Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0830

Desarrollo de un sistema de control para vehículos aéreos no tripulados mediante comandos de voz en castellano

Ángel Rafael Rubio Fernández - Escuela de Ingenierías Industriales, Universidad de Extremadura
anrubiof@alumnos.unex.es

José Luis Herrero Agustín - Departamento de Sistemas Informáticos y Telemáticos, Universidad de Extremadura
jherrero@unex.es

Jesús Lozano Rogado - Departamento de Ingeniería Eléctrica, Electrónica y Automática, Universidad de Extremadura
jesuslozano@unex.es

Resumen

El uso de vehículos aéreos no tripulados se ha extendido de forma considerable en los últimos años tanto en el plano recreativo, científico como cinematográfico. Esta expansión ha motivado el desarrollo de nuevos modos de control que faciliten su manejo y lo adapten a cada situación. En este trabajo se presenta un sistema de control basado en el reconocimiento de voz. Las principales ventajas del diseño que se propone son: la utilización de herramientas de código libre, la capacidad de personalización, la elección del castellano como lengua de reconocimiento y la portabilidad.

Palabras clave: Reconocimiento de voz, UAV, *CMU Sphinx4,* MAVLink, *Raspberry Pi.*

1. INTRODUCCIÓN

Las prácticas con vehículos aéreos no tripulados o UAVs (*Unmanned Aerial Vehicles*) han aumentado de forma considerable en los últimos años, evolucionando de un uso exclusivamente militar a modelos de menor tamaño, coste y variedad de usos[1]. Esto ha supuesto una revolución en tanto que se ha producido un salto del espacio bidimensional de los vehículos no tripulados tradicionales al espacio tridimensional con los vehículos aéreos no tripulados.

El principal motor de esta evolución ha sido una sucesión de avances en diferentes campos de la ingeniería, como son el desarrollo de la fibra de carbono, la aparición de las baterías de polímero de litio, motores *brushless* y especialmente de los sensores de medición inercial y GPS[2].

Como consecuencia, diferentes sectores han buscado nuevas aplicaciones, desde el plano cinematográfico, con capacidad para realizar tomas a gran altura y sin necesidad de equipos más sofisticados[3], el reparto de paquetes (*e.g.* Amazon PrimeAir)[4], la medición de la calidad del aire[5], ingeniería civil[6], o el pilotaje acrobático[7], entre muchas otras aplicaciones.

Esta diversificación de usos ha dado lugar a dos nuevas necesidades: el desarrollo de una normativa[8] que regule las prácticas con estos vehículos y la aparición de nuevos y más intuitivos modos de control que faciliten su manejo y lo adapten a cada situación, como el uso de gestos o comandos de voz[9].

El control por voz, en lineas generales, supone un medio de control realmente interesante por dos razones principales[10]. En primer lugar, porque sus elementos de trabajo son conocidos por la mayoría de usuarios: comandos de voz. Palabras como *arriba*, *despegue* o *derecha* son conocidas por la totalidad de hablantes de un idioma, y por tanto fácilmente reconocibles y recordables. En segundo lugar, porque facilita el control de sistemas electrónicos en condiciones en que las manos estén ocupadas, o bien la movilidad o visión estén reducidas[11].

Derivado de esto se observa un hecho: el desarrollo de cualquier interfaz de interacción con el usuario suele incluir en una de sus fases avanzadas un sistema de reconocimiento de voz[12]. Esto se puede observar en la aparición de asistentes móviles como *Siri*, de *Apple Inc.*, que reduce significativamente el tiempo de navegación hasta activar la función deseada[13]; el control por voz en automóviles como el *Ford SYNC 3*[14], que reduce las distracciones en el conductor; o *Amazon Echo* junto con *Amazon Alexa*[15], un sistema que persigue el control de herramientas domóticas, música o llamadas con la voz como única herramienta de interacción.

Una vez se ha comprendido la evolución de los vehículos aéreos no tripulados y de los sistemas de reconocimiento de voz, parece lógico pensar en las múltiples aplicaciones que podría tener un sistema de reconocimiento de voz en el control de estos vehículos. Actualmente no existe un fuerte desarrollo en este aspecto, al menos en el ámbito comercial.

La mayoría de soluciones que se pueden encontrar poseen una característica en común: requieren de herramientas muy ligadas a modelos concretos de UAV o de interfaz de reconocimiento. Un ejemplo es el sistema de control por ternas de comandos *what3words*[16][17], el cual requiere de *Amazon Alexa*, y por tanto no sólo necesita una conexión a internet sino que además no posee un núcleo de reconocimiento modificable y necesario para diseños más específicos. Otras implementaciones conocidas se han realizado aprovechando los entornos de desarrollo para UAVs de marca DJI™[18], lo cual restringe los diseños a una sola marca, que aún siendo la punta de lanza en lo que a modelos comerciales respecta, está limitada a los modelos que esta fabrica, incompatibles con el resto de alternativas incluidas las de acceso libre.

Otra característica común de la mayoría de estos sistemas es la lengua de reconocimiento: el inglés. La popularización de los UAVs en los países hispanohablantes hace más que interesante la propuesta de un sistema de reconocimiento de voz en castellano.

La referencia más conocida en lo que respecta a un vehículo aéreo no tripulado controlado por comandos en castellano es el *Sky Rover Voice Command*[19], el cual, si bien cuenta con una cámara y un juego de comandos bastante completo, su reducido tamaño, calidad de la cámara, poca robustez y baja autonomía lo descartan de cualquier aplicación profesional.

Es tal vez en la práctica deportiva donde esta necesidad se hace mayor: donde el usuario pueda practicar su actividad mientras es grabado por un *UAV* sin ocupar sus manos, o en trabajos topográficos o agrícolas, donde el técnico pueda realizar el estudio o realizar una captura de la superficie barriendo un terreno sin necesidad de tener conocimientos de pilotaje. Se busca por último reducir al máximo el tamaño del dispositivo que reconozca los comandos, integrándolo en un dispositivo pequeño, ligero y con suficiente autonomía.

La motivación es, por tanto, alcanzar un diseño que cumpla las siguientes especificaciones:

1. La utilización de herramientas de código libre, que reduzcan el coste de la implementación final y la hagan adaptable a diferentes fines.

2. El uso del castellano como lengua de reconocimiento, suponiendo una novedad en el control de UAVs basado en herramientas libres.

3. La posibilidad de integrarse en sistemas empotrados, de forma que el usuario no requiera de un ordenador, de conexión a Internet o de software comercial auxiliar.

Sobre estos objetivos, en el presente artículo se presenta la estructura general del sistema de control de UAVs a través de órdenes de voz, el conjunto de comandos, las diferentes tecnologías utilizadas en el desarrollo de la propuesta. Posteriormente, se describen los componentes del sistema y se explica la utilización de simuladores de vuelo para la fase de prueba e implementación en UAVs. Finalmente se discuten las conclusiones y se proponen algunos trabajos futuros

2. MATERIALES Y MÉTODOS

2.1. Estructura general

Teniendo en cuenta las consideraciones expuestas en el apartado anterior se ha diseñado e implementado un sistema de reconocimiento de voz libre, personalizable y en español. El sistema sigue la estructura que se muestra en la Figura 1.

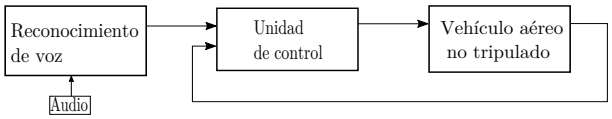

Figura 1: Estructura del sistema de control

En primer lugar, es necesario que la computadora o sistema empotrado reciba las señales de audio procedentes de un micrófono. Estas señales son procesadas por un sistema de reconocimiento de voz que compara las señales recibidas con un modelo, y en caso de coincidir con alguna de las estructuras sintácticas existentes en el modelo, envía un mensaje a la unidad de control.

La unidad de control contiene una máquina de estados finitos que almacena en todo momento el estado del UAV. Puede darse el caso de que se desee despegar el vehículo sin que este se encuentre todavía con los motores armados, o que se deba esperar a que conmute a un determinado modo de vuelo, por ejemplo.

La unidad de control, partiendo del estado del vehículo y de los comandos recibidos debe poder enviar comandos de control reconocibles por este, así como recibir comandos de estado.

2.2. Comandos de voz

El primer paso a la hora de diseñar un modelo de reconocimiento de voz es establecer qué conjunto de comandos se desea reconocer. En función de estos comandos, se adaptarán las tecnologías uti-

lizadas para darles soporte. El desarrollo de estas tecnologías será el objeto de la siguiente sección.

Se muestra en el Cuadro 1 el conjunto de comandos utilizados:

Cuadro 1: Conjunto de comandos.

Nombre	Descripción
Armar	Arma los motores
Desarmar	Desarma los motores
Despegar	Despega el vehículo
Aterrizar	Aterriza el vehículo
Subir	Eleva el vehículo
Bajar	Desciende el vehículo
Parar	Detiene el vehículo en el aire
Giro Izq.	Gira el vehículo a la izquierda
Giro Der.	Gira el vehículo a la derecha
Izquierda	Desplaza el vehículo a la izquierda
Derecha	Desplaza el vehículo a la derecha
Adelante	Desplaza el vehículo hacia adelante
Atrás	Desplaza el vehículo hacia atrás
Más	Aumenta la velocidad de despl.
Menos	Disminuye la velocidad de despl.

Se ha identificado cada comando por un nombre común, aunque como se explicará más tarde, un comando puede ser llamado por diferentes locuciones o expresiones.

3. DESARROLLO DE LA PROPUESTA

Como se ha comentado al comienzo, el sistema se compone de diferentes tecnologías que actúan de forma conjunta. Se ha tratado de utilizar tecnologías libres y gratuitas, consiguiendo así reducir al máximo el coste final del producto.

A continuación se explicará cómo se han aplicado las diferentes tecnologías para diseñar el sistema. Se muestran en la Figura 2 las tecnologías utilizadas.

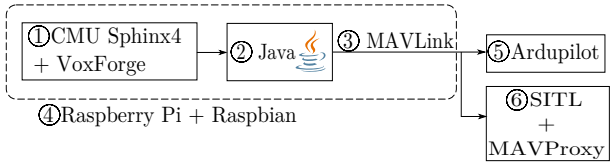

Figura 2: Resumen de las tecnologías utilizadas

3.1. Reconocimiento de voz

El bloque principal del sistema de reconocimiento de voz es la biblioteca *CMU Sphinx4* [20][21], desarrollada por la Carnegie Mellon University, Sun Microsystems Laboratories y el MERL. Se ha escogido por su modularidad y flexibilidad, por ser compatible con múltiples modelos de lenguaje y por estar desarrollada por completo en Java, lo cual le aporta la portabilidad que se requiere.

El reconocimiento está basado en Modelos Ocultos de Markov o *HMM* [22]. En este tipo de modelos se describen una serie de estados con una determinada probabilidad de evolucionar entre ellos.

CMU Sphinx4 sólo requiere por parte del programador tres elementos para poder realizar un reconocimiento de voz[23]:

En primer lugar, un modelo acústico, el cual es específico de cada lenguaje o dialecto y ha sido generado como resultado de un proceso de entremamiento.

En segundo lugar, un diccionario fonético, el cual contiene la relación entre cada una de las palabras a reconocer y los sonidos o fonemas que lo componen.

En tercer lugar, y de especial importancia en este diseño, un modelo de lenguaje. Los modelos de lenguaje determinan qué estructura sintáctica debe seguir una oración a reconocer. De esta forma, el sistema de reconocimiento evalúa si el conjunto de palabras reconocidas coincide con alguna de las estructuras del modelo de lenguaje.

El modelo acústico utilizado ha sido *VoxForge*[24][25], en su versión de 2016 compatible con Sphinx4.

Se ha utilizado el diccionario fonético que proporciona *VoxForge*, añadiendo aquellas palabras que no se encontraran en el diccionario original. Se muestra en el Cuadro 2 una relación entre las palabras principales utilizadas en el reconocimiento y su transcripción fonética. Debido a la escasa documentación encontrada que relacionara los fonemas y consonantes con su transcripción correspondiente, se ha trabajado tomando del diccionario fonético ejemplos de palabras que contuvieran los fonemas a utilizar.

El modelo de lenguaje se ha implementado utilizando el formato JSGF o *JSpeech Grammar Format*[26], el cual permite implementar diagramas sintácticos y modelos de lenguaje.

La Figura 3 expone el diagrama sintáctico general del modelo en forma de *diagrama de Conway*[27].

Por otro lado, en la Figura 4 se muestra un ejemplo de comando implementado en diagrama sintáctico. Como se puede observar, tanto la locución "girar izquierda" como "giro a la izquierda" serían reconocidas como locuciones válidas. No serían válidas las expresiónes "rotar a la izquierda" o "girar ha-

Cuadro 2: Algunas de las transcripciones utilizadas.

Palabra	Transcripción
Armar	a r m a r
Desarmar	d e s a r m a r
Despegar	d e s p e g a r
Aterrizar	a t e rr i z a r
Subir	s u b i r
Bajar	b a j a r
Parar	p a r a r
Giro	j i r o
Derecha	d e r e ch a
Izquierda	i z k i e r d a
Adelante	a d e l a n t e
Atrás	a t r a s
Más	m a s
Menos	m e n o s

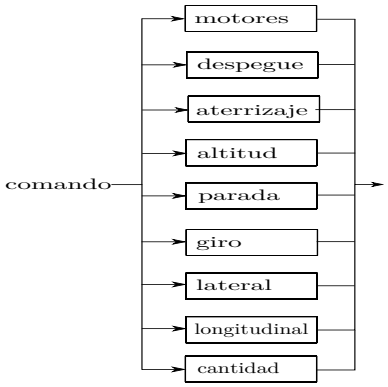

Figura 3: Diagrama sintáctico general

cia la izquierda", al no coincidir con la estructura sintáctica definida. La forma de operar es equivalente para el resto de comandos.

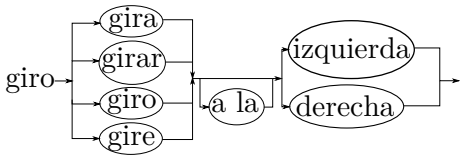

Figura 4: Diagrama sintáctico de un comando del tipo *giro*

Con los tres elementos necesarios para el reconocimiento se ha configurado la biblioteca *Sphinx4* en modo de reconocimiento continuo *click to talk*[23], en el cual, el programador selecciona el momento en que comienza el reconocimiento, y el sistema detecta cuándo se produce un silencio en la locución para finalizar el proceso de reconocimiento.

3.2. Unidad de control

La unidad de control tiene tres funciones fundamentales:

En primer lugar, accionar el sistema de reconocimiento de voz, que en caso de devolver un resultado válido (coincidente con alguna estructura del modelo de lenguaje), lo almacenaría en una cadena de caracteres, denominada *utterance* o declaración.

En segundo lugar, comprobar las cadenas devueltas por el sistema de reconocimiento de voz con una tabla similar a la del Cuadro 1 para enviar al UAV los comandos que correspondan.

Por último, comprobar el estado del UAV, a fin de controlar el flujo de mensajes y evitar que se envíe un comando determinado aún cuando no se cumplan las condiciones adecuadas para enviarse.

El diagrama de flujo de la unidad de control se muestra en la Figura 5.

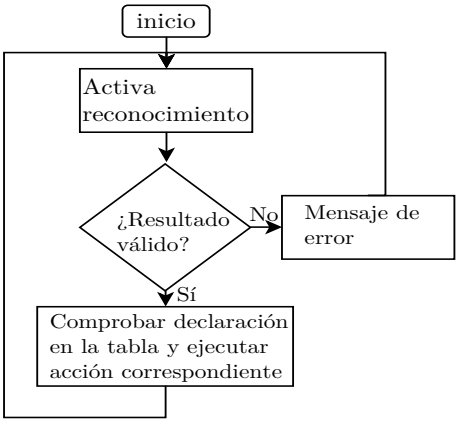

Figura 5: Diagrama de flujo de la unidad de control

Del diagrama de flujo resulta importante prestar especial atención a la comprobación de cada comando y ejecución de la acción correspondiente.

Como se desarrollará más a fondo en el siguiente apartado *"Comunicación con el vehículo"*, cada UAV cuenta con dos variables de estado de especial importancia: la que indica si se encuentra armado, es decir, si los motores se encuentran en movimiento; y la que indica el modo de vuelo.

El modo de vuelo requerido en esta implementación para despegar y mover el vehículo es el modo *guiado*, el cual necesita de GPS y funciona dando órdenes de forma periódica para controlar el estado del UAV en cada momento: una vez termina una acción permanece quieto hasta que recibe una nueva orden.

Por tanto, se debe comprobar antes de ejecutar

cualquier orden que el UAV se encuentra armado y en modo guiado. Por motivos de seguridad, el proceso de armado sólo se puede realizar mediante el comando *"armar"* y nunca de forma automática por medio de la unidad de control. En el caso del modo guiado, en caso de requerirse este, si el UAV no se encuentra en dicho modo, se enviaría la orden de cambiar de modo y se esperaría al cambio para ejecutar la orden correspondiente.

Se muestra en la Figura 6 el diagrama de flujo de este bloque.

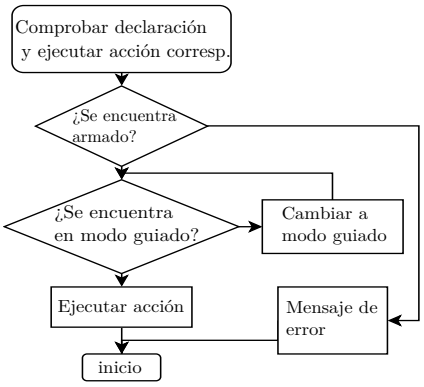

Figura 6: Diagrama de flujo del bloque de comprobación

Toda la unidad de control ha sido programada usando el entorno de desarrollo Java, puesto que se trata de un entorno que cumple con dos especificaciones importantes: ser compatible con Sphinx4 y ser portable a diferentes entornos de ejecución.

3.3. Comunicación con el UAV

A la hora de comunicarse con el *UAV* se han establecido dos canales posibles. En primer lugar, conexión Serie por medio de la biblioteca *rxtxcomm*[28]; y en segundo lugar, conexión TCP/IP por medio de la clase *Socket* de Java[29].

El protocolo de comunicación utilizado es *MAVLink*[30], un protocolo que opera en la capa Aplicación del modelo OSI y por tanto se puede utilizar independientemente del canal. La principal ventaja de este protocolo de comunicación es su compatibilidad con una gran cantidad de proyectos libres para el control de vehículos aéreos no tripulados. Se destacan *Ardupilot* y *MAVProxy* al haber sido utilizados como firmware del *UAV* y en la simulación, respectivamente. Se muestra la estructura de un paquete MAVLink en la Figura 7. Se puede encontrar una descripción de los diferentes campos en [30].

MAVLink es un protocolo orientado a mensajes, por lo que los campos del *payload* serán interpretados por el receptor del mensaje dependiendo del

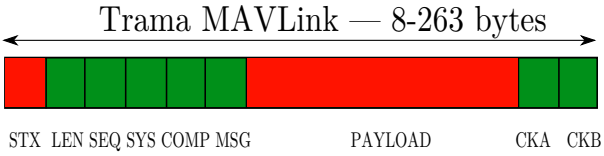

Figura 7: Estructura de un paquete MAVLink

tipo de mensaje que se envíe. La referencia oficial de mensajes MAVLink se encuentra en [31]. Para armar, desarmar, despegar, aterrizar y pasar a modo guiado se utilizarán mensajes específicos preestablecidos para cada opción, los cuales se muestran en el Cuadro 3.

Cuadro 3: Mensajes MAVLink específicos

Mensaje	**Id**	**Parámetro**
Armar	400	1
Desarmar	400	0
Despegar	22	Altura Inicial
Aterrizar	21	0
Modo guiado	11	4 → Id. del modo guiado

El estado del *UAV* se ha capturado a partir del mensaje *Heartbeat* que este envía a una frecuencia de 1Hz. El modo de vuelo está contenido en el campo *custom_mode* del mensaje, mientras que el estado de armado está contenido en el campo *base_mode*. El indicador de modo guiado es igual a 11, mientras que el indicador de que se encuentra armado es igual a 217. Comparando los campos con estos valores se puede ejecutar la secuencia mostrada en la Figura 6.

El movimiento del *UAV* se ha controlado mediante mensajes específicos del modo guiado. Estos comandos tienen una característica importante: no anulan los mensajes enviados por un mando, de forma, que en caso de producirse algún error, se puede tomar el control de este mediante un mando. Los mensajes utilizados son *SET_POSITION_TARGET_LOCAL_NED* y *MAV_CONDITION_YAW*, para desplazamiento y giro, respectivamente, los cuales se encuentran descritos en la documentación oficial[32].

Para el desplazamiento se ha establecido una velocidad base para cada eje de coordenadas de 3 m/s, siendo la velocidad máxima de 5 m/s y la mínima de 1 m/s. Se puede aumentar o disminuir la velocidad en el último eje controlado mediante los comandos *"más"* y *"menos"*, respectivamente. Tal y como se especifica en [32], este mensaje se envía a una frecuencia de 1Hz, de forma que si el *UAV* no recibe un mensaje de desplazamiento en menos de 1s, permanecerá quieto en el aire. Esto se ha conseguido implementando un *Java Thread*

[33] que llama al comando de desplazamiento a dicha frecuencia.

El comando de giro permite girar un determinado ángulo en el sentido que se indica. Si se envía este mensaje a una determinada frecuencia, el *UAV* puede rotar a velocidad constante. Se ha enviado un mensaje cada 100 ms, con parámetro 1º de forma que la velocidad angular es de 10 $^o/s$.

El comando *"parar"* fija a 0 todas las velocidades antes configuradas hasta que se llame a un nuevo comando de desplazamiento o giro.

3.4. Sistema empotrado

Una vez se ha realizado el diseño completo del sistema, se ha implementado en un sistema empotrado. Se ha utilizado una placa *Raspberry Pi 3 Model B* por su altas prestaciones y reducido tamaño, precio y consumo[34]. No obstante, puede ser implementado en placas similares como *Banana Pi* o *Beaglebone*. El sistema operativo utilizado ha sido *Raspbian*, una distribución de *Debian* para *Rasperry Pi* y similares. La aplicación Java se ha configurado para ejecutarse al inicio, de forma que una vez que el sistema operativo se inicializa, la aplicación comienza a ejecutarse.

Puesto que la *Rasperry Pi* no cuenta con una entrada de audio, se ha conectado al puerto USB de esta una tarjeta de sonido *C-Media CM108*[35]. A la entrada de audio de la tarjeta de sonido se ha conectado el micrófono de unos auriculares de diadema *Sennheiser PC 3 Chat*.

4. RESULTADOS

4.1. Simulación

Para comprobar el correcto funcionamiento del sistema se ha utilizado el simulador *SITL*, el cual, junto a la interfaz de de MAVLink *MAVProxy*, permiten simular un entorno en condiciones equivalentes a las de un UAV[37]. El sistema se ha conectado mediante una conexión TCP/IP. Se muestra en la Figura 8 una vista del simulador conectado al sistema.

4.2. Implementación para UAV

Una vez se ha comprobado el correcto reconocimiento de los diferentes comandos y su respuesta en el simulador, se ha comunicado por conexión Serie con un UAV mediante una antena de telemetría USB de 915MHz. La controladora del UAV ha sido una *Pixhawk PX4*©[38] con firmware *Ardupilot*.

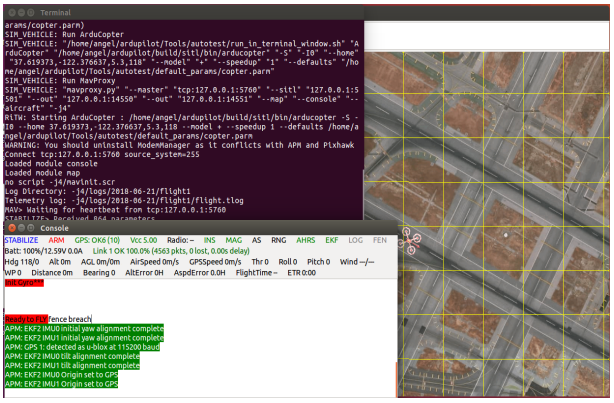

Figura 8: Vista del simulador SITL con MAVProxy

4.3. Datos obtenidos

Tanto en la simulación como en la prueba con UAV, el sistema responde correctamente a los comandos de voz.

Las velocidades de rotación y traslación, medidas en el simulador, coinciden con las esperadas.

Se aprecia un pequeño tiempo de respuesta entre que el UAV recibe una orden de movimiento y la ejecución de este. Se verifica que las ordenes de voz pueden ser anuladas por medio del mando.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha logrado diseñar e implementar un sistema de control para vehículos aéreos no tripulados mediante comandos de voz en castellano. El conjunto de tecnologías software utilizadas ha sido libre, gratuito y abierto. Se ha logrado integrar el sistema en una plataforma de reducido tamaño, precio, y sin necesidad de requerir conexión a Internet.

No obstante todavía se presentan tres retos fundamentales a alcanzar.

El primero de ellos es conseguir un sistema robusto en condiciones de ruido. Las prácticas se han realizado en entornos con bajo nivel de ruido y a una distancia del UAV suficiente como para que el ruido de las hélices no dificulte el reconocimiento de los comandos.

El segundo es mejorar la interfaz de interacción con el usuario, añadiendo verificaciones verbales mediante sistemas *text-to-speech* que devuelvan mensajes de voz indicando la orden a ejecutar o cualquier alerta de importancia. En esta linea, incluir un sistema que permita cancelar la última orden en caso de no haberse reconocido correctamente.

El tercero y último es portar este diseño a teléfonos móviles, aprovechando que la implementación se ha realizado íntegramente en Java, para poder acercarlo a más usuarios y reducir más aún el coste final de este.

Agradecimientos

Este trabajo ha sido financiado con el proyecto Nanosen-AQM (SOE2/P1/E0569) del programa SUDOE de la UE.

English summary

Development of a control system for unmanned aerial vehicles by voice commands in Spanish

Abstract

The use of unmanned aerial vehicles has been extended significantly in the recreational, scientific and cinematographic frame. This expansion has led to the development of new control modes which make handling easier and which can be adapted to each situation. This paper introduces a voice recognition control system. The main advantages this design proposes are: the use of open-source tools, its customization capabilities, the election of spanish as the recognition language and the possible implementation on embedded systems.

Keywords: *Voice recognition, UAV,* CMU Sphinx4, *MAVLink,* Raspberry Pi.

Referencias

[1] Fernández Barrero, M. A. (2018). "Periodismo y drones. Retos y oportunidades del uso de drones para la narración informativa en España." *Doxa Comunicación*, 26, pp 35-58.

[2] Yang, H., Lee, Y., Jeon, SY. et al. (2017) "Multi-rotor drone tutorial: systems, mechanics, control and state estimation." *Intel Serv Robotics*, 10, p 79.

[3] Fleureau J., Galvane, Q, Tariolle, FL. y Guillotel, P. (2016). "Generic Drone Control Platform for Autonomous Capture of Cinema Scenes."pp 35-40.

[4] Singireddy S.R.R., Daim T.U. (2018) "Technology Roadmap: Drone Delivery – Amazon Prime Air.". *Infrastructure and Technology Management. Innovation, Technology, and Knowledge Management.* Springer, Cham

[5] Chen J., Wang S., Qu X., Yi W. (2018) "A Modelling Framework of Drone Deployment for Monitoring Air Pollution from Ships.". *Intelligent Interactive Multimedia Systems and Services.* Smart Innovation, Systems and Technologies, vol 98. Springer, Cham

[6] Oromi Fragoso P., Fernández Archilla A. (2016) "Uso de drones en ingeniería civil: levantamientos topográficos.". *Revista del Colegio de Ingenieros Técnicos de Obras Públicas*, vol. 407, pp 24-28

[7] Kim S.J., Jeong Y., Park S., Ryu K., Oh G. (2018) "A Survey of Drone use for Entertainment and AVR (Augmented and Virtual Reality)." *Augmented Reality and Virtual Reality. Progress in IS.* Springer, Cham

[8] Lopez Herrera J. (2015) "La normativa sobre los drones." *Mundo del agrónomo: la revista del Colegio Oficial de Ingenieros Agrónomos de Centro y Canarias*, vol. 30, pp 10-12

[9] Peshkova E., Hitz M., Ahlström D. (2017) "Exploring User-Defined Gestures and Voice Commands to Control an Unmanned Aerial Vehicle." *Intelligent Technologies for Interactive Entertainment.* INTETAIN 2016 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 178. Springer, Cham

[10] Speicher, S. and Preuß, S. (2004). "Voice-Controlled Ubiquitous Computing", *Proceedings of the International Conference on Wireless Networks, ICWN*, vol. 1, pp 116-122.

[11] D. Boucha, A. Amiri and D. Chogueur, "Controlling electronic devices remotely by voice and brain waves." *2017 International Conference on Mathematics and Information Technology (ICMIT)*, Adrar, 2017, pp. 38-42.

[12] Bohouta G., Z Këpuska V. (2018) "Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)" *The 8th IEEE Annual Computing and Communication Workshop and Conference*

[13] Bellegarda J.R. (2014) "Spoken Language Understanding for Natural Interaction: The Siri Experience." *Natural Interaction with Robots, Knowbots and Smartphones.* Springer, New York, NY

[14] "Ford SYNC 3 con AppLink", *Ford.es*. Disponible en: https://goo.gl/7WdJiC. [Acceso: 18- Jun- 2018].

[15] "What Is Alexa? What Is the Amazon Echo, and Should You Get O", Wirecutter: Reviews for the Real World, 2018. [Online]. Disponible en: https://goo.gl/V6ds35. [Acceso: 18- Jun- 2018].

[16] "La navegación de drones por voz es posible con what3words y DXC Technology — what3words", what3words, 2018. [Online]. Disponible en: https://goo.gl/3KuqAT. [Acceso: 16- Jun- 2018].

[17] "How to Build a Portable Voice Controlled Drone for Under $500 : Alexa Blogs", Developer.amazon.com, 2016. [Online]. Disponible en: https://goo.gl/9ziiFg [Acceso: 16- Jun- 2018].

[18] "DJI Developer", Developer.dji.com, 2018. Disponible en: https://developer.dji.com/. [Acceso: 16- Jun- 2018].

[19] "Háblale a través del micro y disfruta del vuelo — SkyRover Voice Command", Colorbaby.es, 2018. Disponible en: https://goo.gl/VZWAFo. [Acceso: 16- Jun- 2018].

[20] Lamere, Paul, Kwok, Philip, B. Gouv, Evandro, Singh, Rita, Walker, William y Wolf, Peter. (2003). "The CMU Sphinx4 Speech Recognition System". *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing* (ICASSP 2003).

[21] Vojtas P., Stepan J., Sec D., Cimler R. y Krejcar O. (2018) "Voice Recognition Software on Embedded Devices".

[22] Rabiner, L.R. (1989) "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE, 77(2)* pp 257-286.

[23] N. Shmyrev, "Basic concepts of speech recognition", CMUSphinx Open Source Speech Recognition, 2018. [Online]. Available: https://goo.gl/K7P8Ku. [Accessed: 19- Jun- 2018].

[24] "Spanish - voxforge.org", *Voxforge.org*, 2006. Disponible en: https://goo.gl/RU4fBd. [Accessed: 19- Jun- 2018].

[25] N. Shmyrev, "Voxforge Spanish Model Released", CMUSphinx Open Source Speech Recognition, 2010. Disponible en: https://goo.gl/nGT9bi. [Acceso: 19- Jun- 2018].

[26] Hunt, A. (2000). "Jspeech grammar format." *W3C Note* http://www. w3. org/TR/jsgf.

[27] Gálvez S., David, R. y Fernández, T. (2018). "Generación de compiladores mediante diagramas de Conway" *ResearchGate*

[28] RXTX for Java", fizzed.com, 2018. Disponible en: https://goo.gl/t84Nt9. [Acceso: 20- Jun- 2018].

[29] "Socket (Java Platform SE 7)", *Docs.oracle.com*, 2018. Disponible en: https://goo.gl/rzqKXw [Acceso: 20- Jun- 2018].

[30] "MAVLink Micro Air Vehicle Communication Protocol - QGroundControl GCS", *Qgroundcontrol.org*, 2018. Disponible en: https://goo.gl/CQJCpr. [Acceso: 20- Jun- 2018].

[31] "MAVLINK Common Message set specifications", Mavlink.org, 2018. Disponible en: https://goo.gl/QXTXkY. [Acceso: 20- Jun- 2018].

[32] "MAVLink Mission Command Messages (MAV_CMD) — Copter documentation", Ardupilot.org, 2018. [Online]. Available: https://goo.gl/6C3nzP. [Accessed: 20- Jun- 2018].

[33] Hyde, P. (1999). Java thread programming (Vol. 1). Indianapolis: Sams.

[34] Raspberry Pi 3 Model B+ - Raspberry Pi", Raspberry Pi, 2018. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Acceso: 21- Jun- 2018].

[35] "CM108 Datasheet", Qsl.net, 2018. Disponible en: https://goo.gl/jDxFNy. [Acceso: 21- Jun- 2018].

[36] "SITL Advanced Testing — Dev documentation", Ardupilot.org, 2018. Disponible en: https://goo.gl/yAYsZL. [Acceso: 21- Jun- 2018].

[37] "Pixhawk Autopilot - Pixhawk Flight Controller Hardware Project", Pixhawk.org, 2018. [Online]. Disponible en: https://goo.gl/icQijR. [Acceso: 21- Jun- 2018].