Informe Técnico: Robot con Patas (Proyecto Mecatrónica)

Nombre del Vocero: Alvaro Quintana Camacho
Integrante 2: Daniel Montiel Lopez Integrante 3: David Pons Canet

9 de enero de 2026

Índice

1. Introducción y Objetivos

Para la resolución del proyecto, se ha diseñado y fabricado un robot con morfología de **hexápodo** (6 patas). La elección de esta configuración se basa principalmente en la estabilidad estática que ofrece, permitiendo que el robot mantenga siempre al menos tres puntos de apoyo en contacto con el suelo durante el movimiento (marcha trípode), lo cual es crítico para cumplir con el transporte de carga requerido.

1.1. Abordaje de las Pruebas Principales

El diseño del robot y su programación se centraron en superar los dos retos planteados de la siguiente manera:

- **Reto 1: Desplazamiento con Carga (Paralelo a Pared):** Para cumplir con el desplazamiento de 1 metro en línea recta, se implementó un sistema de control de rumbo basado exclusivamente en una **IMU (Inertial Measurement Unit)**. En lugar de depender de referencias externas variables, el robot utiliza el giroscopio interno para establecer un vector de avance constante.
- **Reto 2: Evasión y Autonomía:** Se desarrolló un algoritmo de navegación autónoma basado en una máquina de estados finitos. El robot inicia en el centro del área y utiliza sensores frontales para detectar obstáculos. Al encontrar una colisión potencial, el robot es capaz de pivotar sobre su propio eje central (gracias a la configuración radial de sus patas) para reorientarse rápidamente hacia las paredes colindantes del cuadrado de 2m^2.

El objetivo final es demostrar una integración mecatrónica fluida, donde el diseño mecánico propio y la programación embebida permitan una autonomía completa bajo las restricciones de dimensiones y energía establecidas.

2. Diseño Mecánico y de Detalle

2.1. Diseño Conceptual y Materiales

La elección de una configuración de seis patas (hexápodo) se fundamenta en la búsqueda de la **estabilidad estática total**. A diferencia de los robots bípedos o cuadrúpedos, que requieren algoritmos complejos de equilibrio dinámico, el hexápodo permite implementar una *marcha trípode*. En esta marcha, el robot mantiene siempre tres puntos de apoyo en el suelo formando un triángulo de soporte que contiene el centro de masas, garantizando que el robot no vuelque incluso cuando transporta la carga de 250g.

2.1.1. Justificación de Materiales y Fabricación

Tras las fases de prototipado, se incluyeron las siguientes mejoras técnicas que garantizan el cumplimiento de las restricciones del proyecto:

- **Estabilidad Dimensional del PLA:** Se prefirió el uso de PLA frente a otros polímeros debido a su baja deformación térmica (*warping*). Esto resultó crítico para asegurar que las tolerancias en las uniones de las articulaciones (coxa y fémur) fueran exactas, evitando holguras que degradarían la precisión de la marcha trípode bajo carga.
- **Iteración Rápida de Diseño:** El uso de la impresora **Bambu Lab P1S** permitió un ciclo de diseño ágil. Ante la necesidad de reforzar los apoyos para la carga de 250g, se pudieron rediseñar e imprimir nuevas versiones de los fémures en tiempos reducidos, optimizando la respuesta mecánica del robot antes de las pruebas finales.

2.2. Análisis Cinemático y de Carga

Cada una de las seis patas del robot cuenta con **dos grados de libertad (2 DoF)**, lo que suma un total de 12 actuadores para el sistema completo. Esta configuración se ha diseñado de la siguiente manera:

- **Eje de Abducción/Aducción (Coxa):** Un servo posicionado verticalmente que permite el movimiento de avance y retroceso de la pata respecto al cuerpo central.
- **Eje de Elevación (Fémur):** Un servo posicionado horizontalmente que controla la altura de la pata, permitiendo el despegue del suelo y el soporte del peso.

2.2.1. Distribución de Carga y Estabilidad

El análisis de carga se centró en la capacidad de los servos del fémur para soportar el peso propio del robot más los **250g adicionales** requeridos. Al utilizar una marcha trípode, el peso se reparte en tres patas simultáneamente.

Para minimizar el esfuerzo torsor sobre los ejes de los servos, se diseñó un área de carga de 5x5 cm situada en el centro geométrico del chasis superior. Esto asegura que el **Centro de Gravedad (CoG)** permanezca siempre dentro del triángulo de soporte formado por las patas en contacto con el suelo, cumpliendo con el criterio de estabilidad estática necesaria para el transporte seguro de la carga.

3. Simulación y Modelado

Antes de proceder a la fabricación física de los componentes, se realizó una fase de validación virtual para asegurar la integridad estructural y la viabilidad cinemática del diseño.

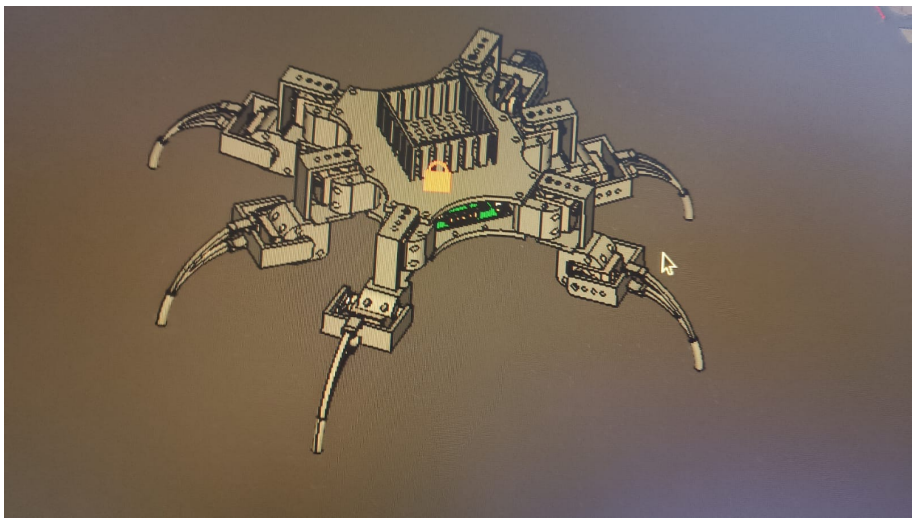

Figura 1: Vista isométrica del ensamblaje completo del hexápodo en entorno CAD.

3.1. Modelado FreeCAD y Blender

El robot fue diseñado utilizando el software paramétrico **FreeCAD**. Esta etapa permitió crear un ensamblaje virtual completo donde se verificaron las tolerancias de ajuste entre las piezas impresas y los servomotores.

Posteriormente, el modelo fue exportado a **Blender** para refinar la malla y preparar el entorno visual de simulación.

- **Detección de Interferencias:** Se simuló el recorrido completo de los servomotores (0° a 180°) para identificar colisiones entre la pierna (fémur) y el chasis principal, corrigiendo la geometría antes de la impresión.

3.2. Simulación Dinámica en PyBullet

Para validar los algoritmos de locomoción antes de implementarlos en el Arduino Mega, se utilizó el motor de física **PyBullet**. Esto permitió ajustar los parámetros de la marcha trípode y los giros sin poner en riesgo el hardware físico.

3.2.1. Simulación de Marcha (Andar)

El algoritmo de marcha implementado en la simulación replica la *marcha trípode*. Se definen dos grupos de patas (A y B) que alternan entre fase de soporte y fase de vuelo. El movimiento se genera mediante funciones sinusoidales para suavizar la aceleración de los servos.

A continuación se presenta el código desarrollado para la prueba de avance lineal:

Listing 1: Código de simulación de marcha trípode en PyBullet

```
import pybullet as p
import pybullet_data
import time
import math

p.connect(p.GUI)
p.setGravity(0, 0, -9.8)
p.setTimeStep(1./240.)

# Configuracion inicial
euler_angles = [0, 0, 0]
startOrientation = p.getQuaternionFromEuler(euler_angles)
p.setAdditionalSearchPath(pybullet_data.getDataPath())
p.loadURDF("plane.urdf")
robotID = p.loadURDF("hexapod_mecatronica/urdf/hexapod_mecatronica.urdf")

# Definicion de IDs de Joints (Simplificado)
J1_RF, J2_RF = 11, 12
J1_RM, J2_RM = 1, 2
J1_RB, J2_RB = 14, 15
J1_LF, J2_LF = 4, 5
J1_LM, J2_LM = 7, 8
J1_LB, J2_LB = 17, 18

# Grupos de tripodes
TRIPOD_A = [(J1_RF, J2_RF, 'RIGHT'), (J1_LM, J2_LM, 'LEFT'), (J1_RB, J2_RB, 'RIGHT')]
TRIPOD_B = [(J1_LF, J2_LF, 'LEFT'), (J1_RM, J2_RM, 'RIGHT'), (J1_LB, J2_LB, 'LEFT')]

# Parametros de la marcha
STEP_AMPLITUDE = 0.8
LIFT_HEIGHT = 0.6
CYCLE_DURATION = 2.0
SPEED_FACTOR = 1.0

def set_leg_position(joint_j1, joint_j2, j1_angle, j2_angle, side):
    if side == 'RIGHT': j1_angle = -j1_angle
    p.setJointMotorControl2(robotID, joint_j1, p.POSITION_CONTROL, targetPosition=
        j1_angle)
    p.setJointMotorControl2(robotID, joint_j2, p.POSITION_CONTROL, targetPosition=
        j2_angle)

def set_tripod_position(tripod, j1_angle, j2_angle):
    for joint_j1, joint_j2, side in tripod:
        set_leg_position(joint_j1, joint_j2, j1_angle, j2_angle, side)

start_time = time.time()
try:
    while True:
        current_time = time.time() - start_time
```

```
        phase = (current_time * SPEED_FACTOR / CYCLE_DURATION) % 1.0

        if phase < 0.5: # TRIPOD A mueve, B soporta
            phase_a = phase * 2
            j1_a = math.sin(phase_a * math.pi) * STEP_AMPLITUDE
            j2_a = -LIFT_HEIGHT * math.sin(phase_a * math.pi)

            phase_b = phase * 2
            j1_b = STEP_AMPLITUDE - (phase_b * 2 * STEP_AMPLITUDE)
            j2_b = 0
        else: # TRIPOD B mueve, A soporta
            phase_b = (phase - 0.5) * 2
            j1_b = math.sin(phase_b * math.pi) * STEP_AMPLITUDE
            j2_b = -LIFT_HEIGHT * math.sin(phase_b * math.pi)

            phase_a = (phase - 0.5) * 2
            j1_a = STEP_AMPLITUDE - (phase_a * 2 * STEP_AMPLITUDE)
            j2_a = 0

        set_tripod_position(TRIPOD_A, j1_a, j2_a)
        set_tripod_position(TRIPOD_B, j1_b, j2_b)
        p.stepSimulation()
        time.sleep(1./240.)
except KeyboardInterrupt:
    p.disconnect()
```

3.2.2. Simulación de Rotación (Girar)

Para la maniobra de evasión, es necesario que el robot gire sobre su propio eje (Yaw). La simulación probó una lógica diferencial donde el "tren"derecho y el "trenïzquierdo se mueven en fases opuestas, generando el par de torsión necesario para rotar sin desplazarse del centro.

El siguiente código valida la cinemática de giro estático:

Listing 2: Código de simulación de rotación sobre el eje Z

```
import pybullet as p
import pybullet_data
import time
import math

p.connect(p.GUI)
p.setGravity(0, 0, -9.8)
p.setTimeStep(1./240.)

p.setAdditionalSearchPath(pybullet_data.getDataPath())
p.loadURDF("plane.urdf")
robotID = p.loadURDF("hexapod_mecatronica/urdf/hexapod_mecatronica.urdf")

# Definicion de Joints (IDs mapeados previamente)
# ... [Definicion de J1_RF hasta J2_LB identica al caso anterior] ...

# Agrupacion por trenes laterales para giro diferencial
RIGHT_TRAIN_J1 = [11, 1, 14]  # RF, RM, RB (J1)
RIGHT_TRAIN_J2 = [12, 2, 15]  # RF, RM, RB (J2)

LEFT_TRAIN_J1 = [4, 7, 17]    # LF, LM, LB (J1)
LEFT_TRAIN_J2 = [5, 8, 18]    # LF, LM, LB (J2)

while True:
    t = time.time()
    phase = (t % 2.0) / 2.0   # ciclo de 2 segundos

    # Movimiento senoidal suave
    # Forward positivo para un lado y negativo para el otro genera giro
    forward  = 0.6 * math.sin(2 * math.pi * phase)
    lift     = 0.4 * math.sin(2 * math.pi * phase + math.pi/2)

    # Tren Derecho (Movimiento positivo)
    p.setJointMotorControlArray(robotID, RIGHT_TRAIN_J1, p.POSITION_CONTROL,
                                targetPositions=[ forward ] * 3)
```

```
    p.setJointMotorControlArray(robotID, RIGHT_TRAIN_J2, p.POSITION_CONTROL,
                                targetPositions=[ lift ] * 3)

    # Tren Izquierdo (Movimiento negativo -> Contra-fase de avance)
    p.setJointMotorControlArray(robotID, LEFT_TRAIN_J1, p.POSITION_CONTROL,
                                targetPositions=[ -forward ] * 3)
    p.setJointMotorControlArray(robotID, LEFT_TRAIN_J2, p.POSITION_CONTROL,
                                targetPositions=[ -lift ] * 3)

    p.stepSimulation()
    time.sleep(1./240.)

p.disconnect()
```

4. Electrónica y Selección de Componentes

Componente	Modelo	Función Principal
Microcontrolador	Arduino Mega	Cerebro del robot
Servomotores	SG90 / Parallax	Movimiento de articulaciones
Sensor Ultrasonidos	HC SR04	Detección de obstaculos
Regulador Tension	ARCELI AA0003	Regular la tension de la bateria
PCA	PCA9685	Control de los servos
Sensor IMU	MPU-6050	Orientacion
Bateria	AMZZN 7.4V 5000mAh	Alimentacion de los componentes
Led	RGB	Interfaz de estado del sistema

Cuadro 1: Listado de componentes principales.

4.1. Unidad de Procesamiento

Como núcleo del sistema de control, se ha seleccionado un microcontrolador **Arduino Mega** basado en el chip ATmega2560. Los motivos técnicos que justifican esta selección frente a otras placas (como el Arduino Uno) son:

- **Capacidad de Control de Actuadores:** El hexápodo requiere el control simultáneo de 13 servomotores (2 DoF por pata y en el sensor de ultrasonidos). El Arduino Mega dispone de 15 pines con modulación por ancho de pulsos (PWM), lo que permite gestionar todos los actuadores de forma nativa.
- **Gestión de Periféricos y Sensores:** Gracias a sus múltiples puertos de comunicación serie (UART) y bus I^2C, se facilita la conexión simultánea de la **IMU** y el sensor de ultrasonidos sin conflictos de direcciones o pines.
- **Memoria y Procesamiento:** Con 256 KB de memoria Flash, la placa permite almacenar algoritmos complejos de cinemática y la máquina de estados de navegación autónoma, además de gestionar las librerías necesarias para el procesamiento de datos inerciales.

4.2. Sistema de Alimentación y Autonomía

Para garantizar la estabilidad del sistema y cumplir con la restricción de autonomía mediante baterías recargables, se ha implementado un esquema de **alimentación aislada** que separa la electrónica de control de la etapa de potencia.

4.2.1. Alimentación de la Lógica (Control)

El microcontrolador **Arduino Mega 2560** se alimenta de forma independiente mediante una **pila de 9V**.

- Esta configuración asegura que las caídas de tensión bruscas (*brown-outs*) provocadas por el pico de arranque de los 12 servomotores no afecten al procesador.
- Se utiliza el regulador interno del Arduino para estabilizar el voltaje de operación de los sensores (IMU y Ultrasonidos).

4.2.2. Alimentación de Potencia (Actuadores)

Para el movimiento de las patas y el soporte de la carga de 250g, se utiliza una batería de alta capacidad:

- **Tipo:** Li-ion de 7.4V (configuración 2S1P).
- **Capacidad:** 5000 mAh, lo que permite múltiples ciclos de prueba sin degradación del torque de los motores.
- **Tasa de descarga:** 15C, garantizando el suministro de corriente necesario cuando el hexápodo realiza movimientos de elevación con carga.

4.2.3. Cálculo de Balance de Potencia y Autonomía

Para justificar técnicamente la elección de la batería de 5000 mAh y 15C, se realiza un análisis del consumo máximo estimado del sistema (excluyendo el Arduino Mega, que cuenta con su propia fuente de 9V). Los componentes alimentados por la batería de Li-ion son:

Componente	Cantidad	Consumo Promedio (mA)	Consumo Total (mA)
Servos SG90 (Articulaciones/Barrido)	7	220	1540
Servos Parallax (Articulaciones)	6	250	1500
LED Indicador de estado	1	20	20
Consumo Total Estimado (I_{total})			**3060 mA**

Cuadro 2: Estimación de consumo de corriente en operación nominal.

1. Cálculo de Autonomía (Tiempo de Operación) Utilizando la capacidad nominal de la batería ($C = 5000$ mAh), calculamos el tiempo teórico de funcionamiento continuo bajo carga máxima:

$$T = \frac{C}{I_{total}} = \frac{5000 \text{ mAh}}{3060 \text{ mA}} \approx 1{,}63 \text{ horas} \approx 98 \text{ minutos}$$

Dado que la suma de las dos pruebas de la práctica no supera los 3 minutos de operación real, la batería ofrece un margen de seguridad superior al 3000 %, permitiendo múltiples sesiones de calibración y pruebas sin necesidad de recarga.

2. Validación de la Tasa de Descarga (C-rating) Es crítico asegurar que la batería pueda entregar picos de corriente si los motores se bloquean (*stall current*). Estimando un consumo de pico de 650mA para SG90 y 800mA para Parallax:

$$I_{pico} = (7 \times 650 \text{ mA}) + (6 \times 800 \text{ mA}) = 4550 + 4800 = 9350 \text{ mA} = 9{,}35 \text{ A}$$

La capacidad de descarga de la batería es:

$$I_{max\_bat} = C \times \text{C-rating} = 5 \text{ Ah} \times 15\text{C} = 75 \text{ A}$$

Como 75 A $\gg$ 9,35 A, la batería opera con total seguridad, garantizando que el voltaje no caerá incluso durante movimientos bruscos o esfuerzos máximos con la carga de 250g.

4.3. Actuadores y Sensores

El sistema de locomoción y percepción del hexápodo se basa en una combinación de actuadores de diferente torque y una suite de sensores para navegación inercial y espacial.

4.3.1. Actuadores (Servomotores)

Se han seleccionado dos tipos de servomotores para optimizar la relación peso-potencia del robot:

- **Servos Parallax (6 unidades):** Utilizados en las articulaciones del *fémur*. Se eligieron por su robustez y mayor torque, encargándose de elevar el chasis y soportar la carga de 250g durante la marcha trípode.
- **Servos SG90 (7 unidades):** Seis de ellos se ubican en las articulaciones de la *coxa* para el movimiento de avance (donde el requerimiento de torque es menor). El séptimo servo se utiliza para el sistema de percepción activa, permitiendo el barrido del sensor de ultrasonidos.

4.3.2. Sensores de Navegación y Percepción

- **Sensor de Ultrasonidos (HC-SR04):** Montado sobre un servomotor en la parte frontal, permite medir distancias y detectar obstáculos mediante la emisión y recepción de ráfagas de ultrasonido, esencial para la Prueba 2.
- **Sensor IMU (Unidad de Medición Inercial):** Dispositivo fundamental para la Prueba 1. Proporciona datos de aceleración y velocidad angular, permitiendo que el robot mantenga un rumbo paralelo a la pared mediante el control del eje de *yaw* (guiñada).

4.4. Conexiones y Gestión de Energía

El diseño eléctrico se ha optimizado para manejar altas corrientes y garantizar una señal de control limpia, utilizando componentes especializados para la regulación y distribución de señales.

4.4.1. Regulación de Voltaje (Buck Converter)

Para alimentar la etapa de potencia, se ha integrado un módulo **ARCELI 20A 300W DC-DC Step Down**. Este regulador es fundamental por las siguientes razones:

- **Estabilidad de Torque:** Reduce el voltaje de la batería de Li-ion (7.4V - 8.4V) a un valor constante de 5.0V, asegurando que los servomotores mantengan un torque uniforme independientemente del estado de carga de la batería.
- **Capacidad de Corriente:** Con una capacidad de hasta 20A, el módulo puede gestionar sin esfuerzo los picos de corriente (*stall current*) de los 13 servomotores cuando el robot transporta la carga de 250g.

4.4.2. Controladora de Servos

Para la gestión de los 13 actuadores, se ha implementado una controladora **PCA9685** conectada al Arduino Mega mediante el protocolo **I^2C**.

- **Descarga del CPU:** Al tener su propio reloj interno, la PCA9685 genera las señales PWM de forma independiente, permitiendo que el Arduino Mega dedique sus recursos exclusivamente al procesamiento de la IMU y la lógica de navegación.
- **Resolución:** Ofrece una resolución de 12 bits, lo que permite un control de posición mucho más fino en las articulaciones del hexápodo.

5. Interfaz de Usuario (HMI)

Para cumplir con los requisitos de interacción y seguridad, el robot integra una interfaz básica pero robusta que permite el control y la monitorización visual del estado interno:

- **Switch de Selección de Modo:** Se ha implementado un interruptor de tres posiciones (o tres estados lógicos) que permite alternar entre las funcionalidades principales sin necesidad de reprogramar el microcontrolador:
  1. **Modo Reposo/Calibración:** El robot mantiene las patas en posición neutra para facilitar el mantenimiento o la carga de los 250g.
  2. **Modo Prueba 1:** Ejecuta el algoritmo de desplazamiento lineal con control de rumbo por IMU.
  3. **Modo Prueba 2:** Activa la navegación autónoma con evasión de obstáculos y barrido ultrasónico.
- **Indicador visual LED RGB:** Actúa como un sistema de diagnóstico en tiempo real. Su código de colores permite al operador conocer el estado del robot a distancia:
  - **Cian:** Calibrando sensor IMU.
  - **Azul:** Modo reposo.
  - **Verde:** Modo Prueba 1.
  - **Morado:** Modo Prueba 2.
  - **Amarillo:** Retrocediendo en la Prueba 2.

6. Programación Embebida y Control de Movimiento

6.1. Algoritmo de Marcha (Gait Control)

Para este hexápodo de 2 DoF, se ha implementado una **Marcha Trípode Alternante**, la cual es la más eficiente en términos de velocidad y estabilidad para esta morfología. El algoritmo divide las patas en dos grupos:

- **Grupo A:** Patas 1, 3 y 5.
- **Grupo B:** Patas 2, 4 y 6.

El ciclo de marcha se coordina mediante una máquina de estados que ejecuta los siguientes pasos:

1. **Elevación y Avance:** El Grupo A eleva sus articulaciones de fémur y avanza mediante la coxa, mientras el Grupo B permanece en el suelo impulsando el cuerpo hacia adelante.

2. **Apoyo:** El Grupo A desciende hasta tocar el suelo, estableciendo un nuevo triángulo de soporte.

3. **Intercambio:** Se repite el proceso de forma inversa para el Grupo B.

Debido a que solo disponemos de 2 DoF, la trayectoria del pie no es una elipse perfecta, sino un movimiento coordinado de "levante-avance-descenso". La programación se optimizó para que los cambios de estado ocurran con un solapamiento mínimo, asegurando que el robot nunca pierda los tres puntos de apoyo, algo vital para que la carga de 250g no sufra inercias bruscas que puedan desestabilizar la estructura.

6.2. Lógica de las Pruebas

La inteligencia del hexápodo se estructura en una máquina de estados finitos que alterna su comportamiento según la fase de la práctica:

6.2.1. Prueba 1: Control de Rumbo (Heading Control) mediante IMU

En esta fase, el objetivo es el desplazamiento lineal paralelo a la pared. A diferencia de un control de distancia reactivo, se ha implementado un algoritmo de **Control de Orientación**:

- **Referencia Inicial:** Al iniciar la prueba, el robot calibra la IMU para establecer el ángulo de desplazamiento deseado como 0°.
- **Lazo de Control:** Durante la marcha trípode, el microcontrolador lee constantemente el valor del *yaw* (guiñada) del giroscopio.
- **Corrección Diferencial:** Si se detecta una desviación angular (causada por el peso de los 250g o el deslizamiento de las patas), el algoritmo modifica el ancho de paso (*stride*) de forma asimétrica. Por ejemplo, si el robot rota a la derecha, se incrementa el avance de las patas izquierdas y se reduce el de las derechas hasta recuperar el ángulo de referencia.

Este método garantiza que el robot se mueva paralelo a la pared sin necesidad de "verla", confiando en su propia estabilidad inercial.

6.2.2. Prueba 2: Evasión y Autonomía con Percepción Activa

Para la navegación en el área de 2m^2, se ha implementado un algoritmo de **exploración por objetivos de orientación** $(0, \pi/2, \pi, 3\pi/2)$. El robot utiliza un sistema de percepción activa compuesto por un sensor de ultrasonidos montado sobre un servomotor independiente.

- **Navegación Orientada:** El robot inicia en el centro y selecciona una de las cuatro direcciones cardinales como objetivo. Utiliza la IMU para mantener el rumbo hacia dicho objetivo mientras avanza.
- **Detección y Barrido (Scanning):** Si durante el trayecto el sensor de ultrasonidos detecta un obstáculo en su eje frontal, el robot detiene su marcha. En ese momento, el servo del sensor ejecuta un *barrido angular* para mapear el entorno inmediato y buscar un espacio libre.
- **Evasión y Retorno:**
    1. Tras el barrido, el hexápodo gira hacia la zona despejada, avanza para sortear el obstáculo y vuelve a ejecutar un barrido.
    2. El algoritmo intenta constantemente retomar la dirección original (objetivo de orientación inicial) mediante correcciones de giro.

- **Detección de Paredes y Cambio de Objetivo:** Si tras un barrido completo el sensor solo detecta obstáculos en todas las direcciones, el sistema interpreta que ha alcanzado una de las paredes del recinto. En este punto, el robot marca el objetivo como alcanzado y cambia su dirección hacia la siguiente de las cuatro orientaciones programadas.

Esta lógica permite al robot cumplir con el requisito de alcanzar todas las paredes colindantes de forma autónoma, gestionando los obstáculos intermedios mediante una toma de decisiones basada en el escaneo dinámico del entorno.

7. Tests

Tras el ensamblaje mecánico y electrónico, se llevó a cabo un plan de validación incremental para asegurar el correcto funcionamiento del hexápodo. Esta fase experimental se dividió en tres etapas críticas.

7.1. Test 1: Diagnóstico de Potencia y Alimentación

En la primera iteración de pruebas, se intentó alimentar tanto la etapa de potencia (servomotores) como la etapa de lógica (Arduino Mega y sensores) utilizando una única fuente de energía: la batería Li-ion de 7.4V.

- **Problema Detectado:** Al activar el ciclo de marcha, el consumo simultáneo de los 12 servo, más el Arduino Mega, provocaba que no había corriente sufuciente en el bus de alimentación común. Esto tenía un efecto negativo:

  1. Los servomotores perdían torque drásticamente, siendo incapaces de levantar el peso del chasis del suelo.

- **Solución Implementada:** Se optó por desacoplar completamente los circuitos. Se introdujo una **pila de 9V dedicada exclusivamente al Arduino Mega**. Esto garantizó una señal de control PWM estable y permitió que la batería Li-ion dedicara toda su capacidad de corriente exclusivamente a generar el torque necesario en los servos para levantar el robot.

7.2. Test 2: Validación Cinemática (Simulación vs. Realidad)

Una vez estabilizada la alimentación, se procedió a cargar los algoritmos validados previamente en PyBullet. El objetivo fue comprobar la fidelidad de la simulación respecto al comportamiento físico real.

- **Prueba de Andar:** Se verificó que la marcha trípode generaba un avance efectivo. Se realizaron ajustes menores en los *offsets* de los servos para corregir desviaciones.

- **Prueba de Giro:** Se validó el algoritmo de giro diferencial (mover los trenes de patas en direcciones opuestas). El robot demostró capacidad para rotar sobre su centro geométrico sin desplazarse, confirmando la viabilidad de la maniobra de evasión planificada.

7.3. Tests Finales: Integración de Funcionalidades

En la fase final, se fusionaron los módulos individuales para cumplir con los requisitos del concurso:

- **Ajuste de Rumbo:** Se sintonizaron las constantes del controlador de la IMU y el ultrasonidos para que el robot corrigiera su trayectoria en tiempo real, logrando avanzar en línea recta.

- **Lógica de Evasión:** Se calibraron los umbrales de distancia del sensor ultrasónico para evitar falsos positivos y asegurar que el robot detectara las paredes y obstaculos a tiempo para frenar y reorientarse.

8. Conclusiones

El desarrollo del proyecto Robot con Patas ha permitido la integración exitosa de conocimientos en diseño mecánico, electrónica de potencia y programación embebida. Tras la ejecución de las pruebas y el análisis del rendimiento del hexápodo, se extraen las siguientes conclusiones:

- **Eficacia de la Morfología:** La configuración de hexápodo con marcha trípode demostró ser una solución óptima para el transporte de carga. La estabilidad estática inherente a este diseño permitió gestionar el peso adicional de 250g sin comprometer el equilibrio, validando el diseño mecánico realizado en PLA.
- **Control Inercial y Percepción:** La implementación de la IMU para el control de rumbo en la Prueba 1 marcó una diferencia técnica significativa, permitiendo un desplazamiento lineal preciso sin depender de referencias externas. Asimismo, el sistema de percepción activa (sensor ultrasónico sobre servo) dotó al robot de una capacidad de toma de decisiones robusta para la navegación autónoma.
- **Gestión Energética:** El uso del regulador ARCELI de 20A y la controladora PCA9685 resultó crítico. La separación de la alimentación entre la lógica (9V) y la potencia (7.4V) garantizó que el microcontrolador operara libre de ruidos electromagnéticos, asegurando la fluidez de los algoritmos de control.
- **Interacción y Diagnóstico:** La interfaz HMI, mediante el switch de tres posiciones y el código de colores del LED RGB, facilitó la supervisión de las tareas en tiempo real, cumpliendo con los estándares de usabilidad requeridos.

Como líneas de trabajo futuras, se propone la implementación de algoritmos de cinemática inversa para permitir trayectorias de las patas más fluidas y la integración de sensores de corriente para monitorizar en tiempo real el esfuerzo de los servomotores bajo diferentes cargas.