Capítulo 3

ESTRUCTURA DEL SOFTWARE EN LA HÉRCULES.

Para el diseño de una aplicación gráfica de control en el ordenador de tierra se ha de estudiar cuál es la estructura del programa en el computador de vuelo, esto es: qué hace, cómo lo hace y lo más importante para nuestro proyecto, cómo se organiza la interfaz de comunicación con el exterior.

3.1. Funciones principales de la aplicación.

- Inicializar los parámetros básicos de la Hércules y de la UMI.
- Configurar las interrupciones en la placa y otros datos importantes para la ejecución de los experimentos.
- Aplicar una secuencia de excitación sobre los actuadores.
- Iniciar o detener las interrupciones.
- Iniciar o detener la captura de los datos de los sensores.
- Pasar los datos registrados en la cola al disco de la Hércules, en un archivo llamado **datos.dat**.
- Detener la grabación de los datos a disco.
- Enviar el archivo de los datos grabados en el experimento al ordenador de tierra.
- Eliminar el archivo de los datos del experimento.
- Finalizar el programa.

Todas estas funciones menos la primera se realizarán tras una orden del control de tierra. La configuración de datos del experimento se realizará cada vez que se reciba desde tierra una **trama de inicialización**, mientras que el resto de funciones se llevarán a cabo según el valor de una variable global de control, ***n_control***, también enviada desde el ordenador de tierra.

3.2. Estructura general de la aplicación.

Para llevar a cabo las funciones ya comentadas, la aplicación se basa en un sistema de programación concurrente, posible gracias a la utilización de **hilos**, **semáforos** e **interrupciones** de usuario.

3.2.1. Arquitectura del proceso en hilos.

Los hilos de ejecución permiten a una aplicación realizar varias tareas simultáneamente. Los distintos hilos comparten una serie de recursos tales como el espacio de memoria y los archivos abiertos. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones al mismo tiempo.

Así pues, el software del helicóptero se estructura con un hilo principal de ejecución, ***main()*** que genera otros cuatro hilos: tres hilos dedicados a la recepción de datos desde tierra y un hilo encargado de pasar los datos almacenados en la cola a un archivo en disco.

Hilo de ejecución principal.

Main() se encuentra en el archivo **main.c**, y es la encargada de inicializar los parámetros básicos de placa, los de la UMI, crear los hilos y gestionar `'switch (n_control)'` en función de los parámetros que llegan desde la red (*n_control* es una variable global, accesible por todos los hilos y, o bien su valor es puesto a '1' por el código en el hilo de recepción de tramas de inicialización cuando se recibe una trama válida, o bien su valor es recibido directamente desde la red).

En `'switch (n_control)'` se entrará en un bucle que ejecutará cada vez que los hilos de recepción de datos desde red abran un semáforo*, y en función de su valor el programa procederá a ejecutar las funciones oportunas para llevar a cabo lo solicitado por el ordenador de tierra.

Hilos de recepción de datos desde tierra.

Tal y como se detallará un poco más adelante, los envíos helicóptero-tierra se hacen a través de funciones y la recepción tierra-helicóptero a través de hilos, lo que permite que las funciones de recepción de datos puedan quedarse 'colgadas' independientemente sin hacerlo todo el programa, debido a la estructura multitarea introducida por los hilos.

Así pues, tenemos tres hilos dedicados exclusivamente a permanecer a la escucha para recibir datos de distinta naturaleza desde el control de tierra. Esto significa que se quedan bloqueados en la función `recvfrom` en espera de datos desde la red. Cuando éstos lleguen activarán el semáforo para que se ejecute el bucle `switch (n_control)` en la función principal ***main()***, que actuará en función de qué datos hayan llegado (ver figura 3.1).

Detallamos la naturaleza de la información que recibe cada hilo desde tierra:

-Hilo21(): recibe una trama que contiene los valores para inicializar diversos parámetros de las interrupciones de usuario. Además también contendrá otros datos de configuración importantes para llevar a cabo cada experimento, como el tamaño de la cola, qué sensores van a ser introducidos en la cola y cuáles serán enviados en tiempo real a tierra.Tras recibir una trama válida, pone *n_control* a '1' y abre el semáforo.

-Hilo22(): recibe un *n_control* cuyo valor debe estar dentro del rango [2 11]. Tras recibir un valor válido abre el semáforo.

-Hilo23(): recibe una trama de excitación para los distintos actuadores del helicóptero. Tras recibir una trama válida abre el semáforo.

*Un semáforo es una variable especial protegida que constituye el método clásico para restringir o permitir el acceso a recursos compartidos en un entorno de multiprocesamiento (en el que se ejecutarán varios procesos concurrentemente), como es el caso de la estructura en hilos.

3.2.2. Rutas de ejecución en el bucle principal.

Vemos a continuación de forma general las distintas rutas de ejecución que sigue el programa según los datos recibidos desde tierra (*n_control*), es decir, las variables globales que se van modificando y las funciones que se van ejecutando en cada caso (ver figura 3.2).

Destacamos las variables globales de control más importantes para entender el funcionamiento de la aplicación:

Variable global	**Función**
col	Cuando vale '1' los datos muestreados son introducidos en la cola cíclica.
grabar	Cuando vale '1' los datos de la cola son traspasados a un archivo en disco.
ter	Cuando vale '1' es finalizada toda la aplicación.

Función *ini()*: esta función es llamada tras recibir una trama de inicialización. Actualiza la frecuencia de las interrupciones, calcula el tiempo de referencia máximo que se ha de tardar en cada interrupción, contabiliza el número de datos que van a ser introducidos en la cola en cada iteración y por último reserva la memoria necesaria para la cola. Todo ello atendiendo a los parámetros enviados desde tierra.

Función *arranca()*: esta función inicia las interrupciones en la placa, asegurando que en cada tiempo de muestreo sea llamada la función ***sec()***.

Función *sec()*: función que llama a las funciones implicadas en la interrupción (***accion, umi, ad*** y ***env_d_red***) además de contabilizar el tiempo que tardan cada una de ellas en ejecutarse.

Función *accion()*: se encarga de aplicar las acciones de control especificadas desde tierra sobre los actuadores del sistema. Esta función siempre es llamada en primer lugar dentro de cada interrupción para que el control se produzca siempre en el mismo momento.

Función *umi1()*: función que lee los datos de la UMI si así se ha especificado desde tierra. Además, si han sido seleccionados para ser registrados y la variable de control *col*=1, entonces serán introducidos en la cola cíclica, junto con otros dos datos: el número de interrupción y el tiempo en milisegundos transcurrido desde el inicio de la aplicación.

Función *ad1()*: lee en cada iteración el valor del potenciómetro que llega por la entrada analógica, hace la conversión a digital y, si ha sido seleccionado para ello, lo introduce en la cola siempre que *col*=1. En la aplicación de partida no se contempla aún la captura por la entrada digital de los valores del sensor de ultrasonidos.

Función *env_d_red()*: esta función envía en cada interrupción los datos de los sensores que hayan sido solicitados desde el control de tierra en tiempo real.

Función *para()*: función que detiene las interrupciones de usuario en la placa.

Función *env_archivo()*: esta función envía el archivo de datos muestreados en el último experimento realizado al ordenador de tierra.

Función *borrado_archivo()*: borra a petición de tierra el contenido del archivo de datos donde se guardan los datos muestreados de los sensores.

3.2.3. Interfaz de comunicaciones helicóptero-tierra.

Como ya se ha comentado, la comunicación tierra-helicóptero se hace a través de hilos y la comunicación helicóptero-tierra a través de funciones.

FLUJO TIERRA-HELICÓPTERO			
Hilo	**Datos**	**Protocolo**	**Puerto**
hilo_21	Trama de inicio para configuración de un nuevo experimento.	UDP	4951
hilo_22	Variable de control 'n_control'.	UDP	4952
hilo_23	Trama de acción sobre actuadores.	UDP	4953

FLUJO HELICÓPTERO-TIERRA			
Función	**Datos**	**Protocolo**	**Puerto**
env_d_red.c	Datos de los sensores en cada interrupción.	UDP	4950
env_archivo.c	Archivo completo de los datos de sensores muestreados.	TCP/IP	4954
env_errores.c	Número de error producido.	UDP	4955

Vemos que en el caso del envío del archivo de datos muestreados se utiliza el protocolo TCP/IP, mientras que en todos los demás canales de comunicación se utiliza el protocolo UDP. El protocolo UDP es **no orientado a conexión**, lo cual quiere decir que, si bien se garantiza que los datos que llegan son correctos, no se garantiza que lleguen todos. El programa del computador de abordo y el de tierra no han de estar conectados entre sí, pudiendo cualquiera de ellos transmitir datos en cualquier momento independientemente de que el otro esté 'escuchando' o no. Este protocolo es utilizado cuando lo más importante es que ninguna de las dos aplicaciones se quede bloqueada en un envío o recepción. Así pues, en el caso de los hilos no importa demasiado que una trama u orden no sea recibida por el helicóptero, ya que siempre es posible reenviarla.

En el caso del envío de los datos muestreados en tiempo real, la operación se realiza a una frecuencia que oscila entre 1Hz y 1KHz, esto supone entre 1 y 1000 muestras por segundo. El objetivo de esto es la monitorización del estado del sistema desde tierra, con lo cual lo primordial es que la aplicación se mantenga activa mandando información, sin importar demasiado que se pierdan algunos paquetes en el proceso (mientras se reciban un porcentaje aceptable de estos paquetes).

El protocolo TCP/IP es **orientado a conexión** y garantiza que todos los datos van a llegar de un programa al otro correctamente. Se utiliza cuando la información a transmitir es importante; no se puede perder ningún dato y no importa que los programas se queden 'bloqueados' esperando o transmitiendo datos. Si uno de los programas está atareado en otra cosa y no atiende la comunicación, el otro quedará detenido hasta que el primero lea o escriba los datos. Por ello es el protocolo utilizado para el envío del archivo de datos registrados en el experimento, así se evita que se pierda información y el archivo quede incompleto. Esto es muy importante ya que son estos datos los que serán procesados y estudiados en siguientes fases del proyecto Hermes.

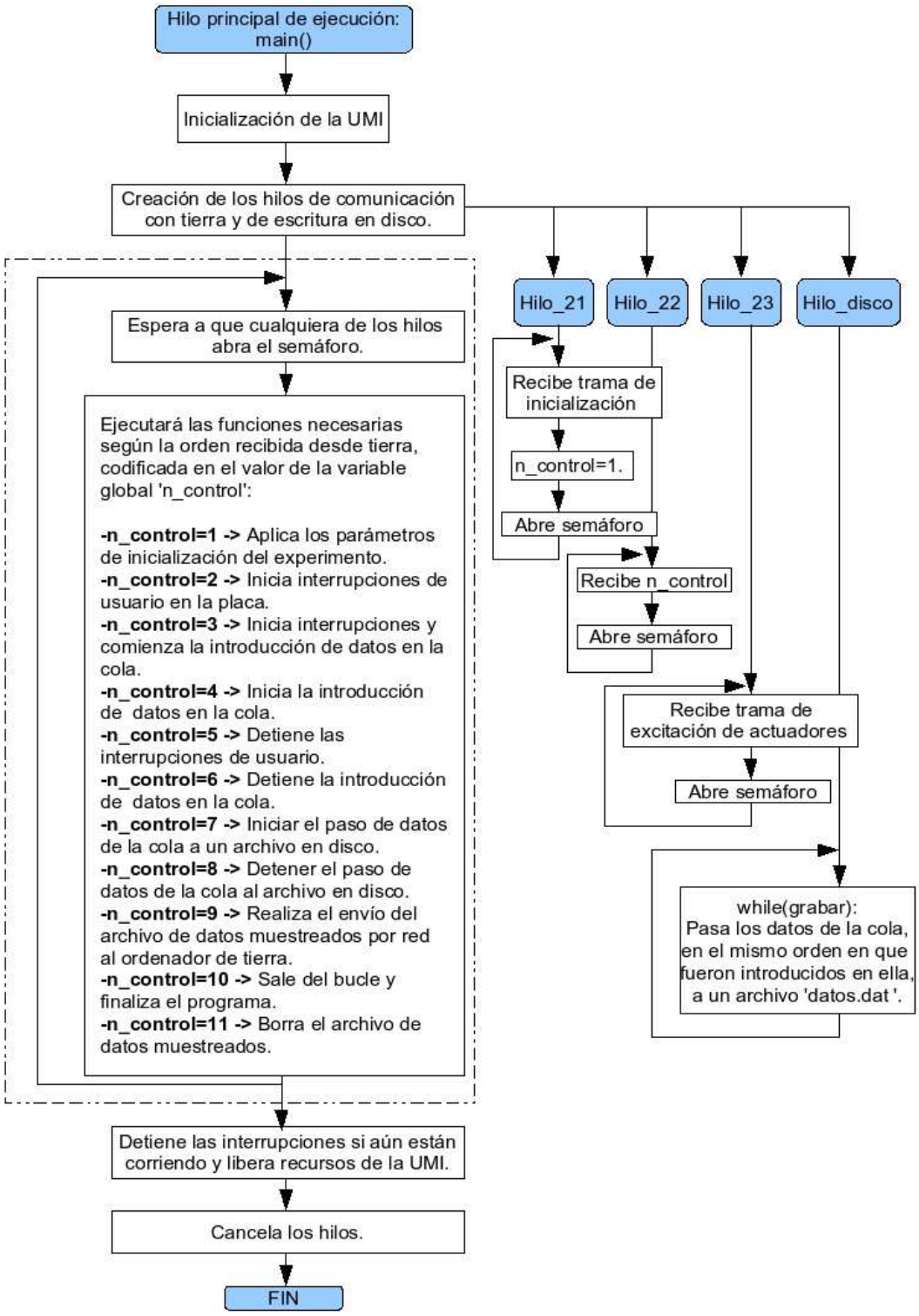

Figura 3.1: Esquema de la estructura en hilos de la aplicación en el computador de vuelo.

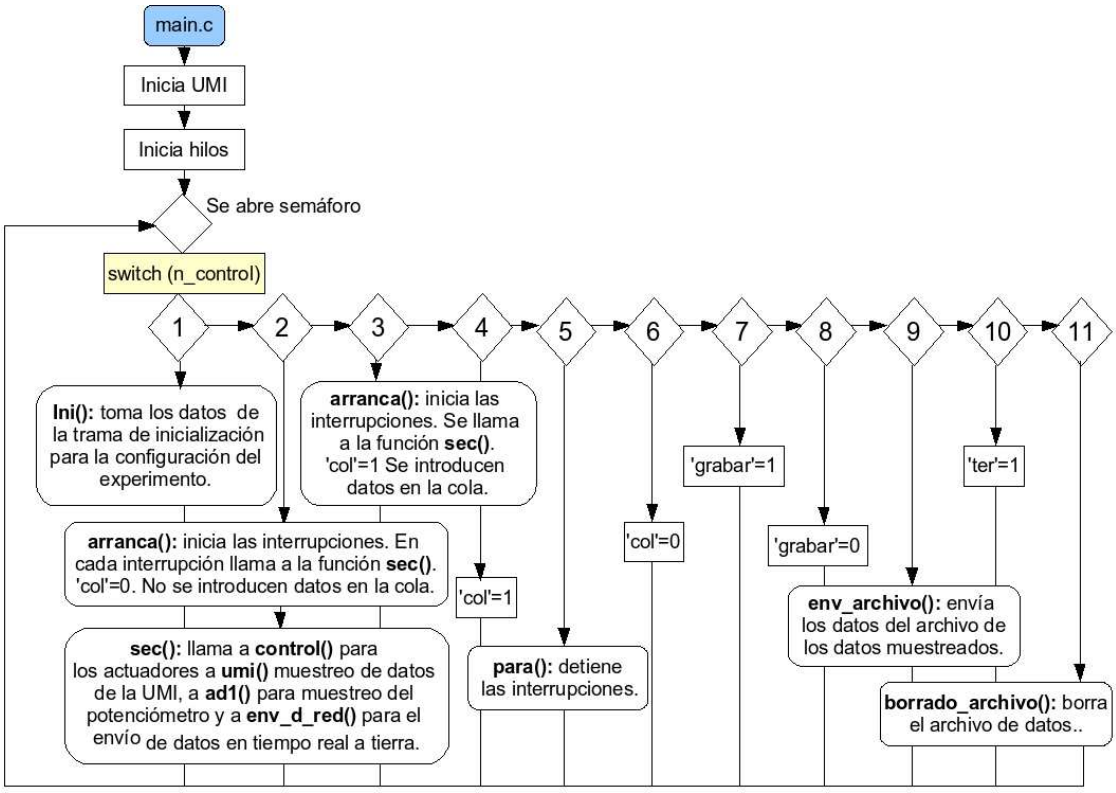

Figura 3.2: Esquema de las distintas funciones que se ejecutan en el bucle principal de la aplicación en el computador de vuelo.