COMPUTADORES III

VIRTUAL CYBERTECH: SYSTEM REFERENCE MANUAL

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS INDUSTRIALES

EQUIPO YELLOW:

Marta Dorado (Project Manager), Álvaro López, Antonio Díez, Carlos Sampedro, Francisco Suárez, Marie Destarac y Ricardo Espinoza.

Índice

0. Seguimiento del documento

Histórico de modificaciones

Versión Número	Fecha <<mm/dd/aa>>	Cambios	Autor	Revisor	Secciones afectadas
1	06/11/12	Creación del documento	Equipo Yellow	Guadalupe Sánchez	Todas
2	06/23/12	Actualización y revisión del documento	Equipo Yellow	Guadalupe Sánchez	Todas

1. Introducción

1.1. Propósito

Este manual de referencia describe los detalles de la implementación del simulador *Virtual Cybertech*, en adelante VCT, que tiene como función simular un entorno para la competición *Cybertech* que se realiza anualmente entre grupos de distintas universidades y es organizada por la División de Ingeniería de Sistemas y Automática, y la asociación de estudiantes Reset de la Escuela Superior de Ingenieros Industriales de la Universidad Politécnica de Madrid.

El propósito de este manual es dar a los programadores las bases necesarias para realizar futuras modificaciones, mejoras y optimizaciones del sistema.

1.2. Alcance del sistema

El sistema VCT provee un entorno virtual donde los participantes del concurso pueden llevar a cabo simulaciones en tiempo real de los códigos desarrollados para su robot. El entorno tiene las mismas características y limitaciones reales del concurso.

Las características principales del sistema se presentan a continuación:

- El sistema provee los recursos necesarios para realizar la competencia CyberTech 2012 en un entorno virtual.
- El entorno de simulación se basa en el Virtual Robot Experimental Plataform (V-REP)
- El robot para el que se realiza la simulación es el Kephera III.
- El ambiente de integración se realizó con la herramienta de IBM, Rational Software Architect.

Lo que se pretende es que los participantes sean capaces de probar los algoritmos de control que implementarán en sus robots incluso antes de la construcción de los mismos, lo que sugiere una gran ventaja, ya que la experiencia en los eventos previos indica que la mayor parte de tiempo que requiere el concurso se invierte en la construcción del hardware.

El alcance de esta primera versión se centra en el desarrollo del entorno de simulación que tiene las siguientes funcionalidades:

- Permite al usuario la elección y carga entre un conjunto de laberintos previamente configurados.
- Provee al usuario la opción de construir diferentes laberintos y guardarlos.
- El usuario puede cargar, modificar, guardar, borrar y cerrar los laberintos que ha creado.
- El usuario puede simular el robot con el código insertado sobre el laberinto previamente cargado.
- El robot se moverá dentro del laberinto elegido en el entorno comenzando en el punto de inicio que indique el usuario. Por defecto, existe un punto inicio.
- La interfaz de usuario posee botón de comienzo, pausa y paro de la simulación. Además, el usuario puede variar la velocidad y la cámara de la simulación.
- El usuario podrá visualizar la posición y velocidad del robot.
- Se le notifica al usuario los diferentes errores que pueden haber, como elegir un punto fuera del laberinto para colocar al robot.

1.3. Visión general del documento

En el apartado 2 se describe de manera general la arquitectura del sistema, se explican los diferentes módulos que lo componen, la función de cada uno y cómo se interconectan entre sí.

En el apartado 3 se describe el flujo de información del sistema, por ejemplo los métodos usados para guardar un laberinto creado por el usuario o una simulación.

El apartado 4 describe detalladamente la implementación de los diferentes módulos que componen el simulador, especificando su función, el proceso que realiza, los datos que utiliza y las dependencias con otros módulos.

La implementación de la interfaz gráfica se trata a detalle en el apartado 5 y en el 6 se explica el software y hardware requerido para construir y usar el simulador, así como el proceso de construcción y ejecución de éste.

1.4. Referencias

Se tomaron como base los siguientes documentos:

- Requisitos iniciales VCT_V0.1.docx con fecha 26/03/12.
- Especificación de requisitos VCT_V0.2.docx con fecha 05/07/12.
- System design description VCT_V0.1.docx con fecha 05/07/12.
- Component Approach for Real-time and Embedded. Documento D2.2-1, version 3.0, con fecha 21/02/07.

1.5. Definiciones y acrónimos

Con la finalidad de mantener claro en todo momento los términos que se usan en este documento, se indican a continuación todas las definiciones, acrónimos y abreviaturas utilizadas:

Definiciones

- Virtual Robot Experimental Plataform: es un simulador de robots en tres dimensiones basado en una arquitectura de control distribuido, lo que quiere decir que los programas de control (o scripts) pueden ser enlazados directamente a los objetos de la escena y ejecutarlos. Permite editar, simular, probar y evaluar sistemas robóticos enteros o sub-sistemas (como sensores, mecanismos, etc.). V-REP puede usarse como una aplicación independiente o puede integrarse de forma sencilla a una aplicación realizada por el cliente.
- QT SDK: software que provee herramientas para simplificar la creación de aplicaciones para teléfonos móviles y plataformas de escritorio. Incluye herramientas de diseño de interfaz de usuario.
- Rational Software Architect: es un entorno de modelado y desarrollo que usa el Lenguaje de Modelado Unificado (UML) para el diseño de una arquitectura en C++ y Java2. Es una plataforma de software de código abierto e incluye la capacidad de construir una arquitectura basada en modelos (Model Driven Development, MDD) con el UML para crear aplicaciones flexibles.
- Doxygen: es una herramienta que convierte los comentarios de C++ en un archivo tipo html. Extrae los comentarios directamente del código fuente y los presenta de forma ordenada y estructurada para su fácil comprensión.

Abreviaturas

- Virtual CyberTech: VCT
- Virtual Robot Experimental Plataform: V-REP
- Rational Software Architect: RSA.
- Kephera III: KIII

2. Arquitectura del sistema

2.1. Descripción general de la arquitectura

El simulador desarrollado se compone de dos grandes módulos, que son el V-REP y la aplicación en QT, los cuales se muestran en la figura 1.

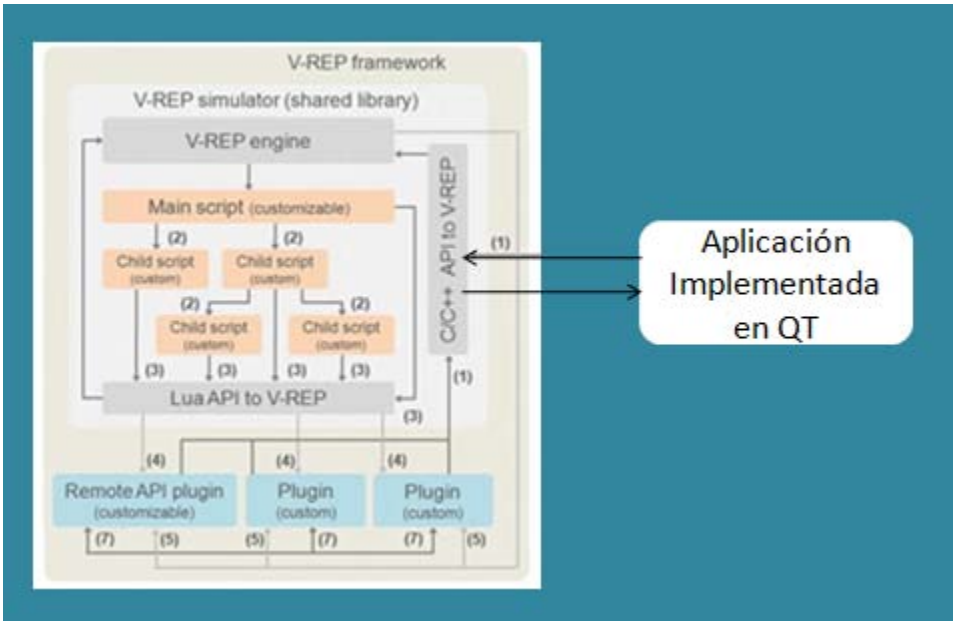

Figura 1. Arquitectura general del sistema

El módulo de la aplicación en QT se encarga de la interfaz gráfica de usuario, la cual controla varios subsistemas encargados de realizar las tareas seleccionadas por el usuario, como guardar una simulación, así como comprobar errores o desplegar gráficos derivados de la simulación. Dentro de este módulo se gestiona también el almacenamiento de los datos que maneja el sistema.

El módulo de V-REP es el encargado de realizar la simulación requerida por el usuario, usando para ello los diferentes API's que posee esta plataforma, que representan en este caso los subsistemas de dicho módulo. Para llevar a cabo la simulación, el V-REP requiere el código ingresado por el usuario, el laberinto seleccionado y el punto de inicio del robot dentro del laberinto. Toda esta información la dará el usuario y la recoge el módulo de aplicación en QT, que a su vez la transfiere al V-REP. Ambos sistemas son compatibles y por ello la arquitectura logra simplificarse.

2.2. Descripción de los subsistemas

En la figura 2 se muestran gráficamente los diferentes subsistemas que conforman el simulador y la interacción que hay entre ellos. A continuación se describe su función:

- Subsistema de Comunicación: los dos módulos del sistema se comunican a través de este subsistema, que lleva a cabo la labor de interpretar las peticiones del usuario y traducirlas a los comandos propios del V-REP. Los datos que comunica entre ambos módulos son los relacionados a la simulación, como el laberinto seleccionado y el código ingresado.

- Subsistema Interfaz: la interacción con el usuario queda a cargo de este subsistema, que recoge las elecciones que hace el usuario y despliega las gráficas y los mensajes pertinentes, como el aviso de errores, por ejemplo.

- Subsistema Comprobación de errores: lleva a cabo la detección de los diferentes errores y se lo comunica al subsistema Interfaz.

- Subsistema gráfico: encargado del despliegue de las diferentes gráficas derivadas de la simulación, como posición y velocidad del robot.

- Subsistema Almacenamiento de Datos: cuando el usuario selecciona almacenar un laberinto o una simulación, es el subsistema Almacenamiento de datos el encargado de gestionar estas acciones. Los datos de la simulación se guardan como un vector de datos y se salvan en tres diferentes formatos: .txt, .csv y .mat

- Archivos guardados: podrán ser mostrados y accedidos por el usuario a través del subsistema Interfaz, como se muestra en la figura 2.

- Subsistemas del V-REP: lo conforman los diferentes API's y partes de la propia plataforma. Para mayor referencia de la arquitectura del V-REP, se sugiere consultar el Manual de Usuario de éste.

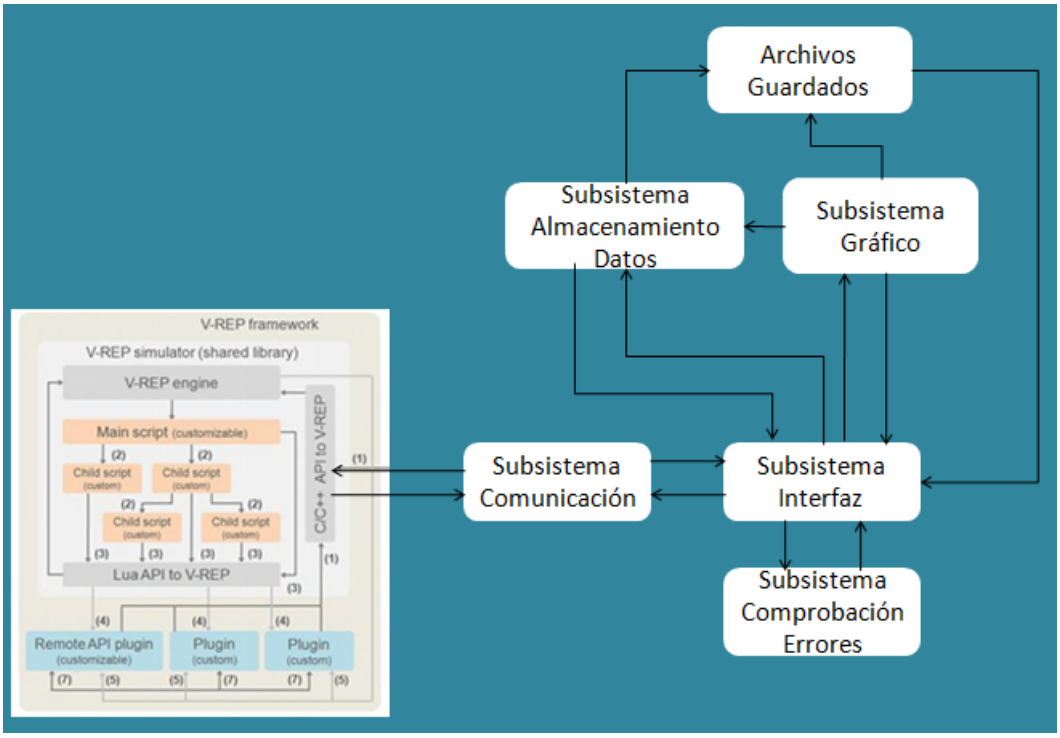

Figura 2. Subsistemas que conforman la aplicación realizada en QT

2.3. COTS

Los subsistemas que se han incorporado como COTS (Commercial off the shelf) son los siguientes:

- **V-REP**: se hace uso de aquellos recursos de esta plataforma que son de utilidad para los fines del sistema, como el modelo del robot y la capacidad de simulación del comportamiento del robot con el código y laberinto seleccionados por el usuario.

- **QT**: debido que este software provee herramientas poderosas y fáciles de usar para la creación de aplicaciones, se hace uso de éste para la implementación de la interfaz de usuario del simulador. Además, se usó también la siguiente herramienta:

    - **QCustomPlot**: es una clase de QT para representar gráficamente los datos de la simulación como posición y velocidad de robot, y se utiliza dentro del subsistema Gráfico.

- **DoxyGen:** es el archivo HTML que genera Doxygen con los comentarios del código realizado en C++ y que ayuda a documentar el sistema.

En la figura 3 se presenta un esquema de cómo se han integrado los COTS en el sistema, pudiendo notarse que doxygen únicamente recibe datos, que son los comentarios del código realizado en C++.

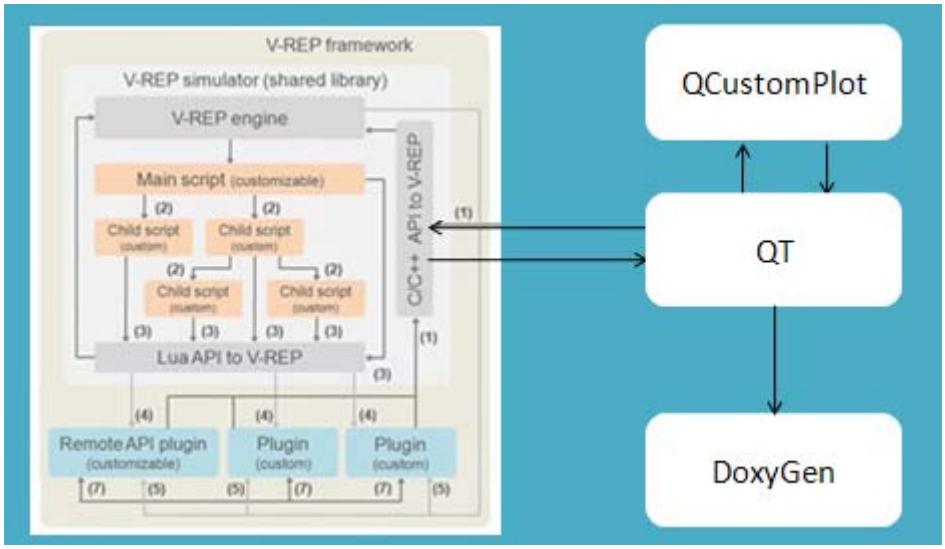

Figura 3. Subsistemas incorporados como COTS

2.4. Integración

En este apartado describiremos lo que tendremos en cuenta para poder realizar la integración entre los diferentes cots:

- **V-REP:** Para la ejecución del simulador no será necesario tener instalada una versión de V-REP, pero será necesaria la introducción de sus librerías. Esto es realizado por instalador de manera automática. Debido el simulador depende de V-REP si se cierra o finaliza este proceso el simulador fallara dando diferentes errores. Para la interconexión se utiliza la API de V-REP (http://www.v-rep.eu/helpFiles/en/apisOverview.htm). Los modelos predefinidos como el del robot o el de las paredes se deben ubicar en el directorio /models.

- **QtSDK:** Para la ejecución del simulador no será necesario tener QtSDK instalado así como tampoco son necesarias las librerías ya que el ejecutable se ha generado de manera estática (static linking). Esta herramienta nos ha dotado varias clases necesarias para la creación del simulador, principalmente la interfaz de usuario, además cabe destacar la clase Qcustomplot utilizada en la creación de las graficas. No es necesario ningún tratamiento especial aparte de la inclusión de las librerías.

- **DoxyGen**: Utilizado para la generación de documentación, no necesita ningún programa especifico para la visualización de esta documentación, será necesario

utilizar una pequeña formula clave para la introducción de los comentarios deseados en la documentación generada por doxyfile.

3. Datos del sistema

3.1. Descripción de los datos y del flujo de datos

El simulador desarrollado maneja diferentes datos proporcionados por el usuario o seleccionados por éste. Para facilitar la comprensión del flujo de datos dentro del sistema, se muestra en la figura 4 los diferentes subsistemas y cómo fluyen los datos entre ellos. Además, a continuación se describe la información que maneja cada clase:

- **Wall:** Encargado de administrar los datos (coordenadas) que conforman cada una de las paredes del laberinto. Estas paredes se enviarán a Maze y serán mostradas en el simulador.
- **Maze:** Recibe los datos que conforman el laberinto (paredes) y se lo devuelve a simulador si no existen errores.
- **Robot:** El dato que maneja es el código del robot.
- **Simulator:** Recibe los datos de robot y de laberinto, además se comunica con el detector de errores de laberinto para corroborar que no existen incongruencias.
- **Writer:** Guarda el resultado de la simulación de forma que el usuario sea capaz de tratar esos datos. El formato de almacenamiento puede ser de3 maneras:
    - **TXT**: Se crean tantas líneas como variables se vayan a guardar, el nombre de la variable se coloca al principio de la línea y los distintos valores se van a guardar uno detrás del otro hasta que se acaben, el salto de línea indica el final de dicho vector.
    - **CSV**: similar al archivo .txt pero con un formato que permite al programa EXCEL introducir cada dato en una celda y las distintas variables en filas.
    - **MAT**: Se crea un fichero con tantas variables como se quieran guardar. Estas variables serán vectores cuyo tamaño concuerda con el número de datos que se desea guardar. De esta forma se tendrán en un fichero AAA.mat, en el que estarán guardadas *n* variables de longitud m.
- **Qt GUI:** Esta en continua comunicación con el simulador. A través de ella el usuario modifica los datos que se manejan.
- **GraphsWindow:** Recibe del simulador la posición y velocidad del robot y los plotea. Estos datos son expuestos al usuario a través de la GUI, y pueden ser

capturadas en formato .png o .pdf. Este subsistema está basado en la clase QCustomPlot (clase de Qt, que hereda de la clase QWidget). Los datos que recibe son:

- **QVector<QVector<float> > &v**: Este parámetro es una matriz con tipo de datos "float", en el que en cada fila se almacena el tipo de parámetro a representar: posición x, posición y, velocidad x, velocidad y; y en las columnas se van almacenando los valores numéricos correspondientes a cada parámetro e instante de simulación.

- **QList<QString> &l**: Este parámetro contendrá una lista con tipo de datos "string", que será empleada para obtener el índice o fila del array de QVectors, correspondiente a cada parámetro de simulación (posición x/y, velocidad x/y); lo que nos permitirá indexar la matriz de floats para obtener los datos numéricos de velocidad y posición del robot, y poder graficarlos. Como es lógico, el índice que representa el string correspondiente dentro de la lista de strings coincide con el índice (fila) que ocupa ese parámetro en el array de QVectors.

- **CodeEditor:** Es la clase encargada de gestionar el editor de código. Cuenta con enfatización de sintaxis con lo cual se facilita la tarea de escribir los programas. Cada vez que el usuario modifica el programa, el simulador lo guarda y lo carga automáticamente en el robot.

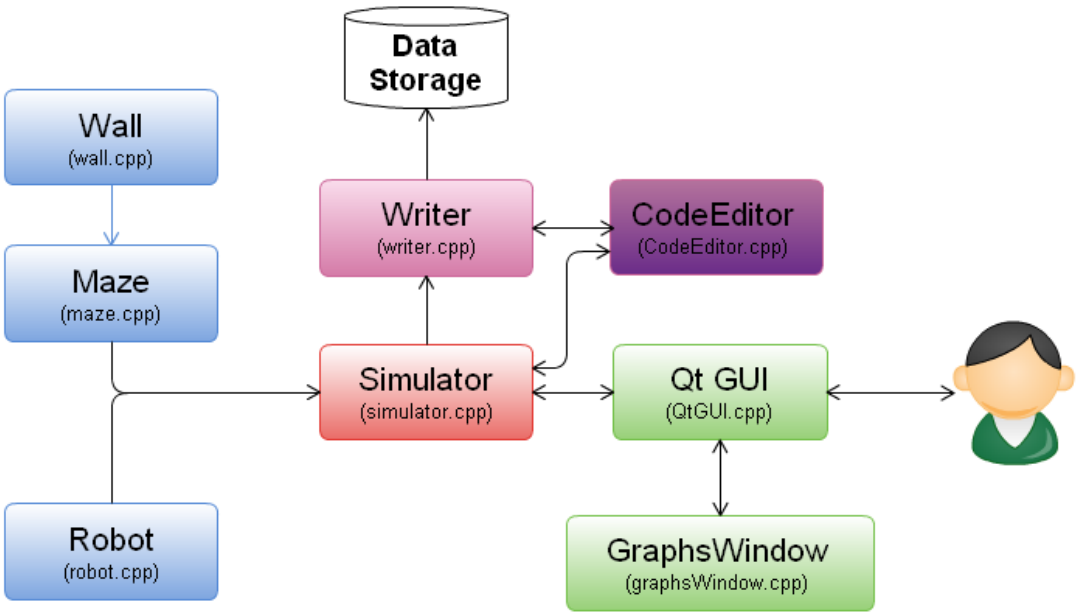

Figura 4. Flujo de datos en el sistema

4. Implementación de componentes

En este apartado se describe lo que hace cada componente del sistema, siguiendo el esquema mostrado en el apartado 2.

4.1. Subsistema Interfaz

La función de este componente es la interacción con el usuario. Recoge las elecciones que hace el usuario y despliega las gráficas y los mensajes pertinentes, como el aviso de errores, por ejemplo. Los datos que usa el subsistema son el código ingresado por el usuario y las elecciones que éste haga de la simulación, por ejemplo, acciones sobre la cámara, creación de un laberinto nuevo y el posicionamiento del robot sobre éste.

El componente interacciona con prácticamente todos los demás subsistemas, como puede verse en la figura 2.

4.2. Subsistema Comunicación

Este componente lleva a cabo la labor de interpretar las peticiones del usuario y traducirlas a los comandos propios del V-REP. Los datos que comunica entre ambos módulos son los relacionados a la simulación, como el laberinto seleccionado y el código ingresado. El flujo de datos es bidireccional entre el subsistema de Interfaz y el de V-REP.

4.3. Subsistema V-REP

Este componente está conformado por los diferentes API's y partes de la propia plataforma. Su función es realizar la simulación del comportamiento del robot ante el código ingresado y el laberinto seleccionado, incluyendo la lectura de los sensores del KIII en tiempo real.

Los datos que requiere para realizar su función son los que recoge del subsistema Interfaz, por lo que tiene dependencia directa con este subsistema y el de Comunicación. Así mismo, tiene dependencia indirecta con el subsistema Gráfico, ya que el V-REP genera los datos de posición y velocidad que permiten realizar las gráficas.

4.4. Subsistema Comprobación de errores

Su función es la detección de los diferentes errores que pueda haber en el código ingresado por el usuario. Los datos que requiere es el mismo código y tiene dependencia directa con la interfaz de usuario, siendo bidireccional la comunicación entre ambos subsistemas. Al detectar errores, éstos serán mostrados en la interfaz de usuario.

4.5. Subsistema Gráfico

Su función es el despliegue de las gráficas derivadas de la simulación, como posición y velocidad del robot. Los datos que requiere éste componente se los envía el V-REP y luego éste los procesa para mostrarlos gráficamente cuando el usuario lo considere oportuno.

4.6. Subsistema Almacenamiento de Datos

La función de este componente es guardar un laberinto creado por el usuario o una simulación. Cuando se selecciona guardar una simulación, los datos que quedan almacenados son la posición y la velocidad, y pueden guardarse en tres diferentes formatos: .txt, .csv y .mat.

Este subsistema tiene dependencia directa con el subsistema Interfaz, subsistema Gráfico y con Archivos Guardados.

4.7. Subsistema Archivos Guardados

La función de este subsistema es el mostrar al usuario el listado de archivos guardados para que éste pueda acceder a ellos posteriormente.

Tiene dependencia directa con el subsistema Interfaz.

5. Implementación de la interfaz de usuario

5.1. Visión general de la interfaz de usuario

La interfaz de usuario desarrollada con el software QT de Nokia contiene las funciones necesarias para que el usuario pueda realizar fácilmente la simulación de su código para el KIII. La interfaz de usuario consta de las partes mostradas en la figura 5. Se describe a continuación cada una:

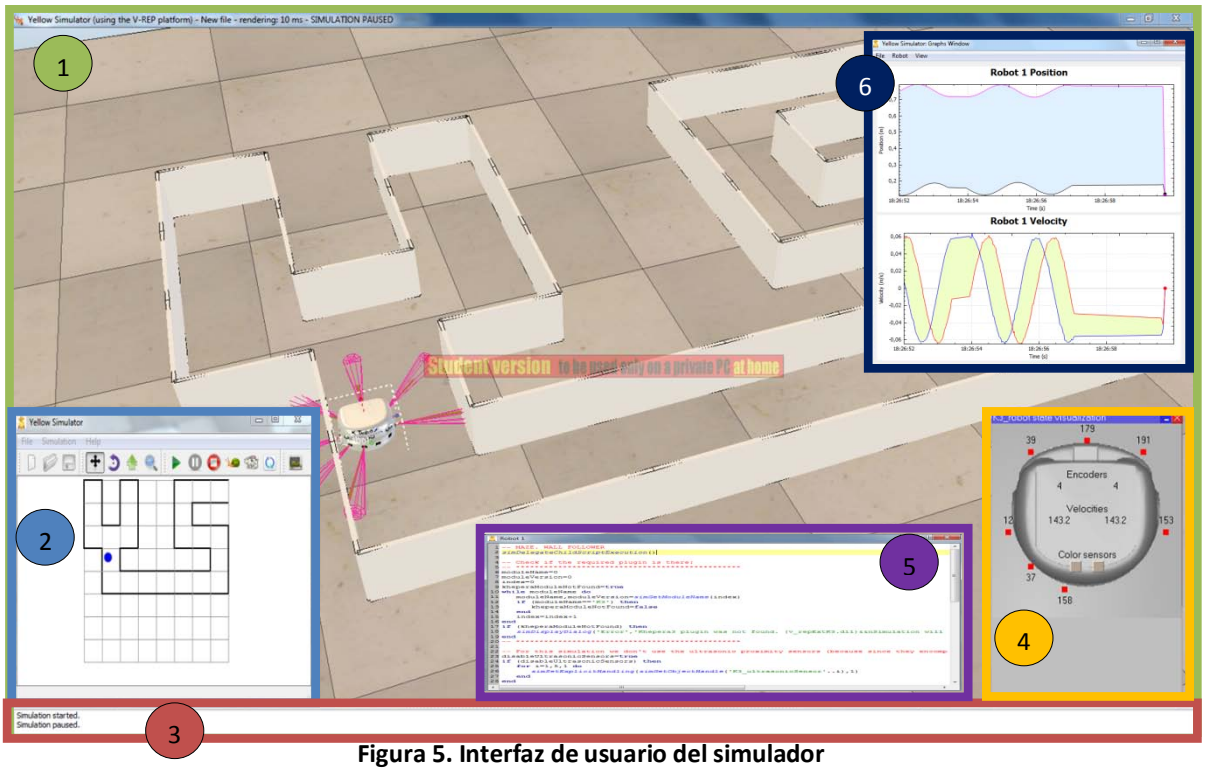

Figura 5. Interfaz de usuario del simulador

Definiremos los elementos integrantes por orden de aparición:

1.-Ventana de simulación: Permite visualizar el laberinto y la interacción del robot con éste durante la prueba del código introducido.

2.-Ventana de controles: Es la ventana principal del software desde la cual se accede a la mayoría de las funcionalidades de la aplicación. En ella es posible controlar la simulación, generar los laberintos, salvar archivos, entre otras funciones.

3.-Barra de estado: En ella se presentan los errores que se produzcan durante la compilación del código, así como el estado de la simulación.

4.-Ventana del estado del robot: En esta ventana se muestra el estado de los sensores del KIII durante la realización de la simulación y esa información se actualiza en tiempo real.

5.-Ventana de código: En esta ventana se introduce y edita el código de control a ejecutar en el robot durante la simulación.

6.-Ventana de información En ella se puede observar gráficamente los valores de velocidad y posición del robot en tiempo real.

5.2. Entidades de la interfaz

La interfaz de usuario se compone de las entidades que se muestran en la figura 6. Cada entidad es una ventana independiente que el usuario puede mover o cerrar y posee una función determinada, que se explica a continuación:

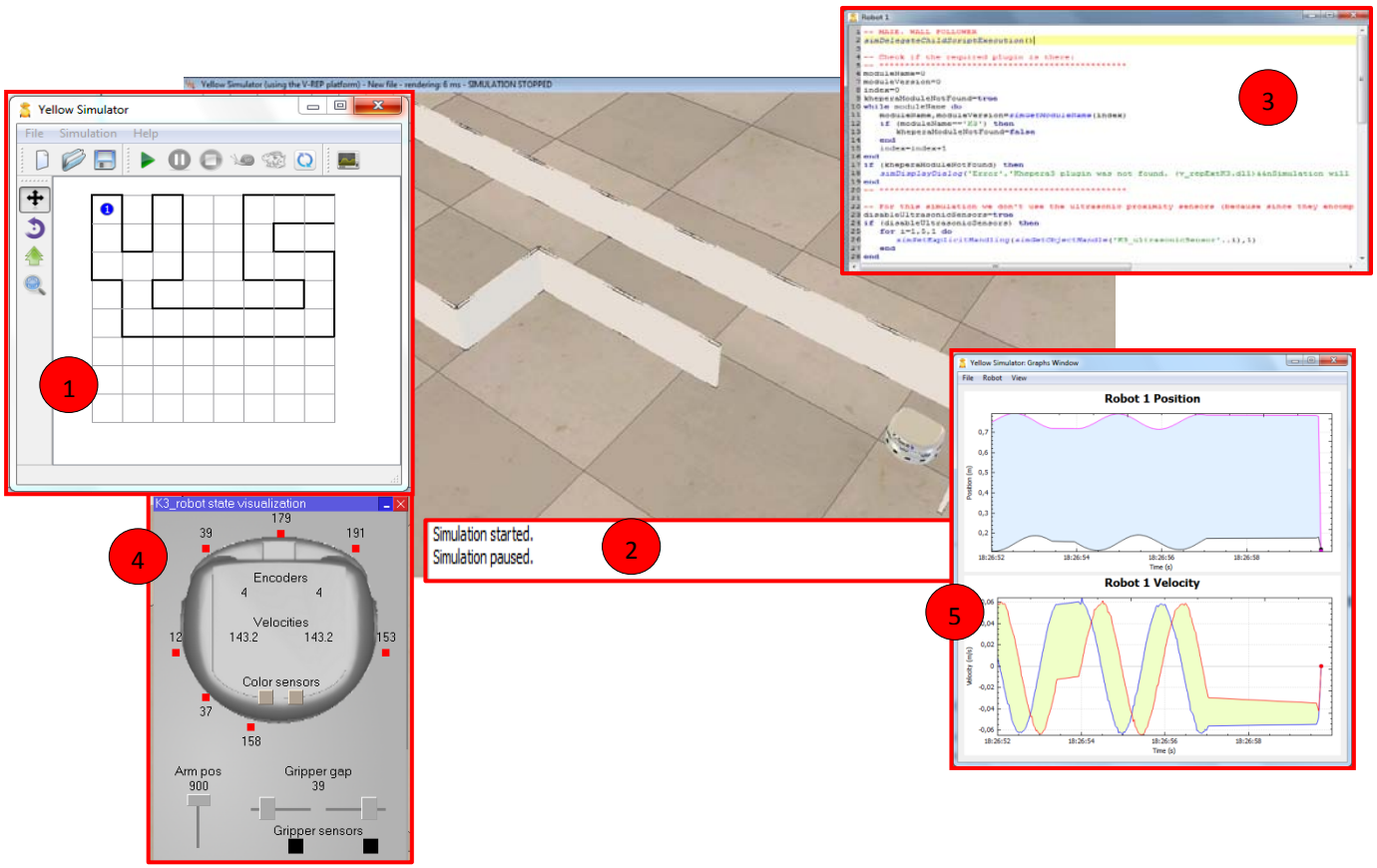

Figura 6. Entidades de la interfaz

1.-Ventana de controles: Es la ventana principal del software desde la cual se accede a la mayoría de las funcionalidades de la aplicación. En ella es posible controlar la simulación, generar los laberintos, salvar archivos, entre otras funciones.

2.-Barra de estado: Muestra los errores que se produzcan durante la compilación del código, así como el estado de la simulación.

3.-Ventana de código: Se introduce y edita el código de control a ejecutar en el robot durante la simulación.

4.-Ventana del estado del robot: Muestra el estado de los sensores del KIII durante la realización de la simulación.

5-Ventana de información: Muestra las gráficas de velocidad y posición del robot en tiempo real.

6. Requisitos de desarrollo

6.1. Requisitos de la plataforma

Requerimientos de Hardware/Software:

- Hardware mínimo requerido:
  CPU: >300 (Valoración CPU)
  RAM: >=512 MB
  Capacidad de almacenamiento: >50 MB
  Tarjeta de vídeo >200 (Valoración G3D)
  Resolución del monitor: >=1024x768
- Software requerido:
  Sistema operativo *Microsoft Windows XP/Vista/7*

6.2. Construcción y ejecución del sistema

La instalación del software VCT se realiza mediante un asistente, que va mostrando ventanas para guiar al usuario durante todo el proceso. Se debe ejecutar el instalador *Yellow Simulator.exe* con permisos de administrador y lo primero que aparecerá, como puede verse en la Figura 7, es una ventana que da la bienvenida al proceso de instalación.

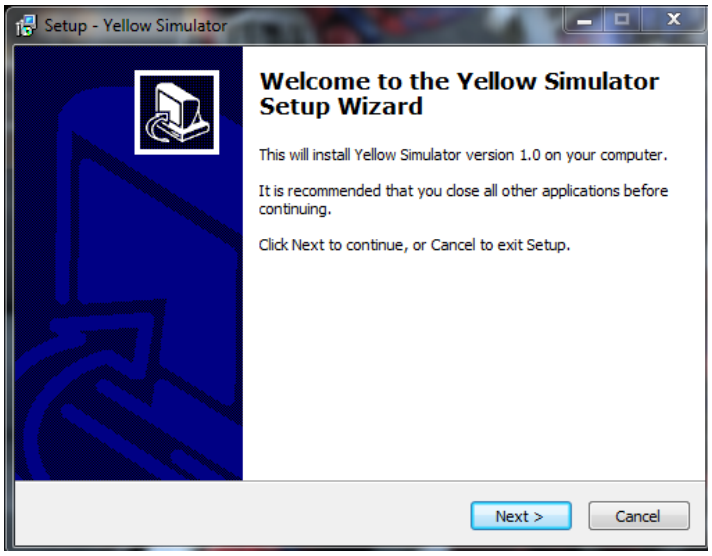

Figura 7. Ventana de instalación

Deben seguirse los pasos que indica el instalador, entre ellos, la selección del directorio donde se instalará la aplicación. Durante el proceso de instalación, el asistente muestra

las acciones que se están llevando a cabo y puede observarse una barra que indica el progreso, como puede apreciarse en la figura 8.

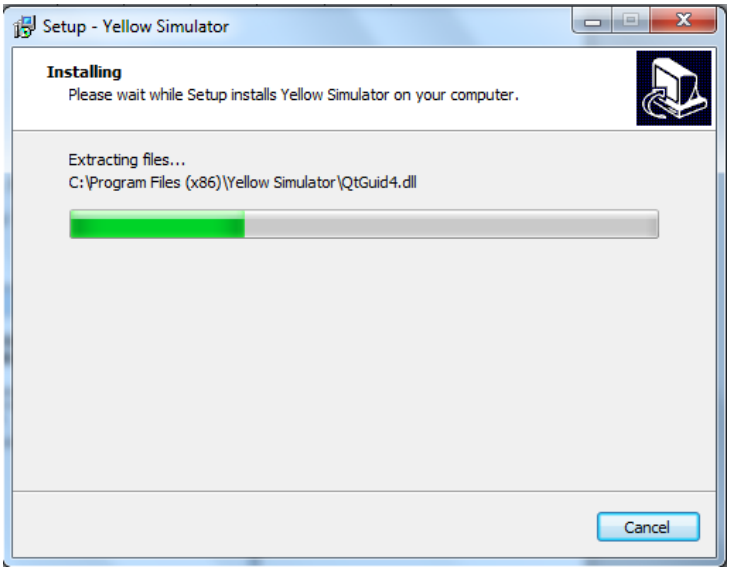

Figura 8. Ventana de progreso de la instalación

Una vez finalizado correctamente todo el proceso, se habrán instalado en el equipo los componentes necesarios para el uso del software VCT, creándose además un acceso directo en el escritorio para acceder de manera rápida y sencilla.