Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Estructura de *Hardware* para autopiloto de *UAV* con posibilidades de navegación inercial basada en modelo.

Autor: Thuy Ninh Khac

Tutor: Msc. Alleiny Machado Sosa

Santa Clara

2012

"Año 54 de la Revolución"

Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Estructura de *Hardware* para autopiloto de *UAV* con posibilidades de navegación inercial basada en modelo.

Autor: Thuy Ninh Khac
email: tninhkhac@uclv.edu.cu

Tutor: Msc. Alleiny Machado Sosa Prof. Asistente
Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV
email: alleini@uclv.edu.cu

Santa Clara

2012

"Año 54 de la Revolución"

Hago constar que el presente TRABAJO DE DIPLOMA fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

Thuy Ninh Khac — Fecha
Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Thuy Ninh Khac — Fecha
Autor

Boris Luis Martínez Jiménez, Dr.C — Fecha
Jefe del Departamento

Responsable ICT o J' de Carrera, (Dr.C., M.Sc. o Ing.) — Fecha
Responsable de Información Científico-Técnica

PENSAMIENTO

Debemos ser libres, no para hacer lo que nos plazca, sino libres para comprender muy profundamente nuestros propios instintos e impulsos. La libertad no es para hacer lo que nos antoja, sino que consiste más bien en estar libres de todo el tormento de la vida, de nuestros problemas, ansiedades, miedos, heridas psicológicas y de todo el conflicto que hemos tolerado en nosotros mismos y en el mundo.

Jiddu Krishnamurti (1895-1986)

DEDICATORIA

A mi padre, **Quang N.**,

su ausencia me hace más fuerte.

A mi tío, **Vinh N.**,

de el he aprendido como enfrentar los obstáculos en la vida.

A mi amiga de la niñez, **Sinh N.**,

nunca olvidaré lo feliz que fuimos,

y para que nuestros objetivos y principios nunca se pierdan por ningún motivo.

A mi **familia**,

siempre esperan el día de mi regreso con la frente en alto.

AGRADECIMIENTOS

*A mi **familia** por apoyarme siempre en los momentos más difíciles de mi vida, por su amor eterno, por ser la razón de mi existencia.*

A mi tutor Alleiny por su ayuda en la confección de este Trabajo de Diploma.

A mis amigos de Cuba por cuanto aprendimos juntos en los años de la carrera.

A los profesores, por brindarnos sus conocimientos.

A los vietnamitas en Cuba, por todas las cosas tanto felices como tristes que aquí compartimos.

Santa Clara, Cuba, 2012

RESUMEN

Una de las principales áreas de investigación a la cual se ha prestado un gran interés por varios años, ha sido al estudio de los vehículos aéreos autónomos (*UAV*). Muchos centros de investigación y universidades del mundo se han dedicado al desarrollo de varias ramas de aplicación de los *UAVs*. El Grupo de Automatización Robótica y Percepción *(GARP)* están interesados en las disímiles aplicaciones que ofrecen los aviones autónomos para desarrollar e implementar autopilotos para estos vehículos, los cuales deben ser de bajo costo, elevada capacidad de cálculo y poco peso para que puedan ir a bordo de las pequeñas aeronaves. Por este motivo en este trabajo se propone una estructura de *hardware* de elevada capacidad de cálculo para implementar autopilotos con posibilidades de navegación inercial basada en el modelo dinámico de la aeronave. Para esto se empleará la computadora en una sola tarjeta MBTI2440, de fabricación nacional, basada en arquitectura ARM9 y corriendo el Sistema Operativo (SO) LINUX, unida a un microcontrolador *PIC*.

TABLA DE CONTENIDO

Página

ÍNDICE DE TABLAS

Tabla Página

ÍNDICE DE FIGURAS

Figura Página

LISTA DE ABREVIATURAS

UAV	Unmanned Aerial Vehicle(Vehículo Aéreo no tripulado).
PIC	Programmable Interface Controller (Controlador de Interfaz Programable).
dsPIC	Developed System Programmable Interface Controller (Sistema desarrollado de Controlador de Interfaz Programable).
GPS	Global Positioning System (Sistema de posicionamiento global).
IMU	Inertial Moment Unit (Unidad de Momento Inercial).
DISAM	División de Ingeniería de Sistemas y Automática.
GARP	Grupo de Automatización Robótica y Percepción.
SPI	Serie Peripheral Interface (Interfaz de Periféricos Serie).
PCB	Printed Circuit Board (Placa de Circuito Impreso).
SO	Sistema Operativo.
GPIO	General Purpose Input Output (Entrada Salida de Propósito General).
UCLV	Universidad Central de las Villas.
DOF	Degrees of Freedom (Grado de Libertad).
SPYOP	Psychological Operation (Operación psicológica).
EEUU	Estados Unidos.
MTi-G	Motion tracker integrated *GPS* (Rastreador de Movimiento integrado *GPS*).
FSK	Frequency Shift Keying (Modulación de frecuencia).
PWM	Pulse Wide Modulation (Modulación de ancho de Pulso).
PCM	Pulse Coded Modulation (Modulación de Pulso codificado).
GND	Ground (Tierra).
MOSI	Master Out, Slave In (Entrada Esclavo, Salida Maestro).
MISO	Master In, Slave Out (Entrada Maestro, Salida Esclavo).
CPOL	Clock Polarity (Polaridad de Reloj).
CPHA	Clock Phase (Fase de Reloj).
SS	Slave Selector (Selector de Esclavo).
UART	Universal asynchronous Receiver/Transmitter (Receptor/Transmisor Universal Asincrónico).
I2C	Inter Integrated Circuit (Entre Circuitos Integrados).
CAN	Controller Area Network (Controlador de Area de Red).
DCI	Data Converter Interface (Interfaz de Conversor de datos).
ICID	Instituto Cubano de Investigaciones Digitales.
INS	Instrucciones.
TFT	Thin-Film Transistor (Transistor de película fina).
RF	Radio Frequency (Radiofrecuencia).
CCD	Charge Coupled Device (Dispositivo de carga acoplada).
LCD	Liquid Crystal Display (Pantalla de cristal líquido).

LISTA DE SIMBOLOS

mW	miliWatt
mA	miliAmpere
ms	milisegundo
Ksps	Kilo Samples Per Second (Kilo muestras por segundo).
VDC	Voltage Direct Current (Voltaje de Corriente Directa).
TTL	Transistor-Transistor Logic (Lógica Transistor-Transistor).
MIPs	millones de instrucciones por segundos.

INTRODUCCIÓN

Uno de los sueños más antiguos del hombre es el de encumbrarse en el viento y despegarse de las ataduras de la gravedad. Muchos hombres de la Edad Media saltaban desde torres intentando alcanzar alturas mayores batiendo alas artificiales, sin éxito alguno. En el año 1680, el italiano Giovanni Borelli luego del estudio de la aplicación de los músculos en el vuelo de los pájaros, concluyó que el humano nunca sería capaz de volar a partir de su propia fuerza. Hasta ese momento, el único trabajo serio en materia de aeronáutica era el de Leonardo da Vinci, quien a partir del estudio del vuelo de los pájaros, concibió máquinas voladoras con alas batientes, con cuyo movimiento se generarían las fuerzas necesarias para volar y desplazarse (sus diseños no contemplaban motores mecánicos para la propulsión) (Abusleme˜H., 2000).

A principios del siglo XIX, el inglés Sir George Cayley, baronet de Yorkshire, estableció el concepto moderno de avión. Reemplazó la idea del ala batiente por un ala fija, capaz de elevar al avión y generar la resistencia al avance, y un mecanismo aparte para generar movimiento y oponerse a dicha resistencia. Imaginó sistemas de propulsión basados en el movimiento generado por la combustión súbita de materiales inflamables, aunque en su época sólo existía la pesada máquina de vapor, poco eficiente para elevar un avión. En 1804 construyó un planeador no tripulado calificado por muchos como el primer aeroplano que voló con éxito. Basado en sus estudios de la aerodinámica, construyó un planeador que en 1853 realizó el primer vuelo tripulado de la historia. Por sus contribuciones al concepto de avión, muchos historiadores lo consideran como el inventor del aeroplano (Abusleme˜H., 2000). Desde el punto de vista histórico, estos ejemplos constituyen los antecedentes de los vehículos aéreos no tripulados (*UAVs*, *Unmanned Aerial Vehicles*).

La historia de los *UAVs* se remonta a mediados del siglo XIX: cuando un primitivo *UAV* formado por un globo cargado de bombas se utilizó el 22 de agosto de 1849 en

un ataque austriaco a la cuidad de Venecia. Posteriormente llegaron los misiles crucero, controlados por un sistema de giróscopos durante la Primera Guerra Mundial y aviones radiocontrolados utilizados para entrenar a los tiradores antiaéreos británicos durante la Segunda Guerra Mundial (Asensio, 2008). En las guerras de Corea y Vietnam, el ejército de los Estados Unidos encontró en los *UAVs* una forma de desviar los ataques enemigos de sus bombarderos y cazas tripulados y se desarrollaron también los primeros *UAVs* de reconocimiento.

El futuro de la aviación abarca tres ramas principales: los satélites, los aviones y los *UAVs*. Estos últimos podrían describir un amplio rango de dispositivos capaces de operar en el espacio aéreo que van desde cometas, globos o dirigibles hasta aviones radio controlados, pasando por los misiles o aeronaves prácticamente autónomas operativamente hablando (Asensio, 2008).

Los (*UAVs*) de pequeño porte destacan entre los aviones por su estabilidad, menor complejidad, considerable ahorro en comparación con los vehículos convencionales en cuanto a costo de la aeronave, el no poner al piloto en peligro y el ahorro de combustible. Poseen una variada gama de aplicaciones que van desde la inspección de infraestructuras, líneas eléctricas, oleoductos y gaseoductos hasta el levantamiento topográfico para la elaboración de mapas y toma de fotografías aéreas, además de ser utilizados en supervisión de tráfico, vigilancia de fronteras y búsqueda de personas desaparecidas en fenómenos naturales. Estas plataformas han sido utilizadas para investigar en áreas que van desde el control no lineal, control multivariable, navegación, planificación de trayectorias hasta la detección y seguimiento visual de objetivos (Mejias, 2006).

Otra de las ventajas es la seguridad, debido a que los pilotos controlan el avión desde tierra, en algunos casos todo el vuelo y en otros sólo el despegue y aterrizaje. Como el trayecto se realiza preprogramado con un sistema de navegación asistido *GPS* (*Global Positioning System*) y la *IMU* (*Inertial Moment Unit*); en algunos casos existe el uso de cámaras de visión (Mejias, 2006) para el reconocimiento del terreno. En el diseño de hardware e implementación de control de *UAVs* existe un gran número de

artículos relacionados con el tema (Barrientos, 2007; Cordoba, 2007; AlSwailem, 2004; Andersen, 2004; Vélez, 2003). La planificación de trayectoria (*Path-planning*) es la parte del autopiloto donde se unen navegación y la misión deseada, este algoritmo determina hacia donde va el *UAV* y en que momento debe cambiar su rumbo para cumplir con la trayectoria planificada.

Por su amplia gama de aplicación los vehículos aéreos autónomos han sido un área de investigación a la que se ha prestado gran interés por muchos años. Actualmente, prestigiosas universidades y centros de investigación, tanto privados como públicos, se encuentran investigando y desarrollando vehículos aéreos autónomos. En Cuba el desarrollo de los vehículos autónomos aún es escaso, no obstante instituciones como *GEOCUBA*, facultada para la elaboración de mapas de relieve; *CEMPALAB*, encargada de investigaciones en agricultura de precisión y *CEDAI* nacional, han mostrado interés en las ventajas que los pequeños aviones no tripulados pueden brindar.

En los últimos años el Grupo de Automatización Robótica y Percepción (*GARP*) ha acumulado un número importante de investigaciones dirigidas al diseño e implementación de vehículos autónomos. Las investigaciones ocupan diferentes aéreas que incluyen: modelado de vehículos subacuáticos (Cañizares, 2010), aéreos autónomos (Pineda, 2008; Martínez-Carmenate, 2009) y minihelicóptero (De˜Ávila, 2008). Además se han realizado trabajos en identificación experimental de vehículos aéreos (Martínez-Jiménez, 2012), y de desarrollo de hardware para vehículos autónomos (Martínez, 2005; Guerra, 2010).

Situación del Problema:

A partir de una solicitud del Centro de Investigaciones Mecatronics, *GARP* en el marco de la Tarea Triunfo tiene como objetivo el desarrollo de un autopiloto para un avión de pequeño porte. En este sentido se puede decir que actualmente existe un diseño basado en microcontroladores PIC que cumple con la tarea del control de vuelo y es capaz de seguir trayectorias a través de un conjunto de puntos ubicados en el mapa realizando navegación por *GPS*. La navegación por GPS es algo ya establecido y para esto es necesario contar

con el servicio, el cual depende de terceros. Por otro lado, la navegación inercial basada en el modelo dinámico de la aeronave podría prescindir en un por ciento elevado del uso del *GPS* y utilizarse sólo en términos de corrección siempre que se disponga del servicio. En este punto es donde entra la incapacidad del hardware actual del avión para realizar la navegación inercial basada en el modelo, puesto que la misma implica un elevado costo computacional imposible de realizar por el microcontrolador *PIC* en uso actualmente.

Objetivos:

Objetivo general:

Implementar un autopiloto para *UAV* con posibilidades de navegación inercial basada en el modelo dinámico de la aeronave empleando para ello una computadora en una sola tarjeta MBTI2440, de fabricación nacional, basada en arquitectura ARM9 y corriendo SO LINUX, unida a un microcontrolador *PIC*.

Objetivos específicos:

- Diseñar una tarjeta basada en *PIC* para control de vuelo que incluya los conectores que permite el acople en modo sándwich con la computadora en una sola tarjeta MBTI2440 encargada de la navegación inercial y el seguimiento de trayectorias.
- Establecer la comunicación *SPI* (Serie Peripheral Interface) entre la tarjeta MBTI2440 y la tarjeta basada en *PIC*.
- Recompilar el Kernel de Linux instalado en la tarjeta MBTI2440 para activar funcionalidades no presentes, por ejemplo *driver SPI*, *driver GPIO* (*General Purpose Input Output*), hacer SO preemptivo, etc.
- Programar en lenguaje C las interfaces y demás algoritmos básicos para el sistema de desarrollo.

Organización del informe:

El trabajo, posterior a esta introducción cuenta con la siguiente estructura: tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. El contenido de los capítulos se resume a continuación.

Capítulo 1

En el primer capítulo se realiza un análisis de la bibliografía consultada. Se abordan la definición, la clasificación así como las aplicaciones de los *UAVs* en varios sectores de la vida. Se presentan los diseños que se han desarrollado en la *UCLV* así como la planificación de un nuevo modelo y los requerimientos necesarios para implementar el esquema del diseño y sus principales componentes.

Capítulo 2

En el capítulo dos se hace una breve descripción de los componentes electrónicos que se van a utilizar en la construcción de dicho *hardware*, las interfaces para establecer la comunicación entre ellos así como al *software* para optimizar el sistema empotrado.

Capítulo 3

En el capítulo tres se muestra el resultado del diseño propuesto así como el circuito impreso. Se muestran además los pasos para la compilación del *Kernel* del sistema operativo de la tarjeta MBTI2440 y los algoritmos básicos para el buen desempeño del sistema. Por último, pero no menos importante se realiza un análisis económico que demuestra la viabilidad del diseño realizado.

CAPÍTULO 1
REVISIÓN BIBLIOGRÁFICA

1.1. Introducción.

En este capítulo se realiza una revisión bibliográfica relacionada con los conceptos generales, el desarrollo de los *UAV* en el mundo, algunos ejemplos, así como sus aplicaciones y bondades a la sociedad. También se establece la arquitectura del diseño que se va a utilizar y los criterios tenidos en cuenta para la selección de los componentes de hardware para implementar dicho diseño. Se presentan las características y ventajas que brindan los sistemas empotrados para optimizar el diseño cooperando con los algoritmos de control que desean aplicar. Finalmente se representa el esquema general de la tarjeta expandida basada en MBTI2440 y dsPIC30F4013.

1.1.1. Definición.

En los últimos años la migración de los *UAVs* desde el sector militar al civil ha propiciado que diferentes grupos de investigadores, procedentes del área de la robótica, hayan dedicado grandes esfuerzos a la investigación y desarrollo de estos. En ocasiones el *UAV* se considera un Robot aéreo, entendido como tal, a un sistema físico capaz de desplegarse de manera autónoma o semiautónoma por el aire para realizar diferentes misiones. En todos los casos la aeronave no transporta persona, con ningún propósito y puede ser totalmente controlado desde la estación de tierra (Barrientos, 2007). En general se puede definir:

Un vehículo aéreo no tripulado, *UAV* por sus siglas en inglés (*Unmanned Aerial Vehicle*), o sistema aéreo no tripulado, es una aeronave que vuela sin tripulación humana a bordo. Son usados mayoritariamente en aplicaciones militares. Para distinguir los *UAVs* de los misiles, un *UAV* se define como un vehículo sin tripulación reutilizable, capaz de

mantener un nivel de vuelo controlado y sostenido, y propulsado por un motor de explosión o de reacción.

1.1.2. Clasificación.

A la hora de realizar la clasificación de los *UAV* es posible atender a diferentes criterios. El más simple puede ser el que se basa en el tipo de aeronave del *UAV*. Con este pueden distinguirse a aquellos de despegue vertical de los que no lo son, dentro de los primeros los de tipo de alas rotativa o hélice (helicóptero y quad-rotors entre otros), los de ala flexible (parapentes, ala delta) y los auto-sustentados (dirigibles y globos). Dentro de los de despegue no vertical se encuentran los de alas fijas (aeroplanos) como se muestran en la Figura 1–1.

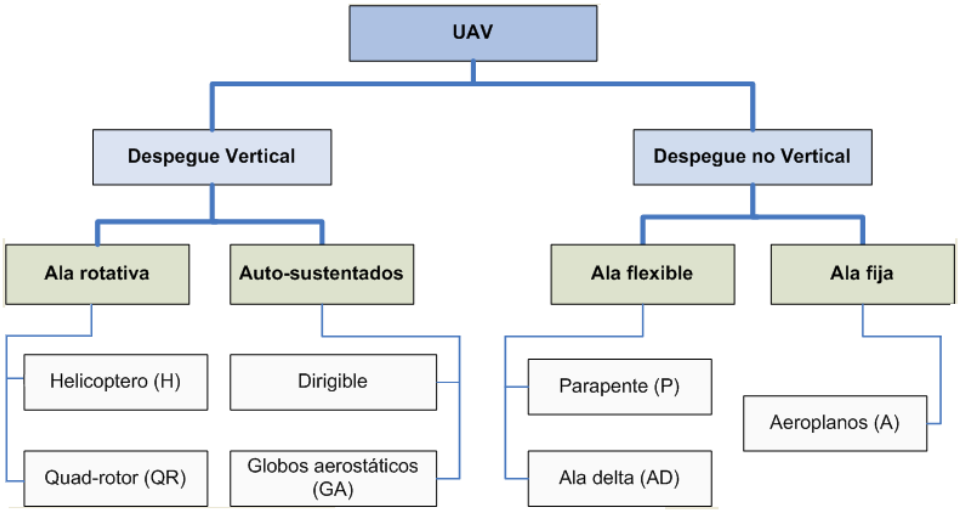


Figura 1–1: Tipos de *UAV*.

Otros criterios de clasificación pueden hacer referencia a las capacidades de vuelo (alcance, altitud, autonomía) o también a la capacidad de carga útil considerando como la capacidad de carga en el despegue. La Figura 1–2 muestra la clasificación de los *UAVs* basados en esos criterios. Esta abarca todas las aplicaciones de los *UAVs* tanto civiles como militares. La mayor parte de los *UAVs* actuales se encuentran en categoría Mini o MR (*Medium Range*), siendo, el vehículo más frecuente utilizado con diferencia al aeroplano (Barrientos, 2007).

Categoría	Acrónimo	Alcance (km)	Altitud de vuelo (m)	Autonomía (horas)	Carga máxima en despegue (kg)	Tipo de aeronave
Micro	μ(Micro)	< 10	250	1	< 5	H,A,otros
Mini	Mini	< 10	150 a 300	< 2	< 30	H,A, P, Otros
Alcance cercano	CR	10 a 30	3.000	2 a 4	150	H,A,P,Otros
Alcance corto	SR	30 a 70	3.000	3 a 6	200	A,Otros
Alcance medio	MR	70 a 200	5.000	6 a 10	1.250	A, Otros
Altitud baja Penetración profunda	LADP	> 250	50 a 9.000	0,5 a 1	350	A
Autonomía media	MRE	> 500	8.000	10 a 18	1.250	A,H
Autonomía alta Altitud baja	LALE	> 500	3.000	> 24	< 30	A
Autonomía alta Altitud media	MALE	> 500	14.000	24 a 48	1.500	A,H
Autonomía alta Altitud alta	HALE	> 2000	20.000	24 a 48	12.000	A
Combate	UCAV	aprox. 1500	10.000	aprox. 2	10.000	H,A
Ofensivo	LETH	300	4.000	3 a 4	250	A
Señuelo	DEC	0 a 500	5.000	< 4	250	A,H
Estratosférico	STRATO	> 2000	Entre 20.000 y 30.000	> 48	ND (no disponible)	A
Exo-estratosférico	EXO	ND	> 30.000	ND	ND	A

Figura 1–2: Clasificación de los *UAVs* en cuanto a su capacidad de vuelo (fuente *U.A.V.S.I*).

1.1.3. Desarrollo de los *UAVs*.

Los *UAVs* hoy en día han llegado a un nivel completamente autónomo. Con el desarrollo de los micro-controladores y de los sensores inerciales de tecnología de estado sólido es posible fabricar estas máquinas a un precio asequible, siendo un área importante de la investigación aeronáutica y aeroespacial durante la última década (Adiprawita, 2007; Woo, 2007; Pajares, 2008; de˜Avila, 2010). La Figura 1–3 muestra el gráfico de desarrollo sobre el tema en el mundo (el eje Y : número de diferentes modelos desarrollados).

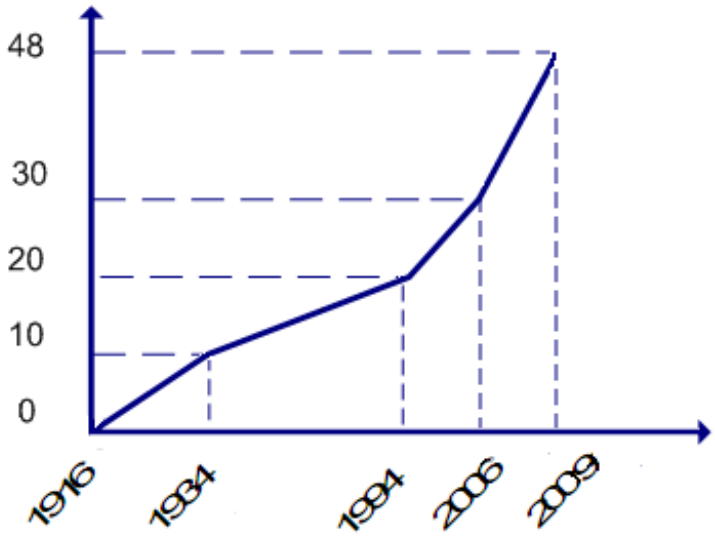


Figura 1–3: Desarrollo de los *UAVs* hoy en día.

Según fuentes de la *U.S.A.F* (*United States Air Force*) dentro de 10 años la tercera parte de las aeronaves militares operativas serán no tripulados. Actualmente *EEUU* es el pais que tiene más aplicaciones y mayor cantidad de *UAV* suponiendo que a medida que la potencia de los sistemas de bordo aumente, tanto en campo militar como el civil también crecerán (Asensio, 2008).

El diseño de estos vehículos aéreos de pequeño porte se considera un gran reto debido principalmente a las restricciones del peso de sus componentes y de su carga. Desde el punto de vista aerodinámico,se debe tener en cuenta los factores de diseño geométrico y de configuración, particularmente la geometría de la sección transversal de las superficies de sustentación o perfiles aerodinámicos. Cualquier mejora aerodinámica se verá reflejada en un incremento su rendimiento (Arias-Montaño, 2007; Pineda, 2008).

En los últimos años una gran cantidad de universidades y centros de investigación a nivel mundial, se encuentran investigando y desarrollando vehículos aéreos autónomos, destacándose por su maniobrabilidad los helicópteros autónomos (Mejias, 2006; Cordoba, 2007) y los aviones autónomos (Graversen, 2001), (Andersen, 2004; Fossen, 2011). Estas plataformas han sido utilizadas para investigar áreas que van desde el control no lineal (Al-Swailem, 2004), control inteligente (Abusleme˜H., 2000), control multivariable (Andersen, 2004), navegación (Torroella, 2011), planificación de trayectorias (Pajares, 2008), hasta la detección y seguimiento visual de objetivo (Mejias, 2006; Woo, 2007).

1.1.4. Ejemplos de *UAVs*.

- La Figura 1–4 muestra dos modelos de los *UAVs* con gran potencial para las aplicaciones *PSYOP* (*psychological operation*): (a) *Global Hawk* desarrollado por *Northrop Grumman-EEUU*, cuenta con una autonomía de 36 horas, la capacidad de carga útil hasta 2.000 libras, provisto de un radar de apertura sintética, penetra fácilmente en zonas nublosas o zonas con tormentas arenas; (b)MQ-1/RQ-1 Predator táctico (desarrollo por *General Atomic Aeronautical System -EEUU*) fue utilizado en misiones de inteligencia, reconocimiento y vigilancia del campo de batalla enemigo.

Figura 1–4: Los UAVs para PSYOP.

- Microdrones (Figura 1–5): Variante más pequeña de los *UAVs* permite realizar con éxito muchas actividades de uso civil como son fotografía aérea y periodista, TV, policía, etc...

Figura 1–5: Los Microdrones.

- *Smart Bird*(Figura 1–6) pesa medio de kilogramo, trabaja con batería de Litio. Es el último invento del mundo de la robótica basado en el movimiento de la gaviota. El robot es capaz de despegar, volar y aterrizar de forma autónoma.

Figura 1–6: *Smart Bird* autónomo.

- En la Figura 1–7 se muestran los dos aviones en los cuales se implementarán los autopilotos. Dichos aviones presentan características diferentes en cuanto a su peso y dimensiones, a pesar de esto se les puede tratar como plantas semejantes ya que aerodinámicamente presentan características similares como son los aviones subsónicos, de peso relativamente pequeño comparado con los aviones que transportan personas.

(a) CV10CT.

(b) N606LS.

Figura 1–7: Aviones en los cuales se implementarán los pilotos automáticos.

1.1.5. Aplicaciones.

El desarrollo de los vehículos autónomos no tripulados tiene un papel muy importante puesto que libera al hombre de múltiples tareas que podrían poner en riesgo su vida, así como también para ahorrar combustible, aumentar la seguridad de control que en algunas situaciones el hombre no es capaz de manejar. Entre las aplicaciones más comunes de los *UAVs* se encuentran:

★ **Militar**

La necesidad de crear armas más sofisticadas dio origen a las investigaciones de los *UAVs*. El prototipo más antiguo fue desarrollado después de la primera guerra mundial, y fue utilizado durante la segunda guerra mundial para entrenar a los operarios de los cañones antiaéreos. Sin embargo, no es hasta finales del siglo XX que los *UAVs* se operan mediante radio control con todas las características de autonomía.

★ **Ciencia y Entorno**

- Topografía: construye de mapas y deslinde de fincas (Asensio, 2008).
- Vigilancia forestal y prevención de incendios (Merino, 2006).
- Vigilancia fronteriza y costera (López, 2008).
- Meteorológica: Estudios atmosféricos y contaminación del medio ambiente.

★ **Transporte y Seguridad**

- Control de tráfico por carretera.
- Policial, vigilancia policial ciudadana: Sobrevuelo a media altura y potentes cámaras motorizadas.

- Vigilancia de viviendas y recintos por las compañías de seguridad.

★ **Industria**

- Eléctrica: Inspección de las líneas eléctricas de alto voltaje (Asensio, 2008).
- Agrícolas: Fotografía aérea agrícola o inmobiliario, fumigaciones, siembras.
- Telecomunicaciones: Seudo-satélite de comunicaciones de emergencia o de uso puntual. Volando a mínima velocidad puede permanecer días en el aire haciendo círculos sobre una misma posición a gran altitud(Barrientos, 2007).

★ **Civil**

- Desastres, emergencias, búsquedas de personas: Por las característica de autónoma, bajo consumo, peso ligero, el avión no tripulado puede permanecer en el aire por largo período de tiempo en las operaciones especiales, lo cual muestra la mayor eficacia en la exploración.
- Humanitario: Transporte de material humanitario a zonas de alto riesgo.

En Cuba el desarrollo en *UAV* es bastante limitado. El grupo de investigación *GARP* (Grupo de Automática, Robótica y Percepción) desde el 2008 se a dado a la tarea de investigar en campos relacionados con vehículos autónomos. Los resultados obtenidos están reflejados en tres tesis de ingeniería y dos tesis de maestría con algunas publicaciones científicas de la universidad *UCLV*. En las cuales se plantean los distintos problemas a los que se han enfrentado en el grupo para establecer una lógica para el modelado matemático de estos vehículos aéreos de seis grados de libertad (*DOF, Degrees of Freedom*) (Pineda, 2008). Hasta el momento el grupo no cuenta con una estructura de *hardware* definitiva capaz de realizar el control de vuelo además de la navegación y seguimiento de trayectorias, alcance de puntos ubicados en un mapa georeferenciado usando para esto solo sensores inerciales.

1.2. Requerimientos del Sistema.

Los modelos de *UAV* desarrollados en la *UCLV* cuentan con sistemas empotrados (sección 1.2.2) basados en el microcontrolador dsPIC30F4013 de capacidad de cálculo limitada solo al control de vuelo y la interacción con sensores y actuadores. Este sistema

es incapaz de realizar navegación inercial para llevar a cabo tareas de seguimiento de trayectorias o misiones de cualquier tipo. A esto se suma el hecho de que dentro de las posibles aplicaciones de mayor impacto se encuentren las relacionadas con captura, almacenamiento y procesamiento de imágenes en tiempo real. Por lo anteriormente expuesto podemos justificar el uso de procesadores con mayor potencia de cálculo en el nuevo diseño que nos permita contar con una plataforma de desarrollo realmente versátil y de amplias posibilidades.

En el mercado especializado en sistemas empotrados como los que requiere el sistema propuesto se puede encontrar infinidad de variantes, unas más potentes, otras más ligeras y de menos consumo, unos más costosos, otros de fácil acceso y prestaciones aceptables. A nuestro alcance se contaba con dos posibles variantes: la primera, una PC-104 de elevadas prestaciones pero consumo considerable, costo elevado y peso considerable para ir a bordo de aeronaves de pequeño porte. La segunda variante sería la MBTI2440. Esta es una computadora en una sola tarjeta (Figura 1–10) basada en procesadores con arquitectura *ARM9* y producida en el país por el Instituto Cubano de Investigaciones Digitales *ICID*. Esta tarjeta presenta muy bajo peso, bajo consumo, gran versatilidad para integrarse en sistemas mayores, y elevado poder de cálculo que unido al hecho de ser de fabricación nacional y soportar *SO LINUX* la hacen ideal para el sistema propuesto. En la Tabla 1–1 se muestra una pequeña comparación entre los medios de computo a nuestro alcance.

Tabla 1–1: Comparación de los *hardware* empotrados.

Aspecto	dsPIC	PC-104 (industrial)	MBTI2440
Frecuencia de W	120 MHz	1.67 GHz	400 MHz
ROM o caché	1 KB (EEPROM)	512 B caché	64 MB (NAND FLASH)
Flash	48 KB		1 MB (NOR FLASH)
RAM	2 KB	2 GB on board	32 MB (SDRAM)
SO	No utiliza	LINUX o WINDOWS	LINUX
Dimensión	53 x 16 mm	96 x 90 mm (2 capas)	104 x 73 mm
Peso	Más ligero	Pesado	Ligero
Potencia de Cálculo	Baja	Alta	Alta

1.2.1. Tareas de hardware

✓ **Lazo interno**:

Control de vuelo, esto es, rumbo y estabilidad. Para esto son necesarios 3 canales de *PWM*(*Pulse Width Modulation*) como señales de mando a lo servos. Un canal RS-232 para comunicación con *IMU*. Este es el sensor encargado de la retroalimentación en los lazos de control (Figura 1–8). Es necesario además una entrada analógica para sensar el nivel de baterías.

✓ **Lazo externo**:

Realiza la tarea de navegación y seguimiento de trayectorias aunque en un futuro pudiera utilizarse en la captura, almacenamiento y procesamiento de imágenes.

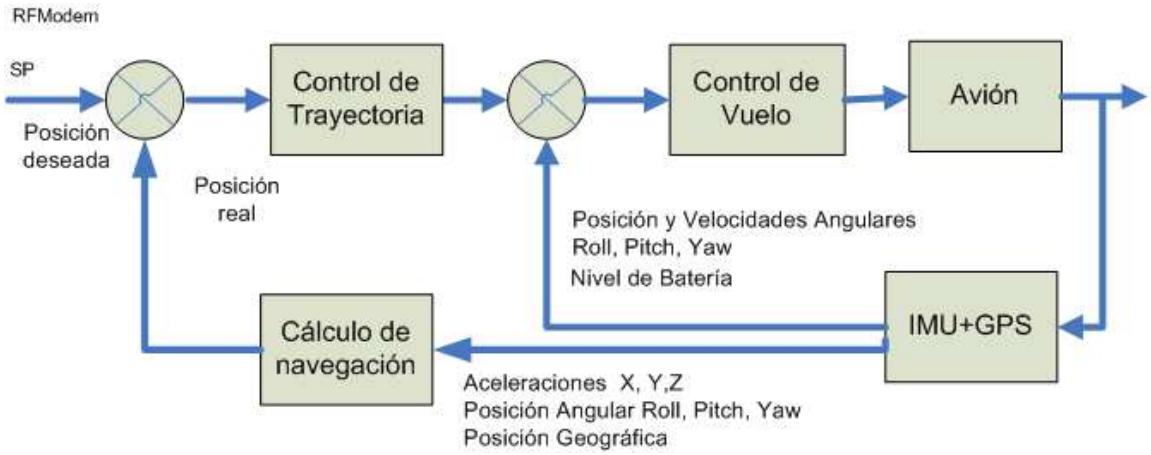


Figura 1–8: Estrategia de navegación y control del vuelo.

1.2.2. Sistema Empotrado.

Un sistema empotrado es una plataforma de cómputo que forma parte de un sistema de ingeniería más amplio, bien un horno microondas, un central nuclear, un sistema de guía de misiles, un automóvil o en este caso un *UAV*. En la última década el interés por los sistemas empotrados ha aumentado notablemente, esto se debe en gran medida a su aplicación en los sistemas de tiempo real.

a) Características de los sistemas empotrados.

Los sistemas empotrados presentan características que justifican su elección en disímiles aplicaciones. A continuación se enumeran algunas de ellas:

- Compacto, ligero, de bajo de consumo, alta fiabilidad y seguridad.
- Capaz de ejecutar algoritmos de elevado costo computacional en tiempo real.

- Realiza un conjunto limitado de funciones que no suelen ser de propósitos generales.
- Emplea una combinación de recursos de hardware y software para realizar una tarea específica.
- Maximiza la robustez ya que las condiciones de uso pueden ser extremas.
- Interactúa con los dispositivos físicos a través de dispositivos E/S no usuales, por lo que suele ser necesario un acondicionamiento de las diferentes señales.
- El co-diseño *hardware/software* tiene un papel importante e indispensable ya que la tarea de diseño del *hardware* y la generación de código debe ser un proceso mixto.

b) **Capas de un sistema empotrado.**

Un sistema empotrado cuenta con cuatro capas siendo la capa de hardware la de más bajo nivel. A continuación se presentan las capas de un sistema empotrado:

- *Hardware.*
- *Firmware.*
- Sistema operativo.
- Aplicación.

El *software* ocupa la mayor parte de las capas, sin embargo el número de las capas pudiera reducirse a tres en dependencia de la complejidad del sistema, muchos de estos pudieran prescindir de un sistema operativo. A menudo un sistema empotrado trabaja con un simple programa dedicado a una tarea en específico. En la Figura 1–9 se muestran las capas de los diferentes sistemas de cómputo, partiendo desde la computadora de escritorio hasta los simples sistemas empotrados.

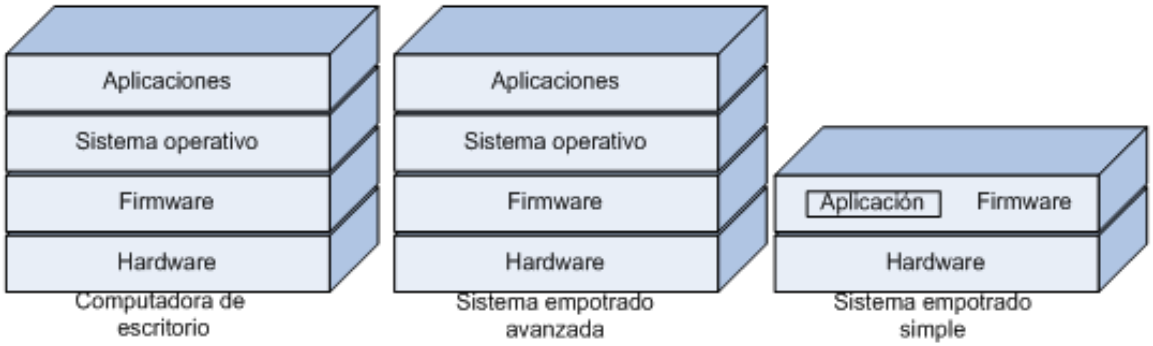


Figura 1–9: Comparación las capas del sistema empotrado con una computadora.

Típicamente una capa sólo consigue interactuar con la capa inmediata superior o inferior a esta (Guerra, 2010). El sistema propuesto realmente representa dos sistemas empotrados en uno, o sea un sistema empotrado avanzado (MBTI2440) unido a uno simple (dsPIC30F4013) comunicados por *SPI*.

1.3. Arquitectura general del sistema propuesto.

En esta última área es que se enmarca esta investigación, cuyo objetivo es desarrollar una estructura de hardware de bajo costo y poco peso para pequeños aviones no tripulados con capacidades de control de vuelo y navegación inercial basada en modelos, esta última, de elevada complejidad y costo computacional.

La arquitectura de hardware que se propone está compuesta por la computadora en una sola tarjeta MBTI2440 (Figura 1–10) de fabricación nacional, la cual cuenta con el microprocesador con arquitectura AMR9, de 32 bits, 400 MHz y elevadas prestaciones, S3C2440 de **Samsung**. Esta tarjeta será la encargada de la navegación, seguimiento de trayectorias, almacenamiento de datos de vuelo etc., tareas que debido a la complejidad y el costo computacional de las mismas justifican su uso.

Figura 1–10: Tarjeta MBTI2440.

Para desempeñar la tarea del control de vuelo, o sea rumbo y estabilidad, se utilizará el microcontrolador de gama alta dsPIC30F4013. Además será el encargado de la integración del sistema sensorial del avión, que incluye un *GPS* y una unidad de medición inercial *IMU*, así como la administración de energía. Todo el sistema irá montado en circuito impreso

tipo sándwich (Figura 1–11) donde el elemento superior del sándwich será la MBTI2440 sin modificaciones y debajo la tarjeta basada en el dsPIC30F4013 con un grupo de conectores tales como RS-232, Ethernet, memoria SD así como los conectores de expansión a la MBTI2440 y los elementos propios para el controlador de vuelo. La comunicación entre tarjetas será vía *SPI*.

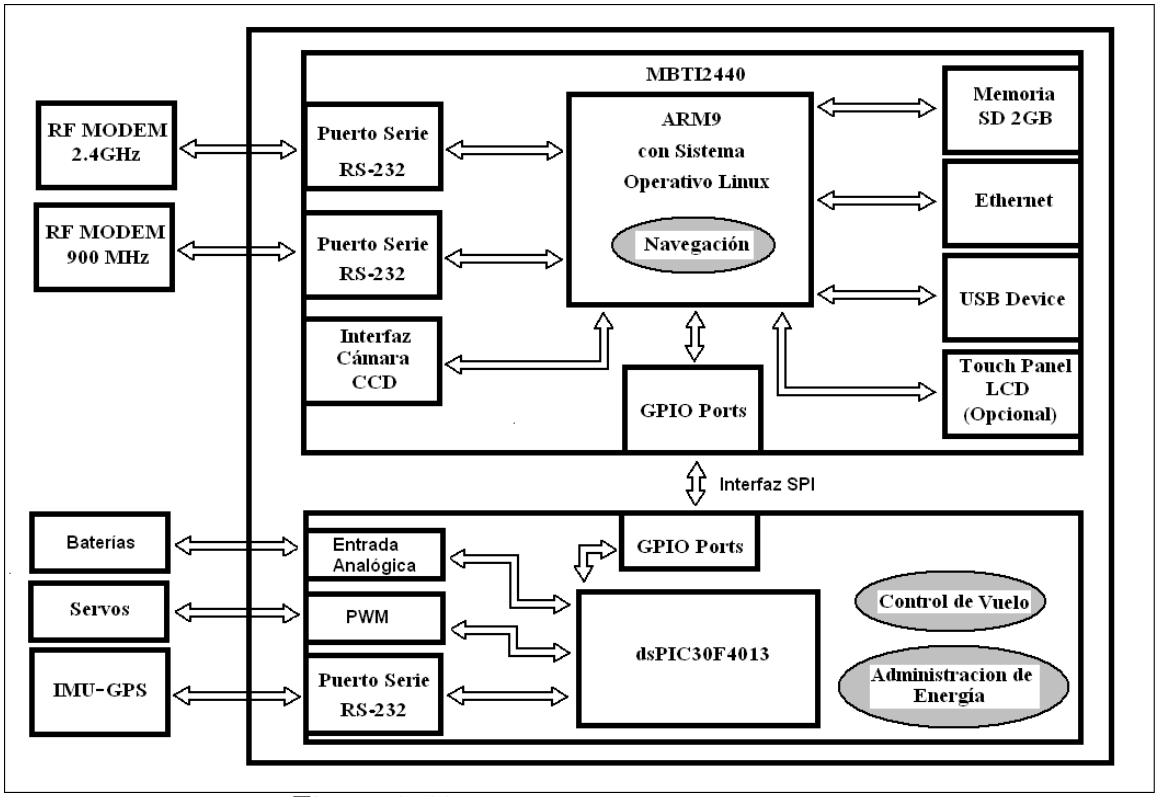


Figura 1–11: Estructura general del sistema.

Aunque MBTI2440 es capaz por si misma de ejecutar la navegación y demás tareas de alto nivel más el control de vuelo, en este trabajo se decidió mantener el microcontrolador *PIC* puesto que las aeronaves utilizan servomotores cuya señal de mando es un *PWM* de características especiales y la tarjeta MBTI2440 es incapaz por si misma de generar los cuatro canales *PWM* necesarios. Es posible que en un futuro y para nuevos diseños se utilice para esta función algún otro elemento o incluso un *PIC* de menos prestaciones y por ende menos costo, realizando la MBTI2440 todo el control de la aeronave.

1.4. Conclusiones del capítulo.

En este capítulo se ha realizado un análisis introductorio al tema de los *UAVs*, su historia, principales conceptos y aplicaciones. Sistemas como el propuesto no se encuentran

reportados en la literatura. Las ventajas que brinda la utilización de este tipo de *hardware* es ilimitada ya que al contar con sistema operativo estándar disfruta de todas las bondades del mismo, tal es el caso por ejemplo, del sistema de archivos que permitirá registrar datos de vuelo, imágenes tomadas de la cámara de a bordo etc. La otra gran ventaja es la relación peso-capacidad de cómputo que comparándola con otras arquitecturas para control empotrado, por ejemplo una PC-104, la MBTI2440 ofrece elevado poder de cómputo con bajo consumo y muy bajo peso.

CAPÍTULO 2
ELEMENTOS DE LA PROPUESTA GENERAL DEL *HARDWARE* .

2.1. Introducción.

En este capítulo se describe el diseño de *hardware* propuesto para el *UAV*. Se exponen los medios de computo utilizados y se incluyen la características de los sensores y actuadores así como componentes auxiliares del sistema tales como comunicación y demás. Finalmente se hace referencia al software que dará la vida al sistema.

2.2. Unidad de medición Inercial *IMU (MTi-G)*:

Los sistemas *UAV* además de requerir un hardware empotrado a bordo, necesitan conocer el estado del vehículo y su entorno en todo momento para desarrollar las misiones. Esto se realiza por medio de los sensores distribuidos en el mismo. Entre los sensores más utilizados en este tipo de aplicación se encuentran: acelerómetros, giróscopos, magnetómetros, altímetros, modems acústicos, sonares, sensores de temperatura, *GPS* entre otros.

En la actualidad existen en el mercado una gran cantidad de dispositivos que integran varios de estos sensores y brindan la información a través de algún tipo de bus o interfaz. Tal es el caso de las llamadas *IMUs* o Unidades de Medición Inercial por sus siglas en inglés. El sistema propuesto será capaz de interactuar con el sensor de tipo *IMU + GPS MTi-G* cuya estructura interna y aspecto exterior se muestran en la Figura 2–1. A las diferentes variables que brinda este sensor se accede a través de comunicación serie RS-232 estándar.

La IMU MTi-G mide las aceleraciones lineales en los ejes *XYZ*, los ángulos de orientación *Roll, Pitch* y *Yaw* así como sus velocidades de rotación. Este sensor es capaz

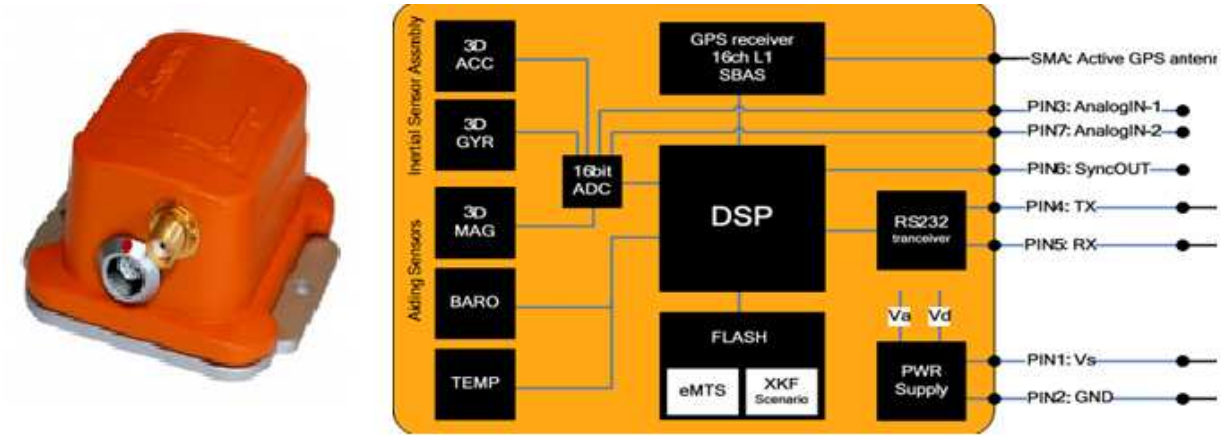


Figura 2–1: Aspecto exterior y estructura del sensor inercial *MTi-G*.

de medir temperatura, presión y orientación magnética, todo esto con gran ancho de banda. Su procesador interno es de bajo consumo, provee una orientación 3D sin deriva y la posibilidad de obtener los datos calibrados. El *MTi/MTi-G* es una unidad de medida excelente para estabilización y control de cámaras, robots, vehículos y otros equipamientos, así como para capturar movimiento, animaciones, realidad virtual, ejercicios y deportes (más información en anexo A).

Es de sobra conocido que los valores de posición medidos con este sensor sólo son fiables durante un breve periodo de tiempo. Esto es debido al error de deslizamiento inherente a los acelerómetros, que hace que al realizar la doble integración, para obtener primero la velocidad y posteriormente la posición, se obtenga un error nada despreciable que se irá acumulando con el tiempo, con lo que la posición estimada distará sensiblemente de la posición real.

2.3. Unidad de comunicación RF *(Radio frequency)*:

El sistema de comunicación por RF permitirá realizar el despegue y aterrizaje al piloto humano. Además por este medio se podrá supervisar el estado del vehículo. En el sistema propuesto se optó por utilizar dos canales, uno a 900 MHz y otro a 2.4GHz- esto por cuestiones de seguridad y robustez además de permitir mayor ancho de banda en la comunicación en caso de no usarse los dos canales de manera redundante.

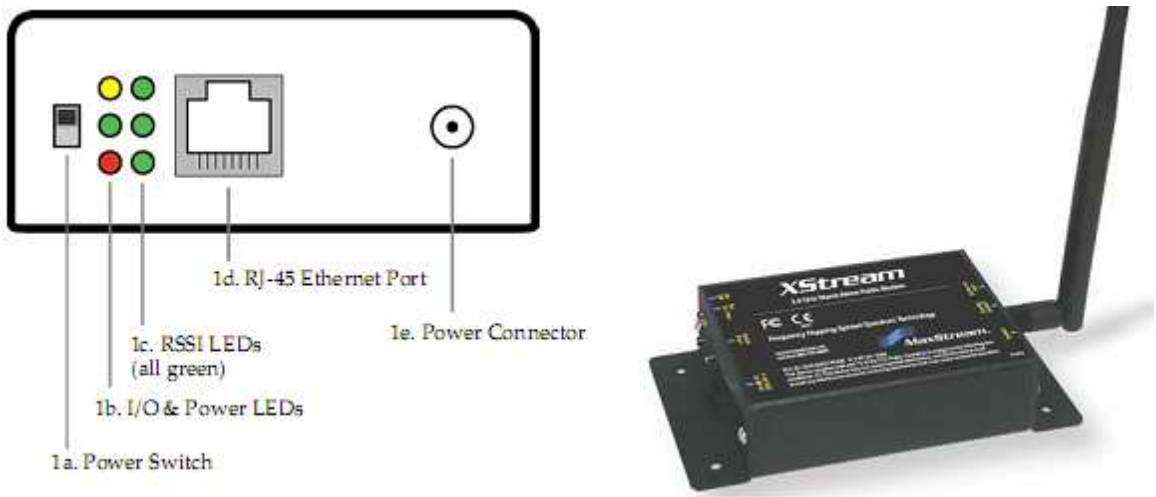


Figura 2–2: Vista frontal de RFmodem

Características técnicas:

- Rango de trabajo: 64km (w/alto ganancia)-14km (w/2.1 dB dipolo).
- Potencia de salida: 1 mW1W (0-30 dBm).
- Sensibilidad: -110 dBm (9600bps data rate).
- Velocidad de transmisión de datos: 10.000 bps- 125.000bps.
- Temperatura de operación: -40°C-85°C.
- Modulación: *FSK* (Modulación por desplazamiento de Frecuencia).
- Topología de red soportada: Punto a Punto. Punto a Multipunto. Repetidor.

Tabla 2–1: Transmisión RFmodem.

Potencia de transmisión	1	10	100	500	1.000	mW
Voltaje de suministro	2.8-5.5	2.8-5.5	2.8-5.5	2.8-5.5	2.8-5.5	V DC
Corriente de transmisión (5V)	110	140	270	500	730	mA
Corriente de transmisión(3.3 V)	90	100	260	600	N/A	mA

2.4. Unidad de actuación:

Los elementos de actuación en el sistemas son los servomotores Figura 2–3 los cuales son los encargados de variar mediante transmisiones y palancas el ángulo de los timones de altitud, timón de dirección y los alerones de la aeronave. La señal de mando al servomotor es un *PWM* cuyas características se muestran en la Figura 2–4.

El eje de rendimiento puede ser llevado a posiciones angulares específicas cuando se envía una señal codificada. Una vez que la señal codificada exista en la línea de entrada,

Figura 2–3: Servomotor Futaba.

el servo mantendrá la posición angular del engranaje. Cuando cambia la señal cambia la posición angular de los piñones. En la práctica se usan servos para posicionar superficies de control, el movimiento de palancas y timones. Se usan también en radio control, títeres y en robots. El motor de servo es pequeño, dirigido por una circuitería de control interna y es sumamente poderoso para su tamaño. En este proyecto se utilizó el servo motor S3013 Futaba, el cual posee unas dimensiones de 1.10x0.51x1.16 pulgada, un peso de 0.6 onzas generando un torque de 27.8 onzas/pulgada (a 3.3 V).

- **Modo de comunicar el ángulo al cual el servo debe posicionarse:**

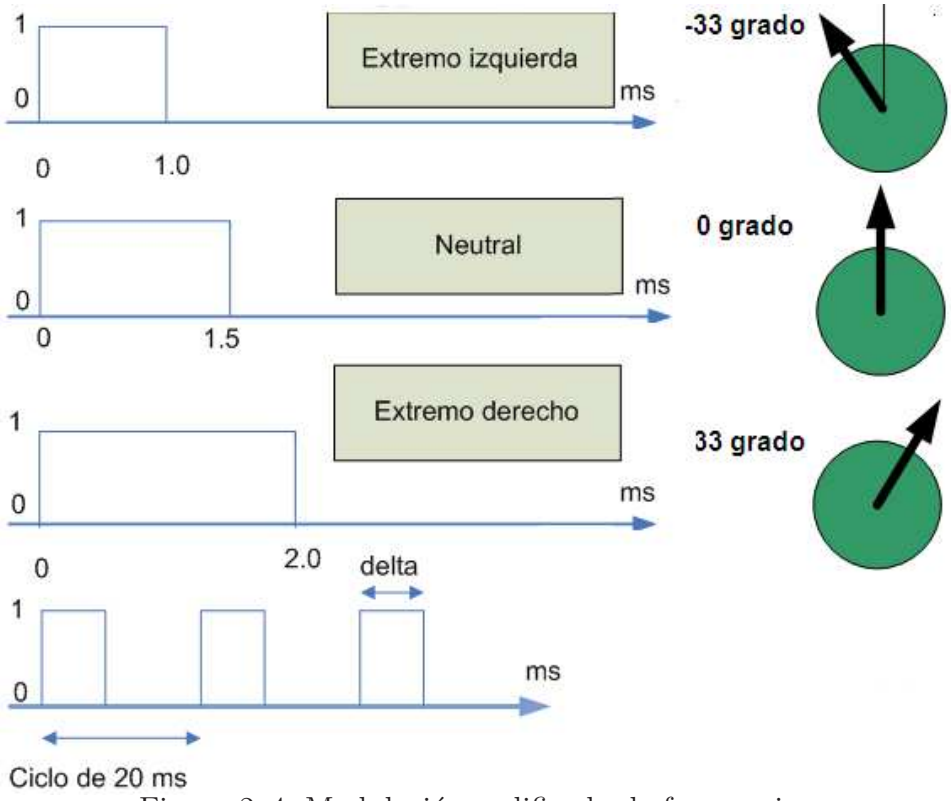


Figura 2–4: Modulación codificada de frecuencia .

- **Distribuciones de cables Futaba:**
  - Señal de control (Amarillo o blanco).
  - Vcc (Rojo).
  - GND (negro).

2.5. Las interfaces de comunicación

2.5.1. Interfaz *SPI*:

La interfaz *SPI* es una de las más comúnmente utilizadas para la comunicación entre periféricos ya que esta es sincrónica y logra alcanzar altas velocidades con gran fiabilidad. Entre los dispositivos que generalmente presentan esta interface se encuentran, memoria no volátil, sensores, conversores análogo-digitales, conversores digito-analógico, entre otros (Catsoulis, 2005). El bus SPI transmite los paquetes de 8 o 16 bits utilizando un simple registro de desplazamiento sincronizado por señal de reloj. El bus contiene tres líneas: CSLK, MOSI (*Master out, Slave In*), MISO (*Master In, Slave Out*). Todas las líneas trasmiten la información sobre una dirección controlada por 2 bits (4 opciones) de configuración CPOL (*clock polarity*) y CPHA (*clock phase*). Así que los dispositivos que quieren comunicarse entre sí deben tener la misma configuración.

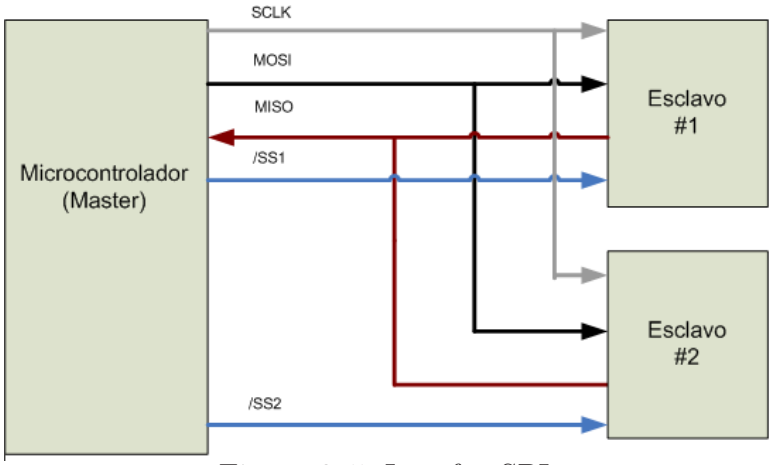


Figura 2–5: Interfaz SPI.

2.5.2. Interfaz RS232:

RS232 es una interfaz de comunicación estándar, es la más usada en temas de diseños empotrados, permitiendo la comunicación con dispositivos a distancias relativamente grandes,

por encima de los 15m con gran fiabilidad, alta velocidad de transferencia de datos. Es la interfaz de comunicación más común de muchos tipos de sensores y dispositivos. En este diseño es utilizada para establecer la comunicación entre los RFmodems con la placa de base y entre el sensor *IMU-GPS* con el dsPIC.

El micro-controlador dsPIC presenta UART que genera tramas con estructura similar a RS232 excepto por los niveles lógicos. El micro-controlador utiliza la lógica de *TTL* con niveles de 0-5 v, mientras que RS232 tiene la lógica negada, bipolar y con niveles de voltaje que varían entre ±3v a ±15v. Para establecer la compatibilidad es necesario adicionar un componente externo desplazador de nivel (Catsoulis, 2005). En este caso se utilizará el circuito integrado MAX235 que es la variante más conveniente para aprovechar la cantidad de canales de transmisión. Esa pastilla soporta cinco canales de RS232 para establecer la comunicación entre dos RFModems con la tarjeta MBTI2440 y el otro entre *IMU (TMi-G)* con el dsPIC.

2.6. MicroControlador dsPIC30F4013

Este microcontrolador destaca por su capacidad de procesamiento, versatilidad y diversidad de interfaces disponibles. Puede alcanzar un desempeño máximo de 30 millones de instrucciones por segundo (*MPIs*) con un oscilador configurable para distintas frecuencias de ejecución.

- 48 KB *Flash*,2 KB *RAM*, 1 KB *EEPROM* (no volátil).
- Un amplio conjunto de instrucciones(84 INS básicas).
- 8 niveles de prioridad de interrupción.
- 5 temporizadores de 16 bits.
- 4 comparadores y moduladores de ancho de pulso *(PWM)* permiten recrear y capturar una amplia variedad de señales.

El dsPIC posee 13 entradas analógicas con un convertidor de 12 bit de resolución y velocidad de conversión de 200 Ksps (kilo muestras por segundo), lo que satisface la totalidad de las necesidades de la aplicación. Presenta las interfaces de comunicación serie

más utilizadas en la actualidad por los distintos tipos de dispositivos empleados en sistemas empotrados, los cuales son: dos módulos *UART* (*Universal Asynchronous Receptor Transmitter*), un *SPI* (*Serial Peripheral Interface*), un *I2C* (*Inter-Integrated Circuit*), un modulo *CAN* (*Controller Area Network*) y un *DCI* (*Data Conversion Interface*).

2.7. Unidad MBTI2440

Es una computadora en una tarjeta basada en el procesador S3C2440A3 de *SAMSUNG* con arquitectura AMR9, fue diseñado con el objetivo de obtener una tarjeta de procesamiento de bajo costo y pequeño tamaño, bajo consumo de potencia (400mA, 5V DC), dimensiones de 104x73mm, con gran variedad de interfaces y elevada conectividad. La tarjeta no posee los conectores estándar de interconexión con los periféricos y dispositivos de interfaz. Todas las líneas de entrada y salida de la MBTI2440 están conectadas a tres conectores de 25x2 contactos. Es ideal para ser utilizada como hardware empotrado en cualquier proyecto. Para emplear la MBTI2440 los usuarios deben desarrollar una tarjeta base que se conectará a la misma, en la cual se colocarán los conectores requeridos en las posiciones apropiadas para cada aplicación. Partes que suministran:

- Tarjeta Base de desarrollo, tarjeta MBTI2440, memoria Flash SD 2 GB.
- Modulo de Cristal líquido *TFT* (*Thin-Film Transistor*) colores 320x240 puntos con cables de conexión.
- Cable USB A-B, COM serie (adaptador macho/hembra), de programación *JTAG*.
- Adaptador de corriente alterna a directa (5V/1.0 A).

★ Microprocesador *AMR9*:

En el mercado especializado existe una gran variedad de tecnologías de sistemas empotrados dentro de las cuales destacan las basadas en procesadores INTEL y ARM. Los hardware empotrados basados en AMR9 son los más extendidos por sus diversas características que favorecen a las aplicaciones con restricciones temporales (González-Vázquez, 2010). La tarjeta empleada en este proyecto está basada en el procesador S3C2440 cuyo

núcleo es un AMR920T de 130nm. Cuenta con un diseño completamente estático muy adaptable a aplicaciones de bajo costo (SAMSUNG˜ELECTRONIC, 2004).

2.8. Distribución de Linux 2.6.29

GNU/Linux es el sistema operativo (SO) de código abierto más extendido en los sistemas empotrados pero su diseño no está realizado para brindar servicios de tiempo real. Existen varias soluciones de código abierto para hacer de *GNU/Linux* un sistema operativo con servicios de tiempo real y algunas de ellas con méritos suficientes para ser consideradas a la hora de elaborar nuestra solución(González-Vázquez, 2010). La MBTI2440 es distribuida con la versión 2.6.29 de Linux para sistemas empotrados con la resolución de los Timers de 1 ms lo que ayuda a mejorar el tiempo de repuesta. Esta versión no está orientada al trabajo en tiempo real no obstante existen algunas soluciones que se acercan mucho brindando excelentes resultados. Para esto, así como para activar en el kernel algunas funcionalidades no presentes, tales como la comunicación *SPI* y lo puertos digitales de propósito general (*GIPO*) fue necesario recompilar y reinstalar la version inicial además fue necesario reconfigurar algunos elementos en la secuencia de arranque e instalar un servicio de acceso remoto. Además se le instalaron herramientas de programación de tecnología QT que facilitan y dinamizan el proceso de programación, test y puesta a punto.

★ La ventaja de los módulos de Kernel:

- Activan las funcionalidades cuando se necesiten.
- Maximizan el uso de la memoria.
- Facilitan el desarrollo de los *drivers*.

2.9. Consideraciones finales del capítulo:

En este punto contamos con toda la base teórica, los requerimientos del sistema y las herramientas necesarias por lo que se puede concluir que estos cumplen las exigencias que imponen los requisitos del proyecto. Todos los componentes están disponibles para realizar el montaje, teste y puesta a punto del sistema.

CAPÍTULO 3
IMPLEMENTACIÓN DEL SISTEMA PROPUESTO

3.1. Introducción

En este tercer y último capítulo detallamos el diseño obtenido justificando la selección de los componentes. Seguidamente hacemos referencia a las herramientas implicadas en el proceso de programación y puesta a punto y finalmente brindamos algunas consideraciones sobre los algoritmos básicos con requerimientos temporales críticos.

3.2. El diseño final

La figura 3–1 muestra el esquema del sistema final realizado en *PROTEUS ISIS* versión 7.7. El corazón de este diseño es el dsPIC30F4013, encargado del control de rumbo y estabilidad, la administración de energía, la adquisición de datos provenientes de la *IMU*, la generación de las señales de mando a los servos y la comunicación vía *SPI* con la MBTI2440. Además están presentes en el diseño un convertidor de normas de *TTL* a RS-232 de cinco canales, MAX235, de las cuales solo se utilizan tres.

El diseño cuenta además con un grupo de conectores que dan funcionalidad a la MBTI2440 tales como: conector de memoria SD, tres conectores DB9 machos, de los cuales dos los utiliza la MBTI2440 y uno el dsPIC, un conector de Ethernet RJ-45 hembra con su respectivo transformador. Los conectores CON1 y CON2 conectan directamente con sus homólogos en la MBTI2440 y a través de los cuales fluyen los datos de la memoria SD, Ethernet, dos de los puertos serie y el puerto de comunicación *SPI* entre los procesadores dsPIC y ARM9. Finalmente encontramos los conectores J4, J5, J6 y J7 destinados a la conexión con los servomotores. El conector J8 para la alimentación que puede ser de 13V

a 18V puesto que pasa a través del regulador de voltaje integrado U3 7805 que brinda a todo el sistema una tensión constante de 5V y hasta 1A. El conector J9 es utilizado sólo en el proceso de grabación de la memoria de programa del dsPIC así como para cuestiones de debugueo. El conector J10 se utiliza para el cristal de cuarzo del dsPIC y el J11 para conectar un LED que indica el estado del hardware.

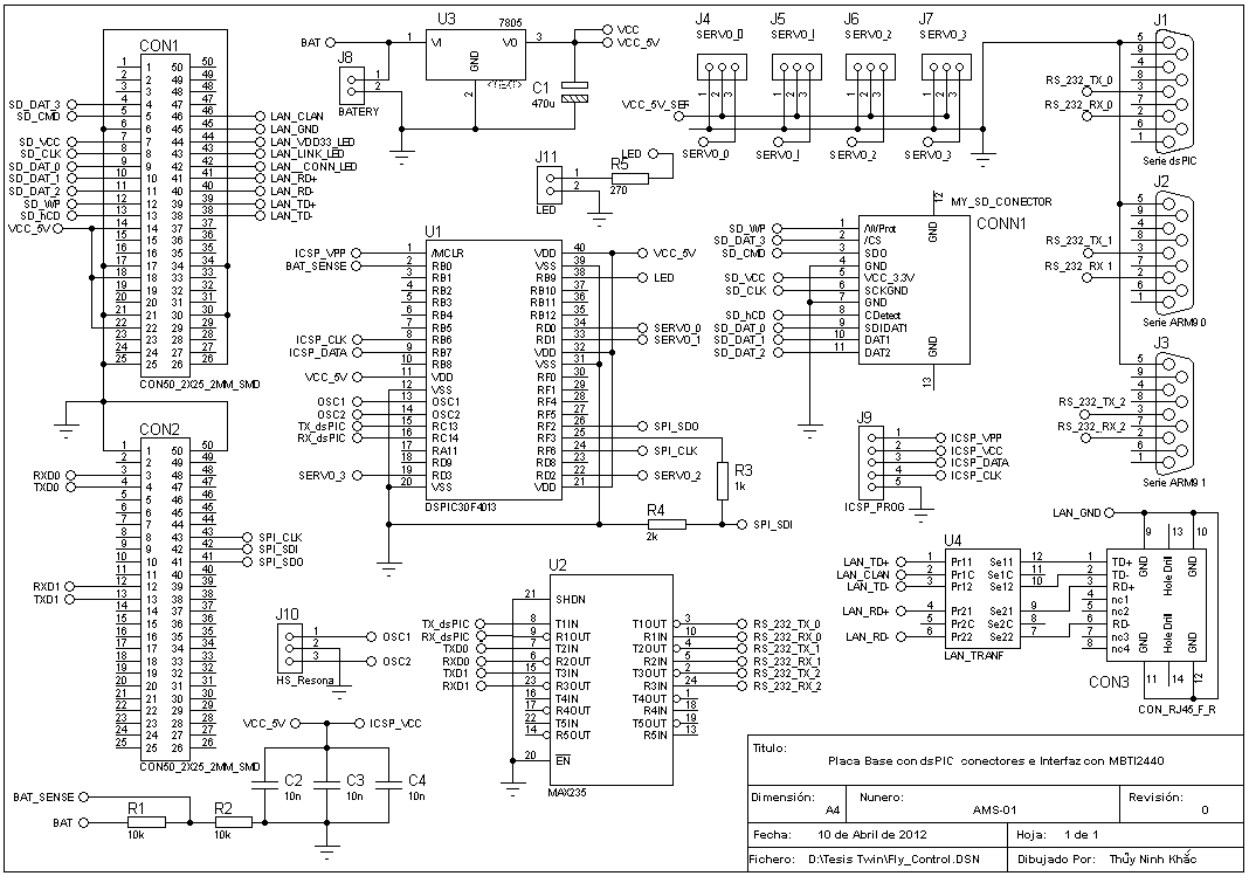


Figura 3–1: Esquema final.

La figura 3–2 muestra el circuito impreso obtenido para la versión prototipo de prueba. Este PCB fue realizado utilizando *PROTEUS ARES* v7.7. Buscando mayor simplicidad a la hora del montaje optamos por el diseño simple cara, las venas de color rojo correspondientes a la cara de los componentes serán hechas con puentes de cables. Con dimensiones de 140x86mm se ajusta perfectamente dentro de cualquier aeronave de pequeño porte.

Y la tarjeta MBTI2440 (dimensiones de 104x73mm) se acopla en la tarjeta base como se muestra en la Figura 3–3.

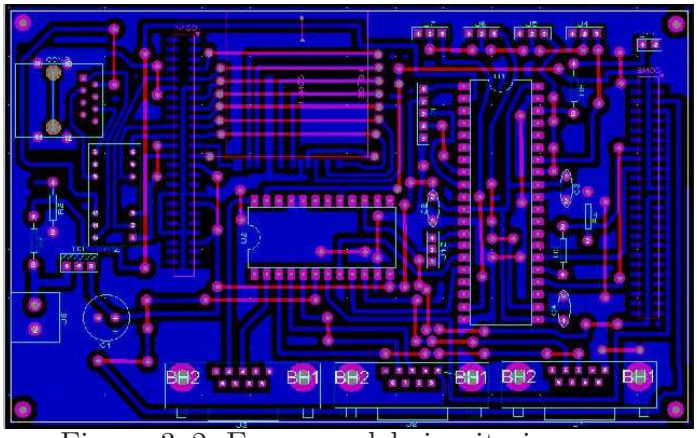


Figura 3–2: Esquema del circuito impreso.

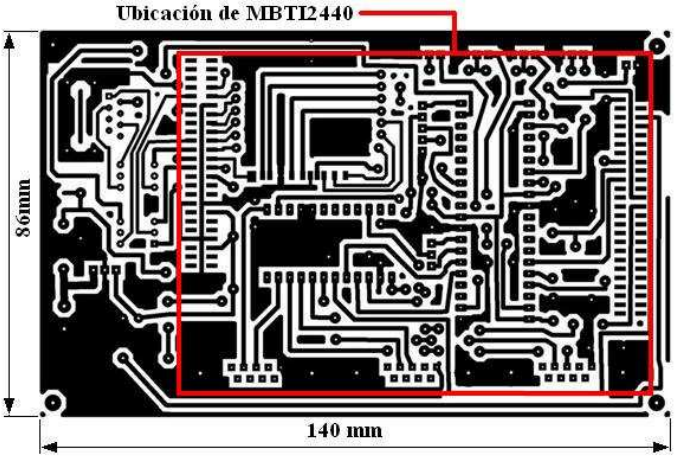


Figura 3–3: Ubicación de la MBTI2440 y dsPIC.

3.3. La compilación del Kernel

La distribución 2.6.29 del *Kernel* de *Linux* suministrada por el fabricante de la tarjeta MBTI2440 debe ser recompilada para activar un grupo de funcionalidades no presentes. Tal es el caso de los *drivers* para la comunicación *SPI, GPIO*, así como para hacer al sistema preemptivo, esto significa que el sistema tendrá baja latencia en la respuesta a las interrupciones acercando su funcionamiento a un sistema de tiempo real. El trabajo de cross-compilación del núcleo se realiza sobre *SO GNU/Linux*, en este caso se utiliza la version 6 de Debian. Para esto se debe añadir su dirección al *PATH*(variable de entorno que contiene una serie de rutas de directorios donde se realiza la búsqueda cada vez que se pretende ejecutar una aplicación) del sistema como se muestra en el siguiente comando:

export PATH=/home/...(direccion del compilador)/:$PATH

Con el comando **apt-get install ncurses 5.6** se realiza la instalación del programa **ncurses** en la PC. Una vez instalados el compilador y el **ncurses** se localiza en la carpeta donde se encuentra el Kernel, desde **Terminal GNOME** del *SO* se introduce el comando **make menuconfig**. Este comando ejecuta el programa **ncurses** (incluye los controladores *GPIO* Figura 3–4) el cual carga un fichero de configuración genérico o definido por el usuario en el cual se agregan o desactivan funcionalidades del SO quedando de esta manera el *Kernel* acorde a nuestras necesidades. Tras de ejecutar el comando **make zImage** se tiene lista el imagen del *Kernel* que se debe cargar a la tarjeta MBTI2440.

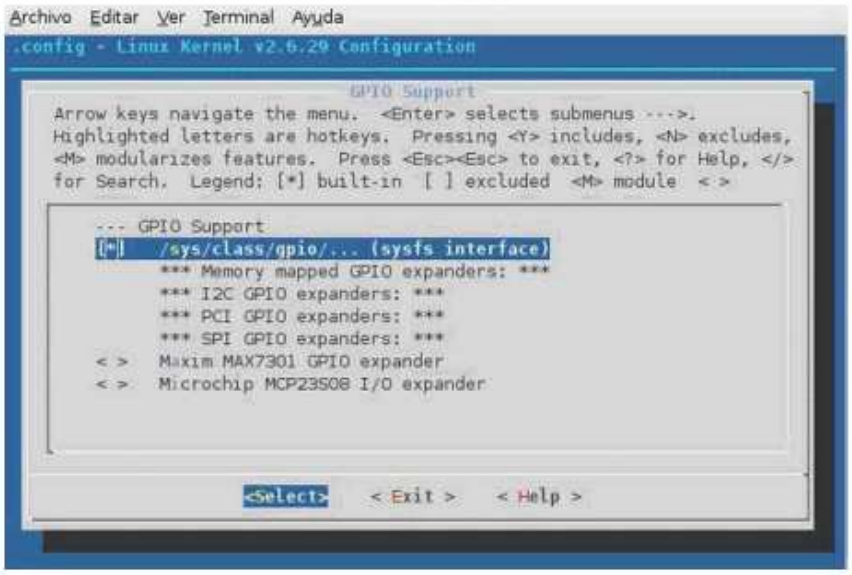


Figura 3–4: Añadir *GPIO* a través **ncurses**.

Pasos para cargar imagen a la tarjeta:

- Se habilita la memoria NOR Flash de la tarjeta MBTI2440 (en esta memoria se ejecuta un programa que permite almacenar en memoria RAM la imagen del kernel).
- En WINDOWS XP se instalan también los programas ***DNW*** y **supervivi**, el ***DNW*** es el software que se comunica con la tarjeta MBTI2440 y el **supervivi** es el responsable de cargar el *Kernel* para el arranque del sistema.
- Con el ejecutable WRITE.BIN se escriben en la memoria NAND Flash el *software* **supervivi** y la imagen del *Kernel*. Para realizar estas operaciones se necesita establecer dos conexiones con la tarjeta MBTI2440, una conexión serie (*master*) para ejecutar

instrucciones y una conexión USB (*slave*) para la copia de los ficheros en la memoria NAND Flash.

- Después de compilado se debe borrar la memoria NAND Flash de nuestra tarjeta, luego se le graba el programa **supervivi** en la dirección **0X30100000** y posteriormente se le graba la imagen del kernel en la dirección **0X30130000**.

3.4. Algunos algoritmos básicos

El sistema propuesto lejos de constituir un producto final lo que realmente pretende es que sea usado para desarrollar algoritmos y métodos de navegación en *UAV*. Por tal razón el sistema debe contar con una pre-programación de los algoritmos o programas básicos de desarrollo. La Figura 3–5 muestra los algoritmos básicos que ejecuta el dsPIC, el código fuente puede ser consultado en los anexos B, aquí se podrían modificar por parte del programador los algoritmos o ecuaciones del control de vuelo porque depende de la aeronave en cuestión. Todo lo demás, referente a la comunicación, generación de interrupciones y *PWM* etc. permanece invariable para cualquier aplicación.

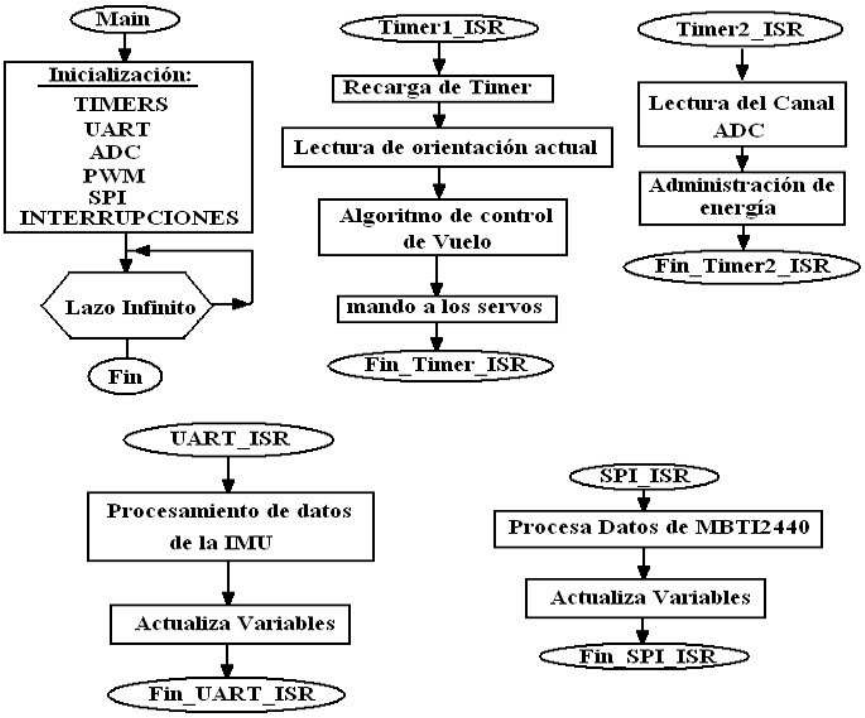


Figura 3–5: Algoritmos básicos desarrollados en el dsPIC .

La Figura 3–6 muestra la contraparte en la MBTI2440, de la misma forma sólo varía en función del desarrollo que se quiera hacer lo referente al algoritmo de navegación.

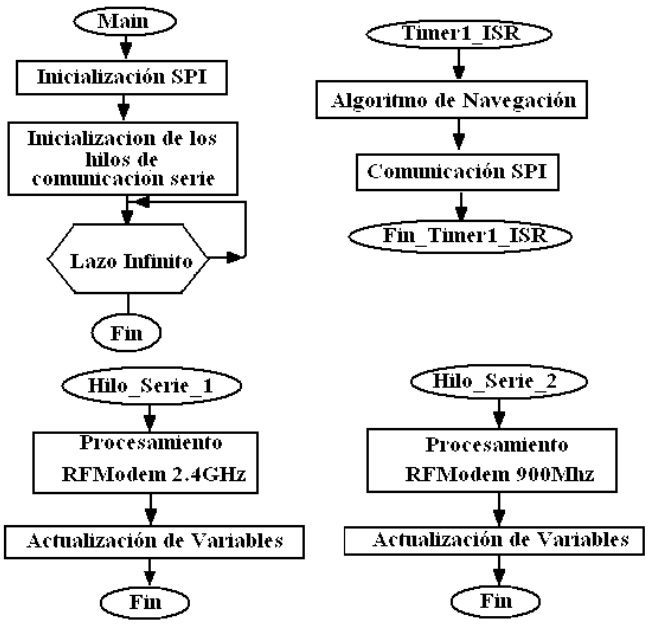


Figura 3–6: Algoritmos básicos desarrollados en MBTI2440 .

3.5. Análisis económico

El sistema obtenido presenta un bajo costo en relación con su capacidad de cómputo lo cual es muy importante tomando en cuenta que se desea utilizar en la ejecución de algoritmos de navegación inercial los cuales implican elevada capacidad de cálculo. Comparados con los sistemas implementados con anterioridad en nuestro grupo de investigación por (Pineda, 2011; Guerra, 2010), el primero utilizando dsPIC como medio de cómputo carecía de potencia de cálculo para navegación inercial y el segundo, usando PC-104, que aunque es válido para la aplicación en vehículos sumergibles en el caso nuestro no sería aplicable por su peso y consumo, además de representar un elevado costo comparado con la MBTI2440. La siguiente tabla muestra el costo total del sistema propuesto.

Tabla 3–1: Costo del *hardware*.

Aspecto	Sistema propuesta	PC-104 (industrial)	unidad
MBTI2440	70	1.000	CUC
dsPIC30F	10		CUC
Misceláneos	70		CUC
Total	150	1.000	CUC

3.6. Consideraciones finales del capítulo

En este capítulo hemos expuesto en detalle el hardware propuesto, así como todos los diferentes elementos implicados. Quedan claramente dichas las potencialidades y ventajas de esta propuesta en cuanto a potencia de cálculo, peso, consumo y costo. Además se muestra el circuito impreso obtenido de fácil fabricación y montaje lo que lo hace realmente compacto y muy fiable. Finalmente se muestran los algoritmos básicos, útiles para el futuro desarrollador de algoritmos de navegación.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Con el presente trabajo se logró desarrollar un sistema empotrado capaz de realizar navegación inercial basada en modelos para *UAV* utilizando para esto hardware de bajo costo, de fabricación nacional y de elevadas prestaciones. Este hardware, dicho sea de paso, está teniendo un gran impacto en el país gracias a las disímiles aplicaciones que tiene, no obstante adolece aun de interfaces para uso industrial y de un sistema operativo de tiempo real. En este trabajo se logró una plataforma de desarrollo realmente poderosa y versátil que abre el espectro de aplicaciones de la MBTI2440 en aplicaciones empotradas y le da un nuevo rumbo al *GARP* en el desarrollo de autopilotos para *UAV* o vehículos autónomos en general.

Recomendaciones

Como recomendaciones del trabajo se plantea que una vez montado y testado el sistema sea introducido en el mini avión que posee el *GARP* para validar su funcionamiento ya que el grupo posee algoritmos de navegación inercial con buenos resultados que no han sido probados aun en un *UAV* real. De obtenerse buenos resultados con el sistema propuesto recomendamos que se reproduzca la placa base de manera profesional. También recomendamos mejorar el Sistema Operativo para hacerlo de Tiempo Real así como investigar en la utilización de la interfaz para cámara presente en la MBTI2440 sin uso actualmente.

REFERENCIAS BIBLIOGRÁFICAS

Abusleme H., Angel C. (2000). Control difuso de vehículo volador no tripulado. Tesis de maestría. Pontificia Universidad Católica de Chile, Escuela de Ingeniería. Santiago de Chile, Chile.

Adiprawita, W.; Ahmad, A. S.; Sembiring J. (2007). Automated flight test and system identification for rotary wing small aerial platform using frequency responses analysis. Technical report. ICIUS. Bali, Indonesia.

AlSwailem, Salah I. (2004). Application of robust control in unmanned vehicle flight control system design. Tesis doctoral. Universidad de Cranfield. Colegio de Aeronautica. Cranfield, Inglaterra.

Andersen, A. T.; Bak, P.; Hansen M. A.; Jensen R.; Airplane A. M.; Sørensen M. H.; Xie J.; Helbo J. (2004). Autonomous model airplane. Proyecto de grupo. Institute of Elektronic Systems, Aalborg University. Dinamarca.

Arias-Montaño, Alfredo (2007). Diseño de perfiles aerodinámicos para aplicaciones a bajos números de reynolds mediante el uso de algoritmos evolutivos. In: *8vo Congreso Iberoamericano de Inegiería Mecánica*. Federación Iberoamericana de Inegiería Mecánica. Cusco, Perú. p. 10.

Asensio, José Luis; Pérez, Fernando; Morán Paola (2008). Uavs beneficios y límites. *I.A.S.S.* p. 5.

Barrientos, A.; del Cerro, J.; Gutiérrez P.; San Martín R.; Martínez A.; Rossi C. (2007). Vehículos aéreos no tripulados para uso civil. tecnología y aplicaciones. In: *I Workshop Español de Robótica, ROBOT 2007*. CEDI. Zaragoza, España. p. 29.

Cañizares, Julio R. (2010). Modelado y control del vehículo autónomo sumergible del $CIDNAV$. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Catsoulis, John (2005). *Designing embedded hardware*. 2nd ed.. O'Reilly Media.

Cordoba, Mario A. (2007). Attitude and heading refernce system i-ahrs for the efigenia autonomous unmanned aerial vehicles uav based on mems sensor and a neural network strategy for attitude estimation. In: *Control & Automation, 2007. MED'07. Mediterranean Conference on Control and Automatization*. Universidad de Cauca. IEEE. Atenas, Grecia. p. 8.

de Avila, Diamir; Hernández-Santana, Luis; Martínez-Jiménez Boris L. (2010). Development of control system for autopilot of small airplane. *DIALNET, I Jornadas científicas UAH-CES de Cuba* pp. 25–29.

De Ávila, Diamir (2008). Estrategias de control basado en modelo para minihelicóptero no tripulado.. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Fossen, T. I. (2011). Mathematical models for control of aircraft and satellites.

González-Vázquez, Fidel; Herrera, Moisés; López Erick; González René; González Luisa (2010). Procesamiento intensivo de datos en aplicaciones empotradas con restricciones de tiempo real. *RCCI, Universidad de Ciencias Informáticas* **4**(3-4), 11.

Graversen, T.; Krogh, K.; Chávez A.; Rui-Pérez R; Vedstesen S. V. (2001). Autonomous aircraft. Technical report. Institute of Elektronic Systems - Departament of Control Engineering, Aalborg University. Dinamarca.

Guerra, Carlos E. (2010). Diseño e implementación de hardware y software de bajo nivel para vehículo submarino autónomo. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

López, Alvaro (2008). I+d+i en sistemas uav: Aplicaciones en entornos aeroportuarios. In: *TECNIBERRIA*. Asociación Española de Empresas de Ingeniería, Consultoría y Servicios Tecnológicos. Madrid, España. p. 23.

Martínez, A. S. (2005). Arquitectura de hardware para Vehículo Autónomo Aéreo.. Trabajo de maestría. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Villa Clara, Cuba.

Martínez-Carmenate, Miguel E. (2009). Desarrollo de sistemas de control para autopiloto de avión de pequeño porte. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Martínez-Jiménez, Boris L.; Pineda-Bombino, Luis M.;Martínez-Carmenate Miguel E.;De Ávila Diamir; Hernández Santana Luis (2012). Identificación de un vehículo aéreo no tripulado. *RIELAC* **33**(1815-5928), 44–45.

Mejias, Luis O. (2006). Control visual de un vehiculo aereo autonomo usando deteccion y seguimiento de caracterısticas en espacios exteriores. Tesis doctoral. Escuela Técnica Superior De Ingenieros Industriales. Universidad Politécnica de Madrid.. Madrid, España.

Merino, Luis. ;Ollero, Anibal; Caballero-Fernando (2006). Cooperación y percepción cooperativa en el sistema multi-uav. comets. In: *XXVI Jornadas de Automática.*. Almería, España.

Pajares, Gonzalo; Ruz, José Jaime; Lanillos Ppablo; Guijarro-María; de la Cruz Jesús Manuel; Santos Matilde (2008). Generación de trayectorias y toma de decisiones para uavs. *RIAII* **5**(1), 83–92.

Pineda, Luis M. (2008). Modelo matemático de un avión autónomo. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Pineda, Luis M. (2011). Control de vehículo aéreo autónomo basado en modelo dinámico. Master's thesis. UCLV. Dpto. de Automática y Sistemas Computacionales.

SAMSUNG ELECTRONIC, Co (2004). *S3C2410A 200MHz and 266MHz 32 Bit RISC Microprocessor*. 1 ed.. SAMSUNG ELECTRONIC Co., Ltd.. San24 Nongseo Ri, Giheung Eup, Yongin City, Gyeonggi Do, Korea.

Torroella, J. C. R. (2011). Control de vehículo aéreo autónomo basado en modelo dinámico. Tesis de maestría. University of Washington. Washington, EEUU.

Vélez, Carlos M.; Álvarez, J.A. (2003). Diseño, implementación y prueba de un sistema de control y navegación para un mini-helicóptero robot-colibrí concepción de la idea. In: *V Congreso de la Asociación Colombiana de Automática (ACA)*. Medellín, Colombia.

p. 6.

Woo, J.; Son, K.; Li T.; Kim G.; Kweon I. S. (2007). Vision-based uav navigation in mountain area. In: *IAPR Conference on Machine Vision Applications*. MVA. Japan. pp. 236 – 239.

Apéndice A
ANEXO 1: UNIDAD IMU MTI-G.

Kit de desarrollo.

- MTi GPS/INS (MTi-G-28A##G##).
- Antena GPS (ANT).
- Certificado individual de calibración.
- Carta con el código de licencia.
- Cables USB de datos y alimentación.
- MTi-Quick Setup and Manual.
- CDROM 3.1:
  - ◇ (PDF) Comunicación MTi de bajo nivel.
  - ◇ (PDF) MTi-G Quick Setup.
  - ◇ MTi SDK 3.1 setup.exe

Pasos para la instalación del *software*.

1. Ejecutar SDK 3.1 setup.exe en el modo "*Administrator*" o "*Power User*" con la licencia antes mencionada.

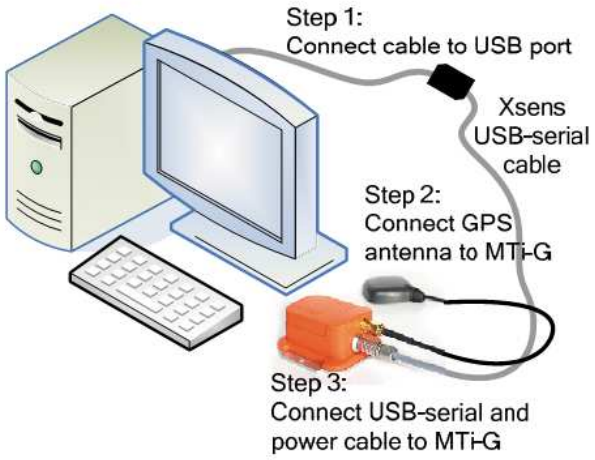
2. Instalar los drivers.

3. Instalar MATLAB runtime (MCR) que es necesario sólo para el *plugin* del campo magnético.

Pasos para la instalación del *hardware*.

1. Sin conectar el MTi-G, conectar el cable Xsens USB-serial a un puerto USB. El Xsens-serie requiere la instalación de dos drivers.

- Xsens USB-serial converter.



- Xsens Virtual COM port.

2. Sin conectar el MTi-G, conectar la antena GPS a MTi-G utilizando el conector SMA.

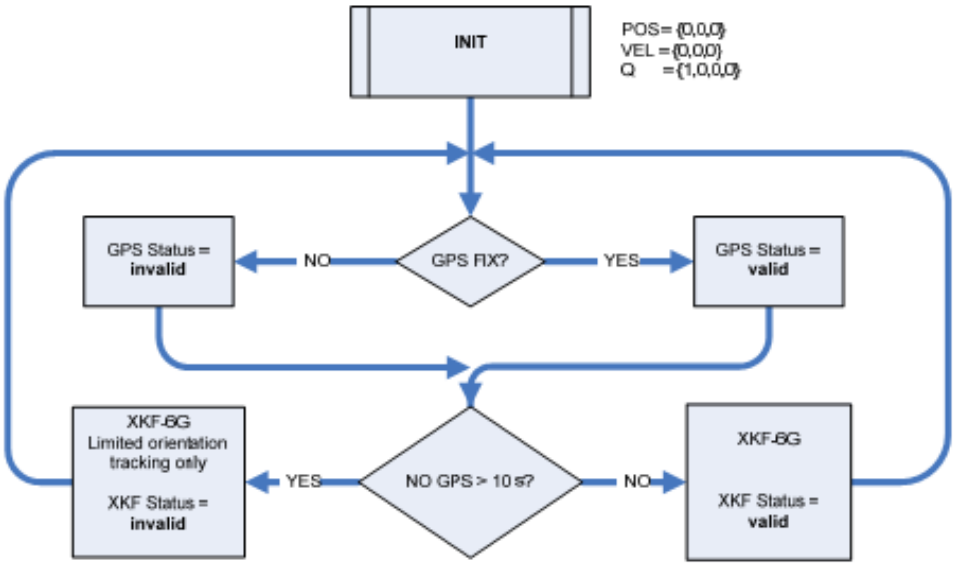
3. Conectar el cable de alimentación del MTi-G (usando el conector de aluminio de 7-pines).

El Filtro Xsens Kalman.

La orientación y posición de MTi-G se estiman utilizando el filtro extendido Kalman (referido a Xsens Kalman Filter *XFK-6G*). El filtro *XFK* se divide en dos pasos: predicción y corrección. En el paso de predicción, el sensor inercial se integra con el tiempo para obtener la posición y orientación estimadas. Debido a cada inexactitud pequeña en el giróscopos y acelerómetros, los valores estimados no serán perfectos y el error de estimación aumentará con el tiempo así como los valores estimados se alejan de los reales. En la corrección de *XFK* usando los datos *GPS* y la presión estática (barómetro), así como las señales o informaciones del magnetómetros o restricción no-holónomas, etc., dependiendo del escenario. Además, la estimación de la posición puede ser actualizada de forma rápida y una latencia corta, si se le compara con un sistema *GPS*. Esto significa que el *MTi-G* trabaja con mayor rapidez y menor desplazamiento que el sistema *GPS*.

Tratamiento de pérdida de la señal del *GPS*.

El *XFK-6G* está diseñado para operar con una buena corrección de *GPS*. No obstante, suelen ocurrir perdidas de señal del *GPS*. Para mejorar la repuesta del *XFK*, la orientación, la posición y la velocidad se estimarán basados en las señales de los sensores disponibles, excepto el *GPS*. Esta estimación perderá la calidad rápidamente, si la pérdida de la señal permanece por más de 10 segundos, el XFK-6G cambiará a un modo diseñado para rastrear solamente la orientación.



Apéndice B
ANEXO 2: CÓDIGO FUENTE.

Main

//

//Programa Principal para MBTI2440 como sistema de navegación para UAV //

//Autores: MSc. Alleiny Machado Sosa (GARP 2012)

// : Thuy Ninh Khac

//

```
#include <time.h>

#include <stdio.h>

#include <pthread.h>

#include <signal.h>

#include <stdlib.h>

#include <assert.h>

#include ¨ my_timer_int.c¨

typedef void*(*THREAD_CLLBK)(void *);

typedef void* THREAD;

typedef pthread_t THREADDATA,*PTHREADDATA;

THREAD THREAD_Start(THREAD_CLLBK callback,int prio,void* param)

int err;
```

```
pthread_t *thread_id= (pthread_t*)malloc(sizeof(pthread_t));

struct sched_param schd_p;

pthread_attr_t thread_attr;

assert(thread_id!=NULL);

if(thread_id!=NULL)

{

schd_p.sched_priority=prio;

pthread_attr_init(&thread_attr);

pthread_attr_setschedparam(&thread_attr,&schd_p);

pthread_attr_setschedpolicy(&thread_attr,SCHED_FIFO);

err=pthread_create(thread_id,&thread_attr,callback,param);

assert(err==0);

if(!err)

{

pthread_attr_destroy(&thread_attr);

return thread_id; }

}

pthread_attr_destroy(&thread_attr);

return NULL;

}

void* THREAD_Wait(THREAD thrd)

{

int err;
```

```
void*res;
pthread_t *thread_id=(pthread_t*)thrd;
assert(thread_id!=NULL);
err=pthread_join(*thread_id,&res);
assert(err==0);
free(thrd);
return (err==0)?res:NULL;
}
void* rf_modem_900(void* param)
{ //lectura y procesamiento de trama de rf_modem 900MHz
return NULL;
}
void* rf_modem_2400(void* param)
{
//lectura y procesamiento de trama de rf_modem 2.4GHz
return NULL;
}
int main()
{
spi_init(); //Inicialización de SPI
THREAD Hilo_rf900=THREAD_Start(rf_modem_900,90,NULL); //hilo para lectura de trama de RF_Modem a 900MHz
THREAD_Wait(Hilo_rf900);
```

THREAD Hilo_rf2400=THREAD_Start(rf_modem_2400,90,NULL); //hilo para lectura de trama de RF_Modem a 2.4GHz

THREAD_Wait(Hilo_rf2400);

printf(S̈istema Funcionando \n ̈); // para ver remotamente

while(1)

{

;

}

return 0;

}

Código fuente para Tarea de dsPIC30F4013

//

//Codigo fuente para controlador de Vuelo de UAV e interfaz SPI con MBTI2440 //

//Autores: MSc. Alleiny Machado Sosa (GARP 2012)

// : Thuy Ninh Khac

//

#include <p30f4013.h>

_FOSC(CSW_FSCM_OFF & XT_PLL16);

_FWDT(WDT_OFF & WDTPSA_8)

_FGS(CODE_PROT_OFF)

void Timer1_Init(void) // 20 ms

{ T1CONbits.TCKPS = 0x02; // divide por 64

IFS0bits.T1IF = 0; // limpio flag de timer1

```
PR1 = 0x0032; // temporización a 20 ms

IPC0bits.T1IP = 7;

IEC0bits.T1IE = 1; // interrupción habilitada

}

void Timer2_Init(void) //10 ms

{

T2CONbits.TCKPS = 0x03; // divide por 256

IFS0bits.T2IF = 0; // limpio flag de timer2

PR2 = 0x007D; // temporización a 10 ms

IPC1bits.T2IP = 1;

IEC0bits.T2IE = 1; // interrupción habilitada

}

void Spi_Init(void)

{

// a definir modo aun

}

void Pwm_Init(void)

{

// a definir

}

void set_pwm_servo(unsigned int porcientos) //0 - 100 % = 1ms a 2 ms

{

// a definir
```

```
}

void Analog_Init(void)

{

TRISB = 0x03f; //0x037 //RB0 y RB1 analog inputs , RB2-RB5 digital inputs y RB6-RB9 digital outs

cs1;

ldac = 1;

ADPCFGbits.PCFG0 = 0; //analog inputs

ADPCFGbits.PCFG1 = 0; //analog inputs

ADPCFGbits.PCFG2 = 1;

ADPCFGbits.PCFG3 = 1;

ADPCFGbits.PCFG4 = 1;

ADPCFGbits.PCFG5 = 1;

ADCON1bits.FORM = 0; // formato int

ADCON1bits.SSRC = 0; // trigger clear SAMP comienza conversion

ADCON1bits.ASAM = 0; // sample comienza con SAMP = 1;

ADCSSL = 0; // No scan

ADCON2 = 0;

ADCON3 = 0x0014; // Tad = 14 Tcy

ADCON1bits.ADON = 1; // Arranco ADC

}

void Serial_Init(void)

{
```

```
U1BRG = 0x10; // 0xD0 (9600) 0x10 (115200)

U1MODEbits.PDSEL = 0;

U1MODEbits.STSEL = 0;

U1STAbits.URXISEL = 0x01; //interrupt siempre que llega 1 dato (03h para 4 datos)

IEC0bits.U1RXIE = 1; //interrupción RX habilitada

IPC2bits.U1RXIP = 2;

U1MODEbits.ALTIO = 1; // pines alternativos

U1MODEbits.UARTEN = 1; //UATR1 habilitada

U1STAbits.UTXEN = 1; //TX habilitado

IFS0bits.U1TXIF = 0;

IFS0bits.U1RXIF = 0;

}

void Init_All(void)

{

RCONbits.SWDTEN = 0;

Spi_IniT();

Analog_Init();

Serial_Init();

Timer1_Init();

Timer2_Init();

}

void My_Delay_Us(int us)

{ int i;
```

```
T3CONbits.TCKPS = 0x01; // divide por 8 (0.25 us)

IFS0bits.T3IF = 0; // limpio flag de timer1

PR3 = 0x0004; // temporización a 1 us

T3CONbits.TON = 1; // T3 arrancado

for(i = 0; i ¡us; i++)

{

while(!IFS0bits.T3IF);

IFS0bits.T3IF = 0;

}

T3CONbits.TON = 0;

}

void My_Delay_Ms(int ms)

{

int i;

T3CONbits.TCKPS = 0x01; // divide por 8 (0.25 us)

IFS0bits.T3IF = 0; // limpio flag de timer1

PR3 = 0x0FA0; // temporización a 1 ms

T3CONbits.TON = 1; // T3 arrancado

for(i = 0; i ¡ms; i++)

{

while(!IFS0bits.T3IF);

IFS0bits.T3IF = 0;

}
```

```
T3CONbits.TON = 0;

}

unsigned int Analog_In(void)

{

ADCHS = 0; //canal 0

ADCON1bits.SAMP = 1; // comienzo sample

My_Delay_Us(5); // espero por Tad

ADCON1bits.SAMP = 0; // paro adquisición y comienzo conversion

while (!ADCON1bits.DONE); // espero por conversion terminada

return ADCBUF0;

}

void control_vuelo(void)

{

//aquí va control de vuelo

}

void _ISR _SPI1Interrupt( void )

{

//funcion de comunicación SPI (a definir paquete aun)

}

void _ISR _T1Interrupt( void ) // a 100 uS

{

INTCON1bits.NSTDIS = 1; // no nesting ints

IFS0bits.T1IF = 0; // limpio flag de timer1
```

```
control_vuelo(); // algoritmo de control de vuelo

INTCON1bits.NSTDIS = 0; // nesting ints enabled

}

void lee_trama_imu(void)

{

//lee trama y actualiza variables

//a definir aun

}

void _ISR _T2Interrupt( void )

{

//función de administración de energía, a definir aun

}

int main(void)

{

Init_All();

while(1)

{

lee_trama_imu();

}

return 0;

}
```

Servicio de la tarjeta MBTI2440

```
//////////////////////////////////////////////////////////////////////////////
```

//Algoritmo de navegacion Inercial basada en modelo para UAV *//*

//Autores: MSc. Alleiny Machado Sosa (GARP 2012) //

// : Thuy Ninh Khac //

//

```
#include <stdio.h>
#include <stdlib.h>
void navegacion(void)
{
//aquí va el algoritmo de navegación a desarrollar
}
```

Inicialización del timer para interrupción de tiempo

//

//Codigo de inicialización del timer para interrupción de tiempo //

//Autores: MSc. Alleiny Machado Sosa (GARP 2012) //

// : Thuy Ninh Khac //

//

```
#include <time.h>
#include <stdio.h>
#include <signal.h>
#include <pthread.h>
#include <unistd.h>
#include ¨ my_navegación.c ¨
#include ¨ my_spi.c¨
```

```
int estado=0;

int timer_interrupt_init(void); inicializa timer

void time_isr(union sigval val); rutina de control

void timer_isr(union sigval val)

{

navegación(); // llama a algoritmo de navegación

spi_io_pack();

}

int timer_interrupt_init(void)

{ int Ret;

pthread_attr_t attr;

pthread_attr_init( &attr );

struct sched_param parm;

parm.sched_priority = 255;

pthread_attr_setschedparam(&attr, &parm);

struct sigevent sig;

sig.sigev_notify = SIGEV_THREAD;

sig.sigev_notify_function = timer_isr;

sig.sigev_value.sival_int = 1;

sig.sigev_notify_attributes = &attr;

timer_t timerid; //crea timer nuevo.

Ret = timer_create(CLOCK_REALTIME, &sig, &timerid);

if (Ret == 0)
```

```
{
struct itimerspec in, out;
in.it_value.tv_sec = 0;
in.it_value.tv_nsec = 1;
in.it_interval.tv_sec = 0;
in.it_interval.tv_nsec = 10000000; // 10 ms
Ret = timer_settime(timerid, 0, &in, &out);
if(Ret == 1)
{
timer_delete(timerid);
}
}
return Ret;
}
```

Comunicación entre MBTI2440 con dsPIC30F4013 vía *SPI*

```
////////////////////////////////////////////////////////////////////////////
//Codigo fuente de interfaz entre MBTI2440 con dsPIC30F4013 via SPI //
//Autores: MSc. Alleiny Machado Sosa (GARP 2012) //
// Thuy Ninh Khac //
////////////////////////////////////////////////////////////////////////////
#include <time.h>
#include <stdint.h>
#include <unistd.h>
```

```
#include <stdio.h>

#include <stdlib.h>

#include <getopt.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <linux/types.h>

#include <linux/spi/spidev.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof((a)[0]))

//////////////////////

// Variables Globales//

//////////////////////

FILE *fp;

FILE *fp0;

int ret = 0;

int fd;

static const char *device = /dev/spidev0.0”;

static uint8_t mode;

static uint8_t bits = 8;

static uint32_t speed = 1000000; //1MHz

static uint16_t delay;

uint8_t tx[] = 0, 0, ; //definir tamaño del buffer necesario

uint8_t rx[ARRAY_SIZE(tx)] = {0, };

struct spi_ioc_transfer tr;
```

```
//////////////////////////
// Prototipo de funciones //
//////////////////////////
void spi_init(void);
void spi_io_pack(void);
////////////////////////////////
// Implementacion de funciones //
////////////////////////////////
void spi_init(void)
{
fd = open(device, O_RDWR);
if (fd ¡0)
{
printf("no se abrió puerto");
exit(1); //no se abrió puerto
}
ret = ioctl(fd, SPI_IOC_RD_MODE, &mode); //modo de lectura SPI
if (ret == -1)
{
printf("no se pudo poner modo de lectura");
exit(1); //no se pudo poner modo de lectura
}
ret = ioctl(fd, SPI_IOC_WR_MODE, &mode); //modo de escritura SPI
```

```
if (ret == -1)
{
printf("no se pudo poner modo de escritura");
exit(1); //no se pudo poner modo de escritura
}
ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits); //bit por words de escritura
if (ret == -1)
{
printf("no se pudo poner bit por palabra de escritura");
exit(1); //no se pudo poner bit por palabra de escritura
}
ret = ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits); //bit por words de lectura
if (ret == -1)
{
printf("no se pudo poner bit por palabra de lectura");
exit(1); //no se pudo poner bit por palabra de lectura
}
ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed); //velocidad de transmisión
if (ret == -1)
{
printf("no se pudo poner maxima velocidad de transmisión");
exit(1); //no se pudo poner maxima velocidad de transmisión
}
```

```
ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);

if (ret == -1)

{

exit(1); //no se pudo poner maxima velocidad de recepción

}

ret = 0;

}

void spi_io_pack(void)

{

tr.tx_buf = (unsigned long)tx;

tr.rx_buf = (unsigned long)rx;

tr.len = ARRAY_SIZE(tx);

tr.delay_usecs = delay;

tr.speed_hz = speed;

tr.bits_per_word = bits;

// Definir paquete aún

// En el buffer tx[ ] lo que quiere transmitir

// Recibo en el buffer rx[ ]

ret = ioctl(fd, SPI_IOC_MESSAGE(tr.len), &tr); //transmisión por SPI (tr.len tamaño
del buffer)

}
```