

Robot Cuadrúpedo: CheapSpot

David Ismael Fosado, Cesar Eduardo Ortega Torres y Jorge Daniel Ríos Arrañaga

*CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS, Universidad de Guadalajara, Guadalajara, Jalisco, México.

*Autor(es) de correspondencia. E-mail(s):
david.fosado5299@alumnos.udg.mx;

Autores contribuyentes: cesar.ortega9537@alumnos.udg.mx;
jorge.arranaga@academicos.udg.mx;

Resumen

Este artículo presenta el diseño, implementación y control de *CheapSpot*, un robot cuadrúpedo de bajo costo fabricado con impresión 3D y basado en una arquitectura de control jerárquica sobre hardware accesible (Raspberry Pi y Arduino). El sistema de control propuesto integra dos modos de operación distintos: un modo de locomoción teleoperada (vía PS4) complementado con un bucle de estabilización activa del cuerpo, el cual utiliza un controlador Fuzzy PID autosintonizado para corregir los errores de Roll y Pitch medidos por una IMU; y un modo de navegación autónoma que emplea un controlador de Servo Visual Basado en Imágenes (IBVS) para seguir un marcador AprilTag. Se detalla la implementación cinemática, incluyendo un generador de marcha (Gait Generator) basado en curvas de Bézier, y el mapeo del error visual a los parámetros de la locomoción. Los resultados experimentales validan ambos modos de operación, demostrando la eficacia del estabilizador Fuzzy PID en la reducción de oscilaciones y la capacidad del controlador IBVS para guiar al robot exitosamente hacia un objetivo.

Palabras Clave: Robot Cuadrúpedo, Control de Locomoción, Servo Visual Basado en Imágenes (IBVS), Controlador Fuzzy PID, Control Jerárquico, Cinemática Inversa.

1. Introducción

Los robots cuadrúpedos constituyen un área de gran relevancia dentro de la robótica moderna, destacando por su capacidad para desplazarse en terrenos irregulares y mantener la estabilidad dinámica durante la locomoción. En los últimos años, su popularidad ha incrementado significativamente tanto en la investigación académica como en el ámbito comercial. Ejemplos representativos incluyen a *Spot* y *BigDog*, desarrollados por Boston Dynamics, así como el *Mini Cheetah* del MIT [1].

A diferencia de los robots con ruedas, los cuadrúpedos presentan una estructura mecánica más compleja y un control cinemático y dinámico considerablemente más desafiante. Entre sus principales ventajas se encuentran la capacidad de adaptarse a terrenos irregulares y la estabilidad ante perturbaciones externas. No obstante, presentan limitaciones como una menor velocidad de desplazamiento, una mayor dificultad de construcción y control, además de un costo elevado de implementación [1].

Para lograr una locomoción robusta, se deben resolver múltiples desafíos de control. Primero, se requiere un modelo cinemático preciso (Sección 4) para traducir las trayectorias deseadas de los pies en ángulos articulares [2]. Segundo, se necesita un generador de patrones de marcha (Gait Generator) que coordine las cuatro patas en un ritmo estable, como el trote, a menudo usando trayectorias suaves como las curvas de Bézier (Sección 5) [3], [4].

Además de la marcha básica, este trabajo aborda dos desafíos adicionales: la estabilidad y la autonomía. Para la estabilidad en plataformas de bajo costo, que sufren de vibraciones y ruido en los sensores, los controladores PID clásicos son insuficientes. Por ello, se explora el uso de un controlador Fuzzy PID auto-sintonizado para mantener activamente la orientación del cuerpo [5]. Para la autonomía, se requiere que el robot perciba su entorno y navegue hacia un objetivo, una tarea comúnmente resuelta mediante el Servo Visual Basado en Imágenes (IBVS, por sus siglas en inglés, Image Based Visual Servoing) [6].

Este artículo presenta la implementación y validación de una arquitectura de control jerárquica en un robot cuadrúpedo de bajo costo (*CheapSpot*), capaz de operar en dos modos: (1) un modo teleoperado con estabilización activa mediante Fuzzy PID y (2) un modo de navegación autónoma guiado por IBVS. El sistema integra estas técnicas de control de alto nivel con un controlador cinemático completo basado en una Raspberry Pi y un Arduino.

El documento se organiza de la siguiente manera: Las secciones 2 a 5 presentan el marco teórico de los controladores Fuzzy PID, el IBVS, el modelo cinemático y los generadores de locomoción. La Sección 6 detalla la plataforma de hardware y la implementación de software de los dos modos de control. Finalmente, la Sección 7 presenta los resultados experimentales que validan la arquitectura propuesta, seguida de las conclusiones y el trabajo futuro.

2. Fuzzy PID

El controlador Proporcional-Integral-Derivativo (PID) es la estrategia de control más utilizada en la industria, estimándose su aplicación en más del 90 % de los bucles

de control debido a su simplicidad estructural, robustez y facilidad de implementación principalmente [5].

La principal limitación del PID clásico, radica en que sus ganancias (K_P, K_I, K_D) son fijas, sintonizadas para un único punto de operación. Cuando el sistema presenta comportamientos no lineales, incertidumbres paramétricas o sufre de perturbaciones externas, el control PID convencional baja en gran medida su efectividad, resultando en oscilaciones, sobre impulsos o tiempos de asentamientos largos [5].

Por ello han surgido los controladores PID basados en lógica difusa (Fuzzy PID). Esta arquitectura imita la heurística de un operador humano que observa el error del sistema y decide como modificar las ganancias. El mecanismo de autoajuste (self-tuning) propuesto en [5] opera en tres fases principales:

1) **Fuzificación:** El error y el cambio de error se transforman en variables lingüísticas (por ejemplo, “error positivo grande.” “error cero”) mediante funciones de membresía.

2) **Inferencia:** Un motor de inferencia evalúa una base de reglas del tipo “*SI-ENTONCES*”, que usa el conocimiento experto en el proceso (por ejemplo, “*Si el error es positivo pequeño Y el cambio de error es cero, ENTONCES aumentar K_I* ”)

3) **Desfuzificación:** La salida resultante se convierte nuevamente en un valor preciso, utilizado para actualizar las ganancias del PID.

Este ciclo se ejecuta continuamente, permitiendo al controlador ajustar sus parámetros. Como resultado, los controladores Fuzzy PID presentan un mejor rendimiento que los controladores PID clásicos [5].

3. Control Visual Basado en Imágenes (IBVS)

El Visual Servoing es un pilar fundamental en la robótica moderna, que consiste en utilizar la información de una cámara dentro del bucle de control para guiar el movimiento de un robot [6].

El enfoque IBVS define la tarea de control y el error directamente en el plano 2D de la imagen. En esta arquitectura, un vector de características visuales \mathbf{s} (como las coordenadas de puntos, centroides o líneas) se extrae de la imagen actual. El objetivo de control es que este vector \mathbf{s} converja a un vector de características deseadas \mathbf{s}_d , que representa la pose objetivo del robot. La ley de control se diseña para regular este error en píxeles, definido como:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}_d.$$

donde el vector de características visuales se expresa como:

$$\mathbf{s} = \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

siendo x y y las coordenadas normalizadas en el plano de la imagen. Estas coordenadas se obtienen a partir de las coordenadas en píxeles (u, v) mediante la relación:

$$x = \frac{u - u_0}{f_u}, \quad y = \frac{v - v_0}{f_v}, \quad (2)$$

donde f_u , f_v , u_0 y v_0 son parámetros intrínsecos de la cámara.

3.1. Modelo de Cámara

Para que el controlador IBVS pueda relacionar los movimientos del robot con las variaciones observadas en la imagen, es necesario modelar matemáticamente el sistema de cámara. Este modelo permite proyectar puntos del espacio tridimensional (X, Y, Z) sobre el plano de imagen (u, v), mediante los parámetros intrínsecos y extrínsecos de la cámara.

3.2. Parámetros Intrínsecos

Los parámetros intrínsecos describen las propiedades ópticas y geométricas internas de la cámara, como la distancia focal y el punto principal. Estos parámetros se agrupan en la matriz intrínseca \mathbf{K} , definida como:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

donde f_u y f_v son las distancias focales expresadas en píxeles, y (u_0, v_0) representa el centro óptico o punto principal de la imagen. Esta matriz permite convertir coordenadas normalizadas del plano de cámara a coordenadas de píxel.

3.3. Parámetros Extrínsecos

Los parámetros extrínsecos describen la posición y orientación de la cámara respecto a un marco de referencia externo, en este caso el cuerpo del robot. Están conformados por una matriz de rotación ${}^c\mathbf{R}_0$ y un vector de traslación ${}^c\mathbf{t}_0$, que definen la transformación entre el sistema de coordenadas de la cámara y la de la imagen. Matemáticamente, esta relación se expresa como:

$${}^c\mathbf{X}_i = {}^c\mathbf{R}_0 {}^0\mathbf{X}_i + {}^c\mathbf{t}_0 \quad (4)$$

donde ${}^c\mathbf{X}_i$ representa las coordenadas de un punto en el marco de la cámara, y ${}^0\mathbf{X}_i$ las coordenadas del mismo punto expresadas en el marco de la imagen.

Este modelo de cámara, compuesto por las matrices \mathbf{K} y $[{}^c\mathbf{R}|{}^c\mathbf{t}]$, constituye la base para establecer la relación entre los cambios en la imagen y las velocidades del robot dentro del esquema de control IBVS.

3.4. Matriz de Interacción y Ley de Control

El componente matemático central del IBVS es la matriz de interacción \mathbf{L}_s , que relaciona la variación temporal de las características visuales con las velocidades de la cámara \mathbf{v}_c . Para el caso de un punto proyectado en la imagen con coordenadas (u, v) y profundidad Z , la cual se obtiene a partir de ${}^c\mathbf{X}_i$ [6], la matriz de interacción se expresa como:

$$\mathbf{L}_s = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix}. \quad (5)$$

La ley de control del IBVS se diseña de manera que el error \mathbf{e} tienda a cero de forma exponencial. Una forma común de definirla es:

$$\mathbf{v}_c = -\lambda \mathbf{L}_s^+ \mathbf{e}, \quad (6)$$

donde $\lambda > 0$ es una ganancia proporcional y \mathbf{L}_s^+ representa la pseudo-inversa de Moore-Penrose de la matriz de interacción [6].

En la implementación de este trabajo, las características visuales corresponden a los vértices de un marcador *AprilTag* de dimensiones conocidas (6 cm de lado), lo que permite estimar la profundidad Z a partir de su tamaño proyectado en la imagen.

4. Cinemática del Robot Cuadrúpedo

4.1. Cinematica directa Full Body

El robot cuadrúpedo es un sistema robótico que consiste de un cuerpo rígido y cuatro piernas con tres grados de libertad (cada pierna tiene la misma estructura). Cada eslabón esta unido por una articulación rotacional. En modelo del robot se encuentra en la Figura 1. Los parámetros del robot se muestran en la Tabla 1. El modelo cinemático del cuerpo completo del robot se basa en el trabajo propuesto por [2]. En donde, a través de la matriz de transformación (7), se puede determinar la posición y orientación del centro del cuerpo del robot en el espacio de trabajo.

$$\mathbf{T}_M = \mathbf{R}_{xyz} \begin{bmatrix} 1 & 0 & 0 & x_m \\ 0 & 1 & 0 & y_m \\ 0 & 0 & 1 & z_m \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

donde:

$$\begin{aligned} \mathbf{R}_x &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\omega) & -\sin(\omega) & 0 \\ 0 & \sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}_y &= \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}_z &= \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}_{xyz} &= \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \end{aligned}$$

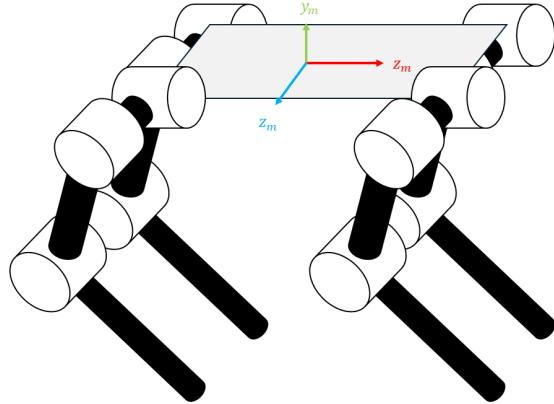


Figura 1 Modelo físico del robot cuadrúpedo

Tabla 1 Parámetros del robot cuadrúpedo

Dimensiones físicas	
Largo del robot	$h_x = 0,18 \text{ m}$
Ancho del robot	$h_y = 0,08 \text{ m}$
Largo del la articulación del hombro	$l_1 = 0,04 \text{ m}$
Largo del la articulación del codo	$l_2 = 0,115 \text{ m}$
Largo del la articulación de la muñeca	$l_3 = 0,125 \text{ m}$

Sistema de Coordenadas	
Sistema de coordenadas del centro de masas del robot	$[x_m, y_m, z_m]$
Sistema de coordenadas del hombro	$[x_0, y_0, z_0]$
Sistema de coordenadas del hombro desplazado	$[x_1, y_1, z_1]$
Sistema de coordenadas del codo	$[x_2, y_2, z_2]$
Sistema de coordenadas de la muñeca	$[x_3, y_3, z_3]$
Sistema de coordenadas del punto final de la pata	$[x_4, y_4, z_4]$

Variables	
Angulo Yaw	ϕ
Angulo Pitch	ψ
Angulo Roll	ω
Angulo del hombro	θ_1
Angulo del codo	θ_2
Angulo de la muñeca	θ_3

La ecuación que relaciona el sistema de coordenadas del centro del robot y el sistema de coordenadas del hombro de cada pata esta dado por las matrices de transformación presentadas en la Ecuación (8), (9), (10) y (11).

$$\mathbf{T}_{FR} = \mathbf{T}_M \begin{bmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) & h_x/2 \\ 0 & 1 & 0 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) & -h_y/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$\mathbf{T}_{FL} = \mathbf{T}_M \begin{bmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) & h_x/2 \\ 0 & 1 & 0 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) & h_y/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$\mathbf{T}_{BR} = \mathbf{T}_M \begin{bmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) & -h_x/2 \\ 0 & 1 & 0 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) & -h_y/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$\mathbf{T}_{BL} = \mathbf{T}_M \begin{bmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) & -h_x/2 \\ 0 & 1 & 0 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) & h_y/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

4.2. Cinemática directa de la pata

En el análisis de la cinemática directa, se cuenta con la información de la posiciones articulares para encontrar la posición y orientación final del cuerpo del robot. El modelo cinemático del robot cuadrúpedo se basa en el trabajo propuesto por [2]. En donde se analiza cada pierna como un brazo manipulador de tres grados de libertad (hombro, codo y muñeca). Ya que todas las piernas comparten la misma estructura es suficiente con el análisis de una sola de ellas.

En la Figura 2 se muestra el sistema de coordenadas y posición angular de la pierna derecha. Los parámetros de la Tabla 2.

Tabla 2 Parámetros Denavit–Hartenberg del robot

Link	α	a	d	θ
0	0	l_1	0	θ_1
1	$-\pi/2$	0	0	$-\pi/2$
2	0	l_2	0	θ_2
3	0	l_3	0	θ_3

Resultando en la matriz de la cinemática directa 0T_4 dada en (12). Los elementos de la matriz de la cinemática directa de una pierna se encuentran en la Tabla 3.

$${}^0\mathbf{T}_4 = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \quad (12)$$

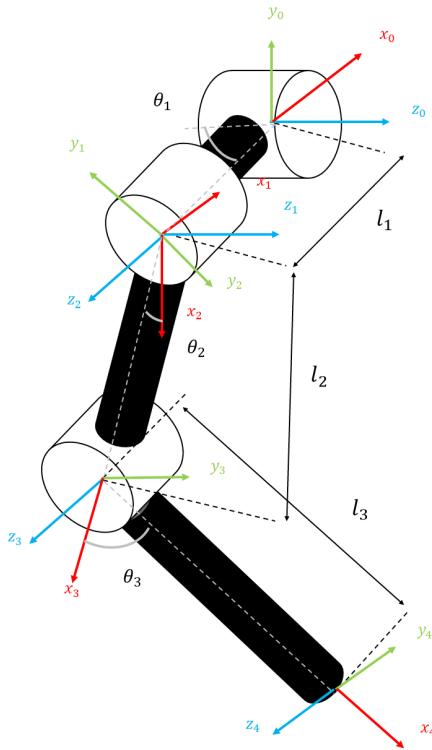


Figura 2 Sistema de coordenadas de pierna derecha

Tabla 3 Elementos de la Matriz de la cinemática directa ${}^0\mathbf{T}_4$

Elemento	Ecuación
m_{11}	$\cos(\theta_2) \cos(\theta_3) \sin(\theta_1) - \sin(\theta_1) \sin(\theta_2) \sin(\theta_3)$
m_{12}	$-\cos(\theta_2) \sin(\theta_1) \sin(\theta_3) - \cos(\theta_3) \sin(\theta_1) \sin(\theta_2)$
m_{13}	$-\cos(\theta_1)$
m_{14}	$l_2 \cos(\theta_2) \sin(\theta_1) - l_1 \cos(\theta_1) + l_2 \cos(\theta_2) \cos(\theta_3) \sin(\theta_1) - l_3 \cos(\theta_1) \sin(\theta_2) \sin(\theta_1)$
m_{21}	$\cos(\theta_1) \sin(\theta_2) \sin(\theta_3) - \cos(\theta_1) \cos(\theta_2) \cos(\theta_3)$
m_{22}	$\cos(\theta_1) \cos(\theta_2) \sin(\theta_3) + \cos(\theta_1) \cos(\theta_3) \sin(\theta_2)$
m_{23}	$-\sin(\theta_1)$
m_{24}	$l_3 \cos(\theta_1) \sin(\theta_2) \sin(\theta_3) - l_2 \cos(\theta_1) \cos(\theta_2) - l_3 \cos(\theta_1) \cos(\theta_2) \cos(\theta_3) - l_1 \sin(\theta_1)$
m_{31}	$\cos(\theta_2) \sin(\theta_3) + \cos(\theta_3) \sin(\theta_2)$
m_{32}	$\cos(\theta_2) \cos(\theta_3) - \sin(\theta_2) \sin(\theta_3)$
m_{33}	0
m_{34}	$l_2 \sin(\theta_2) + l_3 \cos(\theta_2) \sin(\theta_3) + l_3 \cos(\theta_3) \sin(\theta_2)$
m_{41}	0
m_{42}	0
m_{43}	0
m_{44}	1

4.3. Cinemática inversa

En el análisis de la cinemática inversa, se conoce la posición final deseada del pie del robot, a partir de la cual se determinan los ángulos articulares necesarios para alcanzar dicha posición.

Los ángulos articulares θ_1 , θ_2 y θ_3 se obtienen a partir de (13), (14) y (15) dada una posición deseada del pie (x, y, z) . Las ecuaciones utilizadas se basan directamente en [2].

$$\begin{aligned} \theta_1 &= -\arctan 2(-y, x) \\ &- \arctan 2(\sqrt{x^2 + y^2 - l_1^2}, -l_1) \end{aligned} \quad (13)$$

$$\begin{aligned} \theta_2 &= \arctan 2(z, \sqrt{x^2 + y^2 - l_1^2}) \\ &- \arctan 2(l_3 \sin \theta_3, l_2 + l_3 \cos \theta_3) \end{aligned} \quad (14)$$

$$\theta_3 = \arctan 2(-\sqrt{1 - D^2}, D) \quad (15)$$

$$D = \frac{x^2 + y^2 + (z - l_1)^2 - l_2^2 - l_3^2}{2l_2l_3}$$

5. Locomoción y Generación de Trayectorias

5.1. Arquitectura de Control de Locomoción

El sistema de locomoción de un robot cuadrúpedo tiene como objetivo generar y coordinar los movimientos de las extremidades para producir un desplazamiento estable y fluido. En la literatura, los controladores más avanzados, como los implementados en el robot MIT Cheetah [3], utilizan arquitecturas jerárquicas que combinan control de fuerza e impedancia, junto con planificación dinámica.

En este proyecto, se adopta una arquitectura inspirada en dicho enfoque, pero adaptada a un sistema con servomotores controlados por posición, como el utilizado en el robot Spot Mini Mini [4]. Dado que estos actuadores no permiten medir ni controlar el torque, se emplea un esquema completamente cinemático en el que cada pata sigue trayectorias predefinidas y las transiciones entre fases se gestionan mediante una máquina de estados finita (FSM) controlada por tiempo.

El controlador de locomoción se divide en dos niveles principales:

- **Controlador de Alto Nivel (Gait Generator):** Define el tipo de marcha (por ejemplo, trote), la frecuencia de paso, la longitud de zancada y la altura del cuerpo. Además, sincroniza las fases de cada pata para mantener la estabilidad del robot.
- **Controlador de Bajo Nivel:** Genera las trayectorias cartesianas de cada pata (en coordenadas x, y, z) según la fase actual (apoyo u oscilación) y calcula los ángulos articulares correspondientes mediante la cinemática inversa.

5.2. Control de Fases y Generador de Trayectorias

Cada pata del robot alterna entre dos fases principales:

- **Fase de Apoyo (Stance):** El pie se mantiene en contacto con el suelo mientras el cuerpo avanza. En esta fase, la posición del pie se mantiene aproximadamente fija o se interpola linealmente con el desplazamiento del cuerpo.
- **Fase de Oscilación (Swing):** El pie se mueve hacia la siguiente posición de apoyo. En este proyecto, se genera una trayectoria suave en el plano $x-z$ utilizando curvas cúbicas de Bézier, que garantizan continuidad en posición, velocidad y aceleración.

La trayectoria del pie durante la fase de oscilación se define como:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3, \quad t \in [0, 1] \quad (16)$$

donde los puntos de control P_0 y P_3 representan el inicio y el final del paso, mientras que P_1 y P_2 determinan la altura y la curvatura de la trayectoria. De esta forma, es posible ajustar parámetros como la altura del paso o la longitud de zancada modificando dichos puntos.

5.3. Generador de Patrón de Marcha (Gait Generator)

El generador de patrón de marcha es el encargado de definir la secuencia temporal de apoyo y oscilación de cada pata. En este trabajo se implementó el patrón de marcha tipo *trote* (trot), debido a su equilibrio entre estabilidad y velocidad.

En este patrón, las patas se mueven en pares diagonales (frontal derecha/trasera izquierda y frontal izquierda/trasera derecha), manteniendo siempre dos puntos de apoyo en el suelo. Cada pata i se asocia a una variable de fase $\phi_i \in [0, 1]$ que representa su posición relativa dentro del ciclo de marcha:

$$\phi_{FR} = \phi_{HL}, \quad \phi_{FL} = \phi_{HR} = (\phi_{FR} + 0,5) \bmod 1 \quad (17)$$

Las transiciones entre las fases de apoyo y oscilación se determinan únicamente mediante la fase ϕ_i , sin necesidad de sensores de contacto. Este enfoque permite mantener una sincronización precisa entre las patas, facilitando un desplazamiento continuo y estable del robot.

6. Implementación

6.1. Arquitectura del Sistema

La plataforma de hardware se basa en el robot *SpotMiniMini* [4], el cual a su vez se inspira en el robot *Spot* desarrollado por *Boston Dynamics* [1]. El chasis del robot cuadrúpedo fue fabricado mediante impresión 3D utilizando material ABS, adecuado para prototipado por su resistencia mecánica y tolerancia a altas temperaturas.

El robot cuenta con 12 servomotores *PDI-HV5523MG*, seleccionados por su bajo costo, alta velocidad de respuesta y capacidad de carga, características apropiadas para la locomoción de un cuadrúpedo compacto.

Para la percepción visual, el sistema incorpora una cámara *Logitech C920*, montada en la parte frontal del chasis, utilizada para la adquisición de imágenes en los algoritmos de control visual basado en imágenes.

El control del robot está distribuido en dos niveles:

- **Raspberry Pi 4 (4 GB de RAM):** encargada del procesamiento de alto nivel, incluyendo la planificación de locomoción, el control de marcha y la visión por computadora.
- **Arduino UNO:** responsable del control de bajo nivel de los servomotores.

El Arduino gestiona las señales PWM mediante el módulo *PCA9685*, que permite generar hasta 16 canales PWM independientes. La comunicación entre el Arduino y el PCA9685 se realiza mediante el protocolo *I2C*, mientras que la comunicación entre la Raspberry Pi y el Arduino se lleva a cabo a través del puerto serial USB, a través del cual se envían los ángulos articulares calculados por el controlador de locomoción.

El sistema cuenta con una unidad de medición inercial (*IMU*) *MPU-9250*, utilizada para estimar la orientación y aceleración del robot, comunicándose con la Raspberry Pi también mediante el bus *I2C*.

La alimentación del sistema se realiza mediante una batería *LiPo 2S* (7.4 V), la cual suministra energía tanto a los servomotores como a la electrónica de control. Se utiliza un regulador de voltaje *LM2596* para obtener una salida estable de 5 V, que alimenta directamente a la Raspberry Pi. A su vez, la Raspberry Pi proporciona la alimentación al Arduino UNO.

6.2. Diagrama de Flujo de Control

La Figura 3 ilustra el diagrama de bloques de la arquitectura de control implementada. El sistema opera como un bucle cerrado donde la información fluye desde la percepción visual (pixeles capturados por la cámara) hasta los ángulos de las articulaciones (servos).

El flujo de control sigue estos pasos secuenciales en cada ciclo:

1) **Cámara:** El bucle inicia con el módulo de percepción visual. Este bloque captura la imagen mediante la cámara y detecta el AprilTag en la escena. A partir de esto, se extrae el vector de características visuales actuales, denotado como \mathbf{s} .

2) **Controlador IBVS:** El vector de características actuales \mathbf{s} se compara con el vector de características deseadas \mathbf{s}_d . A partir de esta comparación, el controlador IBVS calcula las velocidades de la cámara, las cuales se transforman al marco de referencia del robot cuadrúpedo, generando el vector de velocidades \mathbf{v}_p .

3) **Conversión Cinemática:** Las velocidades \mathbf{v}_p se transforman mediante una matriz de ganancias a parámetros para el Gait Generator \mathbf{u}_g .

4) **Gait Generator:** Este modulo recibe los parámetros del Gait Generator \mathbf{u}_g como entrada y calcular la posición cartesiana deseada de cada pie \mathbf{p}_d .

5) **Cinemática Inversa:** La cinemática inversa recibe la posición deseada de cada pie \mathbf{p}_d y resuelve los ángulos articulares correspondientes, obteniendo el vector $\boldsymbol{\theta}$.

6) **Robot Cuadrúpedo:** El robot cuadrúpedo ejecuta ángulos articulares $\boldsymbol{\theta}$ modificando su pose en el espacio, denotado como \mathbf{x}_p . Este cambio en la pose altera la posición de la cámara, cerrando el ciclo.

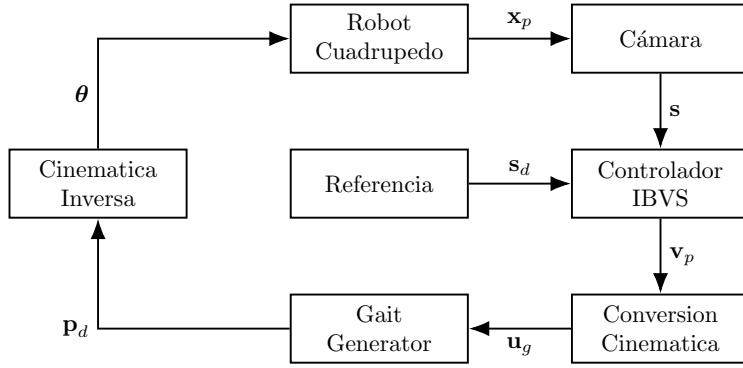


Figura 3 Diagrama general del sistema de control visual y locomoción del robot cuadrúpedo.

6.3. Implementación del Control Visual IBVS

La implementación del control IBVS sigue la arquitectura teórica descrita en la Sección 3, adaptada al sistema de locomoción del robot cuadrúpedo.

6.3.1. Calibración de la cámara

Para la obtención precisa de las características visuales, se requiere una cámara previamente calibrada. En este proyecto se utilizó una cámara *Logitech C920*, cuya calibración se realizó empleando la librería OpenCV en *Python*. Dicha calibración permitió obtener los parámetros intrínsecos contenidos en la matriz \mathbf{K} definida en la ecuación (3), así como los parámetros extrínsecos $^c\mathbf{R}_0$ y $^c\mathbf{t}_0$, que describen la posición y orientación de la cámara respecto a la imagen. Estos parámetros permiten transformar las velocidades calculadas en el marco de la cámara al marco de la imagen.

6.3.2. Extracción de características visuales

La percepción visual se realiza mediante la cámara *Logitech C920* montada en la parte frontal del robot. El sistema de visión emplea el paquete *AprilTag* en *Python*, que procesa cada fotograma y devuelve las coordenadas de las esquinas del marcador detectado (u_i, v_i) , junto con su orientación relativa. El vector de características visuales actual \mathbf{s}_i se define como:

$$\mathbf{s}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix},$$

mientras que el vector deseado \mathbf{s}_d corresponde a la posición del marcador cuando el robot se encuentra alineado con el objetivo. A partir del tamaño proyectado del marcador (*AprilTag* de 13 cm por lado) se estima la profundidad Z , necesaria para el cálculo de la matriz de interacción descrita en la Ecuación (5).

6.3.3. Implementación del controlador IBVS

El controlador se implementa siguiendo la ley de control definida en la Ecuación (6), donde el error visual $\mathbf{e} = \mathbf{s} - \mathbf{s}_d$ se regula mediante una ganancia proporcional λ .

6.3.4. Transformación al marco del robot

Las velocidades calculadas en el marco de la cámara se transforman al marco base del robot mediante la siguiente matriz homogénea:

$$\mathbf{V} = \begin{bmatrix} {}^c\mathbf{R}_p & [{}^c\mathbf{t}_p]_{\times} {}^c\mathbf{R}_p \\ \mathbf{0} & {}^c\mathbf{R}_p \end{bmatrix},$$

donde $[{}^c\mathbf{t}_p]_{\times}$ es la matriz antisimétrica asociada al vector de traslación ${}^c\mathbf{t}_p = [t_x, t_y, t_z]^T$, definida como:

$$[{}^c\mathbf{t}_p]_{\times} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix},$$

con $t_x = 0,07$ m, $t_y = -0,145$ m y $t_z = 0,0$ m.

De este modo, la ley de control puede expresarse como:

$$\mathbf{v}_p = -\lambda (\mathbf{L}_s \mathbf{V})^+ \mathbf{e},$$

donde \mathbf{v}_p representa las velocidades lineales y angulares del cuerpo del robot en su marco base, constituyendo la salida del controlador IBVS. Este vector es enviado al módulo de locomoción encargado de generar la marcha correspondiente.

6.3.5. Integración con la locomoción

Las componentes de \mathbf{v}_p se emplean para ajustar los parámetros del generador de marcha del robot cuadrúpedo, definidos por el vector:

$$\mathbf{u}_g = \begin{bmatrix} L \\ L_F \\ H_p \\ \omega \\ \psi \\ \phi \end{bmatrix}, \quad (18)$$

donde:

- L es la longitud de zancada, ajustada según la magnitud de v_x ;
- L_F es la longitud de zancada lateral, ajustada según la magnitud de v_y ;
- H_p es la altura al centro de masas del robot, ajustada según la magnitud de v_z ;
- ω es el ángulo de orientación *roll*;
- ψ es el ángulo de orientación *pitch*;
- ϕ es el ángulo de orientación *yaw*.

Este mapeo se implementa como una relación proporcional simple, donde un conjunto de ganancias k_i transforma las velocidades deseadas del cuerpo en los parámetros específicos de la marcha:

- $L = k_1 v_x$.
- $L_F = k_2 v_y$.
- $H_p = k_3 v_z$.

- $\omega = k_4 w_x$.
- $\psi = k_5 w_y$.
- $\phi = k_6 w_z$.

De este modo, el controlador IBVS y el generador de marcha conforman un lazo de control visual cerrado, en el cual el movimiento del robot reduce progresivamente el error visual \mathbf{e} hasta alcanzar la referencia deseada \mathbf{s}_d .

6.4. Implementación de la Locomoción

La implementación del sistema de locomoción se realizó tomando como referencia la arquitectura descrita en la Sección 5. En esta etapa, se programaron los módulos de generación de marcha (*Gait Generator*) y de generación de trayectorias (*Trajectory Generator*), integrándolos con el hardware de control y comunicación del robot.

6.4.1. Configuración del Gait Generator

Se implementó el patrón de marcha tipo *trote*, manteniendo la sincronización de las patas diagonales según la relación de fases definida en la ecuación (17). Cada ciclo de marcha se parametrizó mediante la velocidad de paso, el periodo de oscilación, la altura de paso, y el paso de integración adecuados, los cuales determinan la velocidad y estabilidad del movimiento. Los valores finales utilizados durante las pruebas se muestran en la Tabla 4.

Tabla 4 Parámetros empleados para la marcha tipo trote.

Parámetro	Valor
Velocidad de paso	0.0001
Periodo de Oscilación	0.25 s
Altura de paso	0.03 m
Altura del cuerpo (H_p)	0.19 m
Paso de integración (T_s)	0.04 s

6.4.2. Generación de Trayectorias de las Patas

Las trayectorias cartesianas de cada pata se calcularon de acuerdo con el modelo descrito previamente, empleando las mismas curvas cúbicas de Bézier definidas en la ecuación (16). Durante la fase de oscilación, el pie describe una trayectoria suave entre el punto de despegue y el de aterrizaje, mientras que en la fase de apoyo se mantiene fija con respecto al suelo. Los parámetros del generador de trayectorias se actualizan dinámicamente según las velocidades lineales y angulares proporcionadas por el controlador IBVS.

6.4.3. Implementación Cinemática Inversa

La cinemática inversa presentada en la Sección 4 constituye la última etapa del ciclo de control del robot. En esta fase, se recibe la posición cartesiana deseada de cada pata \mathbf{p}_d , a partir de la cual se calculan los ángulos articulares correspondientes mediante las ecuaciones (13), (14) y (15), asociadas a las articulaciones del hombro, codo y muñeca, respectivamente. Los valores obtenidos son enviados a la placa Arduino UNO encargado de accionar los servomotores, completando así el lazo de control de locomoción.

6.5. Implementación del Control Teleoperado

El modo de control teleoperado permite al usuario controlar la locomoción del robot mediante un control de PS4, mientras un bucle de estabilidad activo corrige automáticamente la orientación del cuerpo.

Este modo se implementa mediante dos bucles de control que operan en paralelo: el Bucle de Locomoción (comandado por el PS4) y el Bucle de Estabilidad (comandado por el Fuzzy PID). En la Figura 4 se muestra un diagrama de la implementación del control teleoperado, donde se puede observar un esquema similar al visto en la Figura 3.

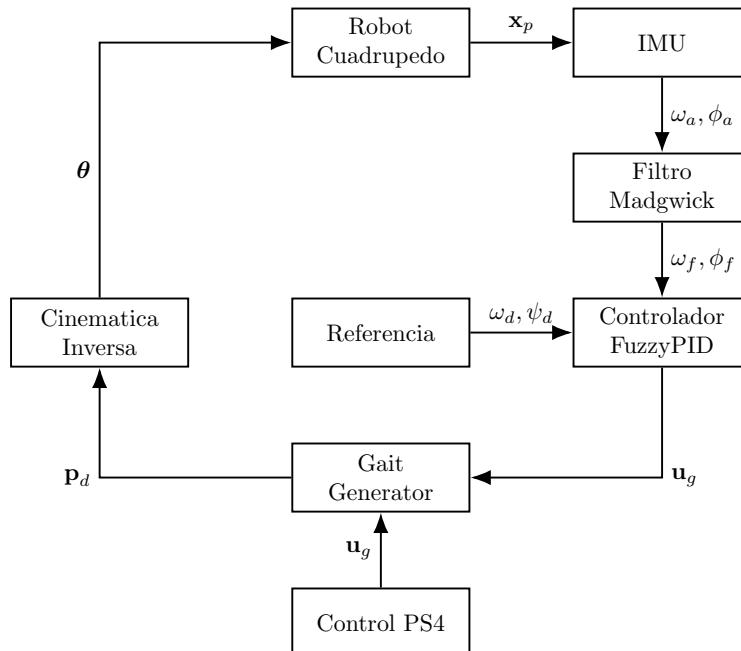


Figura 4 Diagrama general del sistema de control teleoperado del robot cuadrúpedo.

6.5.1. Bucle de Locomoción (Controlado por PS4)

El usuario comanda los parámetros \mathbf{u}_g de la Ecuación (18) al Gait Generator, para que el robot avance o retroceda, se mueva lateralmente o sobre si mismo, se modifique la orientación del robot usando los joysticks del control de PS4. El sistema está diseñado para que los comandos de orientación manuales del PS4 (si el usuario desea inclinar el robot) tengan **prioridad**, desactivando temporalmente la corrección del Bucle de Estabilidad para permitir el control total del usuario.

6.5.2. Bucle de Estabilidad (Controlador Fuzzy PID)

En paralelo, un bucle de control utiliza la IMU (MPU-9250) y el controlador Fuzzy PID (descrito teóricamente en la Sección 2) para mantener el cuerpo del robot nivelado.

1. Estimación de Orientación (Filtro Madgwick): El controlador de estabilidad opera sobre los ejes de Roll (ω) y Pitch (ψ). La orientación absoluta del robot no se puede leer directamente, sino que debe ser *estimada* a partir de los datos ruidosos del acelerómetro (que mide la gravedad) y el giroscopio (que mide la velocidad angular pero sufre de deriva).

Para fusionar estos dos sensores, se utiliza el filtro AHRS (Attitude and Heading Reference System) de Madgwick. Este filtro, implementado a través de la librería `ahrs`, utiliza un cuaternión para representar la orientación y un gradiente descendente basado en el acelerómetro para corregir la deriva del giroscopio.

Las lecturas calibradas del giroscopio $\mathbf{g} = [g_x, g_y, g_z]$ y del acelerómetro $\mathbf{a} = [a_x, a_y, a_z]$ se introducen en la función de actualización del filtro en cada paso de tiempo Δt :

$$\mathbf{q}_k = \mathbf{q}_{k-1} + \dot{\mathbf{q}}_k \cdot \Delta t$$

Donde \mathbf{q}_k es el cuaternión de orientación actualizado y $\dot{\mathbf{q}}_k$ es la derivada del cuaternión, calculada por el algoritmo de Madgwick para minimizar el error entre la gravedad medida y la gravedad estimada.

El resultado \mathbf{q} (un cuaternión) representa la orientación 3D completa del robot.

2. Cálculo del Error (Entradas del Controlador): Para el controlador de postura, solo nos interesa la inclinación relativa (Roll y Pitch). El cuaternión \mathbf{q} del filtro Madgwick se convierte a ángulos de Euler (Roll ω_f , Pitch ψ_f , Yaw ϕ_f).

El error e y el cambio en el error Δe se calculan entonces como:

$$e_\omega(k) = \omega_d - \omega_f(k) \quad , \quad \Delta e_\omega(k) = e_\omega(k) - e_\omega(k-1)$$

$$e_\psi(k) = \psi_d - \psi_f(k) \quad , \quad \Delta e_\psi(k) = e_\psi(k) - e_\psi(k-1)$$

Donde la orientación deseada (ω_d, ψ_d) se establece en 0° . Finalmente, estas entradas se convierten a sus valores absolutos $|e|$ y $|\Delta e|$ para ser usadas por los sistemas difusos.

3. Arquitectura Híbrida de Sintonización Difusa: Se implementó un sistema híbrido donde cada ganancia PID es manejada por un sistema difuso independiente con una lógica especializada, basado en las entradas simétricas $|e|$ y $|\Delta e|$. La logica completa la desicion, incluyendo la plnificación directa de K_p para evitar saturacion y la sintonizacion incremental de K_i y K_d , se detalla en el Algoritmo 1.

Algoritmo 1 Lógica Híbrida de Sintonización de Ganancias (Fuzzy PID)

Entrada: Error actual $e(k)$, Error anterior $e(k - 1)$

Salida: Ganancias actualizadas K_p, K_i, K_d

- 1: $|e| \leftarrow \text{abs}(e(k))$
 - 2: $|\Delta e| \leftarrow \text{abs}(e(k) - e(k - 1))$
 - 3: **1. Planificador de Ganancia Proporcional (K_p)**
 si $|e| > 0,3$ **entonces** ▷ Error Grande
 4: $K_p \leftarrow 1,0$ ▷ Baja ganancia para evitar oscilaciones
 si no si $|e| > 0,02$ **entonces** ▷ Error Medio
 5: $K_p \leftarrow 3,0$ ▷ Alta rigidez para corrección
 si no ▷ Ruido / Zona Muerta
 6: $K_p \leftarrow 1,8$ ▷ Ganancia base
 - 7: **2. Sintonizador de Ganancia Integral (K_i)**
 si $|e|$ es Grande **entonces** ▷ Reducir efecto integral
 8: $\Delta K_i \leftarrow \text{Negativo}$
 si no si $|e|$ es Medio **entonces** ▷ Eliminar error estacionario
 9: $\Delta K_i \leftarrow \text{Positivo}$
 si no ▷ Ruido
 10: $\Delta K_i \leftarrow 0$
 - 11: $K_i \leftarrow K_i + \Delta K_i$
 - 12: **3. Sintonizador de Ganancia Derivativa (K_d)**
 si $|\Delta e|$ es Grande **entonces** ▷ Aumentar freno
 13: $\Delta K_d \leftarrow \text{Positivo}$
 si no si $|\Delta e|$ es Medio **entonces** ▷ Reducir freno
 14: $\Delta K_d \leftarrow \text{Negativo}$
 si no ▷ Ruido
 15: $\Delta K_d \leftarrow 0$
 - 16: $K_d \leftarrow K_d + \Delta K_d$
-

4. Salida Fuzzy (Desfuzzificación): Las salidas de los tres sistemas difusos independientes son una ganancia absoluta $K_p(k)$ y dos cambios de ganancia $\Delta K_i(k)$ y $\Delta K_d(k)$.

5. Cálculo de la Señal de Control (PID Clásico): El controlador PID recibe estas ganancias y las aplica. Las ganancias K_i y K_d son acumulativas y se saturan (limitan) dentro de sus rangos operativos seguros (ej. $K_i \in [0,1,0,4]$, $K_d \in [0,01,0,1]$).

$$K_p(k) = K_{p,\text{fuzzy}}(k)$$

$$K_i(k) = \text{saturate}(K_i(k - 1) + \Delta K_i(k))$$

$$K_d(k) = \text{saturate}(K_d(k - 1) + \Delta K_d(k))$$

Finalmente, un controlador PID discreto estándar utiliza estas ganancias sintonizadas para calcular la señal de control u_ω y u_ψ .

La señal de control $u_\omega(k)$ se calcula como:

$$u_\omega(k) = P(k) + I(k) + D(k) \quad (19)$$

Donde los tres términos (Proporcional, Integral y Derivativo) se calculan en cada instante de muestreo k :

- **Término Proporcional (P):**

$$P(k) = K_p(k) \cdot e_\omega(k)$$

- **Término Integral (I):**

$$I(k) = I(k - 1) + K_i(k) \cdot e_\omega(k) \cdot T_s$$

- **Término Derivativo (D):**

$$D(k) = K_d(k) \cdot \frac{e_\omega(k) - e_\omega(k - 1)}{T_s}$$

En estas ecuaciones, $e_\omega(k)$ es el error actual, $e_\omega(k - 1)$ es el error del paso anterior, $I(k - 1)$ es el valor acumulado del integrador, y T_s es el período de muestreo (definido en la Tabla 4 como el "Paso de integración" de 0.04 s).

El cálculo para $u_\psi(k)$ (el eje de Pitch) es idéntico. Estas señales (u_ω, u_ψ) son las que se envían al vector \mathbf{u}_g para comandar la estabilidad del robot.

7. Resultados

En la Figura 5 se muestra el resultado final del robot cuadrúpedo CheapSpot.

Para validar la arquitectura de control descrita en la Sección 6, se realizaron dos conjuntos de experimentos. El primer conjunto evalúa el rendimiento del sistema de estabilización (Controlador Fuzzy PID) en el modo teleoperado. El segundo conjunto evalúa el rendimiento del sistema de navegación autónoma (IBVS).

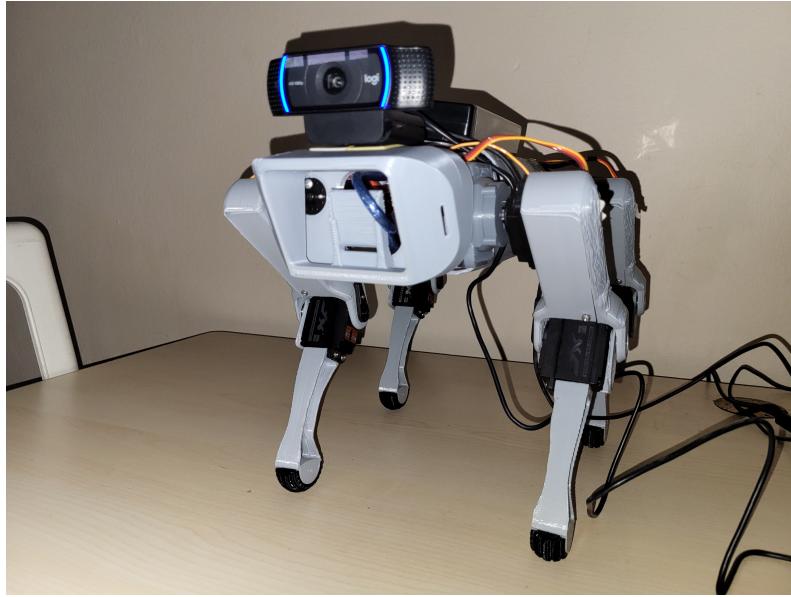


Figura 5 Robot cuadrúpedo CheapSpot

7.1. Validación del Control de Estabilidad (Modo Teleoperado)

El objetivo de este experimento es demostrar la eficacia del controlador de estabilidad Fuzzy PID para mantener el cuerpo del robot nivelado.

Se realizaron dos pruebas comparativas de locomoción teleoperada:

1. **Control PID Clásico:** Usando solo las ganancias base ($K_{p,\text{base}}, K_{i,\text{base}}, K_{d,\text{base}}$) de forma fija.
2. **Control Fuzzy PID:** Usando el auto-sintonizador (Sección 6.5.2) para modular las ganancias.

Para cada prueba, se coloco el robot sobre una superficie plana para después manipular sus patas manualmente, de manera que el robot ajustara su orientación. Se registraron los datos de la IMU (MPU-9250) para los ejes de Roll (ω_a) y Pitch (ψ_a).

La referencia deseada (ω_d, ψ_d) fue de 0° . El controlador PID cuenta $K_p = 1,8$, $K_i = 0,4$ y $K_d = 0,1$. Mientras que el controlador Fuzzy PID inicia con $K_p = 1,8$, $K_i = 0,1$ y $K_d = 0,01$.

7.1.1. Controlador PID Clásico

En la Figura 6, se muestra como el controlador PID clásico corrige la orientación del robot cuadrúpedo. En la Figura 7, se muestra la acción de control resultante del controlador PID clásico.



Figura 6 Gráfica de posición PID Clásico

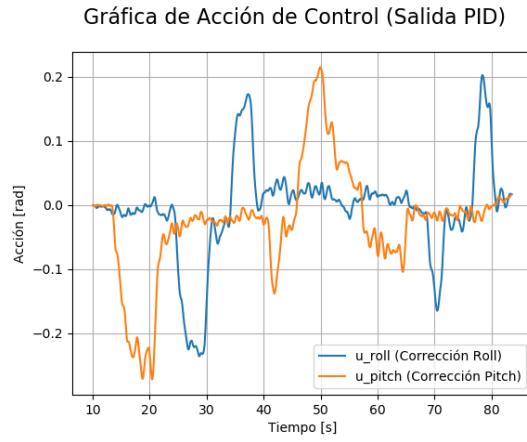


Figura 7 Gráfica de acción de control PID Clásico

7.1.2. Controlador Fuzzy PID

En la Figura 8, se muestra como el controlador Fuzzy PID corrige la orientación del robot cuadrúpedo. En la Figura 9, se muestra la acción de control resultante del controlador Fuzzy PID. En la Figura 10, se muestra la evolución de las ganancias del controlador Fuzzy PID.

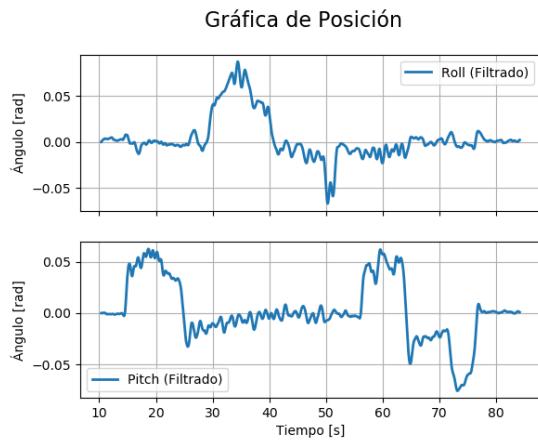


Figura 8 Gráfica de posición Fuzzy PID

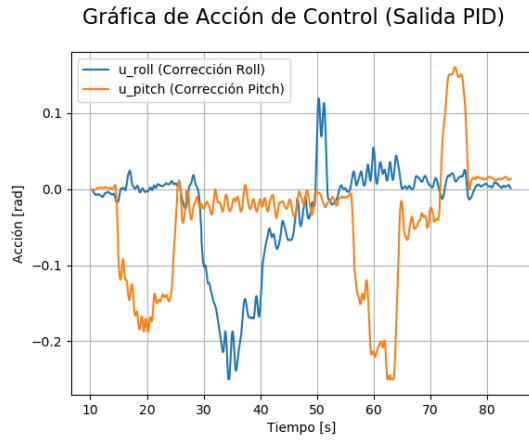


Figura 9 Gráfica de acción de control Fuzzy PID

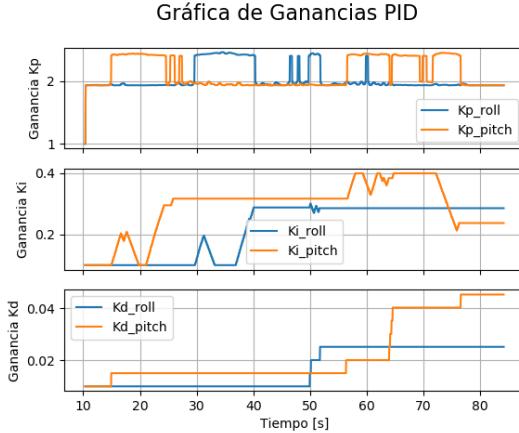


Figura 10 Gráfica de ganancias del control Fuzzy PID

7.1.3. Clásico vs Fuzzy PID

Como se observa en la Figura 6 y en la Figura 8, ambos controladores son capaces de corregir el error en Roll y Pitch correctamente. La diferencia clave es que probando entre ambos controladores se encuentra en la sintonización de ganancias. El PID clásico utiliza ganancias fijas durante toda la marcha, sintonizadas a través de prueba y error. Mientras, el controlador Fuzzy PID es un sistema adaptativo. Este controlador ajusta continuamente las tres ganancias basándose en el error y el cambio de error medidos por la IMU. Aunque en la experimentación no se encontraron diferencias muy remarcables, se espera que el Fuzzy PID ofrezca un desempeño superior al poder adaptarse con el tiempo.

7.2. Validación del Control Visual Autónomo (IBVS)

El objetivo de este experimento es validar la capacidad del robot para navegar de forma autónoma y alinearse con un objetivo el AprilTag de la Figura 11, utilizando el controlador IBVS.

Se diseñó un experimento donde el robot se colocó en una posición inicial con un error de traslación y rotación con respecto al marcador. El marcador se mantuvo estático a una distancia Z conocida. Al activar el modo IBVS, el robot debía navegar autónomamente hasta que el error visual (Ecuación 6) convergiera a cero. La Figura 12 muestra la evolución temporal del error visual en píxeles. La Figura 13 muestra las velocidades de la cámara \mathbf{v}_c , además de el error individual de las características visuales $|\mathbf{s} - \mathbf{s}_d|$. La Figura 14 muestra las velocidades del robot \mathbf{v}_p y su conversión a parámetros \mathbf{u}_g .

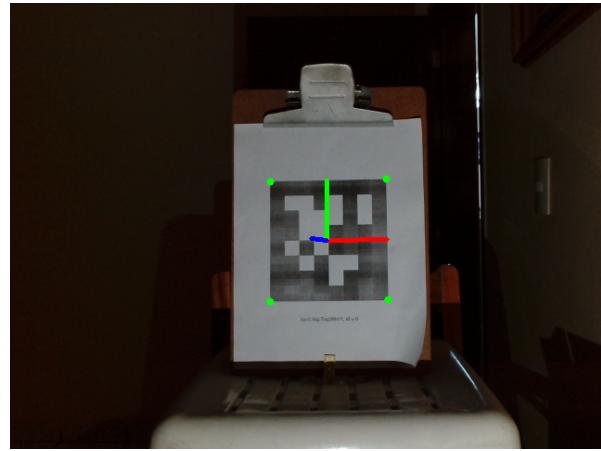


Figura 11 Apriltag 36h11, id=0

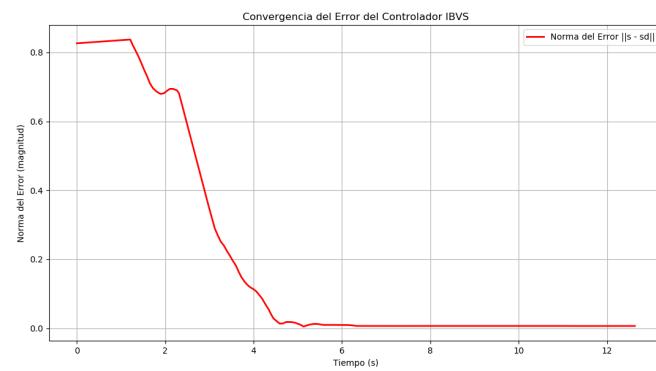


Figura 12 Gráfica del error $|s - s_d|$ IBVS

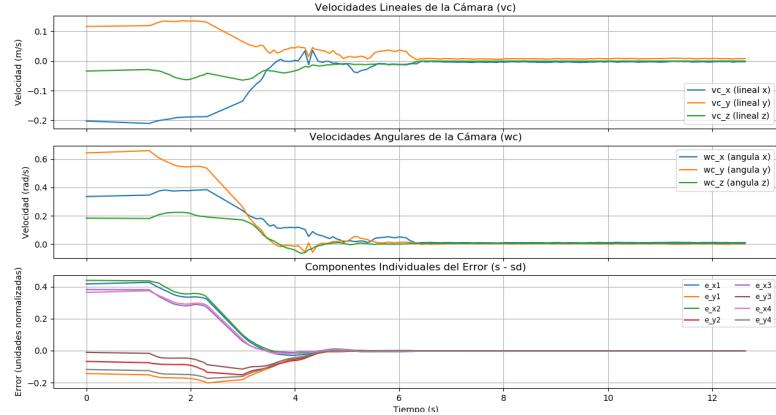


Figura 13 Gráfica de velocidades de la cámara \mathbf{v}_c y errores individuales $\mathbf{s} - \mathbf{s}_d$

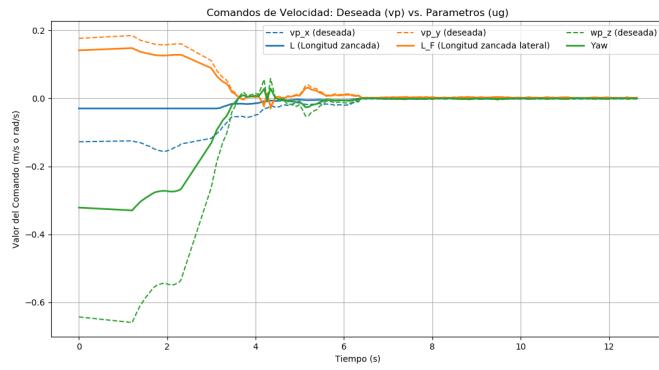


Figura 14 Gráfica de velocidades del robot \mathbf{v}_p y parámetros \mathbf{u}_g

El controlador IBVS funciona correctamente, aunque muchas de las ocasiones el apriltag se sale de foco de la cámara, provocando que se pierda el seguimiento, ademas de que debido a cuestiones de hardware del robot el movimiento tiende a ser errático.

8. Conclusión y trabajo futuro

8.1. Conclusión

En este artículo se ha presentado el diseño, implementación y validación de un sistema de control jerárquico para un robot cuadrúpedo de bajo costo, denominado *CheapSpot*. Se ha demostrado la viabilidad de integrar múltiples bucles de control complejos (locomoción, estabilización y servo visual) en una plataforma de hardware accesible (Raspberry Pi y Arduino).

El sistema implementado validó exitosamente dos modos de operación distintos:

- 1. Modo Teleoperado (con Estabilidad):** Se demostró que el auto-sintonizador Fuzzy PID (Sección 2) es capaz de estabilizar activamente la orientación (Roll y Pitch) del cuerpo del robot durante la marcha, utilizando los datos de la IMU para modular las ganancias K_p, K_i, K_d .
- 2. Modo Autónomo (con IBVS):** Se validó que el controlador IBVS (Sección 3) puede guiar exitosamente la locomoción del robot. El sistema mapea el error visual de un AprilTag a los parámetros del vector de marcha \mathbf{u}_g (Sección VII.C.5), logrando que el robot navegue y se alinee con un objetivo de forma autónoma.

El éxito de la integración se basó en una arquitectura de control cinemático completo, donde tanto el IBVS como el PID de estabilidad comandan parámetros de alto nivel (\mathbf{u}_g), que el Gait Generator (Sección 5) y la Cinemática Inversa (Sección 4) traducen en los 12 ángulos de los servomotores.

8.2. Limitaciones y Trabajo Futuro

A pesar de los resultados exitosos, la implementación en hardware físico reveló varias limitaciones que abren líneas claras de trabajo futuro:

1. Ruido en Sensores y Estabilidad: El problema más significativo fue el ruido en la IMU (MPU-9250), probablemente exacerbado por la vibración de los servomotores y los impactos de la marcha. Aunque se mitigó con un Madgwick, este filtro introduce una pequeña latencia. Como trabajo futuro, se propone implementar filtros más robustos (ej. un Filtro de Kalman Extendido - EKF) para una mejor fusión sensorial y una estimación de la orientación más limpia y rápida.

2. Limitaciones de Hardware (Chasis y Cámara): El chasis de ABS impreso en 3D, aunque económico, presenta una flexibilidad mecánica notable. Esto introduce vibraciones y discrepancias con el modelo cinemático ideal (que asume eslabones perfectamente rígidos). Además, el controlador IBVS es sensible a la oclusión y a la iluminación de la cámara. El seguimiento fallaba si el robot giraba demasiado rápido y el AprilTag salía del campo de visión de la cámara (FOV).

Como trabajo futuro, se propone mejorar la rigidez estructural del chasis y robustecer el sistema visual, implementando una lógica de "búsqueda de objetivo" cuando el marcador se pierde, o utilizando una cámara con un FOV más amplio.

3. Optimización del Controlador: El controlador Fuzzy PID implementado se basó en reglas de reglas heurísticas diseñadas manualmente. Esta sintonización es subjetiva y probablemente no sea la óptima.

Un avance significativo, propuesto como trabajo futuro clave, es la implementación de un controlador difuso óptimo, como el enfoque presentado por Tang et al. [7]. Este método utilizaría algoritmos de optimización (como Algoritmos Genéticos) para encontrar *automáticamente* el conjunto óptimo de reglas y funciones de membresía, eliminando la subjetividad del diseño y maximizando el rendimiento del estabilizador.

Referencias

- [1] Biswal, P., Mohanty, P.K.: Development of quadruped walking robots: A review. *Ain Shams Engineering Journal* **12**(2), 2017–2031 (2020)

- [2] Sen, M.A., Bakircioglu, V., Kalyoncu, M.: Inverse kinematic analysis of a quadruped robot. *International Journal of Mechanical Engineering and Technology (IJMET)* **8**(9), 580–590 (2017)
- [3] Hyun, D.J., Seok, S., Jongwoo, L., Sangbae, K.: High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *The International Journal of Robotics Research* **33**(11), 1417–1445 (2014)
- [4] Rahme, M., Abraham, I., Elwin, M., Murphey, T.: SpotMiniMini: Pybullet Gym Environment for Gait Modulation with Bezier Curves. [Software], version 2.1.0. [Online]. Available: https://github.com/moribots/spot_mini_mini (2014)
- [5] Kumar, V., Nakra, B.C., Mittal, A.P.: A review on classical and fuzzy pid controllers. *International Journal of Intelligent Control and Systems* **16**(3), 170–181 (2011)
- [6] Chaumette, F., Hutchinson, S.: Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine* **13**(4), 82–90 (2006)
- [7] Tang, K.S., Man, K.F., Chen, G., Kwong, S.: An optimal fuzzy pid controller. *IEEE Transactions on Industrial Electronics* **48**(4), 757–765 (2001)