VERIFICACIÓN Y VALIDACIÓN DE SISTEMAS DE CONTROL DE VUELO PARA MAV-VTOL BASADAS EN MATLAB STATEFLOW

Pablo Rodríguez[1], Jesús G. Villagómez[2], Manuel Vargas[2], Francisco R. Rubio[2]
[1]*athlonm@gmail.com* , [2]{*villagomez, mvargas, rubio*}*@us.es*
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla

Resumen

Se presenta el desarrollo de una solución para la verificación y validación de sistemas de gobierno y control de vuelo para pequeñas plataformas de tipo VTOL (Vertical Take-Off and Landing). Para la simulación del vehículo se dispone del correspondiente modelo dinámico, que será interconectado con diferentes subsistemas de control de estabilización y posición, gobernados por una máquina de estados definida en MATLAB Stateflow. El conjunto de simulaciones para la verificación del diseño de la solución propuesta son llevadas a cabo mediante Simulink. El objetivo es disponer de mecanismos que permitan evaluar el correcto funcionamiento de prototipos antes de ser evaluados en condiciones reales.

Palabras clave: Arduino, Docencia, MATLAB Stateflow®, *Quadrotor*, Simulación.

1. INTRODUCCIÓN

El uso de plataformas MAV-VTOL (Miniature Autonomous Vehicle - Vertical Take Off and Landing) [5] ha experimentado un notable crecimiento en los últimos tiempos. Bien para investigación como para uso doméstico, este tipo de vehículos son ideales principalmente gracias a su bajo coste y al amplio conjunto de aplicaciones en las que permite ser utilizado: desde vigilancia aérea hasta transporte de carga. Su uso está cada vez más extendido e incluso normativas de tipo legal están siendo desarrolladas para poder legislar su aplicación futura en entornos urbanos [1]. En el ámbito de la Academia, existen numerosos trabajos que combinan el estudio de controladores para la estabilización y estimación del estado [4][15], bien en interiores o exteriores, gracias a las posibilidades de expansión sensorial que permiten estos vehículos. La inclusión de sensores exteroceptivos (cámaras, sensores de tipo láser, etc.) al sistema no es más que una evolución del mismo que permite posibilidades de navegación autónoma, incluso en entornos no estructurados. Trabajos recientes ponen de manifiesto la capacidad de ampliación funcional e interacción con el entorno que permiten estas plataformas, mediante la inclusión de pequeños brazos manipuladores [7], actuadores de cámara giro-estabilizados [11] o sensores de presión y contacto para diferentes tipos de aplicaciones [13].

Sin embargo, la integración de los diferentes subsistemas que conforman estas plataformas y su configuración y adaptación no es trivial. Como plataforma aérea robótica, es imprescindible que la combinación de software y hardware embarcado funcione según lo establecido, siendo robusto a fallos de tipo sensorial o bien de pérdida de comunicación con el enlace de tierra, entre otros. En caso contrario, cualquier excepción no contemplada en el desarrollo del software, derivará en un fallo que provocaría grandes desperfectos en la plataforma y riesgos para terceros, con el consiguiente coste económico tanto de materiales como de horas de trabajo. Por ello, la evaluación del software embarcado antes de su uso es imprescindible para poder detectar fallos de codificación o situaciones no contempladas.

Las soluciones actuales, económicamente viables, que combinan hardware y software más extendidas están basadas en entornos de microprogramación en lenguage C/C++ y hardware basado en Arduino, conocidas como Arducopter [2]. Periódicamente la Comunidad aporta nuevas actualizaciones del firmware del dispositivo e incluye nuevas mejoras y funcionalidades. A nivel de investigación, lo que se pretende es poder reutilizar parte de estas aportaciones para poder avanzar en campos como el control o la fusión sensorial, pudiendo acceder a cualquier elemento hw/sw del sistema y poder reconfigurarlo, si fuera necesario. Si bien es una plataforma muy extendida, no es posible una comprobación por simulación del sistema, de tal manera que ante cualquier modificación del código fuente, o bien una reformulación de la lógica de aplicación, la única depuración y verificación posibles, incluyendo el comportamiento de los controladores, es mediante el despegue de la plataforma y comprobando que la evolución de la misma no deriva en situaciones de inestabilidad. Para entornos como investigación y docencia, son situaciones no admisibles.

ISBN-13: 978-84-697-0589-6

Se hace por tanto imprescindible el desarrollo de entornos que permitan una simulación completa del sistema para su verificación, control de supervisión, planificación de tareas, gestión de fallos y posterior validación. Es lo que se conoce típicamente como *"Software-in-the-loop"*: como máquina de estados que permite probar el comportamiento del sistema ante cualquier entrada o evento externo, comprobando que la salida se corresponde con lo previamente diseñado. Existen otras técnicas más avanzadas de verificación y validación de modelos software: son las denominadas técnicas basadas en modelo o *"Model checking"* [12][14]. Estas técnicas están basadas en lógicas de tipo modal y descriptiva para la representación del conocimiento y automatización del comportamiento. Son técnicas que permiten una evaluación exhaustiva del sistema. Sin embargo, gráficamente no es posible ver su evolución y a nivel académico son difíciles de implantar a corto o medio plazo.

La solución propuesta en este trabajo, para diseñar e implementar una máquina de control de estados e integrarla en una plataforma de tipo quadrotor, está basada en la combinación de MATLAB Stateflow® y el software de simulación MATLAB Simulink®, muy extendido a nivel académico [16]. Stateflow es un entorno para el modelado y simulación de lógica de decisión combinatoria y secuencial basado en máquinas de estado y diagramas de flujo. Permite la representación de diagramas de transición de estado con el fin de modelar la evolución del sistema ante eventos internos, externos o condiciones basadas en tiempo.

El desarrollo del presente trabajo queda estructurado como sigue: en la sección 2 se presenta brevemente una descripción del modelo dinámico de la plataforma utilizada para la experimentación, de tipo *quadrotor*. Las secciones 3 y 4 presentan en detalle la solución propuesta y diferentes alternativas para la representación de las salidas del sistema, respectivamente. La sección 5 explora la capacidad de integración del diseño propuesto en hardware real y las principales conclusiones y propuestas de trabajo futuro son presentadas en la sección 6.

2. MODELADO DINÁMICO

Uno de los puntos clave de la solución propuesta es la modularidad que presenta. Es posible reutilizar la misma máquina de estados para el gobierno de cualquier plataforma de tipo VTOL, intercambiando el modelo dinámico del vehículo bajo estudio y los controladores de estabilización y posición correspondientes. Así, en esta propuesta, se ha utilizado el modelo dinámico correspondiente a un vehículo denominado *quadrotor*, con 4 motores coplanarios.

Desde el punto de vista dinámico, un *quadrotor* consiste en una masa puntual suspendida en el aire sobre la cual se aplican diferentes fuerzas y momentos, combinando tan sólo las velocidades de los diferentes rotores [4][5]. Para generar un movimiento de balanceo o roll (ángulo ϕ), se produce un desequilibrio entre las fuerzas f_2 y f_4 (Figura 1). Análogamente, para un movimiento de cabeceo o pitch (ángulo θ), el desequilibrio se realizará entre las fuerzas f_1 y f_3. El movimiento de guiñada, o yaw (ámgulo ψ), es resultado del desequilibrio entre los pares de fuerzas (f_1,f_3) y (f_2,f_4) producidas por pares de motores que giran en sentidos opuestos. Como consecuencia de magnitudes positivas en los ángulos de balanceo y cabeceo, se producen movimientos de tipo lateral y longitudinal, respectivamente, consiguiendo así desplazamientos traslacionales. Desde el punto de vista del número de entradas manipulables y grados de libertad, el sistema se considera subactuado, de manera que tan sólo se dispone de 4 señales de actuación (motores $f_{1\ldots4}$) para controlar los 6 grados de libertad (orientación y posición) propios de un sistema aéreo. El empuje total proporcionado por la rotación de los motores se obtendrá como la suma de las fuerzas que ejercen los mismos.

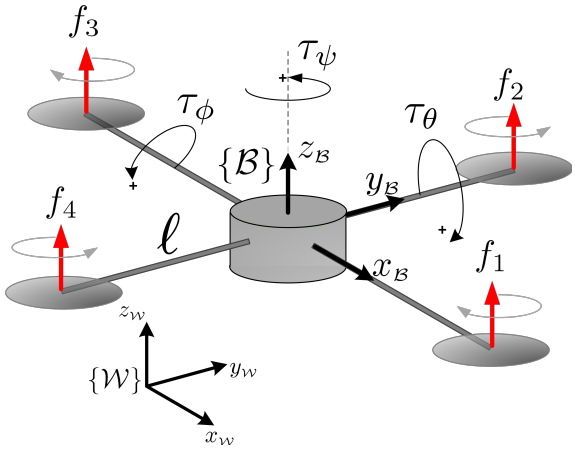

Figura 1: Representación de fuerzas y pares resultantes sobre la estructura de tipo *quadrotor*.

Bajo ciertas consideraciones [15], es posible plantear las ecuaciones que rigen la dinámica en dos subsistemas interconectados: traslación (ecuación 1) y rotación (ecuación 2):

$$m\ddot{\boldsymbol{\xi}} + mg\boldsymbol{e_3} = \boldsymbol{f_\xi} \tag{1}$$

$$\mathbf{M}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \mathbf{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} = \boldsymbol{\tau_\eta} \tag{2}$$

Los principales sistemas de coordenadas relevantes en el modelado son: sistema de referencia inercial,

ISBN-13: 978-84-697-0589-6

supuesto fijo sobre la superficie de la Tierra, $\{\mathcal{W}\}$ (*World coordinate frame*), y el sistema $\{\mathcal{B}\}$ (*Body coordinate frame*), supuesto solidario al movimiento del vehículo. Se tiene que ${}^{\mathcal{W}}\boldsymbol{\xi}_{\mathcal{B}} = [\text{x y z}]^\top \in \mathbb{R}^3$ representa la posición del centro de rotación del vehículo, expresado en el marco de referencia inercial. ${}^{\mathcal{W}}\boldsymbol{\eta}_{\mathcal{B}} = [\phi\ \theta\ \psi]^\top \in \mathbb{R}^3$ son los denominados ángulos de Tait-Bryan; $\boldsymbol{f_\xi} \in \mathbb{R}^3$ son los fuerzas aplicadas en cada eje; ℓ es la distancia entre el origen del sistema $\mathcal{B}$ y el eje de rotación de cada motor (se asume estructura simétrica); $\boldsymbol{\tau_\eta} \in \mathbb{R}^3$ son los pares generados como consecuencia del desequilibrio de fuerzas; m es la masa del helicóptero; g es el módulo de la aceleración de la gravedad; $\mathbf{M}(\boldsymbol{\eta}) \in \mathbb{R}^{3\times3}$ es la matriz de inercia; $\mathbf{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) \in \mathbb{R}^{3\times3}$ es la matriz de Coriolis y $\boldsymbol{e_3} = [0\ 0\ 1]^\top \in \mathbb{R}^3$.

3. DESCRIPCIÓN DE LA SOLUCIÓN

Siguiendo una descripción de tipo *top-bottom*, se dispone de tres elementos principales interconectados que componen la solución propuesta: máquina de estados, o diagrama de Stateflow; bloques de control e implementación del modelo dinámico de la plataforma (Figura 2).

3.1. DIAGRAMA STATEFLOW

El diagrama de Stateflow permite definir lógica de decisión compleja en el control de la plataforma de forma gráfica y estructurada, con el fin de simular la respuesta del sistema a entradas externas, eventos y condiciones temporales. En esta propuesta, existen dos elementos principales para el diseño: estados y transiciones.

Los estados representan modos de funcionamiento del sistema, que se ejecutan de forma exclusiva o en paralelo. En éstos se establecen valores para variables y se incluyen funciones relativas a la semántica de Stateflow, recurriendo a bloques externos que incluyen funciones de MATLAB o diagramas de Simulink. Se organizan de forma jerárquica y pueden relacionarse entre sí mediante transiciones, que incluyen algún tipo de restricción o condición y permiten gestionar el paso de un estado a otro.

3.1.1. Descripción de los estados

Si bien la definición de los diferentes estados no es única, a continuación se expone el conjunto mínimo de estados necesarios para recoger la lógica básica relativa a la evolución de un VTOL, tanto en tierra como en el aire (Figura 3).

Tierra

Considerado como el estado inicial. Se establece para comprobar y calibrar los distintos componentes del *quadrotor*, tales como GPS, enlace RC, sónar, barómetro, IMU, sensor de flujo óptico. etc. Si se realiza correctamente, el sistema estará listo para realizar una transición ante el evento "Volar" y conmutar al estado "Aire".

Aire

El estado "Aire"se define desde que la plataforma está disponible para iniciar el vuelo. A su vez se descompone de los siguientes subsistemas:

- **Despegue** El despegue es la fase siguiente al estado "Tierra", y se encarga de proporcionar referencias en altura para un despegue estable y coordinado de la plataforma, por lo que además establece a cero las referencias de orientación. Establece las condiciones de inicio y monitoriza los bloques de control rotacional y el control de altura.

- **Vuelo** En el modo "Vuelo"se establecen cuatro modos que se diferencian en las referencias proporcionadas a la plataforma y, por tanto, en el tipo de trayectorias a marcar como referencia. Cada uno de estos estados incluye una función de Simulink donde se obtienen las referencias incluidas como entradas externas.

  Modo 0. Estabilización

  Este modo proporciona referencias en posición y velocidades angulares, con el objetivo de mantener estable la plataforma en torno a unos valores fijos de los ángulos que definen la orientación. Se establece por defecto y además, se debe pasar por él en cualquier transición entre los otros tres estados, como se comentará más adelante en la sección (Sección 3.1.2).

  Modo 1. Estabilización en altura

  Similar al modo anterior en cuanto a la estabilización de la plataforma, pero añadiendo como grado de libertad adicional controlado la altura respecto al suelo. Permite el movimiento en la dirección vertical proporcionando un valor de referencia z_{ref}, que puede ser una constante o una variable proporcionada por un generador de trayectorias que proporcione la referencia. Este bloque también puede ser aplicable a los dos siguientes modos.

  Modo 2. Movimiento en el plano

  En el modo 2 se mantiene constante la referencia de altura a la que se encuentra en

ISBN-13: 978-84-697-0589-6

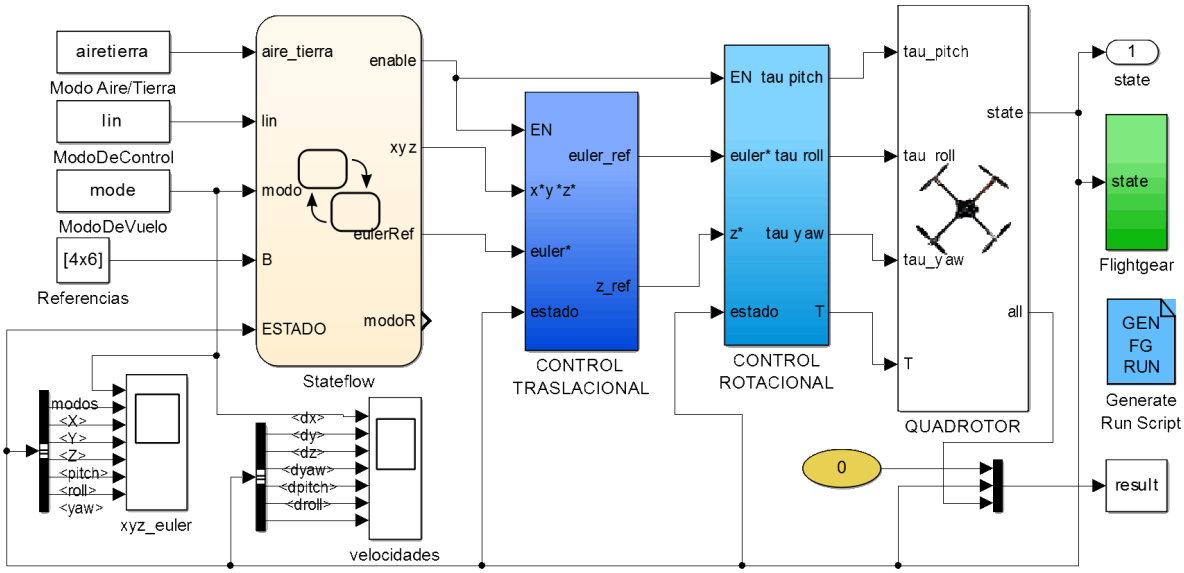

Figura 2: Modelo de la solución. Se representan bloque de Stateflow (izq); bloques de control (centro); y representación del modelo dinámico de la plataforma (dcha.)

ese instante y se proporciona una referencia en posición en el plano XY, bien constante o proporcionada por el generador de trayectorias. En este caso dicho sistema definiría una serie de *waypoints* que conformarían la trayectoria deseada. Este bloque permite incluir algoritmos de planificación de caminos que expanden las utilidades de la plataforma al aumentar su nivel de autonomía.

Modo 3. Movimiento en XYZ

En el último modo son proporcionadas referencias para los 6 grados de libertad de la plataforma, permitiendo un movimiento libre en el espacio tridimensional. Conserva las características del modo 2 pero añade la posibilidad de variar referencias en altura. Monitoriza y gestiona tanto los bloques de control rotacional como del control traslacional.

3.1.2. Transiciones

- **Tierra - Aire**

  Esta transición verifica la comunicación con la estación base, estado de sensores, voltaje de baterías y enlace RC con la emisora, confirmando así que todos los componentes están operativos y la plataforma está lista para volar de forma segura.

- **Despegue - Vuelo**

  Tiene como objetivo confirmar que el *quadrotor* ha despegado y se encuentra estable, comprobando que la altura es la de referencia establecida para este modo, y así poder iniciar el estado Vuelo.

- **Transición entre modos de vuelo**

  El modo de vuelo es una variable externa introducida por el usuario y puede tomar valores consecutivos del 0 al 3. Aunque los cuatro modos de vuelo son independientes y tienen su propia lógica, para cambiar entre sí los modos "1", "2" y "3" es necesario pasar primero por el estado "0" de "Estabilización", con el fin de garantizar una transición segura en tiempo finito. Una vez verificada la estabilidad, se proceden a comprobar en cada caso los siguientes parámetros mediante una función denominada "*ok*", que realiza las siguientes actuaciones según el tipo de control seleccionado:

  1. Control en altura. Se debe comprobar la veracidad de los valores del sónar, así como el margen entre la altura de referencia y la altura actual, para comprobar que es un movimiento realizable y seguro.
  2. Control en posición. Deben verificarse la cobertura GPS, así como los valores de referencia deseados y elaborar unos valores de transición entre el estado y la referencia para suavizar la trayectoria, en tiempo finito.

  El tipo de control en cada caso es el siguiente:

  - Transición $0 \rightarrow 1$: Control en altura.
  - Transición $0 \rightarrow 2$: Control en posición.
  - Transición $0 \rightarrow 3$: Control en altura y en posición.

ISBN-13: 978-84-697-0589-6

3.2. Modelo dinámico

El bloque denominado *Quadrotor* en la figura (Figura 2) recoge la implementación del modelo dinámico de la plataforma presentada en la sección (Sección 2). La entrada al sistema, proporcionada por el bloque de control, se corresponderá con los pares de actuación ($\boldsymbol{\tau}_a$):

$$\boldsymbol{\tau}_a = \begin{bmatrix} \tau_{\phi_a} \\ \tau_{\theta_a} \\ \tau_{\psi_a} \end{bmatrix} = \begin{bmatrix} \ell f_2 - \ell f_4 \\ \ell f_3 - \ell f_1 \\ \kappa((f_1 + f_3) - (f_2 + f_4)) \end{bmatrix} \tag{3}$$

y el empuje total deseado, calculado como:

$$\mathrm{T} = \sum f_i, \quad i = 1 \ldots 4 \tag{4}$$

donde κ recoge diferentes parámetros dependientes del motor y rotor seleccionados. Estas entradas son direccionadas al bloque denominado "*Control Mixer*", encargado de convertir estos pares y empuje total en velocidades de giro de los rotores. La aplicación de éstos provocará la evolución del vector de estados representado como:

$$\mathbf{x} = \begin{bmatrix} \mathrm{x} & \mathrm{y} & \mathrm{z} & \phi & \theta & \psi & \dot{\mathrm{x}} & \dot{\mathrm{y}} & \dot{\mathrm{z}} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix} \tag{5}$$

Adicionalmente, pueden añadirse modelos de perturbaciones externas, tales como viento o efecto suelo, mediante bloques propios que incluyan el modelado de estos efectos, o los propios del Aerospace Toolbox.

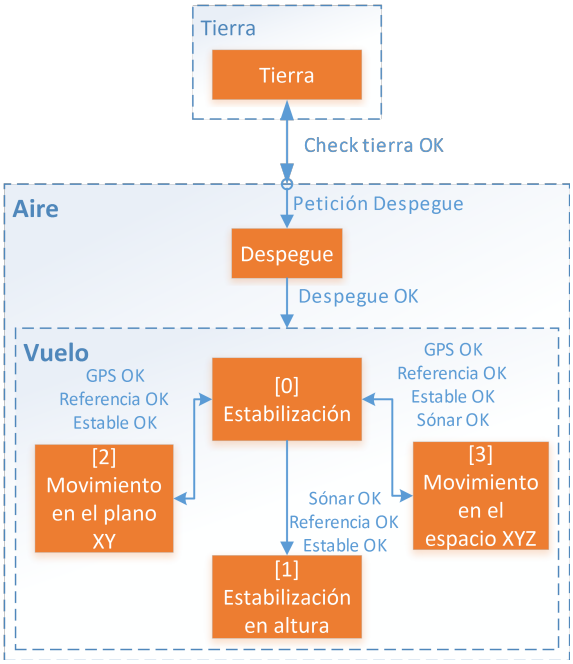

Figura 3: Representación del diagrama principal de estados Stateflow y transiciones.

3.3. CONTROL

El conjunto de bloques relativos al control están gobernados por la máquina de estados y proporcionan las señales de actuación a la plataforma para lograr las referencias deseadas. Si bien es un elemento clave en el diseño de la solución global, para un primer desarrollo se ha optado por reutilizar controladores para plataformas de tipo VTOL disponibles en [6]. Consisten básicamente en controladores lineales en posición y velocidad (lineales y angulares), tanto para estabilización como para traslación. Para un mayor detalle sobre la implementación de las leyes de control, se refiere al lector a la referencia indicada.

Típicamente, debido a la dinámica de los subsistemas que conforman el modelo de este tipo de plataformas (Sección 2), la tarea del control se descompone en control rotacional y control longitudinal [4][5][15] (alternativamente de estabilización y traslación). Bajo estas premisas, el modelo presentado en la figura (Figura 2) presenta esta descomposición, lo que permite implementar una lógica de gobierno de los diferentes bloques de control:

- En función del modo de vuelo actual del sistema, la máquina de estados habilitará o no (señal ENABLED) diferentes subsistemas. Por ejemplo, si el modo de vuelo establecido es "Estabilización", el subsistema de control traslacional será deshabilitado hasta nueva conmutación.
- La interfaz que presentan los bloques de control, tanto al diagrama de estados como al modelo dinámico, es similar, independientemente de las leyes que estén ejecutando (lineales, no lineales, por ejemplo). Esto permite la adición de nuevos bloques de control, sin modificar el diseño global (tan sólo el subsistema al que afecte), para ser evaluados y verificados. Este detalle se puede observar en la figura (Figura 4). Pueden añadirse tantos modos de control como se quiera, teniéndolos en cuenta en la lógica de Stateflow.
- Uno de los objetivos del diseño inicial era poder permitir una conmutación en "caliente" entre las diferentes leyes de control. Tanto el usuario como el sistema, pueden, en cualquier momento y si el sistema lo cree conveniente, poder conmutar entre leyes de estabilización o traslación.

El bloque de control traslacional (Figuras 2,4) toma como referencia la posición (${}^{\mathcal{W}}\boldsymbol{\xi}_{\mathcal{B}}{}^{ref}$) y proporciona las referencias al control rotacional. Los

ISBN-13: 978-84-697-0589-6

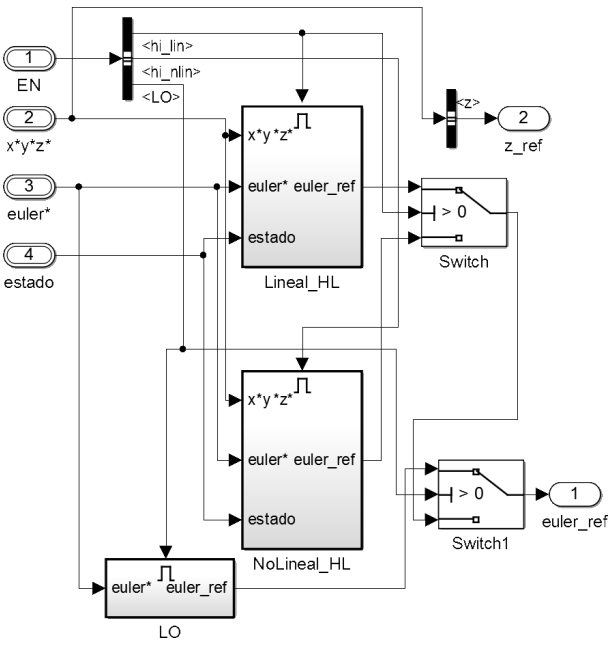

Figura 4: Subsistema de control traslacional. Se observa el detalle de la señal ENABLE para poder simular la habilitación de diferentes leyes de control en "caliente".

modos de vuelo que utilizan este tipo de control son el de "Movimiento en el plano" y "Movimiento en XYZ".

En el caso del control rotacional, se tiene como entrada la posición angular de referencia (proporcionada por el bloque de control traslacional o por el propio modo de vuelo en el bloque de Stateflow) y se obtienen como salida los pares correspondientes a cada ángulo para realizar el movimiento.

4. SIMULACIÓN

El objetivo principal de este trabajo era desarrollar una solución para poder simular los diferentes estados lógicos por los que puede pasar el *quadrotor*, de manera segura. Los resultados de simulación permiten evaluar diferentes aspectos de la solución propuesta: evaluación de las leyes de control, comportamiento ante perturbaciones, conmutación entre diferentes estados, etc.

4.1. RESULTADOS INICIALES

El hecho de utilizar Simulink permite un análisis inmediato de los resultados de simulación mediante gráficas. A continuación se muestran algunos de los resultados más interesantes. En la figura (Figura 5), puede observarse la evolución temporal de los valores correspondientes de posición lineal y posición angular, y ser comparados con el modo de vuelo. La gráfica se ha obtenido a partir de una simulación de $t_{sim} = 40s$, donde:

Modo 0: Intervalo de simulación: [0-10]s. Modo de vuelo "Estabilización": Referencias en posición y orientación a 0: ${}^{w}\boldsymbol{\xi}_{\mathcal{B}}{}^{ref} = \mathbf{0}_{1x3}$ (m), ${}^{w}\boldsymbol{\eta}_{\mathcal{B}}{}^{ref} = \mathbf{0}_{1x3}$. El objetivo es mantener fijo el *quadrotor* en la posición final que alcanza después del despegue.

Modo 1: Intervalo de simulación: [10-20]s. Modo de vuelo "Estabilización en altura": Referencias en posición ${}^{w}\boldsymbol{\xi}_{\mathcal{B}}{}^{ref} = [0\ 0\ 8]^{\top}$ (m), y orientación ${}^{w}\boldsymbol{\eta}_{\mathcal{B}}{}^{ref} = [0\ 0\ 0]^{\top}$. Se proporciona una referencia fija en altura manteniendo la posición XY.

Modo 2: Intervalo de simulación: [20-30]s. Modo de vuelo "Movimiento en el plano XY": Las referencias en posición se establecen como ${}^{w}\boldsymbol{\xi}_{\mathcal{B}}{}^{ref} = [-1{,}5\ -1{,}5\ \ z]^{\top}$ (m). Se conserva la altura del intervalo anterior y se proporciona una referencia para que la plataforma vuele en el plano.

Modo 3: Intervalo de simulación: [30-40]s. Modo de vuelo "Movimiento en el espacio XYZ": Referencias en posición ${}^{w}\boldsymbol{\xi}_{\mathcal{B}}{}^{ref} = [1{,}5\ 1{,}5\ \ 4]^{\top}$ (m). La plataforma realiza una trayectoria 3D hacia un punto fijo de referencia.

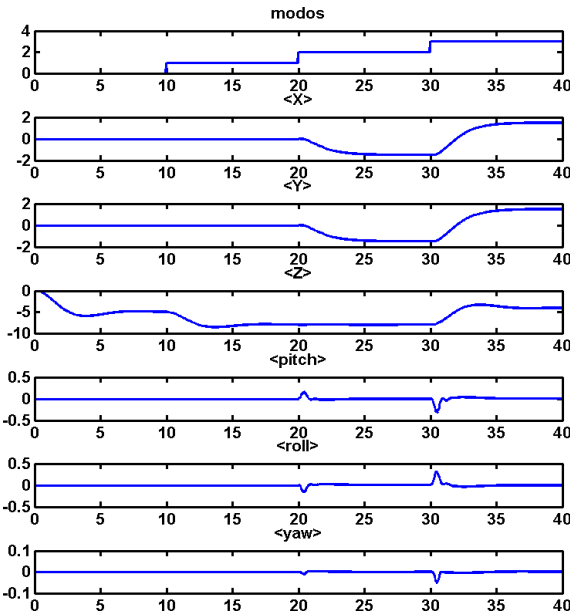

Figura 5: Evolución temporal de las variables "Modo", posición lineal y posición angular durante una simulación

4.2. FLIGHTGEAR

FlightGear [8] es un simulador de vuelo, de código abierto, que permite visualizar el vector de estado

ISBN-13: 978-84-697-0589-6

de la plataforma de una forma gráfica gracias a sus múltiples opciones de conexión con otro software. En este caso, se dispone del bloque "*FlightGear Preconfigured 6DoF Animation*" del Toolbox Aerospace Simulink, cuyas entradas son longitud, latitud, altitud, y posición angular (pitch, roll y yaw) (Figura 6(a)).

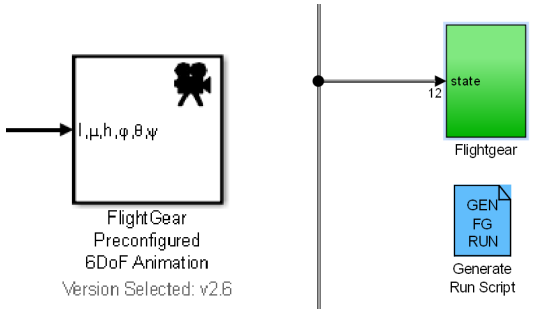

Figura 6: FlightGear (a) Diagrama de conexionado necesario para visualización de resultados en el simulador, y (b) Esquema de conexión con la salida del sistema.

FlightGear dispone de una gran variedad de modelos de aeronaves, tanto de ala fija como rotatoria, sin embargo no se ha encontrado un modelo nativo de *quadrotor*, por lo que se ha optado por disponer de un modelo de helicóptero que ilustra de forma similar la dinámica de éste (Figura 7).

Figura 7: Ejemplo de visualización de la evolución de la plataforma con FlightGear.

Para realizar la conexión de FlightGear y Simulink, es necesario colocar en el modelo un bloque denominado "*Generate Run Script*" (Figura 6(b)). Al abrir FlightGear y ejecutar este script, se sincronizan los datos de conexión por red y permite generar un archivo con extensión **.bat* que servirá de ejecutable para el FlightGear con las condiciones y parámetros iniciales que se hayan indicado en el script [9].

5. DESCRIPCIÓN HARDWARE

5.1. PLATAFORMA HW

La plataforma hardware utilizada durante el desarrollo de este trabajo está basada en la solución Arducopter [2]. Más concretamente, la versión utilizada es la denominada APM 2.5, la cual contiene en una misma placa PCB tanto el microprocesador como todos los sensores relativos a la IMU (Inertial Measurement Unit) o unidad de medida inercial, y externo a la placa proporciona la antena receptora de señal GPS modelo uBlox LEA-6H y el magnetómetro. Para medir la altura respecto al suelo se utiliza un sónar modelo MB1200.

Salvo pequeñas modificaciones, el software embebido proporciona el conjunto de medidas necesarias para la realimentación sensorial (${}^{\mathcal{W}}\boldsymbol{\eta}_{\mathcal{B}}$) y permite el cierre del lazo de control de estabilización a 100 Hz. Para la obtención de los datos necesarios para el cierre del lazo de posición (${}^{\mathcal{W}}\boldsymbol{\xi}_{\mathcal{B}}$), se utilizan en combinación el sensor GPS, sensor de flujo óptico y opcionalmente una cámara. En éste último, es necesario además la inclusión de una placa de hardware más potente que permita el procesamiento de información visual.

Los motores empleados son típicos *brushless*, modelo *T-Motor MT4006* de 740 RPM, los cuales, en función del número de celdas de la batería y la configuración del rotor, proporcionarán un empuje determinado. El control de estos motores se realiza mediante variadores de frecuencia (*ESC-Electronic Speed Controller*) de hasta 30 A, del fabricante *T-Motor*, que implementan un control de velocidad a bajo nivel y permiten envío de referencia desde la placa de control a 400 Hz.

A nivel de comunicaciones, se dispone de un enlace de telemetría de 433 Mhz del fabricante *3DRobotics*, el cual permite el envío periódico de las diferentes variables necesarias para una monitorización permanente. Adicionalmente, se dispone de una emisora para conmutar entre los diferentes modos de vuelo, implementados en la máquina de estados, y para proporcionar referencias en función del controlador activo en cada momento.

5.2. INTEGRACIÓN

Una vez validada la solución en Stateflow, se presenta como objetivo final y parte fundamental del proyecto su integración con el hardware real Arduino descrito anteriormente. Adicionalmente, se desea que el trabajo propuesto sirva tanto de herramienta para investigación como para docencia, especialmente en asignaturas relacionadas con el modelado de sistemas físicos y de Control automático.

ISBN-13: 978-84-697-0589-6

El hecho de que se haya probado en simulación previa a su implementación permite ahorrar tiempo de desarrollo, ya que las pruebas pueden hacerse en un ordenador, y no es necesaria la disponibilidad de la plataforma hasta el momento de la integración. Para ello, se considerará como principal herramienta el Toolbox Real-Time Workshop®, que genera y ejecuta código C/C++ exportado de Simulink, Stateflow y las funciones del diseño, incluso para aplicaciones en tiempo real. Para programar la placa de control, existen dos opciones: bien desde el propio entorno de MATLAB, o bien importando el código anteriormente generado en el *workspace* del IDE genérico de Arduino [3] e integrando las diferentes funciones exportadas en el ejecutable principal.

Es importante destacar que esta propuesta es compatible con otras plataformas VTOL. El procedimiento a seguir, para utilizar la propuesta actual en hardware de control diferente al planteado y otro modelo dinámico sería:

- Sustitución del modelo dinámico correspondiente a otro tipo de plataforma VTOL.
- Configurar adecuadamente las opciones de exportación de código embebido en el Toolbox Real-Time Workshop® para la placa de control destino específica.

De manera opcional, es posible además verificar el correcto funcionamiento del hardware destino del código exportado. Es lo que se conoce como *"Hardware-in-the-loop"*, y para el caso específico de la plataforma Arducopter, existen librerías (bloques "*S-Function*") fácilmente importables al entorno de desarrollo [10].

6. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha presentado el diseño de una máquina basada en MATLAB Stateflow para la verificación y validación del software de control para un equipo robótico aéreo y su integración con plataformas embebidas de tiempo real. Este trabajo es una muestra de las múltiples opciones que dicha herramienta permite a la hora de introducir lógica de control en una plataforma de tipo *quadrotor*, aunque podría extenderse a otras configuraciones. Se han puesto de manifiesto las ventajas de utilizar la simulación a la hora de desarrollar un proyecto de este tipo y la relativa simplicidad a la hora de añadir nuevos bloques que permitan aumentar las funciones del sistema, como por ejemplo, añadir distintas leyes de control y poder elegir en tiempo real cuál es más conveniente usar, o incluso añadir nuevas extensiones a la plataforma, como el control de un brazo manipulador externo, un giro-estabilizador para cámaras o cualquier otro bloque de control de carga, por ejemplo. Una vez finalizado el desarrollo del proyecto se desea poder promover la utilización de esta librería para entornos académicos bajo libre disposición.

Agradecimientos

Los autores desean expresar su agradecimiento al Ministerio de Economía y Competitividad (MINECO) por la financiación de este trabajo, a través de los proyectos DPI2012-37580-C02-02 y DPI2013-44135-R. Del mismo modo, expresar nuestra mayor gratitud a Antonio Ventosa Cutillas por su inestimable colaboración durante el desarrollo de este proyecto.

Referencias

[1] Asociación española de sistemas de vuelo pilotados de forma remota. http://www.aerpas.es/

[2] Plataforma Arducopter. http://copter.ardupilot.com/

[3] Plataforma Arduino. http://www.arduino.cc/

[4] Bouabdallah, S., Murrieri, P., y Siegwart, R., (2004) Design and Control of an Indoor Micro Quadrotor. En Proc. IEEE Int. Conf. on Rob. and Automat., volume 5, pp 4393–4398, New Orleans, USA.

[5] Castillo, P., Lozano R., y Dzul, A.E., (2005) Modelling and Control of Mini-Flying Machines, Springer.

[6] Corke, P., (2011) Robotics, Vision and Control. Fundamental Algorithms in MATLAB®, Springer. http://petercorke.com/Robotics_Toolbox.html

[7] Escareno, J., Rakotondrabe, M., Flores, G., y Lozano, R., (2013) Rotorcraft MAV having an onboard manipulator: longitudinal modeling and robust control, in European Control Conference (ECC). Zürich, Switzerland, pp 267-269.

[8] FlightGear flight simulator web page. Recurso electrónico. http://www.flightgear.org/

[9] Conexión Flightgear con Simulink. Recurso electrónico. http://www.mathworks.es/es/help/aeroblks/working-with-the-flight-simulator-interface.html#f3-19707

[10] Hartley, W., (2012) APM2 Simulink Blockset. http://www.mathworks.com/matlabcentral/fileexchange/39037-apm2-simulink-blockset.

[11] Lee, D., Chitrakaran, V., Burg, T., Dawson, D., y Xian, B., (2007) Control of a remotely operated quadrotor aerial vehicle and camera unit using a fly-the-camera perspective, in 46th IEEE Conference on Decision and Control, pp 6412-6417.

[12] Markosian, L.Z., Mansouri-Samani, M., Mehlitz, P.C., y Pressburger, T., (2007) Program Model Checking Using Design-for-Verification: NASA Flight Software Case Study, Aerospace Conference, 2007 IEEE , vol.1, no.9, pp 3-10.

[13] Parra-Vega, V., Sanchez, A., y Izaguirre, C., (2012) Toward force control of a quadrotor UAV in SE(3), on IEEE 51st Annual Conference Decision and Control (CDC), pp 1802-1809.

[14] PVS Specification and Verification System. http://pvs.csl.sri.com/

[15] Raffo, G.V., (2011) Robust Control Strategies for a quadrotor helicopter. An underactuated mechanical system. Ph.D.Thesis, Engineering School. University of Seville.

[16] Mathworks MATLAB Stateflow®. http://www.mathworks.es/products/stateflow/

ISBN-13: 978-84-697-0589-6