Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FINAL DE CARRERA

TÍTULO DEL TFC: Sistema de Control y Navegación para MAV

TITULACIÓN: Ingeniería Técnica Aeronáutica, especialidad Aeronavegación

AUTOR: Héctor Valle Ruiz

DIRECTOR: Jaime Óscar Casas Piedrafita

FECHA: 10 de junio de 2010

Titulo: Sistema de Control y Navegación para MAV

Autor: Héctor Valle Ruiz

Director: Jaime Óscar Casas Piedrafita

Fecha: 10 de junio de 2010

Resumen

En este trabajo se pretende implementar una plataforma lo más genérica posible, de control y navegación para MAV de bajo coste, que incluya:

1. Una arquitectura programada con software libre y que sea transferible a otras aplicaciones. Para ello se utilizará la plataforma Arduino, cuya programación se realiza en un controlador de la marca Atmel.

2. Incorporar sensores que permitan tanto la localización del vehículo (GPS), analizar la actitud del vuelo (acelerómetros), calcular distancia a obstáculos cercanos (ultrasonidos) y poder realizar navegación visual (cámara de video).

3. Implementar los programas para la adquisición de datos y el control de motores en el microcontrolador de la placa.

4. Comunicar toda la información captada por los sensores a bordo del vehículo de forma inalámbrica, a través de XBee los sensores incorporados directamente a Arduino y por protocolos propios para el resto de sensores.

5. Recibir los datos a través de XBee a un ordenador en que se utilizará un programa realizado con LabView para visualizarlos, procesarlos y grabarlos.

6. Realizar una aplicación para mostrar la información GPS a través de la plataforma Google Earth en tiempo real que permita conocer la posición del vehículo en todo momento.

7. Llevar a cabo el control del vehículo con una aplicación de LabView que permita utilizar un mando de la plataforma para juegos Wii de Nintendo, cuyos datos se recibirán a través de Bluetooth.

Title: Navigation and Control system for MAV

Author: Héctor Valle Ruiz

Director: Jaime Óscar Casas Piedrafita

Date: June 10, 2010

Overview

This work aims to implement a low cost platform for control and navigation for MAV, including:

1. A programmable architecture with free software easy to transfer to other applications. This will be achieved by Arduino platform, which programation is performed by a controller from Atmel.

2. Incorporate sensors that allow vehicle location (GPS), to analyze the attitude of the flight (accelerometers), calculate distance to nearby obstacles (ultrasound) and to perform visual navigation (video camera).

3. Implement programs for data acquisition and motor control in the microcontroller of the plate.

4. Convey all the information captured by sensors on board the vehicle wirelessly, through XBee sensor incorporated directly into Arduino and own protocols for other senors.

5. Receive data via XBee to a computer, which use a program made with LabView for viewing processing and recording.

6. Making an application to display GPS information through the Google Earth platform in real time to show the position of the vehicle at all times.

7. Undertake control of the vehicle with a LabView application that can run a command platform for the Nintendo Wii games and which data is received through Bluetooth.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

1.1. Introducción a los MAVs

1.1.1. Del UAV al MAV

Un MAV (*Micro Air Vehicle*) es un vehículo aéreo no tripulado de pequeñas dimensiones. Para definirlos con corrección, hay que conocer primero las características de los vehículos aéreos no tripulados en general, o UAVs (*Unmanned Aerial Vehicle*) y concretar posteriormente las peculiaridades de los MAVs.

La ausencia de un piloto para dominar el vehículo a bordo del mismo es la principal característica de los UAVs. Las herramientas de control que se utilizan van un paso más allá del uso de mandos directamente conectados a los actuadores y necesitan cerrar el lazo de control entre los sensores de medida y los actuadores a través de un sistema automático empotrado en la aviónica de la nave o a través de un control a distancia desde la base de operaciones. La carencia de tripulante hace de estos vehículos una herramienta ideal para operar en circunstancias en que los seres humanos sencillamente no podrían trabajar, o bien en condiciones peligrosas ya que lo único que se pone en riesgo es la integridad del propio UAV y no vidas humanas. Otros terrenos en los que es adecuado su uso son aquellos en los que la presencia del piloto no es necesaria, como realizar fotografías desde el aire, o supervisar grandes áreas a través de una cámara, por ejemplo en supervisión de catástrofes o de plantaciones agrícolas.

Fig 1.1 UAV Predator

La principal diferencia de los MAVs respecto a los UAVs es su pequeño tamaño, que los hace especialmente interesantes por varias razones.

Una de las principales cualidades de esta clase de vehículos es la facilidad para llegar a zonas de difícil acceso, donde un UAV normal no podría acceder. Se incluye la capacidad de navegación en interiores, fomentada por su facilidad para realizar vuelos estacionarios.

El peso de estos vehículos suele ser menor que el del resto, y aunque esto implique un menor consumo energético también conlleva ciertas limitaciones.

Las condiciones en que puede operar son limitadas, y el entorno ha de ser más controlado, debido a que factores meteorológicos como una ráfaga de viento podrían afectar gravemente a su estabilidad. Su alcance de vuelo es menor que el del resto de UAVs, aunque el uso de estos vehículos ya lo tiene en cuenta, dado que su objetivo no es realizar grandes recorridos, sino poder realizar intervenciones locales en condiciones en que se requiera discreción o alcanzar lugares a los que es difícil llegar por otras vías.

En temas de logística, la facilidad para ser almacenados es mucho mayor (no hace falta disponer de un hangar), así como una mayor viabilidad para construirlos o adquirirlos.

1.1.2. Historia de los UAV - MAV

Dado que la historia de los MAVs en sí es extraordinariamente corta y está muy ligada a la de los vehículos aéreos no tripulados en general, el siguiente repaso engloba las dos categorías.

A lo largo de la historia, un patrón que se repite es el hecho de que los conflictos bélicos suelen coincidir con grandes etapas para el desarrollo tecnológico, ya que las inversiones de los países en armamento y la tecnología de las que se pueda sacar provecho en la guerra se disparan. Así pues, el desarrollo de los UAV no es una excepción ya que prácticamente todos los puntos destacados en su evolución histórica coinciden con algún conflicto.

El primer testimonio histórico de vehículos aéreos no tripulados aparece el 22 de agosto de 1849, cuando Austria atacó la ciudad de Venecia utilizando globos cargados con explosivos. Como primera experiencia no terminó de funcionar del todo, ya que aunque algunos de estos globos sí que alcanzaron su objetivo, otros vieron modificada su trayectoria debido a un repentino cambio en la dirección de viento y volvieron tras las líneas austriacas. Aunque la idea de los globos que arrastra el viento no termina de ajustarse a lo que hoy se considera un UAV, las bases para el futuro uso de estos vehículos ya estaban asentadas.

No fue hasta 1916, coincidiendo con la Primera Guerra Mundial, en que fueron diseñados y puestos en funcionamiento unos nuevos ingenios más cercanos al UAV, utilizando un control mediante una emisora AM. Aunque más que vehículos aéreos, eran lo que hoy llamaríamos misiles, el sistema de control

empezaba a tener un grado de sofisticación considerable. Su primer objetivo fue intentar neutralizar Zeppelines.
Durante el periodo que siguió, destaca, entre 1927 y 1929, la creación del LARYNX por parte de la Royal Navy, un misil de crucero con forma muy similar a un avión que podía salir de un buque de guerra y ser controlado a través del piloto automático. A raíz de los éxitos que se estaban cosechando en la investigación de los aviones sin piloto, en la década de los 30 se crearon, tanto en los EEUU como en Gran Bretaña, programas de radio control para aeronaves. De este modo, en 1935 y después de varias pruebas, los ingleses llegaron a fabricar el DH.82B Queen Bee en gran número.

Aún así, la primera serie de aviones no tripulados fabricados a gran escala fue obra de Reginald Denny. Había servido en la primera guerra mundial, en la Royal Flying Corps, lo cuál le permitió empezar a formarse una idea de lo que podrían llegar a suponer los UAVs. Después de la guerra emigró a los EEUU, para intentar llevar a cabo carrera como actor en Hollywood. Una vez allí, nunca dejó de mostrar interés por el aeromodelismo y el control por radio. El negocio de Denny con los pequeños modelos empezó a aflorar, creando la Radioplane Company. A finales de los años 30, realizó una serie de exhibiciones para el ejército con quienes acabó obteniendo un contrato para la producción de los Radioplane OQ-2 en 1940 (Figura 1.2). Este vehículo era de características muy similares a las aeronaves convencionales y disponía de un motor de 6CV. A lo largo de la Seguna Guerra Mundial se llegaron a producir quince mil de estos modelos para el ejército.

Fig 1.2 Radioplane OQ-2

Por su lado, la marina americana ya había investigado durante esos años y en 1937 ya tenía desarrollado el N2C-2, que ofrecian la posibilidad de ser controlados desde otra aeronave de mayor tamaño, el TG-2. Su objetivo principal fué neutralizar aviones enemigos. Durante la Segunda Guerra Mundial, se realizaron pruebas tanto para controlar remotamente aeronaves de mayor tamaño como bombarderos, como para suministrar propulsión pulsejet, aunque sin demasiado éxito en niguno de los casos.

A partir de aquí, el éxito de los OQ-2 motivó a la creación de nuevos UAV En este caso la nueva generación de vehículos se utilizaría principalmente para realizar ensayos en entrenamientos y pruebas en general.

A partir de los años 50 y coincidiendo nuevamente con el inicio de un conflicto militar, en este caso Vietnam, aparece una nueva utilidad para los UAV. Se decide utilizar aviones no tripulados para realizar la función de señuelos en ataques a gran escala. El primer modelo específico para llevar a cabo esta tarea es el McDonnell Douglas ADM-20 Quail, que se combinó con los bombarderos B-52 para penetrar en el espacio aereo enemigo. También se desarrollaron nuevos modelos para llevar a cabo tareas de reconocimiento como el MQM-57 Falconer o posteriormente el MQM-58 Overseer. A finales de esa década, las velocidades que se alcanzaban en aviación llegaban a Mach 2, con lo cual los modelos tanto de UAVs normales, como los que se utilizaban para interceptar vehículos enemigos se tubieron que adaptar a las nuevas necesidades.

Hasta ahora, sólo hemos visto la evolución de estos vehículos a través de las inversiones realizadas por los EEUU y Gran Bretaña, pero lo cierto es que por su parte la URSS también tenía planes de desarrollo para este tipo de vehículos sobretodo durante los tiempos de la guerra fría, aunque se llevaron a cabo tan en secreto que aún hoy es difícil seguir la pista de dichas investigaciones ya que en muchas ocasiones la información es contradictoria.

No fue hasta la década de los 80, en que se empezó a considerar el uso de UAV como vehículos con potencial militar más allá del reconocimiento de áreas y su uso como antimisiles dirigidos. Han sido nuevamente los conflictos bélicos más recientes en la historia (los Balcanes en los 90, Irak y Afganistán) los que han puesto al descubierto las capacidades de los vehículos aéreos no tripulados para entrar en combate. Esto se debe también a las nuevas herramientas que se pueden incorporar a bordo de las aeronaves, de mayor precisión.

Fig 1.3 MAV de desarrollo reciente del ejército de los Estados Unidos

En la actualidad, la potencia mundial más importante en uso y desarrollo tanto de UAVs como de MAVs son las fuerzas armadas de Israel.

La creación de los primeros MAVs no llega hasta la década de 1990, utilizando las tecnologías más innovadoras del momento. Dada su reciente aparición, su uso militar aún no está tan claramente definido, así que de momento se utilizan indistintamente en ámbitos de investigación y militar.

A nivel de mercado, está claro que es un terreno prácticamente virgen y que se puede explotar en muchos sentidos ya que ofrecen una gran cantidad de posibilidades y accesos hasta ahora no investigados. Los accesos a zonas difíciles y su fácil almacenaje pueden promover su distribución, llegando a sustituir otros elementos más anticuados y menos prácticos.

1.1.3. Categorías funcionales

Los UAVs se pueden clasificar según varios criterios. Por lo que respecta a sus aplicaciones existen seis categorías:

- Previsión de ataques aéreos: Recorren zonas en que se considera probable que puedan descubrir ataques enemigos antes de que estos lleguen a causar daños y dan un margen de tiempo para actuar.
- Reconocimiento: Para grandes áreas o zonas de difícil acceso en que se quiere hacer un seguimiento.
- Misiones de alto riesgo: No se ponen en peligro vidas humanas.
- Logística: Traslado de material y otras operaciones.
- Búsqueda y desarrollo: Investigación
- Aplicaciones civiles o comerciales: Publicidad en el aire, seguimiento del tráfico.

Cada una de estas categorías implica unas características técnicas determinadas según su función.

Otra clasificación se realiza en función si realizan despegue vertical, como helicópteros, o despegue no vertical, como aeroplanos. Concretamente esta distinción incluye:

Tabla 1.1. Clasificación de UAV

UAV			
Despegue vertical		Despegue no vertical	
Ala rotativa	Auto-sustentados	Ala flexible	Ala fija
Helicópteros	Dirigibles	Parapente	Aeroplanos
Quadrotores	Globos aerostáticos	Ala delta	-

Dentro de las categorías mencionadas, los vehículos que permiten despegue vertical se pueden clasificar según las prestaciones que ofrece cada uno de ellos:

Tabla 1.2. Comparativa de vehículos que permiten despegue vertical

	Rotor único	**Rotor axial**	**Rotor coaxial**	**Rotor tándem**	**Quad-rotor**	**Dirigible**	**Tipo pájaro**	**Tipo insecto**
Consumo	2	2	2	2	1	4	3	3
Control	1	1	4	2	3	3	2	1
Carga útil	2	2	4	3	3	1	2	1
Maniobrabilid.	4	2	2	3	3	1	3	3
Simplicidad mecánica	1	3	3	1	4	4	1	1
Complejidad aerodinámica	1	1	1	1	4	3	1	1
Baja velocidad	4	3	4	3	4	4	2	2
Alta velocidad	2	4	1	2	3	1	3	3
Miniaturización	2	3	4	2	3	1	2	4
Supervivencia	1	3	3	1	1	3	2	3
Vuelo estacionario	4	4	4	4	4	3	1	2
TOTAL	**24**	**28**	**32**	**24**	**33**	**28**	**22**	**24**

Los valores enumerados corresponden a una escala que va de 1-Malo a 4-Muy bueno. Lo que obtienen mejor resultado son el quadrotor y el MAV con rotor coaxial.

1.2. Sistemas de navegación

La navegación es un proceso que permite conocer el desplazamiento de un objeto de un punto a otro dentro de un espacio determinado.

Para que este proceso pueda hacerse de forma controlada, es básico que el punto de origen sea conocido. Partiendo de esa base, nos quedará saber cuales son el resto de puntos de la trayectoria del objeto para llegar a la ubicación final.

Los sistemas de navegación permiten llevar a cabo este proceso por diferentes medios. Los más conocidos son: Navegación a estima, por referencias externas, RADAR y GNSS.

1.2.1. Navegación a estima

Este tipo de navegación consiste en llegar a conocer la situación real del objeto extrapolando una posición anteriormente conocida. Es decir, si a las coordenadas iniciales les aplicamos las modificaciones que se realizan en la trayectoria podremos estimar la posición final. Los métodos utilizados para

obtener esta información son los que distinguen los diferentes tipos de navegación a estima, aunque como ya se verá, el sistema más extendido es el sistema de navegación inercial.

Teniendo en cuenta esto último, es muy importante que el nivel de precisión del sistema sea lo mayor posible, ya que uno de los principales inconvenientes de la navegación a estima es que los errores son acumulativos. Por ello es necesario reajustar el sistema de vez en cuando, dándole nuevas posiciones de referencia para eliminar estos posibles errores.

Una de las mayores ventajas de este tipo de navegación es que es autónoma y por tanto no necesita de equipos de apoyo terrestre. El sistema conoce por si mismo la posición que se ocupa en cada momento, ya que la extrapolación de la posición se lleva a cabo a bordo del propio objeto.

El tipo de sistema de navegación a estima más conocido es el que se conoce como *sistema de navegación inercial (INS)*.

Los INS se basan en medidas de aceleración realizadas en los tres ejes del espacio. Integrando estas aceleraciones se obtienen las componentes de velocidad correspondientes, y si se vuelve a integrar, se obtienen los valores de posición estimada. Como ya se ha comentado anteriormente, los sistemas han de ser lo más precisos posibles ya que los errores que se puedan cometer en las medidas iniciales también serán integrados dos veces, con lo cuál aumenta su relevancia en el cálculo del emplazamiento final.

$$v = \int a\,dt + v(0) \qquad \textbf{(1.1)}$$

$$x = \iint a\,dt^2 + v(0)t + x(t) \qquad \textbf{(1.2)}$$

1.2.2. Navegación por referencias externas

En este caso, el cálculo de la posición se realiza a partir de información enviada desde equipos instalados en tierra. También se la conoce como radionavegación.

Para calcular la posición instantánea basándonos en esta navegación utilizaremos la información recibida a través de ondas electromagnéticas. En función del tipo de radioayuda que estemos captando dispondremos de unos datos (distancia, ángulo), que combinados nos permitirán saber dónde se encuentra el vehículo.

Algunas de las radioayudas más conocidas son las siguientes:

- VOR (Very high frequency Omni Range): Proporciona guiado acimutal respecto al norte.
- DME (Distance Measuring Equipment): La información que proporciona esta radioayuda es simplemente la distancia total que la separa del vehículo.

- NDB (Non Directional Beacon): En este caso lo que se indica a bordo de la aeronave es la dirección en que se encuentra la baliza, de modo que el guiado que se proporciona es también acimutal.
- ILS (Instrumental Landing System): Es un tipo de radioayuda utilizada para aproximaciones de precisión en la aviación convencional, y que describe una senda de planeo y otras referencias para facilitar el aterrizaje al piloto.

1.2.3. RADAR

El uso del RADAR en ocasiones se incluye en la navegación por referencias externas ya que no es autónomo, y necesita información que va más allá del propio sistema, aunque es un tipo de "radioayuda" tan específico que se separa del resto.

El RADAR consiste en la emisión de una señal conocida que se refleja en el objetivo y es recibida de nuevo con un retraso determinado. En función de este tiempo que tarda en ser recibida la señal, se conoce la distancia a la que se encuentra nuestro *target*.

El uso más extendido es el que muestra la posición de la aeronave a un controlador que se encuentra en una base terrestre, y que comunica con el piloto vía oral para guiarlo hacia su destino.

Como en este trabajo nos centramos en el uso de MAVs, no hay piloto a bordo para interpretar la información, de modo que no nos centraremos en el uso del RADAR como sistema de navegación, sino que lo utilizaremos como un sistema de apoyo para el aterrizaje de nuestro vehículo, basado en el uso de un sensor de ultrasonidos. Otra posible utilización del este tipo de sensor podría ser detectar y evitar obstáculos que detectara. El funcionamiento de estos métodos se basa en los mismos principios ya explicados de RADAR.

1.2.4. GNSS

Existen varios GNSS (Global Navigation Satellite System), como el ruso GLONASS, otros que aún se han de poner en funcionamiento como GALILEO, de desarrollo europeo, pero el más conocido es sin duda el GPS.

Hoy en día se dispone de acceso libre a la red de satélites de GPS y su uso está de lo más extendido a pesar de sus limitaciones, como el hecho de que su disponibilidad depende directamente del gobierno de EE.UU.

El funcionamiento de todos estos sistemas se basa en la radiotelemetría. Este tipo de navegación consiste en obtener la distancia que separa el receptor de un mínimo de 3 satélites. Con estas tres distancias se puede calcular la posición del receptor, al realizar la intersección de las tres superficies esféricas que se obtienen de cada satélite.

Nos centraremos en la descripción del GPS, que es el que utilizaremos en el sistema de navegación para MAVs.

1.3. Control y estabilidad

La estabilidad se encarga de analizar el tipo de respuesta que da un sistema cuando sufre una perturbación que lo altera su condición de equilibrio inicial.

El control estudia la respuesta del sistema a las órdenes que se le dan a través de los mandos, cuando el controlador provoca una perturbación para llevar el vehículo de una condición de equilibrio inicial a otra.

1.3.1. Estabilidad Estática

La estabilidad estática es la tendencia inicial de un objeto para volver a una posición de equilibrio después de haber sufrido una perturbación que le haya alejado de dicha posición.

Existen tres tipos de estabilidad estática:

- Estabilidad estática positiva (estable)
- Estabilidad estática neutra (indiferente)
- Estabilidad estática negativa (inestable)

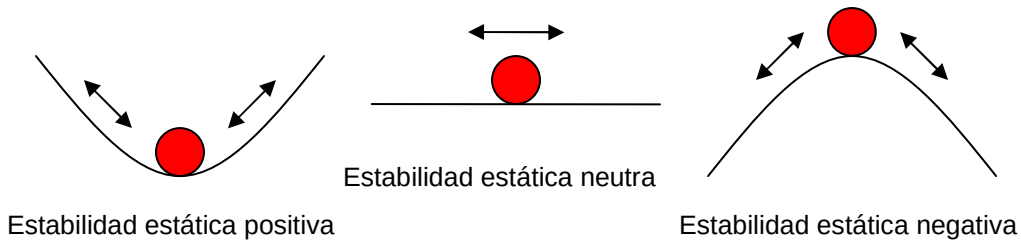

Fig 1.4 Tipos de estabilidad estática

1.3.2. Estabilidad Dinámica

La estabilidad dinámica es, en cambio, la tendencia final para volver a una posición de equilibrio, después de haber sufrido una perturbación. Es la forma que tiene el objeto de volver al equilibrio al cabo de un tiempo.

La estabilidad dinámica se opone al movimiento perturbador. Es lo que se conoce como amortiguamiento.

También existen tres tipos de estabilidad dinámica:

- Estabilidad dinámica positiva (estable)
- Estabilidad dinámica neutra (indiferente)
- Estabilidad dinámica negativa (inestable)

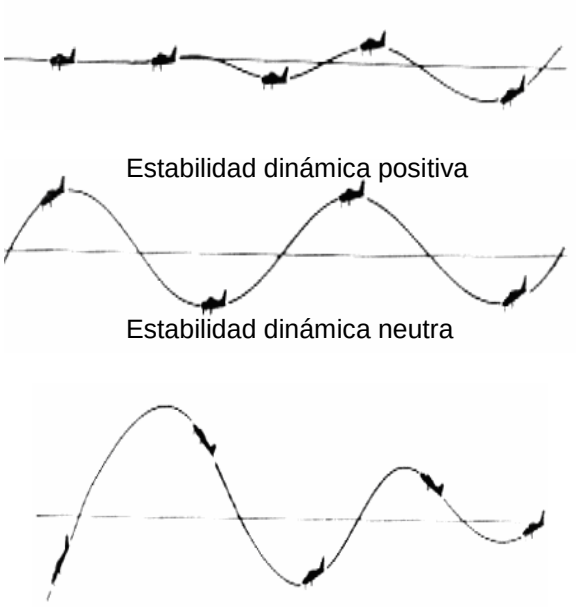

Fig 1.5 Tipos de estabilidad dinámica

1.3.3. Bases de control

Como ya se ha comentado anteriormente, el control estudia la respuesta del sistema a las órdenes que se le dan a través de los mandos, cuando el controlador provoca una perturbación para llevar el vehículo de una condición de equilibrio inicial a otra. Los sistemas de control pueden tener niveles de sofisticación muy variables, pero todos tienen puntos comunes.

Lo primero que hay que conocer es la situación inicial del vehículo, el punto de partida que queremos modificar, para ver precisamente como tendremos que actuar para que el resultado sea el que deseamos.

A partir de ahí, se compara la situación inicial que tenemos con la situación final que deseamos. La diferencia entre ambas es lo que tenemos que modificar de nuestro sistema.

El siguiente paso es actuar sobre el sistema para modificar sus condiciones y llevarlo a la posición final deseada. Una vez llegados a este punto sólo queda comprobar que realmente se ha alcanzado el objetivo haciendo otra lectura y comparándola con la situación deseada, y en caso de que no se correspondan se volvería a iniciar el proceso.

1.4. Objetivos del proyecto

El **objetivo** de este proyecto es diseñar una arquitectura hardware de navegación y control de un MAV y en general de cualquier otro vehículo aéreo, sonda o robot, de bajo coste y suficientemente escalable para que pueda utilizarse en diferentes plataformas.

Para conseguir un sistema de navegación y control aplicable a MAVs utilizaremos la mayoría de tecnologías hasta ahora mencionadas, cada una de ellas para resolver una necesidad concreta.

Para la navegación, utilizaremos un sistema GPS que nos mantendrá al corriente en todo momento de la posición del objeto, suministrando información de latitud y longitud.

Además de las coordenadas obtenidas por el GPS se utilizarán dos sistemas de apoyo para la navegación. Uno de ellos será un sensor de ultrasonidos (funcionamiento similar al RADAR) que se utilizará para conocer la altura y la distancia a posibles obstáculos con mayor precisión, ya que esta información es de gran importancia sobretodo en aterrizajes, y el GPS no la calcula con la misma precisión que la latitud o la longitud. El otro sistema de apoyo es un conjunto formado por dos acelerómetros que nos permitirán obtener información tanto de aceleración como de velocidad, lo cual nos permitirá prever la actitud de nuestro vehículo. No se llega a completar el sistema inercial calculando la posición, porque los acelerómetros integrarían errores demasiado grandes y los datos del GPS son más precisos.

El control de nuestro vehículo lo llevaremos a cabo mediante un mando a distancia sensible al movimiento, que transmitirá las órdenes de actuación a los motores de forma inalámbrica.

Como último punto, un sistema más a añadir que no está directamente relacionado con el control y la navegación pero que puede resultar extremadamente útil en determinadas circunstancias. Una cámara inalámbrica incorporada al vehículo que transmite la imagen en tiempo real. En caso de fallo en el sistema de navegación, el reconocimiento visual del entorno puede llegar a ser suficiente para poder devolver el vehículo al punto de origen o cualquier otra zona controlada.

CAPÍTULO 2. ARQUITECTURA DEL SISTEMA

2.1. Descripción general

Para el diseño del conjunto del sistema se distinguen dos partes fundamentales que se comunican de forma inalámbrica a través de la plataforma XBee. En el siguiente esquema (Figura 2.1) se muestran las diferentes partes que formarán el sistema.

- Se muestran en color azul los sensores, que miden cambios en diferentes magnitudes físicas.
- En verde los componentes que se encargan de procesar información.
- Los cuadros de color gris muestran todos los elementos que intervienen en la transmisión de datos.
- El componente de color naranja es el sistema actuador.
- En amarillo se muestra el componente controlador.

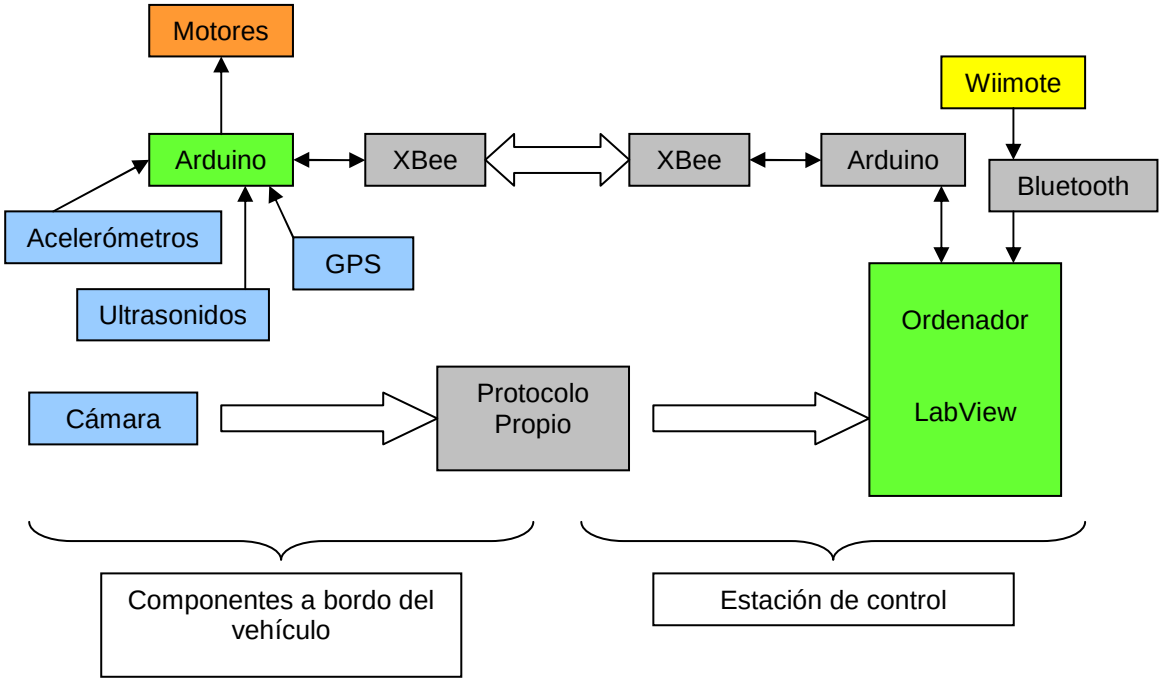

Fig 2.1 Arquitectura del sistema

El objetivo es poder crear una plataforma de bajo coste que permita interactuar entre el vehículo no tripulado y una serie de componentes gestionadas por el usuario que permitan controlarlo conociendo una serie de parámetros proporcionados por el propio vehículo. Al estudiar cada componente de esta arquitectura se indicará el precio de cada uno de ellos para tener siempre en cuenta que estamos hablando de un sistema de control y navegación de muy bajo coste y que es realmente asequible para aplicarlo a otros sistemas.

2.1.1. Componentes integrados a bordo del vehículo

Es la parte que forman todos los elementos que se encuentran incorporados al MAV. Se incluyen:

- Sensores que realizan las medidas correspondientes: Acelerómetros, sensor de ultrasonidos, antena GPS y cámara de vídeo.
- Módulo XBee para transmitir la información.
- Motores a través de los cuales actúa el controlador.
- Placa Arduino: Se encarga de adquirir los datos de los sensores. Incorporado en esta placa tenemos un microcontrolador de Atmel, que permite programar sus funciones para poder trabajar con los datos obtenidos.

Arduino es una plataforma de hardware libre, que permite llevar a cabo múltiples diseños, programándolos a través de su software, que se puede descargar gratuitamente de internet. Su diseño gira alrededor de un núcleo formado por un microcontrolador de Atmel, el ATMEGA 328. A parte de la capacidad de programar el microcontrolador para la correspondiente adquisición de datos, las características de la placa Arduino que debemos tener en cuenta para nuestras necesidades son que dispone de varios conversores A/D de 10 bits, y un fondo de escala de 5 V.

El hecho de ser libre, implica que puede utilizarse para implementar cualquier sistema sin necesidad de obtener licencias. El precio de cada una de estas placas se encuentra alrededor de los 40 €.

La gama de productos de Arduino es amplia, pero en nuestro caso utilizaremos la placa Arduino Duemilanove (Figura 2.2). Dispone de una serie de pins tanto de entrada como de salida, a través de los cuales se puede obtener información de sensores o conectar otros módulos adaptados a este tipo de placas. La alimentación se realiza a través del mismo puerto USB cuando está conectado al ordenador, o a través de una entrada de alimentación de 9 a 12 V.

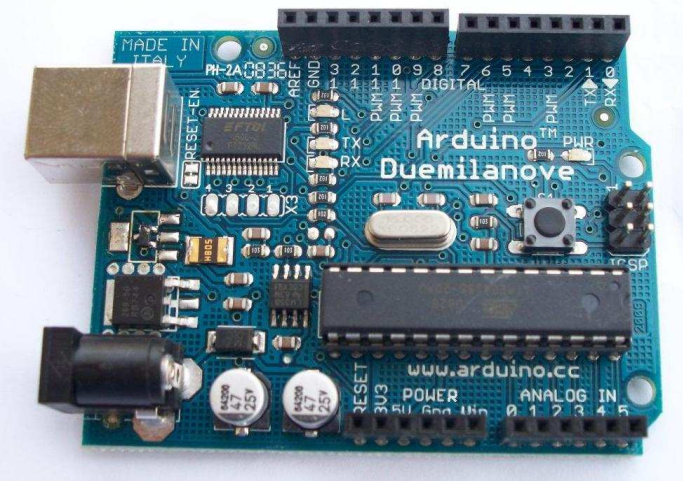

Fig 2.2 Placa Arduino Duemilanove

2.1.2. Estación de control

Son todos los componentes que se encuentran en tierra y trabajan con la información que reciben del vehículo. Son los siguientes:

- Módulo XBee para recibir y transmitir información
- Placa Arduino sin microcontrolador que permite que llegue la información al ordenador.
- El propio ordenador, que también forma parte de los elementos de control, ya que incorpora la mayor parte del software necesario para hacer una correcta gestión e interpretación de los datos obtenidos. El software se implementa en el programa LabView, de National Instruments.
- Mando de control. Un mando obtenido de la plataforma para juegos Wii, de Nintendo, que permitirá controlar el vehículo a distancia. Este mando actúa como un sensor, que transmite la información al ordenador vía Bluetooth.

CAPÍTULO 3. SISTEMA DE MEDIDA. SENSORES

3.1. Actitud del MAV

3.1.1. Acelerómetros

Dispondremos de dos acelerómetros como sensores para realizar las medidas correspondientes a bordo de nuestro vehículo. Sin embargo, en el sistema que nosotros vamos a estudiar, el uso de dichos acelerómetros no será enfocado a obtener valores de posición final ya que el resultado se ve muy afectado por los errores de integración y disponemos de un receptor GPS cuya información es más precisa.

Los datos que se obtendrán serán de aceleración y de velocidad ya que no necesitaremos integrar más que una vez y los posibles errores cometidos en ese caso son tolerables.

Para poder analizar los tres ejes, utilizaremos dos acelerómetros de dos ejes colocados formando 90º. De este modo vemos que las medidas en las coordenadas x, y, z se realizarán incluso con un dato redundante. Una posible solución es ignorar una de las medidas repetidas, o bien realizar una media de ambos valores para así dar utilidad a toda la información recibida.

Así pues, el proceso a realizar una vez leídos los datos consiste únicamente en mostrar la información de la aceleración directamente, e integrar en función del tiempo para obtener la velocidad:

$$v = \int a\, dt + v(0) \qquad \textbf{(3.1)}$$

Como ya se ha comentado, los errores acumulativos del sistema son el principal inconveniente de la navegación inercial, por lo cual se estima conveniente reiniciar la lectura cada cierto tiempo y así “limpiar” las medidas realizadas, aunque en este caso la integración sólo se realice una vez y por tanto los errores no sean tan significativos.

En este caso, dada la sencillez del sistema, no se incorporan giróscopos para estabilizar el sistema, así que el sistema de referencia respecto al cual se harán las medidas, serán los propios ejes con los que esté alineado el MAV.

Los datos obtenidos al realizar las medidas podrán ser introducidos a cualquier algoritmo de control PID para estabilizar el sistema. Este tipo de algoritmos se utilizan para corregir el error entre un valor medido y el valor deseado realizando la acción correspondiente, que puede ser multiplicar, integrar, derivar, o una combinación de ellas. El error se calcula mediante un sistema de control con realimentación.

3.1.2. MMA6270Q

Estos acelerómetros proporcionan una salida analógica en milivoltios, en función de la aceleración a la que es sometido cada uno de sus ejes. Se trata de acelerómetros capacitivos y tienen como característica destacada que su nivel de sensibilidad es seleccionable. Las especificaciones técnicas figuran en el anexo 1. El precio de este sensor es de aproximadamente 10 €.

Como ya se ha comentado, disponer de dos acelerómetros de dos ejes cada uno, obliga a posicionarlos en un ángulo de 90º. Para mantener ese ángulo constante, se he fabricado un soporte de aluminio en que se pueden sujetar firmemente ambos acelerómetros, y así disponer de la aceleración en los tres ejes, tal y como se muestra en la imagen siguiente.

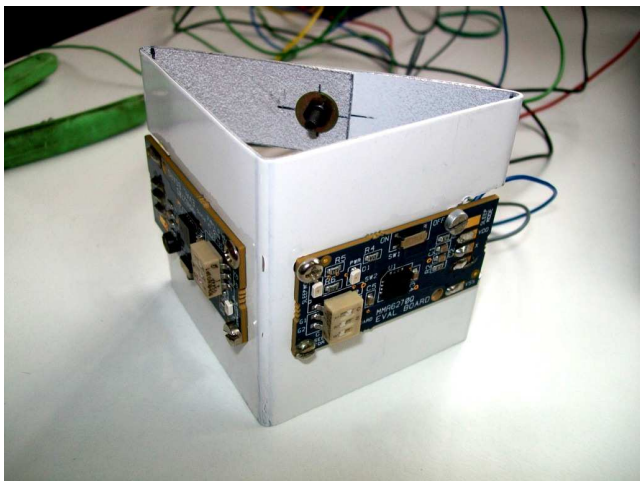

Fig 3.1 Soporte de acelerómetros en 90º

La salida de estos sensores se conecta directamente al conversor A/D del microcontrolador. En función de la combinación que hayamos seleccionado dispondremos de unos valores de sensibilidad que son:

- 800 mV/*g* para un rango de ±1,5 *g*
- 600 mV/*g* para un rango de ±2 *g*
- 300 mV/*g* para un rango de ±4 *g*
- 200 mV/*g* para un rango de ±6 *g*

El proceso de adaptar la señal de los acelerómetros se lleva a cabo en el propio microcontrolador. El margen de salida del sensor va de 0 V a 3 V, aunque útiles se miden 2,4 V. Este valor se obtiene de la relación entre la sensibilidad seleccionada y el rango de medidas asociado que se acaba de ver:

- 0,8 V/*g* · 3 *g* = 2,4 V
- 0,6 V/*g* · 4 *g* = 2,4 V
- 0,3 V/*g* · 8 *g* = 2,4 V
- 0,2 V/*g* · 12 *g* = 2,4 V

El fondo de escala de que dispone Arduino es de 5 V. A la entrada del microcontrolador, se dispone de un conversor A/D que permite utilizar 9 bits de los 10 de que dispone, ya que hemos perdido uno al no adaptar el margen dinámico de la señal a la entrada de la placa. El rango de medidas que se tiene para escalar la lectura de aceleración es de 2^9, es decir 512 valores. Esto nos deja una resolución eficaz que varía en función del rango seleccionado:

- 3 *g* / 512 = 5,86 m*g*
- 4 *g* / 512 = 7,81 m*g*
- 8 *g* / 512 = 15,63 m*g*
- 12 *g* / 512 = 23,44 m*g*

Hay que tener en cuenta que este tipo de sensor responde a la gravedad, es decir que no está calibrado para ignorar el valor del campo gravitatorio y por tanto aunque se encuentre quieto sufrirá una aceleración de 1 *g* en el eje que se encuentre en posición vertical. Este es una característica que hay que tener presente a la hora de gestionar las medidas que se realicen a partir de los acelerómetros.

Una vez la señal está adaptada, en la parte de adquisición de datos se verá como los valores se escriben en el puerto serie sin modificar.

3.2. Posición

3.2.1. GPS

Contar con una unidad GPS nos permite conocer con gran precisión, el emplazamiento del vehículo en el espacio, obteniendo los datos de latitud y longitud (cómo ya ser verá en el apartado siguiente, la altura o distancia a obstáculos cercanos se obtiene a partir de un sensor de ultrasonidos).

Algunas de las principales características de este sistema de posicionamiento por satélite son las siguientes:

El GPS está formado por tres segmentos:

- Segmento espacio: Formado por la constelación de satélites en órbita.
- Segmento control: Las estaciones situadas en tierra que permiten gestionar las señales enviadas.
- Segmento usuario: Receptor que obtiene la posición final.

El funcionamiento de este sistema se basa en la telemetría. Los satélites del segmento espacio emiten una onda que tarda un tiempo determinado en ser recibida por la antena.

Conociendo la velocidad a la que viaja dicha onda (aproximadamente la velocidad de la luz), podemos obtener la distancia a la que se encuentra nuestro receptor del primer satélite.

Con un solo satélite, la única información que tenemos es que la antena receptora se encuentra en algún punto ubicado en la superficie esférica que rodea dicho satélite y cuyo radio es igual a la distancia obtenida.

Con la medida del segundo satélite, la información pasa a ser más concreta, ya que la ubicación real se encuentra en la intersección de las dos superficies esféricas que obtendríamos ahora, es decir, una circunferencia.

Así que es con la medida obtenida del tercer satélite (y considerando que el vehículo debe encontrarse cercano a la superficie terrestre) con la que obtendríamos un punto con unas coordenadas concretas de latitud y longitud (Figura 3.2).

A pesar de todo, con sólo tres satélites detectados la información podría no ser suficientemente exacta, así que cuantos más satélites se puedan utilizar de los 24 que hay en órbita, para realizar la medida, más preciso será el resultado.

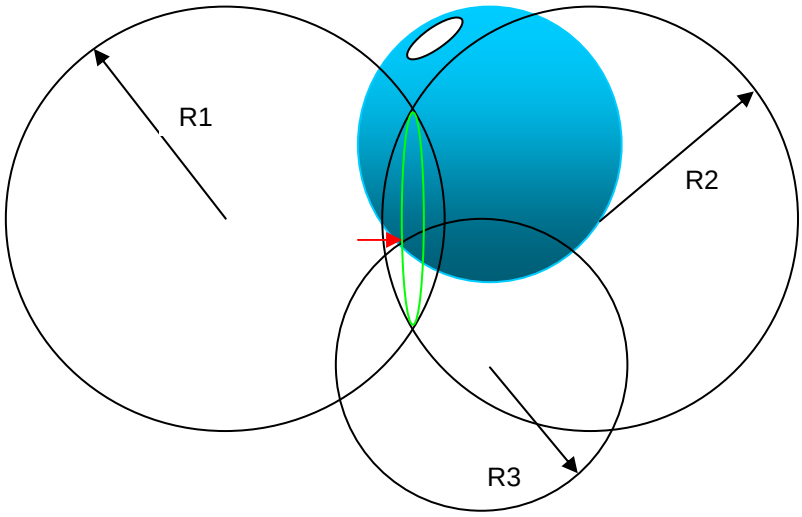

Fig 3.2 Intersección de tres satélites GPS

Con el tiempo, el uso del GPS se ha extendido a gran escala, y algunos procedimientos en los que se usa el GPS (p. ej. análisis topográfico) requieren gran precisión en las medidas realizadas, de manera que hoy existen muchos sistemas de aumento que se han ido perfeccionando para que el error que se puede aspirar a cometer sea de unos pocos centímetros.

Además de contar con la lectura de GPS estándar de latitud y longitud, se utilizará una interfície más amigable como es Google Earth para mostrar de forma gráfica el movimiento que sigue el vehículo. Su funcionamiento se explica en el apartado de software de procesado.

3.2.2. Glonass-07

Esta antena de pequeñas dimensiones permite ser incorporada con facilidad en cualquier sistema de modo que queda perfectamente integrada. Un pequeño módulo la adapta para pasar la información a la placa Arduino.

Además del tamaño reducido también tiene la característica de estar magnetizada en la base, de modo que permite una firme fijación a superficies metálicas sin necesidad de soportes añadidos. Se puede obtener en el mercado por un precio de unos 40 €.

Para transmitir los datos GPS existen varios protocolos, y esta antena da la información en varios bloques de datos. En nuestro caso, la línea que utilizamos es una que sigue el protocolo NMEA, iniciada por las siglas GPGGA. En este protocolo, cada elemento de la línea tiene su significado que se debe analizar por separado como se muestra en el siguiente ejemplo:

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

- $GPGGA: Tipo de sentencia. Global Positioning System Fix Data
- 123519: Hora GPS. 12:35:19
- 4807.038,N: Latitud. 48º 07.038’ Norte
- 01131.000,E Longitud. 11º 31.000’ Este
- 1: Calidad de la señal. 0 = Inválida, 1 = GPS, 2 = DGPS
- 08: Número de satélites captados
- 0.9: HDOP. Referencia del nivel de precisión horizontal
- 545.4,M: Altura sobre el nivel del mar. 545.4 metros
- 46.9,M: Altura medida en geoide sobre el elipsoide modelo WGS84
- *47: Número de chequeo de errores

Una vez se dispone de toda la información necesaria seleccionada, ya se puede trabajar con cada uno de los elementos, aunque previamente hará falta descomponerla para tener cada dato por separado.

Mientras la antena espera la convergencia que le permita calcular la posición en que se encuentra a través de los satélites, las coordenadas que se muestran son las de un punto de referencia cerca de la ciudad sueca de Malmö. Esta convergencia puede tardar desde unos pocos minutos si nos encontramos en campo abierto y cerca de la última posición captada por la antena, hasta más de una hora en caso de que nos encontremos en una zona rodeada de edificios que impidan una buena recepción de la señal.

El sensor está formado por la antena GPS que actúa como receptor y por un módulo añadido de la marca Libelium, que permite la conexión entre la antena y Arduino, y realizar las conexiones de alimentación de 5 V y tierra. Los datos son adquiridos por el microcontrolador y se transmiten vía serie para su posterior procesado. En este caso, para realizar la gestión de la señal GPS, es necesario inicializar un puerto serie extra en la misma placa de Arduino. En el apartado de adquisición de datos se verá que se declara en los pins 10 y 11 y como se gestiona.

3.3. RADAR

3.3.1. Ultrasonidos

Para llevar a cabo un control de altura más exhaustivo que el que nos proporciona el GPS, en los aterrizajes se dispondrá de un sensor de ultrasonidos que nos permitirá conocer la distancia al obstáculo más cercano con mayor precisión.

El funcionamiento básico de estos sensores consiste en emitir un pulso a una frecuencia alrededor de los 40 kHz (superior a los sonidos perceptibles por el oído humano, de ahí el término ultrasonido), y medir el tiempo que tarda en rebotar en el objeto más próximo (Figura 3.3). El campo de acción tiene forma cónica.

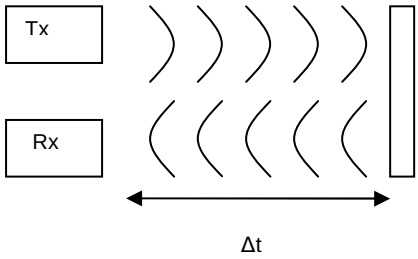

Fig 3.3 Funcionamiento ultrasonidos

Una vez medido el tiempo de retorno, las leyes de la física básica nos permiten relacionar este intervalo con la distancia a la que se encuentra el objeto siempre que conozcamos la velocidad de propagación de la onda en el aire.

$$d = \tfrac{1}{2} \cdot v \cdot \Delta t \qquad \textbf{(3.2)}$$

3.3.2. LV-MaxSonar-EZ0

El sensor utilizado en este caso será un LV-MaxSonar-EZ0 de 43 gramos de paso (Figura 3.4), cuyas especificaciones técnicas figuran en el anexo 2. Permite tres tipos de salida: analógica, digital o a través de pulsos.

Este sensor puede realizar medidas que pueden ir de los 14 cm, por debajo de los cuales no mide cambios de distancia, hasta los 6.45 m. Realiza mediciones cada 50 ms (20 Hz). Requiere una tensión de alimentación de 5 V y su precio ronda los 15 €.

En este caso utilizaremos la salida que se basa en la medición de pulsos (PWM), ya que la salida digital se realiza por el puerto serie y este está

ocupado por la placa XBee, del mismo modo que los pins analógicos que son los que leen la información de los acelerómetros.

La medición a través de la PWM tiene una resolución de 147 µs/pulgada, que será la que se utilice para obtener el valor de distancia real, y se realiza en la banda de 42 kHz. Para obtener unas medidas que se correspondan con la realidad, es importante que al principio de su uso, el sensor esté lo más libre de obstáculos posible, ya que utiliza el primer ciclo de medición para calibrarse.

Los datos obtenidos con el sensor se traducen en la misma placa Arduino utilizando la función *micros* como se verá en el apartado de software de adquisición de datos. En este caso, el sensor incorpora un microcontrolador que se encarga de realizar el acondicionamiento de la señal.

Cuando se realizan medidas con ultrasondos, estas se ven afectados por la temperatura ambiente. Existen sistemas de compensación del posible error que se comete, pero en este caso no las aplicaremos ya que no afecta al tipo de medidas que se van a realizar.

Fig 3.4 LV-MaxSonar-EZ0

3.4. Soporte visual

3.4.1. Cámara

A nivel de navegación, el hecho de contar con una cámara no tiene una utilidad inmediata, sin embargo para poder realizar el control más preciso a través del mando “Wiimote” puede ser de utilidad establecer de forma visual cual es la situación en que se encuentra el propio vehículo para poder actuar al respecto y que el propio controlador pueda tomar las medidas correspondientes.

Del mismo modo, el uso de una cámara permite observar en tiempo real lo que está ocurriendo alrededor de nuestro vehículo, lo cuál nos permite obtener información del entorno, así como detectar posibles amenazas para nuestro vehículo.

Para interferir lo menos posible en las condiciones de vuelo, la cámara deberá ser del menor tamaño posible y estar bien ensamblada en el cuerpo de la aeronave. Lo propio en este caso es buscar cámaras de espionaje porque son las que mejor se adaptan a nuestros requisitos.

3.4.2. WS-007 AS

El sistema utilizado es una cámara del tipo WS-007 AS con su correspondiente receptor inalámbrico. Sus dimensiones son realmente reducidas con lo cual no debe presentar graves problemas a la hora de incorporarla al MAV (Figura 3.5). La cámara cuenta con una alimentación independiente proporcionada por una pila de 9 V y un adaptador para cable coaxial, mientras que el receptor se alimenta con un transformador a 12 V. El precio de la cámara junto con el receptor es de aproximadamente 40 €.

Es capaz de transmitir video y sonido con un alcance que puede llegar a los 250 m en campo abierto, trabaja en la banda de frecuencia de 1,2 GHz y con una frecuencia de escaneo de 50 Hz. La imgen que proporciona es de 628x582 píxeles en formato PAL.

Fig 3.5 WS-007 AS y receptor

Para recibir la imagen en el ordenador, se requiere un sistema que permita conectar la salida del receptor a un USB 2.0, en este caso, uno de la casa EasyCap.
Como ya se ha mostrado en el capítulo de “Arquitectura del sistema”, la comunicación inalámbrica que se establece entre la cámara y el receptor se realiza de forma independiente a XBee, a través de un protocolo propio en la banda de 1,2 GHz.

CAPÍTULO 4. COMUNICACIÓN

4.1. Comunicación serie / USB

Para establecer un sistema de comunicación que permita gestionar una señal externa desde el PC, se utilizará una conexión a través del puerto serie que incorpora Arduino.

Para sincronizar el puerto serie, en primer lugar debemos poder recibir y transmitir desde LabView. Existen varios ejemplos en los tutoriales que incorpora el programa. Los únicos valores necesarios que hay que indicar son:

- El nombre del puerto a utilizar (COM): Que se puede obtener en el Administrador de Dispositivos del Panel de Control de Windows.
- La frecuencia de transmisión: En este caso 19200 baudios, para que el ritmo de lectura del puerto serie sea el mismo al que se escribe en él.

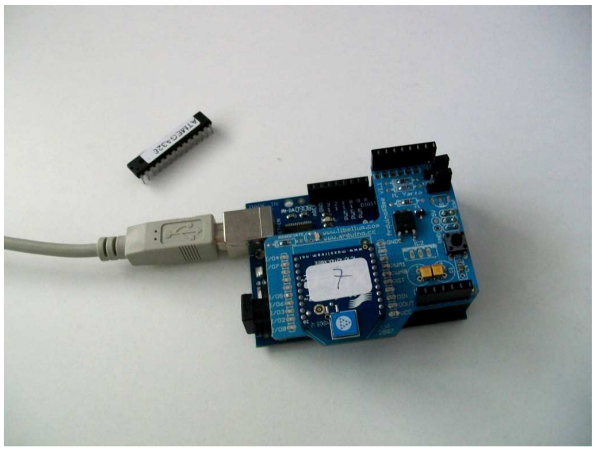

Fig 4.1 Placa Arduino con XBee y sin microcontrolador

Para esta conexión, se dispondrá de una placa Arduino sin necesidad de microcontrolador a través de un puerto USB (Figura 4.1). El hecho de no utilizar microcontrolador se debe a que esta placa no es la que gestiona la información de los sensores, sino que es el punto intermedio para poder conectar XBee y conseguir la comunicación inalámbrica. Quitando el microcontrolador se evita que se sobrescriba.

4.2. Transmisión a través de XBee

XBee es un sistema de comunicación inalámbrica que utiliza el sistema de comunicación Zigbee, fabricado por MAXTREAM, y que es compatible con Arduino.

Su funcionamiento se basa en la lectura y escritura en el puerto serie (UART) que incorpora Arduino. Cuando la placa XBee se encuentra añadida a la de Arduino, interpreta la información del puerto serie y la transmite inalámbricamente en la banda de 2,4 GHz, para que otra placa pueda leerla. El alcance de este tipo de comunicación llega desde los 30 metros en interiores hasta 100 metros al aire libre. El protocolo de comunicación que utiliza es el 802.15.4. Cada módulo XBee tiene un precio alrededor de 20 €.

Cada placa XBee incorpora dos “jumpers” para indicar por que ruta gestionará la información: USB o XBee. En el caso que nos incumbe, la placa de conectada al PC debe llevar los “jumpers” en posición USB, mientras que la que se incorpora a bordo del vehículo los debe llevar en la posición “XBee”. Toda la información se gestiona a través del puerto serie.

4.2.1. Plataforma Arduino

Al disponer de la placa Arduino completa, con todos los sensores incorporados y la gestión de los datos correspondiente realizada, el paso que corresponde es escribir las líneas de información en el puerto serie, para que sea la plataforma XBee quien se encargue de transmitirlos (Figura 4.2).

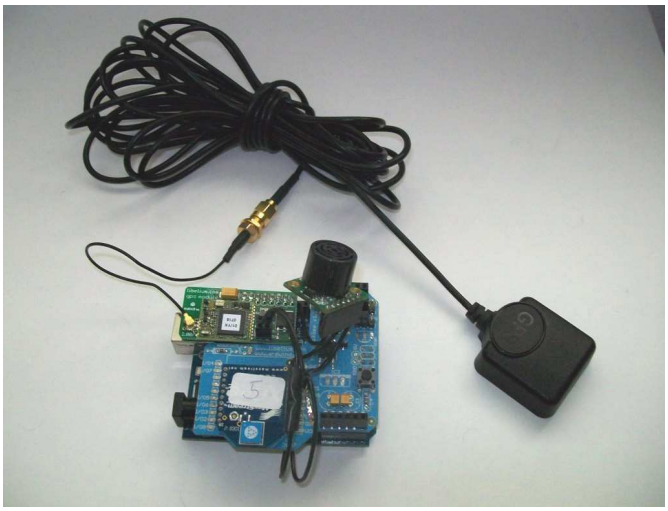

Fig 4.2 Placa Arduino con sensores y XBee

Para que el funcionamiento sea el correcto, se deben combinar los pins de que dispone Arduino, para poder compaginar todos los sensores, con sus correspondientes entradas de alimentación, y la placa XBee montada.

En caso de incompatibilidad, se pueden realizar empalmes de cable y uniones que permitan llegar a un resultado satisfactorio.

El programa incorporado al microcontrolador para gestionar la información de los sensores y leer el mando de control se incorpora en el Anexo 4.

CAPÍTULO 5. SOFTWARE DE ADQUISICIÓN Y PROCESADO

Como ya se ha visto en el capítulo “Arquitectura del sistema”, hay dos partes del conjunto del sistema que incorporan software: Una a bordo del vehículo que se encarga de la adquisición de los datos y transmisión, y otra que se encarga de gestionar la información que recibe y mostrarla por pantalla.

En la parte incorporada a bordo del vehículo se pretende realizar la mínima gestión necesaria para que la información pueda ser transmitida lo más rápidamente posible a la estación de control, donde se lleve a cabo la mayor parte del procesado.

5.1. Software de adquisición. Arduino

En primer lugar se analiza la componente que se encuentra a bordo del vehículo y que obtiene la información para realizar un primer procesado de los datos a través del microcontrolador Atmel.

A excepción de la cámara, el resto de sensores pasa la información a Arduino, que los gestiona por separado, por tanto hay una componente de software destinada a cada sensor:

- Acelerómetros:

```
int pinPortX1 = 2;
int pinPortY1 = 3;
int pinPortX2 = 4;
int pinPortY2 = 5;
void setup(){
Serial.begin(19200)}
void loop(){
  int valueX1=0;
  int valueY1=0;
  int valueX2=0;
  int valueY2=0;
  valueX1=analogRead(pinPortX1);
  valueY1=analogRead(pinPortY1);
  valueX2=analogRead(pinPortX2);
  valueY2=analogRead(pinPortY2);
    Serial.print(valueX1);
  Serial.print(",");
  Serial.print(valueY1);
  Serial.print(",");
  Serial.print(valueX2);
  Serial.print(",");
  Serial.print(valueY2);
}
```

La adquisición de datos que se realiza de los sensores acelerómetros es la más básica, que consiste en asignar los pins de entrada de cada uno de los cuatro ejes que se van a medir, asignar un valor a la medida que realiza cada uno de ellos, y escribirlos en el puerto serie, intercalando una coma entre cada uno de estos valores.

- Antena GPS:

```
#include <SoftwareSerial.h>
#define rxPin 11
#define txPin 10
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);
byte byteGPS = 0;
int i = 0;
int indices[13];
int cont = 0;
int conta = 0;
char inBuffer[300] = "";
int k = 0;
void setup(){
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  mySerial.begin(4800);
  Serial.begin(19200);
  Serial.println("Configurando GPS...");
  delay(1000);
  mySerial.println("$PSTMNMEACONFIG,0,4800,1,1");
  delay(100);
mySerial.println("$PSTMINITGPS,4140.000,N,00053.000,W,0197,
22,10,2007,11,40,00");}
void loop(){
  byteGPS = 0;
  i = 0;
  while(byteGPS != 42){
    byteGPS = mySerial.read();
    inBuffer[i]=byteGPS;
    i++;                      }
  k = 1;
  while(inBuffer[k] != 42){
    Serial.print(inBuffer[k]);
     k++;
  }
}
```

Para adquirir los datos de la antena GPS se realizan varias funciones, empezando por declarar un nuevo puerto serie en la placa Arduino en los pins 10 y 11, a través del cual se interactuará con la antena a una velocidad de 4800 baudios. Una vez inicializado el nuevo puerto serie, se inicializan los valores iniciales de latitud y longitud que muestran unas coordenadas fijas mientras se recibe señal de los satélites. La gestión de la información que se

realiza en la placa es la mínima, teniendo en cuenta que toda la información obtenida se presenta como un único bloque mostrado directamente desde la antena receptora.

El último paso consiste en escribir los datos leídos en el puerto serie añadido, por el puerto serie que incorpora la propia placa inicialmente.

- Ultrasonido:

```
unsigned long micro1,micro2;
unsigned long ultr;
int pinPortUS = 9;
void setup(){
Serial.begin(19200);}
void loop(){
int value=0;
  value=digitalRead(pinPortUS);
  while (value==HIGH){
    value=digitalRead(pinPortUS);}
  while(value==LOW){
    value=digitalRead(pinPortUS);
    micro1 = micros();}
  while(value==HIGH{
    value=digitalRead(pinPortUS);
    micro2 = micros();}
  ultr = ((micro2 - micro1)/147.0)*2.54;
  Serial.println(ultr);
}
```

Para realizar las medidas de ultrasonidos, se utilizará una función que incorpora Arduino llamada micros.

Esta función permite capturar un instante de tiempo determinado en unidades de microsegundos. En el primer caso en que la utilizamos, el valor que registramos corresponde a la última lectura en que el valor del pulso se ha leído como bajo (no se recibe el rebote de la señal), que se corresponde con el primero en que se lee un valor alto. Este valor queda almacenado como “micro1”. A partir de este punto, el valor de la señal recibida se mantiene alto hasta que termina el rebote. Este último instante en que se recibe valor el retorno del pulso es el que queda almacenado como “micro2”.

Una vez tenemos el valor de los dos instantes de tiempo, lo único que queda por hacer es restarlos para obtener el intervalo total que ha tardado en rebotar el pulso y hacer el cambio de unidades para obtener la distancia a la que se encuentra el objeto, en función de la resolución del sensor:

- Sensibilidad del sensor: 147 µs/pulgada
- 1 pulgada = 2,54 cm

Este es el único elemento sensor cuya información es mínimamente procesada a bordo del vehículo.

Cuando se obtiene este valor sólo queda escribirlo en el puerto serie para que XBee se encargue de transmitirlo.

5.2. Software de procesado. LabView

El programa LabView es una herramienta de National Instruments de gran utilidad en muchos campos de la ingeniería, especialmente valorado por lo intuitivo y gráfico que resulta a la hora de desarrollar programas relativamente complejos. La primera versión se mostró al público en 1986 y sólo estaba destinada a MAC. Actualmente existen muchas versiones para todo tipo de sistemas operativos, siendo la más reciente la 2009 que incorpora las últimas mejoras llevadas a cabo por NI.

En los casos en que los datos que se van a utilizar no son genéricos, es decir, se obtienen de una serie de sensores concretos, el programa permite incorporar librerías externas al programa base, en ocasiones desarrolladas por otros usuarios. En nuestro caso, se han tenido que incorporar librerías y "juegos de herramientas" para algunos de nuestros sensores como IMAQ para poder utilizar herramientas de visión, o para poder utilizar el controlador Wiimote. De estas librerías existe una gran variedad para poder interactuar con sistemas externos a los de NI y la inmensa mayoría se encuentran publicados en internet con acceso libre.

Como hasta ahora se he ido definiendo, la obtención de los datos se lleva a cabo a bordo del MAV en que se incorpora la placa Arduino, ésta transmite la información a través de XBee, y es recibida a través de otra placa en el ordenador. Es en este punto en que se utilizará LabView, tanto para gestionar como para presentar los datos obtenidos.

Dentro de LabView son fácilmente reconocibles dos pantallas que se conocen como diagrama de bloques y panel frontal. En el diagrama de bloques se realiza la programación en sí. De forma intuitiva permite conectar entradas con herramientas de gestión y modificar a nuestro antojo toda la información obtenida. En el panel frontal, en cambio, lo que se tiene es la imagen que se va a mostrar al usuario, con sus correspondientes indicadores, así como los controles que se permite que manipule dicho usuario.

Durante el desarrollo del programa se permite observar tanto el panel frontal como el diagrama de bloques (aunque no interferir en este último). Del mismo modo existe un sistema de depuración para ver como se gestionan todos los datos paso a paso, herramienta muy útil en caso de dudas o que se quiera analizar un fallo en el sistema.

Para obtener información del puerto serie debemos introducirle los valores típicos y elementos que suelen ser necesarios para este tipo de interacción: Puerto en el que está conectado para que lo reconozca, velocidad de

transmisión, bits de información, paridad, stop, si lo que queremos es leer o escribir, y designar cuando cerrar el puerto al finalizar su uso.

En el anexo 3 se presenta el panel frontal y el diagrama de bloques utilizados en el programa final.

Hasta ahora hemos visto únicamente como se obtiene la información y se comunica, pero en muchas ocasiones esa información puede encontrarse en un formato inadecuado para nuestros intereses o sencillamente no corresponderse con la realidad si no se aplican los cambios adecuados en cada caso. Ahora se verá como utilizando LabView se pueden adaptar las señales obtenidas por los sensores y transmitidas a través de XBee, para mostrar unos resultados útiles para llevar a cabo el control y la navegación de nuestro MAV.

Toda la información obtenida a través del puerto serie, por tanto transmitida por XBee, llega en un solo bloque de información, que hay que leer a través de LabView. La lectura de puerto serie implica conocer los datos como el puerto COM que se está utilizando y la frecuencia a la que se transmite en el origen, y estos son los datos que hay que el usuario deberá proporcionar antes de iniciar el programa (Figura 5.1 y Figura 5.2).

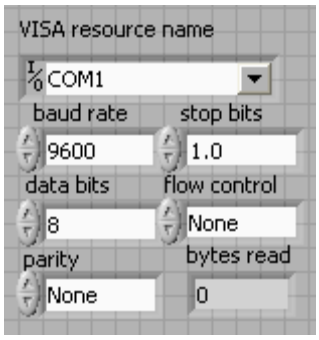

Fig 5.1. Selección de puerto serie en el panel frontal.

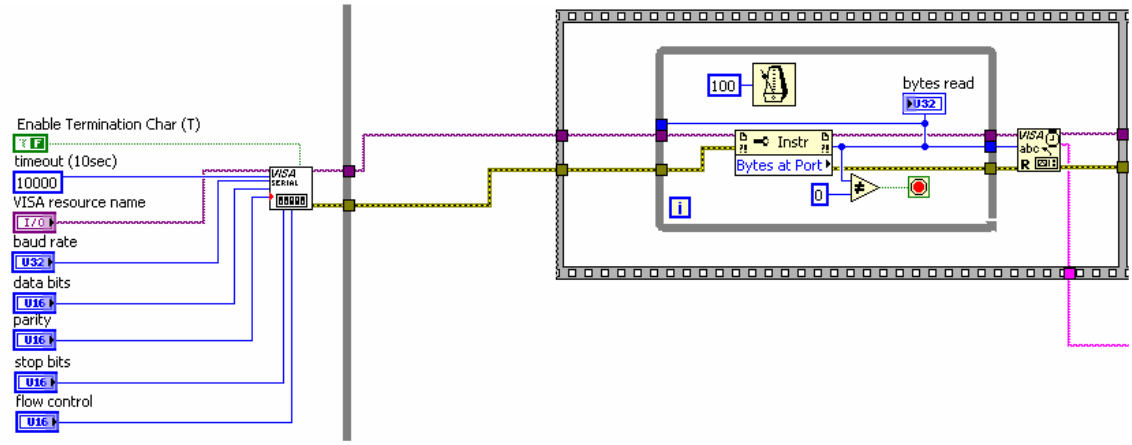

Fig 5.2. Adqusición del puerto serie en el diagrama de bloques.

5.2.1. Extracción de la información. Acelerómetros

La información que se obtiene por los sensores acelerómetros es suministrada directamente a través de XBee sin ninguna modificación a bordo del vehículo, y por tanto es recibida y totalmente gestionada en LabView (Figura 5.3).

El primer paso a realizar es identificar la línea de datos que contiene la información de los acelerómetros y extraerla. En nuestro caso, la hemos definido previamente como una línea de información con una letra A al inicio. Dentro de esta misma línea se encuentra también la información de los ultrasonidos, por tanto debemos tener claro el formato de dicha línea para poder separarlos adecuadamente.

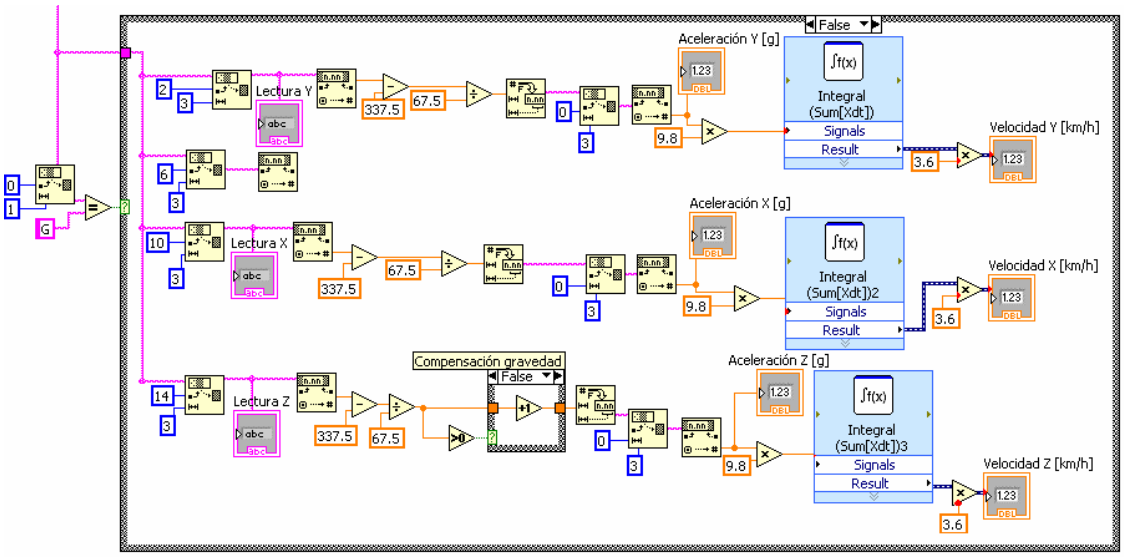

Fig 5.3. Gestión de acelerómetros en el diagrama de bloques.

Para ello se utiliza una estructura del tipo “Case” y como condición introducimos que el primer valor de la línea recibida sea una letra G (que es el primer valor de la línea de GPS). Teniendo en cuenta que solamente llegan dos líneas distintas, una empieza por G y la otra por A, los datos de aceleración y ultrasonidos cumplirán la condición de “False” dentro de esta estructura.

Al obtener la línea entera, que llega en el formato de una cadena de caracteres o “string”, debemos separar ahora la información de aceleración de la de ultrasonidos. Esto nos lo permite la herramienta de “string subset” a la cual debemos introducir el valor de la posición que ocupa la primera componente de nuestro valor dentro de la cadena de caracteres (o string) y la longitud de este. Una vez hecho esto, obtenemos los componentes cada uno independientemente del resto.

Una vez obtenidos los datos por separado debemos transformar su formato: es decir, pasar de una cadena de caracteres a un número. Este paso es directo gracias a la herramienta “Fract/Exp String to Number”. A partir de aquí ya

podemos operar con ellos, es decir, aplicarles los factores de conversión necesarios que se obtienen de las características del propio sensor. Para ello se pueden utilizar varios métodos, aunque el más fácil y rápido de entender es el que realiza las operaciones matemáticas paso a paso. Se dispone de iconos matemáticos estándar (+,-, * y /) en la paleta de “Mathematics” del diagrama de bloques. Otros sistemas permiten construir fórmulas, lo cual resulta muy útil en cálculos complejos, pero en nuestro caso no merece la pena.

Finalmente sólo queda mostrar por pantalla el resultado de los valores de aceleración ya corregidos, con un indicador estándar en el panel frontal. El panel frontal ofrece una gran variedad de elementos para mostrar información al usuario, hay que saber elegir el más adecuado en cada caso, así como los embellecedores correspondientes para hacer que la interfície sea lo más amigable posible.

5.2.2. Obtención de la velocidad

Una vez obtenida la aceleración el único paso que hay que añadir es la integración para llegar a la componente de velocidad.

LabView utiliza un sistema de integración en función del tiempo, que básicamente consiste en realizar pequeños sumatorios a gran velocidad. La función que hay que utilizar es: Integral (Sum [Xdt])

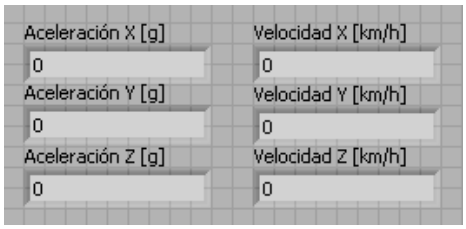

Fig 5.4 Presentación de datos de aceleración y velocidad

A la salida de este icono ya se tiene el valor de la velocidad y del mismo modo que antes sólo falta otro indicador para poder mostrar los resultados finales por pantalla (Figura 5.4). No hay que olvidar los posibles cambios de unidades derivados del cálculo de aceleración en [g] o en [m/s^2], así como la compensación de la gravedad.

5.2.3. Extracción de la información. GPS

La información de GPS que llega a través de la recepción XBee está siguiendo el formato GPGGA anteriormente explicado. Por ello, la línea formada es fácilmente identificable al iniciarse con una letra G. Esta es la condición que se debe cumplir para iniciar la manipulación de la información GPS.

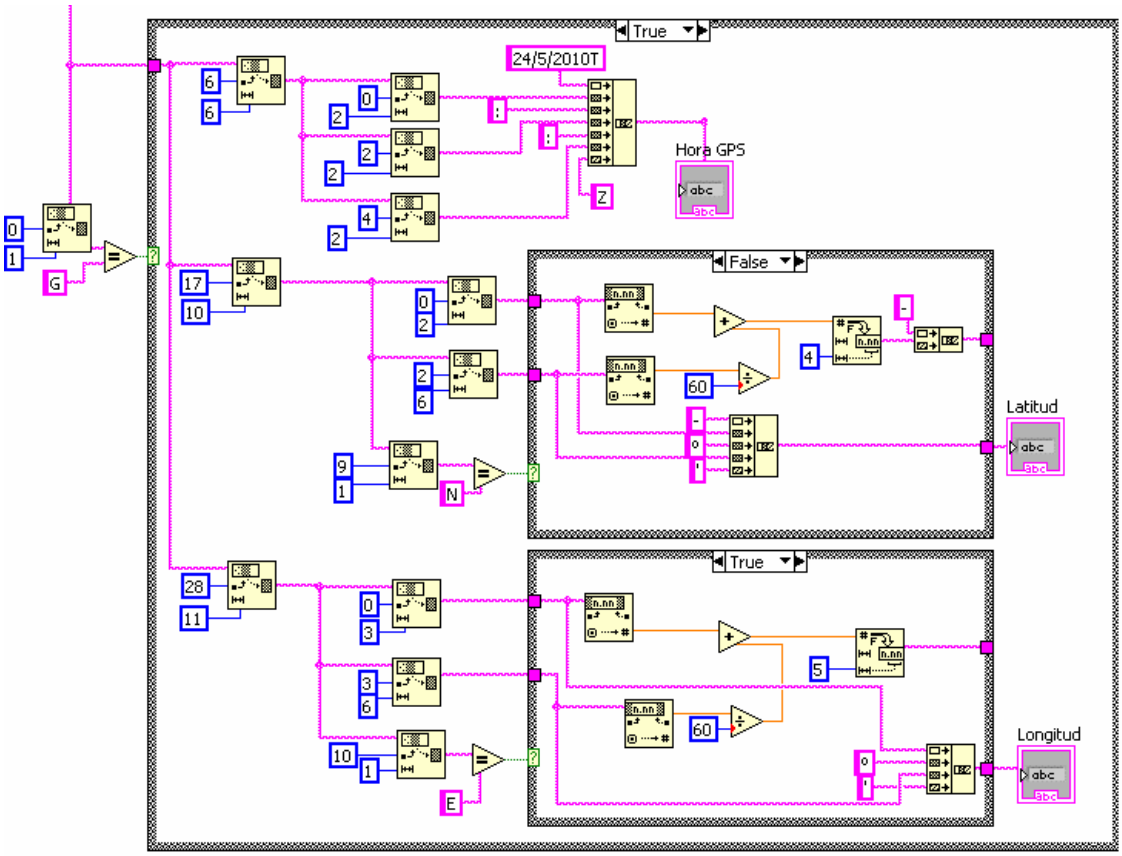

Fig 5.5 Gestión de GPS en el diagrama de bloques

Dado que en la misma línea se adjunta toda la información del GPS lo único que debemos hacer es descomponerla usando el "String Subset" indicando el origen y tamaño de cada uno de los elementos que nos interesa. De esta forma obtenemos: Hora GPS, Latitud y Longitud (Figura 5.5). Aquí encontramos un problema con los formatos, ya que el tiempo se presenta como horas, minutos y segundos de forma ininterrumpida sin ninguna separación, y la latitud y longitud con grados y minutos. El nuevo formato al que lo adaptaremos es el adecuado para que Google Earth lo pueda interpretar correctamente, este es el siguiente:

- Tiempo: 12/05/2010T13:55:22Z
- Latitud: 41.6666 (con un signo menos si es Sur)
- Longitud: 2.16175 (con un signo menos si es Oeste)

Para obtener los signos adecuados se utilizará otra estructura del tipo "Case" en que se comparará la letra de la línea obtenida con una S o W en cada caso. Si coincide, se añadirá el "–" antes de cada valor final.

El formato GPGGA da la información de latitud y longitud considerando los minutos de ángulo, en vez de decimales de grado. Esto nos obliga a crear otro "substring" para obtener los minutos de forma independiente del resto del valor, cambiarlos de formato y volverlos a añadir a la medida original. Lo mismo

ocurre con la hora, la cual nos obliga a añadir los dos puntos, la T y la Z intercalados en el valor original recibido. Cuando los datos se encuentran en el formato correcto ya se pueden presentar por pantalla (Figura 5.6).

Fig 5.6 Presentación de datos de GPS

El siguiente paso es conseguir guardarlos en archivos .txt (Figura 5.7). Para ello, LabView nos ofrece la posibilidad de almacenarlos en la ubicación específica que nosotros le asignemos del disco duro del ordenador, y con el nombre que nosotros le introduzcamos. Los archivos de texto resultantes deben quedar del siguiente modo:

- ltime.txt: En el cual se almacena la fecha y la hora en el formato ya comentado.
- position: Dónde se guardan los valores de posición en formato Longitud, Latitud respetando los parámetros de decimales anteriormente explicados.
- previous: En el que se almacenarán los valores de posición anteriores a cada instante analizado. Este punto se crea para que Google Earth pueda crear una línea de recorrido formada por pequeños segmentos. Para crear cada uno de estos segmentos necesita dos puntos distintos en cada instante de tiempo que se tiene en cuenta.
- heading: En nuestro caso introduciremos un valor constante ya que no tendremos en cuenta el valor de rumbo en cada instante.

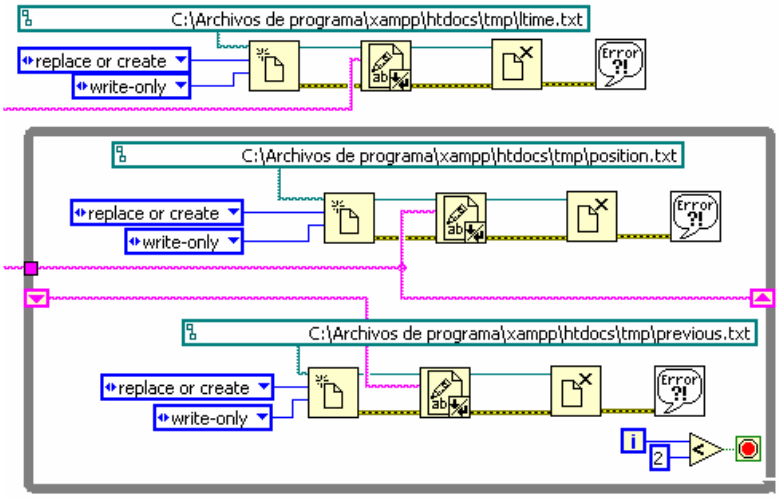

Fig 5.7 Guardar datos del GPS

Para poder visualizar los puntos obtenidos por el GPS en Google Earth hay que crear un pequeño servidor local a través del programa Xampp, fácil de obtener gratuitamente por internet. Una vez instalado este programa, la ubicación de los archivos de texto creados en LabView será la siguiente:

C:\Archivos de programa\xampp\htdocs\tmp

Y a partir de aquí lo único que queda es obtener unos archivos específicos que puede leer Google Earth en formato .php (Anexo 5) y que se deberán guardar en la ubicación siguiente:

C:\Archivos de programa\xampp\htdocs

Una vez llevados a cabo estos pasos, para poder ejecutar el sistema, los pasos a seguir son los siguientes:

- Iniciar Xampp.
- Activar (botón Start) “Apache” que suele ser la primera opción que aparece en el panel de control de Xampp.
- Ejecutar LabView para empezar a obtener datos del GPS y crear los correspondientes ficheros de texto.
- Ejecutar Google Earth.
- Añadir un enlace de red, opción de la barra de herramientas del mismo programa. El primer enlace será llamado “primero” y el vínculo será http://localhost/PrimerPunt.php. Esta dirección permite a Google Earth acceder al servidor local creado a través de Xampp y de ahí obtener la información del GPS. El fichero PrimerPunt.php remite automáticamente a los .txt creados anteriormente, que son los que contienen la información (Figura 5.8).
- Añadir un segundo Enlace de Red, esta vez llamado “actual” (los nombres son arbitrarios, pero de esta manera resulta más intuitivo) y cuyo vínculo será http://localhost/Actualitzacio.php. En las propiedades de este último enlace hay que añadir una actualización periódica (2 segundos, por ejemplo) para que siga registrando los cambios de posición y los consulte de nuevo en los ficheros que a su vez van cambiando según marca LabView.
- Observar como aparece un símbolo en la pantalla de Google Earth con la posición ocupada en cada instande de tiempo (Figura 5.9).

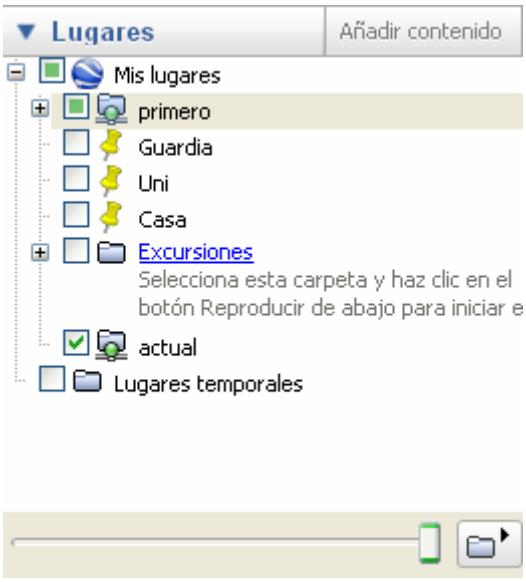

Fig 5.8 Enlaces de red creados

La imagen mostrada en Google Earth es resultado de la combinación de la información de los ficheros .txt con las características específicas de Google que se añaden a través de los archivos .php

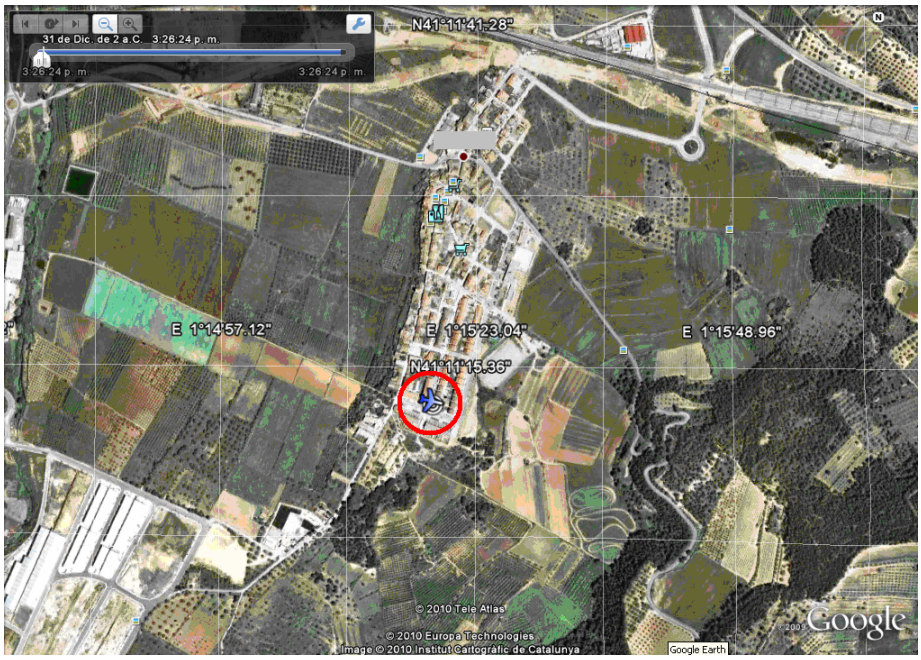

Fig 5.9 Imagen de Google Earth con posición real indicada.

Es importante añadir que para conseguir un correcto funcionamiento, hay que habilitar en la opciones del panel de control que el símbolo decimal sea “.”, así como disponer de una versión de Google Earth más o menos actual, ya que las más antiguas tienen problemas para obtener los enlaces de red.

5.2.4. Corrección de la trayectoria

Con los datos obtenidos, procesados y mostrados por pantalla, el usuario empieza a interactuar con el sistema, en este caso, corrigiendo y modificando la trayectoria del MAV en cuestión. Hasta ahora, sin información sobre el estado del vehículo, cualquier intento de cambiar su actitud habría sido sin ningún criterio empírico, y por tanto habría dependido de la habilidad del propio usuario, pero una vez obtenida la información correspondiente ya se dispone de información objetiva con la que trabajar y los controladores pueden realizar su función basándose en ella.

La información de posición, velocidad y aceleración del MAV permiten realizar las actuaciones pertinentes para cambiar estos valores, y de nuevo poder comprobar en tiempo real si los cambios realizados han surtido algún efecto.

Para llevar a cabo estas modificaciones se utilizará un sistema basado en los mandos Wiimote, los cuales permiten transmitir una excitación a los motores del vehículo en función de las aceleraciones a las que es sometido el propio mando controlado por el usuario.

5.3. Sistemas de apoyo

A pesar de que el objetivo de este proyecto es crear un sistema de control y navegación, a veces es conveniente compaginar los elementos puramente referidos a propia navegación, con otros elementos que pueden ser más representativos o intuitivos para controlar el vehículo en determinadas circunstancias, como son la cámara o el sensor de ultrasonidos.

Del mismo modo, se incluyen aquí otros elementos cuyos valores de entrada son necesarios para poder interactuar entre el vehículo y el controlador o usuario, en este caso el mando sensor de movimiento Wiimote.

En este apartado se incluye la parte de gestión de todos estos componentes extras, y que a pesar de no formar parte puramente de la navegación pueden llegar a tener componentes y requerir gestión con un importante grado de sofisticación.

5.3.1. Cámara

Para obtener la información procedente de la cámara, hace falta disponer del toolkit “Vision” que ofrece National Instruments para todas las aplicaciones relacionadas con la gestión de imágenes. También es necesario contar con IMAQ, también de National Instruments, que nos permite mostrar este tipo de imágenes en tiempo real.

La comunicación que se establece entre la cámara y el ordenador es completamente independiente del puerto serie utilizado para gestionar el resto de información, de modo que el receptor lleva la señal obtenida por otra vía.

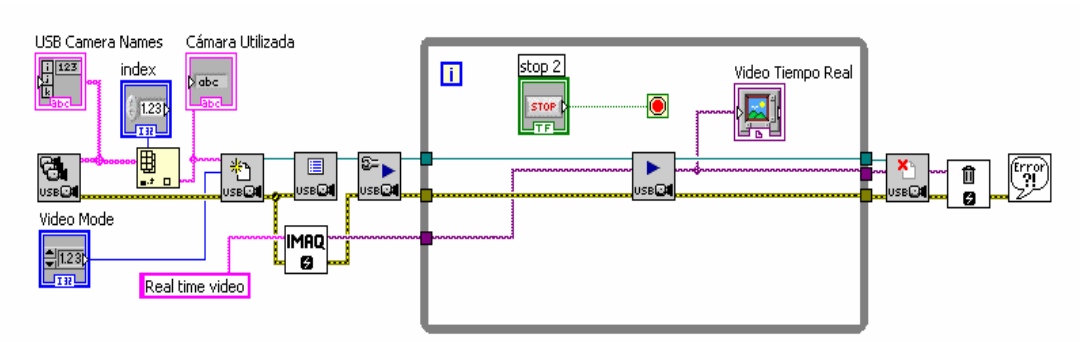

Fig 5.10 Gestión de la cámara en el diagrama de bloques

Una vez obtenidas estas herramientas, para mostrar la imagen por pantalla sólo hay que utilizar los iconos correspondientes que aparecen con el toolkit de "Vision" y la instalación de IMAQ, asignando los valores correspondientes, en este caso el nombre o ubicación de la cámara a través de la cuál llegan las imágenes (Figura 5.10). A pesar de tratarse de una cámara inalámbrica, una vez establecida la comunicación, el receptor actúa como una cámara web estándar, de modo que esto nos permite mostrar imágenes de cualquier cámara conectada a través de USB al ordenador.

Una vez obtenida la señal, sólo hay que añadir un indicador que también viene incorporado a IMAQ para ver las imágenes en el panel frontal (Figura 5.11).

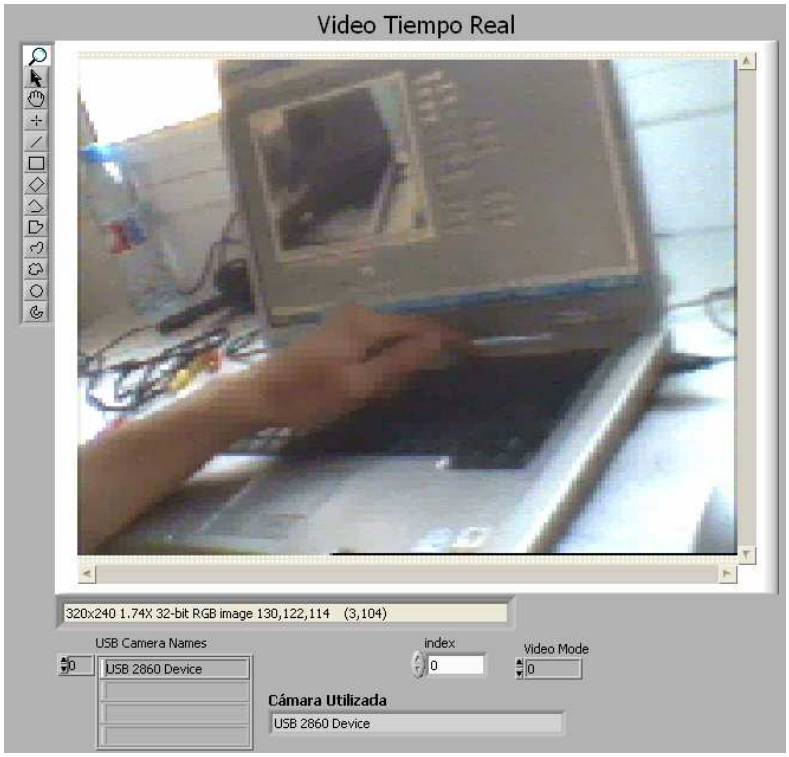

Fig 5.11 Presentación de imagen de la cámara en el panel frontal

5.3.2. Ultrasonidos

La gestión del valor del sensor de ultrasonidos se lleva a cabo en su mayor parte en la misma placa Arduino a bordo del vehículo, de modo que el valor final en la unidades adecuadas [cm] ya llega directamente a LabView.

Sabiendo esto, lo que queda por hacer es descomponer el paquete de información obtenido para dejar el valor de la distancia medida como un único valor que mostrar (Figura 5.12).

Como ya se ha introducido antes, este valor se encuentra en la cuarta posición de la línea que empieza por una letra “A”. La cadena de caracteres inicialmente recibida debe pasar por dos filtros de “String Subset”. El primero de ellos diferencia entre las dos líneas diferentes: La que empieza por “G” y contiene toda la información GPS ya explicada, y la que empieza por “A” y contiene la información combinada de acelerómetros y ultrasonidos. Una vez aislada esta línea, el segundo “String subset” nos permite extraer el cuarto valor de la secuencia que queda, utilizando el método ya conocido de introducir los valores de “offset” y tamaño de “Substring”. Cuando se dispone ya de este valor de forma independiente del resto ya se puede mostrar por pantalla a través de un indicador estándar, pasándolo antes por un conversor de cadena de caracteres a número.

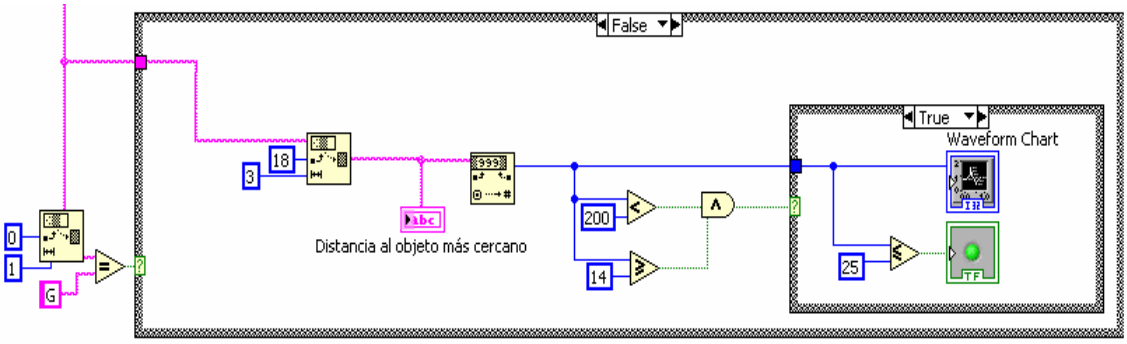

Fig 5.12 Gestión de la señal de ultrasonidos en el diagrama de bloques

Dado que este valor tiene una importancia añadida, teniendo en cuenta que indica la distancia a la que se encuentra nuestro vehículo del obstáculo más cercano, se considera adecuado mostrar el valor de forma gráfica, no sólo con el número de centímetros que mide el sensor, sino además introduciendo el valor en un gráfico “Waveform Chart” que permite observar de forma más clara e intuitiva la separación real que existe (Figura 5.13).

Fig 5.13 Presentación de la medida del sensor de ultrasonidos en el panel frontal

Del mismo modo, se añade un indicador del tipo “Boolean” en forma de LED que se ilumina cuando el valor de la distancia se encuentra por debajo de los 25 centímetros, y por tanto se considera que el riesgo de colisión requiere atención inmediata.

5.3.3. Mando sensor de movimiento

Este componente realiza dos funciones a la vez dentro del sistema: Por un lado actúa como sensor aunque sufre modificaciones provocadas por el propio usuario, estas son transmitidas al ordenador para ser gestionadas como corresponda y por otro lado, es el principal actuador del sistema.

Para poder gestionar la señal obtenida por el mando de Wii es necesario obtener el “toolkit” para Wiimote, que también se puede obtener por internet sin ningún problema, para que LabView sea capaz de interpretar la señal que genera el mando y que es recibida por Bluetooth.

La señal recibida, del mismo modo que la cámara, sigue un recorrido distinto al de los demás sensores obtenidos a través de Arduino. En este caso, la comunicación se lleva a cabo a través de Bluetooth, para la cual hace falta añadir el receptor correspondiente en caso de que no lo incorpore el ordenador.

La información que se recibe incluye tanto los valores de aceleración medidos en los tres ejes como los botones del mando que son pulsados, aunque en nuestro caso nos los utilizaremos. Una vez se dispone de los valores de aceleración transmitidos por el mando, sólo hace falta realizar los factores de conversión correspondientes para obtener la medida en el formato adecuado (Figura 5.14).

Cuando el formato es el correcto, lo que hay que hacer con estos valores es escribirlos en el puerto serie a través de LabView, teniendo en cuenta que no deben solaparse con los valores recibidos de la misma placa.

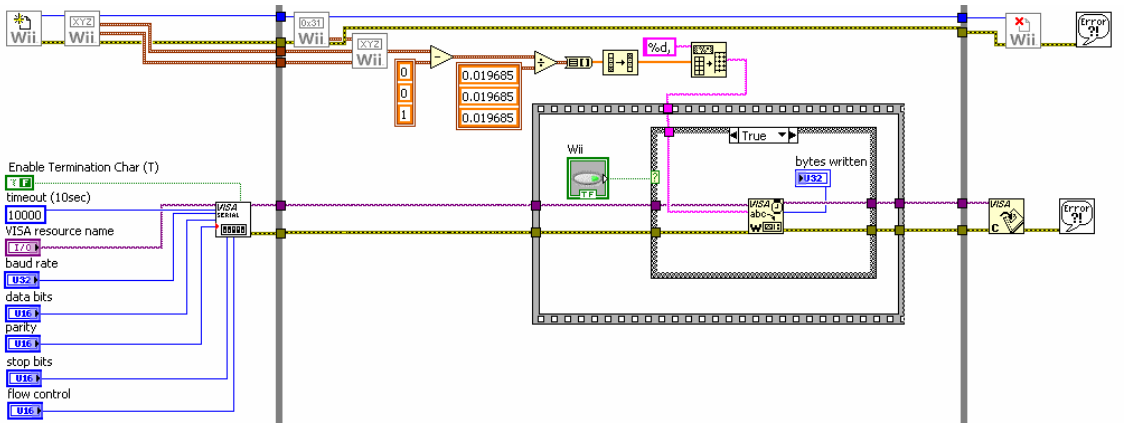

Fig 5.14 Gestión de la señal del Wiimote en el diagrama de bloques

En arduino ya se incorpora el sistema de actuación para el mando, adaptado al tipo de vehículo en que se incorpora el sistema. Se debe tener en cuenta el número y disposición de los motores para una correcta respuesta aerodinámica.

CAPÍTULO 6. SISTEMA DE ACTUACIÓN

6.1. MAVs con superficies de control

Para llevar a cabo el control de vehículos MAV, existen dos sistemas claramente diferenciados. El primero de ellos es el que sigue un funcionamiento más parecido a las aeronaves convencionales, es decir, con superficies de control.

Aunque para nuestro caso, el sistema ideado no utilizará este tipo de actuaciones de control, es importante saber que existen vehículos con estas características.

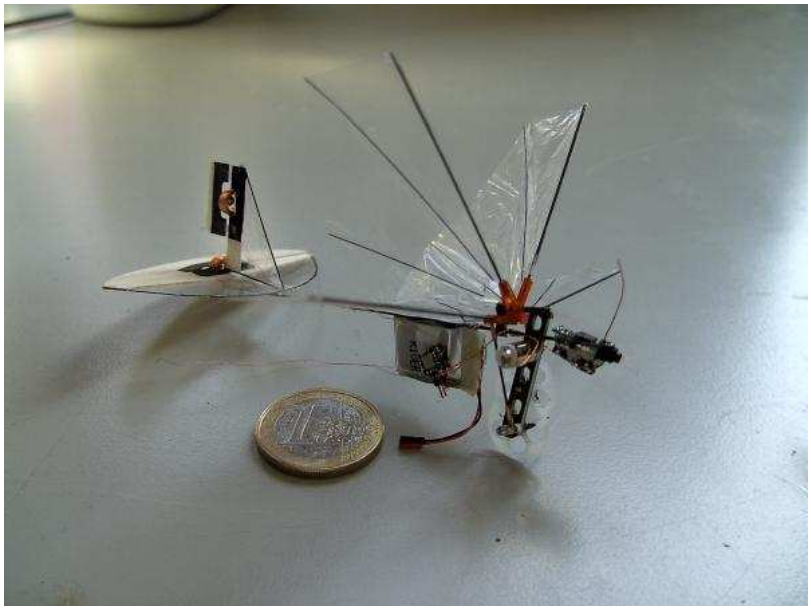

Fig 6.1 MAV con superficies de control

Para poder desarrollar estos sistemas, la capacidad de actuación de las distintas superficies de control ha de ser bastante elevada, ya que se deben utilizar servo-controladores para alerones, timón de dirección, timón de profundidad, etc. En este caso, para obtener la sustentación necesaria para levantar el vuelo es necesario mantener siempre una componente de velocidad relativa importante. De este modo, se pierde una componente muy útil para MAVs, que es el vuelo estacionario.

Su característica más destacada es la similitud de actuación con las aeronaves convencionales, aunque hay que tener en cuenta que su menor tamaño también resulta una limitación respecto a las anteriores en cuanto a que una atmósfera hostil puede resultarle mucho más perjudicial y llevarlo incluso a perder el control.

6.2. MAVs con control único sobre motores

Este es el caso que se considera para nuestro sistema, ya que es mucho más sencillo de diseñar, y nos permite llevar a cabo un control mucho más parecido al que puede ser el de los helicópteros, con ventajas, como la capacidad para llevar a cabo vuelos controlados a velocidades reducidas.

Fig 6.2 MAV con control único sobre motores

La capacidad de realizar vuelos estacionarios resulta de gran importancia en los MAVs, para algunas de sus actividades más típicas como el reconocimiento.

Además resultan muchos más sencillos de diseñar, en cuanto al sistema de actuación ya que este únicamente tiene en cuenta la potencia que se suministra a cada motor y que genera una sustentación de forma prácticamente independiente del resto de elementos.

6.2.1. Valores de entrada

En nuestro caso, los valores de entrada que obtendrá el vehículo serán valores (uno por cada motor que se quiera controlar) separados por comas. En ellos, el valor máximo de potencia que puede suministrar cada motor corresponde con el 255 y el mínimo con 0. La relación de valores corresponde de forma proporcional al valor de aceleración que se capta del mando Wiimote y que son gestionados por LabView y enviados por XBee.

6.2.2. Energía

El elemento que interacciona entre los sistemas de control y actuación es el que transmite la energía para que actúen los motores en función de lo que se haya regulado a partir de la gestión de los datos obtenidos.

Los valores obtenidos a través de Wiimote se traducen en energía suministrada a cada motor en forma de pulso del tipo PWM. Estos pulsos son los que provocan el movimiento de los motores, y en función de cual sea su ciclo de trabajo, la sustentación para cada motor será diferente.

Este tipo de modulación se conoce como modulación por ancho de pulsos. Para ello, se realiza un control del ancho de cada pulso "τ" que se transmite, en función del periodo de la señal transmitida "T". La relación entre estos dos valores es lo que se conoce como ciclo de trabajo "D" y rige la energía total que se transmite al actuador.

$$D = \tau/T \qquad \textbf{(6.1)}$$

Un ciclo de trabajo del 0% (Figura 6.3) implica que no hay ancho de pulso en la señal:

Fig 6.3 Ciclo de trabajo del 0%

En cambio, un ciclo de trabajo superior, como puede ser del 25% (Figura 6.4) hace que el valor de energía útil transmitida ocupa una cuarta parte del total de la señal:

Fig 6.4 Ciclo de trabajo del 25%

A medida que aumentamos el valor del ciclo de trabajo, la relación entre "τ" y "T" aumenta. Este es el caso en que "D" es de un 50%(Figura 6.5):

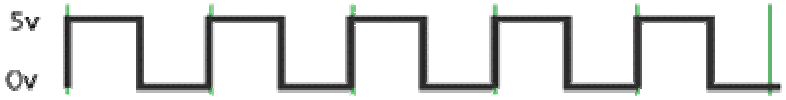

Fig 6.5 Ciclo de trabajo del 50%

El ciclo de trabajo puede seguir aumentando a medida que se requiere suministrar más energía a los motores del vehículo (Figura 6.6):

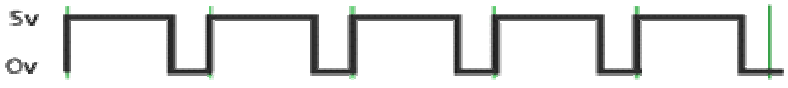

Fig 6.6 Ciclo de trabajo del 75%

Hasta llegar al límite en que se alcanza un valor de ciclo de trabajo del 100% (figura 6.7), en cuyo caso la señal es continua y transmite el valor máximo en todo momento.

Fig 6.7 Ciclo de trabajo del 100%

El resto de casos realiza una transmisión de energía parcial, pero de la que resulta un valor efectivo prácticamente equivalente a una señal continua de menor intensidad, ya que el sistema no tiene tiempo para asimilar la caída de tensión instantánea.

Antes de transmitir la señal PWM a cada motor, ésta ha de pasar por un *driver* de potencia, ya que la señal en sí no tiene energía para alimentar el motor. Un ejemplo de este tipo de *drivers* es el LP239. Este componente consiste en un puente en H controlado por una señal del tipo OWM, que permite dar potencia a los motores.

Para que cada motor disponga de una fuente de energía, el vehículo incorporará una batería que suministrará los valores de tensión correspondientes para llevar a cabo el control de cada motor en función de los requisitos que en cada momento considere oportunos el controlador o usuario.

Esta energía se traducirá en valores de velocidad angular correspondiente al giro de los motores. De este modo, si se quiere mantener una condición de equilibrio la velocidad de giro de las aspas de cada motor será la misma (caso de mando controlador quieto). En caso de querer provocar un movimiento controlado, las perturbaciones que induzca el usuario se traducirán en cambios de la velocidad angular de cada motor.

6.2.3. Sustentación suministrada por cada motor

Para mantener el vuelo del vehículo, cada motor gira a una velocidad angular determinada que depende de la energía que consuma. La dependencia entre la energía entregada a casa motor y la velocidad a la que gira depende únicamente del valor del ciclo de trabajo de la PWM en cada caso.

Esta velocidad angular “ω” se relaciona con la velocidad lineal “v” siguiendo:

$$v = \omega \int r \qquad \textbf{(6.2)}$$

Siendo “r” la distancia al eje de rotación definida entre 0 y la longitud de cada aspa “R”. Por tanto, después de realizar la integral definida obtenemos:

$$v = \omega \cdot R^2/2 \quad \textbf{(6.3)}$$

Para conocer el valor de la sustentación que proporciona cada motor hay que tener en cuenta una ecuación básica de la aerodinámica que es:

$$L = ½ \cdot \rho \cdot v^2 \cdot S \cdot Cl \quad \textbf{(6.4)}$$

Siendo “L” la sustentación, “ρ” la densidad del aire, “v” la velocidad lineal que proporciona el motor, “S” la superficie alar y “Cl” el coeficiente de sustentación, que si suponemos un ángulo de ataque constante, no varia.

Combinando las dos ecuaciones anteriores y teniendo en cuenta que consideramos que cada motor dispone de dos aspas que suministran sustentación cada una por su cuenta obtenemos:

$$L = \rho \cdot (\omega \cdot R^2/2)^2 \cdot S \cdot Cl \quad \textbf{(6.5)}$$

Todos los valores son constantes excepto “ω” que hace variar la sustentación de forma cuadrática:

$$L = k\ \omega^2 \quad \textbf{(6.6)}$$

Siendo “k” una constante que depende únicamente de las características de las aspas que se utilicen y de la densidad del aire, que a bajas alturas también se considera constante.

Finalmente, con este capítulo se cierra el lazo del sistema de control y navegación, habiendo completado todos los pasos, empezando por la obtención de información relativa al vuelo, comunicándola, procesándola y actuando para corregir los errores registrados.

CAPÍTULO 7. CONCLUSIONES

Se ha cumplido con el objetivo de diseñar e implementar una plataforma de navegación y control de vehículos aéreos. En concreto:

1. Se ha presentado una arquitectura de bajo coste, programable con software libre que puede utilizarse para diferentes vehículos aéreos. Esta se basa en el uso de la plataforma Arduino, centrada alrededor de un microcontrolador de Atmel.

2. Se han diseñado los sistemas sensores que permiten la localización del sensor (GPS) así como el estudio de la actitud en vuelo del mismo (Acelerómetros), el cálculo de distancia a obstáculos (radar por ultrasonidos) y de la navegación por visión (Cámara de vídeo).

3. A nivel de comunicación se ha realizado mediante el protocolo XBee que trabajando a 2,4 GHz permite la transmisión de los datos de los sensores en tiempo real hasta distancias de 100 metros. La comunicación de los datos de la cámara de vídeo se ha realizado con un protocolo propio que trabaja en paralelo en la banda de 1 GHz.

4. Se han implementado todos los programas del microcontrolador de Atmel que permiten la adquisición de los datos de los sensores y el control de los motores. Estos datos son enviados con el sistema XBee a un ordenador donde un programa realizado en LabView permite su visualización y procesado.

5. A partir de los datos de GPS se ha realizado la aplicación que permite su transferencia en tiempo real a la aplicación libre de Google Earth para saber en todo momento su posición.

6. El control del vehículo se puede realizar mediante un mando de la plataforma de juegos Wii de Nintendo ya que se ha programado en Labiew el control del mando en tiempo real a partir de un puerto Bluetooth.

7. Aunque la plataforma está perfectamente preparada para poder ser utilizada en cualquier vehículo será necesario estudiar en trabajos futuros la compatibilidad del protocolo XBee de transmisión de datos con su conexión, también serie, a un ordenador portátil.

REFERENCIAS

[1] Barrientos, A., *Vehículos aereos no tripulados: Tecnología y aplicaciones*, Universidad Politécnica de Madrid.

[2] Castillo, P., *Modelling and Control of Mini-Flying Machines*, Ed. Springer, Estados Unidos, 2005.

[3] Kimon P. Valavanis, *Advances in Unmanned Aerial Vehicles*, Ed. Springer, 2007.

[4] http://www.arduino.cc/es/ última visita 29-05-2010

[5] http://www.ni.com/labview/ última visita 23-05-2010

[6] http://www.portalplanetasedna.com.ar/gps.htm última visita 17-04-2010

[7] http://www.maxbotix.com/MB1000__LV-MaxSonar-EZ0.html última visita 14-04-2010

[8] http://www.gpsinformation.org/dale/nmea.htm última visita 17-04-2010

[9] http://www.wiiprojects.org/bluetooth.html última visita 16-04-2010

[10]http://www.freescale.com/files/sensors/doc/fact_sheet/MMA7260QFS.pdf última visita 13-03-2010

[11] http://www.vectorsite.net/twuav.html última visita 30 de Mayo de 2010

ANEXOS

SISTEMA DE CONTROL Y NAVEGACIÓN PARA MAV

TITULACIÓN: Ingeniería Técnica Aeronáutica, especialidad Aeronavegación

AUTOR: Héctor Valle Ruiz

DIRECTOR: Jaime Óscar Casas Piedrafita

FECHA: 10 de junio de 2010

ANEXO 1. DATASHEET MMA6270Q

1. The MMA6270Q (X and Y axes) and MMA6280Q (X and Z axes) MEMS accelerometers (bottom) and the MMA7261Q (X, Y, and Z axes) sensor (top) represent a new class of low-g, low-cost, selectable-level devices.

ANEXO 2. DATASHEET LV-MaxSonar-EZ0

ANEXO 3. PROGRAMACIÓN LABVIEW

ANEXO 4. PROGRAMACIÓN ARDUINO

```
// libreria para agregar puertos serie
#include <SoftwareSerial.h>

// declaracion pins del nuevo puerto serie
#define rxPin 11
#define txPin 10
#define INLENGTH 16
#define INTERMINATOR 13
#define MAX_STRING_LEN 16

// declaracion del nuevo puerto serie
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);

// variables GPS
byte byteGPS = 0;
int i = 0;
int indices[13];
int cont = 0;
int conta = 0;
char inBuffer[300] = "";
int k = 0;

//entradas aceleraciones
int pinPortX1 = 2;
int pinPortY1 = 3;
int pinPortX2 = 4;
int pinPortY2 = 5;

//variables Ultrasonido
unsigned long micro1,micro2;
unsigned long ultr;
int pinPortUS = 9;

//variables control motores
int inCount;
int engine1Port = 3;
int engine1Value = 0;
int engine2Port = 5;
int engine2Value = 0;
int engine3Port = 6;
int engine3Value = 0;
char* control= "                ";
int testChar;

//procedimiento para almacenar la lectura del puerto serie
void readSerialData()
{
  for (inCount = 0; inCount<=INLENGTH; inCount++)
  {
```

```
    delay(1);
    if (Serial.available() == 0)
    {
      control[inCount] = 0;
      break;
    }
    delay(1);
    testChar = Serial.read();


    control[inCount] = testChar;
    if (control [inCount] == INTERMINATOR) break;
  }
}

//procediemiento para obtener un SubString
char* subStr (char* str, char *delim, int index) {
   char *act, *sub, *ptr;
   static char copy[MAX_STRING_LEN];
   int i;

   strcpy(copy, str);

   for (i = 1, act = copy; i <= index; i++, act = NULL) {
     //Serial.print(".");
     sub = strtok_r(act, delim, &ptr);
     if (sub == NULL) break;
   }
   return sub;

}

void setup(){

  //setup de los puertos serie
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  mySerial.begin(4800);
  Serial.begin(19200);

  //setup GPS
  Serial.println("Configurando GPS...");
  delay(1000);
  mySerial.println("$PSTMNMEACONFIG,0,4800,1,1");        //
configure NMEA sentences to show only GGA sentence
  delay(100);

mySerial.println("$PSTMINITGPS,4140.000,N,00053.000,W,0197,
22,10,2007,11,40,00");
  }
```

```
void loop(){

  //variables aceleraciones
  int valueX1=0;
  int valueY1=0;
  int valueX2=0;
  int valueY2=0;

  //variable Ultrasonido
  int value=0;

  //lectura GPS
  byteGPS = 0;
  i = 0;
  while(byteGPS != 42){                    // read the GGA
sentence
    byteGPS = mySerial.read();
    inBuffer[i]=byteGPS;
    i++;
  }

  //gestion aceleracion
  valueX1=analogRead(pinPortX1);
  valueY1=analogRead(pinPortY1);
  valueX2=analogRead(pinPortX2);
  valueY2=analogRead(pinPortY2);


  //lectura Ultrasonido
  value=digitalRead(pinPortUS);

  while (value==HIGH) //espera a que vaya a LOW
  {
    value=digitalRead(pinPortUS);
  }
  while(value==LOW)  // captura el ultimo momento conocido
en LOW
  {
    value=digitalRead(pinPortUS);
    micro1 = micros();
  }
  while(value==HIGH)      //  captura  el  ultimo  momento
conocido en HIGH
  {
    value=digitalRead(pinPortUS);
    micro2 = micros();
  }
  ultr = ((micro2 - micro1)/147.0)*2.54;


  //escritura en el puerto serie
```

```
  k = 1;
  while(inBuffer[k] != 42){
    Serial.print(inBuffer[k]);
     k++;
  }
  Serial.println();
  delay(10);
  Serial.print("A,");
  Serial.print(valueX1);
  Serial.print(",");
  Serial.print(valueY1);
  Serial.print(",");
  Serial.print(valueX2);
  Serial.print(",");
  Serial.print(valueY2);
  Serial.print(",");
  Serial.println(ultr);
  delay(50);

  //Se precede a leer la información de control de motores
procedente de LabView

  while(!Serial.available())
   {
   }
  readSerialData();
  engine1Value = atoi(subStr(control,",",1));
  engine2Value = atoi(subStr(control,",",2));
  engine3Value = atoi(subStr(control,",",3));

  analogWrite(engine1Port,engine1Value);
  analogWrite(engine2Port,engine2Value);
  analogWrite(engine3Port,engine3Value);

}
```

ANEXO 5. ARCHIVOS .php

PrimerPunt.php

```
<?php

// read initial position
$handle = fopen("tmp\position.txt", "r");
$htime = fopen("tmp\ltime.txt","r");

if ($handle) {

     // get current date/time
          $timestamp = fgets($htime);
     $position = fgets($handle);

     $kml =
        '<?xml version="1.0" encoding="UTF-8"?>' .
        '<kml xmlns="http://www.opengis.net/kml/2.2">' .
        '<Document id="primerpunt">' .
        '  <Style id="originSty">' .
        '    <IconStyle>' .
        '      <Icon>' .
        '
<href>http://maps.google.com/mapfiles/kml/shapes/target.png
</href>' .
        '      </Icon>' .
        '    </IconStyle>' .
        '  </Style>' .
        '  <Placemark id="plane">' .
        '    <Point>' .
        '                <coordinates>'   .   $position   .
'</coordinates>' .
        '    </Point>' .
        '<TimeStamp>' .
        '   <when>' . $timestamp . '</when>' .
        '    </TimeStamp>' .
        '  </Placemark>' .
        '  <Placemark>' .
        '    <styleUrl>#originSty</styleUrl>' .
        '    <Point>' .
        '                <coordinates>'   .   $position   .
'</coordinates>' .
        '    </Point>' .
        '<TimeStamp>' .
        '   <when>' . $timestamp . '</when>' .
        '    </TimeStamp>' .

        '  </Placemark>' .
        '</Document>' .
```

```
        '</kml>';
     fclose($handle);
}

// write last read waypoint to previous file
$handle = fopen("tmp\previous.txt", "w");
fwrite($handle, $position);
fclose($handle);
fclose($htime);

echo $kml;

?>
```

Actualitzacio.php

```
<?php

$hCurrent  = fopen("tmp\position.txt", "r");
$hPrevious = fopen("tmp\previous.txt", "rw");
$hHeading = fopen("tmp\heading.txt", "rw");
$hTime = fopen("tmp\ltime.txt", "rw");

$current  = fgets($hCurrent);
$previous = fgets($hPrevious);
$heading  = fgets($hHeading);
$timestamp = fgets($hTime);

$kml =       '<?xml version="1.0" encoding="UTF-8"?>' .
        '<kml xmlns="http://www.opengis.net/kml/2.2">' .
        '<NetworkLinkControl>' .
        '  <Update>' .
        '
<targetHref>http://localhost/PrimerPunt.php</targetHref>' .
        '    <Create>' .
        '   <Document targetId="primerpunt">' .
        '        <Style id="pathSty">' .
        '          <LineStyle>' .
        '            <color>ff00aa00</color>' .
        '            <width>3</width>' .
        '          </LineStyle>' .
        '          <IconStyle>' .
        '            <heading>'.$heading.'</heading>' .
        '
<Icon>http://maps.google.com/mapfiles/kml/shapes/airports.p
ng</Icon>' .
        '          </IconStyle>' .
        '        </Style>' .
        '      <Placemark>' .
        '        <styleUrl>#pathSty</styleUrl>' .
        '        <LineString>' .
           '
<altitudeMode>absolute</altitudeMode>' .
        '
<coordinates>'.$previous.','.$current.'</coordinates>' .
        '        </LineString>' .
          '         <TimeStamp>' .
        '       <when>' . $timestamp . '</when>' .
        '        </TimeStamp>' .

        '    </Placemark>' .
        '      </Document>' .
           '    </Create>' .
        '    <Change>' .
        '      <Placemark targetId="plane">' .
```

```
        '        <styleUrl>#pathSty</styleUrl>' .
        '        <Point>' .
          '
<altitudeMode>absolute</altitudeMode>' .
        '          <coordinates>'.$current.'</coordinates>'
.
        '        </Point>' .
                    '<TimeStamp>' .
        '       <when>' . $timestamp . '</when>' .
        '        </TimeStamp>' .
        '      </Placemark>' .
        '    </Change>' .
           '  </Update>' .
        '</NetworkLinkControl>' .
        '</kml>';

fclose($hCurrent);
fclose($hPrevious);
fclose($hTime);
fclose($hHeading);

// write current to previous file
copy("tmp\position.txt", "tmp\previous.txt");

echo $kml
?>
```