Sistema de Control para un Robot Aerodeslizante

Ezequiel Pecker Marcosig[1,5], J. Francisco Presenza[1], Juan I. Giribet[1,2,3], and Ignacio Mas[3,4]

[1]GPSIC - Facultad de Ingeniería, Universidad de Buenos Aires, Argentina
[2]Instituto Argentino de Matemática "Alberto Calderón" (IAM), Argentina
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina
[4]Instituto Tecnológico de Buenos Aires, Argentina
[5]Instituto de Ciencias de la Computación (ICC), Argentina

***Resumen*—En este trabajo se presenta la construcción de un robot aerodeslizante, y la implementación del sistema de medición y control utilizando ROS (Robot Operating System). Este vehículo fue diseñado para el estudio de estrategias de control híbrido en sistemas ciber-físicos. Para comprobar el funcionamiento de esta plataforma se cerró el lazo utilizando una estrategia de control simple (PID). Se presentan los resultados experimentales preliminares obtenidos para el seguimiento de referencias.**

***Keywords*—Robot Aerodeslizante, ROS, robótica móvil, fusión de sensores, sistemas de control.**

I. Introducción

La variación del periodo de muestreo en los controladores de tiempo discreto tiene efectos en el desempeño del sistema en lazo cerrado. Para estudiar este fenómeno, es útil construir sistemas físicos susceptibles a dichas variaciones, de manera tal que sus efectos se vean magnificados. Por ejemplo, los sistemas físicos con baja fricción tienen dinámicas que resultan muy sensibles tanto al tiempo entre actuaciones del controlador como a las perturbaciones externas.

En este trabajo se presenta un tipo particular de robot aerodeslizante. El TACHOBot (Fig. 1) cuyas siglas en inglés corresponden a *Test Assembly for the Control of Hybrid Objectives*, es una plataforma experimental de 3-DOF, construida a partir de materiales reciclados y piezas confeccionadas mediante impresión 3D, que se desliza sobre una superficie con muy bajo rozamiento. Este vehículo fue diseñado para estudiar estrategias de control híbrido en sistemas ciber-físicos [1], [2].

En el estudio de estrategias para el control de formaciones de satélites se utilizan vehículos similares al TACHOBot. Estos vehículos han sido desarrollados para tener baja fricción de modo que simulen el movimiento en el espacio. Los primeros proyectos que proponen una plataforma de simulación para estudiar estrategias de control para formaciones de satélites se realizaron en el JPL de NASA [3]. En ese trabajo se proponen vehículos de 6-DOF, en donde el vínculo en las partes móviles y entre el vehículo y el piso es mediante una capa de aire (*air-bearing*). Existen otros proyectos con plataformas similares, como los TEAMS-3D y 5D del Instituto de Sistemas Espaciales (ISS) en Alemania [4], [5] y del Laboratorio de Sistemas Espaciales Distribuidos del Instituto Tecnológico de Israel [6]. Estos dos últimos vehículos son de 3-DOF y 5-DOF respectivamente. En todos estos proyectos utilizan superficies especiales de forma de reducir al mínimo la fricción para realizar los experimentos, ya sea utilizando perforaciones por donde sale aire a presión o bien superficies extremadamente lisas y niveladas. En este último caso, los vehículos utilizan tanques de aire a presión para alimentar los *air-bearings*.

Email address: epecker@fi.uba.ar

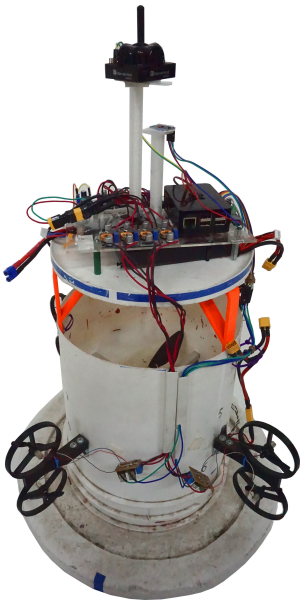

Figura 1. TACHOBot: Robot autónomo aerodeslizante. Los motores laterales producen las fuerzas necesarias para controlar el movimiento. El álabe circular inferior distribuye una película de aire debajo del robot.

En resumen, tanto el TACHOBot como los vehículos mencionados, sirven como plataformas experimentales para el estudio de distintos fenómenos. En nuestro caso, nos interesa investigar el efecto de la variación del periodo de muestreo en controladores de tiempo discreto. A diferencia de los diseños presentados en dichos trabajos, proponemos un vehículo dotado de propulsores que generan una película de aire debajo del mismo capaz de sustentarlo.

El trabajo se organiza de la siguiente manera. En la sección II se desarrolla el modelo físico del robot. El hard-

ISBN: en trámite

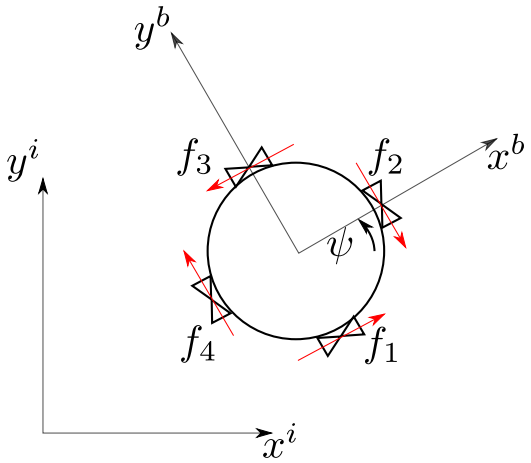

Figura 2. Diagrama de fuerzas y ternas de referencia.

ware utilizado en la construcción del robot se describe en la sección III, mientras que en la sección IV se desarrolla el control y el observador utilizado para fusionar datos de distintos sensores. En la sección V se presenta ROS (Robot Operating System) y se describe la implementación del sistema de control en ROS. Finalmente en la sección VI se muestran los resultados de algunos ensayos realizados y la sección VII concluye el trabajo.

II. Física del Robot

Para generar las fuerzas $\{f_i,\ i = 1, ..., 4\}$ que se observan en la Fig. 2, el TACHOBot cuenta con 8 propulsores unidireccionales $\{M_j,\ j = 1, ..., 8\}$. Los mismos se encuentran ubicados de a pares sobre los 4 puntos de aplicación de estas fuerzas y con sentido de giro opuesto entre sí. De este modo, la fuerza f_i se genera combinando los motores $\{M_{2i-1}, M_{2i}\}$. Estas fuerzas se utilizan para controlar la posición en el plano *XY* y el ángulo de rotación ψ. Se observa que se trata de un sistema sobre-actuado ya que cuenta con 4 acciones forzantes y 3-DOF.

Para mitigar los efectos del rozamiento, y permitir un movimiento libre del vehículo en el plano, un propulsor central ubicado en el interior del TACHOBot empuja aire hacia la parte inferior del mismo formando una película de aire debajo del álabe circular del robot (ver Fig. 1). De esta manera, el TACHOBot es capaz de desplazarse con pequeños esfuerzos de control.

II-A. Modelo Dinámico

Para establecer el modelo dinámico simplificado del TACHOBot es suficiente considerar el movimiento en un plano paralelo a la superficie de deslizamiento. Se definen dos ternas de referencia: una ubicada en el centro del vehículo y solidaria al mismo $\{b\}$, y otra considerada inercial $\{i\}$ (Fig. 2). También nos referimos a estas ternas como local y global respectivamente. El estado cinemático del TACHOBot, en la terna inercial, queda definido por las coordenadas del centro del vehículo y el ángulo de rotación, agrupados en el vector $\mathbf{r}^i = \begin{bmatrix} x^i & y^i & \psi \end{bmatrix}^T$.

Debido a la capacidad de deslizamiento del TACHOBot, es posible despreciar las fuerzas de rozamiento, por lo que aplicando la 2$^{\text{da}}$ Ley de Newton en terna $\{i\}$, se llega a (1). En (2) se muestran la matriz $\mathbf{M}$, que contiene información de la masa (m) y el momento de inercia (I_{zz}) del vehículo, y la matriz de rotación $\mathbf{C}(\psi)$ que transforma de la terna $\{b\}$ a $\{i\}$. El vector $\mathbf{F}^i = \begin{bmatrix} F_x^i & F_y^i & \tau_\psi \end{bmatrix}^T$ contiene las fuerzas y torque resultantes expresados en terna $\{i\}$. Análogamente, se define $\mathbf{F}^b$ en terna $\{b\}$, y sus componentes pueden expresarse en función del conjunto de fuerzas $\mathbf{f}^b = [f_1\ f_2\ f_3\ f_4]^T$ (3). De esta manera, el modelo dinámico del vehículo está dado por:

$$\mathbf{M}\ddot{\mathbf{r}}^i = \mathbf{F}^i = \mathbf{C}(\psi)\,\mathbf{F}^b \tag{1}$$

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad \mathbf{C}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$\mathbf{F}^b = \begin{bmatrix} F_x^b \\ F_y^b \\ \tau_\psi \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ R & -R & R & -R \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}}_{\mathbf{f}^b} \tag{3}$$

En particular para este robot, $m = 2{,}9$ kg, $I_{zz} = 0{,}04\ \text{kg}\,\text{m}^2$, y $R = 0{,}22$ m, siendo esta última la distancia entre el centro del vehículo y el punto de aplicación de las fuerzas.

Reemplazando (3) en (1), se puede describir la dinámica del TACHOBot en función de las fuerzas $\{f_i\}$:

$$\ddot{\mathbf{r}}^i = \mathbf{M}^{-1}\mathbf{C}(\psi)\,\mathbf{T}\cdot\mathbf{f}^b \tag{4}$$

De (4), es posible armar un modelo de variables de estado de la siguiente manera:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}^i \\ \dot{\mathbf{r}}^i \end{bmatrix} \ \therefore\ \dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^i \end{bmatrix} \tag{5}$$

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}}\cdot\mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{C}(\psi)\,\mathbf{T} \end{bmatrix}}_{\mathbf{B}}\cdot\mathbf{f}^b \tag{6}$$

Del modelo (6), se puede observar que se trata de un sistema MIMO (Multiple Input Multiple Output, en inglés), con un doble integrador, no lineal y cuyas variables de estado están acopladas por las funciones trigonométricas de la dependencia de $\mathbf{B}$ respecto de $\psi = \psi(t)$.

ISBN: en trámite

II-B. Linealización del modelo

Se utilizó una técnica de linealización exacta por realimentación denominada Control por Torque Computado (CTC), de amplio uso en robótica. La misma permite linealizar y desacoplar el modelo del sistema mostrado en (6). Ello se logra a partir de la realimentación de ψ, computando el conjunto de fuerzas $\mathbf{f}^b$ necesario para accionar al sistema con las fuerzas resultantes deseadas, que se igualan a la acción de control $\mathbf{u}$. De (1) y (3) se llega a (7), y a partir de ella se encuentra la $\mathbf{f}^b$ necesaria despejando y reemplazando $\mathbf{F}^i$ por $\mathbf{u}$ en (8). Llamando $\hat{\psi}$ a la estimación del ángulo, se deducen las siguientes expresiones:

$$\mathbf{F}^i = \mathbf{C}\left(\psi\right)\mathbf{T}\cdot\mathbf{f}^b \tag{7}$$

$$\mathbf{f}^b = \mathbf{T}^\dagger\mathbf{C}\left(-\hat{\psi}\right)\cdot\mathbf{u} \tag{8}$$

$$\mathbf{T}^\dagger = \mathbf{T}^T\left(\mathbf{T}\mathbf{T}^T\right)^{-1} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{4R} \\ 0 & -\frac{1}{2} & -\frac{1}{4R} \\ -\frac{1}{2} & 0 & \frac{1}{4R} \\ 0 & \frac{1}{2} & -\frac{1}{4R} \end{bmatrix} \tag{9}$$

Donde $\mathbf{C}\left(-\hat{\psi}\right)$ representa la rotación de la terna $\{i\}$ a $\{b\}$, y $\mathbf{T}^\dagger$ es la pseudoinversa de Moore-Penrose de $\mathbf{T}$, cuya expresión se muestra en (9). El uso de esta pseudoinversa, permite obtener la solución de mínima norma, reduciendo la potencia requerida en los propulsores. Reemplazando (8) en (6), obtenemos la siguiente descripción del modelo:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{A}_l}\cdot\mathbf{x} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{M}^{-1}\end{bmatrix}}_{\mathbf{B}_l}\cdot\mathbf{C}\left(\psi\right)\mathbf{T}\mathbf{T}^\dagger\mathbf{C}\left(-\hat{\psi}\right)\cdot\mathbf{u} \tag{10}$$

Es pertinente aclarar que el operador $\mathbf{C}\left(\psi\right)\mathbf{T}$ es parte de la física del TACHOBot, mientras que el operador inverso $\mathbf{T}^\dagger\mathbf{C}\left(-\hat{\psi}\right)$ es implementado numéricamente a partir de la estimación de ψ. El primer operador difiere del segundo únicamente en posibles errores en la estimación de ψ y fenómenos no contemplados en el modelo, de manera que $\mathbf{C}\left(\psi\right)\mathbf{T}\mathbf{T}^\dagger\mathbf{C}\left(-\hat{\psi}\right) \approx \mathbf{I}$. Teniendo en cuenta esto, puede verse cómo el efecto de la linealización exacta radica en compensar numéricamente las no linealidades del sistema, para todos los valores posibles de ψ. De esta manera, obtenemos la expresión del modelo lineal aproximado, siendo $\mathbf{A}_l$ y $\mathbf{B}_l$ las matrices del mismo:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{A}_l}\cdot\mathbf{x} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{M}^{-1}\end{bmatrix}}_{\mathbf{B}_l}\cdot\mathbf{u} \tag{11}$$

Esta técnica de linealización por realimentación permite desacoplar los estados del sistema: al ser $\mathbf{M}$ una matriz diagonal independiente del estado, se puede reordenar el vector de estados de manera tal que las matrices $\mathbf{A}_l$ y $\mathbf{B}_l$ resulten diagonales por bloques, donde cada bloque corresponde a un subsistema independiente. Esto permite diseñar una ley de control lineal sencilla.

III. Descripción del Hardware

Para realizar el control del TACHOBot utilizamos una Raspberry Pi 3 con Ubuntu Xenial (16.04) que se comunica con los sensores y actuadores por medio de un bus I^2C y un puerto serie.

Para la estimación de la posición y orientación del vehículo, se utilizó una Unidad de Medición Inercial (IMU) MPU9250 que combina dos chips: el MPU-6500 que contiene un acelerómetro y un giróscopo triaxiales, y el magnetómetro digital AK8963. Un filtro RTQF, que consiste en una versión simplificada de un filtro de Kalman, es utilizado para obtener la orientación del vehículo $\hat{\psi}$ a $40\,\mathrm{Hz}$. A su vez, las mediciones del acelerómetro fueron fusionadas con los datos provenientes del Sistema Marvelmind [7], como se verá en la Sección IV-B. El mismo consiste en un sistema de posicionamiento para interiores basado en ultrasonido, que fue configurado para entregar datos a una tasa de $16\,\mathrm{Hz}$.

Para generar cada una de las fuerzas $\{f_i\}$ utilizamos motores de corriente continua sin núcleo con hélices de 75 mm que empujan en sentidos opuestos (Fig. 1). Los mismos son alimentados con señales de PWM de 400 Hz cuyo ciclo de trabajo varía alrededor del $50\,\%$. Esto permite suponer que los motores trabajan en su zona lineal. De este modo, si los dos motores correspondientes a un mismo actuador trabajan al $50\,\%$ el empuje resultante es nulo debido a la cancelación de fuerzas. Alterando este equilibrio de fuerzas se obtiene una resultante en el sentido deseado.

Para generar el colchón de aire que permite reducir el rozamiento con el suelo se utilizan dos motores de corriente continua trifásicos MultiStar 2213 de 935 KV con hélices 1045. Los motores están montados sobre el mismo eje, dentro del ducto central de aire (Fig. 1), fuerzan aire hacia abajo mientras giran en sentidos opuestos para cancelar el torque resultante. El empuje generado es suficiente para mantener al robot despegado del suelo.

Utilizamos dos baterías de Litio-Polimero (LiPo) de 3 celdas para alimentar, por un lado a los actuadores centrales, y por el otro la computadora, sensores y actuadores laterales.

IV. Sistema de Control de Posición y Sensado

En esta sección se presenta el sistema de control utilizado para manejar la posición y orientación del robot. Partiendo del modelo lineal aproximado del sistema (11), se desarrolló un esquema de control capaz de accionar los propulsores para seguir la referencia dada. Para que el método de linealización empleado sea apropiado y lograr un buen desempeño de control, es necesario que la estimación del estado del vehículo sea precisa. Esta estimación se realizó a partir de la fusión de datos de la IMU y el sistema de posicionamiento Marvelmind.

ISBN: en trámite

IV-A. Estrategia de control

Como resultado de la linealización del sistema, se elimina el acoplamiento entre los estados. Por lo tanto, el sistema mostrado en (11) queda compuesto por tres subsistemas SISO (Single Input Single Output, en inglés) y es posible utilizar tres controladores independientes, uno por cada estado $\begin{bmatrix} x^i & y^i & \psi \end{bmatrix}$. Se utilizaron controladores PID por su simplicidad y extendido uso.

IV-B. Fusión de datos de la IMU y el Sistema Marvelmind

Debido al bajo rozamiento que presenta el TACHOBot, el desempeño del control es fuertemente dependiente del tiempo entre ejecuciones de los controladores. La tasa de ejecución de las leyes de control utilizadas están fijadas actualmente por los sensores. Por este motivo es importante tener mediciones a la tasa más alta posible.

Como se mencionó en la sección III, el sistema cuenta con diversos sensores de navegación. Estos sensores tienen características complementarias. Por un lado los sensores inerciales (IMU) entregan información a una tasa de muestreo alta. Integrando la medición de la IMU es posible calcular la orientación y la posición del robot, pero con un error que crece en el tiempo de manera no acotada. Por otro lado, el sensor de posición Marvelmind y el magnetómetro entregan información a una tasa de muestreo menor, pero con un error acotado. Teniendo en cuenta el carácter complementario de estas clases de sensores, puede implementarse una estrategia de fusión de datos que permita obtener una estimación de la posición del robot a una alta tasa de muestreo y con un error acotado, lo que usualmente se denomina un sistema de navegación integrada [8].

Dada la información provista por la IMU, tomamos la aceleración horizontal [1] en terna del cuerpo $\{b\}$:

$$\hat{\mathrm{u}}_k = \begin{bmatrix} \ddot{x}^b_k & \ddot{y}^b_k \end{bmatrix}^T$$

A partir de la misma, es posible estimar la posición $(\hat{x}^i, \hat{y}^i)$ robot mediante las ecuaciones:

$$\hat{\mathrm{x}}_{k+1} = \begin{bmatrix} \mathbf{I} & T_s\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathrm{x}}_k + \begin{bmatrix} \mathbf{0} \\ T_s\hat{\mathbf{c}}_\psi \end{bmatrix} \hat{\mathrm{u}}_k \tag{12}$$

$$\hat{\mathbf{c}}_\psi = \begin{bmatrix} \cos\hat{\psi} & -\sin\hat{\psi} \\ \sin\hat{\psi} & \cos\hat{\psi} \end{bmatrix} \tag{13}$$

$$\hat{\mathrm{x}}_k = \begin{bmatrix} \hat{x}^i_k & \hat{y}^i_k & \hat{\dot{x}}^i_k & \hat{\dot{y}}^i_k \end{bmatrix}^T \tag{14}$$

$$\hat{\mathrm{y}}_k = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \hat{\mathrm{x}}_k \tag{15}$$

Con cada nueva medición del sistema Marvelmind y_k, se ejecuta la actualización de la estimación.

$$\hat{\mathrm{x}}_{k+1} = \hat{\mathrm{x}}_k + K_k \left(\mathrm{y}_k - \hat{\mathrm{y}}_k \right), \tag{16}$$

donde la matriz de ganancia K_k del estimador puede obtenerse mediante diversas estrategias, dependiendo de las hipótesis que se realicen sobre el sistema, siendo una de las más usuales las extensiones a modelos no lineales del filtro de Kalman. La estimación $\hat{\mathrm{x}}_k$ se utiliza para alimentar al sistema de control.

[1] Como hipótesis simplificativa se supone que el eje z de la IMU está perfectamente alineado con la vertical

V. Implementación del Sistema de Control utilizando ROS

ROS es un entorno de trabajo libre de código abierto para el desarrollo de aplicaciones con robots. A pesar de su nombre, ROS no es un sistema operativo en sí mismo, sino que provee a los sistemas operativos sobre los que se ejecuta de una capa estructurada de comunicaciones para vincular un conjunto heterogéneo de computadoras [9]. Al mismo tiempo, permite lidiar con las complejidades que aparecen en la construcción de robots debido a la variedad de hardware disponible. La mayor ventaja de ROS es que provee una interfaz común para el desarrollo de software para robots favoreciendo la reutilización, el intercambio de código y el prototipado rápido.

Un sistema ROS se compone de varios procesos que se ejecutan simultáneamente llamados *nodos* que se comunican entre ellos por medio de mensajes sobre enlaces punto-a-punto. Cada nodo es una aplicación de software con una función definida. Debido a su estructura modular, un nodo puede ser lanzado o detenido en cualquier momento, facilitando la depuración de cada uno de ellos. Existe un nodo maestro, denominado *roscore*, que es quien conoce qué nodos existen en la red, qué tópicos y servicios están disponibles, y provee a cada nuevo nodo en la red de la información necesaria para comunicarse con otros nodos. Adicionalmente, el *roscore* es quien maneja el reloj de ROS. Debido a la existencia del nodo maestro, ROS no es un sistema totalmente distribuido. En ROS2 ya no hay un nodo coordinador central, no obstante su utilización no está aún tan extendida.

Por otro lado, ROS permite la comunicación de forma natural entre nodos corriendo en una misma PC o en distintas PCs conectadas en la misma red. Sólo es necesario indicar en dónde se está ejecutando el *roscore* para que un nodo pueda comunicarse con éste y obtener información del grafo.

ROS provee dos mecanismos para comunicar distintos nodos: *tópicos* y *servicios*. Los tópicos siguen un esquema publicador/suscriptor, común en sistemas distribuidos, que permite una comunicación asincrónica unidireccional de uno a muchos. Cada vez que se publica un mensaje en un tópico se ejecutan las funciones de *callback* de los nodos suscriptos al mismo. A diferencia de los tópicos, los servicios son adecuados para la comunicación sincrónica tipo cliente/servidor entre nodos. Los servicios también tienen asociados funciones de *callback*. Tanto los servicios como los tópicos utilizan mensajes de ROS que son estructuras de datos fuertemente tipadas. A pesar que existe una gran cantidad de librerías de mensajes predefinidos, también se pueden crear nuevos ajustados a las necesidades.

Para visualizar la topología del sistema ROS se utiliza un grafo como el de la Fig. 3. Las burbujas y los arcos corresponden respectivamente a los nodos y tópicos que componen al sistema.

En general, el nombre de los tópicos a los que se suscribe y en donde un nodo publica se pueden configurar mediante parámetros. Los archivos *launchfile* se utilizan para automatizar el lanzamiento de nodos de ROS. Al lanzar un nodo se le pueden pasar parámetros. De esta manera, la topología del

ISBN: en trámite

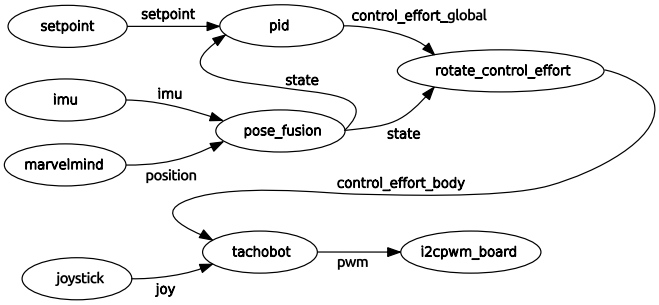

Figura 3. Grafo de ROS utilizado para el sistema de control de posición.

grafo de ROS se puede cambiar modificando el *launchfile*. Así como en otros entornos de desarrollo para aplicaciones de control, como Simulink o LabView, la interconexión entre distintos bloques funcionales se realiza mediante *drag-and-drop*, en ROS se hace por medio de los *launchfiles*.

Para este trabajo adoptamos ROS Kinetic debido a su masiva utilización y a que es una versión con soporte extendido en el tiempo. El nodo *roscore* y los nodos del sistema de control se ejecutan en una Raspberry Pi colocada en el TACHOBot, mientras que los nodos para visualización y manejo del joystick se ejecutan en dos PCs portátiles. Todas estas computadores se comunican por WiFi. Aquí se puede ver la versatilidad de ROS para interconectar un clúster heterogéneo de computadoras con distintos sistemas operativos.

Los nodos que componen al sistema de control implementado para el TACHOBot se pueden observar en la Fig. 3. Los nodos `imu`, `marvelmind`, `i2cpwm_board` y `joystick`, proveen de abstracción de hardware al interactuar con la IMU, el sistema de posicionamiento Marvelmind, el módulo generador de las señales de PWM y los comandos del joystick respectivamente. El procesamiento intermedio de los datos se realiza en los nodos `pose_fusion` y `rotate_control_effort`, que ejecutan el observador de la posición en terna global y la rotación de las acciones de control de la terna global a la terna del vehículo. Finalmente, el nodo `pid` ejecuta la ley de control en terna inercial, y `tachobot` convierte las acciones de control en las fuerzas $\{f_i\}$, y a partir de ellas obtiene las ocho señales de PWM aplicadas a los motores $\{M_j\}$. Es importante destacar que en esta implementación conviven nodos que están programados en C++ con otros escritos en Python.

Dada la modularidad de ROS, resulta sencillo incorporar nuevos nodos en el grafo, e incluso intercalar nodos entre los ya existentes, para realizar nuevas tareas.

VI. Ensayos experimentales

Para evaluar el comportamiento del sistema desarrollado se realizaron dos experimentos. En el primero (Fig. 4) se impuso una referencia fija de posición $x^i = y^i = \psi = 0$, y en el segundo (Fig. 5) se modificó la referencia en y^i para que el vehículo siga una rampa que comienza en el punto $y^i = 0$ y concluye en $y^i = 1\,\mathrm{m}$, mientras x^i y ψ se mantienen iguales a cero.

Para los ensayos, se ajustaron las ganancias del control PID de manera experimental. Esto implica encontrar un equilibrio entre las respuestas proporcional, derivativa e integral. Para el ángulo ψ, se implementó un control PD, ya que se observó que el término integral genera oscilaciones indeseadas. Para las posiciones (x^i, y^i) los controladores fueron diseñados con las mismas ganancias PID, debido a la simetría del sistema.

En la Fig. 4 (arriba), se puede observar que el controlador es capaz de seguir la referencia cero, con errores acotados de valor máximo aproximadamente igual a $0{,}5\,\mathrm{m}$ tanto en x^i como en y^i, y de $0{,}25\,\mathrm{rad}$ en ψ. La presencia de oscilaciones fue atenuada con la ganancia derivativa, que igualmente debe ser limitada, por el efecto amplificador sobre el ruido de medida. Por otro lado, se aprecia cómo el término integral es capaz de reducir paulatinamente el error estacionario, que se encuentra corregido a partir de los $40\,\mathrm{s}$ aproximadamente (apreciable en y^i). No obstante, se observó que este término aumenta las oscilaciones por lo que se limitó su valor. Se puede apreciar un mayor error estacionario en y^i con respecto al de x^i, a pesar de utilizar las mismas ganancias de control. Esto significa que existe un desbalanceo en el sistema que afecta su simetría. La causa con la cual asociamos esta diferencia, es desbalanceo en el empuje de motores que, por distintas razones físicas, no generan las mismas fuerzas para iguales valores de PWM.

En la Fig. 4 (abajo), se muestran las señales PWM enviadas a los motores durante ese mismo ensayo. Allí se observa que los motores que accionan en el eje *Y* $\{M_3, M_4, M_7, M_8\}$ tienen mayores excursiones que el resto, lo que evidencia el mayor esfuerzo que debe realizar el controlador para compensar el error estacionario en esta dirección. Esta observación se puede realizar dado que al imponer $\psi = 0$, el eje *Y* de las ternas local y global es coincidente. Sus valores se mantienen por debajo de su límite de saturación, lo que evidencia que se cuenta con margen para aumentar los esfuerzos de control y poder disminuir los errores obtenidos.

En la Fig. 5, se muestra la trayectoria en y^i que desarrolla el vehículo cuando se le impuso una referencia tipo rampa. Se observa que el sistema sigue la referencia con errores acotados. Las oscilaciones presentes tienen una amplitud máxima de aproximadamente $0{,}4\,\mathrm{m}$, correspondiente al sobrepico. Las mismas comienzan a disminuir hasta un valor similar al ensayo de referencia fija, hasta los $68\,\mathrm{s}$ cuando la referencia comienza a decrecer a $y^i = 0$.

VII. Conclusiones y Trabajo futuro

En este trabajo presentamos el TACHOBot, un robot útil para el estudio de problemas de control en sistemas ciberfísicos. Junto con éste presentamos el sistema de medición, y el control de posición y ángulo; y ensayamos la habilidad del TACHOBot para seguir referencias simples de posición y rechazar perturbaciones. Para cumplir con los objetivos de este trabajo se construyó y ajustó la mecánica del robot, los sensores, actuadores y el sistema de control, siendo suficiente utilizar una ley de control sencilla como el PID. Contar con esta plataforma nos habilita a probar distintas estrategias de

ISBN: en trámite

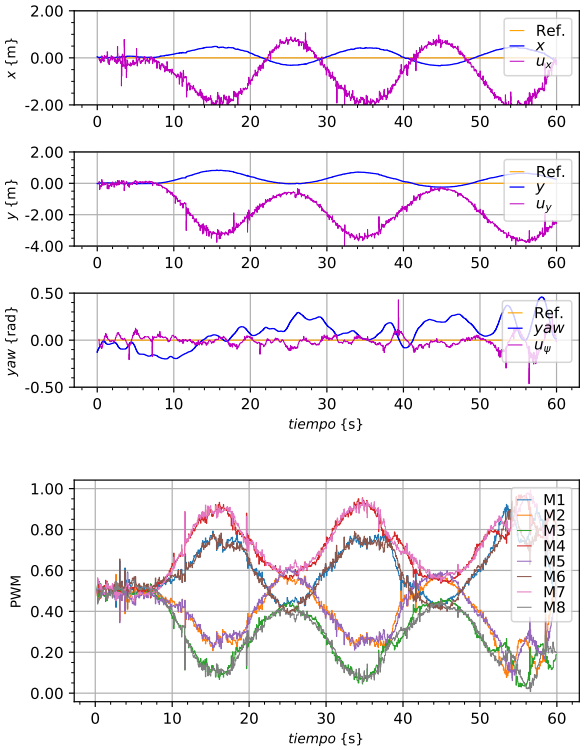

Figura 4. Respuesta a referencia de posición $\{x^i = 0, y^i = 0, \psi = 0\}$.

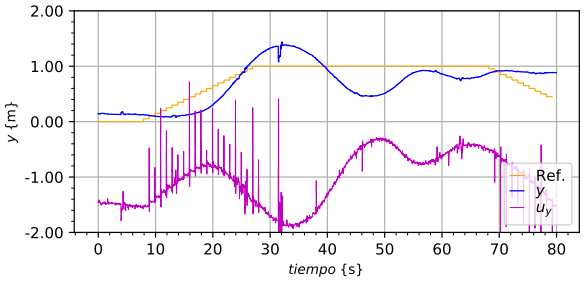

Figura 5. Respuesta a referencia tipo rampa en y^i de 0 a 1 m.

control más sofisticadas. Asimismo nos permitirá estudiar el efecto que tiene la variación del periodo de muestreo en el desempeño de controladores de sistemas ciber-físicos.

Como trabajo a futuro identificaremos el modelo dinámico del TACHOBot, porque de las pruebas realizadas se advierten discrepancias respecto al modelo teórico. El hecho de tener un primer control que estabilice al sistema será de gran utilidad para la etapa de identificación. De igual modo, realizaremos ensayos para relevar la curva de respuesta, de PWM a empuje, de los actuadores laterales.

A partir del análisis de las curvas obtenidas durante los ensayos se puede presuponer la existencia de retardos en el lazo. Por este motivo, se buscará identificar posibles fuentes de estos retardos y, en caso de ser necesario, se estudiarán estrategias para compensarlo.

El ajuste de las ganancias de los controladores se realizó enteramente por medio de ensayos sobre el sistema real. No obstante, el diseño de controladores basados en modelado y simulación es una técnica útil que se puede complementar con la anterior [1], [2]. Por este motivo, se desarrollará un simulador del robot para ensayar y ajustar estrategias de control.

Asimismo se estudiará el seguimiento de trayectorias que incluyan la variación del ángulo de ψ. Desde el punto de vista teórico, esto es posible sin realizar cambios en el sistema, debido a la linealización exacta realizada (sección II). Al no tratarse de una linealización local sobre el entorno de un ψ particular, estas trayectorias no deberían agregar complejidad al problema.

A partir de la experiencia ganada en la construcción del TACHOBot, se continuará con el armado de otros robots similares y con estos se estudiará el control de formaciones de robots como en [4]–[6].

Agradecimientos

Ezequiel Pecker Marcosig agradece a la Fundación Peruilh por el financiamiento de su beca doctoral. J. Francisco Presenza agradece a la Universidad de Buenos Aires por el financiamiento de su beca UBACyT de doctorado. Este trabajo fue financiado por el subsidio FONCYT PICT 2016-2016, de la Agencia Nacional de Promoción Científica y Tecnológica, Argentina y el subsidio ITBACyT DT13/2018.

Referencias

[1] E. Pecker Marcosig, J. I. Giribet, and R. Castro, "DEVS-OVER-ROS (DOVER): A framework for simulation-driven embedded control of robotic systems based on model continuity," in *2018 Winter Simulation Conference (WSC)*, 12 2018, pp. 1250–1261.

[2] R. Castro, E. Pecker-Marcosig, and J. I. Giribet, *Simulation model continuity for efficient development of embedded controllers in cyber-physical systems, in S. Mittal and A. Tolk (eds.) Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation to Support Intelligence, Adaptation and Autonomy*. Wiley, 2019.

[3] M. W. Regehr, A. B. Acikmese, A. Ahmed, M. Aung, K. Clark, P. MacNeal, J. Shields, G. Singh, R. Bailey, C. Bushnell *et al.*, "The formation control testbed," in *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, vol. 1. IEEE, 2004, pp. 557–564.

[4] M. Schlotterer and S. Theil, "Testbed for on-orbit servicing and formation flying dynamics emulation," in *AIAA Modeling and Simulation Technologies Conference*, 2010, p. 8108.

[5] H. Daitx, M. Schlotterer, J. Whidborne, and M. Sagliano, "Development of a combined attitude and position controller for a satellite simulator," in *67th International Astronautical Congress (IAC). International Astronautical Federation*, 2016.

[6] M. Schlotterer, E. Edlerman, F. Fumenti, P. Gurfil, S. Theil, and H. Zhang, "On-ground testing of autonomous guidance for multiple satellites in a cluster," in *Proceedings of the 8th International Workshop on Satellite Constellations and Formation Flying*, vol. 6, 2015.

[7] Marvelmind Robotics, "Precise (±2cm) indoor GPS for autonomous robots, drones and VR," https://marvelmind.com/.

[8] M. España, *Fundamentos de la Navegación Integrada*. AADECA, 2010.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *Proceedings of the Open-Source Software Workshop of the International Conference on Robotics and Automation (ICRA)*. May 12th-17th, Kobe, Japan, 1-6., 2009.