1 Introducción

1.1 Descripción del proyecto

Un simulador es un sistema generalmente informático, que permite reproducir sensaciones que en realidad no están sucediendo. Además, disponen de muchos subsistemas *software* y electrónicos que deben trabajar juntos correctamente. En la mayoría de los simuladores, como los aeroespaciales en este caso, crean un entorno virtual en el que un usuario puede interactuar de diversas formas. Para algunos simuladores es crítica la atención completa por parte de la persona que los utilice, como por ejemplo los simuladores de vuelo; otros en cambio, ofrecen la posibilidad de introducir parámetros para llevar a cabo una simulación preestablecida mientras que muestra al usuario los datos de interés que se van generando.

Todos sin embargo, requieren un gran proceso de calibración y montaje, siendo sistemas muy complejos a la hora de ampliarlos y mantenerlos. Esto hace que la interconexión entre simuladores y otros sistemas posteriores sea una ardua tarea si no inviable, ya sea por el tiempo requerido o económicamente.

Además, al diseñar un simulador, se requiere manejar hardware de diferentes fabricantes, y por necesidades de procesamiento, robustez, etc. se pueden necesitar componentes que a priori no sean compatibles entre ellos, lo que conlleva a realizar interfaces que puedan conectar dichos componentes.

El software desarrollado en este proyecto abstrae al programador de la complejidad que acarrea manipular directamente los equipos de medición, sensores, actuadores, etc.

Mediante el uso de una librería de comunicación desarrollada previamente a este proyecto llamada CL, se consigue enviar los datos por red en tiempo real, por lo que el resultado es un sistema distribuido, áltamente flexible y ampliable mediante la incorporación de nuevos dispositivos. Para ello se crean librerías que encapsulan los drivers de los equipos hardware y que envían y reciben los datos mediante CL, puediendo así crear simuladores y/o aplicativos distribuidos eligiendo únicamente los componentes necesarios y pudiendo compartir la misma fuente de información.

Este proyecto tiene como fin una aplicación gráfica de alto nivel para monitorizar vehículos aéreos no tripulados que se mueven dentro de un recinto cerrado que calcula en todo momento su posición, rotación y velocidad entre otros parámetros, con el objetivo de que un operario pueda utilizar el aplicativo como una estación de tierra pero con la ventaja de ser distribuida, es decir, varios operarios pueden monitorizar en sus respectivos monitores los mismos o diferentes vehículos a la vez, y estar o no presentes en el recinto físico en dónde se lleva a cabo la acción real.

1.2 Motivación

El desarrollo tecnológico se ocupa de la obtención y desarrollo de conocimiento y capacidades cuya meta es la solución de problemas prácticos con ayuda de la técnica. Cuanto más avanzada sea dicha técnica, mayores las capacidades para desarrollar nuevas tecnologías. Este proyecto surge a raíz de la necesidad de interacción de multitud de sistemas completamente distintos entre sí. Entre estos sistemas, se encuentran por ahora robots industriales articulados, UAVs, cuadricóptero, sensores de posicionamiento, y manipuladores hápticos. Con la idea de crear una herramienta avanzada principalmente para el mundo de la simulación. Se persigue la integración total de los distintos sistemas en un único sistema de datos a los cuales acceder fácilmente sin estar sujeto a restricciones de red, protocolos, o equipamiento. Ésta memoria recoge la realización de la primera parte de ésta potente herramienta.

1.3 Alcance del proyecto

La finalidad de este proyecto final de carrera, es el desarrollo de una aplicación de alto nivel que permita la monitorización de sistemas conectados en red, recibiendo datos remotos y representándolos debidamente por pantalla en tiempo real mediante una interfaz gráfica táctil que posibilite la movilidad del usuario mediante su instalación en equipos portátiles. El proyecto se encuentra en el ámbito de la aeronáutica, y está orientado a captar datos de un *testbed* o banco de pruebas consistente en una jaula cerrada por dónde diversos cuadricópteros simulan el vuelo de aeronaves reales.

La aplicación se divide claramente en dos partes: una capa de comunicación que se encarga de recibir datos de la red, y la propia interfaz gráfica que representa dichos datos. El desarrollo se ha llevado a cabo en C++, sobre plataforma linux, y utilizando SCRUM como metodología de gestión de proyectos así como técnicas de programación ágil como TDD y BDD para la obtención de un código limpio, robusto y fiable.

1.4 Recursos utilizados

Para el desarrollo de este proyecto se han utilizado una serie de recursos facilitados por el *Centro Avanzado de Tecnologías Aeroespaciales* (CATEC), entre los cuales cabe destacar:

- Testbed:
  Se denomina con este nombre a un banco de pruebas para vehículos aéreos, en este caso cuadricópteros comerciales. El testbed consta de una zona de seguridad de 12x12x10 metros en la que se pueden mover los vehículos y en todo momento se sabe su ubicación exacta en el espacio gracias a un sistema de cámaras de infrarrojos que captan el movimiento gracias a unos reflectores situados estratégicamente en todos los objetos deseados.
  Este sistema permite probar autopilotos y controladores para vehículos no tripulados, realizar simulaciones a escala de aeronaves reales, o realizar pruebas de carga, etc. Es un recurso valioso no sólo por ofrecer grandes posibilidades sino además por ser uno de los más grandes de Europa.

- Software de distribución de datos
  Para el desarrollo de este proyecto se ha utilizado un software de distribución

de datos basado en el protocolo DDS de la empresa RTI, el cual se presentará debidamente más adelante. RTI (de *Real-Time Innovations*), es uan empresa norteamericana y actualmente líder en el sector de sistemas distribuidos en tiempo real.

- Hardware para desarrollo
  El proyecto se llevó acabo bajo el amparo del departamento de Simulación y Software, teniendo acceso a equipos de alta gama como *workstations* de alta capacidad de procesamiento gracias a su Intel Xeon de 16 núcleos, 12Gb de RAM y gráfica Nvidia GTX480. Esta configuración ha permitido gran agilidad tanto para la compilación de código (ya que es la tarea que más CPU consume para proyectos grandes), como para el diseño gráfico (tarea que requiere de mucha memoria RAM y GPU).
  Para la realización se utilizó una *workstation* para el desarrollo de código, además de otras tres para tareas de integración contínua y desarrollo ágil como se explica en los anexos.

1.5 Estructura del proyecto

Los contenidos de este proyecto se estructuran como a continuación se describen, desarrollándose en cada capítulo los temas que seguidamente se detallan brevemente:

- Entornos virtuales: Primeramente se introducen algunos datos históricos relevantes a los inicios de las interfaces Hombre-Máquina y que se consideran como el punto de partida hacia los sistemas de entornos virtuales. Seguidamente se explica qué es la realidad virtual y los dos enfoques (inmersivo y no inmersivo) de los que dispone. También se habla de realidad aumentada y su estado actual viendo cómo está avanzando a pasos agigantados este sector.
  Para seguir en la línea de la realidad virtual y aumentada, se explican conceptos básicos sobre diseño gráfico ya que son de máxima importancia para realizar correctamente aplicaciones visuales para dichas tecnologías. Así, se detallan algunas técnicas como por ejemplo el uso del color, maquetación y acercamiento al usuario.
  Por último, se introduce el concepto de instrumento virtual como una herramienta

creada a partir de los conceptos anteriores, explicando las ventajas e inconvenientes de estos sistemas virtuales.

- Sistemas en tiempo real: Empieza con una breve definición de los sistemas en tiempo real los cuales se ordenan en dos grupos según su nivel de criticidad. Así, se pasa a definir los sistemas distribuidos y sus características en caso de ser en tiempo real también.
  Esto lleva a explicar el concepto de *middleware* y los conceptos importantes para este proyecto. Además se introduce DDS, que es el *middelware* utilizado, y por qué éste y no otro. Como consecuencia, se explica brevemente la importancia de la librería CL creada para hacer de *wrapper* de DDS y así facilitar la tarea al desarrollador.

- Diseño del sistema: Como bien anuncia el título, se detalla el proceso de creación de las distintas partes que hacen posible el proyecto. Para una mejor comprensión, se han detallado por orden cronológico de desarrollo, haciendo ver así, la necesidad de avanzar por orden ya que todas las partes son eslabones igualmente importantes de una misma cadena.
  Siendo así, la primera atención recae en la capa de comunicación (o librería CL), explicando brevemente su diseño y estructura. Después se trata el servidor de dispositivo (o HALService), que es la aplicación que hace posible la comunicación del dispositivo bajo observación con el resto de la red. El siguiente paso lógico entonces, es detallar cómo es la librería que debe utilizar el desarrollador para acceder a esos datos, resultando así la librería del dispositivo o HALLib.
  Por último, se termina la cadena exponiendo el diseño y estructura de la aplicación de alto nivel que presenta los datos al usuario final. Esta aplicación de alto nivel o HMI (ya que en este caso es una interfaz gráfica), presenta unas capacidades interesantes gracias a la tecnología utilizada para su elaboración, la cual se comenta debidamente así como el instrumental virtual elegido.

- Conclusiones: Habiendo terminado ya el contenido técnico de la memoria, este apartado contiene una breve evaluación del proceso a modo personal, así como una posible evolución a tener en cuenta para mejorar la herramienta creada.

- Anexos: Están formados por dos secciones que explican distintas metodologías

que se utilizaron en el proyecto y que han sido de vital importancia. La primera, hace referencia a Scrum, una metodología de gestión de proyectos que consigue una reducción importante del tiempo de realización de un proyecto gracias a la correcta planificación y subdivisión de tareas.
La segunda, explica las diferentes técnicas de programación que se deben seguir en absolutamente todos y cada uno de los proyecto software que uno pueda tener, ya que garantizan un código limpio, fiable y reutilizable; lo que genera aplicaciones de alta calidad, fáciles de mantener y/o actualizar.