Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Proyecto Terminal II

"Programación de Servicio a cubículos del Departamento de Sistemas por medio del Robot P3-AT"

Alumno:
Alonso Romei Federico Emmanuel
Matrícula: 205200356

Trimestre: 11O

Asesora:
Dra. Silvia Beatriz González Brambila

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El objetivo de este proyecto consistió en configurar y programar el Robot P3-AT para que de acuerdo a una ruta previamente establecida vaya a los diferentes cubículos del Departamento de Sistemas de la Universidad Autónoma Metropolitana Azcapotzalco (UAM Azcapotzalco).

Una vez establecido el objetivo, se identificaron y analizaron los componentes que hacen posible el desarrollo del proyecto:

- Robot P3-AT.
- Departamento de Sistemas.

El robot P3-AT, es una plataforma robótica desarrollada por la compañía *Mobile Robots* [1], su tarea es recorrer el Departamentos de Sistemas y visitar algunos de los cubículos que lo conforman. Para realizar el recorrido el robot hace uso de un juego de sonares para la detección de obstáculos, dos pares de ruedas para realizar acciones básicas (avanzar, girar, detenerse), tres baterías de 12V para su alimentación y un puerto serial para establecer la comunicación con el servidor de software.

El Departamento de Sistemas, es la estructura organizacional dedicada a la docencia e investigación, para la resolución de problemas que requieren un enfoque de sistemas. Es el sitio que debe de recorrer el robot P3-AT, se eligió porque abarca un solo piso y cuenta con pocos cubículos, todos ellos fáciles de identificar.

Definidos los componentes esenciales, se diseñó la aplicación que se encarga de indicarle al robot la ruta que debe seguir y los cubículos a visitar, el software lleva por nombre **PECRI** (**Programa Encargado de Controlar el Robot Inteligente**).

PECRI está conformado por tres módulos:

- ***Interfaz Gráfica:*** Permite al usuario seleccionar los cubículos que debe visitar el robot en su recorrido.
- ***Módulo de establecimiento de la ruta seleccionada:*** Establece el orden en que se visitan los cubículos. Los datos de visita son guardados en un archivo de texto para ser cargados en el módulo de ejecución del robot.
- ***Módulo de ejecución del robot:*** Es el módulo encargado de poner en marcha el robot, aquí se encuentran los comandos encargados del control. El tiempo del recorrido depende de la cantidad de cubículos seleccionados.

Se decidió que la *interfaz gráfica* (GUI) sería un gráfico interactivo del Departamento de Sistemas, donde los cubículos jugarían el rol de elementos seleccionables. Cada cubículo es un elemento gráfico que necesita ser pulsado para ser activado. En el panel de control de usuario se incluyen las funciones básicas (inicio, Ayuda y Salir) accesibles a través de botones.

Para diseñar los *módulos de establecimiento de la ruta* y *ejecución del robot* fue necesario definir lo siguiente:

- El orden en que se recorren los pasillos del Departamento de Sistemas.
- El orden de visita de los cubículos.
- La rutina que debe de ejecutar el robot para visitar un cubículo.

Una vez que se define la información de los tres puntos anteriores, el paso final en el diseño fue identificar las estructuras de control (avanzar, girar y detener) y crear el algoritmo con el que funciona PECRI.

Terminado el análisis y el diseño, se procedió a instalar las herramientas necesarias para implementar, probar y simular PECRI. Antes de contar con la versión final del software, se hicieron experimentos con el fin de observar el funcionamiento del robot y obtener las instrucciones que serían útiles. Gracias a los resultados arrojados por los experimentos, se logró desarrollar la primera versión de PECRI (beta).

Ya con la versión beta desarrollada, se procedió a realizar las pruebas. Primero se simuló, para afinar algunos detalles de implementación y después, se probó directamente sobre el robot para identificar los ajustes que se hicieron sobre PECRI.

1 INTRODUCCIÓN

Dentro del Departamento de Sistemas todos los días, se realizan desplazamientos de un cubículo a otro con el ánimo de entregar algún material, llevar un documento u objetos, que normalmente son realizados por una persona. Este tipo de tarea es repetitiva y requiere de poco esfuerzo intelectual, por lo cual un robot es una herramienta útil para llevarla a cabo.

El desarrollo y las pruebas de este proyecto se llevaron a cabo en el Departamento de Sistemas, este sitio fue elegido debido a que abarca un solo piso y cuenta con pocos cubículos, todos ellos fáciles de identificar.

El recorrido que realiza el robot al Departamento de Sistemas debe presentar las siguientes características:

- No se plantea que el robot valide si el cubículo está ocupado o vacío.
- El algoritmo asume condiciones ideales, es decir, no debe existir ningún obstáculo en el recorrido del robot.
- Un cubículo se considera "visitado" sólo cuando el robot se posicione enfrente de la puerta.
- Este proyecto, inicialmente, se consideró que concluiría cuando el robot fuera capaz de visitar, de manera autónoma, cada uno de los cubículos seleccionados por el usuario.

El objetivo de este proyecto consiste en configurar y programar el Robot P3-AT para que de acuerdo a una ruta previamente establecida vaya a los diferentes cubículos del Departamento de Sistemas de la UAM-Azcapotzalco. Se utiliza el software proporcionado por el fabricante del robot (*Mobile Robots*) y su interfaz serial.

Para cumplir con el objetivo se creó una aplicación de nombre ***Programa Encargado de Controlar al Robot Inteligente*** (**PECRI**) que está compuesta de tres módulos. El primer módulo es la ***interfaz gráfica***, permite al usuario seleccionar los cubículos que debe de visitar el robot en su recorrido. El segundo, el ***módulo de establecimiento de la ruta seleccionada***, establece el orden en que se visitan los cubículos. Y por último, ***módulo de ejecución del robot*** se encarga de poner en marcha el robot.

El desarrollo del proyecto se llevó a cabo bajo el sistema operativo Microsoft Windows 7, el lenguaje de programación fué C++, para crear la interfaz gráfica se usó el *framework* Qt, al tratarse de un robot las aplicaciones requeridas son ARIA, *MobileSim* y *Mapper 3* todas ellas proporcionadas por el fabricante (*Mobile Robots*).

Dentro de la UAM podemos encontrar los siguientes proyectos relacionados:

- "*Puesta en marcha del Robot P3-AT*" de Ingeniería Electrónica, en el trimestre 08-O [2]. Este proyecto es la base para continuar con otros proyectos porque consistió en poner en marcha y configurar los movimientos básicos del Robot P3-AT, pero no se programó ninguna tarea ni ruta en específico, como en este caso.

- "*Programación de una visita guiada al Departamento de Sistemas por medio del Robot P3-AT*" de Ingeniería en Computación, finalizado en el trimestre 10-P [3], es un proyecto similar en cuanto a programación, sin embargo sólo hace un recorrido sin detenerse en un punto específico.

El reporte se estructura de la siguiente manera. En el capítulo 2, se describen los componentes esenciales de este proyecto. En el capítulo 3, se presenta la estructura lógica de la aplicación (PECRI), se define el recorrido del robot y orden de visita de los cubículos, además se presentan los algoritmos y estructuras de datos del programa. En el capítulo 4, en la sección 4.1 se encuentra el ***manual de instalación***, sección 4.2 el ***manual de uso***, sección 4.6 la ***documentación***, además se muestran los experimentos y pruebas que se realizaron antes de llegar a la aplicación final, junto con sus resultados En el capítulo 5 se presentan las conclusiones y por último en el capítulo 6 se tiene la bibliografía.

2 ANÁLISIS

El objetivo del proyecto es configurar y programar el robot P3-AT para que, de acuerdo a una ruta establecida, pueda situarse enfrente de la puerta de cada cubículo seleccionado perteneciente al Departamento de Sistemas de la UAM-Azcapotzalco.

De acuerdo al objetivo general, el análisis del proyecto se desglosa en tres componentes:

- ***El robot P3-AT***. Se encarga de recorrer el Departamento de Sistemas y visitar sus cubículos.
- ***El Departamento de Sistemas***. Es el sitio que recorre el robot. El plano del departamento, es la base para definir la ruta que debe de seguir la unidad robótica.
- ***La interfaz gráfica***. Su función es indicar al robot que cubículos deben de visitarse.

2.1 Robot Pioneer 3-AT

Pioneer 3-AT (P3-AT) es un robot de cuatro ruedas todo terreno desarrollado por la compañía *ActivMedia Robotics* ver la figura 1. Diseñado principalmente para la navegación de forma autónoma en tiempo real, cuenta con las siguientes especificaciones:

- Dimensiones: Largo x ancho x altura = 50cm x 49cm x 26cm.
- Peso: 14kg.
- Máxima velocidad: 700mm/s.
- Máxima velocidad de rotación: 140grados/s.

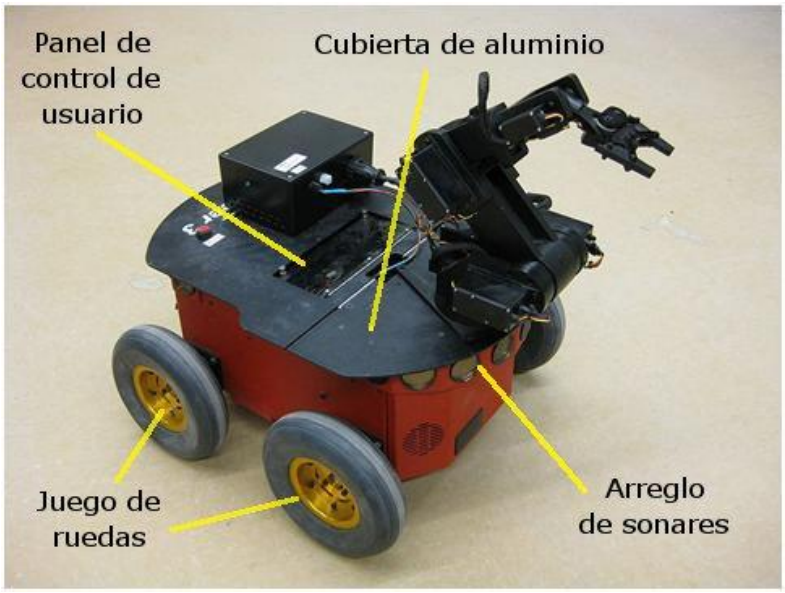


Figura 1.- Robot Pioneer 3-AT y sus partes principales

Las partes que constituyen al robot P3-AT son [4]:

- ***Cubierta de aluminio*** situada en la parte superior, su función es proteger los componentes internos del robot y proporcionar el suficiente espacio para montar accesorios y/o dispositivos. Además cuenta con un puerto de acceso, de ésta manera es posible disponer de cualquier cable proveniente de la tarjeta microcontroladora, ver la figura 2.

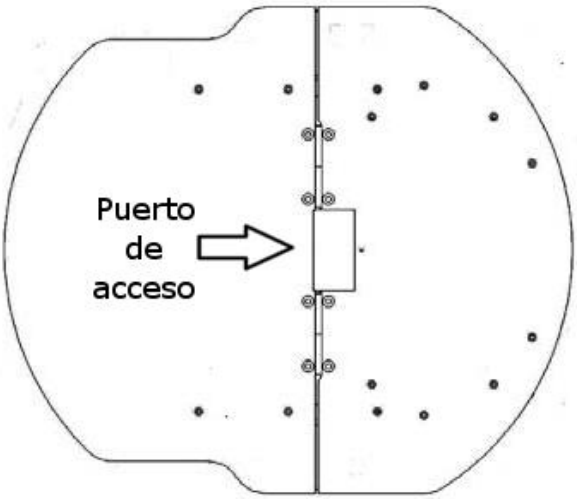


Figura 2. Cubierta de aluminio del robot.

- ***Panel de control de usuario*** localizado en la parte central de la cubierta de aluminio. Permite la comunicación del robot con el servidor de software a través de un puerto serial (RS-232), ver figura 3.

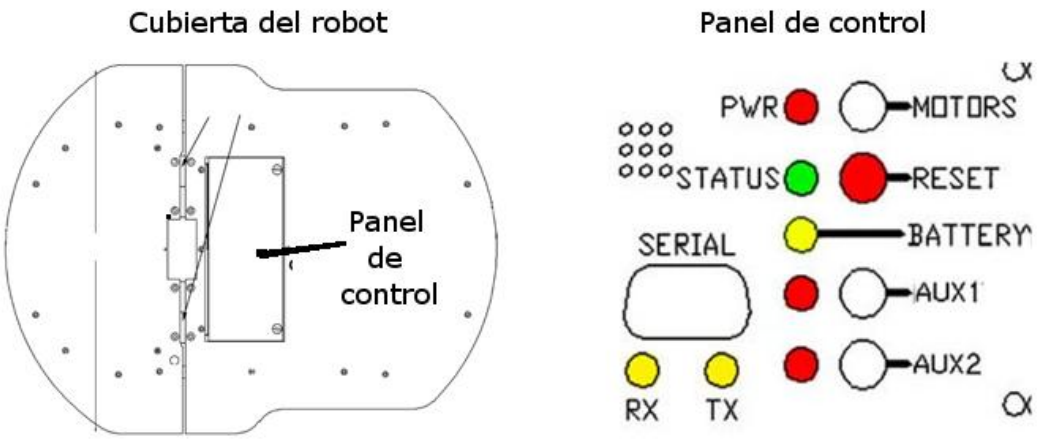


Figura 3. Panel de Control de Usuario

- Un ***arreglo de sonares*** distribuidos en la parte frontal. Su función es recabar información sobre los obstáculos que pueda presentar el terreno, con el fin de evitar colisiones, ver la figura 4.

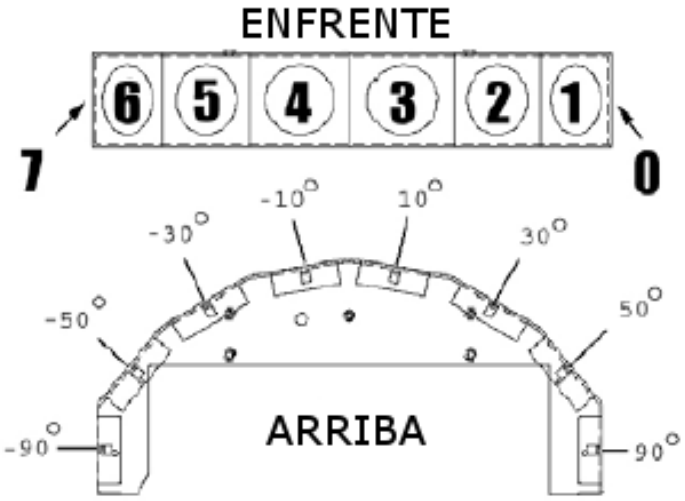


Figura 4. Arreglo de Sonares

- Tres ***baterías recargables de 12 volts*** que permiten hasta 3hrs de funcionamiento continuo, ver la figura 5.

Figura 5. Batería recargable del robot

- Un *par de* ***motores reversibles*** para cada par de ruedas y su respectiva ***tarjeta controladora*** como se muestra en la figura 6.

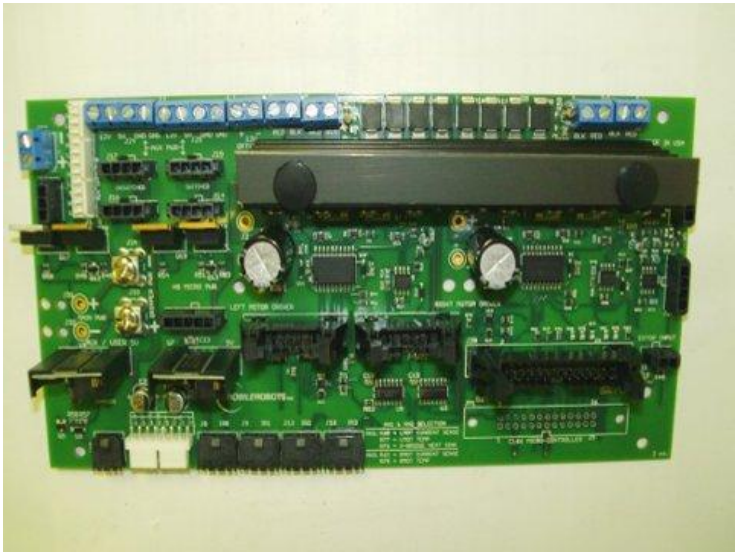


Figura 6. Tarjeta encargada de controlar los motores del robot

- Y por último, una ***tarjeta microcontroladora*** que opera bajo el sistema operativo *AROS Research Operating System* (AROS). Es el cerebro del robot, se encarga de administrar los dispositivos e interpretar las instrucciones recibidas, ver la figura 7.

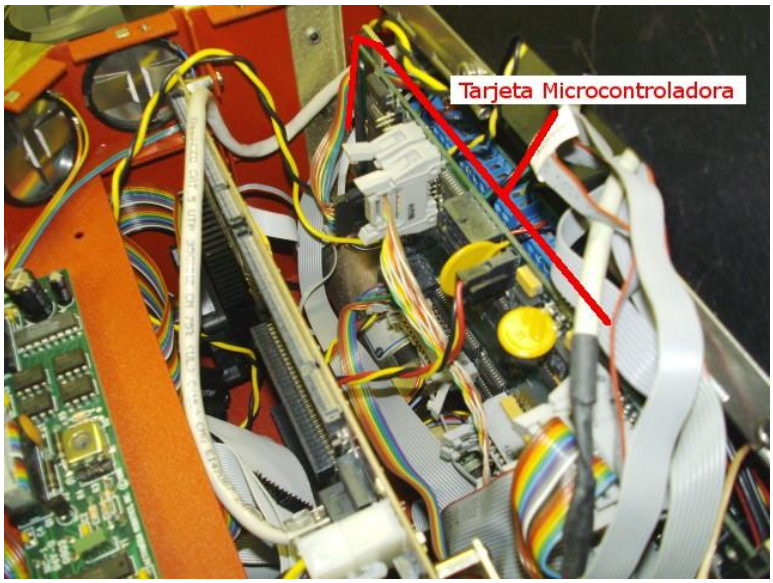


Figura 7. Tarjeta microcontroladora del robot

Una vez descritos los componentes principales, se analizan los programas necesarios para interactuar con el robot del lado del servidor software.

2.1.1 ARIA

Advanced Robot Interface for Applications (ARIA) es una librería desarrollada en C++, cuya finalidad es controlar los parámetros de movimiento del robot, por medio de comandos simples. Pero no se limita a solo enviar datos, sino que también recibe los parámetros de operación que envía el robot [5].

ARIA es una API (*Applications Programming Interface*) que opera bajo el esquema cliente/servidor y funciona de la siguiente manera:

- ***Cliente (Robot P3-AT)***: Tiene como sistema operativo AROS, su función es codificar los comandos provenientes del servidor de software en instrucciones de bajo nivel. Sin embargo, no puede llevar a cabo tareas inteligentes o de control.
- ***Servidor***: Puede ser cualquier computadora que tenga instalada la librería ARIA. Gracias a esta API, es posible establecer comunicación con el cliente vía puerto serial, además de enviar y recibir datos, como ya se había mencionado anteriormente.

ARIA es compatible con sistemas operativos *Microsoft Windows* o *GNU/Linux*. Para desarrollar aplicaciones en ARIA con *Windows*, es necesario tener instalado el compilador *Visual C++* en su versión 6 (o posterior) [1]. Al instalar la librería ARIA recomiendo buscar los siguientes directorios dentro de la carpeta de instalación (*C:\Program Files\MobileRobots\Aria*).

Docs: Manual de referencia de la API.

Examples: Contiene ejemplos de las funciones básicas del robot, cada ejemplo tiene su archivo extensión *sln* para ser compilado con *Visual C*.

2.1.2 MobileSim

Es el software encargado de simular el funcionamiento del *Pioneer 3-AT* y su entorno. Se utiliza para experimentar con las aplicaciones escritas en C++/ARIA.

MobileSim convierte un mapa creado con *Mapper 3 Basic* en un entorno virtual, dispuesto para que un robot simulado pueda recorrerlo. A continuación, emula la conexión del robot con el servidor de software a través del puerto TCP[2] [6].

---

[1] Entorno de desarrollo integrado (IDE) para lenguajes de programación C y C++. Esta especialmente diseñado para el desarrollo y depuración de código escrito para las *API's* de *Microsoft Windows*. El lenguaje de programación utilizado por esta herramienta, está basado en C++ y es compatible en la mayor parte de su código con este lenguaje, a la vez que su sintaxis es exactamente igual.

[2] ARIA es capaz de conectarse al puerto TCP en lugar del puerto serial. La clase *ArSimpleConnector* por defecto intenta establecer conexión con el puerto TCP 8101.

MobileSim presenta las siguientes características útiles para el proyecto, ver la figura 8:

- El entorno y los obstáculos son cargados a través de un archivo extensión *map*.
- Los obstáculos pueden ser configurados para ser detectados por el sonar.
- Permite salvar un *snapshot*[3]de la simulación o una serie de imágenes para hacer videos.
- Opciones de visualización para los sonares y el robot, con la finalidad de indicar el estado en que se encuentran los sonares y la ruta que ha seguido la unidad robótica.

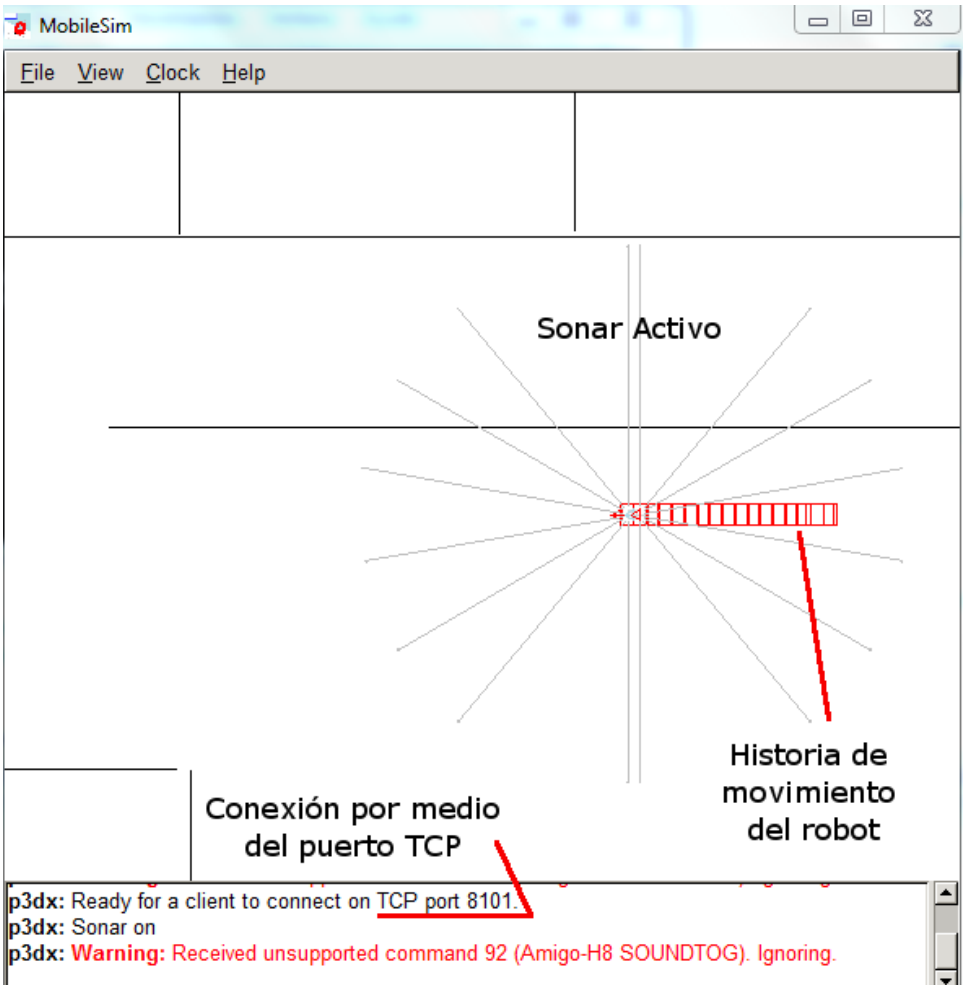


Figura 8. MobileSim en ejecución

[3] Foto instantánea.

2.1.3 Mapper 3 Basic

Es el software encargado de crear y editar los mapas que son usados por *MobileSim* y ARIA. Con *Mapper 3* es posible crear obstáculos (como paredes) y elementos lógicos (puntos de inicio, estaciones de recarga, metas y áreas prohibidas,) como se muestra en la figura 9, pero carece de la habilidad para importar y procesar los parámetros de operación que envía el robot [7].

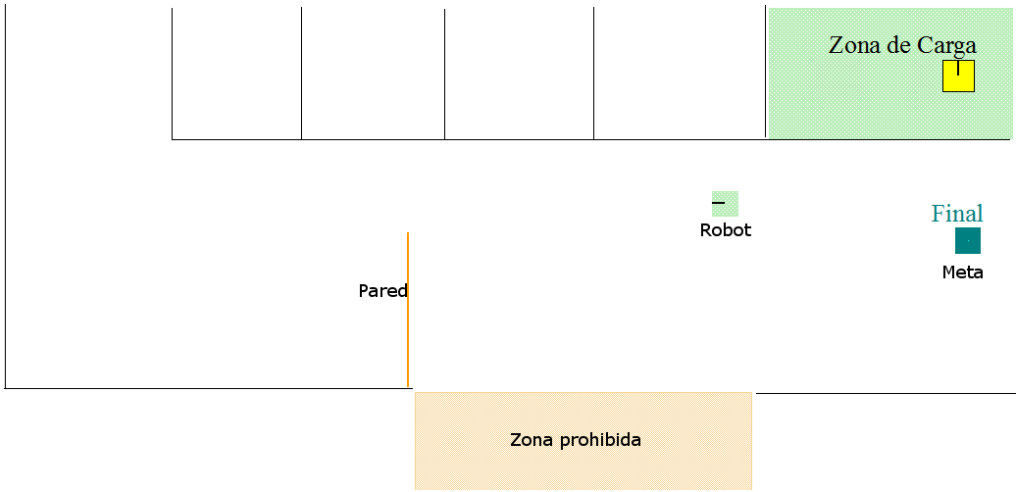


Figura 9. Obstáculos y elementos lógicos creados con Mapper 3

Todos los obstáculos y elementos lógicos son guardados en un archivo extensión *map*, que utiliza MobileSim para tareas de navegación y localización. Otra característica con la que cuenta Mapper 3, es la posibilidad de crear un repositorio con los archivos que se han creado, el único requisito es tener acceso al equipo que desempeña la labor de servidor de mapas.

2.2 El Departamento de Sistemas

Es la estructura organizacional encargada de la investigación y docencia, para la resolución de problemas que requieren de un enfoque de sistemas. Se eligió como sitio del recorrido porque abarca un solo piso y cuenta con pocos cubículos, todos ellos fáciles de identificar

El Departamento de Sistemas se localiza en la segunda planta de los edificios H y HP, está constituido de 53 cubículos repartidos de la siguiente manera: 34 pertenecen al edificio H y 19 al edificio HP como se muestra en la figura 10.

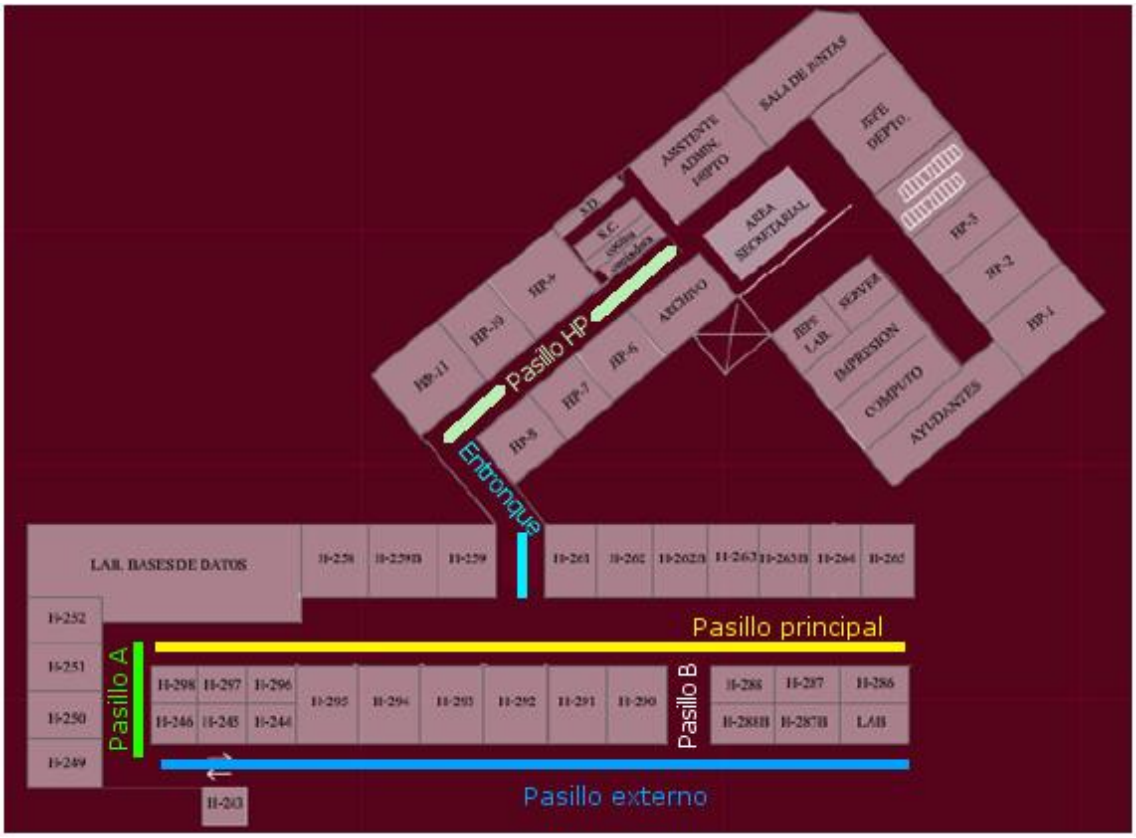


Figura 10. Plano del Departamento de Sistemas

Para visitar cada uno de los cubículos, el robot P3-AT debe de ser capaz de recorrer los pasillos que conforman el Departamento de Sistemas. Por comodidad he etiquetado cada uno de los pasillos e incluyo una breve descripción de los mismos.

- ***Pasillo Externo***. No posee ningún punto conflictivo, es el pasillo con mayor amplitud, como se muestra en la figura 11.

Figura 11. Pasillo Externo

- ***Pasillo A***: Al igual que el anterior, no posee ningún problema, ver la figura 12.

Figura 12. Pasillo A

- ***Pasillo Principal***: Tiene problemas en los tramos comprendidos entre los cubículos 298 al 295 y del 288 al 286, ambos son estrechos y como consecuencia el robot puede quedarse atascado, ver la figura 13.

Figura 13. Pasillo Principal

- ***Pasillo B***: Es un pasillo estrecho, además el robot puede quedarse atascado en el tope de la puerta que se muestra en la figura 14.

Figura 14. Pasillo B

- ***Entronque***: El edificio HP está construido a desnivel y es necesaria un rampa para poder acceder a éste, ver la figura 15.

Figura 15. Entronque

- ***Pasillo HP***: Se encuentra rotado unos 45 grados, pero no supone ningún problema, ver la figura 16.

Figura 16. Pasillo HP

2.3 La Interfaz Gráfica

Una vez definida el área de operación del robot y las herramientas para interactuar con éste, es necesario desarrollar una interfaz gráfica de usuario (GUI) que haga amigable la interacción hombre-robot. Para desarrollar la GUI es necesario cumplir con las siguientes condiciones:

- El *framework*[4] usado para implementar la GUI debe utilizar C++ de manera nativa.
- Debe integrarse con Visual C++ en su versión 8 o 9.
- Debe tener clases y métodos para dibujar gráficos 2D e interacción con elementos en pantalla.

2.3.1 Qt

Qt es un *framework* para el desarrollo de aplicaciones multiplataforma creado por la compañía *Trolltech* y que actualmente es propiedad de *Nokia*, la función más conocida de *Qt* es la creación de interfaces de usuario. También provee varias clases para facilitar tareas de programación como comunicación con bases de datos, manejo de objetos 3D, creación y manipulación de objetos 2D, entre otras cosas [8].

Qt utiliza C++ de manera nativa. A partir de la versión 4.6 *Nokia* incluyó un *plug-in* para el IDE (*Integrated Development Environment*) Visual Studio en su versión 2008; permite la integración del *framework* al compilador *Visual C++* de forma transparente.

2.3.2 QGraphicsView

QGraphicsView es el *framework* dentro de *Qt* que permite la creación e interacción de elementos gráficos 2D [9]. De manera simple, varias vistas pueden observar una misma escena y una escena puede contener elementos (*items*) de diferentes formas geométricas.

El *framework* está compuesto de 3 elementos, como se muestra en la figura 17:

- **QGraphicsScene (La escena).** Representa una escena con *items*. Es la clase que se encarga de almacenar los *widgets*[5], así como manejar y propagar eventos a cada *item*.

[4] Es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.
[5] Es un elemento de la interfaz gráfica que muestra información con la cual el usuario puede interactuar. Por ejemplo: ventanas, cajas de texto, entre otros.

- ***QGraphicsView* (La Vista)**. La clase que se encarga de proporcionar los *widgets* que se visualizan en la escena
- ***QGraphicsItem* (El *Item*)**. Representa un grupo de *items*. Es una clase para el manejo de elementos gráficos en la escena y proporciona varios *items* estándar para formas típicas como rectángulos, elipses y textos. También soporta eventos del mouse como mover, soltar, presionar y eventos del teclado.

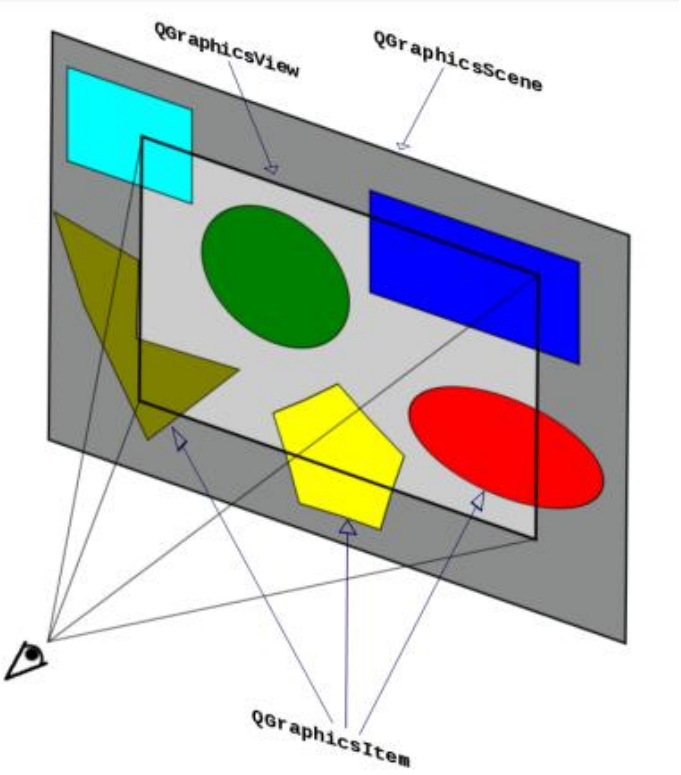


Figura 17. Modelo QGraphicsView

3 DISEÑO

"De acuerdo a los objetivos planteados, el robot P3-AT debe visitar los cubículos del Departamento de Sistemas a partir de una selección previa. "

El ***Programa Encargado de Controlar al Robot Inteligente*** (PECRI), es la aplicación responsable de hacer que el robot P3-AT visite los cubículos del Departamento de Sistemas. A continuación se representa el funcionamiento interno de PECRI, ver la figura 18.

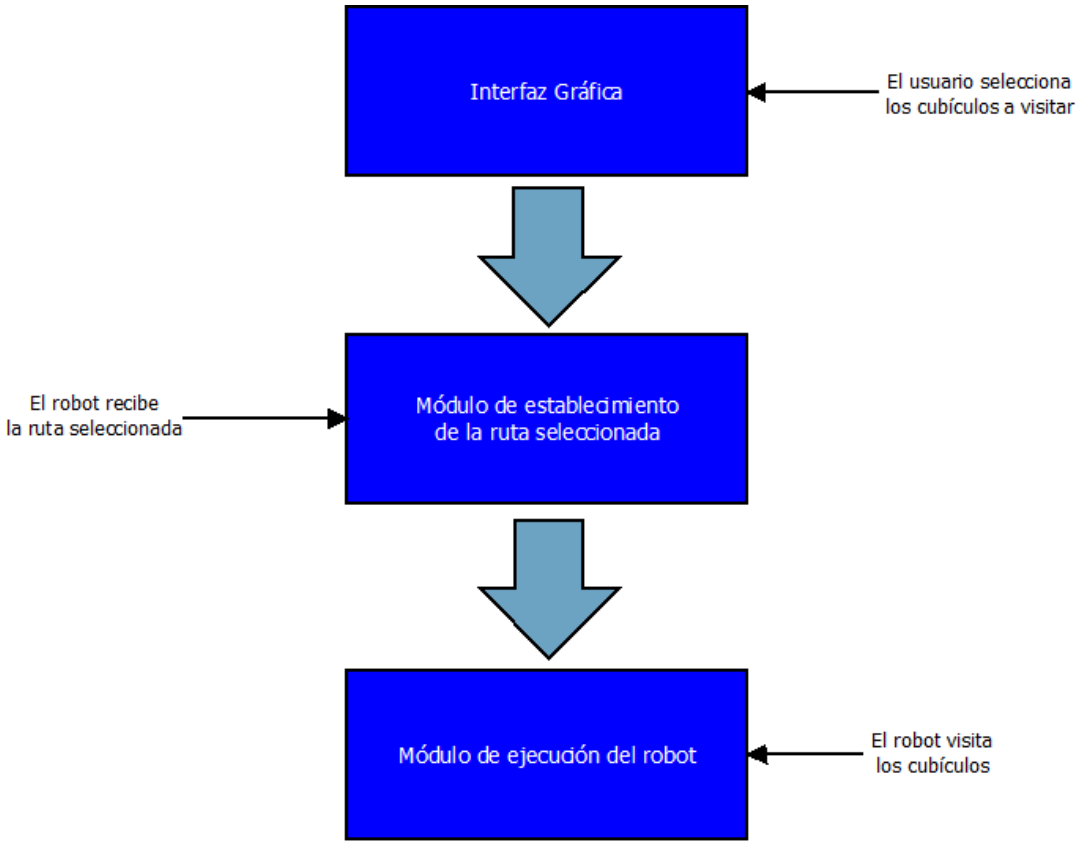


Figura 18. Diagrama de bloques de PECRI

Interfaz Gráfica

Permite al usuario seleccionar los cubículos que debe visitar el robot en su recorrido.

Módulo de establecimiento de la ruta seleccionada

Establece el orden en que se visitan los cubículos. Los datos de visita son guardados en un archivo de texto para ser cargados en el módulo de ejecución del robot.

Módulo de ejecución del robot

Es el módulo encargado de poner en marcha el robot, aquí se encuentran los comandos encargados del control. El tiempo del recorrido depende de la cantidad de cubículos seleccionados.

3.1 Recorrido del Robot P3-AT

Definir el recorrido del robot fue el primer paso para construir los módulos de "ejecución del robot" y de "establecimiento de ruta seleccionada". La idea básica es simple, establecer el orden en que se recorren los pasillos del Departamento de Sistemas. En el capítulo anterior se etiquetó cada uno de los pasillos de la siguiente forma, ver la tabla 1.

Tabla 1. Relación pasillo/cubículo

Etiqueta	Edificio (localización)	Cubículos a los que se puede acceder
Pasillo Externo	H	244, 245, 246
Pasillo A	H	249, 250, 251, 252
Pasillo Principal	H	298, 297, 296, 295, 294, 293, 292, 291, 288, 287, 286 265, 264, 263B, 263, 262B, 262, 261, 258, 259B, 259
Pasillo B	H	Ninguno
Entronque	H y HP	Ninguno
Pasillo HP	HP	11, 10, 09, 08, 07, 06

En base al análisis del Departamento de Sistemas, se decidió que el orden en que se recorren los pasillos es el siguiente, ver figura 19:

1. ***Pasillo Externo***. El punto de partida/llegada es la parte trasera del cubículo H-295, se eligió este sitio porque se localiza enfrente de la entrada del Departamento de Sistemas**.** Tomando como referencia el punto de partida, el cubículo más cercano al que se puede acceder es el H-244. De esta manera, se declaró que el robot recorre el pasillo Externo con dirección al pasillo A.
2. ***Pasillo A***. El robot recorre el pasillo con dirección al pasillo Principal.
3. ***Pasillo Principal***: El robot recorre todos los cubículos del pasillo, da vuelta en "u" enfrente del H-265 y se dirige al Entronque que conecta los edificios H y HP.
4. ***Entronque***. Se recorre el Entronque para acceder al pasillo HP.
5. ***Pasillo HP***. Es el último pasillo que se recorre por la dificultad que supone librar el Entronque.

El regreso al punto de llegada se definió de la siguiente manera, ver la figura 20:

1. **Pasillo HP**. Se toma como punto de inicio la puerta del cubículo HP-06. Se recorre todo el pasillo con dirección al Entronque.
2. **Entronque**. Es la única forma de salir del edificio HP.
3. **Pasillo Principal**. Se recorre el pasillo con la finalidad de quedar enfrente del cubículo H-262B.
4. **Pasillo B**. Se recorre con el fin de llegar al pasillo Externo.
5. **Pasillo Externo**. Hay que recorrer el pasillo con dirección al punto de llegada (parte trasera del cubículo H-295).

Figura 19. Recorrido del robot (ida)

Figura 20. Recorrido del robot (regreso)

A continuación, se identificaron las ***estructuras de control*** que permiten el movimiento del robot a través de los pasillos del Departamento del Sistemas.

- ***Avanzar***, mueve al robot en línea recta.
- ***Girar***, rota el robot sobre su eje.
- ***Detener***, detiene el movimiento del robot.

Con las ***estructuras de control*** definidas, se desarrolló el algoritmo que se usa para recorrer los pasillos, ver la figura 21.

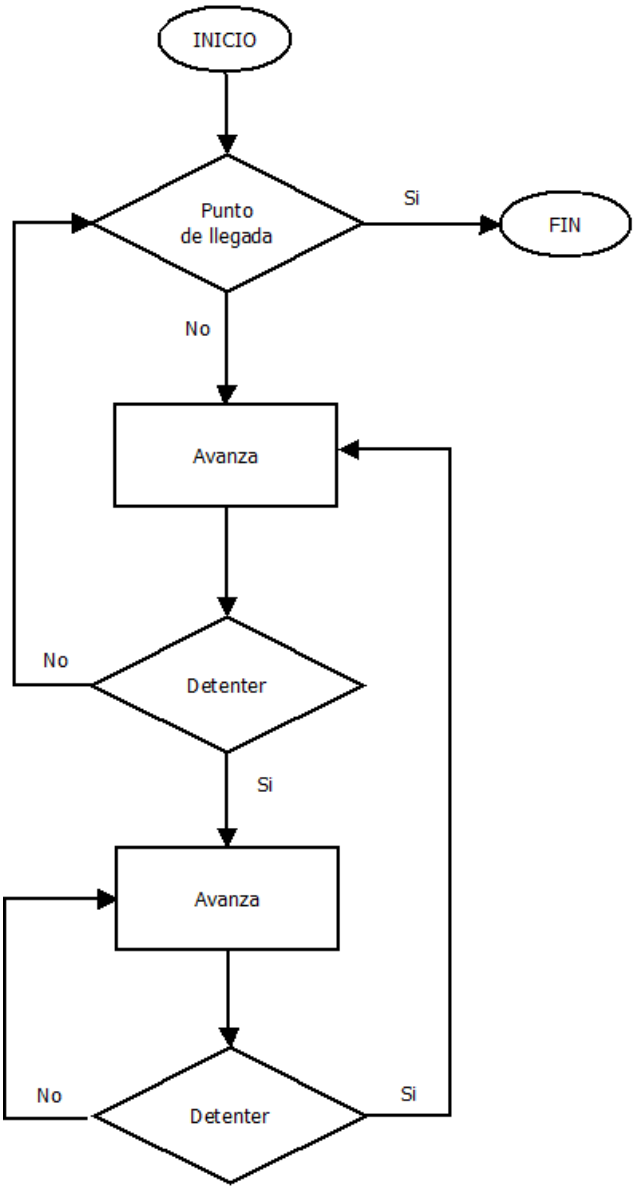


Figura 21. Diagrama de flujo del recorrido del robot

El algoritmo es muy simple, avanzar hasta topar con pared y girar; en algunos casos es necesario establecer un punto específico donde debe detenerse el robot, pero todo esto queda detallado más adelante.

3.1.1 Orden de visita de los cubículos

Establecer el orden de visita de los cubículos, fue el paso siguiente en el desarrollo los módulos de "ejecución del robot" y de "establecimiento de ruta seleccionada". Una vez definida la ruta es casi automático el ordenamiento de los cubículos, salvo algunos casos. Por ejemplo, los cubículos H-258/H-259B/H-259 y H-295/H-294/H-293 se encuentran en forma paralela y a la

misma altura, pero la puerta de los primeros está antes, así que se decidió intercalarlos quedando de la siguiente manera:

H-258 / H-295 / H-259B / H-294 / H-259 / H-293

Los demás cubículos fueron ordenados de acuerdo al recorrido de los pasillos. A continuación se muestra la tabla 2 que contiene el orden de visita de los cubículos.

Tabla 2. Orden de visita de los cubículos

Etiqueta del pasillo	Orden de visita	Cubículo
Pasillo Externo	1	H-244
	2	H-245
	3	H-246
Pasillo A	4	H-249
	5	H-250
	6	H-251
	7	H-252
Pasillo Principal	8	H-298
	9	H-297
	10	H-296
	11	H-258
	12	H-295
	13	H-259B
	14	H-294
	15	H-259
	16	H-293
	17	H-292
	18	H-291
	19	H-290
	20	H-288
	21	H-287
	22	H-286
	23	H-265
	24	H-264
	25	H-263B
	26	H-263
	27	H-262B
	28	H-262
	29	H-261
Pasillo HP	30	HP-08
	31	HP-07
	32	HP-06
	33	HP-11
	34	HP-10
	35	HP-09

Dentro de PECRI, la lista anterior se utiliza de la siguiente manera:

- **Módulo de establecimiento de la rutina seleccionada:** Se guarda en un archivo de texto para posteriormente ser cargada en el módulo de ejecución del robot.
- **Módulo de ejecución del robot:** Los datos cargados se envían a un arreglo de números, donde 1 representa la operación de visita.

3.1.2 Rutina de visita del cubículo

Es el último paso, se refiere a las instrucciones que ejecuta el robot para posicionarse enfrente de la puerta del cubículo a visitar. La rutina se divide en los siguientes pasos, ver la figura 22.

a) Posicionarse enfrente de la puerta del cubículo.
b) Girar 90° a la izquierda o derecha, dependiendo de la posición del cubículo.
c) Avanzar de frente de manera que el robot no entre al cubículo.
d) Regresar a la posición original (punto de visita).

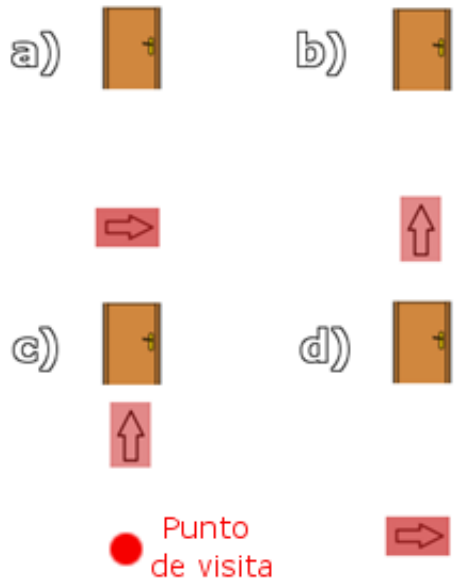


Figura 22. Rutinas de visita

Ejecutar una rutina de visita, involucra las siguientes ***estructuras de control***:

- ***Avanzar***, mueve al robot en línea recta.
- ***Girar***, rota el robot sobre su eje.
- ***Detener***, detiene el movimiento del robot.

El algoritmo que se usó para diseñar la rutina de visita es el siguiente, ve la figura 23:

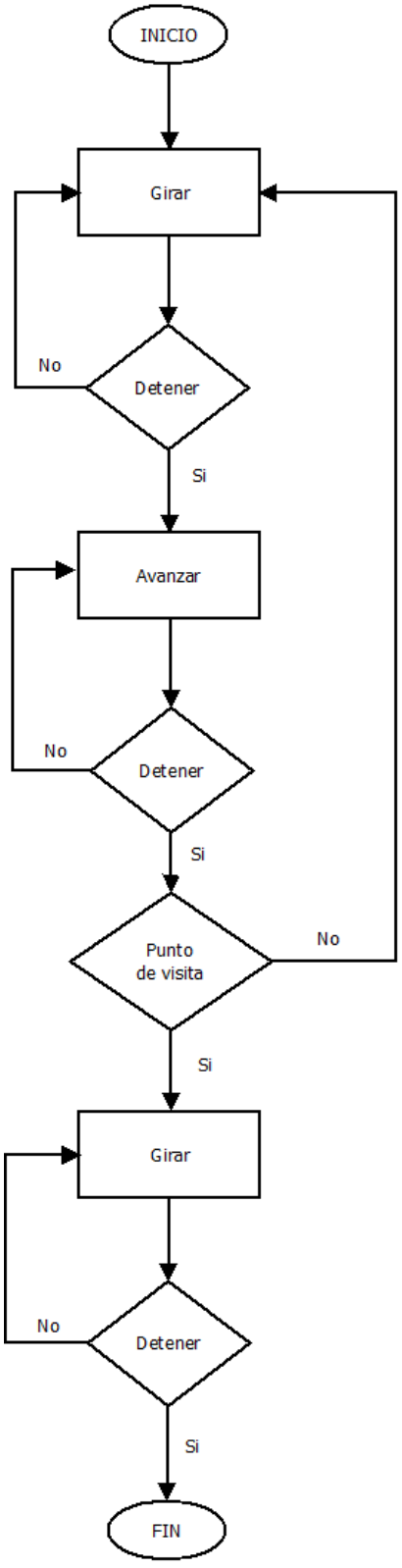


Figura 23. Diagrama de flujo de la rutina de visita del robot

Cabe mencionar que el *"punto de visita"*, es la posición original de donde partió el robot. También es un algoritmo sencillo, involucra girar y avanzar hasta topar con pared y por último regresa a la posición original.

Una vez que se junta el algoritmo encargado de recorrer los pasillos y la rutina de visita de cubículo. Se obtiene como resultado el algoritmo con el que funciona PECRI, como se muestra en la figura 24.

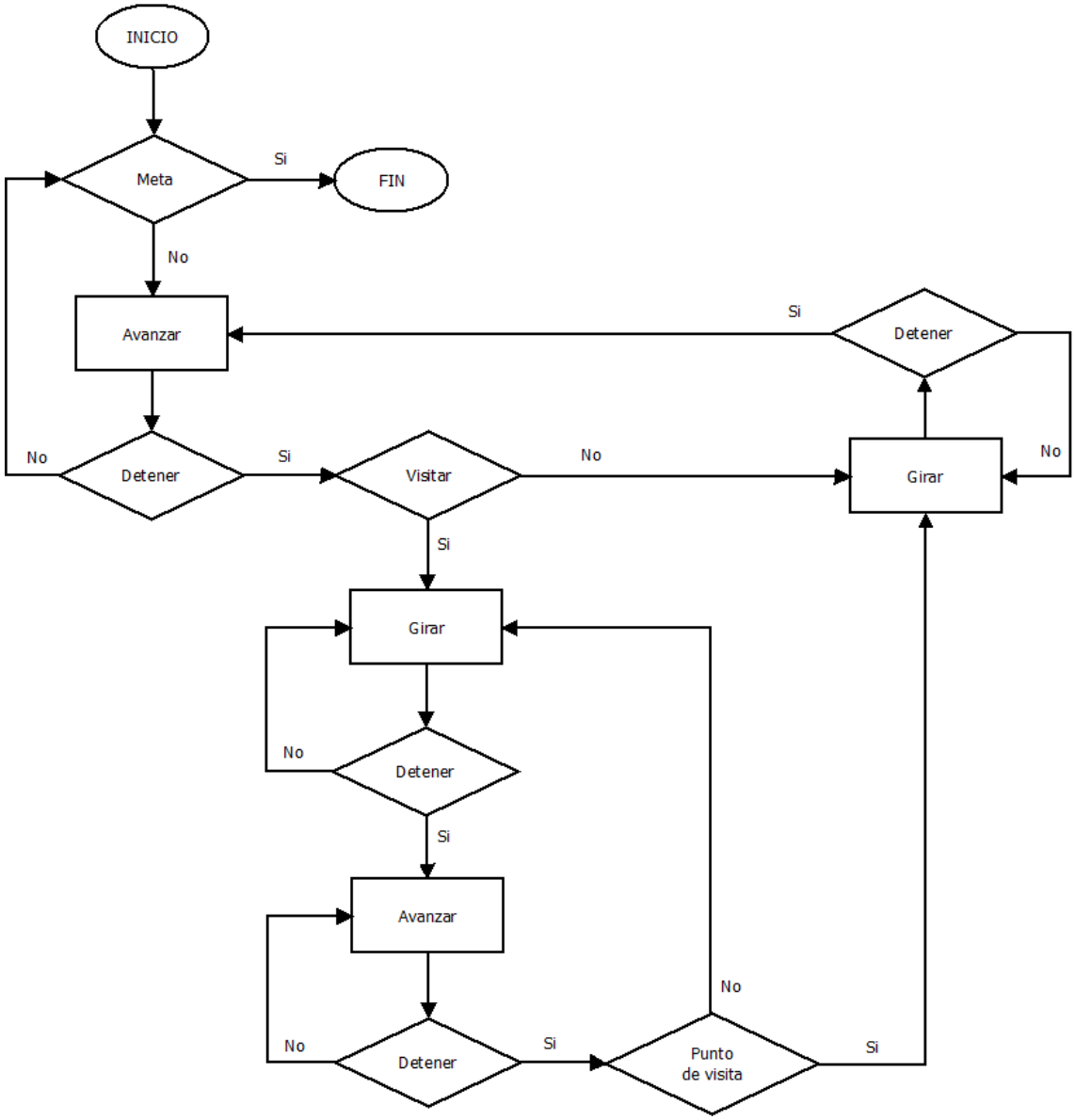


Figura 24. Diagrama de flujo de PECRI

3.2 Interfaz Gráfica

La interfaz gráfica tiene como punto de referencia el mapa del Departamento de Sistemas; la idea central es tener un panel de selección de cubículo y controles de usuario, ver la figura 25.

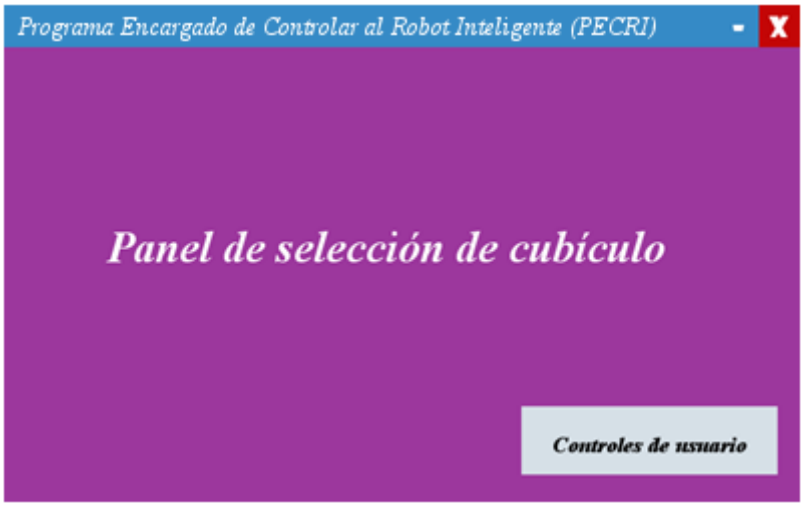


Figura 25. Bosquejo de la GUI de PECRI

Panel de selección de cubículo. El usuario puede seleccionar los cubículos que visita el robot. Cada cubículo es un elemento interactivo y está distribuido de manera similar al plano del Departamento de Sistemas.

Controles de usuario. Está conformado por los siguientes botones:

- *Inicio*: Su función es iniciar la aplicación, una vez presionado la GUI será bloqueada hasta que el robot termine su recorrido.
- *Ayuda*: Abre la documentación del PECRI.
- *Salir*: Encargado de cerrar la interfaz gráfica

De acuerdo al bosquejo de la interfaz gráfica de PECRI, se tomó la decisión que el panel de selección sería un gráfico interactivo del Departamento de Sistemas, donde los cubículos jugarían el rol de elementos seleccionables.

Se definió que la secuencia para seleccionar un cubículo sería la siguiente:

- Posicionar el puntero del ratón sobre el cubículo que se desee seleccionar.
- Presionar el botón izquierdo del ratón sobre el gráfico del cubículo. Si se presiona fuera del elemento gráfico, la selección no debe efectuarse.
- Si la selección fue exitosa, el elemento gráfico debe cambiar de color.

Los colores que indican el estado de un cubículo son los siguientes: azul cielo si el cubículo fue seleccionado exitosamente (activo) y magenta para aquellos que aún no han sido elegidos (pasivo), ver figura 26. Por defecto, todos los elementos gráficos (cubículos) se inicializan en color magenta.

Figura 26. Estado activo y pasivo de los cubículos

Como se observa en la figura 27, el diseño de la interfaz gráfica de PECRI es sencillo pero eficiente. Los elementos gráficos están dispuestos de forma similar al plano del Departamento de Sistemas y cambian de color al hacer *click* sobre ellos, de este modo el proceso de selección de cubículo se vuelve una tarea intuitiva para el usuario.

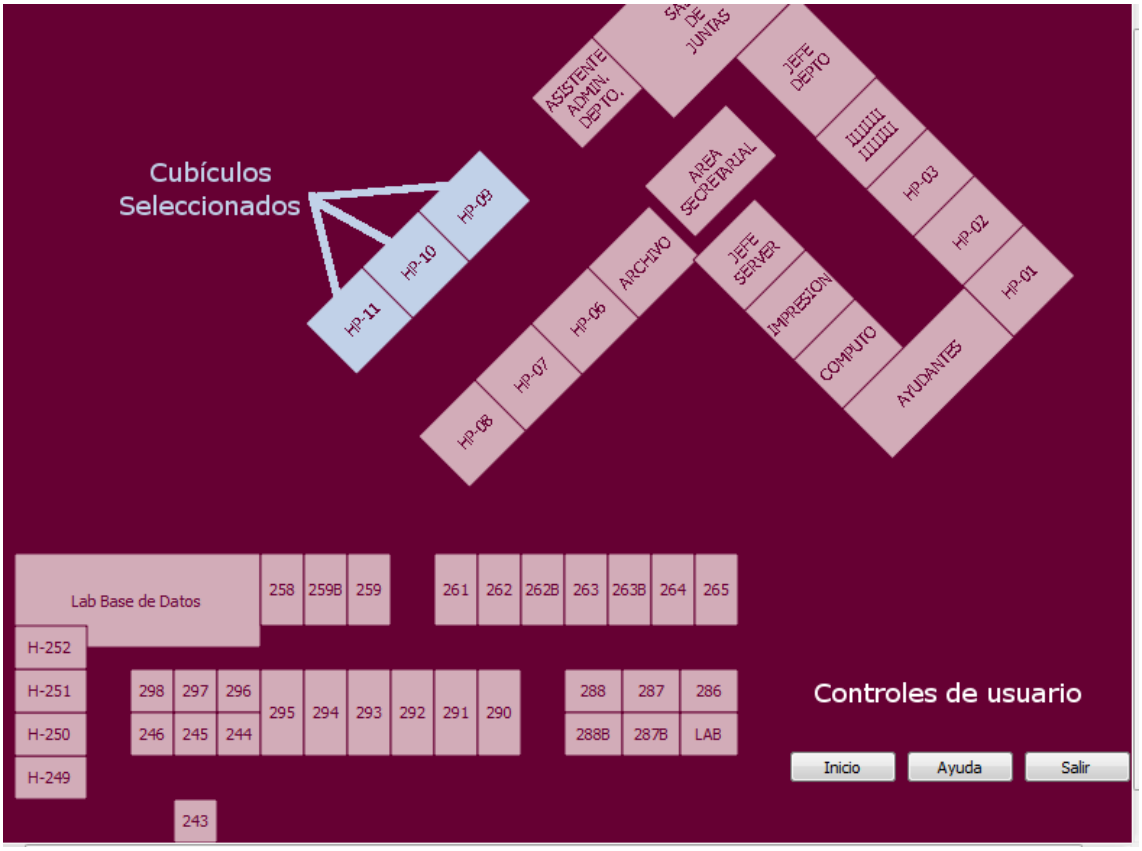


Figura 27. Interfaz gráfica de PECRI

4 PRUEBAS Y EXPERIMENTOS

En esta parte del reporte se muestran las pruebas y experimentos que se realizaron sobre al robot P3-AT. El capítulo se estructura de la siguiente forma:

Sección 4.1, es una guía de instalación del software indispensable para hacer las pruebas.

Sección 4.2, se proporciona la guía de uso del software PECRI.

Sección 4.3, está constituida por un pequeño tutorial para crear programas con ARIA, además se incluye una descripción de las aplicaciones creadas antes del desarrollo de PECRI (primeros experimentos).

Sección 4.4, se simula el funcionamiento de PECRI con MobileSim.

Sección 4.5, se recorre el Departamento de Sistemas con el robot P3-AT.

Y por último se incluye la documentación, indispensable para hacer cambios sobre el código.

4.1 Instalación del software

Antes de comenzar es necesario cumplir con los siguientes requisitos:

- Tener instalado un sistema *Microsoft Windows NT* (*XP, Vista, 7*) [10].
- Contar con la instalación de *Microsoft Visual Studio 2008 Professional* [11].

A continuación se presenta la estructura de esta sección:

1. ***Sección 4.1.1***, contiene el procedimiento de instalación de *Qt*, *framework* indispensable para desarrollar interfaces gráficas y necesario para ejecutar PECRI.
2. ***Sección 4.1.2***, contiene el procedimiento de instalación de ARIA, *MobileSim* y *Mapper*, programas necesarios para trabajar con el robot P3-AT.
3. ***Sección 4.1.3***, contiene el procedimiento para instalar PECRI.

4.1.1 Guía de instalación: Qt

Paso 1: Archivos necesarios.

- *Qt SDK Open Source v.4.7.3* [12].
- *Qt Add-in* para *Visual Studio 2008* [13].

Paso 2: Instalación Qt SDK

- Ejecutar el archivo que contiene "*Qt SDK Open Source v.4.7.3"*, si el usuario posee un sistema *Windows Vista o 7*, es necesario pulsar botón derecho del mouse y seleccionar la opción "*Ejecutar como administrador*". De lo contrario, el ejecutable no terminará de instalar las librerías necesarias. Hay que pulsar "siguiente" en cada una de las pantallas y aceptar el acuerdo de licencia. El fabricante recomienda utilizar la ruta por defecto (C:\Qt\4.7.3), ver figura 28.

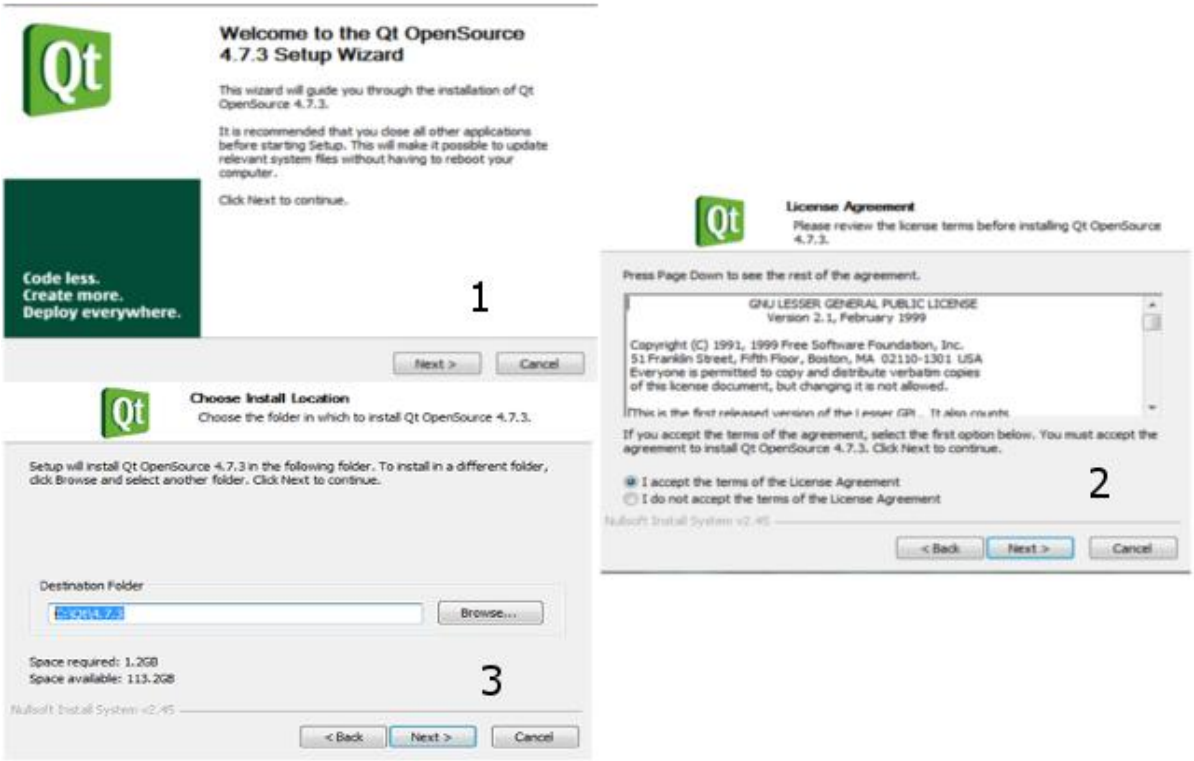


Figura 28. Instalación Qt

Paso 3:Add-in para Visual Studio

Qt al ser desarrollado para "*MinGW*" no posee compatibilidad con el compilador *Visual C++* de forma nativa. Gracias a éste *plugin*, es posible construir soluciones con el compilador de *Microsoft, s*olo hay que seguir los siguientes pasos:

- Ejecutar el archivo que contiene "*Qt Add-in* para *Visual Studio 2008"*.

Nota: Debe estar instalado Qt SDK (*Software Development Kit*)*, de otra forma no se podrá continuar con la instalación.*

- Una vez ejecutado, debemos pulsar "siguiente", aceptar el acuerdo de licencia, seleccionar las características a instalar (por defecto dejarlas tal y como están) y definir ruta de instalación (*C:\Program Files\Nokia\Qt4VSAddin*), ver la figura 29.

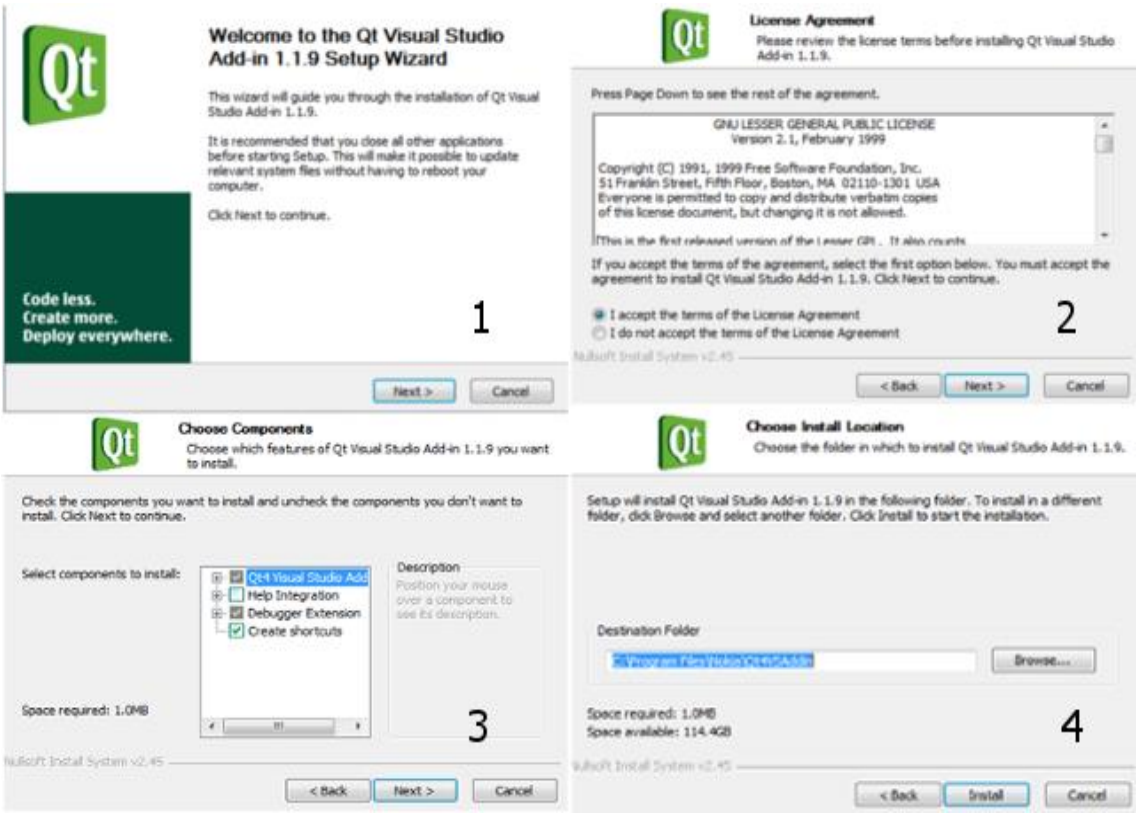


Figura 29. Instalación Qt Add-in

4.1.2 Guía de instalación: ARIA, MobileSim y Mapper 3 Basic

Paso 1: Archivos necesarios.

- *ARIA 2.7.3*
- *MobileSim 0.5.0*
- *Mapper 3 Basic 2.2.5*

Paso 2: Instalación de la suite

- El procedimiento es el mismo para los 3 archivos, hay que pulsar “siguiente” en cada una de las pantallas y aceptar el acuerdo de licencia. El fabricante recomienda utilizar la ruta por defecto.

    - **ARIA:** *C:\Program Files\MobileRobots\Aria*. Ver la figura 30.
    - **MobileSim:** *C:\Program Files\MobileRobots\MobileSim*. Ver la figura 31.
    - **Mapper 3:** *C:\Program Files\MobileRobots\Mapper3Basic*. Ver la figura 32.

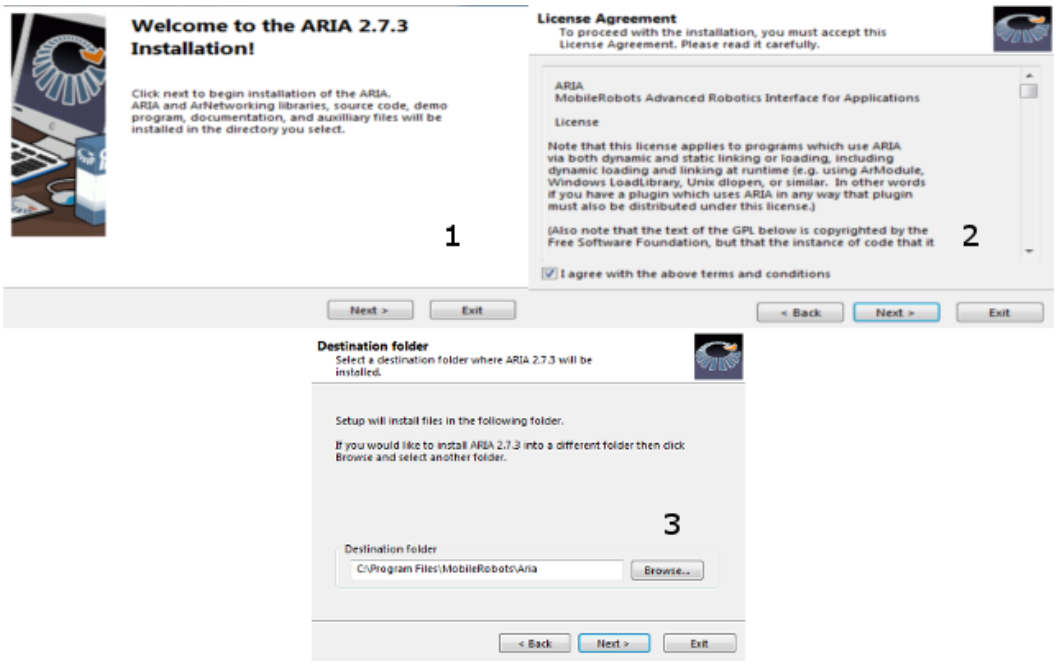


Figura 30. Instalación ARIA

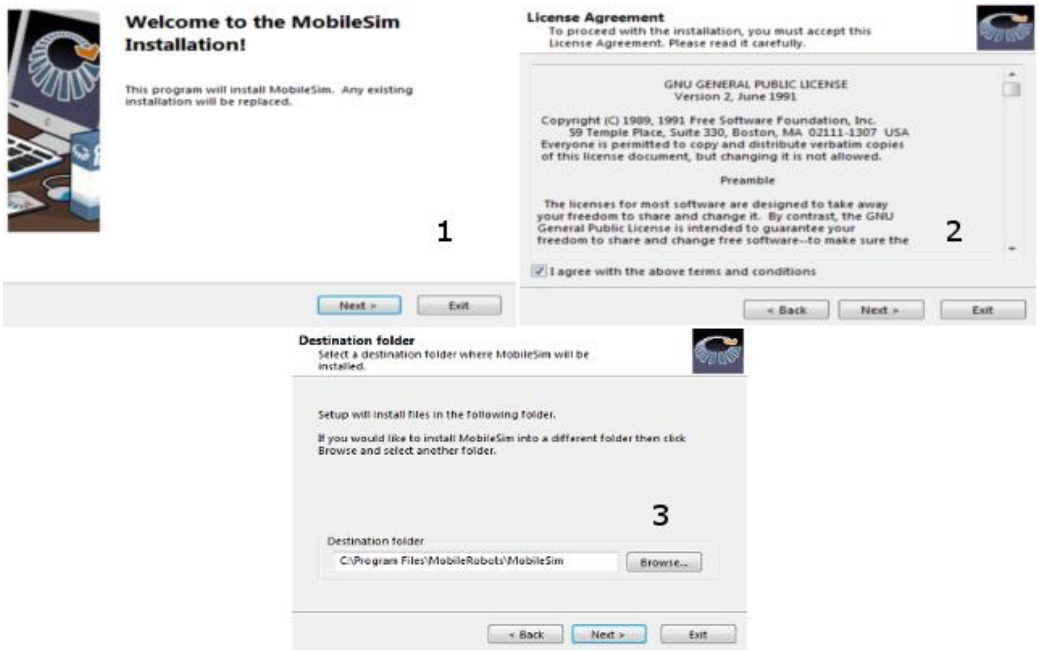


Figura 31. Instalación MobileSim

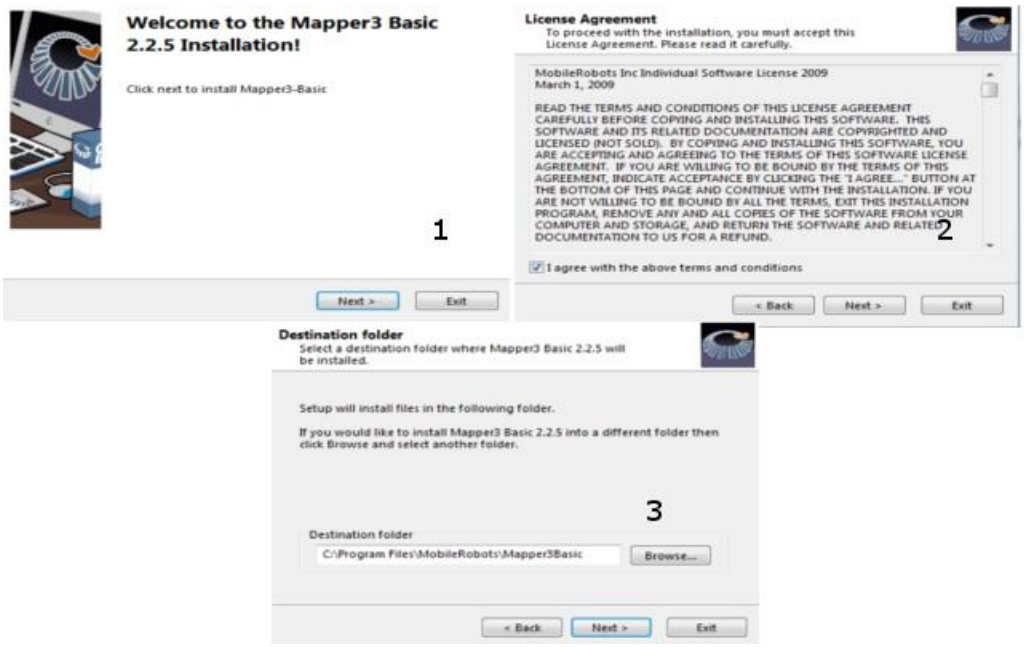


Figura 32. Instalación Mapper Basic

4.1.3 Guía de instalación: PECRI

Antes de instalar el ***PECRI***, es necesario cumplir los siguientes requisitos:

- Tener instalado *Qt SDK Open Source v.4.7.3* (ver la sección 4.1.1).
- Tener instalado *ARIA 2.7.3* (ver la sección 4.1.2).

Paso 1: Archivos necesarios.

- PECRI (*pecri.exe*). Este archivo es la GUI.
- PECRI (*robot.exe*). Este archivo es el recorrido.

Paso 2: Instalar "robot.exe"

- Primero hay que crear una carpeta en el directorio raíz (por defecto, "*C*:\") y asignarle el nombre "pecri".
- Para instalar el programa que ejecuta el recorrido del robot P3-AT, es necesario mover el archivo "robot.exe" al siguiente directorio "*C*:\pecri".
  Nota: En la documentación incluida en la sección 4.6 se explica cómo definir una ruta distinta.
- Y por último, hay que copiar el contenido de la carpeta "*C:\Program Files\MobileRobots\Aria\bin*" y vaciarlo en "*C*:\pecri".

Paso 3: Instalación de PECRI

- El primer paso es instalar las bibliotecas necesarias para ejecutar PECRI. Existen dos formas de hacerlo.

1. Mover el archivo “pecri.exe” al siguiente directorio:

C:\Qt\4.7.3\bin

2. Mover los siguientes archivos localizados en la carpeta “*C:\Qt\4.7.3\bin*” al directorio donde se encuentra “pecri.exe”:

QtCore4.dll, QtCored4.dll, QtGui4.dll, QtGuid4.dl

- Dar doble *click* en el archivo “pecri.exe” para ejecutar la aplicación.

4.2 Guía de uso PECRI

Para ejecutar PECRI, hay que dar doble click en el archivo (*pecri.exe*) ver la figura 33.

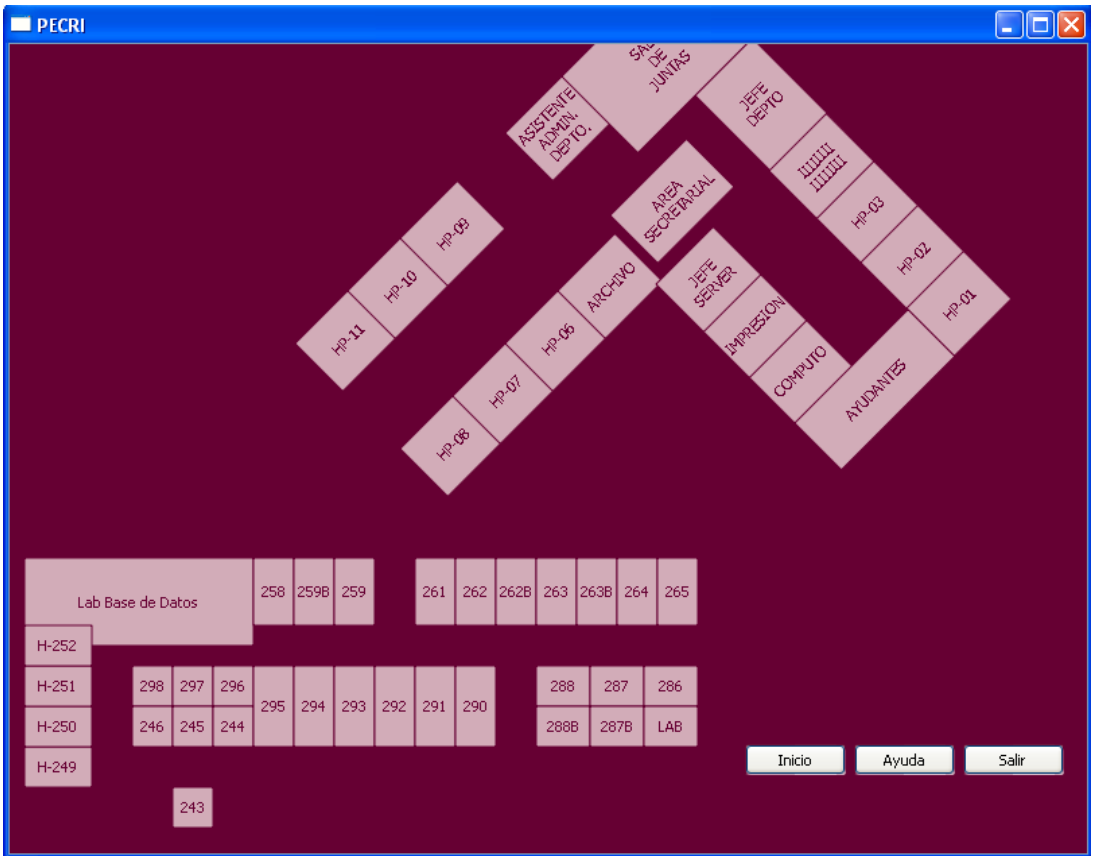


Figura 33. Ejecución de PECRI

El software se compone de tres elementos principales:

- La ventana.
- Los cubículos.
- Los botones.

La ventana: Es el elemento encargado de mostrar la aplicación. La resolución mínima que maneja es de 800x600 *pixeles*, pero puede ajustarse al ancho y alto de la pantalla sin perder el aspecto.

Los cubículos: Es el elemento principal de la aplicación, tiene la propiedad de ser un objeto seleccionable. Para activar/desactivar cada cubículo, es necesario posicionarse en el centro de éste y pulsar botón izquierdo del ratón. Tabla 3.

Tabla 3. Estado de activación

Estado	Cubículo
Apagado	265
Encendido	265

Los botones.

- ***Inicio***: Encargado de poner en marcha el Robot P3-AT. Una vez pulsado, el robot inicia la visita de cada cubículo.
- ***Ayuda***: Abre la documentación de PECRI
- ***Salir***: Como su nombre lo indica, sale de la aplicación.

4.3 Desarrollar aplicaciones con ARIA

De acuerdo a la documentación de ARIA [14], para desarrollar aplicaciones con esta librería utilizando *Visual Studio* como IDE, es necesario ir a la carpeta *examples* (por defecto, *C:\ProgramFiles\MobileRobots\Aria\examples)* abrir cualquier archivo solución (archivo extensión *sln*) y modificar el código dependiendo de lo que se necesite.

Para compilar el código hay que presionar "F5", el ejecutable se crea en la ruta *C:\Program Files\MobileRobots\Aria\bin*. Para cambiar la ubicación del ejecutable, simplemente hay que mover el archivo generado junto con todo el contenido de la carpeta *bin*.

4.3.1 Primeros experimentos

En las primeras etapas del proyecto, se crearon una serie de programas que tenían como finalidad resolver el problema de la visita de los cubículos y que a final de cuentas, se convirtieron en los experimentos que proporcionaron los conocimientos clave para el desarrollo de PECRI. En total se hicieron tres aplicaciones distintas y las etiqueté de la siguiente forma, ver la tabla 4.

Tabla 4. Primeros experimentos con ARIA

Experimento	Descripción
Comandos de movimiento	Se creó con la finalidad, de saber si era posible visitar los cubículos del Departamento de Sistemas sólo con comandos básicos como: avanzar, girar y detener.
Comandos de movimiento avanzados	Al igual que la aplicación anterior se trató de realizar la visita a los cubículos, utilizando comandos de movimiento que permiten controlar la distancia que se recorre y el ángulo de giro del robot.
Navegación automática	Se creó para ver si el robot era capaz de mantenerse al centro de cada pasillo y esquivar obstáculos.

El modelo actual (PECRI) es la combinación de los experimentos *comandos de movimiento* y *navegación automática,* y realiza el recorrido sin estrellarse en las paredes.

A continuación se presentan los resultados que arrojaron los experimentos.

Comandos de movimiento

El primer experimento aportó información sobre los siguientes comandos:

- ***setVel***. Establece la velocidad de las cuatro ruedas del robot, pero también sirve para avanzar en línea recta.
- ***setRot***. Establece la velocidad de rotación, pero sirve para hacer girar el robot en su propio eje.
- ***stop***. Detiene el robot.

Todos los comandos funcionaron, pero fue necesario establecer el tiempo que deben de estar operando y como consecuencia se pierde precisión en tareas como: rotar en un cierto ángulo y recorrer una distancia definida. Debido a la falta de precisión, tarde o temprano el robot se estrella contra la pared (en la simulación).

Comandos de movimiento avanzados

Entonces se creó el segundo experimento, para resolver el problema de precisión que poseía el primero, aportando las siguientes herramientas.

- ***moveTo, isMoveDone***. Hace que el robot avance en línea recta, el usuario debe establecer la distancia (máximo 1 metro). Sus principales defectos son: es imposible controlar la velocidad de movimiento, y avanza en reversa.
- ***setHeading, isHeadingDone***. Rota el robot sobre su eje, dependiendo del valor ingresado por el usuario. Este comando no sirve, el robot gira sin control en ambas direcciones.

Fue un completo fracaso la aplicación, pero sirvió para definir los comandos básicos de movimiento (*setVel, setRot, stop*) que fueron utilizados en PECRI. Cada uno de los comandos analizados está incluido dentro del manual de referencia de la clase *ArRobot* [15].

Navegación automática

Tomando como base el primer experimento, se decidió que la mejor forma de completar el recorrido al Departamento de Sistemas y visitar los cubículos que lo conforman, era evitar que el robot se estrelle contra las paredes y mantenerlo lo más centrado posible en los pasillos.

Navegando en la documentación de ARIA, se encontró la clase *ArActionAvoidFront* [16].

ArActionAvoidFront. Esta clase le proporciona al robot los métodos necesarios para esquivar los obstáculos que presente el camino. Hace uso del sonar para realizar las labores de detección de objetos (máximo 70 centímetros), como se muestra en la figura 34. Cuando un obstáculo es detectado, la clase le indica al robot que debe de girar en el ángulo contrario al objeto (el usuario define el ángulo de giro).

Es importante mencionar que el robot nunca detiene su marcha, por tanto si lleva mucha velocidad, es posible que no detecte el obstáculo o no tenga el suficiente tiempo para esquivarlo.

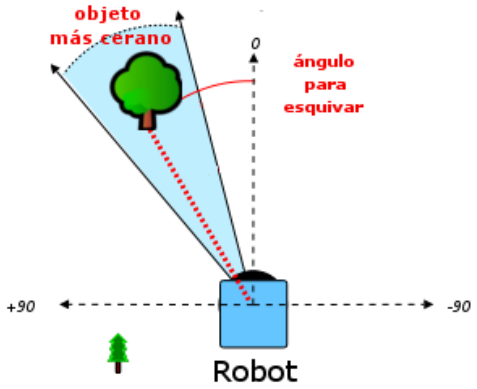


Figura 34. Funcionamiento de la clase

El experimento fue un éxito, en la simulación muy pocas veces se estrelló contra los muros debido a que el robot llevaba mucha velocidad, yo recomiendo poner valores por debajo de 400*mm/s,* con velocidades mayores el robot ya no detecta los obstáculos de forma tan precisa. Es necesario desactivar *ArActionAvoidFront* cuando se visite cualquier cubículo, porque involucra quedar muy cerca de los muros y puertas.

4.4 Simulación de PECRI

Para simular el funcionamiento de PECRI con *MobileSim*, es necesario cumplir con los siguientes requisitos:

- Tener instalado *Qt*, *MobileSim*, ARIA y *Mapper 3 Basic*. En la *secciones 3.1.1, 3.1.2 y 3.1.3* se explica el procedimiento de instalación.

- Tener creado un mapa con *Mapper 3 Basic*.

El primer paso es abrir *MobileSim* e importar el archivo creado en *Mapper*, como se muestra en la figura 35.

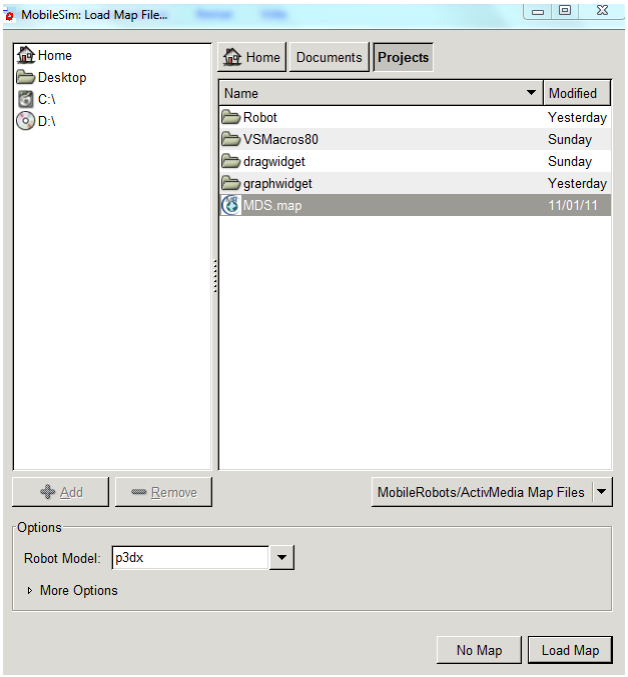


Figura 35. MobileSim, importar un mapa

A continuación, abrimos PECRI y seleccionamos los cubículos a visitar. Damos *click* en iniciar para que la simulación comience, ver la figura 36.

Para la simulación fueron seleccionados los cubículo H-244, H-246, H-297 y en MobileSim se activó la opción "*trails*" para mostrar la "historia de movimiento" del robot.

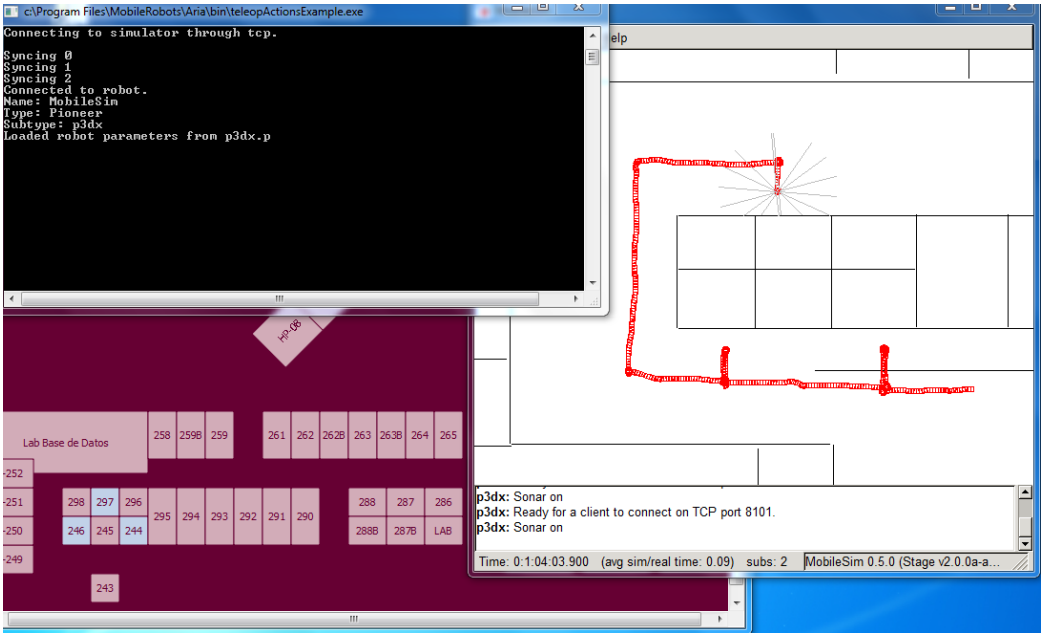


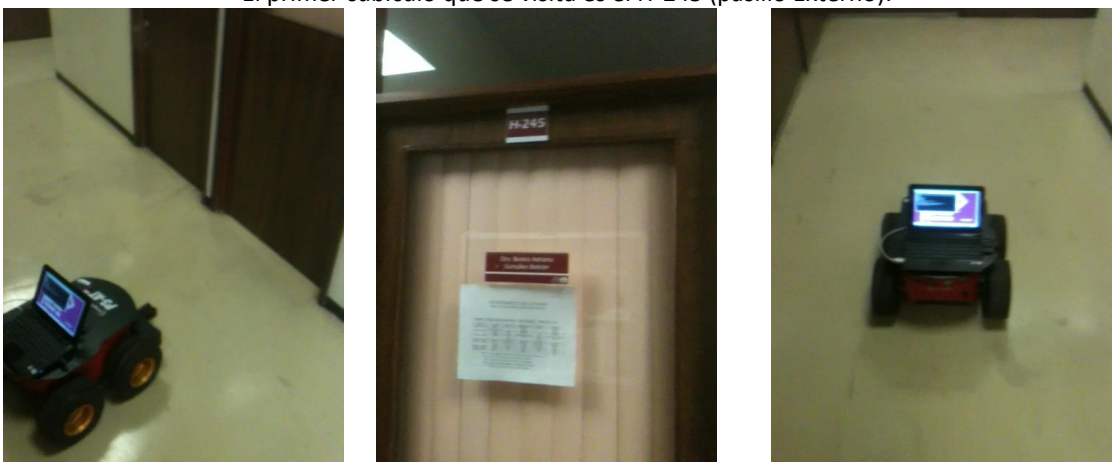
Figura 36. Simulación de PECRI con MobileSim

4.5 Pruebas sobre el robot

El último paso en el desarrollo del proyecto fue realizar las pruebas sobre el robot. A continuación se muestra la visita del robot a los siguientes cubículos, ver tabla 5:

H-245, H-251, H-259, H-292, H-288, H-287, H-264, H-262.

Tabla 5. Recorrido del robot al Departamento de Sistemas

El punto de partida es la parte trasera del cubículo H-295 (pasillo Externo). 
El primer cubículo que se visita es el H-245 (pasillo Externo). 
El robot da vuelta para entrar al pasillo A y se dispone a visitar el cubículo H-251. 

Se visita el cubículo H-251 y el robot da vuelta para entrar al pasillo Principal.


Se recorre el tramo comprendido entre los cubículos H-298 y H-295 (pasillo Principal).



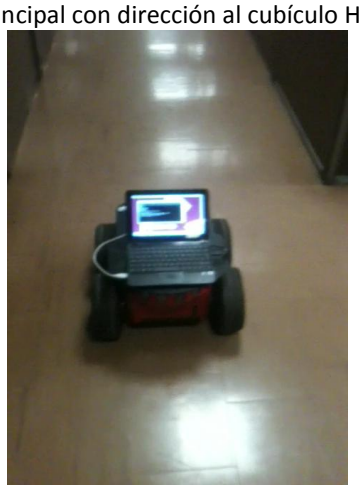
El robot visita el cubículos H-259 (pasillo Principal).



Justo después visita el cubículo H-292.



El robot recorre el pasillo Principal con dirección al cubículo H-288 y se dispone a visitarlo.


Visitado el cubículo H-288 inmediatamente el robot se prepara a hacer lo mismo con el H-287.


Se visita el cubículo H-287, el robot da vuelta en “u” para visitar el cubículo H-264.



El robot visita el cubículo H-264.



Y por último se dirige a visitar el cubículo H-262.



4.6 Documentación

4.6.1 Crear un elemento gráfico en PECRI

Existen dos tipos de clases cubículo definidos en PECRI, los cubículos rotados y los no rotados. Los cubículos rotados son aquellos que aparecen girados 45 grados en sentido de las manecillas del reloj; los cubículos no rotados carecen de algún tipo de transformación.

Cada clase cubículo posee sus archivos fuente (*.cpp*) y de cabecera (*.h*).

- ***Cubículos rotados (Clase Cubb)***: *cubb.cpp* y *cubb.h*.
- ***Cubículos no rotados (Clase Cuba)***: *cuba.cpp* y *cubb.h*.

Para crear una clase cubículo debemos abrir el archivo cabecera de la aplicación (*graphwidget.h*) y agregar la siguiente línea de código con el atributo *prívate*, ver la figura 37.

[Clase] [*nombre del objeto];

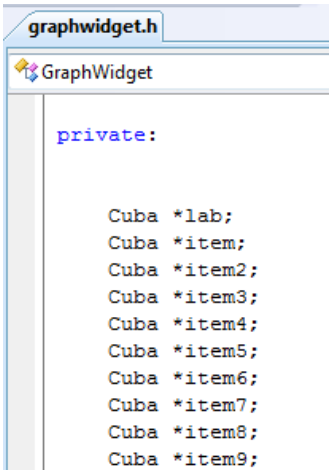


Figura 37. Declarar una clase cubículo

Modificar los parámetros de la clase cubículo. Cada que creamos una clase cubículo es necesario editar los siguientes parámetros:

- ***Tamaño*** (alto y largo).
- ***Posición*** (x,y). La esquina superior izquierda de la escena es el origen.
- ***Zona seleccionable***. Es el área que puede ser seleccionada (con el *mouse*) de cada elemento gráfico (cubículo).
- ***Rótulo del objeto***.

Todos estos datos se encuentran establecidos en el archivo fuente *graphwidget.cpp*, en el método constructor de clase, ver la figura 38.

```
item = new Cuba(0, 0, 50, 30, "H-249");//H-249
item->setPos(11, 521);
scene->addItem(item);
```

Figura 38. Modificar una clase cubículo

Las primeras dos cifras (primera línea) es la definición del área seleccionable; por defecto se deja “cero” para habilitar la selección y “OFF” para deshabilitarla. El par siguiente establece el tamaño del objeto (*pixeles*). Y al final tenemos el rótulo que se desplegará.

La siguiente línea es la posición que ocupa y por último, se agrega el cubículo para su despliegue en pantalla.

A continuación se muestra una tabla con los datos de inicialización de cada clase cubículo, ver la tabla 6.

Tabla 6. Valores de inicialización de las clases cubículo

Cubículo (Rótulo)	Tamaño (Alto x Largo)	Posición (x, y)
Lab Base de Datos	170, 65	11, 381
H-249	50, 30	11, 521
H-250	50, 30	11, 491
H-251	50, 30	11, 461
H-252	50, 30	11, 431
H-298	30, 30	91, 461
H-297	30, 30	121, 461
H-296	30, 30	151, 461
H-246	30, 30	91, 491
H-245	30, 30	121, 491
H-244	30, 30	151, 491
H-243	30, 30	121, 551
H-295	30, 60	181, 461
H-294	30, 60	211, 461
H-293	30, 60	241, 461
H-292	30, 60	271, 461
H-291	30, 60	301, 461
H-290	30, 60	331, 461
H-258	30, 50	181, 381
H-259B	30, 50	211, 381
H-259	30, 50	241, 391
H-261	30, 50	301, 381
H-262	30, 50	331, 381
H-262B	30, 50	361, 381
H-263	30, 50	391, 381
H-263B	30, 50	421, 381
H-264	30, 50	451, 381
H-265	30, 50	481, 381

Cubículo (Rótulo)	Tamaño (Alto x Largo)	Posición (x, y)
H-288	40, 30	391, 461
H-287	40, 30	391, 461
H-286	40, 30	471, 461
H-288B	40, 30	391, 491
H-287B	40, 30	461, 491
LAB	40, 30	471, 491
HP-11	60, 50	212, 222
HP-10	60, 50	251, 183
HP-09	60, 50	290, 144
HP-08	60, 50	290, 300
HP-07	60, 50	329, 261
HP-06	60, 50	368, 222
ARCHIVO	60, 50	407, 183
HP-03	60, 50	598, 128
HP-02	60, 50	632, 162
HP-01	60, 50	666, 196
COMPUTO	60, 50	548, 246
IMPRESIÓN	60, 50	514, 212
AREA SECRETARIAL	60, 50	446, 127
JEFE DEPTO	60, 100	509, 38
ASISTENTE/ADMIN/DEPTO	60, 50	368, 66
JEFE|SERVER	60, 50	480, 178
ESCALERAS	60, 50	564, 94
SALA DE JUNTAS	119, 80	410, 24
AYUDANTES	119, 49	583, 281

4.6.2 Definir el orden de visita

La comunicación entre los componentes de PECRI, se hace mediante un documento "*txt*". El archivo guarda el orden de visita de cada cubículo. Es importante definir la ruta en ambas aplicaciones.

Nota: Se tiene que especificar una ruta válida y sin espacios, de lo contrario la aplicación no va a iniciar.

- ***graphwidget.cpp*** en el método *execute()*, como se muestra en la figura 39.

```
void GraphWidget::execute(){
    pf = NULL;
    pf=fopen("C:/Users/Fede/Documents/dat.txt", "w");
```

Figura 39. Escritura del archivo que contiene el orden de visita

- ***robot.cpp*** en el *main()*. Ver la figura 40.

```
if((pf=fopen("C:/Users/Fede/Documents/dat.txt", "r")) == NULL){
        perror("El fichero no se puede abrir \n");
        return -1;
}
```

Figura 40. Lectura del archivo que contiene el orden de visita

Nota: La ruta debe de coincidir en ambas partes.

Para establecer el orden de visita es necesario editar el archivo fuente *graphwidget.cpp*, en el método *execute()*, ver la figura 41. El orden de visita se asigna de manera descendente e involucra escribir la siguiente instrucción:

print([Objeto cubículo]->activate());

```
print(item10->activate());//H-244
print(item9->activate());//H-245
print(item8->activate());//H-246
print(item->activate());//H-249
print(item2->activate());//H-250
print(item3->activate());//H-251
print(item4->activate());//H-252
print(item5->activate());//H-298
```

Figura 41. Asignación del orden de visita

4.6.3 Ejecución del recorrido del robot

Para comunicar los componentes de PECRI, debemos establecer la ruta donde se encuentra el ejecutable que contiene el recorrido del robot. Abrimos el archivo *graphwidget.cpp,* dentro del método *execute()* escribimos lo siguiente:

system("…/recorrido.exe");

Nota: Debe de ser la última instrucción en el método. La ruta no debe contener espacios o no funcionará la aplicación.

5 CONCLUSIONES

De acuerdo a los resultados obtenidos se concluyó que el proyecto fue un éxito, el uso de los sonares para detectar los obstáculos que presenta el Departamento de Sistemas, fue la clave para visitar los cubículos y también para mantener al robot lejos de los muros. Pero aún así, pienso que no servirán para recorridos más complicados por su limitado rango de operación (70 centímetros), si el robot lleva mucha velocidad no alcanza a librar los obstáculos.

Otro punto que me gustaría comentar, es sobre el servidor de software. Se desperdicia el poder de procesamiento de la computadora que va a bordo del robot; sería muy útil que guardara los datos que recoge el robot en tiempo de ejecución y que los comparara con los de otros recorridos, con la finalidad de prevenir errores. Si asociamos este punto con el anterior, es posible que el almacenamiento de datos del recorrido pueda solventar las limitaciones que presentan los sonares.

Cabe mencionar que programar usando ARIA fue de las labores que más tiempo consumió durante el desarrollo de los experimentos. Algunos de los comandos no funcionaron y se tuvo que consultar la documentación de la API, en busca de algún indicio sobre los errores que se cometieron a la hora de implementarse. Al final, las instrucciones no se pudieron hacer funcionar. Sin embargo, ARIA tiene licencia GPL y es posible modificar el código fuente, con la finalidad de arreglar los comandos que no funcionan o crear clases que agreguen nuevas características al robot. Por ejemplo, compatibilidad con otros periféricos o lenguajes de programación (C#, Python, etc).

Por último, la línea de investigación que sigue el robot *Pioneer* 3-AT nos brinda un abanico de posibilidades, a continuación menciono algunas de ellas:

- Hacer que el robot pueda tocar la puerta de los cubículos, determinar si el profesor se encuentra en su cubículo, llevar objetos de papelería, decir el nombre del profesor y lo que le va a entregar.
- Agregar al robot la capacidad de entrar y salir de los cubículos.
- Configurar al robot para que la aplicación PECRI pueda ser controlada desde un teléfono celular.
- Extender el recorrido del robot a los departamentos vecinos para entregar documentos importantes.
- Programar el robot para que pueda evitar obstáculos en movimiento.

6 BIBLIOGRAFÍA

[1] *Mobile Robots* [En línea]. Disponible

http://www.mobilerobots.com/Mobile_Robots.aspx

[2] Ríos J., "*Puesta en marcha del Robot P3-AT*", Proyecto Terminal de Ingeniería Electrónica, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2008.

[3] Escamilla J. y García L., "*Programación de una visita guiada al Departamento de Sistemas por medio del Robot P3-AT*", Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2010.

[4] Mobile Robots Inc, *Pioneer 3 Operations Manual*. Vermont, 2006.

[5] *ARIA - MobileRobots Research and Academic Customer Support*. [En línea]. Disponible

http://robots.mobilerobots.com/wiki/ARIA

[6] *MobileSim – Documentation*. [En línea]. Disponible

http://robots.mobilerobots.com/MobileSim/README.html

[7] *Pioneer SDK – Mapper 3 Basic.* [En línea]. Disponible

http://www.mobilerobots.com/researchrobots/PioneerSDK/Mapper3.aspx

[8] Blanchette, J., Summerfile, M. "C*++ GUI Programming with Qt 4*", 2nd Ed, New Jersey, Prentice-Hall, 2008, pp. 14-16.

[9] *Graphics View Framwork*. [En línea]. Disponible

http://doc.qt.nokia.com/latest/graphicsview.html

[10] *Microsoft Windows*. [En línea]. Disponible

http://windows.microsoft.com/es-XL/windows/home

[11] *Microsoft Visual Studio 2008 Professional*. [En línea]. Disponible

http://www.microsoft.com/spain/visualstudio/products/2008-editions

[12] *Qt for Open Source C++ development on Windows* [En línea]. Disponible

http://qt.nokia.com/downloads/windows-cpp-vs2008

[13] *Visual Studio Add-in*. [En línea]. Disponible

http://qt.nokia.com/downloads/visual-studio-add-in

[14] *ARIA Documentation*. [En línea]. Disponible

http://robots.mobilerobots.com/ARIA/README.txt

[15] *ArRobot Class Reference*. [En línea]. Disponible

http://www.ai.rug.nl/vakinformatie/pas/content/Aria-manual/classArRobot.html

[16] *ArActionAvoidFront Class Reference* [En línea]. Disponible

http://www.ai.rug.nl/vakinformatie/pas/content/Aria-manual/classArActionAvoidFront.html