiantes e investigadores que quieran comenzar a experimentar con la robótica o desarrollar ejercicios de mediana complejidad.

El cuerpo presenta un diseño compacto para permitirle moverse con agilidad, ayudando a evitar que se quede estancado. Pretende también brindar un diseño modular y así las capacidades y posibilidades de los desarrollos con el robot.

3.1.4.1 Microcontrolador. Emplea un microcontrolador PIC16F876A de Microchip que trabaja a 4MHz. Permite además la programación mediante un adaptador especial llamado **Base Moway**.

3.1.4.2 Sensores. Dos sensores de línea. Un par de opto acopladores montados en la parte inferior delantera del robot.

Dos detectores de obstáculos. Conformados por un emisor infrarrojo, KPA3010-F3C de Kingbright, y dos receptores, PT100F0MP de Sharp, ubicados en los extremos del Moway.

Figura 9. Robot Moway.

Sensor se Luz, APDS-9002 de Avago Technologies, ubicado en una pequeña abertura con media luna en la parte superior del chasis.

3.1.4.3 Actuadores. Dispone de dos servo motores con las siguientes funcionalidades: contador de avance general, control de velocidad, tiempo, distancia recorrida y ángulo.

Dos LED rojos frontales. Y un LED bicolor superior, ubicado en la misma abertura del sensor de luz.

3.1.4.4 Comunicaciones. La **Base Moway** permite comunicación por RF.

3.1.4.5 Programación. Para descargar los programas al Moway es necesario tener instalado el *software Moway Center* y la **Base Moway**.

Dispone de la capacidad de usar un programa incluido con capacidad de programación gráfica.

Es posible también realizar programas en lenguajes Assembler y C.

Figura 10. Tres robots moway. Para trabajo en robótica cooperativa.

3.1.4.6 Extensiones. El Moway dispone de un conector de expansión que permite conectar dispositivos I2C o SPI. Además existe el módulo RF BZI-RF2GH4 que permite la comunicación del Moway con otros Moway y el con el PC mediante la Base Moway.

Un Pad libre, ubicado dentro del robot al lado de los sensores de línea permite la conexión de circuitos electrónicos por parte del usuario.

3.1.5 Spykee. Un robot autodenominado “espía” por su capacidad de visión artificial y conectividad WiFi para ser accedido desde cualquier parte del mundo. Diseñado para monitorear, enviar fotos y videos, también es capaz de reproducir

audio, auto recargar su batería y ser armado de diversas formas, sin cambiar su funcionalidad.[21]

3.1.5.1 Microcontrolador. Utiliza un procesador ARM9 a 200 MHz con conectividad 802.11b/g y USB2. Su placa base está basada en la CNX 111 de WaveStorm.

3.1.5.2 Sensores. El Spykee dispone de pocos sensores a comparación de los otros robots disponibles, debido a su concepción como un robot teleoperado por personas. Sin embargo es posible encontrar en él un receptor infrarrojo, un detector de carga de la batería, un micrófono y una cámara de video.

Figura 11. Robot Spykee.

La cámara posee una resolución de 320x240 pixeles con una rata de captura de 15 fps. Puede tomar fotos por modo manual o automático con la detección de movimiento activada. Además es posible obtener 8 tipos de imágenes dependiendo del filtro implementado, entre ellos térmico o deformación. Finalmente está se encuentra conectada al sistema principal por medio de un

21 Hardware - Spykee developer wiki [en línea]. Sypkee wiki [Consultado Nov 2010]. Disponible en Internet: http://spykee.duskofsolace.com/index.php?title=Hardware

cable USB.[22]

3.1.5.3 Actuadores. Dispone de un par de motores y ruedas en configuración de orugas. Además dispone de altavoces, 3 luces LED en el cuerpo y un LED para mejorar la visión de la cámara.

3.1.5.4 Comunicaciones. Tiene conectividad inalámbrica WiFi 802.11 b/g.

3.1.5.5 Programación. El robot viene con un programa por defecto que permite únicamente la conectividad remota con un programa para computador ofrecido por el fabricante. Sin embargo, recientemente liberaron el código para permitir modificaciones en las funcionalidades por parte de los posean acceso al robot, siendo posible, incluso, instalar un sistema Linux en él.

3.1.6 Bioloid. Un robot avanzado con propósito educativo, con la concepción de ser armado con distintas articulaciones y eslabones unidos por servomotores.

Es característico porque permite ensamblar humanoides que mantienen fácilmente su equilibrio y postura al caminar.

3.1.6.1 Microcontrolador. Utiliza un microcontrolador ATMEGA 2561.

Está incluido en un bloque central, denominado CM-5, donde se encuentran una gama de sensores, así como puertos de entrada y salida análogos para otros sensores, el conector para los servomotores, y 6 botones para permitir al robot ejecutar diferentes acciones.

3.1.6.2 Sensores. El bloque CM-5 incorpora un micrófono, un sensor de temperatura y uno de voltaje. Además permite conectar sensores infrarrojos para detección de obstáculos y medición de distancia, giroscopios y las lecturas de los servomotores.

3.1.6.3 Actuadores. Los actuadores que esencialmente utiliza son los servomotores *Dynamixel*, para su principal función que es el movimiento. Dependiendo del kit que se haya adquirido es posible encontrar entre 4 y 18 de

22 Spykee Self assembly [en línea], Spykee.org The 100% unofficial Spykee site [Consultado en Oct 2010]. Disponible en Internet: http://www.spykee.org/AboutSpykee/Selfassembly/tabid/234/Default.aspx

ellos, la universidad cuenta con 18, lo que permite crear hasta 26 configuraciones de robots, entre ellos: araña, grúa, dinosaurio, perro y humanoide.

También es posible, por medio de las salidas en el CM-5, conectar diferentes actuadores de terceros.

Figura 12. Robot Bioloid armado como humanoide.

3.1.6.4 Comunicaciones. Disponen de un módulo ZigBee ZIG-111 para habilitar la comunicación inalámbrica.

3.1.6.5 Programación. Incluye un software llamado **RoboPlus** el cual permite programación gráfica y en lenguaje C, diseñado con el objetivo que sea fácil de usar incluso por aquellos que poseen poca experiencia con computadores hasta aquellos profesionales en lenguaje C.

Este programa permite a su vez hacer simulaciones y pre visualizaciones de las

acciones y respuestas que va a ejecutar el robot.[23]

Sin embargo es posible instalar en el robot un sistema operativo diferente, lo que brinda la opción de poder correr sistemas personalizados de terceros o propios, o Linux.

3.2 ESPECIFICACIONES DEL PRODUCTO.

3.2.1 La identificación de necesidades. Por medio de consultas con estudiantes, personal del área de laboratorios y docentes del área de automática y electrónica de la universidad se llegó a las siguientes necesidades latentes para el proyecto:

- Los robots podrán ser accedidos desde diversas partes, diferentes al laboratorio de robótica móvil.

- La aplicación será multiplataforma.

- La aplicación podrá correr en equipos de bajo rendimiento.

- La aplicación tendrá una interfaz amigable.

- La aplicación permitirá la interacción usuario - robot en tiempo real.

- Los estudiantes pueden escoger, entre los robots disponibles, el robot con el cual quieran trabajar.

- Los estudiantes podrán manipular al robot sin la necesidad de subir un programa a este.

- Los estudiantes podrán ejecutar sus propias rutinas en el robot.

- Los estudiantes podrán correr rutinas de demostración en los robots.

23 Robotis [en línea]. Robotis Inc [Consultado Oct 2010]. Disponible en Internet: http://www.robotis.com/xe/bioloid_en

- Los estudiantes podrán conocer el estado de los sensores del robot.

- Los estudiantes podrán ver al robot.

- Los robots dispondrán de un espacio adecuado.

Como es requerido por parte del proceso del desarrollo de especificaciones del producto para una buena percepción de problemas y ventajas, se localizaron restricciones y se aplicaron métricas a las necesidades.

3.2.2 Restricciones. El tiempo de duración de la carga de las baterías de los robots.

- La comunicación en tiempo real (cliente - servidor - robot).

- La posibilidad de establecer comunicación inalámbrica (servidor - robot).

- Algunos robots requieren de una orden física que permite programarlos.

3.2.3 Métricas. En la generación de las métricas, basadas en las necesidades, se asignaron también unos valores de medida y propusieron unos valores marginales considerando el estado del arte de la tecnología actual y los patrones observados de los estudiantes que podrían ser usuarios del sistema.

Cuadro 2. Generación de métricas y sus valores.

Métrica	**Medida**	**Valor marginal**
Seguridad	Restricciones del espacio de trabajo.	
Cantidad de usuarios	Numero conectados	>1

Cuadro 2. (Continuación)

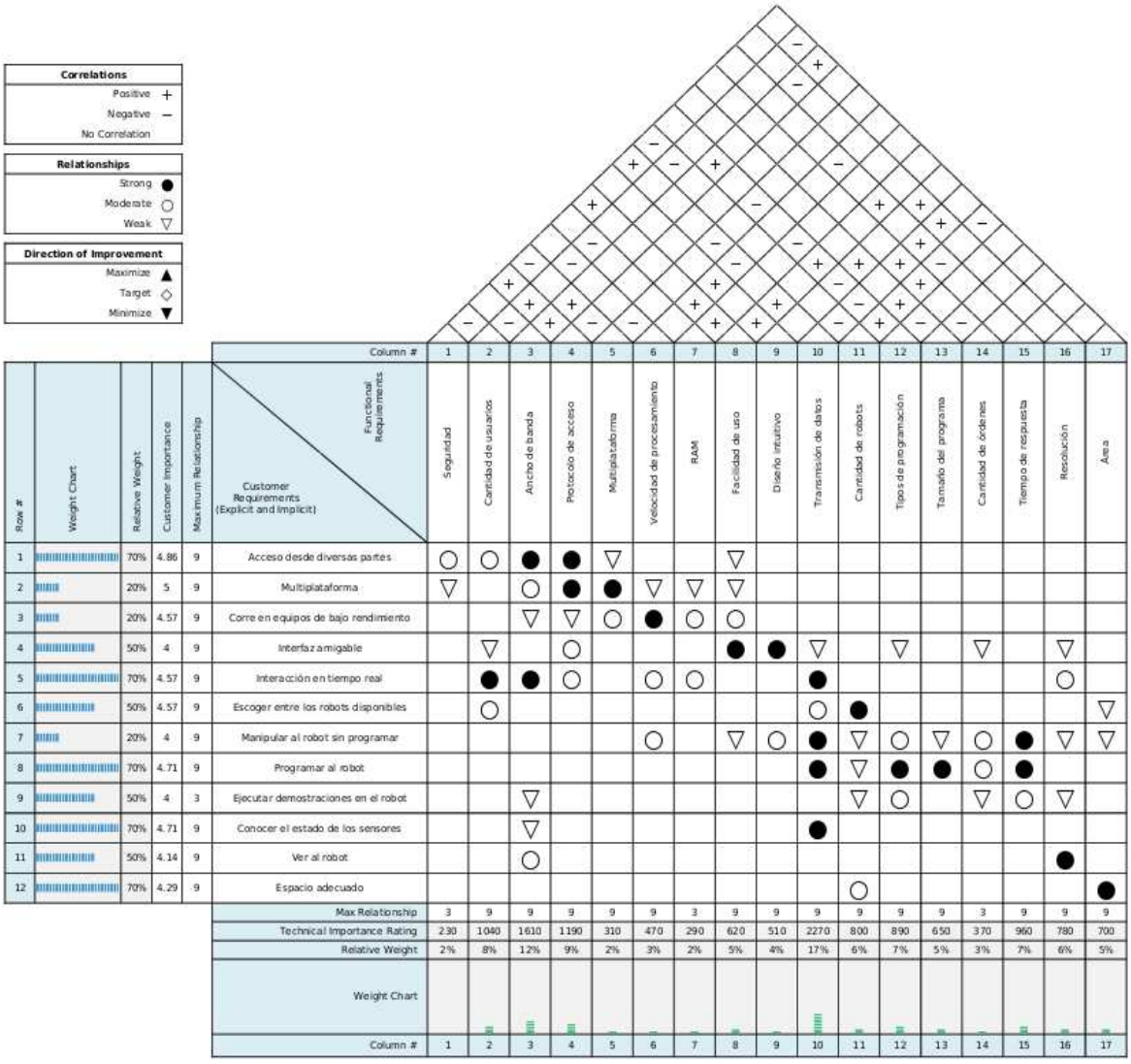
Ancho de banda	Velocidad de conexión	10/100/1000 Mbps cableado o 54mbps inalámbrico
Ruta o protocolo de acceso	API	Explorador de internet, Java, C o conexión remota.
Multiplataforma	Sistemas operativos soportados.	WinXP, Win7, Linux, MacOSX
Velocidad de procesamiento de los equipos	Velocidad del procesador en GHz	>1.6GHz
RAM de los equipos	Capacidad de RAM	256 MB
Facilidad de uso	Número de clics	<5
Diseño intuitivo	Herramientas que faciliten la interacción	Imágenes, animaciones, letras
Transmisión de datos al robot	Velocidad de la conexión.	bytes/s
Cantidad de robots	Disponibles para el uso.	=> 1
Tamaño de programa	La RAM y ROM del robot.	>500 líneas de código, >50 variables de almacenamiento.

Cuadro 2. (Continuación)

Cantidad de órdenes .	Posibles a darle al robot.	Adelante, atrás, giro, arriba.
Tiempo de respuesta del robot.	Retardos en el tiempo.	<50ms.
Resolución	Tamaño en pantalla de la interfaz.	<800 x 600 pixels/mm ^{2}
Área	Del espacio de trabajo	=> 1 m^2
Tipos de programación	Donde corren los programas.	Local o remota.

3.2.4 Generando la QFD

Figura 13. Generación de la QFD a partir de las necesidades de los clientes y las métricas planteadas.



Con una encuesta realizada y este resultado se pudo determinar que las necesidades más importantes son: En acceso desde varias partes (1), La interacción en tiempo real (5), la programación del robot (8), conocer los estados de los sensores (10) y el espacio adecuado (12).

También, a partir del análisis de QFD se llegó a la conclusión que las características técnicas que tienen mayor importancia son aquellas involucradas con la transmisión de datos: Ancho de banda (3), protocolo de acceso (4) y transmisión de datos (10).

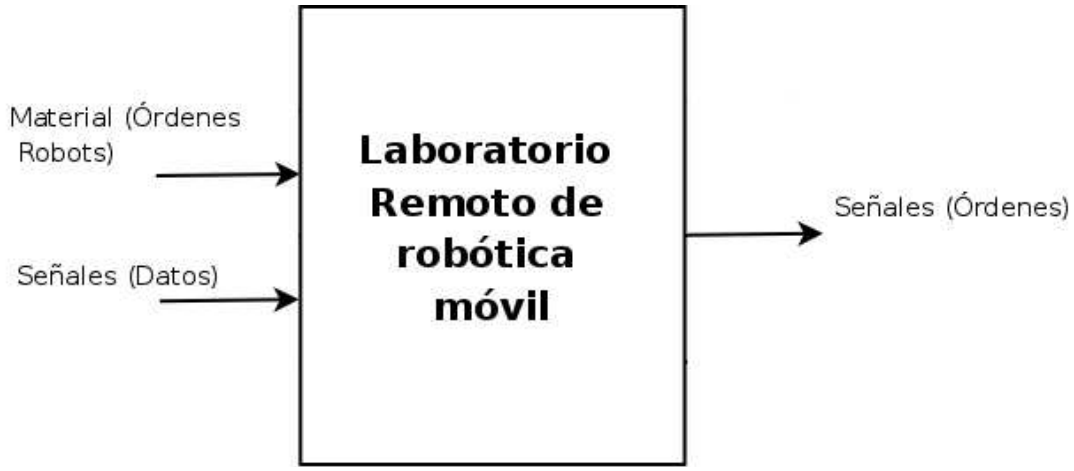
3.3 GENERACIÓN Y SELECCIÓN DE CONCEPTOS

Siguiendo con el proceso del diseño recurrente, con las necesidades identificadas, se deben identificar las funciones que tendrá la plataforma y evaluar las alternativas tecnológicas que existen para realizarlo.

3.3.1 Descomposición funcional. Por medio de este método es posible partir desde un modelo simple, el cual poco a poco se desglosa desde elementos que a simple vista parecen más complejos, en subelementos más sencillos, que permiten trabajar de modo más cómodo y enfocarse en los elementos más críticos.

Se comienza con una caja negra (figura 14), la cual es una simple representación de un sistema, que contiene entradas y salidas transformadas de materiales, energías y señales.

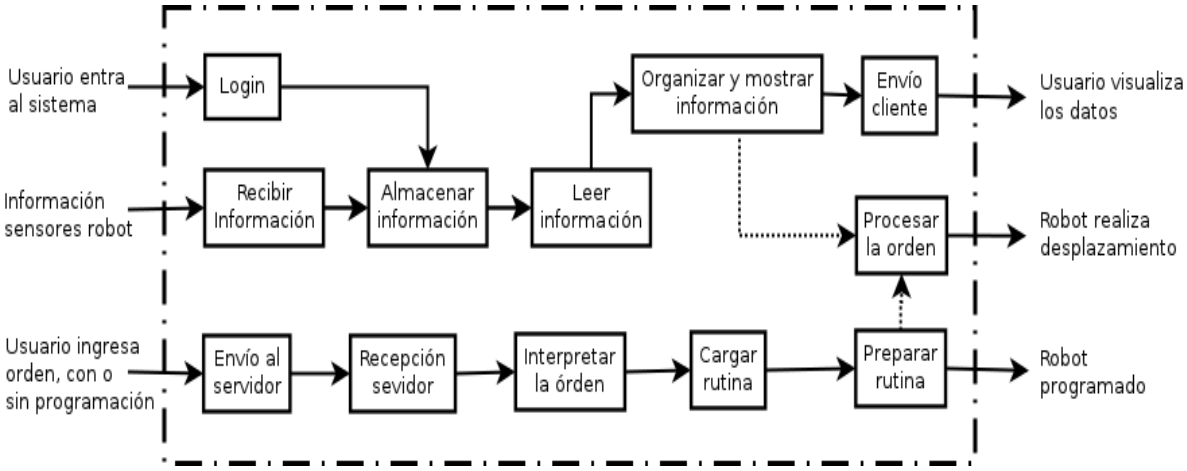
Figura 14. Caja negra



Así, un modelo más elaborado (figura 15):

3.3.2 Exploración sistematizada El paso siguiente en el proceso del diseño es probar los conceptos que mejor se desempeñen con las funciones que ya se han determinado o que, gracias a la metodología concurrente, permita redefinir las mismas. Se definen de manera muy corta las diferentes funciones y las posibles soluciones que puedan tener, sin embargo ya se ha realizado un primer filtro, que elimina las alternativas menos adecuadas.

Figura 15. Descomposición funcional de la caja negra



Alimentar sistemas de cómputo. Para que los equipos de cómputo, donde está instalada la aplicación, funcionen necesitan energía eléctrica.

- Sistema eléctrico de 110 V AC disponible en la universidad

- Instalación de sistemas de paneles solares.

Alimentar robots móviles. Esencial para que los robots móviles funcionen.

- Los diversos sistemas de baterías y recargas que incluye cada robot de la universidad.

- Adecuar los robots a un uso de energía cableada.

- Instalar nuevos sistemas de alimentación por baterías a los robots.

Login. Etapa de seguridad, solo para usuarios con permisos y cuentas activas, para poder ingresar al curso y escoger robots.

- Cuentas de la universidad.
- Usuarios almacenados en una base de datos.
- IP's reservadas.
- Sala de cómputo en la U dedicada a los entornos remotos.

Recibir información. El servidor debe interpretar la información recibida por el robot y ser capaz de manipularla.

- Programas en diversos lenguajes. (C. JAVA, Basic)
- Aplicaciones de terceros.

Almacenar información. La información recogida por el sistema es almacenada, para posterior manipulación por cualquiera de los servicios.

- Archivos de texto.
- Bases de datos.
- Archivos temporales.

Leer información. La información almacenada es seleccionada adecuadamente y enviada al servicio necesario

- Rutina en diversos lenguajes (Java, C, Basic) .

Organizar y mostrar información. El usuario desea conocer el estado de los sensores, demás dispositivos del robot y el laboratorio, además requiere que se

muestre de manera organizada.

- Aplicación WEB (html, css)
- Aplicación de terceros.
- Aplicación de diversos lenguajes. (Java, C, Basic).

Método de conexión. La forma en que ambas partes (cliente – servidor) pueden comunicarse.

- Internet.
- Intranet.
- Ad - Hoc.

Envío de información entre cliente y servidor. El modo que la información es enviada del cliente al servidor y viceversa.

- Archivos planos de texto.
- XML.
- Hoja de Excel.

Recepción servidor. La información es recibida por el servidor.

- Apache.
- Windows Server (IIS).
- Tomcat.

- Cherokee.

- Lighttpd.

Interpretar la orden (envío al servidor). El usuario envía diversas órdenes: movimiento, programas.

- Aplicación JAVA.

- Aplicación WEB (PHP).

- Formulario WEB (html, css).

- Aplicación FLASH.

Cargar rutina. El usuario puede implementar sus propias rutinas para el robot y probarlas en el mismo.

- Desde el cliente.

- Desde el servidor.

- Subir la aplicación directamente al robot.

Preparar rutina. El servidor alista las acciones necesarias para ejecutar correctamente la orden que desea el usuario.

- Rutina en diversos lenguajes (Basic, C, Java)

- Rutinas o programas de terceros.

Procesar la orden. El servidor con la información de sensores, datos del usuario y capacidades del robot. Organiza y envía las órdenes a ejecutar por el robot.

- Rutina en diversos lenguajes (Basic, C, Java)

- Rutinas o programas de terceros.

3.3.3 Prueba de conceptos. Se llevaron a cabo diferentes investigaciones y pruebas, con las cuales se buscó que todas las funciones se desempeñen de mejor manera y sean compatibles entre sí.

Comenzando con los criterios de los robots existentes:

- Los Legos NXT son una primera opción altamente posible, su programación es flexible, su modularidad para implementar diferentes prácticas, su conectividad cableada e inalámbrica, y su concepción como un sistema para principiantes en robótica. Sin embargo, su demanda es muy alta por los estudiantes, lo que complica disponer del robot para realizar las prácticas remotas hasta no tener una plataforma inicial completa.

- Los Moway, aunque sus alternativas de programación lo convierten en una opción atractiva, su conectividad dependiente de la torre transmisora y la imposibilidad de usarla con rutinas propias o externas. Obliga a ceder ante esa opción.

- La conectividad del Spykee es un gran atractivo, sin embargo el hecho que solo pueda ser usado con el programa que incluye, no brinda posibilidad alguna, por el momento, de implementar rutinas de robótica móvil.

- El VEX no dispone medios adecuados de comunicación inalámbrica, además, su configuración es muy grande, lo que podría generar problemas al momento de definir el espacio de trabajo.

- El Bioloid, por no disponer de sistemas de ruedas, no se acomoda a los propósitos de este trabajo.

- Finalmente, solo los e-puck reúnen las condiciones necesarias para poder implementar satisfactoriamente el laboratorio remoto. Su amplia gama de sensores, su conectividad y configuración.

Con esta primera selección se obtiene un primer enfoque sobre las principales premisas que tendrá el sistema en adelante: conexión Bluetooth y programación en C para los robots.

Siguiendo este orden se buscan los sistemas operativos que mejor soporten Bluetooth. El robot e-puck se comunica por medio del protocolo RFCOMM, lo que significa que es posible desarrollar la aplicación en cualquier sistema operativo. y bajo cualquier plataforma.

Una preselección de los sistemas operativos arroja los siguientes resultados:

- Symbian es un sistema operativo para celulares, específicamente Nokia, por lo tanto no se toma en cuenta para este desarrollo.

- Mac OS X es una alternativa viable, sin embargo, durante el proceso de desarrollo no se pudo tener acceso a estos sistemas para realizar las pruebas de viabilidad.

- Windows Xp presenta buen soporte para el desarrollo con las herramientas y protocolos, sin embargo es un producto que se encuentra en el final de su vida útil y pronto dejará de recibir soporte y actualizaciones.

- Windows 7 es un sistema operativo reciente y el más distribuido en la actualidad por los diferentes fabricantes de computadores, lo que lo convierte en uno, con ciclo de vida vigente, de los más usados en la actualidad. Sin embargo presenta grandes problemas de incompatibilidad con sistemas de software y hardware, por lo que muchas empresas no han desarrollado herramientas compatibles con este sistema operativo todavía.

- Los sistemas operativos basados en *kernel* Linux se caracterizan por su bajo costo, gran estabilidad, buena protección contra software mal intencionado y, gran soporte para el desarrollo de distintos protocolos y estándares.

Figura 16. Robot e-puck. Seleccionado como principal y más adecuado para el desarrollo de la plataforma.

Otras herramientas que se hallaron durante el estudio de los sistemas Linux fueron las de diagnóstico de conexiones y flujo de datos para las conexiones en redes y, en especial, *bluetooth*.

Finalmente, su presentación como el sistema operativo más popular en los servidores del mundo lo convierte en una alternativa bastante fiable para el desarrollo de este trabajo.

Es necesario, a continuación, definir un primer modelo sin detalle de cómo se desea implementar la arquitectura del sistema completo para proseguir con la selección de conceptos.

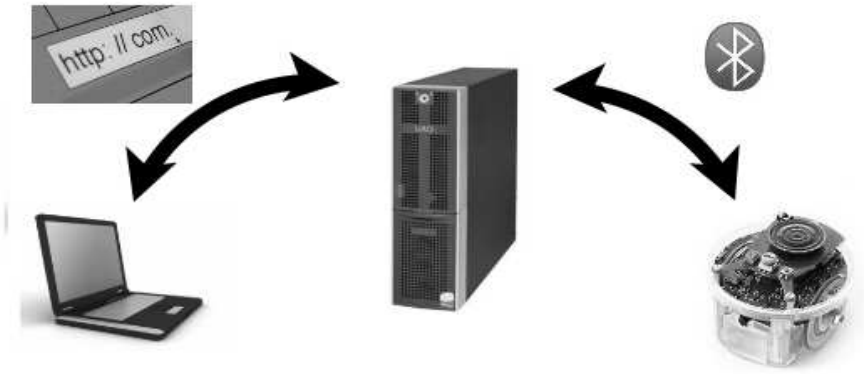
Pensando en un ambiente remoto, cabe pensar en un modelo cliente – servidor, así se pueden ofrecer algunas alternativas de diseño:

- Ambas partes (cliente y servidor) utilizan una aplicación Java para enviarse la información.

- Se instala un servidor web en el servidor y el cliente accede a la aplicación a través de un navegador web.

• El cliente corre una aplicación, la cual envía y recibe información al servidor que dispone de un servidor web instalado.

Figura 17. Primer modelo de conectividad del laboratorio remoto



Siguiendo la premisa de las necesidades del cliente, la transmisión de datos, se analiza no solo la compatibilidad con las distintas herramientas requeridas para la plataforma, se tienen en cuenta también simplicidad del diseño, robustez y velocidad de transmisión de datos.

Una primera opción de desarrollo de la aplicación fue en Java, debido a sus grandes capacidades para el manejo de redes, el soporte y el conocimiento previo de la herramienta, por parte del cliente y del servidor. Sin embargo, el bajo soporte de hardware dificulta su uso e implementación.

Por lo tanto, se desarrolló un procedimiento de selección evaluando todas las alternativas posibles, considerando, como premisa, la tendencia que se está viviendo actualmente en las tecnologías de la información, donde la web ha obtenido suficientes librerías y soporte como para migrar o desarrollar aplicaciones en ella.

3.3.4 Selección de conceptos. Se presenta a continuación de la selección de conceptos para el servicio del cliente (Cuadro 3) y servidor (Cuadro 4), teniendo en cuenta la disposición y accesibilidad al hardware y herramientas requeridas.

Cuadro 3. Criterio de selección de la aplicación del servidor

	Apache	**Java**	**Win Server**	**C/C++**
Fácil Instalación	4	3	4	4
Configuración	4	4	3	4
Estabilidad	4	3	3	3
Instalación Multiplataforma	5	4	2	4
Prestaciones	3	4	3	4
Costo	5	5	2	5
Soporte	5	5	4	5
Afinidad	4	3	1	3
Consumo Recursos	5	3	4	4
TOTAL	**39**	**34**	**26**	**36**

Estos resultados comprueban la ventaja que tiene no solo para el usuario sino para la aplicación correr sus programas principalmente en el lado del servidor. No requiere mayores criterios de configuración para el cliente, el control y configuración de los componentes principales del programa son más fáciles, aumenta la compatibilidad de la aplicación con diferentes dispositivos y mejora la seguridad.

Cuadro 4. Criterio de selección de la aplicación cliente

	Java	**Explorador Web**	**C/C++**
Multiplataforma	4	5	4
Costo	5	5	5
Consumo Recursos	3	5	4
TOTAL	**12**	**15**	**13**

Sin embargo esto genera, para el propósito de este ejercicio, el requisito de correr un programa que corra en el servidor. Aunque Apache no corre ASP, es igual necesario hacer una comparación directa para asegurar resultados correctos. El análisis se puede ver en el Cuadro 5.

Cuadro 5. Criterios de selección de lenguajes servidor

	PHP	**ASP**
Costo	5	2
Multiplataforma	4	2
Soporte	4	4
Configuración	4	5
TOTAL	**17**	**13**

Finalmente se realiza un análisis del lenguaje programación que realice un puente de comunicación entre el robot y el programa del servidor, ya que este solo no es suficientemente capaz de realizar la tarea, además que se agrega una capa adicional de seguridad y se asegura que extraños no puedan acceder al hardware desde la red (Cuadro 6).

El medio por el cual ambos lenguajes, servidor y programación, comparten información es XML ya que además de ser estándar, es almacenado como un archivo texto, posee buen soporte para los lenguajes y es fácil de entender.

Durante la selección del sistema operativo, fue necesario considerar herramientas que permitieran diagnosticar el estado y ver los paquetes que eran transmitidos a través las conexiones *Bluetooth*. Los sistemas GNU/Linux con el paquete **Bluez** disponen de estas ayudas. Sumando a esta utilidad la estabilidad, su mayor participación en el mercado de servidores, capacidad de fácil integración y todas las características populares en estos sistemas operativos; se decide usar un sistema GNU/Linux para la implementación del computador servidor del laboratorio.

Cuadro 6. Criterio selección del lenguaje de programación

	Java	**C/C++**	**Visual Basic**
Multiplataforma	5	5	3
Soporte	4	5	4
Integración con hardware	3	5	4
Afinidad	4	4	2
TOTAL	**16**	**19**	**13**

Entre las diferentes versiones de GNU/Linux existentes se seleccionó GNU/Linux **Debian** principalmente por las siguientes causas:

- Es reconocido por su estabilidad.

- Dispone de cientos de programas compatibles y probados para que se integren lo mejor posible con el SO.

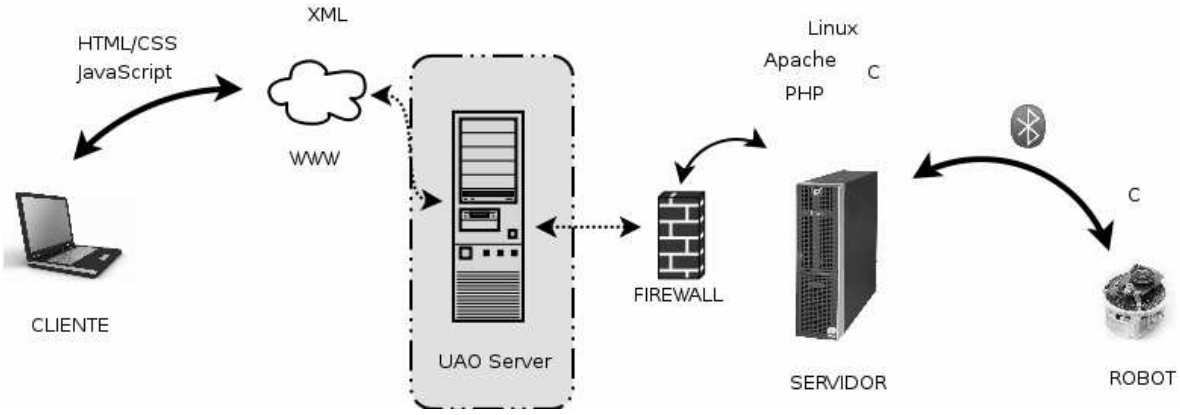
- La forma de instalar programas es muy fácil.

- Su filosofía como distribución libre.

- Su compatibilidad con múltiples arquitecturas de hardware.

Siendo así, el modelo que finalmente queda para la implementación del laboratorio remoto se puede ver en la figura 18:

Figura 18. Esquema preliminar de la arquitectura del Laboratorio Remoto



Se espera que las características y elementos empleados cumplan con los diferentes estándares que requieran, logrando así un mayor entendimiento y capacidad de mantenimiento. De igual manera este se concibe como un diseño modular o abierto, de modo que si se desea cambiar o simplificar algún módulo o etapa del esquema planteado, no se presente un mayor problema.

Elegir un computador como servidor y establecer en él un programa servidor brinda también la posibilidad de implementar diversos protocolos de comunicación web y así aumentar la compatibilidad con diferentes tipos de clientes.

La etapa del servidor UAO representa los requisitos y limitantes que impone la universidad para realizar una conexión remota desde internet, por lo que las pruebas se realizan en una red de área local y conexiones Ad-Hoc.

El uso de corta fuegos brinda una característica de protección adicional, ya permite bloquear los puertos innecesarios, limitar los permisos de conexión de los programas y realizar filtros de IP para negar el acceso a posibles usuarios peligrosos.

Esta implementación brinda un beneficio adicional, reduce al mínimo todos los costos de licencias en la implementación del prototipo y los requisitos de hardware se reducen también ya que un sistema Linux requiere de menor características del sistema para funcionar.

3.4 IMPLEMENTACIÓN DEL LABORATORIO REMOTO DE ROBÓTICA MÓVIL

3.4.1 Instalación de los programas necesarios. Con el sistema Debian GNU/Linux instalado, se procede a instalar los programas y aplicaciones necesarias para el desarrollo del proyecto. Para ello existen dos opciones, usar la línea de comando o usar el programa *Synaptics*. Este programa permite, por medio de una interfaz gráfica, seleccionar todos los programas que se requieran instalar o borrar. Las aplicaciones que fueron instaladas son (cuadro 7):

Cuadro 7. Programas instalados para el funcionamiento de la plataforma.

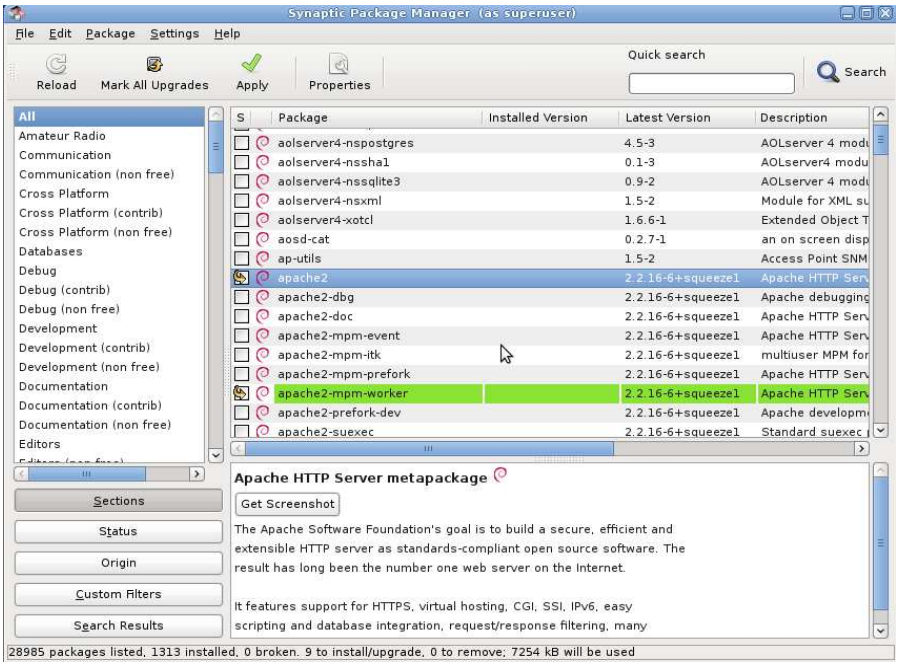
gcc	cpp	gdb	apache2	libxml2	libxml2-doc
bluez-hcidump	bluez-firmware	bluez-utils	bluez-gnome	bluetooth	gnome-bluetooth
libbluetooth2	libbluetooth-dev	apache2.2-common		apache2-utils	librxpat1
apache2-mpm-prefork		libexpat1	ssl-cert	libapache2-mod-php	
php5-common	php5-curl	php5-dev	php5-gd	php5-imagick	php5-mcrypt
php5-memcache		php5-mhash	php5-snmp	php5-sqlite	php5-xmlrpc
php5-xsl	vim	vim-common	iptables	iceweasel	epiphany
dia	gconfig	gdebi	unzip	unrar	

Adicionalmente estas muestran otras dependencias, las cuales deben ser también instaladas para que los programas funcionen correctamente, también es posible que se hayan omitido algunos programas ya que estos vienen instalados por defecto con el sistema operativo.

El servidor apache requiere que se le asigne un nombre a la máquina en su archivo de configuración el cual, con permisos de administrador se asigna, escribiendo en la última línea archivo /etc/apache2/apache2.conf[3].

Para la instalación de las librerías que permiten compilar las rutinas que serán enviadas al e-puck se deben buscar, descargar e instalar los siguientes programas:

Figura 19. El administrador de paquetes Synaptics. Escogiendo la aplicación Apache



- pic30-binutils_2.01-1_i386.deb
- pic30-gcc_2.01-1_i386.deb

- pic30-support_2.01-1_all.deb[2]

También deben ser descargadas las librerías propias del robot y el programa para subir los programas a este, herramientas que pueden ser descargadas desde la página oficial del robot[1].

3.4.2 Configuración del dispositivo Bluetooth y comunicación con el e-puck.

Teniendo permisos de super usuario (administrador), lo primero que se debe hacer es hacer que el computador esté visible para los otros dispositivos Bluetooth. Esto es posible con el comando:

```
# hciconfig hci0 piscan
```

Para asegurarse que el computador conecta correctamente con el e-puck es recomendable modificar o crear el archivo /etc/bluetooth/rfcomm.conf de la siguiente manera:

```
rfcomm0 {

        bind yes;

        device 10:00:E8:6C:D7:FA;

        channel 1;

        comment “e-puck_1600”;

}
```

Como se trabaja con el e-puck con identificación 1600, la configuración queda de esa manera, sin embargo, si se trabaja con otro robot se debe modificar el *device* y *comment* para que concuerden con los datos del robot. Tras esta modificación se debe reiniciar el módulo *bluetooth*, esta acción se realiza con el comando: # /etc/init.d/bluetooth restart

Para comprobar que el equipo se puede encontrar al e-puck y revisar su número

de identificación b*luetooth* se puede usar el siguiente comando: # hcitool scan

Si llegase a existir algún problema o fallo en la conexión se puede ejecutar: # rfcomm release rfcomm0

Y si se desea realizar alguna conexión con el robot para una aplicación aparte: # rfcomm connect rfcomm0

3.4.3 Preparar las librerías del e-puck. Lo primero que se debe hacer es instalar las diferentes librerías .deb del pic30 descargadas. Esto se puede lograr ubicándose en la carpeta donde residen los archivos y ejecutando cada paquete en el mismo orden que se mostró en la descarga. También es posible instalarlas con el comando: # dpkg -i *paquete*.deb

En segundo lugar se debe escoge una localización, preferiblemente dentro de las carpetas de usuario (/home/javier/), donde se desee tener almacenada las librerías propias del robot (/home/javier/epuck/lib/), el programa para subir las rutinas al e-puck (/home/javier/epuck/epuckuploadbt/) y las rutinas que se vayan a programar (/home/javier/remotelab/). De este modo se podría, por ejemplo, descomprimir el archivo “epucklib.zip” en la carpeta /home/javier/.

Una vez realizados ambos pasos, se deben compilar las diferentes librerías del e-puck para que estas puedan ser reconocidas satisfactoriamente por el compilador cuando se creen las rutinas propias. Esto es posible ejecutando los comandos:

$ cd /home/javier/epuck # para ubicarse en la carpeta donde se encuentran las librerías

$ pic30-elf-gcc -c -I library -I/usr/pic30-elf/include *ubicación/del/archivo.c* -o *ubicación/del/archivo.o* -Wall

Si se desea compilar una librería que está en Assembly (ej. los uart) se puede ejecutar:

$ pic30-elf-as -p30F6014A -g -I library/uart -I/usr/share/pic30-support/inc library/uart/e_init_uart1.s -o library/uart/e_init_uart1.o

Donde se debe modificar primero, en los archivos .s y .inc el nombre del

dispositivo referencia “p30f6014a.inc” a “p30f6014A.inc”.

Todos estos procesos de compilar rutinas y preparar las conexiones Bluetooth deben hacerse una única vez, si el proceso no presenta errores.

3.4.4 Compilar las rutinas del robot. Una vez creado un programa para que el e-puck realice cualquier tarea se debe compilar en tres etapas. Se considera igualmente que se encuentra en la carpeta /home/javier/epuck/:

El primer paso es la creación del .o, con el que se corrigen todos los errores de sintaxis: $ pic30-elf-gcc -c -I library -I/usr/pic30-elf/lib *ubicación/del/programa.c* -o *ubicación/del/archivo.o*

Luego viene la creación del binario donde se deben incluir todos los .o correspondientes a las librerías que fueron realacionadas en el programa y el .o recién creado (remoto.o), un ejemplo: $pic30-elf-gcc library/motor_led/e_init_port.o library/motor_led/e-motors.o library/motor_led/e_led.o library/a_d/e_accelerometer.o library/a_d/e_prox.o library/a_d/e_micro.o library/a_d/e_ad_conv.o library/uart/e_init_uart1.o library/uart/e_uart1_rx_char.o library/uart/e_uart1_tx_char.o programas/remoto.o -L/usr/pic30-elf/lib -T/usr/share/pic30-support/gld/p30f6014a.gld -o programas/remoto.elf

Finalmente, convertir a .hex para que pueda ser interpretado por el robot. Dirigiéndose primero a la ubicación del .elf: $pic30-elf-bin2hex remoto.elf

3.4.5 Subir rutinas al robot. El archivo epucklib.zip incluye un programa escrito en C++ que permite subir los programas al e-puck sin mayor inconvenientes, este debe ser compilado antes de poder ser usado, su nombre es **epuckupladbt**. Este se encuentra en una carpeta con el mismo nombre del archivo. Este proceso de compilación debe realizarse una única vez.

Esto se puede realizar, una vez ubicado en la carpeta del **epuckuploadbt** con el comando: $ g++ -o epuckuploadbt epuckuploadbt.cpp -lbluetooth

Finalmente, para subir un programa se ejecuta la siguiente línea de comando (ubicado en la carpeta epuckuploadbt): $ ./epuckuploadbt */ubicacion/y/nombre/del/archivo*.hex 1600

Donde 1600 es el número de identificación del e-puck.

3.4.6 El programa del e-puck. Su función es recibir las órdenes del computador para generar un cambio en los actuadores y enviar información de los sensores, según le sea requerido.

Se diseña este modelo de funcionamiento con el propósito de establecer una comunicación coordinada entre ambas partes y evitar que se estén enviando datos que en algún momento no sean necesarios, que se lleguen a producir pérdidas de datos o que alguna de las partes mal interprete la información recibida.

El funcionamiento del programa se puede describir de la siguiente forma:

- Cuando inicia el programa, éste ejecuta una rutina de inicialización de sus puertos, que incluye los sensores, motores y los parámetros de la comunicación a establecer.

- Restablece los valores de los motores y sus velocidades en cero (“0”) para evitar que ejecute movimientos no esperados durante el principio de la rutina.

- A partir de este punto, comienza un bucle el en que comienza esperando que exista un bit de entrada en el puerto UART1 que equivale a la comunicación *bluetooth* que está estableciendo con el computador.

- Si los bits que ingresan por el puerto UART1 son una cadena de caracteres válidos, procede a buscar la acción a la que hacen referencia.

- Si los bits corresponden a una actividad equivalente a la solicitud de información de los sensores (proximidad, acelerómetros, infrarrojos, velocidad o pasos), se procede con recolectar la información respectiva de la orden y enviarla nuevamente por el mismo puerto de comunicaciones. De ahí procede a reiniciar el bucle.

- Si los bits hacen referencia a un *set* en los actuadores o acciones del sistema (velocidad de motores, LED, pasos de motores, reinicio general). Ejecuta la orden y vuelve a comenzar el bucle.

El diagrama de flujo que representa gráficamente este proceso se encuentra en los anexos.

3.4.7 Archivos XML. Como medio de comunicación y transmisión de información interna entre la interfaz web de usuario y el programa usuario-robot se emplean archivos planos de texto con formato XML. Siendo dos en total: actuadores.xml que contiene la información que será procesada para enviar al robot y sensores.xml con el contenido correspondiente a los valores de los sensores del robot.

El contenido del archivo sensores.xml presenta la siguiente estructura:

Tabla 1. Estructura del archivo "sensores.xml"

Archivo “sensores.xml”	Descripción
<sensores>	Elemento raíz.
<robot>	Etiqueta para identificación del robot.
<stepr> </stepr>	Pasos del motor derecho.
<stepl> </stepl>	Pasos del motor izquierdo.
<accx> </accx>	Valor del acelerómetro en el eje x.
<accy> </accy>	Valor del acelerómetro en el eje y.
<accz> </accz>	Valor del acelerómetro en el eje z.
<prox0> </prox0>	Valor del sensor de proximidad 0.
<prox1> </prox1>	Valor del sensor de proximidad 1.
<prox2> </prox2>	Valor del sensor de proximidad 2.
<prox3> </prox3>	Valor del sensor de proximidad 3.
<prox4> </prox4>	Valor del sensor de proximidad 4.
<prox5> </prox5>	Valor del sensor de proximidad 5.
<prox6> </prox6>	Valor del sensor de proximidad 6.
<prox7> </prox7>	Valor del sensor de proximidad 7.
</robot>	Cierre de etiqueta
</sensores>	Cierre de etiqueta

El archivo actuadores.xml fue concebido con una estructura variable dependiendo de la rutina que se desee implementar en el robot, sus presentaciones pueden ser

(Tablas 2, 3 y 4):

Tabla 2. Estructura del archivo "actuadores.xml" para la rutina de teleoperación.

Rutina: Teleoperación	Descripción
<actuadores>	Elemento raíz.
<robot>	Etiqueta para descripción del robot.
<speedr> </speedr>	Velocidad del motor derecho.
<speedl> </speedl>	Velocidad del motor izquierdo.
<rst_stp> </rst_stp>	Reiniciar contador de pasos de motores.
<led0> </led0>	Acción del led0.
<led1> </led1>	Acción del led1.
<led2> </led2>	Acción del led2.
<led3> </led3>	Acción del led3.
<led4> </led4>	Acción del led4.
<led5> </led5>	Acción del led5.
<led6> </led6>	Acción del led6.
<led7> </led7>	Acción del led7.
<confirm> </confirm>	Código de confirmación
<routine> 00 </routine>	Código de rutina.
</robot> </actuadores>	Cierre de etiquetas

Tabla 3. Estructura del archivo "actuadores.xml" para la rutina de cinemáticas.

Rutina: Cinemáticas	Descripción
<actuadores>	Elemento raíz.
<robot>	Etiqueta del robot.
<x0> </x0>	Pose x inicial.
<y0> </y0>	Pose y inicial.
<theta0> </theta0>	Ángulo inicial.
<x1> </x1>	Pose x final.
<y1> </y1>	Pose y final.
<confirm> </confirm>	Código de confirmación.
<routine> 01 </routine>	Código de rutina.
</robot> </actuadores>	Cierre de etiquetas

Tabla 4. Estructura del archivo "actuadores.xml" para la rutina de seguimiento de trayectorias

Rutina: Seguimiento de trayectoria	Descripción
<actuadores>	Elemento raíz.
<robot>	Etiqueta del robot.
<x0> </x0>	Pose x inicial.
<y0> </y0>	Pose y inicial.
<theta0> </theta0>	Ángulo inicial.
<x1> </x1>	Pose 1 en x.
<y1> </y1>	Pose 1 en y.
<x2> </x2>	Pose 2 en x.
<y2> </y2>	Pose 2 en y.
<x3> </x3>	Pose 3 en x.
<y3> </y3>	Pose 3 en y.
<confirm> </confirm>	Código de confirmación.
<routine> 10 </routine>	Código de rutina.
</robot>	Cierre de etiqueta.
</actuadores>	Cierre de etiqueta.

Las diferentes posibilidades de archivos actuadores.xml emplean dos etiquetas para identificarlos: <confirm> para reconocer el momento de creación del archivo y evitar la ejecución de las mismas órdenes reiteradamente, entrando en un error de funcionamiento y, <routine> como un código de confirmación del algoritmo de robótica móvil que se desea emplear.

3.4.8 El programa usuario – robot del servidor. Este programa se encarga de leer los valores que el usuario ha enviado para los actuadores, de recibir y almacenar los datos de lectura de sensores para que sean enviados al cliente.

Para propósitos de este ejercicio, el programa ejecuta las siguientes subrutinas:

- Comunicación *bluetooth*.

- Lectura y escritura de archivos .xml.

- Ejercicio de teleoperación.

- Movimiento hacia un punto del espacio de trabajo.

- Seguimiento de una trayectoria de tres (3) puntos dentro del espacio de trabajo.

Para poder leer los archivos .xml el programa usa las librerías libxml2 previamente descargadas e instaladas. Del mismo modo, deberá tener las librerías lbluetooth2 para poder establecer adecuadamente la conexión *bluetooth*.

De este modo, la concepción del programa usuario – robot se estructura según UML que se muestra en la figura 68 de los anexos. Con esto es posible aumentar la complejidad del programa, cambiar o agregar otras librerías para que se ajusten a desarrollos posteriores; así como la interoperabilidad entre las librerías de manera más asequible.

De este modo, el funcionamiento de este programa está segmentado en tres partes principales:

- Módulo de comunicación con los robots: Encargado de establecer y mantener la comunicación con el robot o los posibles robots conectados al sistema. Para efectos de este trabajo está configurado para operar únicamente con un dispositivo *bluetooth*, pero es posible incorporar otras tecnologías como WiFi, radio frecuencia, módulos de comunicación externos u otros según requieran los diferentes robots a adaptarse.

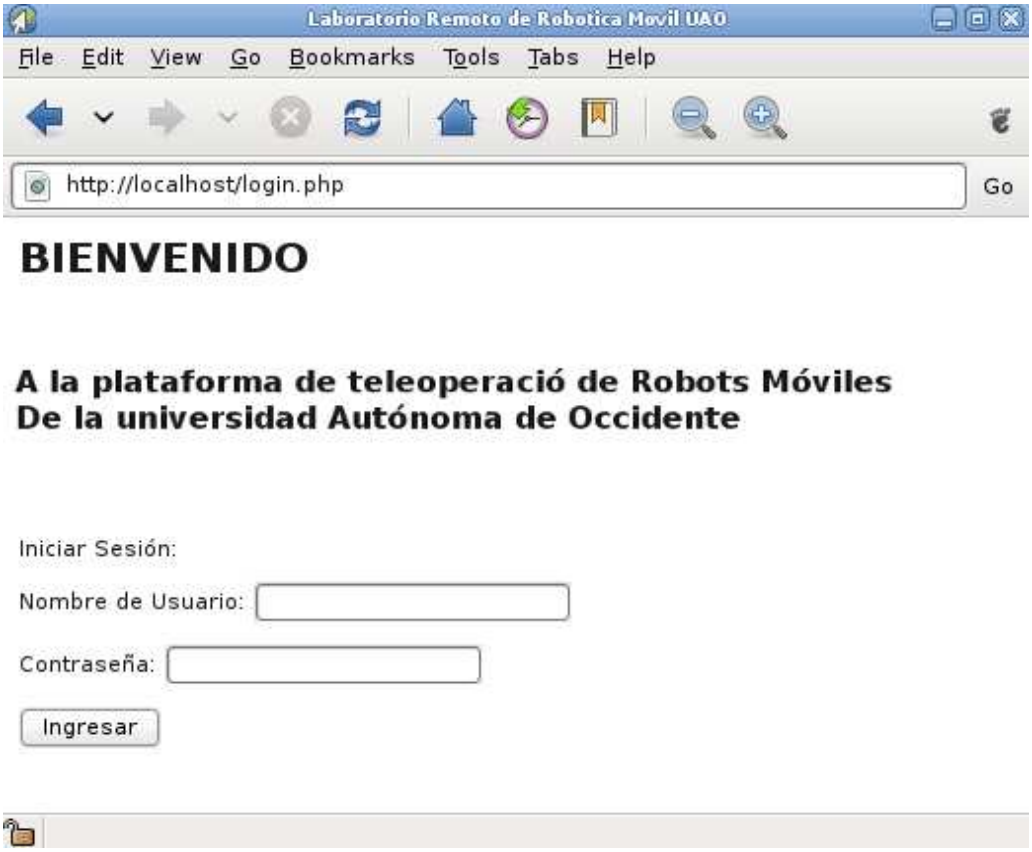
- Lectura y escritura de archivos xml: Sirviendo como puente para envío de información entre el programa y la interfaz de usuario. Debido a los requerimientos, el volumen de información y el uso de estándares, se escogió el uso de archivos de texto plano con la información organizada en formatos xml. Sin embargo, es posible adaptar el modelo al uso de bases de datos u otras herramientas de almacenamiento o minería de datos.

- Ejecución de rutinas: Se encarga de ejecutar las rutinas con las que se demuestran el funcionamiento de los algoritmos de robótica móvil. Se ha implementado la teleoperación del robot y el uso de las cinemáticas directa e inversa para el movimiento hacia un punto y el seguimiento de trayectorias.

3.4.9 Programa de interfaz de usuario. Concebido como la capa de toda la aplicación con la que el usuario final tiene contacto. Busca aprovechar todos los avances en desempeño que se han realizado sobre las herramientas de internet, principalmente los exploradores web, para ello se emplean las herramientas y lenguajes html, css, javascript y php.

La escritura de los archivos actuadores.xml se realiza por medio de pequeñas rutinas php ubicadas dentro del archivo index.php y path.php, mientras que la lectura de sensores.xml se realiza por javascript en el archivo script.js que es el que controla todas las acciones dinámicas de la página.

Figura 20. Página de inicio de sesión



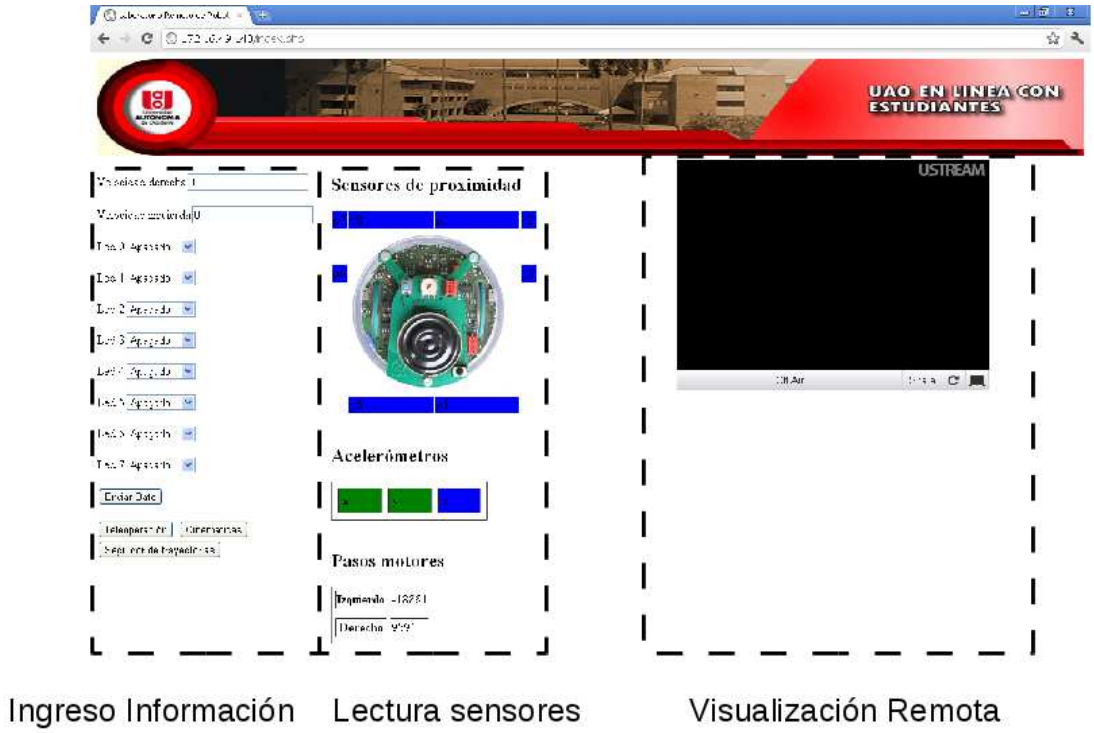
Para ingresar a la aplicación y con el explorador web abierto, debe ingresar la siguiente dirección en la barra de direcciones: http://localhost si se encuentra en el computador servidor, http://172.16.49.143 si se encuentra dentro de la red donde se encuentre el servidor, donde 172.16.49.143 es la ip del servidor del laboratorio

remoto de robótica móvil o, cual sea la dirección ip o de internet asignada por el administrador principal de la red para el acceso desde internet (ej. http://www.uao.edu.co/laboratorioroboticamovil).

Si el acceso es por primera vez durante la sesión, será enviado a la página de confirmación de usuario para el inicio de la sesión como se observa en la figura 20. En este caso, el registro de usuario es: javier, y la contraseña: UAO2011. Si el registro es correcto será enviado a la página principal de la aplicación, de lo contrario tendrá la opción de volver a ingresar los datos de inicio de sesión.

La interfaz principal está divida en tres partes principales, como se puede apreciar en la figura 21: Ingreso de información, lectura de los sensores y visualización remota.

Figura 21. Pantalla principal de la interfaz de usuario.



3.4.9.1 El ingreso de la información. Esta sección es principalmente con la que el usuario presenta mayor interacción, debido a que es ahí donde da instrucciones al sistema de la rutina que va a ejecutar el robot, que pueden ser: teleoperación,

movimiento hacia un punto o seguimiento de trayectorias.

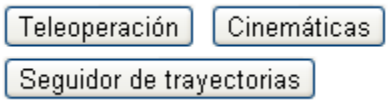
El algoritmo con que el usuario es recibido, una vez carga la página, es el de teleoperación, donde tiene la opción de ajustar la velocidad de cada rueda, reiniciar el contador de pasos de los motores o conmutar cualquiera de los 8 (ocho) LED que incorpora el robot.

Figura 22. Menú del algoritmo de teleoperación.

Puede cambiar la operación que va a efectuar el robot, haciendo clic en el panel de opciones que aparece en la parte inferior de este segmento, como se detalla en

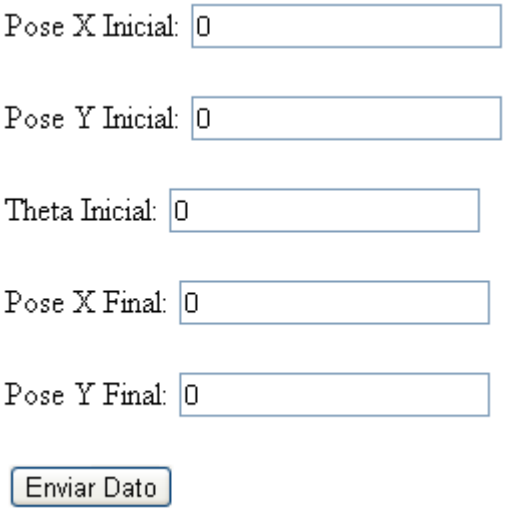
la figura 23.

Figura 23. Botones de selección entre los diversos algoritmos posibles.



La rutina de **movimiento hacia un punto** sirve para demostrar la aplicación de las cinemáticas directa e inversa, por lo tanto el usuario debe ingresar la información sobre la posición "actual" del robot (x0, y0, theta0) y la posición donde se desea que se desplace (x1, y1).

Figura 24. Menú del algoritmo de movimiento hacia un punto.



El seguimiento de una trayectoria consiste en: con una posición "actual" dada (x0, y0, theta0), el usuario observa un cuadro con bordes rojos, que representa el espacio de trabajo del robot, en el cual hará clic sobre los 3 (tres) diferentes puntos que desea que el robot recorra. Para este ejercicio el robot emplea las mismas cinemáticas directa e inversa, pero se espera que de este modo se pueda dar a comprender al estudiante sobre el error que se produce y ejemplificar algunas de sus causas y métodos de disminución del error.

Figura 25. Menú de la rutina: Seguimiento de una trayectoria.

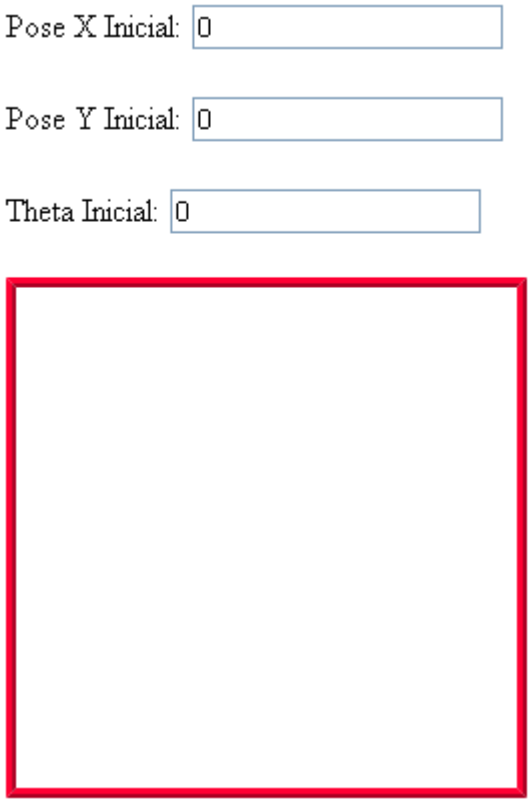
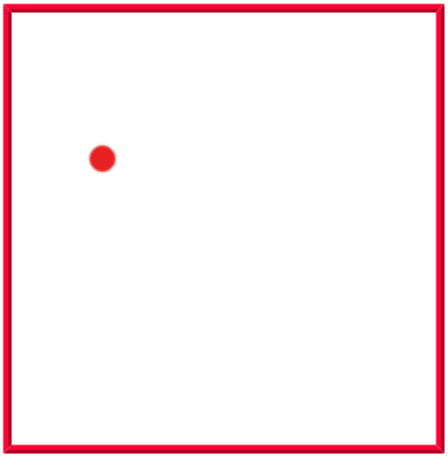
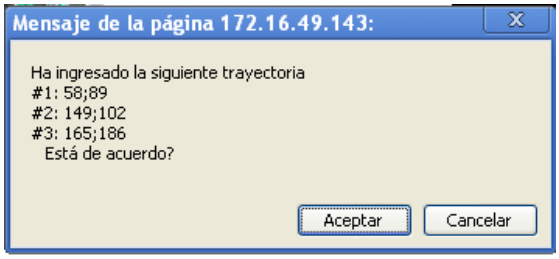


Figura 26. a (Izquierda): Representación de un punto de la trayectoria. b (derecha) Menú de confirmación de los tres puntos seleccionados.





a) b)

3.4.9.2 Lectura de sensores. En este segmento de la aplicación se muestra la información sobre algunos de los sensores que hacen parte del robot, de ellos: los 8 (ocho) sensores infrarrojos de proximidad, el acelerómetro en “x, y, z” y los valores en los contadores de pasos de los motores.

Figura 27. Sector de visualización de sensores.

La información de los sensores pretende brindar mayor información visual para el usuario, por ello, a excepción de los contadores de pasos, la información es visualizada como cambios de tonos de colores para los diferentes sensores. De este modo los sensores de proximidad tienen una transición de colores fríos (como azul y verde) a cálidos (amarillo y rojo) cuando se encuentran más cerca de algún obstáculo y, los acelerómetros presentan un cambio de fríos a cálidos cuando la aceleración en alguno de sus ejes es más intensa. Las transiciones se presentan en el siguiente orden desde el más frío al más cálido: azul, verde, amarillo, naranja y rojo.

3.4.9.3 Visualización remota. Esta característica le permite al usuario ver en tiempo real el estado y funcionamiento del robot y así tener una mayor apreciación y atractivo por el uso del sistema. Para el uso de este servicio se abrió una cuenta en http://www.ustream.tv y se creó un canal al que se puede acceder por la dirección http://www.ustream.tv/channel/robotica-uao.

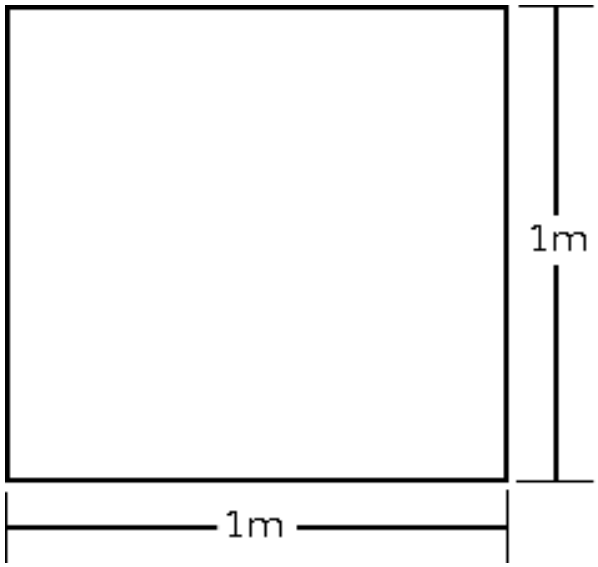
La página permite el uso embebido de las transmisiones en otras páginas web, como la empleada por este laboratorio, de modo que se copió el código de enlace que brinda esta página en la página del laboratorio para hacer posible la transmisión en vivo de los eventos del laboratorio.

Figura 28. Cuadro de visualización remota.

Este método presenta sus ventajas y desventajas. La ventaja consiste en que es más difícil que la calidad del video disminuya sin importar el número de usuarios conectados, por lo que el ancho de banda no depende de la infraestructura de la universidad, sino del sitio que está diseñado para soportar el transporte de un amplio ancho. Su desventaja es, como la universidad no dispone de un ancho de banda suficiente o saturado por el uso excesivo por parte de los estudiantes, y el mismo recorrido que debe hacer el video desde el servidor del laboratorio hasta el usuario final, la transmisión presenta un letargo de 3 a 8 segundos en la visualización de los eventos que ocurren en la planta física.

3.4.10 Configuración del espacio de trabajo. El espacio de trabajo para ejecutar las rutinas del robot es de 1 x 1 m . Su vista superior se representa en la figura 29.

Figura 29. Vista superior del espacio de trabajo para el robot e-puck



Para este ejercicio, el servidor se montó sobre un computador **Dell Optiplex GX 520** el cual fue facilitado para el uso continuo y personalizado a los requerimientos de este trabajo. El computador tiene las siguientes características.

- Procesador Intel Pentium IV con HiperThreading a 3.0 Ghz.
- 4 Gb de memoria RAM DDR2.

- Debian GNU/Linux 6.0.6 (codename Lenny).

- Disco duro Sata de 80Gb.

Figura 30. El computador Dell Optiplex GX520 empleado como servidor con la cámara usb y bluetooth conectados.

El hecho que se disponga de este computador de baja prestaciones (en comparación con la tecnología actual) sirvió para probar el rendimiento y eficiencia de los algoritmos y todo el entorno del laboratorio remoto. Se obtuvo finalmente un buen desempeño, que demuestra que el buen balance escogido para la carga computacional entre el cliente y el servidor.

Las pruebas fueron hechas con un solo usuario conectado, sin embargo se estima que en el caso que existan varios usuarios conectados simultáneamente la carga computacional para este computador sea superior a sus prestaciones, por lo que sería aconsejable distribuir la tarea con otro servidor.

Figura 31. Imágenes del espacio del trabajo del robot e-puck. A la izquierda, imagen con ángulo de perspectiva desde el espectador. A la derecha, vista desde la cámara web.

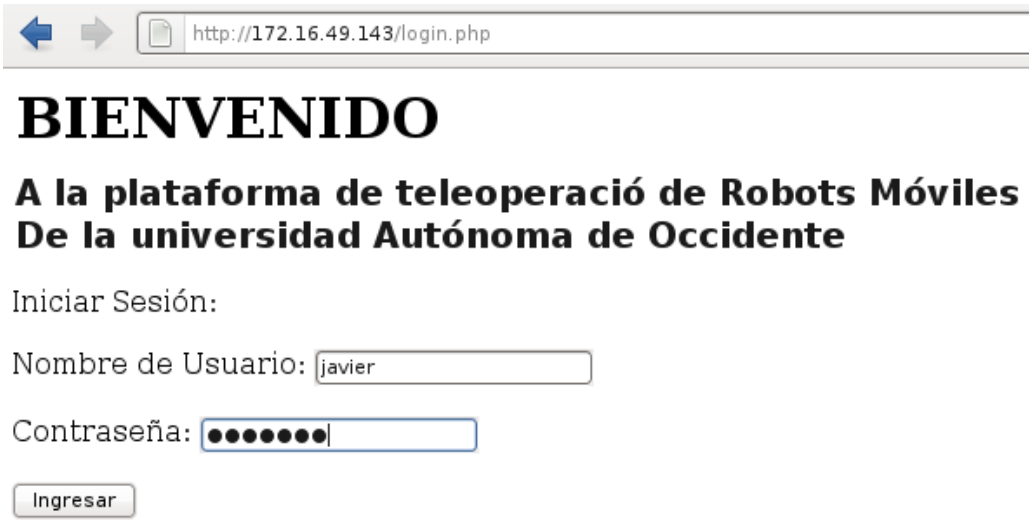
3.5 EJEMPLOS DE FUNCIONAMIENTO DE LA PLATAFORMA

El ingreso a la plataforma se realiza por medio de un explorador web, por ejemplo Firefox, Safari, Explorer, etc. Con la página abierta, se escribe en la barra de direcciones la ip del servidor donde está alojado el laboratorio, en este caso es 172.16.49.143 (figura 32).

Figura 32. Ingreso a la página desde la barra de direcciones

Como es la primera vez que se ingresa al laboratorio y no hay ninguna sesión iniciada, el usuario es redirigido a la página de inicio de sesión como la de la figura 20. En esta se escriben los datos del usuario, en este caso el nombre de usuario es “javier” y contraseña “UAO2011” (figura 33).

Figura 33. Ingresando datos de inicio de sesión



Con los datos ingresados correctamente, la aplicación redirige a la página principal de la plataforma mostrada en la figura 21.

3.5.1 Ejercicio de teleoperación. Es el primer ejercicio con el que el usuario se puede encontrar una vez inicia sesión. En este campo, el usuario ingresa órdenes de movimiento o encendido de los LED directamente al robot. Esto significa que los movimientos o acciones no presentan ningún tipo de control o restricción durante la ejecución de las órdenes.

De esta manera la información que se ingresan para las velocidades son, números enteros en el rango entre [-1000, 1000]. Asimismo, para cada acción de los LED está una lista desplegable con las tres opciones de encender, apagar o conmutar el estado actual de cada uno.

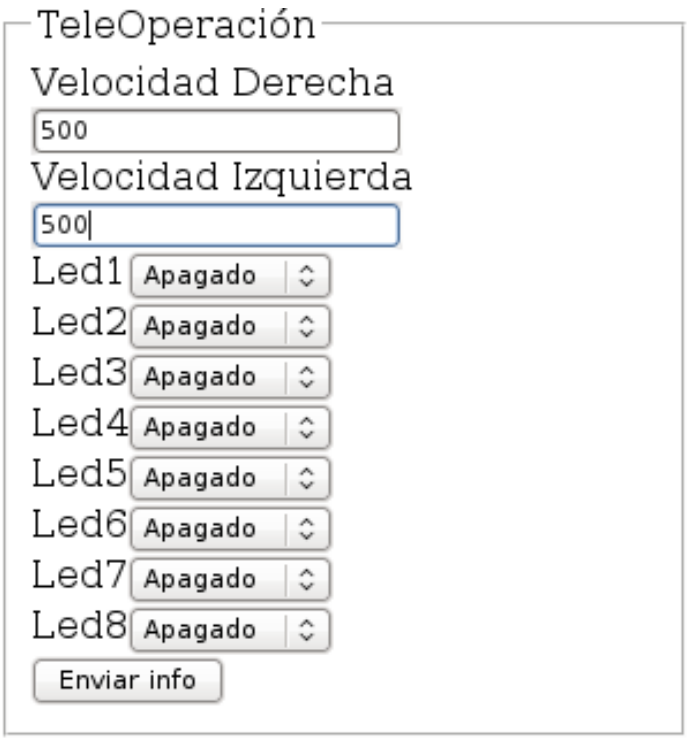
Si se deseara que el robot ejerciera un movimiento rectilíneo hacia adelante, los cuadros se llenarían de la siguiente forma (figura 34):

- Velocidad derecha = 500

- Velocidad izquierda = 500

- Los demás campos sin modificar.

Figura 34. Ejemplo teleoperación. Ingresando velocidad de avance a 500

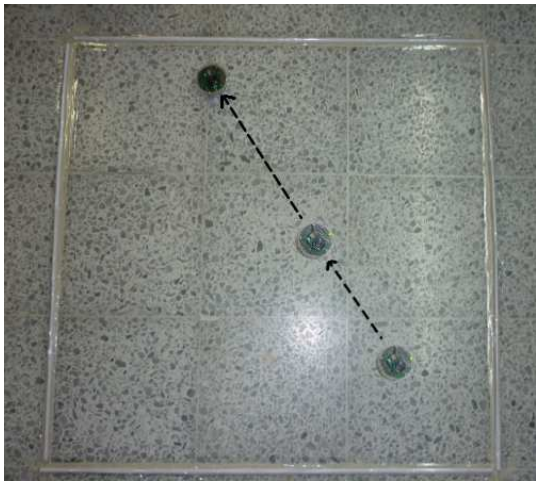


Una vez presionado el botón “Enviar Info” el robot comenzará a moverse en línea recta a una velocidad media como se observa en la secuencia de imágenes de la figura 35.

Por lo tanto, para que el robot pueda realizar movimientos en línea recta, tanto la velocidad derecha como la izquierda deben tener el mismo valor y que este sea mayor de 0.

Figura 35. Ejemplo teleoperación. Secuencia avance robot hacia adelante.

a. b.



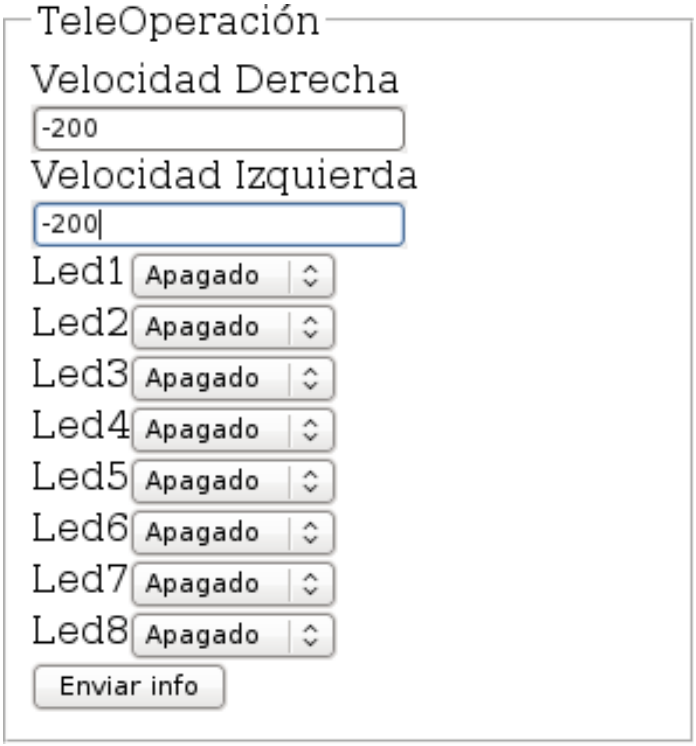
c.

Si el movimiento a realizar fuera en línea recta pero hacia atrás y más despacio que el movimiento anterior, el cuadro de teleoperación se llenaría de la siguiente manera (figura 36):

- Velocidad derecha = -200,
- Velocidad izquierda = -200,

- Los demás campos sin modificar.

Figura 36. Ejemplo teleoperación. Ingresando velocidad de avance a -200



Al presionar el botón “Enviar Info” el robot recibirá la orden y comenzará a desplazarse hacia atrás como aparece en la secuencia de imágenes de la figura 37.

De esta manera se concluye que para que el robot realice movimientos hacia atrás en línea recta, este debe recibir la orden de movimiento a sus ruedas con una misma velocidad menor que 0.

Figura 37. Ejemplo teleoperación. Secuencia de movimiento con avance hacia atrás.

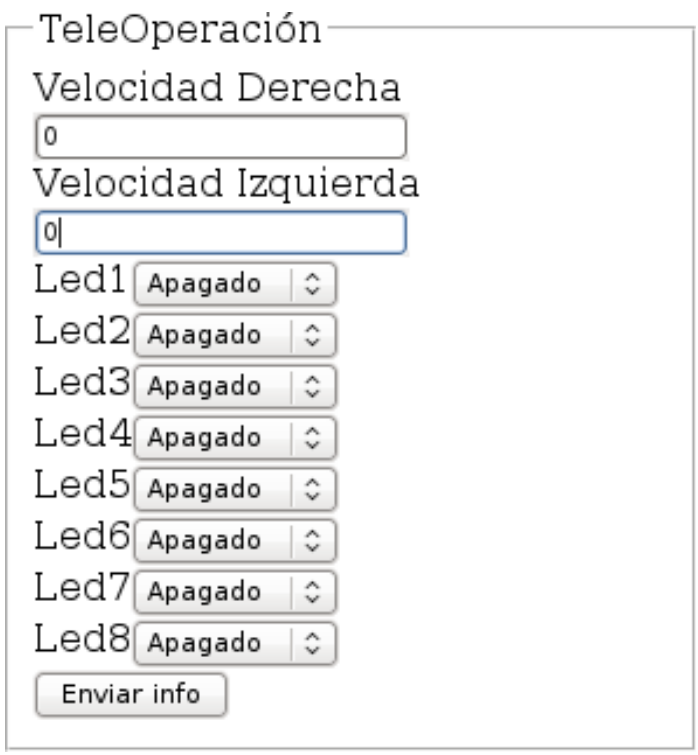
a.

b.

c.

En el momento que se desee detener el robot, se envía a este la orden que ambas ruedas estén en cero como aparece en la figura 38.

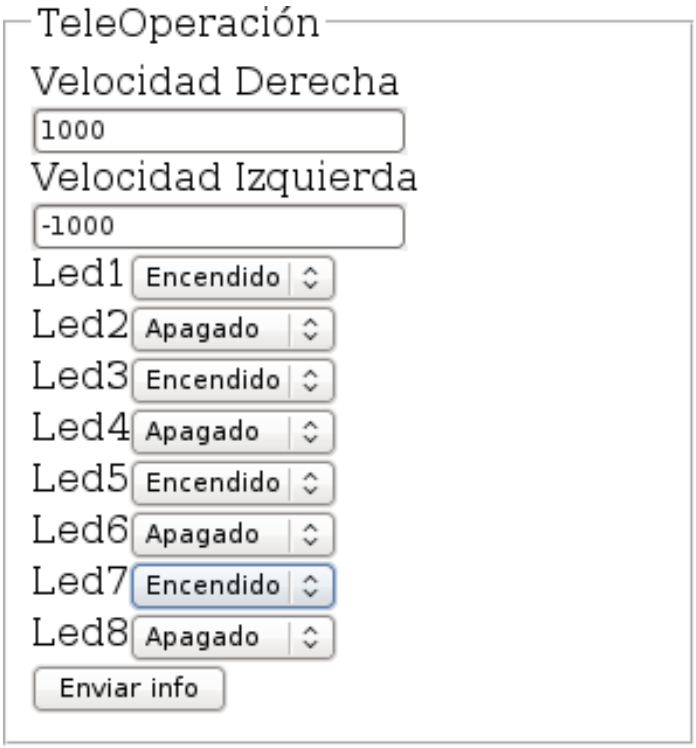
Figura 38. Ejemplo teleoperación. Deteniendo el robot.



En el caso que se desee que el robot gire sobre su propio eje a una rápida velocidad y que además, encienda sus LED impares, se ingresa la siguiente información en el cuadro de diálogo:

- Velocidad derecha = 1000,
- Velocidad izquierda = -1000,
- LED 1 = seleccionar “Encender”,
- LED 3 = seleccionar “Encender”,
- LED 5 = seleccionar “Encender”,
- LED 7 = seleccionar “Encender”,

Figura 39. Ejemplo teleoperación. Rotación sobre su propio eje y encendido de algunos LED.



Inmediatamente el robot encenderá los LED impares y comenzará a girar a su máxima velocidad sobre su propio eje (figura 40).

Figura 40. Ejemplo teleoperación. Secuencia rotación sobre su propio eje y manipulación de LED.

a.

b.

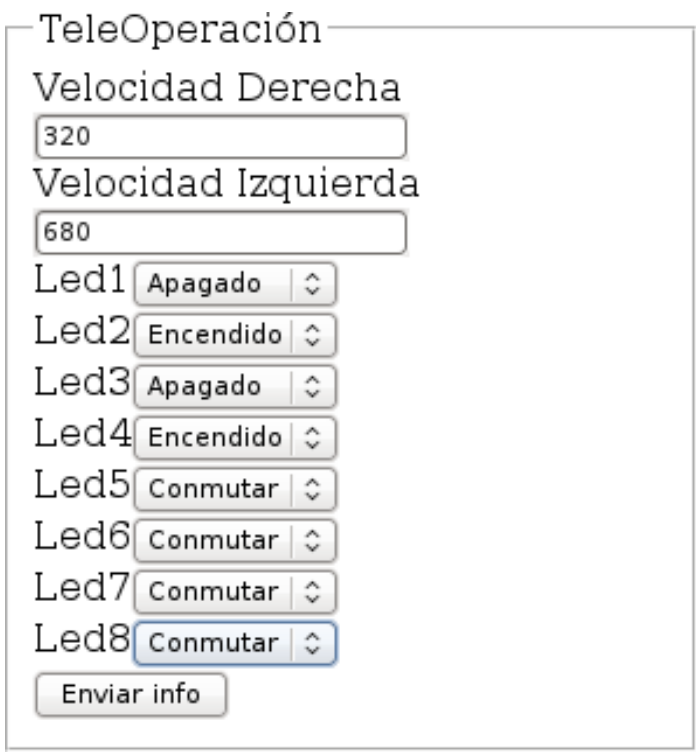
c.

Finalmente, si se desea comenzar un giro en una trayectoria de arco del robot, y apagar los LED impares pero encender los pares, este puede recibir la siguiente orden (figura 41):

- Velocidad derecha = 320,

- Velocidad izquierda = 680,
- LED 1 = apagar,
- LED 2 = encendido,
- LED 3 = apagar,
- LED 4 = encender,
- LED 5 = conmutar;
- LED 6 = conmutar;
- LED 7 = conmutar;
- LED 8 = conmutar;

Figura 41. Ejemplo teleoperación. Movimiento aleatorio y conmutación de LED.



Con la orden enviada, al presionar el botón “Enviar Info”, el robot comenzará su acción de movimiento en arco, apagará los LED impares y encenderá los pares (figura 42).

Nótese que para los LED a partir del 5°, se usa la acción de conmutar, por lo que la acción que se está realizando en los LED es una continuación de la establecida en la rotación sobre su propio eje, de otro modo, el resultado puede llegar a ser diferente.

Figura 42. Ejemplo teleoperación. Secuencia giro en un arco y manipulación de LED.

a.

b.

c.

3.5.2 Ejercicio de cinemáticas. Para iniciar el ejercicio de cinemáticas, lo mejor es detener por completo el robot. De este modo, con el robot detenido, se selecciona la rutina de cinemáticas (figura 43) y se reemplazará el diálogo de **teleoperación** por el de **Seguimiento a un punto** (figura 44).

Figura 43. Menú selección ejercicios. Enfocando a "cinemáticas".

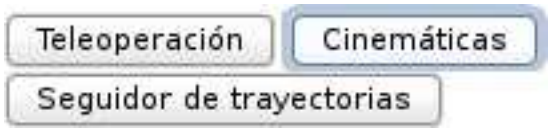
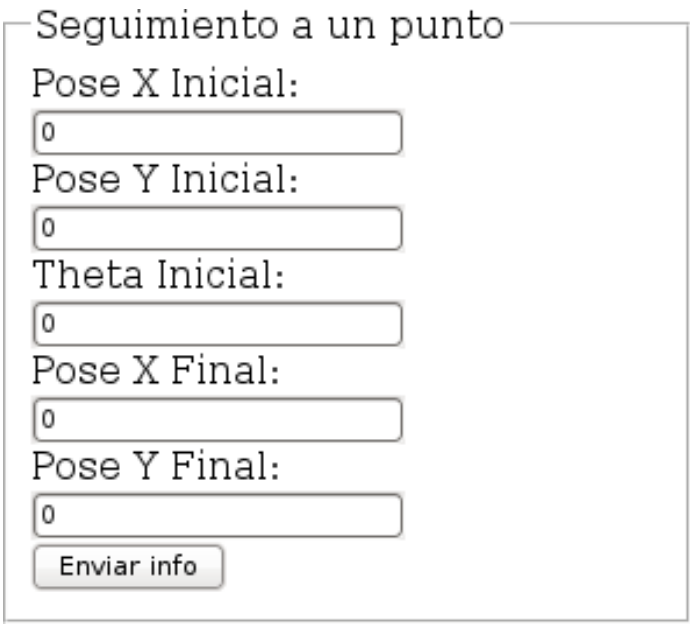
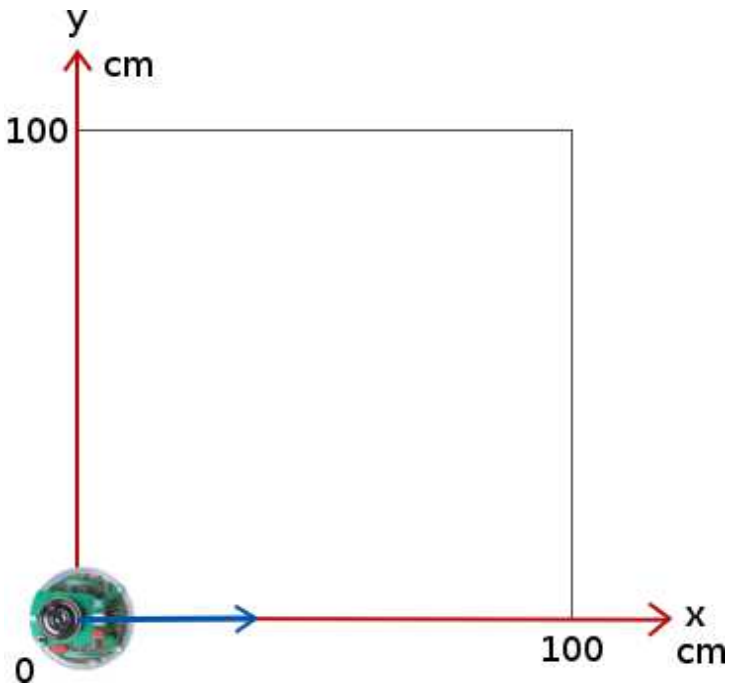


Figura 44. Ejercicio cinemáticas. Panel inicial.



Las dimensiones máximas que se deben ingresar, no deben ser mayores a los 250 cm que tiene cada lado del espacio de trabajo. Siendo así, el modo en que el sistema interpreta las dimensiones ingresadas se muestra en la figura 45.

Figura 45. Referencia ejes coordinados del espacio de trabajo



El movimiento más sencillo posible es el desplazamiento en un solo eje, por lo tanto, si se tiene el robot en el origen del espacio de trabajo y se desea llevarlo a la posición 100 en el eje *x*; se ingresan los siguientes datos (figura 46):

- Pose X inicial = 0,
- Pose Y inicial = 0,
- Theta inicial = 0,
- Pose X final = 75,
- Pose Y final = 0.

Figura 46. Ejercicio de cinemáticas. Movimiento en un solo eje.

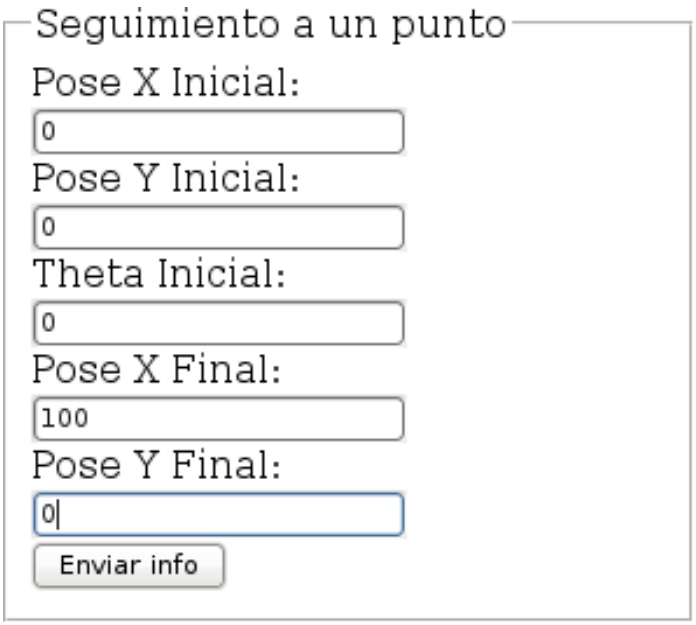
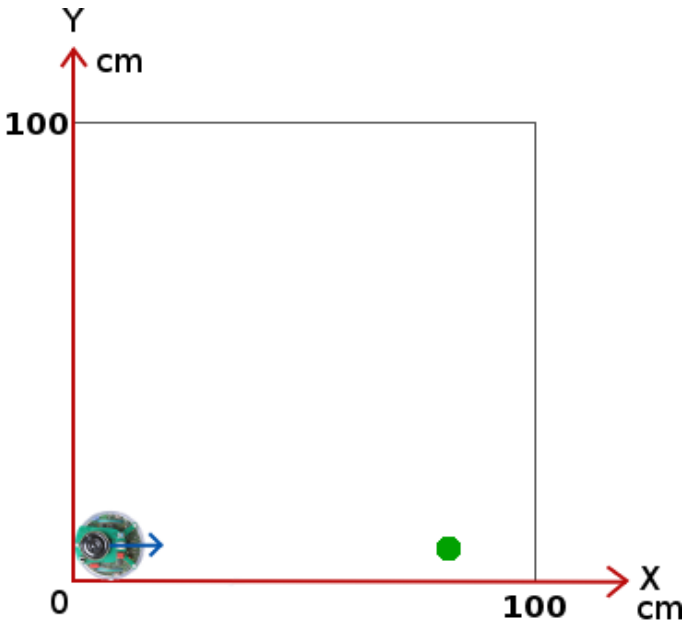
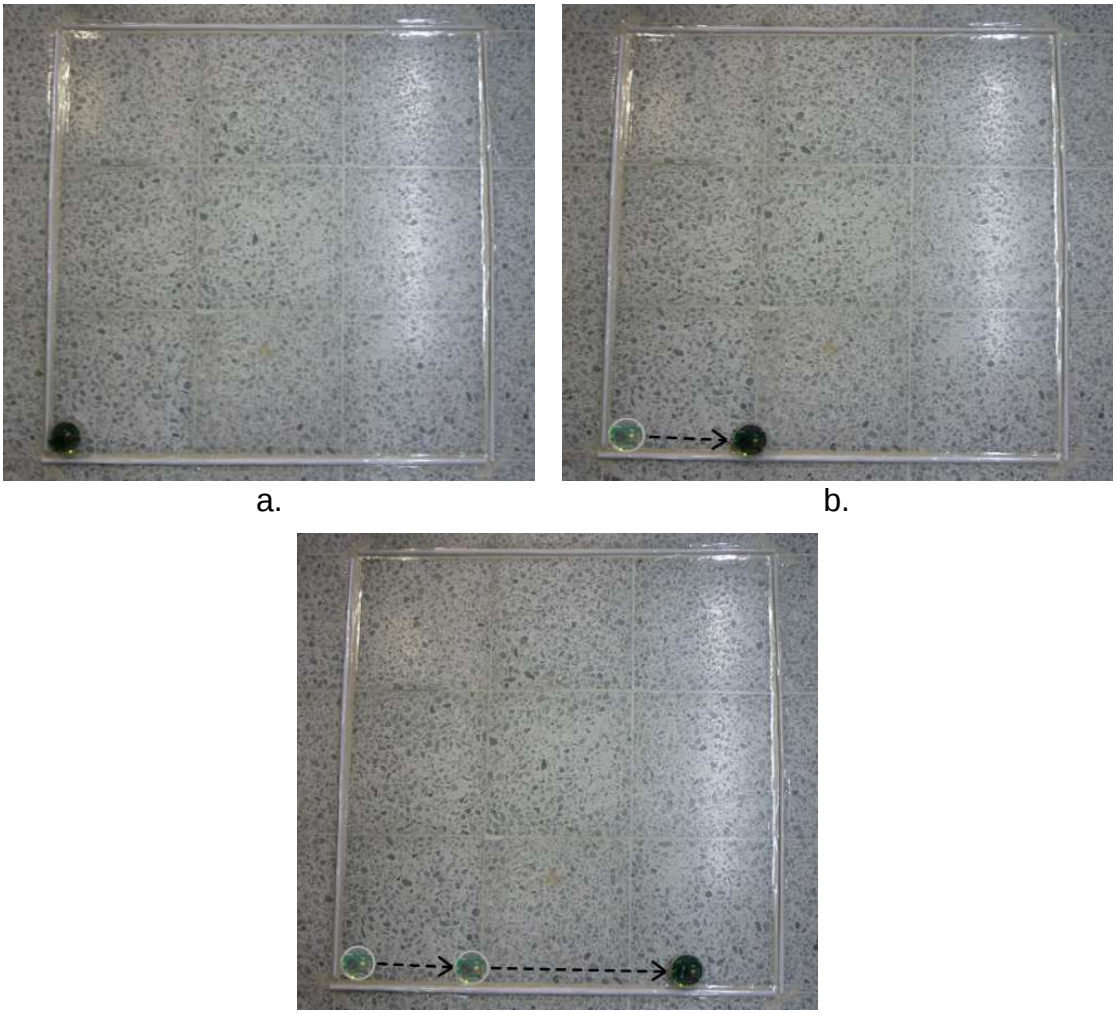


Figura 47. Ejercicio cinemáticas. Representación gráfica. el robot representa las condiciones iniciales y el punto verde las coordenadas objetivo.



Estas rutinas están programadas con una velocidad predeterminada que no permite ser modificada por el usuario. El robot comenzará a avanzar en línea recta hasta la posición *100i + 0j* siguiendo una trayectoria que se puede observar en la secuencia de la figura 48.

Figura 48. Ejercicio cinemáticas. Secuencia desplazamiento sobre el eje de origen.



a. b.

c.

Considerando, esta vez, que el robot se encuentra nuevamente en el origen del espacio de trabajo y se desea que se dirija al centro de este, los campos se han de llenar de la siguiente manera (figura 49):

- Pose x inicial = 0,
- Pose y inicial = 0,
- Theta inicial = 0,
- Pose x final = 50,
- Pose y final = 50.

Figura 49. Ejercicio cinemáticas. Movimiento hacia el centro del espacio de trabajo.

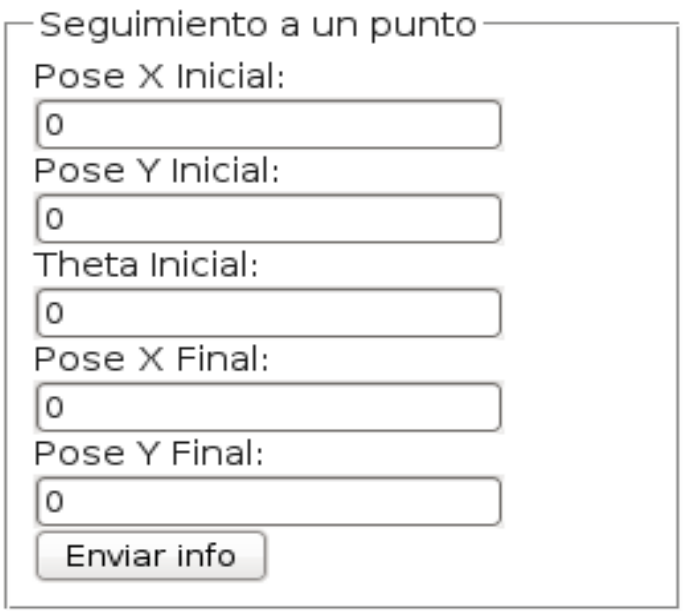
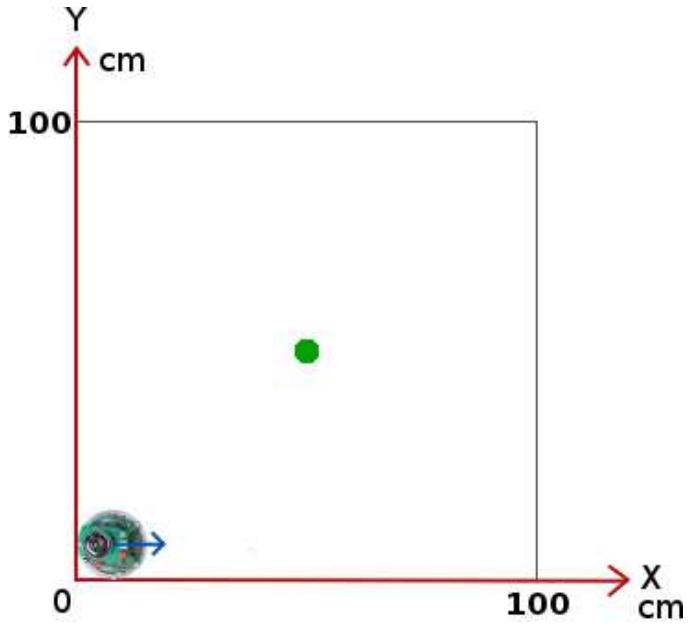


Figura 50. Ejercicio cinemáticas. Movimiento hacia el centro del espacio de trabajo. Descripción gráfica. El robot representa las condiciones iniciales y el punto verde el objetivo de desplazamiento.

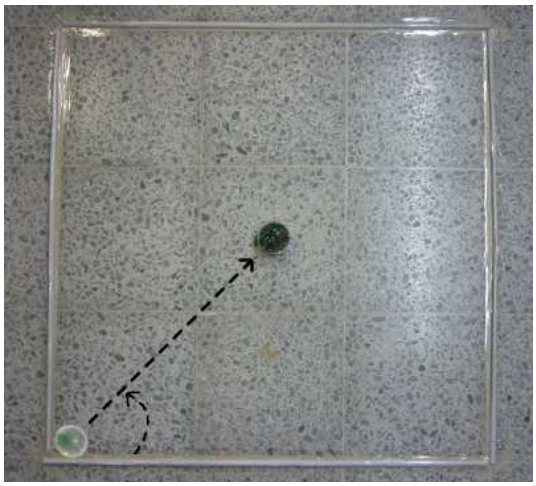


Una vez se presiona sobre “Enviar info”, el robot comenzará a rotar 45° y posteriormente a avanzar en línea recta hasta la posición *50i+50j*. Una vez termina su rutina, el robot estará en el centro del espacio de trabajo y con su pose angular en *45*°.La rotación y el desplazamiento se pueden apreciar en la secuencia de capturas de la figura 51.

Figura 51. Ejercicio cinemáticas. Secuencia avance al centro del espacio de trabajo.

a.

b.



c.

Con este ejercicio, es posible deducir el modo en que opera el algoritmo: Primero, realiza el giro con el que apuntará hasta donde se encuentra el destino final, para luego iniciar su avance hasta las coordinadas objetivo.

Como un último ejercicio, asumiendo que continúa de la posición anterior (*50i +50j y 45*°) y se desea que se dirija a un punto cualquiera del mapa, por ejemplo *10i+ 80j*. Entones la información que se va a proporcionar es (figura 52):

- Pose X inicial = 50,

- Pose Y inicial = 50,
- Theta inicial= 45,
- Pose X final = 10.
- Pose Y final = 80.

Figura 52. Ejercicio cinemáticas. Movimiento a un punto cualquiera.

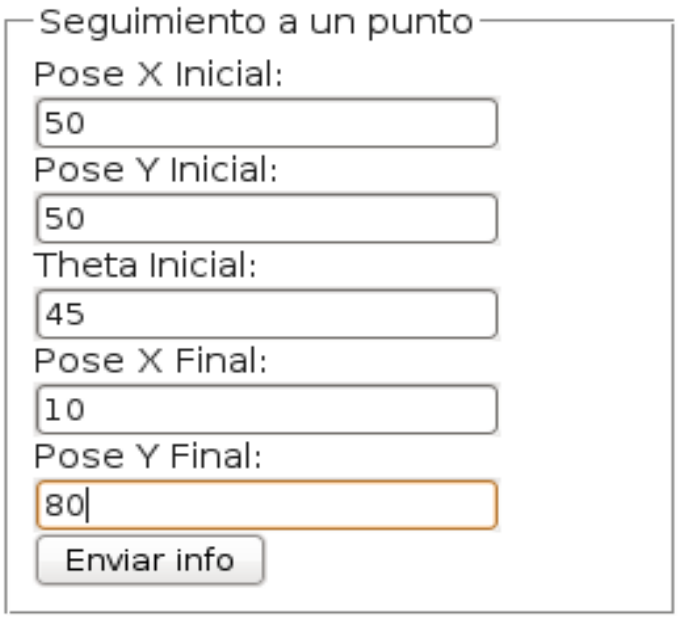
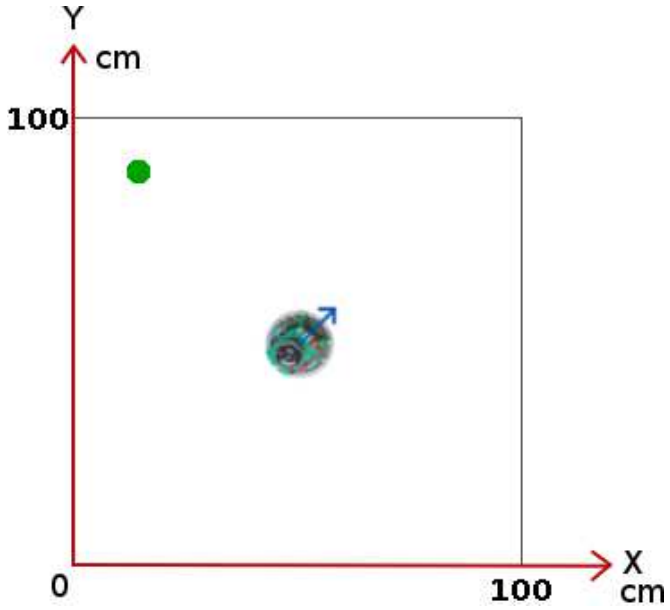


Figura 53. Ejercicio cinemáticas. Movimiento a un punto cualquiera. Descripción gráfica. El robot representa las condiciones iniciales y el punto verde el objetivo de desplazamiento.



Al igual que en el ejercicio anterior el robot iniciará un giro con la cantidad angular que necesita para poder iniciar su desplazamiento hasta la posición deseada. La trayectoria que sigue el robot durante este ejercicio se observa en la figura 54.

Figura 54. Ejercicio cinemáticas. Desplazamiento a una posición aleatoria desde una pose en el centro del espacio de trabajo.

a. b.

Figura 54. Continuación.

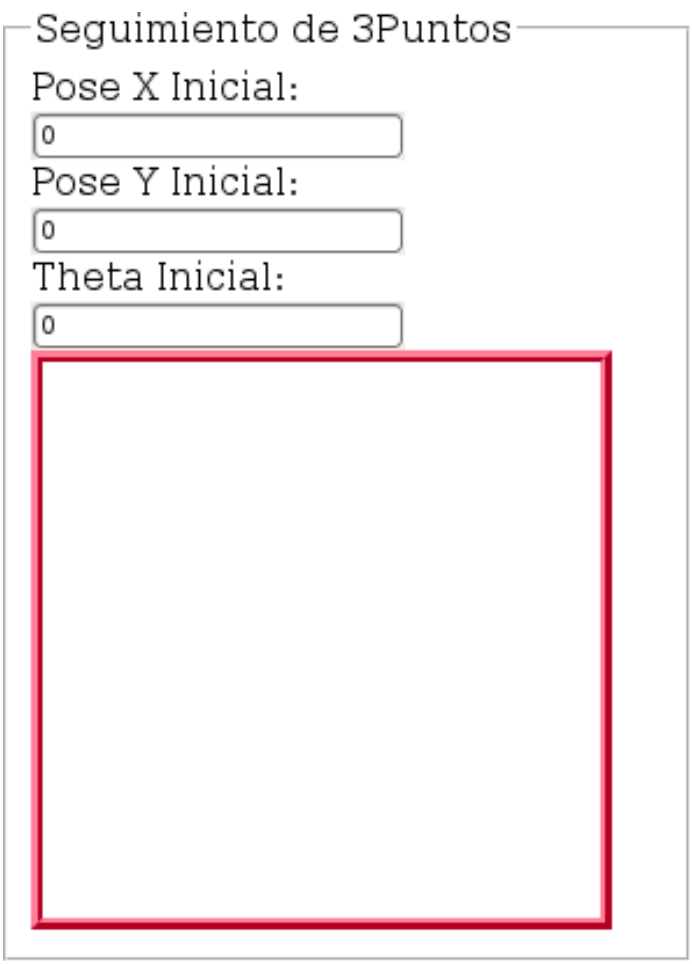
c.

Algo importante de mencionar de estos ejercicios de cinemática es el hecho que las poses iniciales del robot deben ser ingresadas correctamente para que los ejercicios se desarrollen satisfactoriamente.

3.5.3 Ejercicio de ruta de tres puntos. Para comenzar con el ejercicio de trayectoria de 3 puntos se recomienda haber detenido completamente el robot primero. Luego se selecciona “Seguidor de trayectorias” como se muestra en la figura 55.

Figura 55. Menú selección de ejercicios, enfocando a "Seguidor de trayectorias".

Figura 56. Ejercicio ruta de tres puntos. Panel inicial.



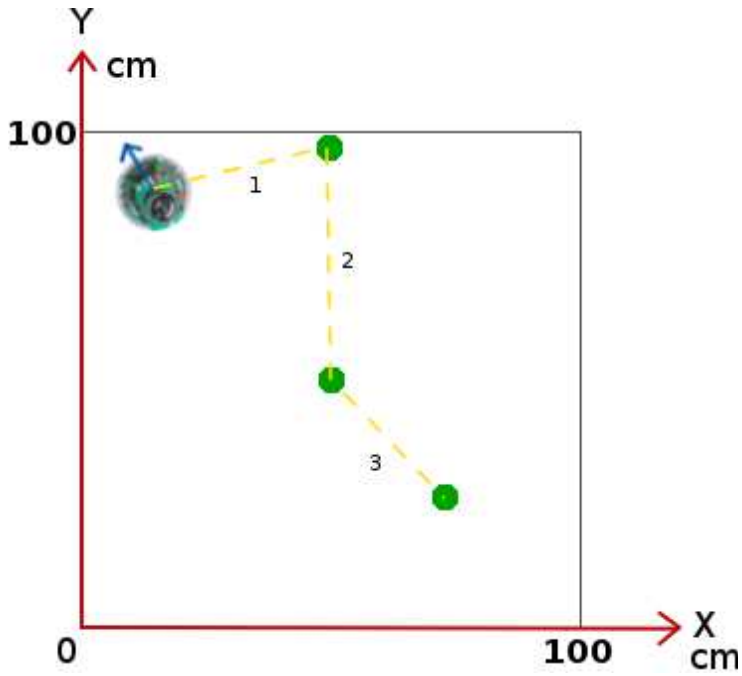
El entorno de este ejercicio consta de dos partes (figura 56), la superior donde se ingresan los datos iniciales del robot, su posición coordenada y angular; y un recuadro rojo donde se especifican los puntos a los que se desea que el robot se dirija.

El recuadro rojo posee la misma orientación y las mismas dimensiones de la figura 45, Por lo que los bordes representan los mismos bordes del espacio de trabajo del robot móvil.

Si se tiene el robot en la posición *10i + 80j* con un ángulo de *120°* y se desea establecer una trayectoria que vaya al centro del borde superior, luego se dirija al centro del espacio de trabajo y, finalmente, al punto medio entre el centro y la

esquina inferior derecha, se procede con los pasos como se describe a continuación:

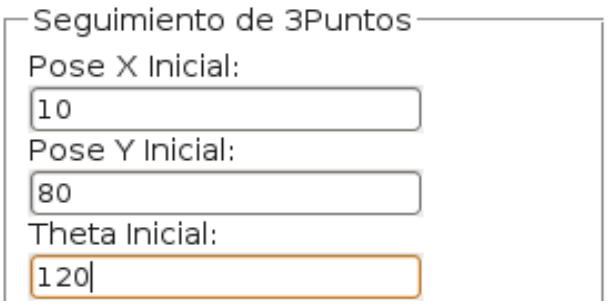
Figura 57. Ejercicio ruta de tres puntos. Descripción gráfica de la trayectoria.



El formulario inicial se llena con la siguiente información (figura 58):

- Pose X inicial = 10,
- Pose Y inicial = 80,
- Theta inicial = 120.

Figura 58. Ejercicio ruta de tres puntos. las condiciones iniciales.



Con el ratón se pulsa sobre la parte central superior del recuadro y aparecerá un punto encima del puntero (figura 59).

Figura 59. Ejercicio ruta de tres puntos. Selección del primer tramo.

Con la primera ruta establecida, para luego dirigirlo hacia el centro, se repite el paso anterior, pero esta vez con un clic sobre el centro del mapa (figura 60). Finalmente para que el robot termine su recorrido en el punto medio entre el centro y la esquina inferior derecha, con el ratón se pulsa sobre este punto deseado (figura 61).

Figura 60. Ejercicio tramo de tres puntos. Estableciendo la segunda trayectoria.

Figura 61. Ejercicio ruta de tres puntos. Último tramo.

Una vez los tres puntos de la trayectoria han sido seleccionados, aparece un cuadro de confirmación como el de la figura 62, mostrando la posición de cada uno de los tres puntos seleccionados, En este caso los tres puntos escogidos fueron:

- 1: *45i + 94j*,
- 2: *46i + 52j*,
- 3: *75i + 27j*.

Figura 62: Ejercicio trayecto de tres puntos. Dialogo de verificación.

Escogiendo el botón “Ok” el robot comienza su trayecto como se observa en las secuencias de las figuras 63 a 65:

Figura 63: Ejercicio de trayectorias. Secuencia inicio del desplazamiento.

a.

b.

Figura 63. Continuación.

c.

Figura 64: Ejercicio trayectorias. Secuencia segunda parte del recorrido.

a.

b.

Figura 65: Ejercicio trayectoria. Secuencia última parte del ejercicio.

a.

b.

4.CONCLUSIONES

- El desarrollo de nuevas plataformas como alternativas a las ya implementadas, brinda un mayor espectro de posibilidades al cual mirar en el caso que se deseen implementar este tipo de sistemas remotos.

- La educación remota facilita el desarrollo de los proyectos estudiantiles y permite que su desarrollo sea más ágil al ser posible que el acceso se ejecute desde cualquier parte con conexión a internet.

- El prototipo de teleoperación puede brindar un soporte a los educadores en su tarea de ilustrar ejemplos reales sobre los temas vistos en clase.

- Las herramientas educativas remotas brindan una mayor cobertura de educación a los jóvenes e investigadores que deseen desarrollar sus estudios en los campos de la robótica móvil.

- La evolución de las tecnologías web permiten el desarrollo de herramientas de alto nivel, que pueden llegar a brindar la experiencia, incluso, de estar en el lugar de la práctica.

- El uso de diversas capas en la programación aumenta la modularidad del sistema, por lo que es posible adaptar el diseño aquí presentado, a otras plantas del laboratorio u otros campos de la educación.

- Esta herramienta permite que incluso equipos con poca capacidad de cómputo tengan acceso al laboratorio sin sacrificar mucho el desempeño de las distintas actividades.

5.TRABAJOS FUTUROS

- Es posible migrar el trabajo realizado a una plataforma de desarrollo de web, lo cual le daría robustez y orden a la plataforma.

- Adaptar los diferentes códigos para que permitan el desarrollo de rutinas propias por parte de los estudiantes, docentes e investigadores.

- Adaptar las API's de la plataforma para que soporten más plantas del laboratorio u otros equipos.

- Modificar el modo que se transmite el video por medio de internet, de modo que sea procesado directamente desde el servidor y no de un tercero.

BIBLIOGRAFÍA

HUANG, Albert y RUDOLPH, Larry. Bluetooth Essentials for Programmers. Nueva York, Cambridge University Press, 2007. 198 p.

What are the features of 'C' Language [en línea]. Nueva York [Consultado en Oct 2010]. Disponible en internet: http://wiki.answers.com/Q/What_are_the_features_fo_'C'_language

BRANWYN, Gareth, Absolute Beguinner's Guide to Building Robots. Estados Unidos, Que Publishing, 2003. 384 p.

CAÑAS, José María. Jerarquía dinámica de esquemas para la generación de comportamiento autónomo. Doctor en telecomunicaciones. Madrid. Universidad Politécnica de Madrid. 2003.

CHOSET, Howie; LYNCH, Kevin; SETH, Hutchinson; KANTOR, George; BURGARD, Wolfram; KAVRAKI Lydia y THRUN, Sebastian. Principles of robot motion. Theory, algorithms and implementation. Londres, MIT Press, 2005. 603 p.

Cross compiling for dsPic [en línea] IRIDIA [Consultado Nov 2010]. Disponible en Internet: http://iridia.ulb.ac.be/~e-puck/wiki/tiki-index.?page=Cross+compiling+for+dsPic

Debian Leny installing Apache2 and PHP5 [en línea]. Slicehost [Consultado Nov 2010]. Disponible en Internet: http://articles.slicehost.com/2009/4/9/debian-lenny-installing-apache2-and-php5

Download epucklib [en línea]. Ecole Polytechnique Fédérale de Lausanne [Consultado Nov 2010]. Disponible en línea: http://www.e-puck.org/index.php?option=com_phocadownload&view=file&id=43:epuck-library-compiled&Itemid=38

GIBILISCO, Stan. Concise Encyclopedia of Robotics. Estados unidos, McGraw Hill, 2003. 365 p.

Hardware - Spykee developer wiki [en línea]. Sypkee wiki [Consultado Nov 2010]. Disponible en Internet: http://spykee.duskofsolace.com/index.php?title=Hardware

HORTON, Ivor, Beginning Java 2. JDK. 5 edición. Indianapolis, Wiley Publishing, 2005. 1470 p.

HTML 5 Introduction [en línea]. Refsnes Data [Consultado Oct 2010]. Disponible en línea: http://w3schools.com/html5/html5_intro.asp

HTML & CSS. [en línea]. World Wide Web Consortium [Consultado Oct 2010] Disponible en línea: http://www.w3.org/standards/webdesign/htmlcss

Java Features [en línea]. Rose India technologies Ltd [Consultado Nov 2010].Disponible en Internet: http://www.roseindia.net/java/java-introduction/java-features.shtml

JONES, Joseph L; FLYNN, Anita y SEIGER, Bruce. Mobile Robots. Inspiration to implementation. 2 ed. Estados Unidos, A K Peter, 1998. 486 p.

LAM, Hoang y THAI, Thuan L. NET Framework Essentials. 3 ed. Estados Unidos, O'Reilly, 2003. 284p.

NARAMORE, Elizabeth; GERNER, Jason y SCOUARNEC, Yann Le. Beginning PHP5, Apache and MySQL Web Development. Indianápolis, Wiley Publishing, 2005. 798 p.

PFAFFENBERGER, Brian, SCHAFER, Steven; et al. HTML, XHTML and CSS Bible. 3 ed. Indianápolis, Wiley Publishing, 2004. 790 p.

Robotis [en línea]. Robotis Inc [Consultado Oct 2010]. Disponible en Internet: http://www.robotis.com/xe/bioloid_en

Radish: The Robotics Data Set Repository [en línea]. HOWARD, Andrew. ROY, Nick [Consultado Ago 2010]. Disponible en Internet: http://radish.sourceforge.net

SCHILDT, Herbert. C/C++ Programmer's Reference. 3 ed. California, McGraw Hill, 2003. 358 p.

Scripting and ajax [en línea]. World Wide Web Consortium [Consultado Oct 2010] Disponible en línea: http://www.w3.org/standards/webdesign/script

SIEGWART, Roland y NOURBAKSHS, Illah. Introduction to autonomous mobile robots. Londres, MIT Press, 2004. 321 p.

Spykee Self assembly [en línea], Spykee.org The 100% unofficial Spykee site [Consultado en Oct 2010]. Disponible en Internet: http://www.spykee.org/AboutSpykee/Selfassembly/tabid/234/Default.aspx

Spykee. The spy robot [en línea]. Mecano [Consultado Oct 2010]. Disponible en Internet: http://www.spykeeworld.com/spykee/UK/index.html

Visual Studio Home [en línea], Microsoft Corporation [Consultado Oct 2010]. Disponible en Internet: http://www.microsoft.com/visualstudio/en-us/

WELSH, Matt; et al. Running Linux. 4 ed. California, O'Reilly, 2002. 692p.

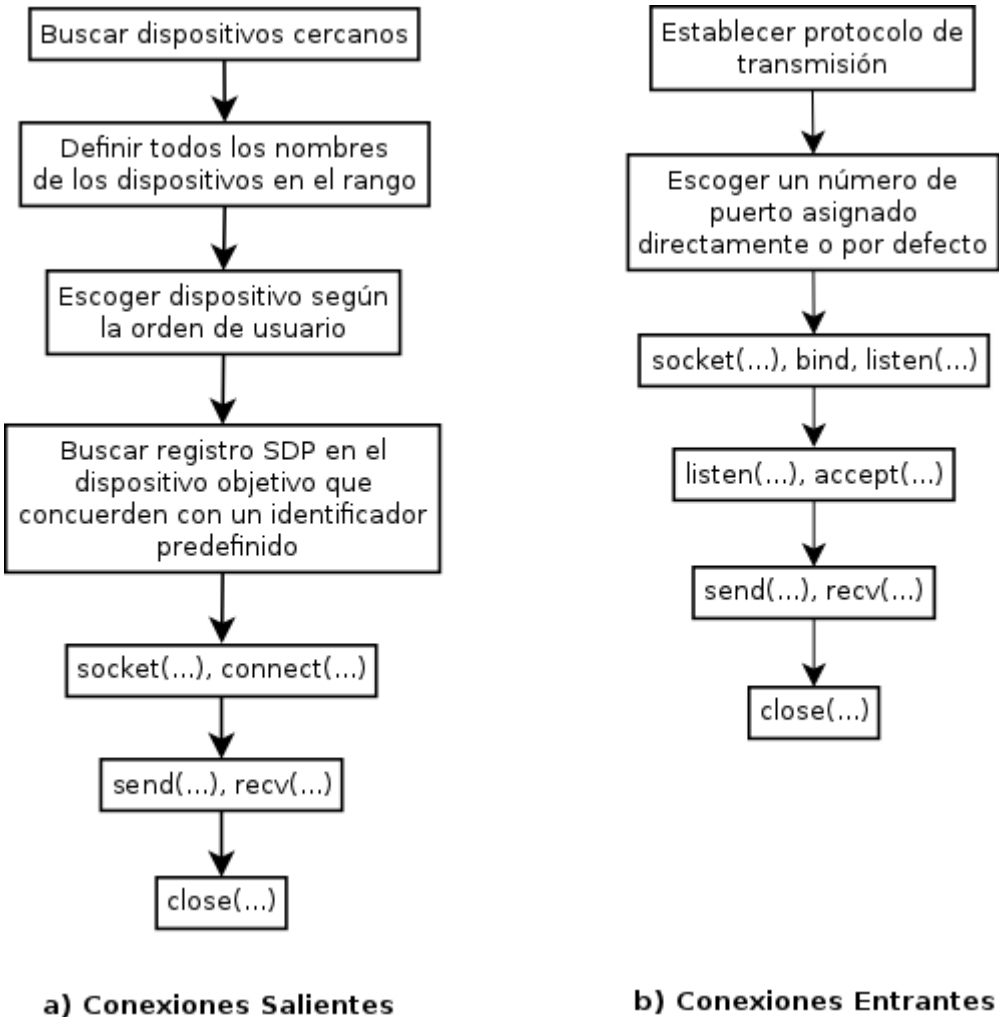
ANEXOS

ANEXO A: PROCESO DEL DISEÑO concurrente aplicado en el diseño mecatrónico y en el desarrollo de este proyecto.

Figura 66: Proceso del diseño concurrente aplicado en el diseño mecatrónico y en el desarrollo de este proyecto.

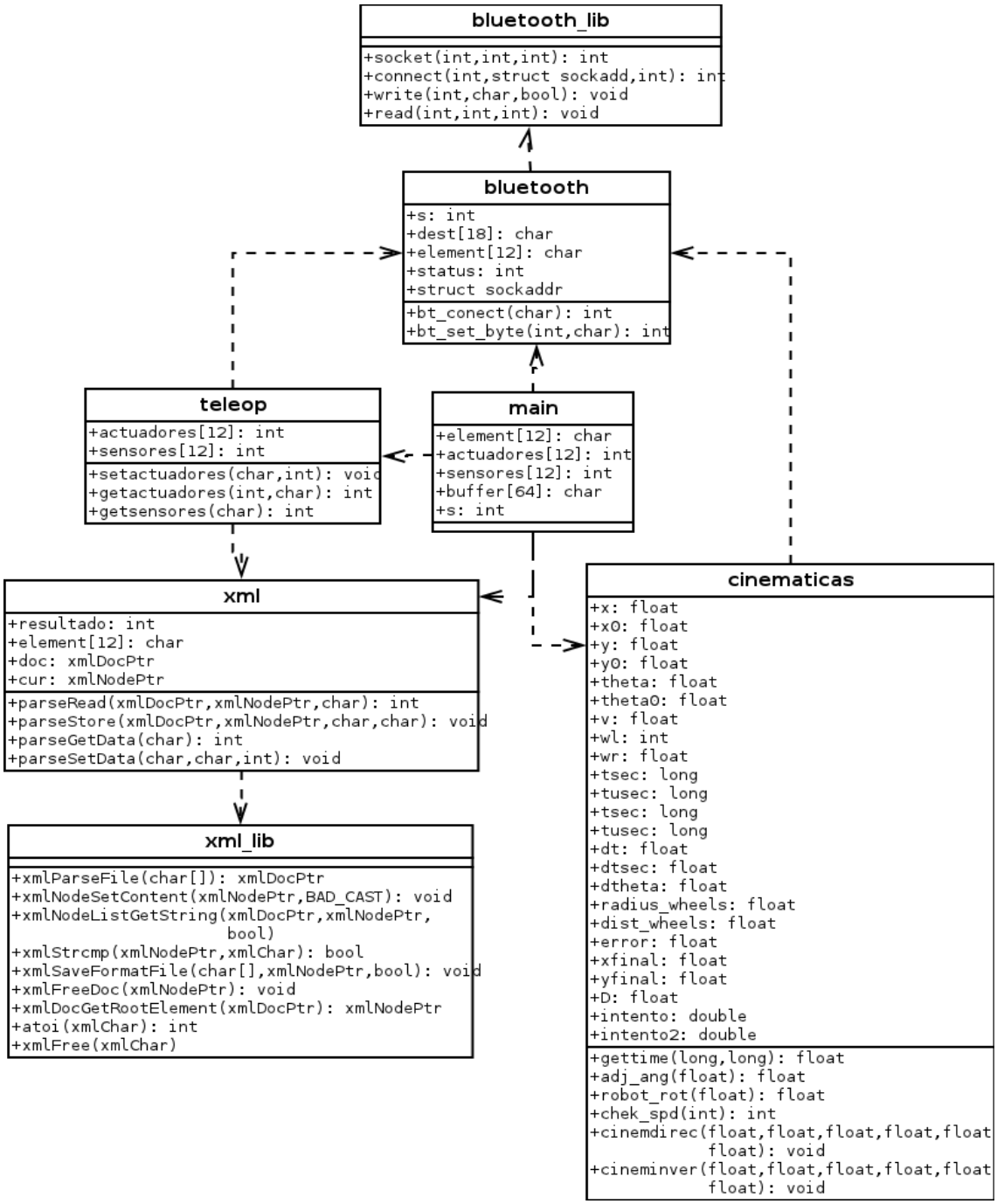
ANEXO B: ETAPAS DEL PROCESO de conexión entre dos dispositivos con conectividad bluetooth.

Figura 67: Etapas del proceso de conexión entre dos dispositivos con conectividad bluetooth.



ANEXO C: DIAGRAMA UML para describir el funcionamiento del programa usuasio- robot del servidor.

Figura 68: Diagrama UML para describir el funcionamiento del programa usuario - robot del servidor.



ANEXO D: ÁRBOLES XML de los archivos

Figura 69: Arbol XML del archivo sensores.xml

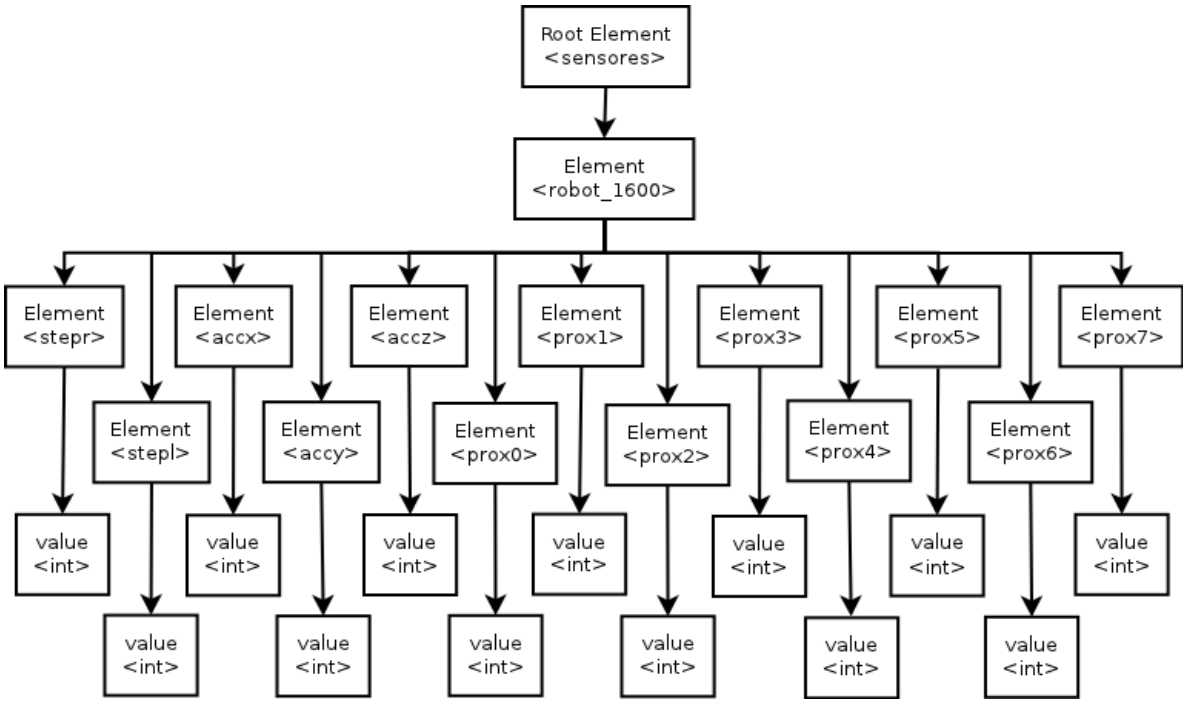
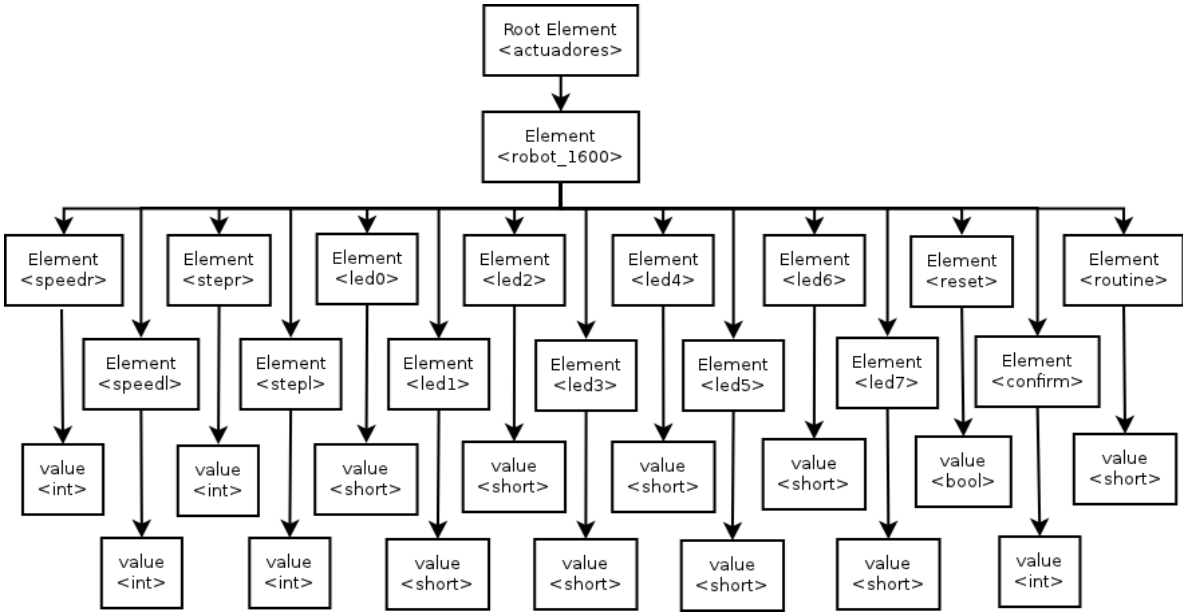


Figura 70: Árbol xml del archivo actuadores.xml



ANEXO E: DIAGRAMA DE SECUENCIA de la rutina del e-puck

Figura 71: Diagrama de secuencia de la rutina del e-puck.

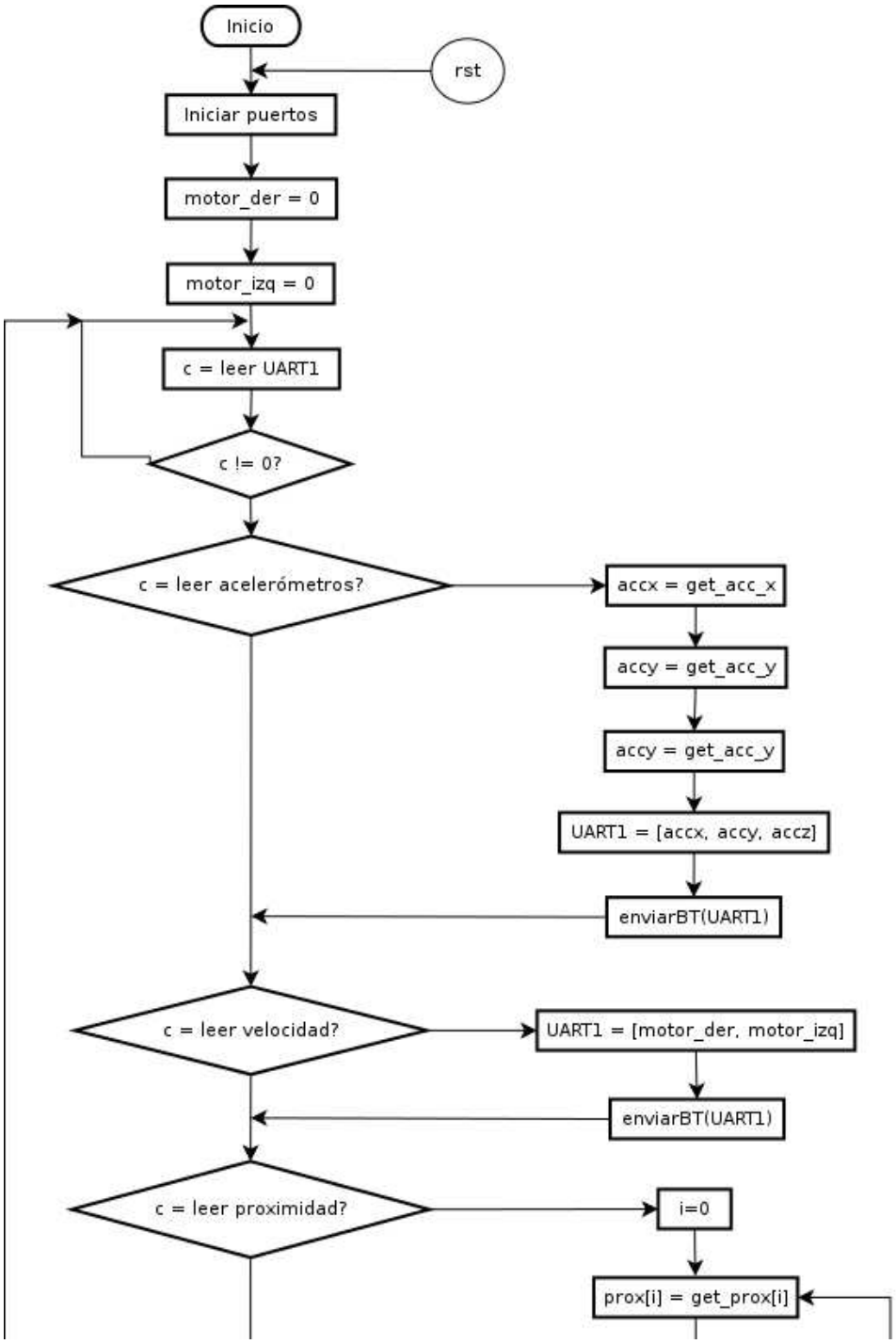
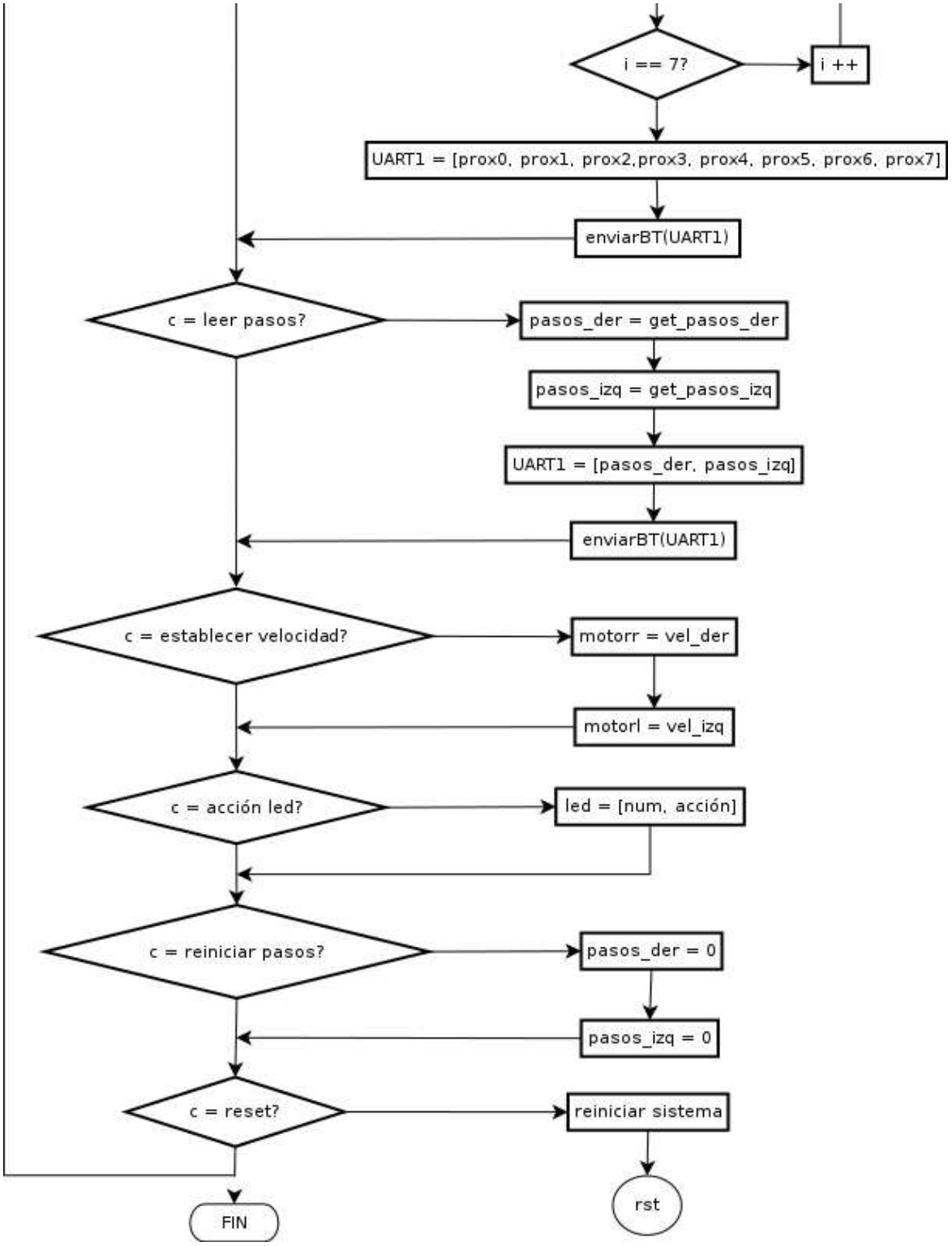


Figura 71: Continuación



ANEXO F: DESCRIPCIÓN de los algoritmos de robótica móvil

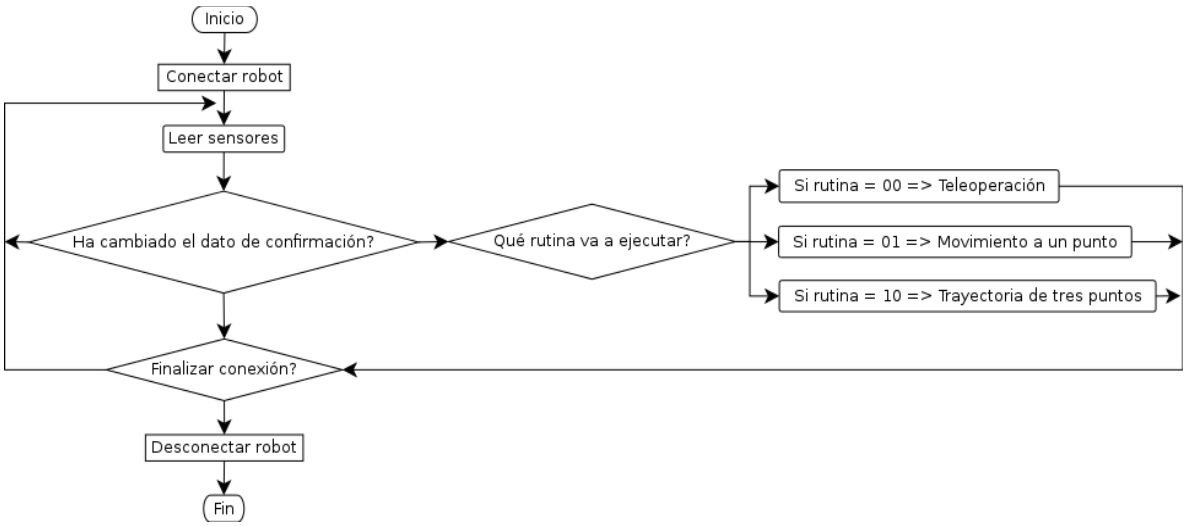
El programa comienza inicializando las variables y la configuración del robot y su conexión para poder conectarse con este. A partir de este paso, comienza la lectura de cada uno de los sensores del robot enviándole una cadena de caracteres que indica la información que espera recibir como muestra el cuadro 8.

Cuadro 8: Significado de las órdenes enviadas al robot e-puck por el programa usuario-robot

Palabra enviada	Orden solicitada
A	Lectura de los acelerómetros
D	Lectura de los pasos de motores
C	Lectura de los sensores de proximidad

A continuación realiza la lectura del archivo actuadores.xml verificando que este haya sido modificado desde su última lectura por medio de la variable *confirm*, la cual se genera automáticamente por el servidor tomando el valor presente en el contador de reloj interno del servidor y asignándolo a esta variable cada que sobre escribe el archivo con las nuevas órdenes que se deben enviar al robot, siendo así único e irrepetible. Si el archivo ha sido modificado, dependiendo del valor de la variable *routine,* se dirige a la rutina a ejecutar: teleoperación, cinemáticas a un punto, cinemáticas en trayectoria de tres puntos (figura 72).

Figura 72. Diagrama de flujo del programa PC - e-puck



A.1 TELEOPERACIÓN

Cuando el programa usuario-robot comienza a ejecutar la rutina de teleoperación, lee el contenido del archivo “actuadores.xml” para encontrar los valores correspondientes a las variables de velocidad en las ruedas y actividad en los LED según se muestra en el cuadro 9.

Cuadro 9. Valores aceptados por el algoritmo de teleoperación.

Variable	**Valor**	**Acción**
speedr	[0 - 1000]	Rango de velocidad derecha.
speedl	[0 - 1000]	Rango de velocidad izquierda.
Led[0,1...7]	[0, 1, 2]	[apagado, encendido, conmutar] LED

A continuación encapsula cada uno de estos valores en una cadena de caracteres con la información que permitirá que el robot interprete correctamente las órdenes(cuadro 10).

Cuadro 10. Órdenes de teleoperación que van desde el programa "usuario - robot" hasta el robot e-puck.

Variable	Orden	Caracteres que recibe el robot
speedr, speedl	Velocidad derecha e izquierda	E,speedl,speedr
led0	Estado Led0	F,0,led0
led1	Estado Led1	F,0,led1
led2	Estado Led2	F,0,led2
led3	Estado Led3	F,0,led3
led4	Estado Led4	F,0,led4
led5	Estado Led5	F,0,led5
led6	Estado Led6	F,0,led6
led7	Estado Led7	F,0,led7

El robot recibe un rango de velocidades que van entre 0 y 1000 que equivalen al número de revoluciones de cada una de las ruedas del robot y las órdenes de cada LED pueden ser 0, 1 o 2 equivalentes a los estados de apagado, encendido y conmutar respectivamente.

A.2 CINEMÁTICAS DIRECTA E INVERSA

Las rutinas de “desplazamiento a un punto” y “trayectoria de 3 puntos” hacen uso de los mismos algoritmos cinemáticos, con la diferencia que el ejercicio de trayectoria repite el ciclo tres veces para llegar a los tres puntos requeridos.

A.2.1 Procesamiento de los datos.

El robot e-puck posee ciertas características en su locomoción:

- Relación de la caja reductora = 50:1
- Diámetro de las ruedas = 41 mm
- Distancia entre ruedas (L) = 53 mm

- 20 pasos/revolución

- Velocidad máxima de los motores de cada rueda = 1000 pasos/s

Debido a la configuración de las ruedas cada mil revoluciones equivalen a un giro completo para cada una de ellas, por ello el programa debe procesar primero la información que recibe de pasos al avance correspondiente para poder realizar cualquier ejercicio de cinemática u otra rutina.

El dato de pose angular es solicitado al usuario en grados por su facilidad de interpretar, sin embargo, para que puedan ser válidos en el programa, se deben convertir a radianes.

A.2.2 Rotación en las cinemáticas.

Para establecer la distancia angular (D) se deben tener algunas consideraciones que darán realmente con el total:

- Si $\Delta X = 0$ y $\Delta Y > 0$, entonces $D = \pi/2$.

- Si $\Delta X = 0$ y $\Delta Y < 0$, entonces $D = 3\pi/2$.

- Si $\Delta Y = 0$ y $X_f > 0$, entonces D = 0.

- Si $\Delta Y = 0$ y $X_f > 0$, entonces $D = \pi$.

- Si $\Delta Y < 0$ entonces $D = atan(\frac{\Delta Y}{\Delta X}) + \pi$

- Si no cumple ninguna de las condiciones anteriores entonces $D = atan(\frac{\Delta Y}{\Delta X})$

Con los datos de pose final, inicial y el recorrido del ángulo necesario, se realiza el cálculo del tiempo que tardará el robot en rotar hasta ángulo final con la ecuación 8:

$$\Delta t = \frac{c}{L}(wr - wl)[(0.001)2\mathrm{v}](2\pi) \quad (8)$$

Una vez inicia el movimiento toma una medida de tiempo e inicia un ciclo en el cual vuelve a obtener el tiempo anterior para procesar un Δt y actualiza su ángulo constantemente hasta finalizar el recorrido aplicando la ecuación 9:

$$\theta = \theta_0 + \frac{c}{L}(wr - wl)\,0.001(2\pi)\Delta t \quad (9)$$

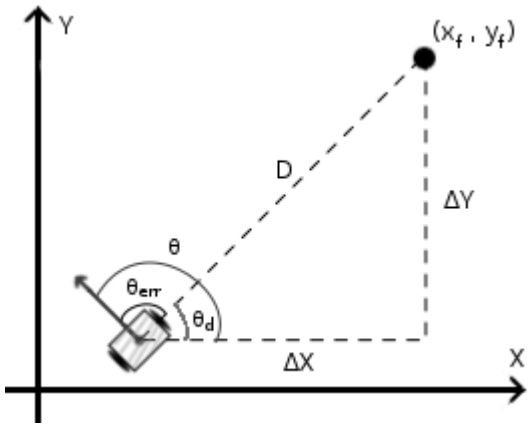
A.2.3 Avance en las cinemáticas.

Una vez finaliza la rotación detiene el robot y calcula el tiempo que le tomará llegar hasta la posición de destino haciendo los siguientes cálculos (ecuaciones 10 y 11:

$$D = \sqrt{\Delta x^2 + \Delta y^2} \quad (10)$$

$$t = \frac{D}{v\ c\ (0.001)\ 2\pi} \quad (11)$$

Figura 73: Representación del cálculo de las distancias para el cálculo de las cinemáticas



Del mismo modo que en el movimiento angular comienza su desplazamiento, capturando el instante de tiempo en que lo inicia, para entrar en un ciclo en el que vuelve a tomar el tiempo para obtener el `Δt` y actualiza su posición cartesiana en *X* y *Y* con las ecuaciones 12 y 13; hasta llegar a la pose solicitada, detener el movimiento y volver a iniciar el proceso de lectura de sensores y del archivo ”actuadores.xml”.

$$x = x_0 + \left(\frac{c}{2} (wr + wl)\, 0.001 (2\pi) \cos(\theta)\, \Delta t \right) \tag{12}$$

$$y = y_0 \left(\frac{c}{2} (wr + wl)\, 0.001 (2\pi) \sin(\theta)\, \Delta t \right) \tag{13}$$