CAPÍTULO 4

Conclusiones y Posibles ampliaciones.

4.1. Conclusiones.-

El software diseñado ha tratado de seguir, en la medida de lo posible, la estructura del código del ROMEO4R(*). Esto se debe a que dicho código es bastante conocido dentro del grupo de Visión, Robótica y Control del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla y, por lo tanto, utilizar la misma estructura simplificará la tarea de los investigadores del grupo. Para facilitar posibles ampliaciones se ha tratado de hacer el programa lo más escalable y modular que fuese posible. De este modo para añadir nuevos elementos software al sistema, es suficiente con la creación de nuevas clases que los gestionen, sin necesidad de cambiar la estructura del resto del código.

---

* Romeo 4R es una plataforma robótica de investigación basada en un vehículo elétrico convencional. Ha sido desarrollado por el grupo de Visión, Robótica y Control en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla. Para más información léase el PFC de Rafael Martín de Agar Tirado que se encuentra en la bibliografía.

Para facilitar la portabilidad del código se ha tratado de seguir la norma POSIX de manera que sea posible migrar la aplicación a un sistema operativo de tiempo real que cumpla dicha norma, como por ejemplo QNX.

Han resultado patentes las dificultades que presenta la ingeniería de sistemas: sincronización de eventos, comunicaciones entre distintos dispositivos, etc.

Por otro lado, ha resultado notoria la dificultad que presenta la programación de una aplicación de tiempo real. Para hacer frente a dichas dificultades siempre resulta preferible conocer el código del sistema operativo que estás usando: es aconsejable utilizar código abierto. Esta es una de las razones por las que se ha decidido utilizar el sistema operativo Linux. Otra de las razones por las que se ha usado Linux, Debian más concretamente, es por la amplia comunidad que soporta a esta distribución: los problemas que van surgiendo se pueden resolver investigando en los foros de internet. Por último, Linux ofrece ventajas frente a otros sistemas operativos en materia de seguridad, robustez y estabilidad, características deseables todas ellas para un sistema de control.

Desde el punto de vista formativo, la realización de este proyecto fin de carrera ha supuesto una experiencia enormemente enriquecedora: aprendizaje de un sistema operativo relativamente poco conocido pero muy potente como es Linux, perfeccionamiento de los conocimientos sobre programación orientada a objetos (C++) y adquisición de una metodología de trabajo que permita resolver cualquier problema, ya sea de software o de hardware.

4.2. Posibles ampliaciones.-

Los programas realizados tienen un comportamiento de tiempo real muy bueno. Sin embargo existen ciertas partes del programa que se podrían mejorar pero se han dejado como están por falta de tiempo o de recursos. Los recursos de los que hablo estarán disponibles en un futuro y sería interesante utilizarlos. Las partes del programa que se pueden mejorar son:

- Temporizadores.-

  Los temporizadores utilizados son temporizadores software de Linux que tienen una precisión de 10 ms. Esta precisión normalmente no es adecuada para los sistemas de tiempo real que suelen requerir una temporización más estricta. El comportamiento del programa con estos temporizadores es aceptable pero sería interesante utilizar temporizadores de alta resolución que cumplan la norma POSIX con lo que se conseguiría una resolución del orden de la decena de microsegundo. Estos temporizadores están incluidos en una librería llamada *"hrtimers"* que se encuentra en desarrollo para el núcleo 2.6.5 en el momento en que escribo esta memoria (existen parches para núcleos inferiores al 2.6.0-test5 pero estos núcleos no dan tan buen resultado como el 2.6.5 por lo que no merece la pena utilizarlos).

- Utilizar nuevas versiones del núcleo de Linux a medida que vayan saliendo.-

  El núcleo 2.6.5 tiene algunas características que lo hacen adecuado para el tiempo real, pero no convierte a Linux en un sistema operativo de tiempo real. Se dice que el núcleo 2.8 será un núcleo de tiempo real. Si es así sería interesante ejecutar la aplicación sobre este nuevo núcleo.

  Por otro lado, las nuevas versiones del núcleo 2.6 introducirán mejoras y corregirán *BUGS* por lo que es aconsejable actualizar el núcleo del sistema operativo a medida que salgan dichas nuevas versiones del núcleo.

- Identificación del modelo del helicóptero.-

  El programa permitirá obtener datos sobre el estado del helicóptero. Estos datos pueden ser procesados por el investigador para obtener un modelo del helicóptero.

- Algoritmos de control.-

  El programa diseñado proporciona una infraestructura de comunicaciones con otros ordenadores (PC de tierra (PST) y PC de percepción y supervisión (PPSE)) y con el CBN. Sin embargo el programa no tiene ninguna funcionalidad específica, es decir, sólo es un esqueleto. Será tarea del investigador o del usuarioel dar una funcionalidad al programa, o lo que es lo mismo, el implementar algoritmos de control de alto nivel.

- Supervisión por parte de tierra y comunicaciones con el PC de percepción y supervisión (PPSE).-

  Los clientes de los Pcs de tierra (PST) y de percepción y supervisión (PPSE) solamente tienen implementado el envío y la recepción de cadenas de caracteres de usuario sin ningún formato. Será tarea del usuarioel dar formato a dichas cadenas de caracteres e interpretarlas de acuerdo a dicho formato para efectuar tareas de supervisión, en el caso del PC de tierra (PST), o de seguimiento de trayectorias, en el caso del PC de percepción y supervisión (PPSE), y de obtención de datos.

- Posibilidad de colocar el código del bucle de control en un proceso distinto para no modificar el código fuente del servidor.

- Utilización del BBCS (Blackboard Communication System) en lugar de TCP/IP para las comunicaciones de red.-

  Para las comunicaciones de red se han utilizado conexiones TCP/IP. El uso de éstas presenta una serie de inconvenientes que se enumeran a continuación:

1) TCP tiene una arquitectura de tipo cliente-servidor que hace necesario decidir si el nodo es un cliente o un servidor TCP. Sería deseable que cada uno de los nodos tuviese la misma importancia para facilitar posibles expansiones. El BBCS tiene una arquitectura descentralizada sobre UDP.
2) Cuando se pierde una conexión, TCP requiere código de usuario en ambos extremos de la conexión para restablecer el enlace. Esto contradice la idea de rutado transparente y automático.
3) TCP es un protocolo fiable que comprueba errores y realiza control de flujo. Esto puede perjudicar el comportamiento de tiempo real de nuestra aplicación al retrasar nuevos datos a causa de que datos más antiguos sufrieron errores de transmisión.
4) Está comprobado que el comportamiento de TCP es malo sobre Ethernet inalámbrica. El esquema de control de flujo implementado normalmente en TCP no es capaz de adaptarse eficientemente a este tipo de conexión.

Por el contrario el BBCS está contruido sobre UDP cuyas características son que no es fiable, no tiene control de flujo y puede ser configurado para ignorar la ausencia del otro extremo de la conexión.

- Conversión del programa supervisor de tierra en una estación de teleoperación gráfica: línea de horizonte artificial, modelos 3D del helicóptero, medida de sonares, etc.