Capítulo 8
Diseño del Sistema de Control Terrestre

En este capítulo:

La arquitectura del sistema de control terrestre

El proyecto se centra principalmente en el diseño del hardware y software del sistema de control de la aeronave por lo que el software del PC se le ha dado menor importancia y se ha hecho muy sencillo con el fin de visualizar los parámetros necesarios. El software en el sistema de control terrestre está compuesto por tres elementos principales. El Sistema de control de tierra llamado "EPF *Ground Control Station*", el Google Earth y el Simulador de vuelo X-Plane.

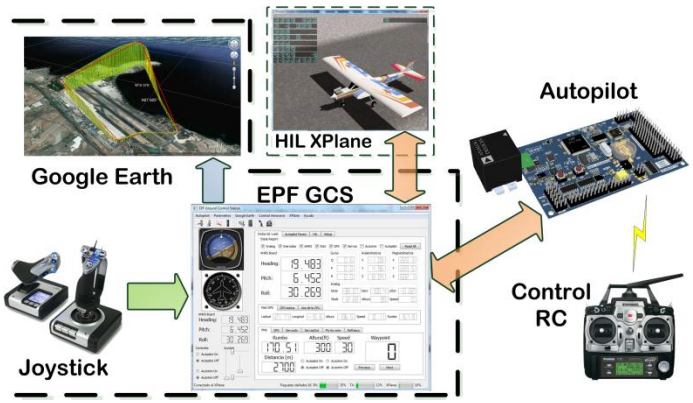


Figura 1 – Diagrama de conexión del software en el sistema de control terrestre.

EPF Ground Control Station

El EPF-GCS (*Ground Control Station*) es el corazón del sistema de control de tierra. Éste realiza tres funciones principales.

- Se encarga de realizar la comunicación con el piloto automático y de mostrar los parámetros más importantes. También realiza la función de interfaz con el usuario y permite interactuar con el hardware en tiempo real.
- Se comunica con el Google Earth para dibujar la ruta de vuelo que está siguiendo el avión en tiempo real.
- Se comunica con el simulador de vuelo X-Plane para realizar simulaciones de *hardware in the loop*.

La interfaz gráfica diseñada se muestra en la Figura 2. Dispone de una barra de menús, una barra de herramientas en la zona superior. En la zona inferior se encuentra la barra de estado que muestra mensajes de información del estado del software, un indicador de paquetes dañados, un indicador del uso del canal de recepción, un indicador de uso del canal de transmisión y un indicador del uso del canal de comunicaciones con el X-Plane. En la zona izquierda se ha colocado un instrumento gráfico (ADI) para indicar el *pitch* y *roll*. Debajo del ADI encontramos información de la *attitude* de la placa y debajo se muestra la información de los controles actuales que gobiernan la aeronave. Por último las pestañas de navegación principal permiten navegar para mostrar los distintos parámetros y opciones que ofrece el piloto automático.

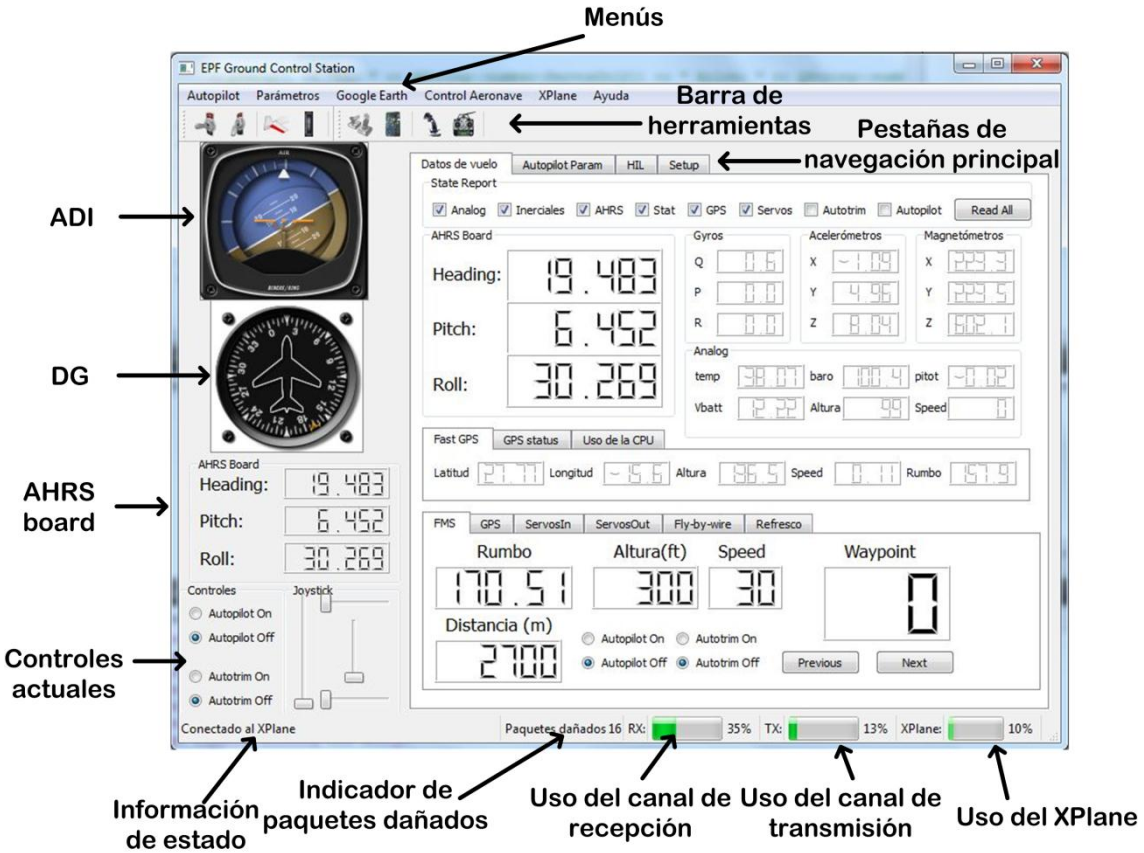
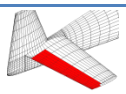


Figura 2 – Interfaz gráfica del EPF *Ground Control Station*.

Barra de herramientas

Existen dos barras de herramientas que se pueden mover y colocar en cualquier lado de la ventana, o dejarla flotante. La información de cada acción en la barra de herramienta se muestra en la Tabla 1.

Tabla 1 – Botones de la barra de herramientas.

Acción	Descripción
	Piloto automático On. Conecta el piloto automático cuando está en modo de control por joystick.
	Piloto automático Off. Desconecta el piloto automático cuando está en modo de control por joystick.
	Autotrim On. Activa el compensador automático para el timón de profundidad.
	Autotrim Off. Desactiva el compensador automático para el timón de profundidad.
	Control manual del X-Plane. Permite controlar el X-Plane a través del joystick o del radiocontrol. No hay intervención de los algoritmos implementados.
	Control del X-Plane a través del ordenador de a bordo. La información de control se envía a la placa del piloto automático. Ésta realiza los algoritmos implementados y envía la información de control al X-Plane. El conjunto de este botón y el anterior se pueden considerar como un conmutador de manual o asistido.
	La señal de entrada del control en el piloto automático lo toma del joystick. Esto incluye la información del control de los alerones, timón de profundidad, timón de cola, gases, el control del autotrim y autopilot los toma de los botones antes mencionados.
	La señal de entrada del control en el piloto automático se toma del mando de radio control. Esto incluye la información del control de los alerones, timón de profundidad, timón de cola, gases, el control del *autotrim* y *autopilot* los toma de un interruptor en el mando de radio control.

En la Figura 3 se puede ver un diagrama simple del uso de los conmutadores para el control del X-Plane. El conmutador más a la derecha permite seleccionar entre un control manual y control a través del hardware diseñado. En el control manual, la señal que proviene tanto del joystick o del mando de RC se pasa directamente a las superficies de control de vuelo del X-Plane. Aunque no se indica por simplificar el diagrama, la señal del mando de RC proviene de la lectura que hace el ControlDSC sin ningún tipo de procesado. El conmutador de más a la izquierda permite controlar la señal que se usa como señal de control de entrada, o el joystick o el mando de RC. Cada conmutador indica con el mismo icono que se usó en la barra de herramientas que conexión realiza. La posición que indica la imagen, es la posición por defecto que aparece cuando se abre la aplicación la primera vez. Cuando se selecciona el modo asistido, la señal de control se envía al ordenador de a bordo y éste utilizando los algoritmos de *fly-by-wire* implementados, calcula la mejor señal de control que debe enviar a las superficies de control de vuelo. Este modo, también permite que cuando se conecte el

piloto automático, el ordenador de abordo tome el control total de la aeronave para cumplir con la misión asignada. Cuando está en modo piloto automático, las señales de entrada de control provenientes tanto del joystick, como del mando de RC son ignoradas.

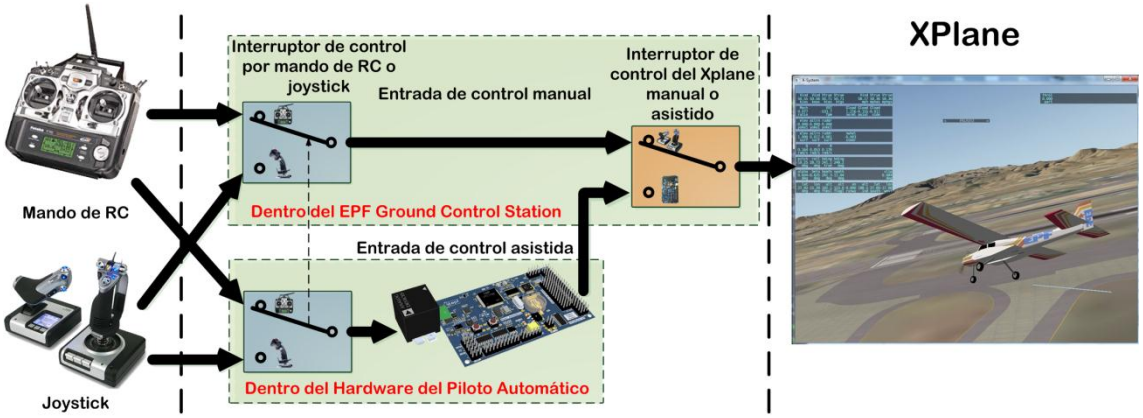


Figura 3 – Diagrama de uso de los conmutadores de la barra de herramientas para controlar el X-Plane.

Menús

En los menús se encuentran controles que permiten interactuar con el hardware del piloto automático. A continuación se listan la organización de los menús.

- Autopilot
  - Autopilot On
  - Autopilot Off
  - Autotrim On
  - Autotrim Off
  - Selector de velocidad
    - GPS
    - Tubo Pitot
  - Selector de altura
    - GPS
    - Baro
- Parámetros
  - Start
  - Stop
  - Sensores Analógicos
    - On
    - Off
  - Sensores Inerciales
    - On
    - Off
  - AHRS
    - On
    - Off
  - Estadísticas
    - On
    - Off
  - GPS
    - On
    - Off
  - ServosIn

        - On
        - Off
    - Fly-by-wire
        - On
        - Off
    - FMS Fly plan
        - On
        - Off
    - FMS HIL Fly plan
        - On
        - Off
- Google Earth
    - Reset
- Control Aeronave
    - Plane RC
    - Plane Joystick
    - Control ServosOut
        - On
        - Off
- XPlane
    - HIL Load Gando
    - HIL Load Ruta Isla
    - HIL Load Alternativa
    - HIL Load Vuelo
    - HIL Next Waypoint
    - HIL Prev Waypoint
    - Joystick
    - Fly-by-wire
- Ayuda
    - about

ADI + DG

El ADI (*Attitude Director Indicator*) también llamado AI (*Attitude Indicator*) es un instrumento usado en las aeronaves para informar al piloto de la orientación de ésta relativa al horizonte de la tierra. Indica el ángulo pitch y el ángulo de bank (correspondiente al roll).

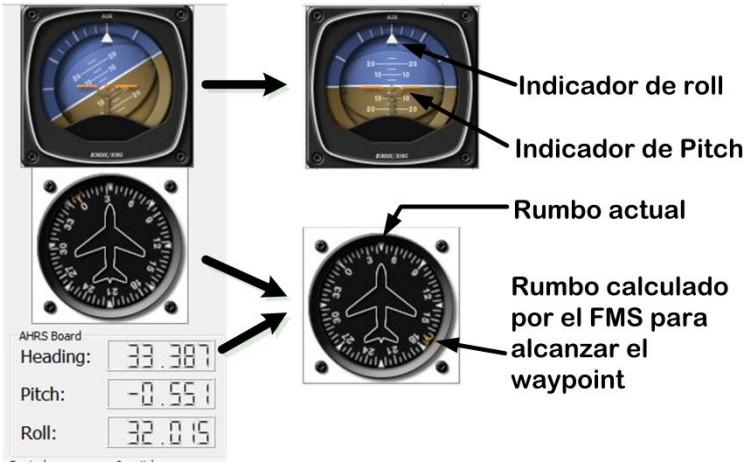


Figura 4 – ADI + DG.

El DG (*Directional Gyro* o *heading indicator*) informa del rumbo de la aeronave.

Indicador de controles actuales

En la zona izquierda inferior del software se puede encontrar el estado de las entradas de control para los controles actuales elegidos según los conmutadores antes explicados. Si se elige como control de entrada el joystick, aparecerá el estado del joystick de los ejes de profundidad (*elevator*), alerones (*aileron*), timón de cola (*rudder*) y gases (*throttle*), así como el control actual de piloto automático como del autotrim. En el caso de seleccionar el mando de RC como entrada, estos ejes son gobernados por el movimiento de los ejes del mando de RC, y el control de piloto automático, como del autotrim se controlan con los interruptores que incorpora el mando de RC.

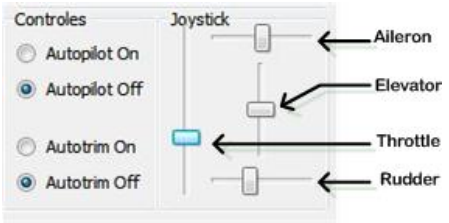


Figura 5 – Controles actuales.

Pestañas de navegación

Las pestañas de navegación se utilizan para navegar a través de los distintos parámetros del software. Cada pestaña agrupa información en común. Comenzando con la pestaña de "Datos de vuelo", se puede encontrar todos los parámetros más importantes que se calculan en la placa del piloto automático.

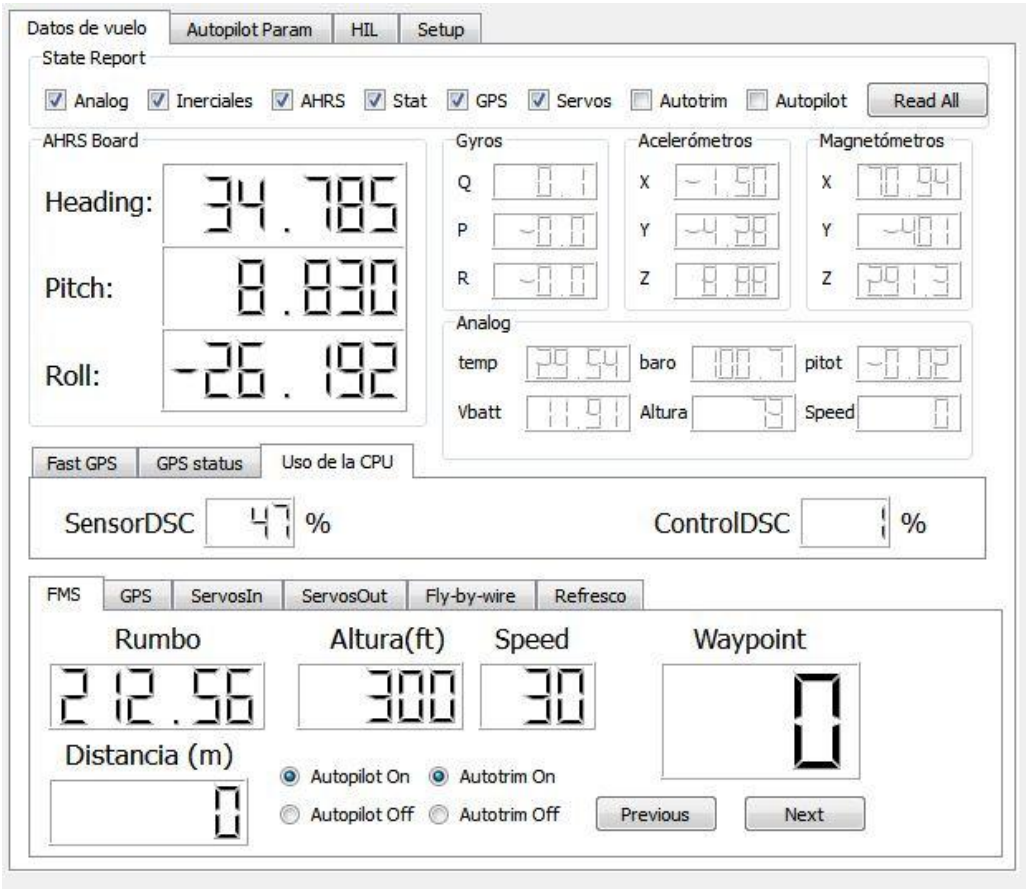


Figura 6 – Pestaña principal de Datos de vuelo.

Cada uno de estos parámetros se pueden activar o desactivar utilizando el menú parámetros. Por ejemplo, si se desea activar la información del AHRS, se puede activar con el menú Parámetros->AHRS->On (o desactivar con Parámetros->AHRS->Off). Para cada parámetro existe una opción en el menú parámetros que activa o desactiva su actualización.

Esta pestaña muestra información del estado interno del piloto automático, del AHRS calibrado, de los sensores inerciales (gyros, acelerómetros, magnetómetros) calibrados, de los sensores analógicos (batería, temperatura, presión barométrica y diferencial, altura y velocidad aérea calibrada).

Debajo se encuentran tres grupos agrupados en pestañas. En la pestaña "Fast GPS" (Figura 7) se puede encontrar información rápida de los datos más importantes del GPS. En la pestaña "GPS status" (Figura 8) se muestra el número de satélites usado para hacer la estimación de la posición y un indicador de que los datos proporcionados son válidos. Por último en la pestaña "Uso de la CPU" (Figura 9) se muestra el uso de CPU de los dos procesadores que se encuentran en el hardware del piloto automático.

Figura 7 – Pestaña Fast GPS.

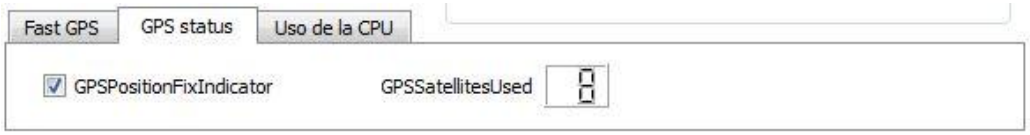


Figura 8 – Pestaña GPS status.

Figura 9 – Pestaña Uso de la CPU.

En la parte inferior se encuentra otro grupo de pestañas que contienen la siguiente información.

- La pestaña de FMS (Figura 10) contiene información del plan de vuelo actual que está siguiendo el avión real. Se puede saltar al siguiente o anterior waypoint.
- La pestaña GPS (Figura 11) contiene la información extraída del GPS.
- La pestaña ServosIn (Figura 12) muestra información de la lectura de los 12 canales de entrada.
- En la pestaña de ServosOut (Figura 13) se puede modificar la posición de cada uno de los servos de salida.
- La pestaña Fly-by-wire (Figura 14) muestra información del valor de los cuatro canales de control calculados por el sistema de control de vuelo *fly-by-wire*.
- Por último, en la pestaña Refresco (Figura 15) se puede cambiar la tasa de refresco de las tareas que se encargan de enviar la información al EPF GCS. Por defecto se ha fijado en 100 ms todos los parámetros y la señal de información de los servos de entrada y del fly-by-wire se actualizan cada 20ms.

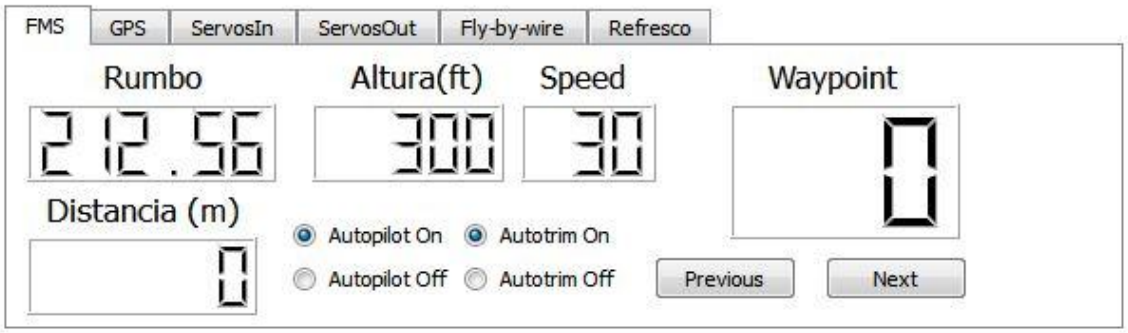


Figura 10 – Pestaña FMS.

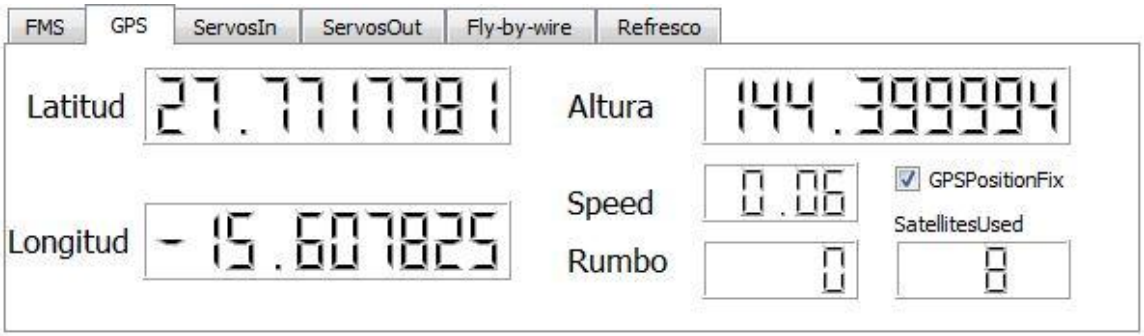


Figura 11 – Pestaña GPS.

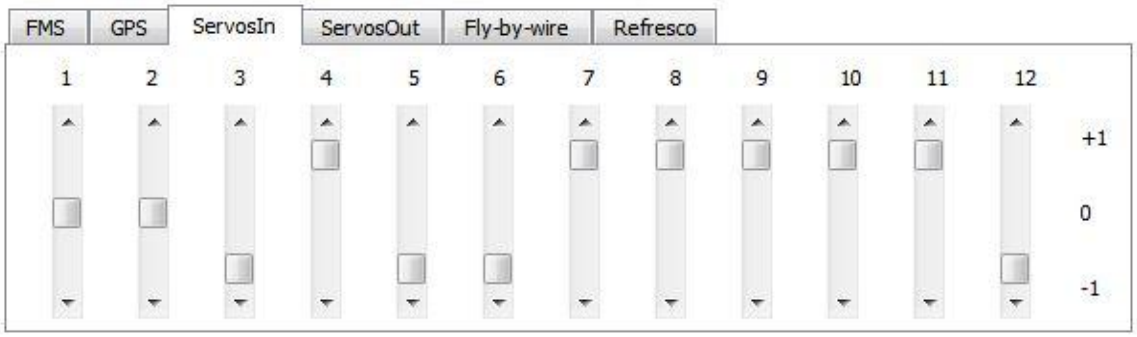


Figura 12 – Pestaña ServosIn.

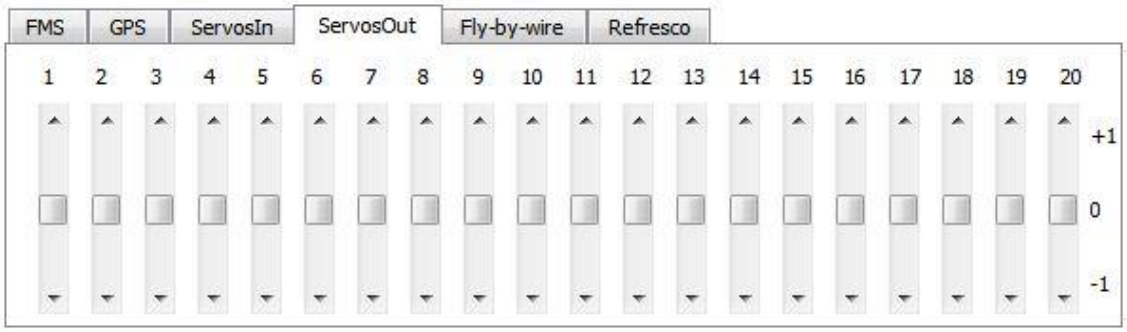


Figura 13 – Pestaña ServosOut.

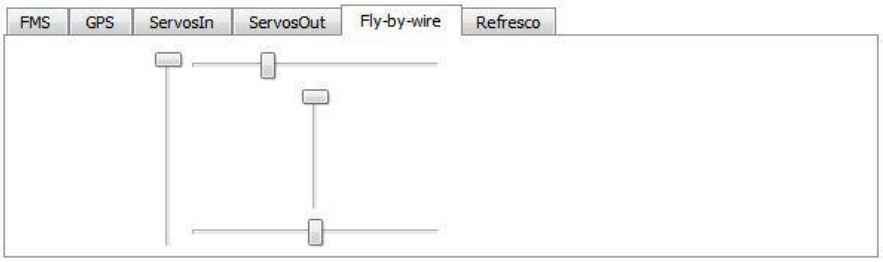


Figura 14 – Pestaña Fly-by-wire.

Figura 15 – Pestaña Refresco.

En la pestaña de navegación "Autopilot Param" (Figura 16) se encuentran los parámetros del algoritmo del piloto automático. Pulsando en el botón load se cargan los parámetros actuales, y modificando cada campo se actualizan en tiempo real en el hardware. Con el botón Store, se almacenan estos parámetros en la tarjeta de memoria SD.

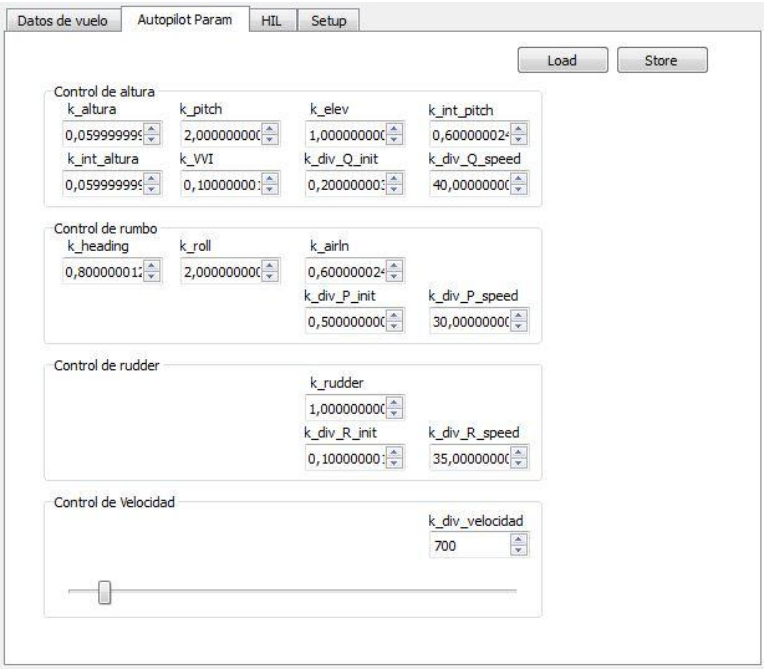


Figura 16 – Pestaña de navegación Autopilot Param.

La pestaña de navegación HIL (Figura 17) se utiliza para realizar simulación con *hardware in the loop*. En esta pestaña se presentan los parámetros más importantes extraídos del simulador y la zona inferior los parámetros de navegación calculados en el hardware del piloto automático. Los ejes marcados como fly-by-wire corresponden al control de la aeronave calculado por los algoritmos implementados en el piloto automático. La pestaña Setup (Figura 18) permite configurar la comunicación con el hardware y con el XPlane. También permite enviar comandos de forma manual por el puerto serie y muestra toda la información recibida del simulador de vuelo.

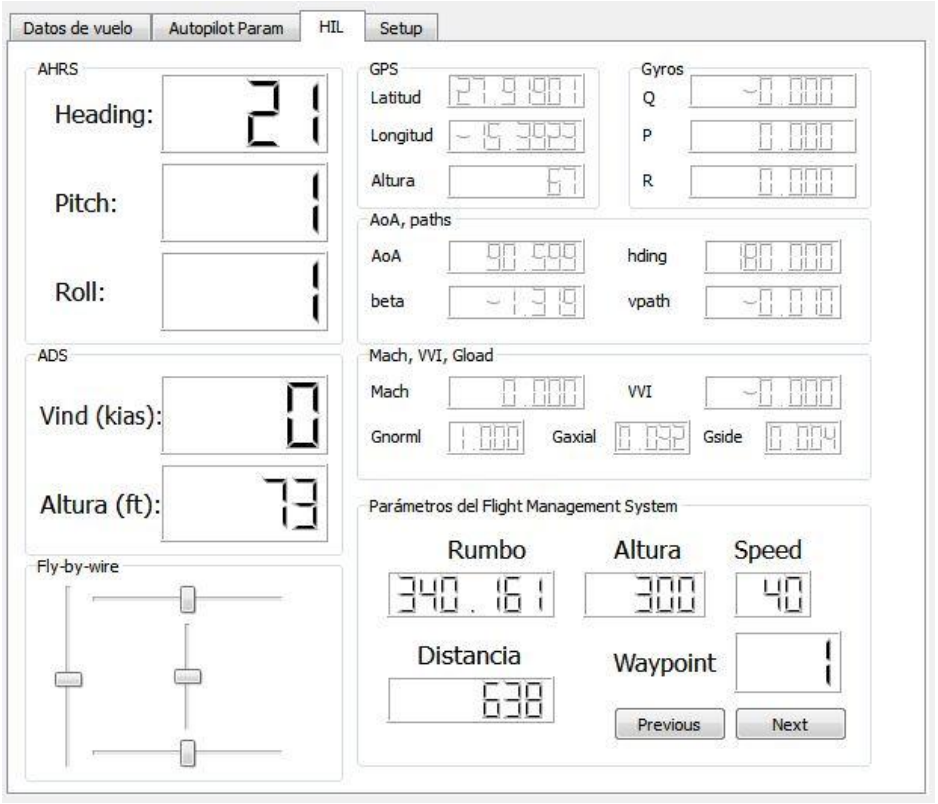
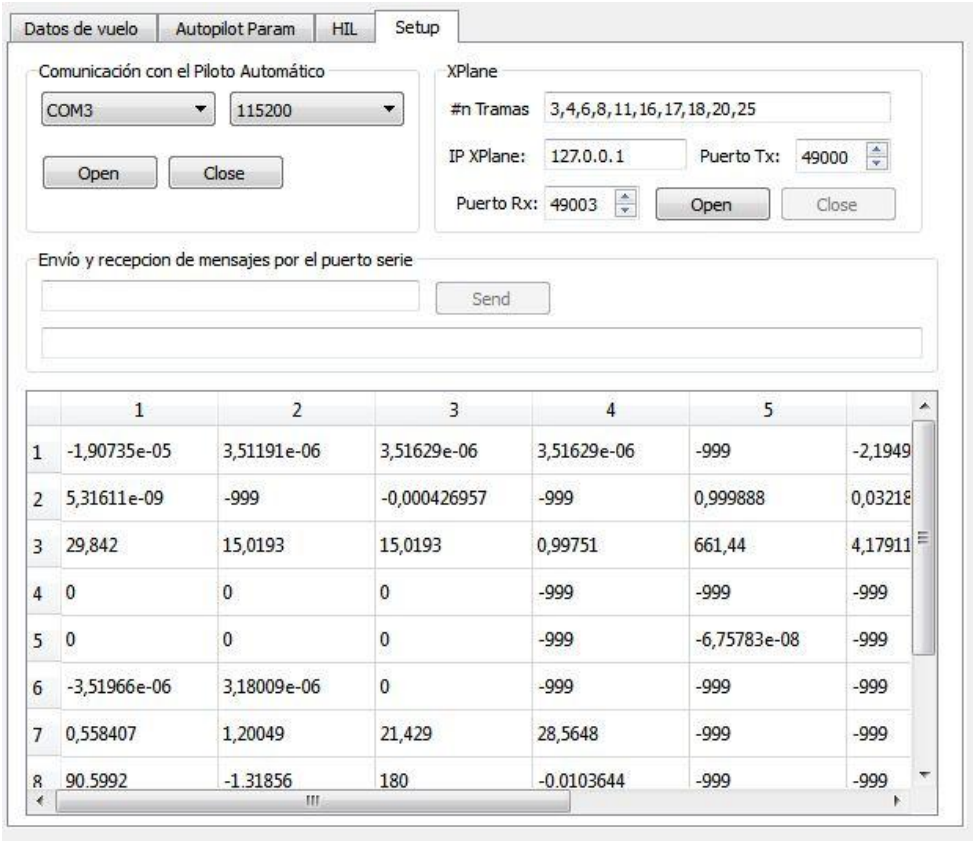


Figura 17 – Pestaña de navegación HIL.



	1	2	3	4	5	
1	-1,90735e-05	3,51191e-06	3,51629e-06	3,51629e-06	-999	-2,1949
2	5,31611e-09	-999	-0,000426957	-999	0,999888	0,03218
3	29,842	15,0193	15,0193	0,99751	661,44	4,17911
4	0	0	0	-999	-999	-999
5	0	0	0	-999	-6,75783e-08	-999
6	-3,51966e-06	3,18009e-06	0	-999	-999	-999
7	0,558407	1,20049	21,429	28,5648	-999	-999
8	90,5992	-1,31856	180	-0,0103644	-999	-999

Figura 18 – Pestaña de navegación Setup.

Software en la placa

Para comunicarse con el GCS, el hardware del piloto automático utiliza principalmente cuatro tareas como se muestra en la Figura 19.

- La tarea qServosTask, se encarga de enviar información de la lectura de los 12 servos de entrada y del control calculado por el fly-by-wire.
- La tarea qFMSTask, envía información sobre el plan vuelo real, el plan de vuelo del HIL, de las estadísticas del uso de las dos CPU e información del GPS.
- La tarea qParamTask envía la información del AHRS, de los sensores inerciales, de los sensores analógicos, y está continuamente enviando información acerca del estado del sistema. Éste último no tiene opción de desconectarse como los anteriores parámetros y se utiliza como *Heartbeat*.
- Por último, cuando el software se conecta con la placa, la tarea ttyTask entra en modo de comandos binarios. Esto permite optimizar el uso del canal de comunicaciones ya que se envían muchos datos. Esta tarea implementa un intérprete de comandos binarios y se encarga del control y la gestión de las otras tareas. Los comandos que es capaz de aceptar se verán más adelante. También se encarga de procesar los algoritmos del *hardware in the loop*.

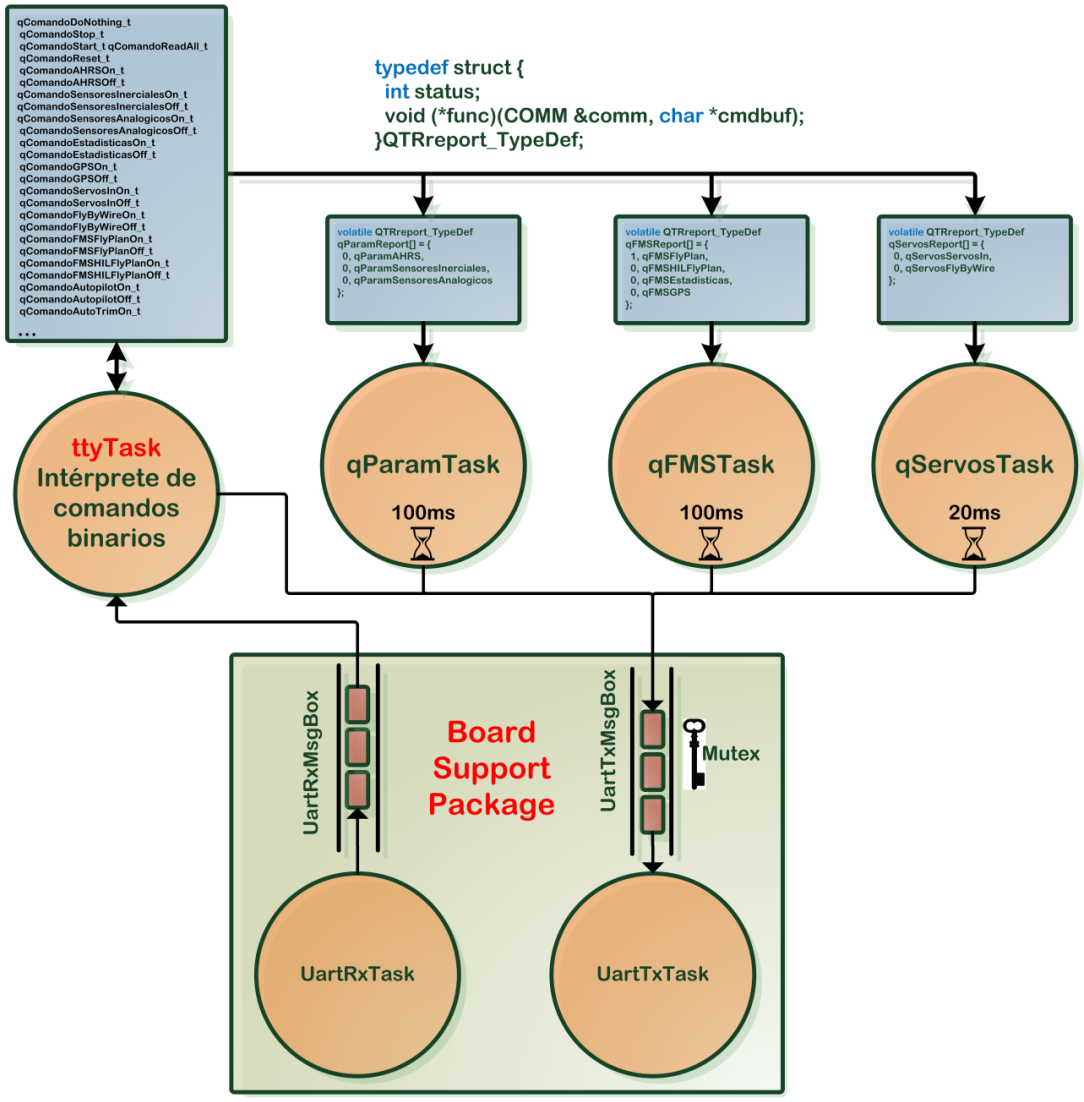


Figura 19 – Arquitectura del software en el hardware del piloto automático para comunicarse con el EPF GCS.

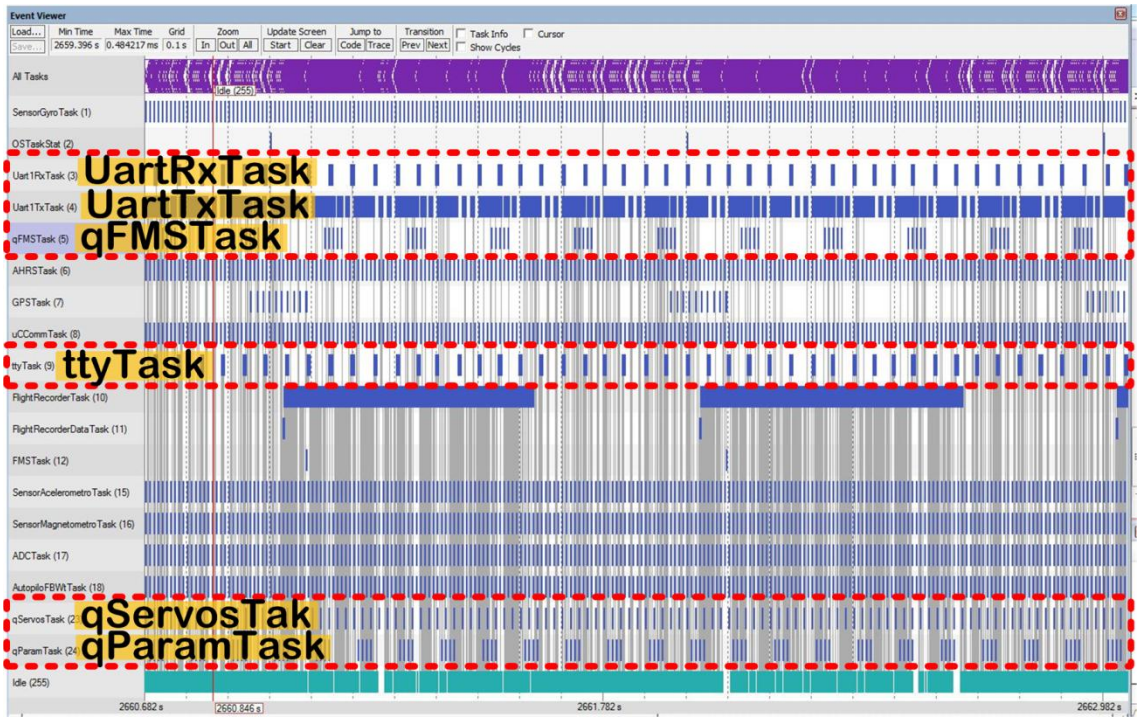


Figura 20 – Debugging en hardware. Vista global del uso de la CPU por las diferentes tareas que se comunican con el EPF GCS.

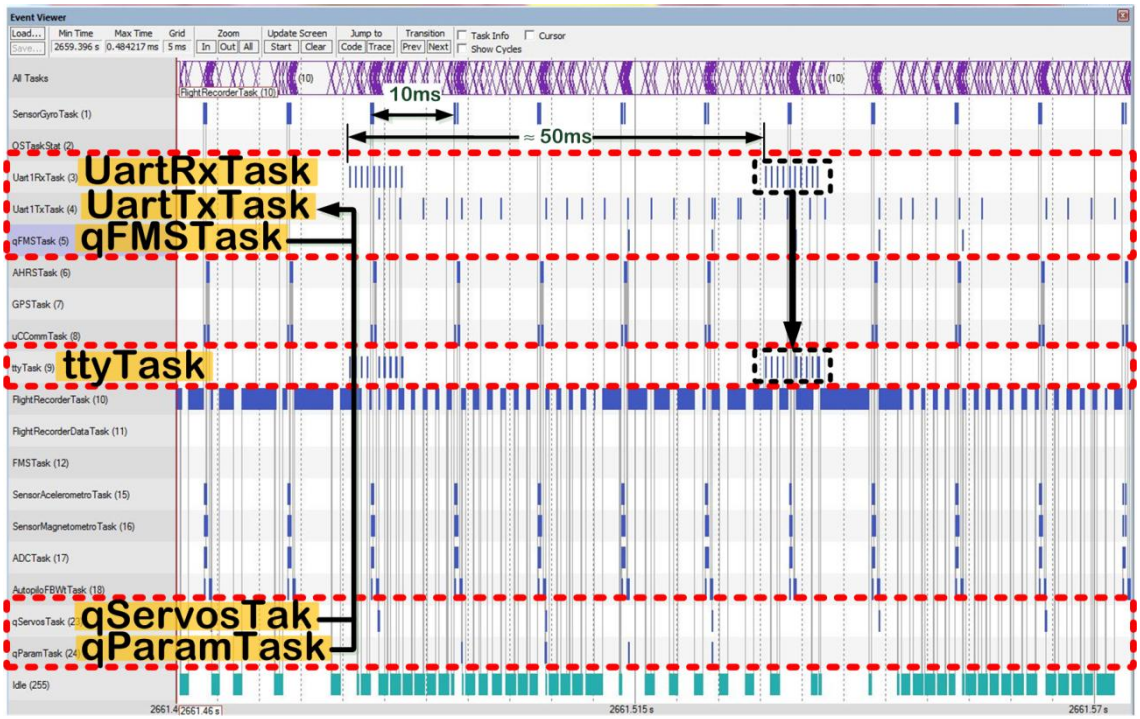


Figura 21 – Debugging en hardware. Uso de la CPU por las diferentes tareas que se comunican con el EPF GCS.

La Figura 20 y Figura 21 muestran el uso de la CPU por las tareas que se encargan de la comunicación con el GCS. Las tareas ttyTask, qParamTask, qServosTask y qFMSTask tienen las menores prioridades. De esta forma se asegura que en ningún momento se vayan a apropiar de la CPU cuando alguna de las tareas importantes necesite usarla. Con esto se consigue que se siga cumpliendo las restricciones de *hard real time*, y como se puede ver en las imágenes anteriores el sistema responde en tiempo real a las peticiones del GCS.

Para el ejemplo, se han activado todos los parámetros y se está ejecutando la simulación HIL. El simulador se configuró para enviar paquetes cada 50 ms aproximadamente. Como se verá en siguientes apartados, la trama del HIL que se envía al piloto automático ocupa 76 bytes (1xHeader + 1xTipo + 15xdata + 1xCRC + 1xFin). El umbral de la FIFO de recepción del UART se fijó en 8 bytes, por lo que el sistema recibe la trama en 10 paquetes como se muestra en la Figura 22. La tarea UartRxTask lee paquetes de 8 bytes y los coloca en el *mailbox* de recepción. La tarea ttyTask lee del *mailbox* los paquetes recibidos y ejecuta el algoritmo del intérprete de comandos que se explica en el siguiente apartado. Una vez interpreta el comando, si necesita enviar datos, los coloca en el *mailbox* de transmisión y es la tarea UartTxTask quién se encarga de transmitirlos por el puerto serie. Lo mismo ocurre para las otras tareas que intentan enviar datos al GCS. Cada vez que desean enviar una trama, lo coloca en el *mailbox* de transmisión y es la tarea UartTxTask quién se encarga de transmitirla como se explicó en el capítulo 6. Para evitar que los datos de distintas tareas se mezclen, el *mailbox* está protegido con un mutex.

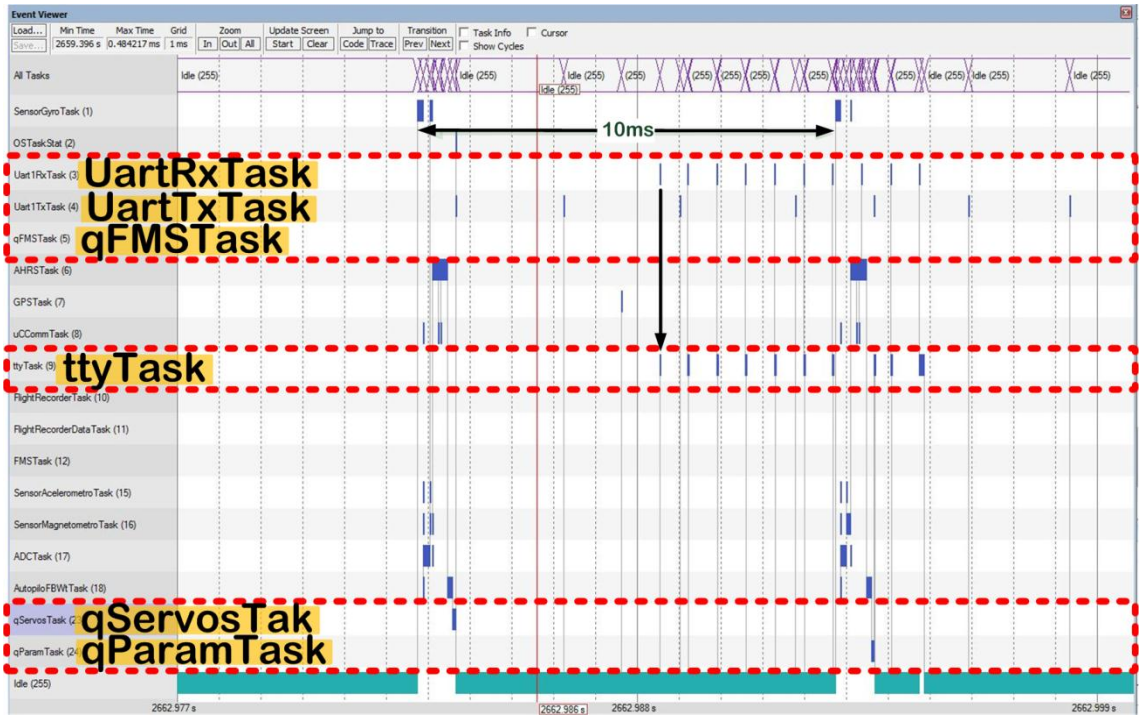


Figura 22 – Debugging en hardware. Zoom del uso de la CPU por las diferentes tareas que se comunican con el EPF GCS.

Formato de las tramas

El formato de las tramas usadas para la comunicación entre el hardware del piloto automático y el GCS tiene la forma que se muestra en la Figura 23. La longitud de las tramas es variable y depende del tipo. A continuación se describe cada campo.

Figura 23 – Formato genérico de las tramas utilizadas para la comunicación con el EPF GCS.

- Header (4 bytes). Indica el comienzo de la trama. En la comunicación de la placa al PC contiene "\rQT," en ASCII y del PC a la placa "#QT,".
- Tipo (integer de 4bytes). Indica el tipo de la trama.
- Contenedor de datos. Contiene los datos a transmitir. Puede ser un campo vacío para algunos tipos de tramas.

- CRC (4 bytes). Se utiliza para el control de errores. En el proyecto se realiza el campo tipo y el contenedor de datos para generar este campo. La operación realiza un *checksum* de 32-bits.
- Fin de trama (4 bytes). Contiene "\r\n\r\n" en ASCII.

El intérprete de comandos debe ser capaz de recibir las tramas y decodificarlas. Para ello se implementó un algoritmo que tiene en cuenta el peor caso que se muestra en la Figura 24. Los datos se reciben divididos en paquetes. Para este ejemplo se reciben 3 paquetes. Estos paquetes pueden contener datos no válidos como una trama anterior donde se haya modificado el campo fin de trama. La trama se puede recibir en varios paquetes distintos como en el ejemplo de la Figura 24. Estos paquetes se van acumulando en un buffer interno y cuando se detecta el fin de trama en el paquete actual recibido, se busca el comienzo de la trama en el buffer, se extrae la trama completa y se envía al decodificador. Los datos restantes se mantienen en el buffer para unirlos con los datos futuros que se esperan recibir. A continuación se lista el algoritmo implementado en el software del PC.

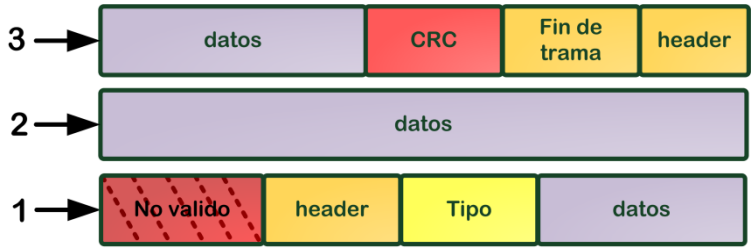


Figura 24 – Peor caso de recepción de una trama en varios paquetes.

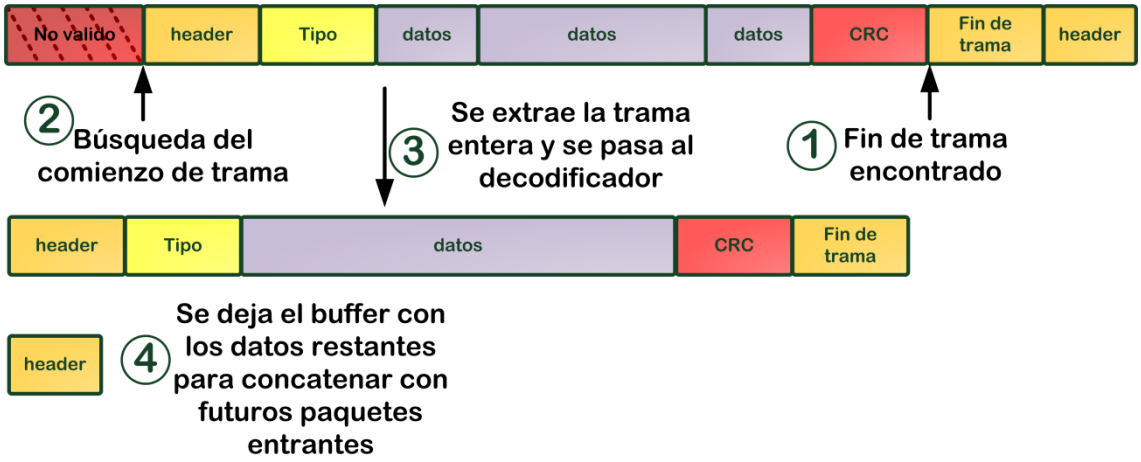


Figura 25 – Proceso de extracción de la trama.

Apartado: Formato de las tramas

```
QByteArray trama = serial->readAll();
emit rxserialctr(trama.size());
int indice = 0;
paqueterecibido += trama;
indice = trama.indexOf("\r\n\r\n");
while( indice != -1){
   indice = paqueterecibido.indexOf("\rQT,");
   if (indice != 0) emit fallopaquete(1);
   paqueterecibido.remove(0,indice + 4);
   indice = paqueterecibido.indexOf("\r\n\r\n");
   QByteArray data(paqueterecibido.data(),indice);
   emit falconReceived(data);
   paqueterecibido.remove(0,indice + 4);
   indice = paqueterecibido.indexOf("\r\n\r\n");
}
```

Comandos del PC a la placa

En la Tabla 2 se resumen los comandos implementados que envía el GCS al hardware del piloto automático.

Tabla 2 – Comandos del PC al hardware del piloto automático.

Nombre	Tipo	Contenedor de datos	Descripción
qComandoDoNothing	0	qComandoSimple_t	
qComandoStop	1	qComandoSimple_t	
qComandoStart	2	qComandoSimple_t	
qComandoReadAll	3	qComandoSimple_t	
qComandoReset	4	qComandoSimple_t	
qComandoAHRSOn	5	qComandoSimple_t	
qComandoAHRSOff	6	qComandoSimple_t	
qComandoSensoresInercialesOn	7	qComandoSimple_t	
qComandoSensoresInercialesOff	*21*	qComandoSimple_t	
qComandoSensoresAnalogicosOn	22	qComandoSimple_t	
qComandoSensoresAnalogicosOff	23	qComandoSimple_t	
qComandoEstadisticasOn	24	qComandoSimple_t	
qComandoEstadisticasOff	25	qComandoSimple_t	
qComandoGPSOn	26	qComandoSimple_t	
qComandoGPSOff	27	qComandoSimple_t	
qComandoServosInOn	28	qComandoSimple_t	
qComandoServosInOff	29	qComandoSimple_t	
qComandoFlyByWireOn	30	qComandoSimple_t	
qComandoFlyByWireOff	31	qComandoSimple_t	
qComandoFMSFlyPlanOn	33	qComandoSimple_t	
qComandoFMSFlyPlanOff	34	qComandoSimple_t	
qComandoFMSHILFlyPlanOn	*35*	qComandoSimple_t	
qComandoFMSHILFlyPlanOff	36	qComandoSimple_t	
qComandoAutopilotOn	37	qComandoSimple_t	
qComandoAutopilotOff	38	qComandoSimple_t	
qComandoAutoTrimOn	39	qComandoSimple_t	
qComandoAutoTrimOff	40	qComandoSimple_t	
qComandoFMSNextWaypoint	41	qComandoSimple_t	
qComandoFMSPrevWaypoint	42	qComandoSimple_t	
qComandoHILFMSNextWaypoint	43	qComandoSimple_t	
qComandoHILFMSPrevWaypoint	*45*	qComandoSimple_t	
qComandoSetParamTaskRate	46	qComando1var_t	
qComandoSetFMSTaskRate	47	qComando1var_t	
qComandoSetServosTaskRate	48	qComando1var_t	

qComandoExitQTMode	49	qComandoSimple_t
qComandoxplanInfo2board	50	qHILXPlaneInfo_t
qComandoHILLoadGando	51	qComandoSimple_t
qComandoHILLoadRutaIsla	52	qComandoSimple_t
qComandoHILLoadAlternativa	53	qComandoSimple_t
qComandoHILLoadVuelo	54	qComandoSimple_t
qComandoControlModeJoysticRC	55	qComando1var_t

qComandoSimple_t

La Figura 26 muestra el formato de la trama utilizada para los comandos de tipo qComandoSimple_t.

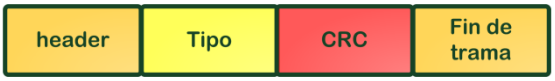


Figura 26 – Formato de la trama qComandoSimple_t

qComando1var_t

La Figura 27 muestra el formato de la trama utilizada para los comandos de tipo qComando1var_t.

Figura 27 – Formato de la trama qComando1var_t.

qHILXPlaneInfo_t

La Figura 28 muestra el formato del contenedor de datos que se utiliza para los comandos de tipo qHILXPlaneInfo_t.

Figura 28 – Datos incluidos en el contenedor de datos qHILXPlaneInfo_t. Cada variable sigue el formato de punto flotante de precisión simple (32bits) IEEE 754.

qServosOut_t

La Figura 29 muestra el formato del contenedor de datos que se utiliza para los comandos de tipo qServosOut_t;

Figura 29 – Contenedor de datos de los comandos de tipo qServosOut_t. servoNum es un entero y *value* un float.

Comandos de la placa al PC

En la Tabla 3 se resumen los tipos de paquetes que envía el hardware del piloto automático al *Ground Control Station*.

Tabla 3 – Tipos de paquetes enviados desde el piloto automático al EPF GCS.

Nombre	Tipo	Contenedor de datos	Descripción
qPCParamHeartbeat	0	qPacketParameHeartbeat_t	
qPCPacketParamAHRS	1	qPacketParamAHRS_t	
qPCPacketParamSensoresInerciales	2	qPacketParamSensoresInerciales_t	
qPCPacketParamSensoresAnalogicos	3	qPacketParamensoresAnalogicos_t	
qPCPacketServosServosIn	4	qPacketServosIn_t	
qPCPacketServosFlyByWire	5	qPacketServosFlyByWire_t	
qPCPacketFMSFlyPlan	6	qPacketFMSFlyPath_t	
qPCPacketFMSHILFlyPlan	7	qPacketFMSFlyPath_t	
qPCPacketFMSEstadisticas	8	qPacketFMSEstadisticas_t	
qPCPacketFMSGPS	9	qPacketFMSGPS_t	
qPCPacketHILXPlane	10	qPacketHILXPlane_t	
qPCPacketParamAutopilotFBW	11	qPacketParamAutopilotFBW_t	

qPCParamHeartbeat

La tarea qParamTask envía continuamente datos del estado del sistema enviando paquetes de tipo qPacketParameHeartbeat_t. La Figura 30 muestra los datos que se envían. Cada campo es de tipo char (8bits). Un uno indica activado, y un cero desactivado. El capo autopilotHILStatus codifica en el bit 0 el modo autopiloto para HIL y el bit 1 indica el estado del modo de control, o joystick o mando de RC.

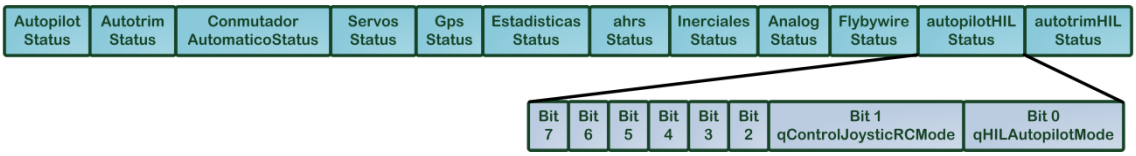


Figura 30 – Contenedor de datos de paquetes de tipo qPacketParameHeartbeat_t.

qPCPacketParamAHRS

La Figura 31 muestra el formato del contenedor de los paquetes de tipo qPacketParamAHRS_t. Cada parámetro es de tipo float.

Figura 31 – Contenedor qPacketParamAHRS_t.

qPCPacketParamSensoresInerciales

La Figura 32 muestra el formato del contenedor de los paquetes de tipo qPacketParamSensoresInerciales_t. Cada parámetro es de tipo float.

Figura 32 – Contenedor qPacketParamSensoresInerciales_t.

qPCPacketParamSensoresAnalogicos

La Figura 33 muestra el formato del contenedor de los paquetes de tipo qPacketParamensoresAnalogicos_t. Cada parámetro es de tipo float.

Figura 33 – Contenedor qPacketParamensoresAnalogicos_t.

qPCPacketServosServosIn

La Figura 34 muestra el formato del contenedor de los paquetes de tipo qPacketServosIn_t. Cada parámetro es de tipo float y valor entre -1.0 y +1.0.

Figura 34 – Contenedor qPacketServosIn_t.

qPCPacketServosFlyByWire

La Figura 35 muestra el formato del contenedor de los paquetes de tipo qPacketServosFlyByWire_t. Cada parámetro es de tipo float.

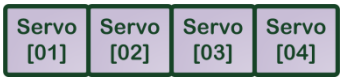


Figura 35 – Contenedor qPacketServosFlyByWire_t.

qPCPacketFMSFlyPlan y qPCPacketFMSHILFlyPlan

La Figura 36 muestra el formato del contenedor de los paquetes de tipo qPacketFMSFlyPath_t. Los parámetros *bearing*, altura y velocidad son de tipo float y el resto de tipo integer.

Figura 36 – Contenedor qPacketFMSFlyPath_t.

qPCPacketFMSEstadisticas

La Figura 36 muestra el formato del contenedor de los paquetes de tipo qPacketFMSEstadisticas_t. Cada parámetro es de tipo short.

Figura 37 – Contenedor qPacketFMSEstadisticas_t.

qPCPacketFMSGPS

La Figura 36 muestra el formato del contenedor de los paquetes de tipo qPacketFMSGPS_t. Los parámetros latitud, longitud, altura, velocidad y rumbo son de tipo float, el resto de tipo int.

Figura 38 – Contenedor qPacketFMSGPS_t.

Simulador de vuelo X-Plane

El simulador de vuelo elegido para realizar las simulaciones con *hardware in the loop* fue el X-Plane (versión 10). Éste se puede ejecutar en Windows, Linux o Mac. X-Plane es un simulador de vuelo civil creado por Austin Meyer. Es uno de los principales simuladores de vuelo que compiten contra Microsoft Flight Simulator. De acuerdo con su desarrollador, se trata de un simulador extremadamente preciso, basado en el cálculo del efecto del flujo de aire sobre las superficies de los aviones simulados.

La Administración Federal de Aviación (FAA) de Estados Unidos de América ha autorizado su uso, con hardware específico, para el entrenamiento de pilotos de vuelo instrumental.

Figura 39 – Simulador de vuelo X-Plane con el modelo del avión de RC utilizado en la realidad.

X-Plane funciona leyendo la forma geométrica de cualquier aeronave y luego averigua cómo va a volar ésta. Esto se logra mediante un proceso llamado "*blade element theory*", que consiste en romper el avión en muchos elementos pequeños para luego encontrar las fuerzas que actúan sobre cada elemento muchas veces por segundo. Estas fuerzas se convierten en aceleraciones, que luego se integran y se obtienen las velocidades y posiciones.

En este proyecto se eligió X-Plane como simulador de vuelo para realizar las simulaciones HIL principalmente por la facilidad de extracción de parámetros. X-Plane envía y recibe los datos a través de tramas UDP con el formato de la Figura 40.

Figura 40 – Trama de datos UDP que envía X-Plane.

Cada trama incluye:

- Header (4bytes). Contiene “DATA” en ASCII para todos los paquetes.
- T (1 byte). Tag interno usado para indicar la dirección del flujo de datos. 0x40 (‘@’) para datos de salida del simulador y 0x00 para entrada.
- Un contendor de datos que se compone de varios paquetes concatenados. Cada paquete contiene información de la opción marcada para salida en el menú “Data Output” del simulador (Figura 41).
    - Data Ref. (4bytes) de tipo entero. Indica el tipo del paquete. Coincide con el número que aparece junto a cada opción de la Figura 41.
    - 8 variables codificadas en punto flotante de precisión simple que contienen el valor de los distintos parámetros.

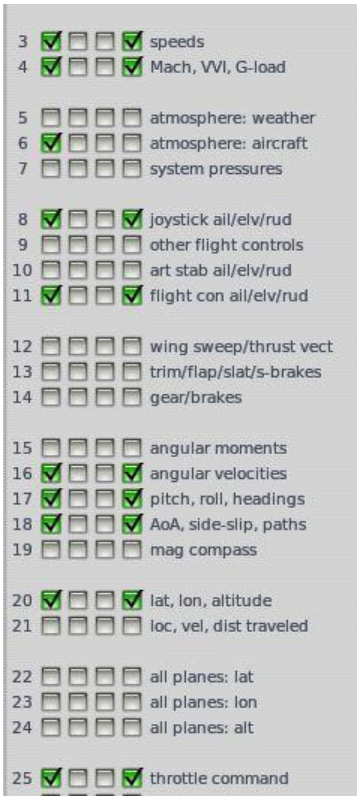


Figura 41 – Datos de salida del simulador.

Google Earth

Google Earth es un programa informático que muestra un globo virtual que permite visualizar múltiple cartografía, con base en la fotografía satelital. El mapa de Google Earth está compuesto por una superposición de imágenes obtenidas por Imagen satelital, fotografía aérea, información geográfica proveniente de modelos de datos SIG de todo el mundo y modelos creados por ordenador. El programa está disponible en varias licencias, pero la versión gratuita es la más popular y es la utilizada en este proyecto, disponible para móviles, tablets y PC's.

Por simplicidad, en este proyecto se utilizó Google Earth para indicar la posición de la aeronave en tiempo real. Para futuras versiones se pretende integrar en el software del GCS. Google Earth permite actualizar la posición y la *attitude* cada segundo. También permite dibujar el plan de vuelo en 3D como se muestra en la Figura 42 y el recorrido seguido por la aeronave en 3D como se muestra en líneas amarillas en la Figura 43.

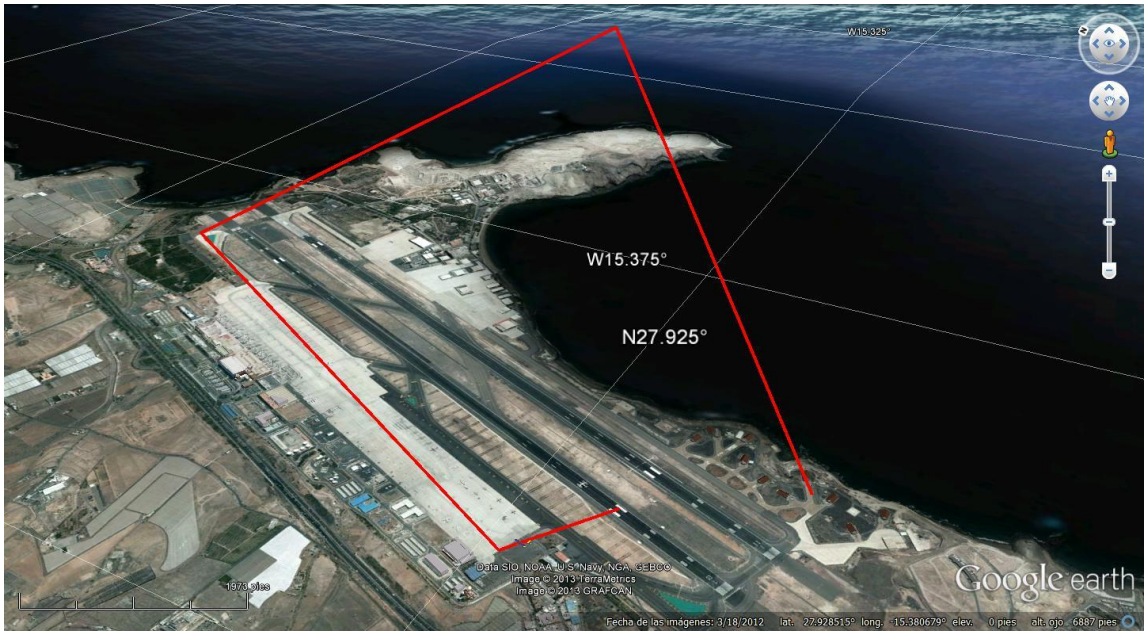


Figura 42 – Plan de vuelo marcado en Google Earth.

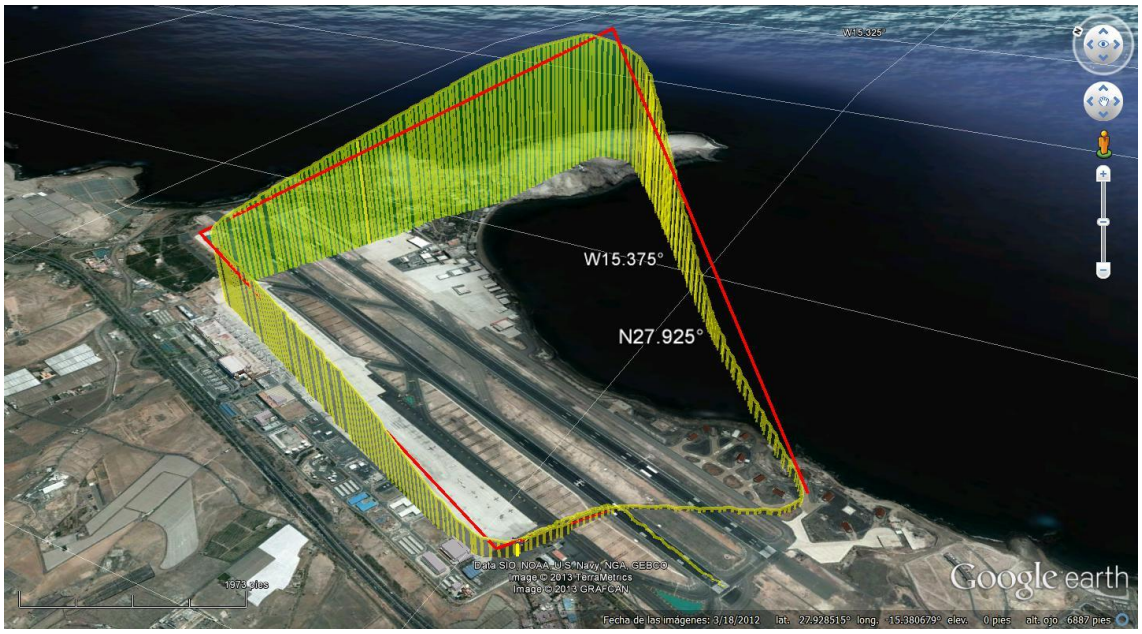


Figura 43 – Plan de vuelo superpuesto con la trayectoria seguida por la aeronave.

Test AHRS con Processing

Para realizar los test del módulo AHRS, se diseñó una interfaz gráfica utilizando Processing y las librerías de OpenGL. El objetivo es hacer un objeto virtual en 3D que siga el movimiento que realiza la placa en tiempo real. En la Figura 44 y Figura 45 se puede ver un ejemplo de cómo el avión virtual sigue el movimiento de la PCB diseñada en este proyecto. También se puede ver un vídeo del sistema funcionando en el siguiente link: http://youtu.be/U07sFiWqxSM.

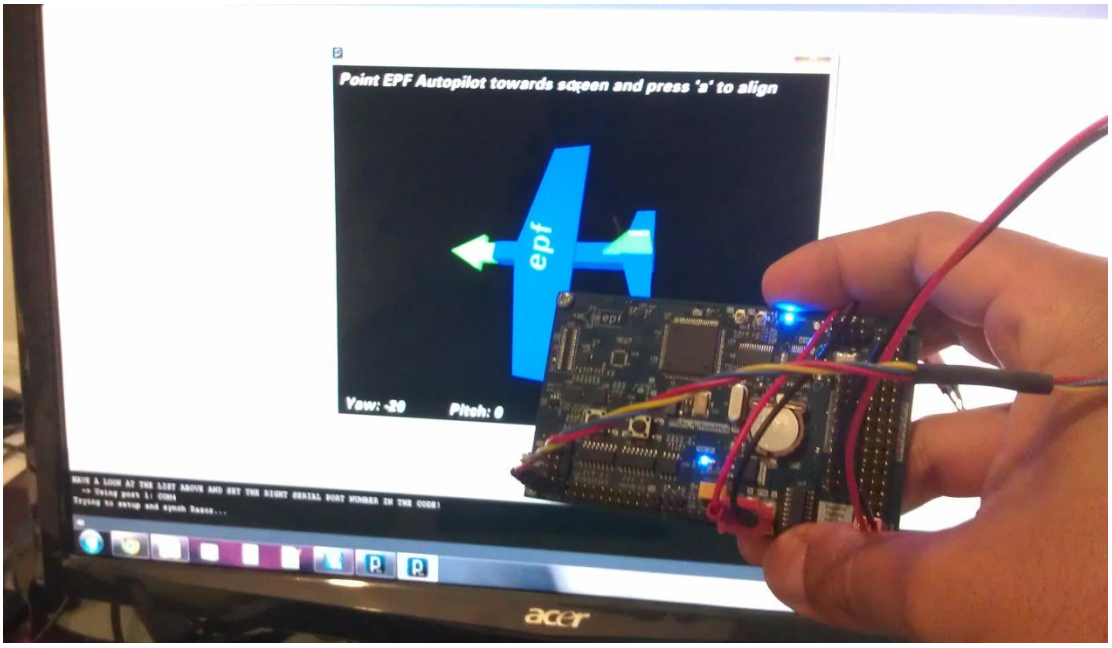


Figura 44 – Software 3D para el test del AHRS hecho en Processing.

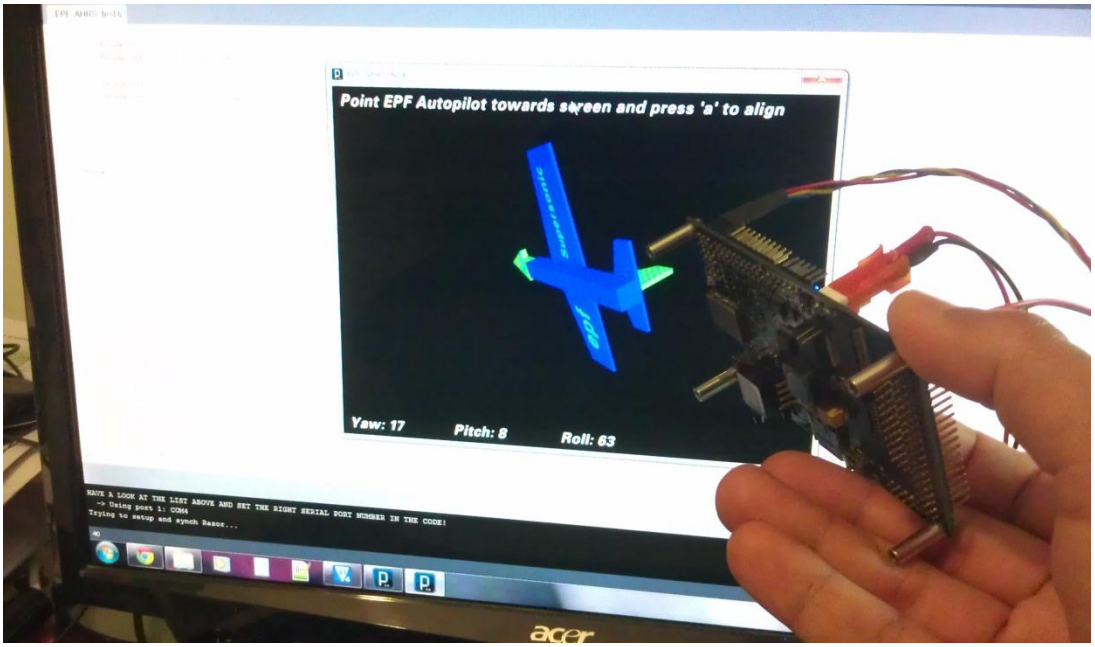


Figura 45 – Software 3D para el test del AHRS hecho en Processing.

Bibliografía

[1] Alan Ezust. An Introduction to Design Patterns in C++ with QT. Prentice Hall Open Source Software Development.

[2] Javier Pérez Mato. Master Thesis: “Design and Implementation of an Optical Wireless Interface for the ARINC 429 AVIONICS Bus”.