Congreso Internacional en Ingeniería Electrónica. Mem. Electro 2015, Vol. 37, pp 153-157, Chihuahua, Chih. México
http://depi.itchihuahua.edu.mx/display/memorias_electro/MemoriaElectro2015
ISSN 1405-2172

PLATAFORMA PARA EL CONTROL DE MOTORES DE UN CUADROPTERO

Ureña Acuña Alejandro, Manuel Moisés Miranda Velasco, Miguel Enrique Martínez Rosas, Edgar Omar Cadena Zepeda, José Miguel Pérez Uribe y Manuel Alejandro Juárez Avilés.
Facultad de Ingeniería, Arquitectura y Diseño.
Universidad Autónoma de Baja California
Carretera transpeninsular Ensenada-Tijuana número 3917, colonia playitas.
Ensenada, B.C., C.P. 22860. Teléfono 646-1750744, fax 646-1744333.
urenaa@uabc.edu.mx, mmiranda@uabc.edu.mx, emartine@uabc.edu.mx, edgar.cadena@uabc.edu.mx, jm.perez.uribe@gmail.com, mjuarez2187@gmail.com

RESUMEN.

En este trabajo se desarrolló e implementó una plataforma para el control de los motores de un vehículo aéreo no tripulado (UAV por sus siglas en ingles), en configuración cuadróptero con un bajo costo. El usuario tendrá la posibilidad de implementar algoritmos de control, para realizar diversos experimentos. El cuadróptero seleccionado para este desarrollo es el, Ar.Drone v1.0, el cual tiene un costo accesible y se encuentra disponible en el mercado nacional. Además, cuenta con sensores dan información acerca del estado del vuelo a los algoritmos de control implementados.

En la plataforma el control de los motores se realiza a través de programa en lenguaje C, los datos de los sensores fueron adquiridos a través de uno de los puertos del sistema de procesado del UAV. Lo anterior permite que el algoritmo de control se realice en el sistema de a bordo.

Se implementó un control PID diferente al que el fabricante tiene de fábrica, y con esto comprobar el funcionamiento de la plataforma con algoritmos de control desarrollados por el usuario.

ABSTRACT.

This paper presents a platform for individual control of each of the engines of an unmanned aerial vehicle (UAV) in quadcopter configuration, in which the user can implement their own algorithms to control the UAV. The control engine is performed in C language, with prior knowledge of the GPIO ports, and sensor data were acquired through one of the ports of UAV microcontroller.

Also a PID control was implemented to ensure that the user can implement their own control algorithms to an UAV manufactured.

To make the experimental part, the Ar.Drone 1.0 quadcopter was used. This UAV has sensors that control the stability of flight and provide information about the status.

1. INTRODUCCIÓN

En la actualidad hay poca diversidad de vehículos aéreos no tripulados (UAV) capaces de realizar un vuelo con algoritmos realizados por los usuarios, en el mercado nacional. Sin embargo, hay un gran interés por parte de investigadores en desarrollar investigaciones relacionadas con los UAV del tipo cuadróptero, para poder implementar algoritmos propios que puedan controlar la posición y las distintas variables que puedan afectar su operación durante el vuelo.

Un cuadróptero o cuadricóptero, es una aeronave que entra en vuelo gracias a la fuerza de cuatro rotores que usualmente se montan en forma de cruz. Es un vehículo totalmente diferente si se compara con un helicóptero, principalmente por la manera en que ambos son controlados. Los helicópteros son capaces de cambiar el ángulo en el que ejercen fuerza sus hélices, mientras que un cuadróptero no. Otra diferencia entre ellos es que los cuadrópteros no requieren conexiones mecánicas para variar el ángulo de paso de las palas del rotor a medida que giran. Lo anterior los hace sistemas que requieren poco mantenimiento y a la vez presentan un problema como objetivo de control, ya que son sistemas subactuados.

A la fecha, muchos de los cuadrópteros de bajo costo que se desarrollan están destinados a usarse para el entrenamiento y recreación. Aunque, dichos sistemas pueden ser utilizados como prototipos, no presentan una plataforma necesaria para poder realizar diversos experimentos.

El cuadróptero utilizado para este trabajo es el AR.Drone de la compañía francesa Parrot, el cual es un vehículo diseñado para volar por control remoto via una conexión Wi-Fi. Inicialmente fue diseñado para ser controlado por los sistemas operativos móviles o tabletas, tales como iOS o Android. Parrot también lanzó una API abierta para drones[1] que permite el desarrollo de aplicaciones de terceros. Sin embargo, en esta API no permite la implementación de nuevos algoritmos de control de los motores, solo permitiendo controlar la referencia de control de algunas de las variables del sistema.

Dado su bajo costo, su disponibilidad a nivel mundial y de contar con piezas de refacción, se tiene una comunidad muy grande de usuarios e información de su funcionamiento. Este trabajo solo pretende crear una plataforma sistematizada para el uso de estos cuadrópteros, en el Cuerpo Académico Comunicaciones e Instrumentación Electrónica de la Facultad

[1] El termino drone es el de uso común para denominar a los UAV y es derivado del inglés.

de Ingeniería, Arquitectura y Diseño de la Universidad Autónoma de Baja California.

En el trabajo se divide de la siguiente manera; en la siguiente sección se presenta una descripción del Hardware y Software utilizados en el desarrollo, en la sección 3 se describe la metodología para usar la plataforma, en la sección 4 se muestra la implementación de un control PID y sus resultados experimentales, en la última sección se dan las conclusiones del trabajo.

2. HARDWARE Y SOFTWARE

El AR.Drone se compone de varios de un sistema de telemetría y control a través de un sistema embebido (ARM) corriendo una distribución especializada del sistema operativo Linux. Así como, de diversos sensores y etapas de potencia conectadas al sistema de telemetría por medio de microcontroladores (atmega328) de uso específico.

2.1. Hardware.

El drone es propulsado por motores sin escobillas con tres fases de corriente controlada por un micro-controlador. El drone detecta si cualquiera de las hélices está bloqueada y si una hélice que gira encuentra un obstáculo que detiene todos los motores inmediatamente, en caso de que alguno de los motores no funcione no enciende el resto. Este sistema de protección evita choques repetidos.

Una batería de LiPo de 1000mAh a 11.1 V se utilizan para hacer volar el cuadróptero. Cuando dicha nave detecta una tensión de batería baja, primero envía un mensaje de advertencia al usuario, y luego aterriza de forma automática. Si el voltaje alcanza un nivel crítico, todo el sistema se apaga para evitar cualquier comportamiento imprevisto [1].

Los sensores se encuentran por debajo del casco central. El Ar.Drone 1.0 cuenta con un sensor ultrasónico que proporciona las mediciones de altitud para la estabilización de altitud automática y asistida además del control de la velocidad vertical. Además, usa una cámara VGA (640 * 480), que sólo puede transmitir imágenes QVGA (320 * 240). La cámara vertical del Ar.Drone 1.0 utiliza un QCIF (176 * 144) a 60fps. El sistema cuenta con un giroscopio de tres ejes que permiten medir los tres ángulos relacionados con el movimiento del UAV.

Figura 1. Ar.Drone V1.0

2.2. Software.

Ar.Drone puede ser controlado desde cualquier dispositivo móvil que soporte Wi-Fi a través de una que crea el sistema de control. El control de la Ar.Drone se realiza a través de 3 canales principales de comunicación.

El control y la configuración del drone se realiza mediante el envío Comandos AT en el puerto UDP 5556. La latencia de transmisión de los comandos de control es fundamental para la experiencia del usuario. Los comandos se deben enviar de forma regular (por lo general 30 veces por segundo). Los comandos AT son cadenas de texto codificados como caracteres ASCII y son generados por las bibliotecas Ar.Drone.

La Información sobre el cuadróptero y su estado interno como la posición, velocidad, la velocidad de rotación del motor, entre otros son llamados NAVDATA, se envían por el drone a través del puerto UDP 5554. Son enviados aproximadamente 15 veces por segundo en el modo de demostración, y 200 veces por segundo en el modo completo (depuración).

Los Datos de flujo de vídeo son enviados por el Ar.Drone a través Puerto TCP 5555. Las imágenes del Ar.Drone se decodifican por códec UVLC (MJPEG-similares) o P264 (H.264 similares).

Un cuarto canal de comunicación, llamado puerto de control, está establecido en el puerto TCP 5559 para transferir datos críticos, se utiliza para recuperar los datos de configuración, y para reconocer información importante.

El canal FTP 5551 se utiliza para transferencia de archivos, los cuales se depositaran en la carpeta ¨update¨ del sistema operativo del cuadróptero [2].

Tabla 1. Puertos de comunicación del sistema embebido.

Puerto	*Numero*	*Función*
UDP	5556	Control con comandos AT
UDP	5554	Datos de Navegación
TCP	5555	Transmisión de video
TCP	5559	Datos críticos
FTP	5551	Transferencia de archivos

El sistema de embebido de telemetría y control consiste en un microprocesador ARM9 RISC de 32 bits @ 468 MHz, una memoria DDR SDRAM de 128 MB, un sistema operativo con núcleo Linux, un modem Wi-Fi b/g y además dispone de un conector USB de alta velocidad para poder almacenar los vídeos y las fotos que se realicen.

3. METODOLOGIA.

La plataforma desarrollada hace uso de que el sistema de telemetría y control establece un servidor FTP para conexión remota, de forma tal que se puede crear un programa que se ejecute en el drone. Por lo que, el primer paso es conocer las

funciones básicas de las GPIOs (General Purpose Input/Output) y con eso poder desarrollar un sistema alternos de control. El esquema general para la implementación de nuevos esquemas de control se muestra en la Figura 2.

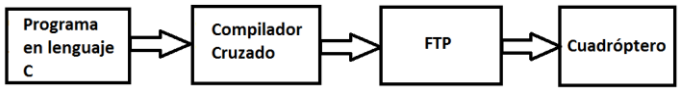

Figura 2 Esquema para la implementación de un nuevo controlador

3.1. Control de los motores del Ar Drone.

El control se realiza a través de los puertos GPIO, los cuales que realizan funciones específicas en el sistema de telemetría y control del cuadróptero. Los puertos utilizados para el control se muestran en la **¡Error! No se encuentra el origen de la referencia.**.

Tabla 2. GPIO relacionadas al control de los motores.

Puerto GPIO	*configuración*	*Descripción*
63	Salida	LED rojo 1=encendido
64	Salida	LED verde 1=encendido
68	Salida	Motor 1 =Seleccionado
69	Salida	Motor 2 =Seleccionado
70	Salida	Motor 3 =Seleccionado
71	Salida	Motor 4 =Seleccionado

3.2. Compilador cruzado (cross-compiler).

Un compilador cruzado o cross compiler es un compilador capaz de generar un código ejecutable para una plataforma distinta a la que se está desarrollando. Dichos compiladores son muy utilizados en sistemas embebidos en donde los recursos que se tienen son muy limitados [3].

La desventaja de este tipo de compiladores es que cada arquitectura utiliza un compilador específico, si se utiliza un cross-compiler diferente, el sistema enviara un error de segmentación y el programa compilado no se ejecutara.

Como sistema de desarrollo se utiliza un sistema de cómputo con el sistema Ubuntu 14.04, para instalar el cross-compiler se utiliza el comando

$ sudo apt-get install gcc-armlinux-gnueabi

Para entrar en ambiente de desarrollo del cross-compiler se escribe en la terminal el comando

$codesourcery-arm-2009q3.sh

Una vez en el entorno de desarrollo del cross-compiler, escribimos los permisos del archivo que queremos cambiar a arquitectura ARM mediante el comando:

$ chmod 777 "archivo"

Es así como tendremos un archivo compatible con la arquitectura ARM y listo para ejecutarse.

3.3. Transferencia de archivos vía FTP.

El protocolo FTP es un protocolo de comunicaciones destinado al intercambio informático de archivos sobre una red TCP/IP. Permite, desde una computadora, copiar archivos de una computadora a otra dentro de la misma red [4].

Para transferir archivos al cuadróptero, se debe utilizar un cliente FTP, para este proyecto se utilizó "FILEZILLA", el cual es un programa gratuito y de código abierto. La transferencia de archivos hacia el cuadróptero se realizará sobre el puerto 5551, dicho puerto es el que está configurado para utilizar el protocolo FTP.

Una vez transferido el archivo ejecutable aparecerá en la carpeta "UPDATE" del cuadróptero, listo para ejecutarse.

3.4. Implementación del programa.

La implementación del programa se realiza comunicándose con el cuadróptero por la dirección IP 192.168.1.1 usando el protocolo TELNET.

```
BusyBox v1.14.0 (2012-06-01 16:35:43 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cd update
# cd bin
# ls
attitude    gpio        navboard    video
fly         motorboard  vbat
# killall program.elf
# ./motorboard
programa control de motor ... Presione q para salir
Motor: 1,2,3,4=correr cada motor al 50% 5=correr todos los motores al 50% ,=desa
celerar al 1% .0=acelerar al 1 8.401459E-310spacio=parar motores
Leds:  a=leds apagados  s=modo vuelo  d=modo emergencia
-
```

Figura 3. Ejecución del programa controlador de motores.

Con el procedimiento anterior y el conocimiento de los GPIO de control y medición es posible desarrollar nuevos algoritmos de control sobre el Ar.Drone. A través del control individual de los motores, así como el control de la aceleración y desaceleración de cada uno, y de todos en conjunto.

Cabe destacar que los motores no arrancaran si la velocidad está por debajo del 40%, y la velocidad mínima que alcanzan al desacelerar es del 33%, si la velocidad está por debajo de este valor, los motores se apagaran y el cuadróptero podría aterrizar bruscamente.

4. EXPERIMENTOS

Para comprobar el funcionamiento de la plataforma desarrollada, se implementó un controlador PID para controlar los ángulos y velocidades del cuadróptero.

4.1. Control PID

Debido a que el controlador PID que posee el Ar drone no es del todo adecuado, se propuso implementar un nuevo control basado en la configuración en paralelo cuya ecuación característica está dada por:

$$y(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d\,e(t)}{dt}, \qquad (1)$$

dónde e(t) Representa la función de error, K_p la constante Proporcional, K_i la constante Integradora y K_d la constante Derivativa. En el que el diagrama a bloques se caracteriza como:

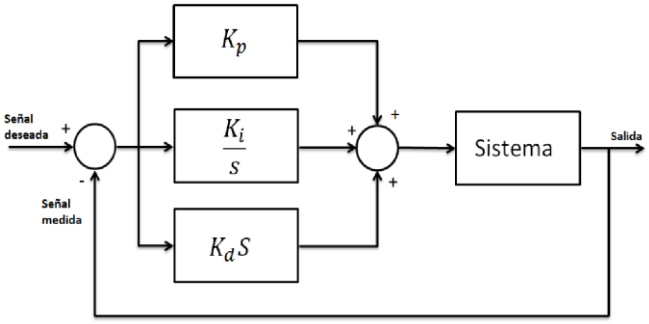

Figura 4. Modelo a bloques del controlador PID implementado [5].

4.2. Implementación del controlador PID.

La primera implementación del PID, fue sobre la plataforma de un solo eje. El comportamiento fue el esperado, consiguiendo un equilibrio a cero grados y la poción de referencia deseada.

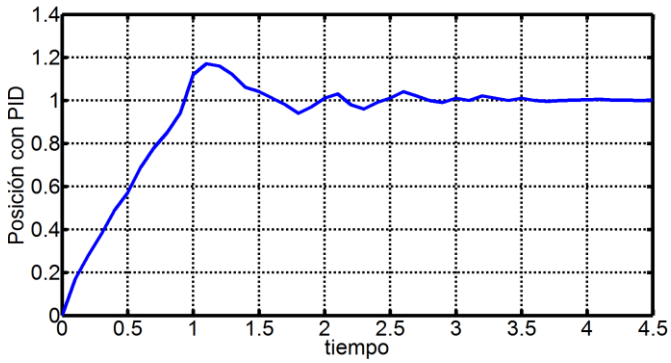

Figura 5. Posición del cuadróptero en el eje z.

En la figura anterior observamos la posición en el eje z del cuadróptero al utilizar del controlador PID durante el vuelo del drone. En este experimento se configuró el drone para que se mantuviera a una altura de un metro. Cabe destacar que esta prueba se realizó a techo cerrado controlando perturbaciones externas que afectaran el vuelo. Los datos fueron recuperados del puerto UDP 5554, los cuales arrojan los datos de navegación.

Posteriormente se realizaron pruebas bajo efectos de ráfagas de viento, siendo capaz de recuperar la posición de equilibrio.

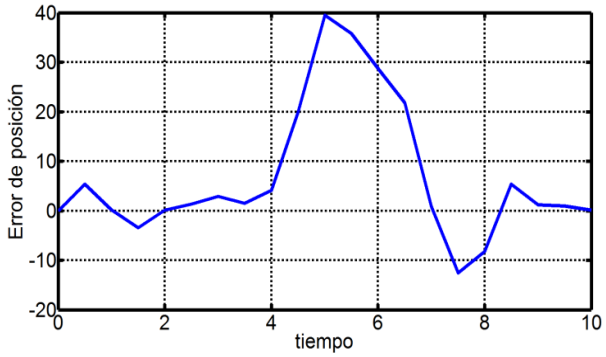

Figura 6. Acción del controlador PID para control posición.

Como se muestra en la figura anterior, el cuadróptero se estabiliza ante perturbaciones manteniendo el error lo más cercano a cero hasta que sufre una fuerza externa, provocando la desestabilización de la plataforma, donde se nota la consiguiente corrección por parte del control.

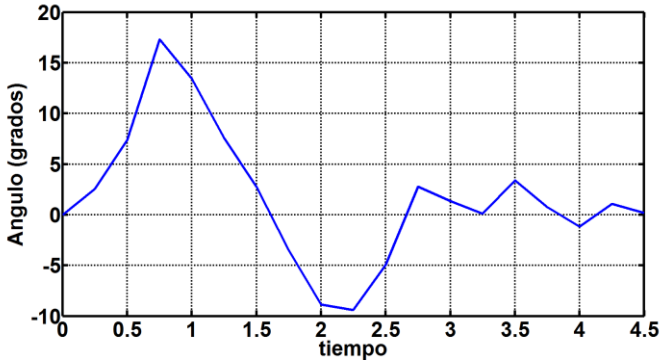

Figura 7. Corrección del ángulo de alabeo (Roll).

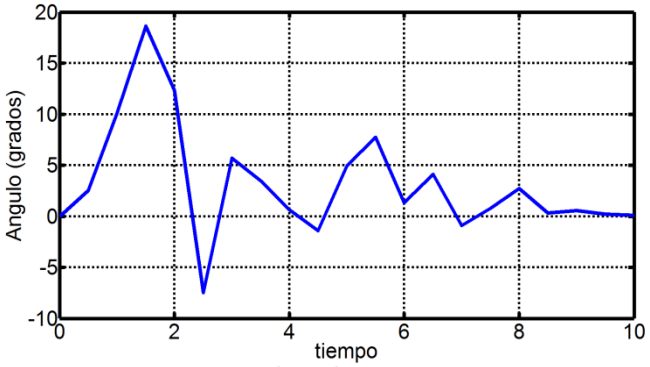

Figura 8. Corrección del ángulo de guiñada (Yaw).

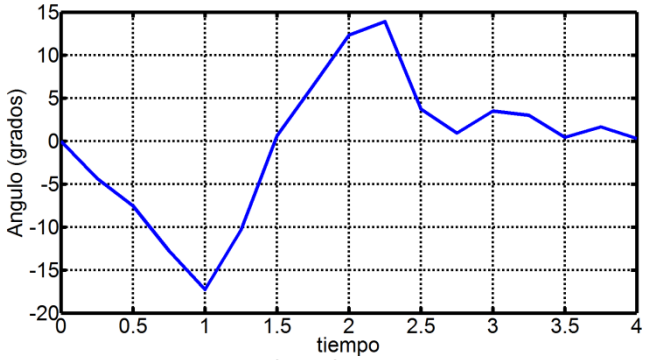

Figura 9. Corrección del ángulo de cabeceo (Pitch)

En las figuras 7, 8, y 9 observamos las correcciones que el controlador hacer a cada uno de los ángulos (yaw, raw y pitch), estabilizando el cuadróptero a 0º, cabe destacar que dichas pruebas se realizaron al aire libre por lo que se pueden apreciar perturbaciones externas (aire, brisa de mar, entre otros)

5. CONCLUSIONES

Durante los experimentos se pudo comprobar la altamente inestable dinámica del cuadróptero Ar.Drone, por lo que el diseño de un controlador capaz de corregir la posición ante cualquier perturbación condiciones externas e incertidumbres es indispensable.

La plataforma para el control autónomo de los motores se realizó exitosamente, lo que permitirá en un futuro mayor autonomía y poder realizar rutas de vuelo prefabricadas sin necesidad de utilizar el GPS. El programa que controla los motores es totalmente independiente del SDK que proporciona la empresa francesa *Parrot*, esto permitió realizar un controlador PID e implementar algoritmos propios.

6. REFERENCIAS.

[1] Control library for AR.Drone 2.0, Jakub Hvizdoš, prof. Ing. Peter Sinčák CSc, Enero de 2015.

[2] Modelamiento Matemático y control de un helicóptero de cuatro motores. Miguel Parra Muñoz, Eugenio Feitosa Fortaleza, Jones Mori Alves Da Silva. Diciembre de 2013.

[3]http://docsetools.com/articulos-noticias consejos/article_126239.html. Revisado el 2 de mayo de 2015.

[4] Ingeniería de Software: Una Guía para Crear Sistemas de Información. Alejandro Peña Ayala. 2006

[5] Teaching a drone autonomous target tracking. Joost Baptist, MARCEL beishizen, Elbert fliek.