ISSN: 2594-2492

Simulación de un Cuadri-rotor en el Software Webots

Edmundo M. Cortes Vazquez* Amparo D. Palomino Merino*
César Martínez Torres Gibran Etcheverry****

** Benemérita Universidad Autónoma de Puebla, Puebla de Zaragoza, Puebla, México*
(e-mail: emiguel_cortes@hotmail.com, ampalomino@gmail.com).
*** Universidad de las Américas Puebla, San Andrés Cholula, Puebla, México*
(e-mail: cesar.martinez@udlap.mx, gibran.etcheverry@udlap.mx)

Resumen: En este trabajo se presenta la simulación de un Cuadri-rotor en el software de simulación robótica de código abierto Webots, con el objetivo de verificar el desempeño de un algoritmo de control para seguimiento de trayectorias. El modelo del Cuadri-rotor se construye en el mundo virtual a partir de nodos basados en lenguaje de modelado de realidad virtual (VRLM). Para verificar la simulación en Webots se utiliza el algoritmo de control Proporcional Derivativo (PD) usando los mismos parámetros que una simulación en Matlab. Adicionalmente se modifica la resolución del sensor de posicionamiento global(GPS) y de la unidad de medición inercial (IMU) para observar su influencia en el seguimiento de trayectoria.

Palabras clave: Cuadri-rotor, Webots, Control PD, Simulación 3D, Seguimiento de trayectoria, Simulador Código abierto.

1. INTRODUCCIÓN

La investigación sobre el control de vehículos aéreos no tripulados (VANT), en particular de cuatro rotores, también conocido como Cuadri-rotor, figura 1, es una importante área de investigación de los vehículos aéreos no tripulados, donde probar las estrategias de control en los prototipos reales significa riesgo de daño por fallas, por lo que es de lo más recomendable realizar simulación numérica para verificar el rendimiento del diseño de control; sin embargo, los resultados de la simulación por lo general, no permiten visualizar de una manera más clara, el comportamiento real del Cuadri-rotor.

Los entornos de simulación son siempre una herramienta importante en investigación, es de gran importancia el contar con un instrumento que permita simular condiciones del mundo real, así como probar previamente el uso de sensores que pudieran ser de alto costo antes de implementarlos en el prototipo. En el área es comúnmente utilizada la simulación en Matlab como en Peng y Qingbo (2011) y en Paiva et al (2015), en estas simulaciones sería complicado dotar al Cuadri-rotor con sensores. En Bouabdallah et al. (2005) se ocupa una simulación en Webots para validar un PID, desde esa fecha el software se ha dotado de mejoras en la simulación de la dinámica y la integración de sensores.

En este trabajo, se ha desarrollado una simulación del Cuadri-rotor basada en el software Webots. Podemos ver que en Cyberbotics (2018a) Webots es un simulador de robótica de código abierto que proporciona un entorno de desarrollo completo para modelar, programar y simular robots. Aunque existen ambientes de simulación similares, como lo son Gazebo, V-Rep entre otros, existen varios aspectos que hacen atractivo este software, por ejemplo, la programación se puede realizar en el lenguaje que se encuentre familiarizado el usuario (C / C ++, Java, Python y *MATLAB*), además este simulador es uno de los pocos que funcionan en multiplataforma, incluyendo Windows. Webots sigue en continua revisión y desarrollo, anteriormente era un software comercial con licencia profesional y estudiantil, pero desde diciembre de 2018 se lanza la versión R2019a en código libre, en la cual se desarrolla este trabajo.

Fig. 1. Vehículo aéreo no tripulado tipo Cuadri-rotor.

2. MODELO DEL CUADRI-ROTOR

En este trabajo se asumieron las siguientes hipótesis para el desarrollo del modelo, la estructura se supone rígida y simétrica, el centro de masa y el origen de las coordenadas coinciden, las hélices son rígidas, el empuje y la resistencia al avance son proporcionales al cuadrado de la velocidad de las hélices, el modelo se obtiene de la misma manera que en Bouabdallah (2007) y Vianna (2007).

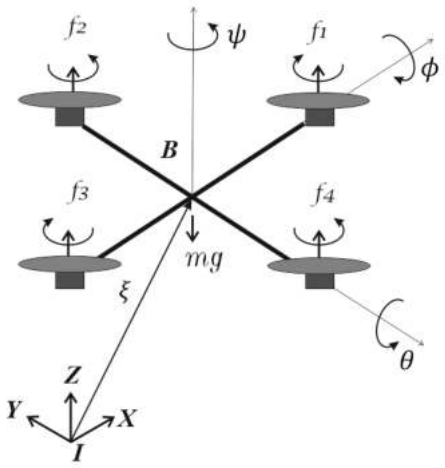

Fig. 2. Posición y orientación del Cuadri-rotor.

El funcionamiento del Cuadri-rotor es sencillo. La posición ($\xi = x, y, z$) y la orientación ($\eta = \phi, \theta, \psi$) deseada se logra al variar la velocidad y par de los cuatro rotores, como se muestra en la figura 2 y en Bouabdallah et al (2005).

Usando el formalismo de Euler-Lagrange se obtiene el modelo del Cuadri-rotor, el cual tiene seis ecuaciones que describen su dinámica, como en Zulu y John (2014) y Bouabdallah (2007):

$$\ddot{x} = \frac{1}{m}(cos\psi sen\theta cos\phi + sen\psi sen\phi)U_1 \quad (1)$$

$$\ddot{y} = \frac{1}{m}(sen\psi sen\theta cos\phi - cos\psi sen\phi)U_1 \quad (2)$$

$$\ddot{z} = -g + \frac{1}{m}(cos\theta cos\phi)U_1 \quad (3)$$

$$\ddot{\phi} = \frac{(I_y - I_z)}{I_x}\dot{\psi}\dot{\theta} - \frac{J_R\Omega}{I_x}\dot{\theta} + \frac{l}{I_x}U_2 \quad (4)$$

$$\ddot{\theta} = \frac{(I_z - I_x)}{I_y}\dot{\psi}\dot{\phi} + \frac{J_R\Omega}{I_y}\dot{\phi} + \frac{l}{I_y}U_3 \quad (5)$$

$$\ddot{\psi} = \frac{(I_x - I_y)}{I_z}\dot{\phi}\dot{\theta} + \frac{1}{I_z}U_4 \quad (6)$$

Donde U_1 es el empuje total de la suma de cada uno de los empujes $f_1, f_2, f_3\ y\ f_4$. En términos de las velocidades de los rotores:

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (7)$$

Y los pares U_2, U_3, U_4 que representan las entradas al sistema.

$$U_2 = bl(\Omega_4^2 - \Omega_2^2) \quad (8)$$

$$U_3 = bl(\Omega_1^2 - \Omega_3^2) \quad (9)$$

$$U_4 = k_T(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (10)$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \quad (11)$$

Tabla 1. Definición de los símbolos

Símbolo	Definición
ϕ	ángulo de roll
θ	ángulo de pitch
ψ	ángulo yaw
Ω	velocidad general de los rotores
I_x, I_y, I_z	inercias de la estructura
J_R	inercia del rotor
b	factor de empuje
l	distancia de los rotores al centro de la estructura
k_T	factor de arrastre
m	masa de la estructura

3. MODELO EN WEBOTS

El lenguaje utilizado en Webots es el de modelado de realidad virtual (VRML de sus siglas en inglés de Virtual Reality Modeling Language), es el lenguaje de programación utilizado para modelar el mundo virtual donde se simulará el comportamiento de los robots. Este lenguaje posibilita la descripción de objetos 3D a partir de prototipos basados en formas geométricas básicas o estructuras descritas a partir de vértices y aristas como se explica en Cyberbotics (2018b).

La figura 3 muestra la interfaz gráfica del software de simulación Webots que está conformada por:

- El Árbol de escenas (izquierda): Muestra la información de describe un mundo simulado en forma jerárquica. El árbol de escenas esta estructurado como un archivo VRML97, compuesto por una lista de nodos y cada nodo contiene campos.
- Ventana 3D (centro): Permite interactuar con la simulación 3D. En esta ventana se puede, navegar en

la escena, mover un objeto, aplicar fuerza a un objeto y aplicar torque a un objeto.

- Editor de texto (derecha): Es un editor de texto de múltiples pestañas que permite desarrollar controladores. El editor presenta resaltado de sintaxis para el lenguaje compatible con Webots.

- Consola (abajo): Muestra mensajes de compilación y salidas de ejecución.

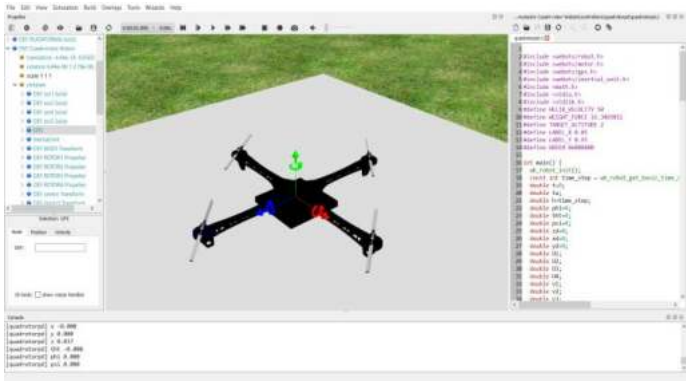

Fig. 3. Interfaz gráfica de Webots.

El Cuadri-rotor se construye a partir de un Nodo *Robot* en el árbol de escena como se muestra en la figura 4, en *children* del Nodo *Robot* como se ve en la figura 5, se añaden los Nodos *Transform* para generar la estructura, agregando formas geométricas para las carcasas de los motores y para mejorar el detalle de los brazos de la aeronave se importa un archivo VRLM97 de los brazos a *Transform*. Para que le sea posible sustentarse se agregan los nodos *Propelle*r como se muestra en la figura 6, donde se agregan las características de las propelas como empuje y arrastre.

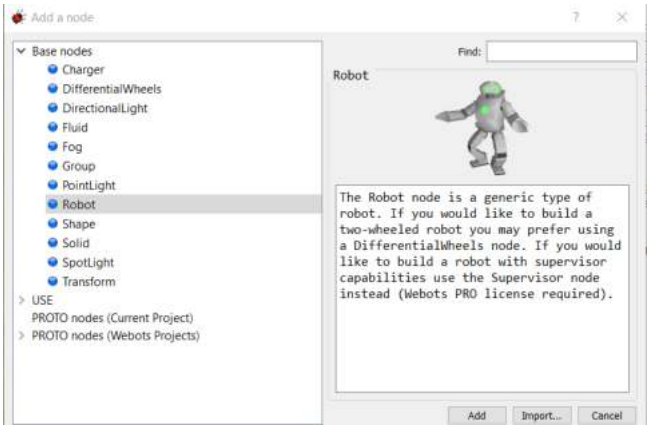

Fig. 4. Nodo *Robot* en Webots.

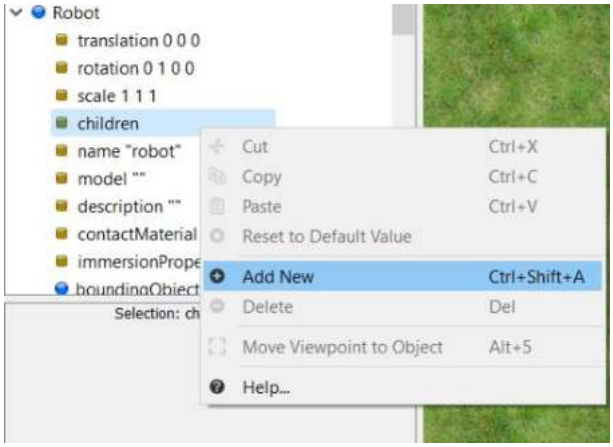

Fig. 5. Pestaña *children* del nodo *Robot*.

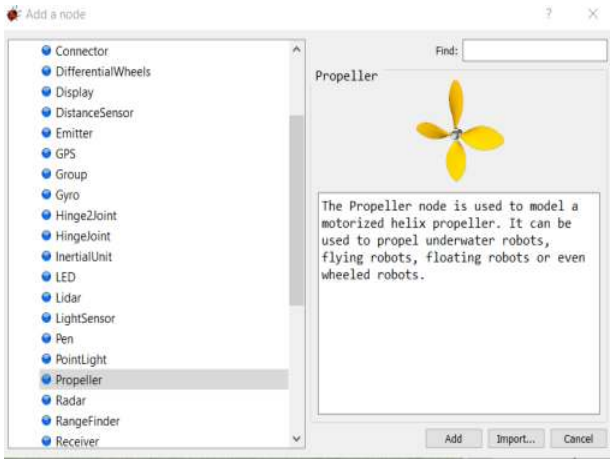

Fig. 6. Nodos admitidos por Nodo *Robot*. Los Nodos *GPS*, *InertialUnit* y *Propeller* son utilizados en el Cuadri-rotor.

Además de adicionar los sensores *GPS* y *InertialUnit* mostrados en la lista de nodos de la figura 6, colocándolos en el centro geométrico del vehículo, quedando terminado el Cuadri-rotor como se ve en la figura 7 y al centro de la figura 3. En los sensores en Webots es posible modificar algunas de sus características como lo es la resolución, el ruido, el ángulo de detección, tiempo de muestreo entre algunas otras. Adicionalmente en este trabajo se plantea la interesante opción de modificar la resolución con la idea de poder probar el comportamiento del Cuadri-rotor a una resolución de un sensor comercial.

Es de gran importancia cuidar las dimensiones de los elementos creados y propiedades de material ya que el software calcula sus centros de masa y los momentos de inercias de cada elemento para generar la simulación dinámica.

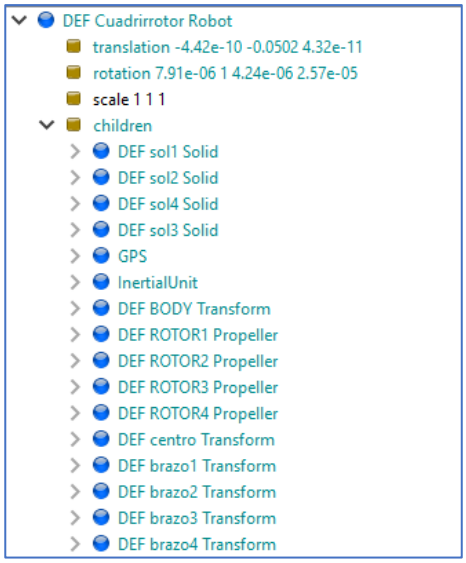

Fig. 7. Árbol de nodos del Cuadri-rotor en Webots.

La implementación del controlador para este trabajo se realiza en código C, en el editor de texto de Webots se programa siguiendo el diagrama de flujo de la figura 8, para después ser ligado al Nodo *Robot* del Cuadri-rotor.

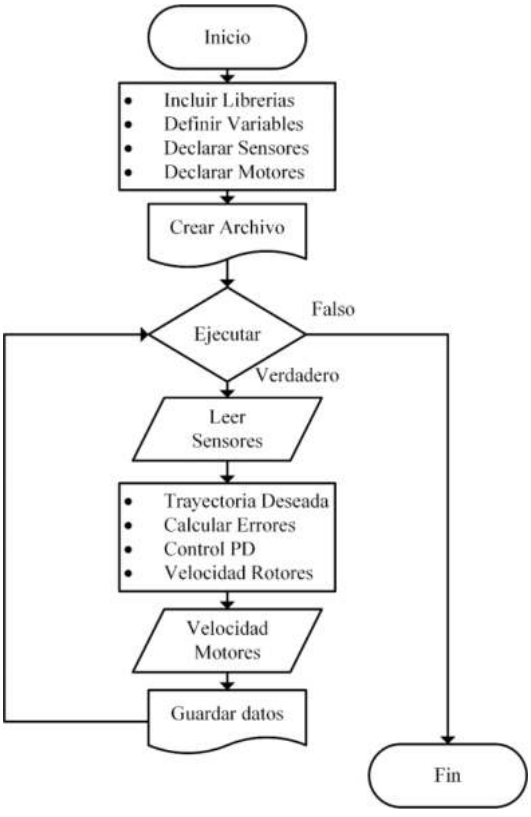

Fig. 8. Diagrama de flujo del Controlador en Webots.

4. ESTRATEGIA DE CONTROL

Para probar que el modelo creado en Webots funciona en congruencia al modelo simulado en Matlab, en este trabajo se utiliza un controlador Proporcional Derivativo (PD).

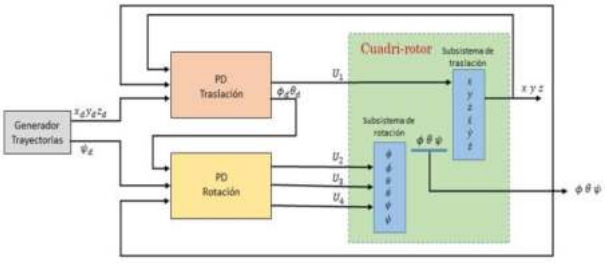

Fig. 9. Diagrama a bloques de la estrategia de control PD.

La implementación del control PD, siguiendo el esquema de la figura 9, puede ser definida por:

$$U = K_p e(t) + K_d \frac{d}{dt} e(t) \tag{12}$$

donde K_p es la ganancia proporcional y K_d es la ganancia derivativa, $e(t)$ es la diferencia entre el valor deseado y el valor medido.

Las entradas de control U_2, U_3, U_4 se tienen similar a Dikmen et al (2009) con:

$$U_2 = K_{p,\phi}(d_\phi - \phi) + K_{d,\phi}(d_{\dot{\phi}} - \dot{\phi}) \tag{13}$$

$$U_3 = K_{p,\theta}(d_\theta - \theta) + K_{d,\theta}(d_{\dot{\theta}} - \dot{\theta}) \tag{14}$$

$$U_4 = K_{p,\psi}(d_\psi - \psi) + K_{d,\psi}(d_{\dot{\psi}} - \dot{\psi}) \tag{15}$$

Las ecuaciones (13)-(15) representan los torques para cada eje debido al empuje de los rotores U_2, U_3 y U_4 respectivamente y con $K_{p,\phi}$, $K_{p,\phi}$ y $K_{p,\psi}$ que son las ganancias proporcionales. $K_{d,\phi}$, $K_{d,\theta}$ y $K_{d,\psi}$ son las ganancias derivativas. d_ϕ, d_θ y d_ψ son pitch, roll y yaw deseados, ϕ, θ y ψ corresponden a los valores actuales de orientación.

Las ecuaciones (1)-(3) se pueden resolver para determinar las variables de control:

$$U_1 = \frac{1}{cos\theta cos\phi}(z_3 + mg) \tag{16}$$

$$d_\phi = arcsen\left(\frac{m}{U_1}(z_1 sen\psi - z_2 cos\psi)\right) \tag{17}$$

$$d_\theta = arcsen\left(\frac{m}{U_1}(z_1 cos\psi + z_2 sen\psi)\right) \quad (18)$$

$$z_1 = K_{px}(d_x - x) + K_{d,x}(d_{\dot{x}} - \dot{x}) \quad (19)$$

$$z_2 = K_{py}(d_y - y) + K_{dy}(d_{\dot{y}} - \dot{y}) \quad (20)$$

$$z_3 = K_{pz}(d_z - z) + K_{dz}(d_{\dot{z}} - \dot{z}) \quad (21)$$

Considerando la posición deseada d_x, d_y y d_z son los valores conocidos, mientras que x, y, z son los valores medidos.

Donde (16) es la fuerza de control del desplazamiento vertical z, (17) y (18) son los ángulos deseados para colocar el cuadrirrotor en x, y deseadas del plano cartesiano, z_1, z_2 y z_3 son las señales de salidas de control.

5. SIMULACION Y RESULTADOS

Las simulaciones se realizan con los mismos parámetros en Webots y en Matlab; en Matlab está construido el modelo del Cuadri-rotor a partir de bloques en Simulink del modelo dinámico descrito por (1) a (6).

Tabla 2. Parámetros de simulación

Parámetro	Valor	Parámetro	Valor
I_x, I_y, I_z	$0.001\ Nms^2$	d_ψ	$\pi/3$ rad
b	$0.0001\ Ns^2$	d_z	0.1 tm
l	$0.35\ m$	m	1.2 kg
k_T	$0.0001\ Nms^2$	$K_{p,\phi}, K_{p,\theta}, K_{p,\psi}$	1
g	$9.81\ ms^{-2}$	$K_{d,\phi}$,$K_{d,\theta}$,$K_{d,\psi}$	0.3
J_R	$0.0001\ Nms^2$	$K_{p,x}$,$K_{p,y}$, $K_{p,z}$	1
d_x	$0.5\cos(0.2t)$ m	$K_{d,x}$,$K_{d,y}$, $K_{d,z}$	0.8
d_y	$0.5sen(0.2t)$ m		

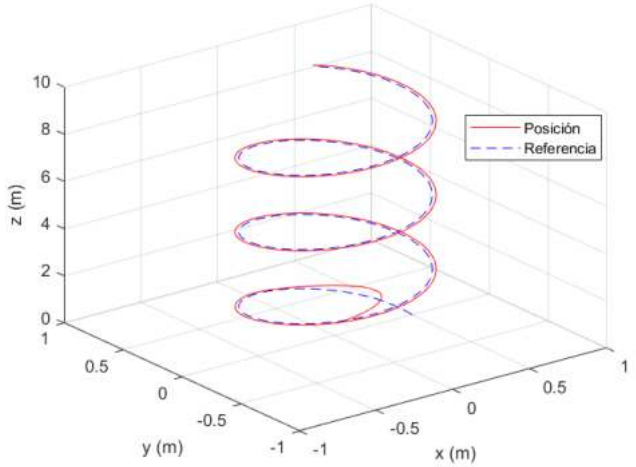

Fig. 10. Variables de posición en trayectoria 3D en Matlab.

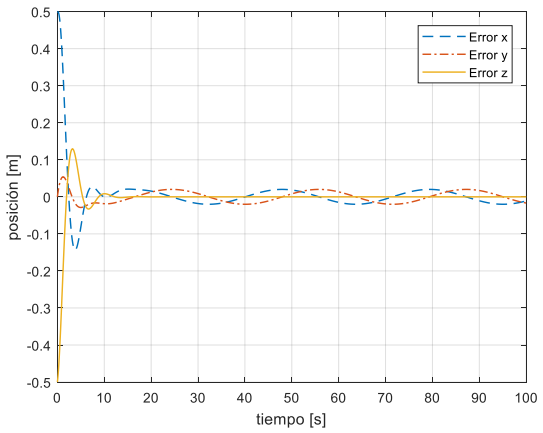

Fig. 11. Errores de posición en seguimiento de trayectoria en Matlab.

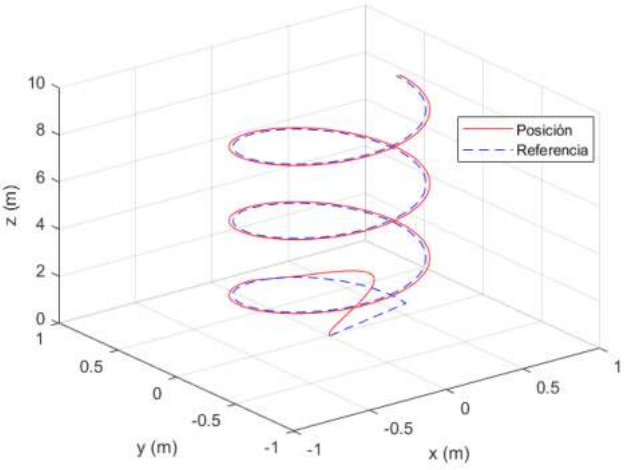

Fig. 12. Variables de posición en trayectoria 3D en Webots.

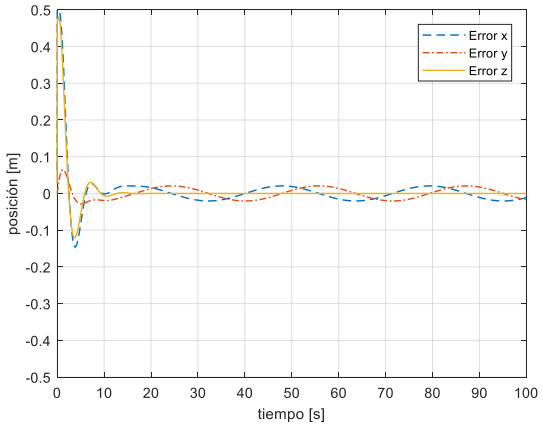

Fig. 13. Errores de posición en seguimiento de trayectoria en Webots.

La simulación en Matlab; del seguimiento de la trayectoria se observa en la figura 10 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida, se observa que Cuadri-rotor es capaz de seguir la trayectoria deseada. El error de las posiciones está acotado, esto se observa en la figura 11.

La simulación en Webots; del seguimiento de la trayectoria se observa en la figura 12 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida, se observa que Cuadri-rotor es capaz de seguir la trayectoria deseada y el error en las posiciones en la figura 13 donde se observa que el error es muy similar a la simulación en Matlab.

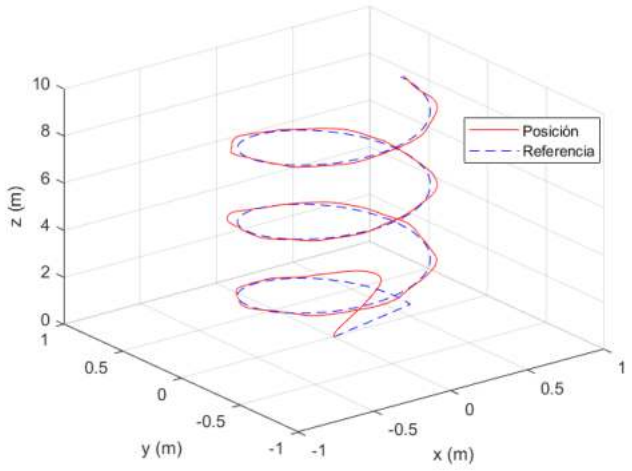

Fig. 14. Variables de posición en trayectoria 3D en Webots modificando la resolución de los sensores.

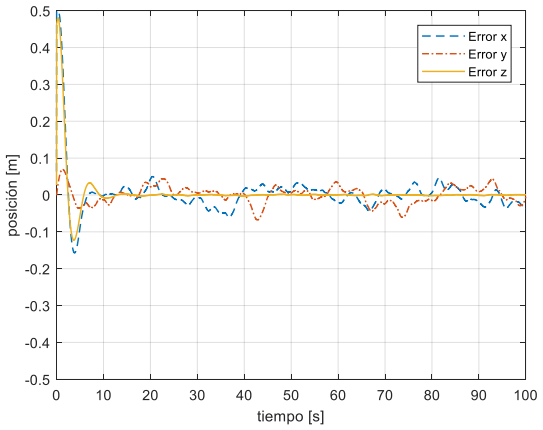

Fig. 15. Errores de posición en seguimiento de trayectoria en Webots, con resolución modificada.

También se realiza la simulación en Webots modificando la resolución de la IMU y del GPS 0.001rad y 0.001 m respectivamente, debido a que en la anterior están considerados como instrumentos ideales, con una resolución máxima y sin ruido en Webots. En la figura 14 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida y el error en las posiciones en la figura 15 donde se observa que el error es mayor a la simulación anterior, pero ejemplificando que es posible modificar los parámetros de los sensores y observar que para esta resolución logra seguir la trayectoria el control PD, pero, aumenta el error de posición.

6. CONCLUSIONES

Al implementar la simulación en Webots del Cuadri-rotor con el control PD y compararla con la simulación en Matlab, se logra verificar que la simulación en Webots realizada en este trabajo, entrega resultados satisfactorios, generando con esto, la confianza de usar el ambiente de simulación Webots, para la evaluación de estrategias de control. Como trabajo futuro se planea adecuar la simulación en Webots para que funcione como un laboratorio virtual para la prueba de algoritmos de control.

REFERENCIAS

Bouabdallah, S. (2007). Design and Control of Quadrotor with Application to Autonomus Flying. Ph.D. Dissertation, Lausanne Polytechnic University. Suiza.

Bouabdallah, S., Murrieri, P., y Siegwart, R. (2005). Towards Autonomous Indoor Micro VTOL. 18(2), 171-183.

Cyberbotics. (2018a). Webots User Guide. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

Cyberbotics. (2018b). Webots Reference Manual. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

Dikmen, I., Arısoy, A., y Temelta, H. (2009). Attitude control of a quadrotor. 4th International Conference on Recent Advances in Space Technologies, pp. 722–727. Turquia.

Paiva, E., Soto, J., Salinas, J., y Ipanequé. W. (2015). Modeling and PID Cascate Control of a Quadcopter for Trajectory Tracking. Conference on Electrical, Electronic Engineering, Information and Communication Technologies, 809-815. Chile.

Peng, L., Qingbo, G. (2011). Real-Time Simulation System for UAV Based on Matlab/Simulink. Conference on Computing, Control and Industrial Engineering. China.

Vianna, G. (2007). Modelado y Control de un Helicoptero Quadrotor. Tesis M.S., Dept. Ingenieria de Sistemas y Automática, Universidad de Sevilla, Sevilla.

Zulu, A., John S. (2014). A Review of Control Algorithms for Autonomous Quadrotors. Open Journal of Applied Sciences, 4, 547-556.

