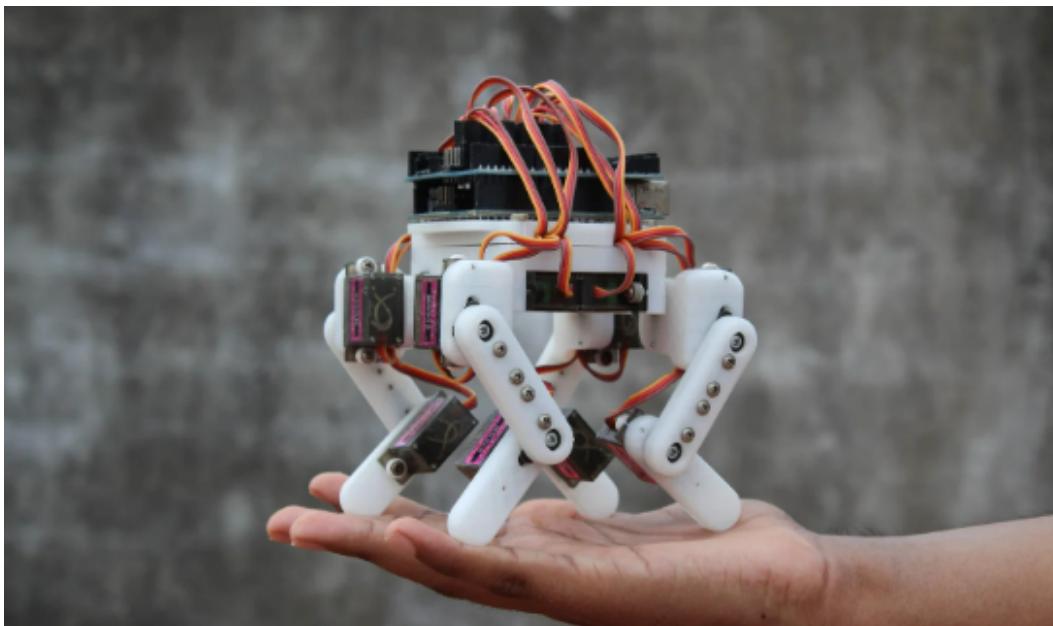


9 de febrero de 2024

Taller de Proyecto II

Informe Final Proyecto A3. Micro Quadruped.



Grupo de Desarrollo

- ★ Gobbi, Leandro - 02955/1
- ★ Gonzalez, Martina Lucía - 01807/4

Índice

1. Descripción General del Proyecto.....	3
2. Presentación esquemática del proyecto.....	5
3. Sensor ADXL345.....	8
3.1 Características.....	8
3.2 Implementación en el proyecto.....	12
4. Documentación del Software del proyecto.....	14

1. Descripción General del Proyecto

El proyecto Micro Quadruple se centra en optimizar el control y el desplazamiento de un robot cuadrúpedo de pequeño tamaño. La propuesta inicial planteaba rearmar un prototipo existente para continuar la investigación del desplazamiento del mismo y agregar nuevas funcionalidades al proyecto. Estas nuevas funcionalidades consisten en inclinarse hacia adelante, desplazarse hacia adelante y tener una posición de equilibrio central.

Objetivos principales

- Control remoto a través de un Access Point Wi-Fi: Permitir la interacción con el robot sin depender de cables, facilitando el acceso y manejo desde cualquier dispositivo conectado a la red.
- Implementación de un acelerómetro ADXL345 para detectar inclinaciones: Este sensor permitiría al robot identificar desequilibrios en base a la posición de los ejes X, Y y Z actuales, lo cual es esencial para mejorar su estabilidad y lograr su capacidad de inclinarse sin perder el equilibrio.

Modificaciones

Durante el desarrollo, la cátedra sugirió varias modificaciones y ajustes técnicos para alinear el proyecto con los objetivos académicos:

- Clarificación de términos técnicos: Se solicitó una revisión del uso de conceptos como "tiempo real" para evitar confusiones en la evaluación y documentación. Este término se refinó para referirse específicamente a la detección inmediata y la respuesta automatizada del robot a cambios en su entorno.
- Cambio de plataforma: Originalmente, se utilizaba un Arduino UNO como base del proyecto. Sin embargo, se recomendó migrar al NodeMCU (ESP8266), ya que este microcontrolador integra capacidades Wi-Fi, optimizando el manejo de la conectividad y reduciendo la complejidad del diseño electrónico.
- Cambio en el lenguaje utilizado: El proyecto realizado por el grupo de alumnos anterior utilizó ReactJS para el Frontend del proyecto mientras que en nuestro caso se utilizó HTML y CSS.

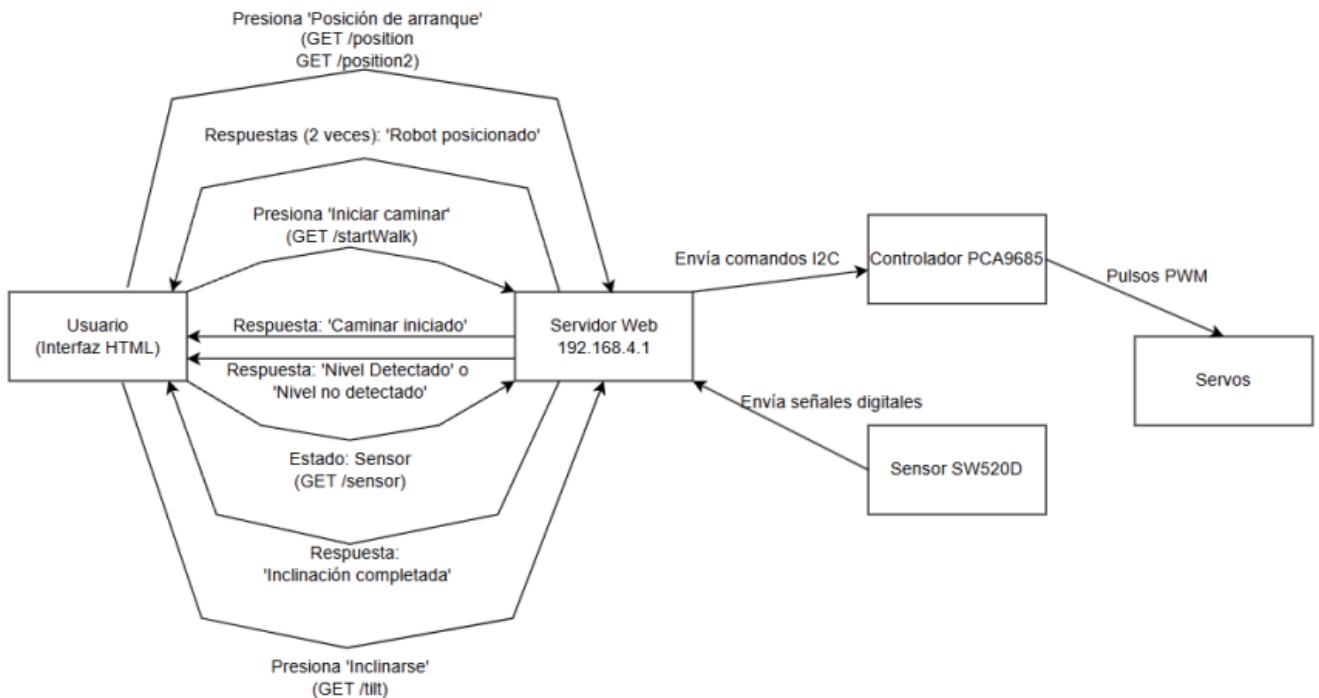
Cambios durante el desarrollo del proyecto

A medida que se fue progresando en la resolución de las distintas mecánicas del proyecto, se implementaron mejoras basadas en las necesidades detectadas. Entre las decisiones más destacadas se encuentran:

- Implementación de una interfaz web: Se desarrolló una plataforma interactiva que permite visualizar el estado de los servomotores y la posición XYZ obtenida del sensor ADXL345. Esta interfaz facilita al usuario el control de las funciones del robot, incluyendo desplazamientos, posicionamiento e inclinación.
- Control dinámico de inclinación: Se definió un comportamiento específico para manejar el equilibrio del robot. Cuando se presiona el botón correspondiente para iniciar la inclinación del robot, éste comienza a desnivelarse con las patas delanteras. Una vez que el sensor retorna una posición de desequilibrio límite, el robot se ajusta automáticamente a su posición inicial. Este mecanismo asegura que el robot sea capaz de adaptarse y estabilizarse sin intervención manual ante el estado de inclinación iniciada.

2. Presentación esquemática del proyecto

El gráfico que se muestra a continuación describe las interacciones entre los componentes principales del sistema desde el punto de vista del usuario hasta el control del hardware del robot:



Descripción del flujo de trabajo:

- **Usuario (Interfaz HTML)**: El usuario interactúa con la interfaz web alojada en el MCU (NodeMCU ESP8266), presionando botones para realizar acciones como "Posición de arranque", "Iniciar caminar" o "Inclinarse". Cada acción del usuario genera una solicitud HTTP GET al servidor web en el MCU.
- **Servidor Web (NodeMCU ESP8266)**: Recibe solicitudes del usuario y responde con mensajes de estado, como "Robot posicionado" o "Caminar iniciado". Envía comandos I2C al controlador PCA9685 para mover los servos y ejecutar los movimientos correspondientes.
- **Controlador PCA9685 y Servos**: El PCA9685 recibe comandos I2C del MCU y genera pulsos PWM para controlar los servos. Los servos realizan los movimientos del robot según las instrucciones recibidas.

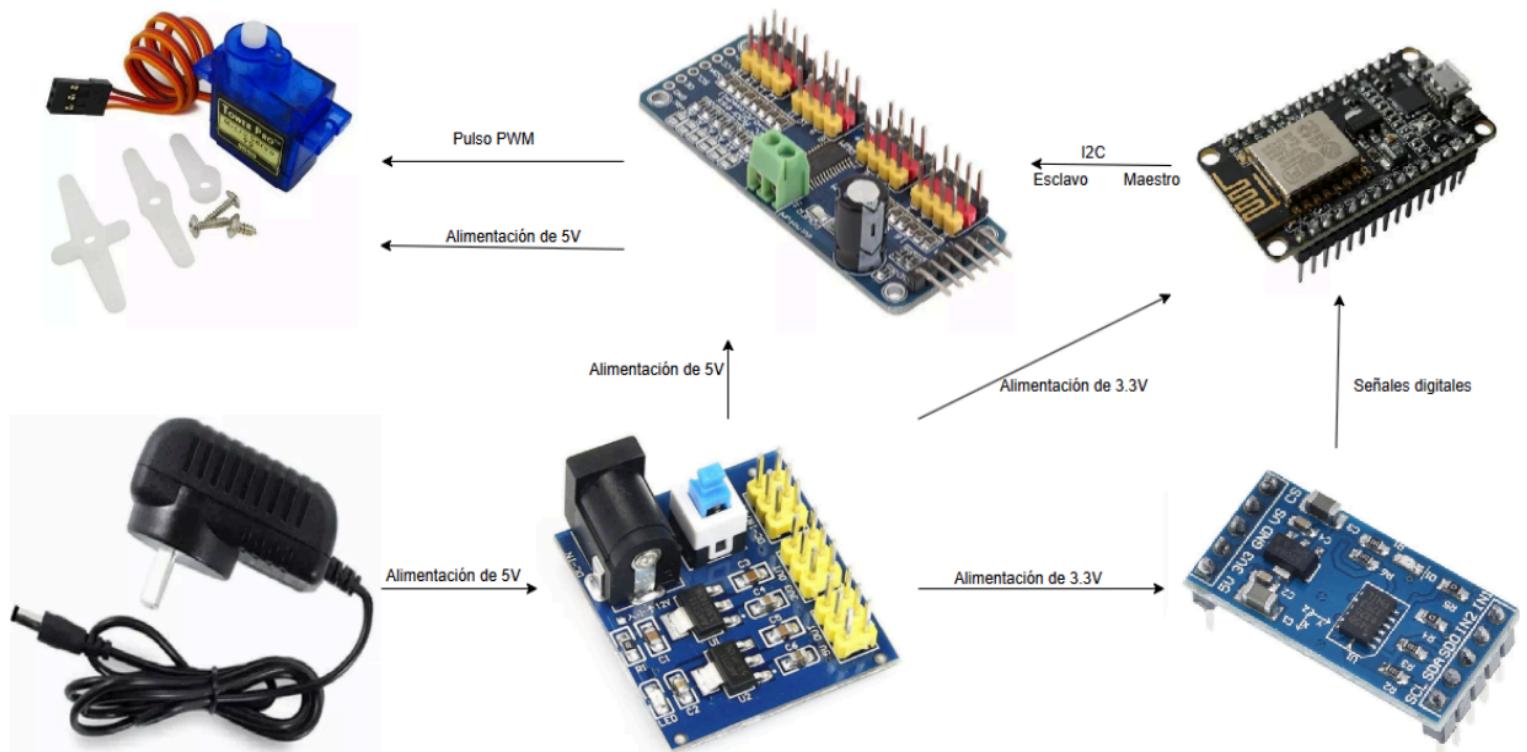
- **Sensor ADXL345:** Proporciona datos digitales al MCU sobre las lecturas de aceleración en las diferentes coordenadas de cada eje. Esto permite que el sistema tome decisiones, como volver a una posición inicial cuando se detecta que el robot está desnivelado.

Implementaciones realizadas por el grupo de desarrollo

- Desarrollo del servidor web en el NodeMCU para manejar solicitudes HTTP.
- Creación de la interfaz HTML para el control del robot.
- Implementación de la lógica de control de movimiento e inclinación en el código del MCU.
- Configuración del protocolo I2C para la comunicación con el PCA9685.
- Integración del sensor ADXL345 para detectar la posición del robot también con protocolo I2C.

Esquemático de conexiones del hardware utilizado

El siguiente esquema de hardware detalla las conexiones físicas y las relaciones entre los diferentes componentes del sistema:



Descripción de los componentes y conexiones

- **Fuente de alimentación:** Alimenta al controlador PCA9685 y los servos con 5V. Proporciona 3.3V para el NodeMCU ESP8266 y el sensor ADXL345 a través de un regulador de voltaje.
- **NodeMCU ESP8266:** Se comunica con el PCA9685 a través del protocolo I2C para enviar comandos de control a los servos. Recibe señales digitales de las lecturas realizadas por el ADXL345 para detectar inclinaciones.
- **Controlador PCA9685:** Controla hasta 16 servos (en nuestro caso particular 12) utilizando señales PWM generadas a partir de los comandos recibidos del NodeMCU.
- **Servos:** Realizan los movimientos físicos del robot cuadrúpedo, como caminar, inclinarse o volver a la posición inicial.
- **Sensor ADXL345:** Mide los valores de aceleración de los ejes X, Y y Z para detectar la posición actual del robot.

Funcionalidades y requerimientos cumplidos

Cumplidos completamente:

- Control del robot cuadrúpedo mediante una interfaz web.
- Implementación de una red WiFi mediante el NodeMCU para servir la interfaz al usuario.
- Control de servos a través del controlador PCA9685 y del NodeMCU utilizando el protocolo I2C.
- Integración del sensor ADXL345 para detectar la posición del robot en todo momento.

Cumplidos parcialmente:

- Optimización de los movimientos del robot, que puede requerir ajustes adicionales en la lógica de control y los ángulos de los servos para mejorar la estabilidad.
- Resaltado dinámico de los servos activos en la interfaz web mediante texto actualizado periódicamente.

Agregados durante el desarrollo:

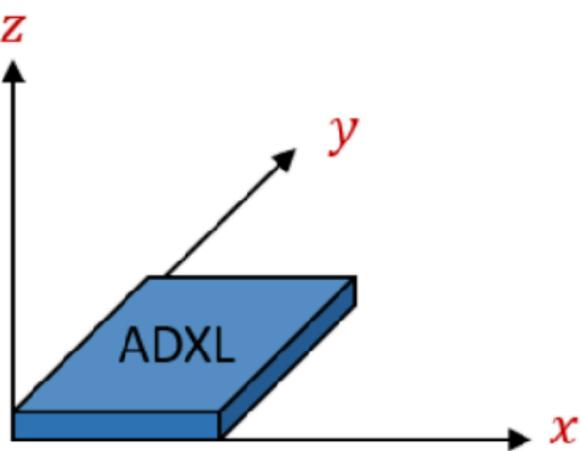
- Implementación de una lógica específica para la inclinación del robot, que lo obliga a seguir inclinándose hasta que el sensor ADXL345 detecte los ángulos límites de inclinación.

3. Sensor ADXL345

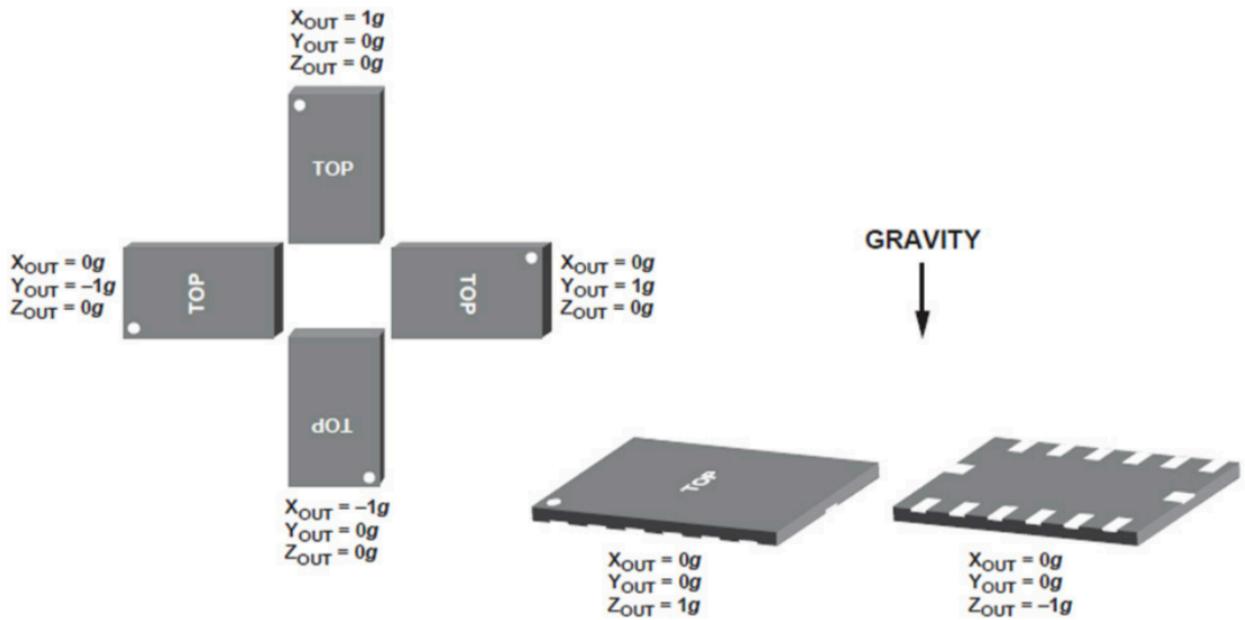
Como parte del proyecto de investigación desarrollado a partir del proyecto original, se implementaron las funciones de estabilidad del Micro Quadruple utilizando el acelerómetro ADXL345. El sensor de inclinación utilizado originalmente era el SW520D, en donde su lógica consistía en retornar 0 si no detectaba inclinación y un 1 en caso contrario, sin ofrecer información acerca del grado de inclinación o vibración. La implementación del ADXL345 en el proyecto conlleva mejoras en cuanto al control y la estabilidad del Micro Quadruple, ya que éste mide la aceleración en los tres ejes (X, Y, Z) y proporciona datos precisos que permiten calcular la inclinación, orientación y movimiento del robot con un mayor nivel de detalle.

3.1 Características

El sensor ADXL345 es un acelerómetro micromecanizado capacitivo de 3 ejes independientes que permite detectar la aceleración en los ejes X, Y y Z (en un plano tridimensional) tal como se muestra en la siguiente figura:

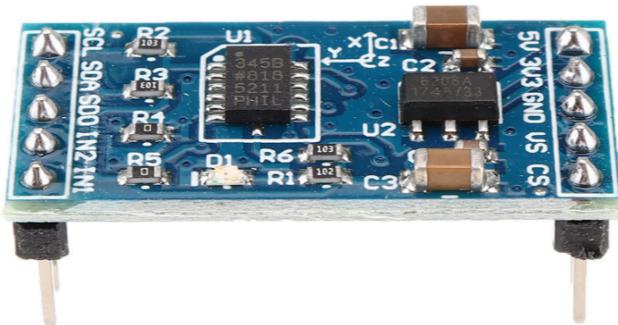


Los valores de los diferentes ejes varían según la posición del sensor, donde X e Y miden en el plano horizontal y Z en el plano vertical, tal como se muestra en la siguiente imagen:



Este sensor es un dispositivo de ultra bajo consumo, con $45 \mu\text{A}$ en modo de medición y $0,1 \mu\text{A}$ en standby. Su tensión de alimentación es de bajo voltaje que varía entre 2.0V y 3.6V y su comunicación con el microcontrolador puede realizarse tanto por protocolo SPI como por I₂C, contando con cierta flexibilidad en cuanto a su implementación y obtención de datos.

El ADXL345 cuenta con los siguientes pines:



- **5V:** alimentación principal del módulo con 5V . Está conectado a un regulador de voltaje integrado que reduce el voltaje de 5V a los 3.3V necesarios para el correcto funcionamiento del sensor.
- **3V3:** alimentación principal del módulo con 3.3V , evitando el uso del regulador de voltaje.

- GND: conexión a tierra.
- VS (Voltage Supply): pin opcional que permite alimentar los circuitos analógicos del sensor de manera independiente, lo que puede mejorar la precisión de las mediciones. En caso de no estar conectado a una fuente separada, se conecta internamente con VCC, por lo que los circuitos digitales y analógicos del sensor comparten la misma fuente de voltaje.
- CS (Chip Select): determina el protocolo de comunicación que el sensor utilizará con el microcontrolador. Si se deja en estado flotante o se conecta con VCC, el sensor funcionará con protocolo I2C, y si se conecta a GND, funcionará con protocolo SPI.
- SDA: pin de datos para comunicación I2C.
- SCL: pin del reloj para comunicación I2C.
- SD0: pin de salida de datos para comunicación SPI. En modo I2C, define la dirección del sensor: 0x53 (conectado a GND o estado flotante) o 0x1D (conectado a VCC).
- IN1 e IN2: pines de interrupción configurables para detectar ciertas actividades y salida de datos del sensor. Su configuración se basa en el valor del registro INT_MAP, si es un 0 la interrupción la emite IN1, y en caso contrario IN2.

El sensor ADXL345 presenta diferentes rangos de medida que afectan a la resolución y la precisión de los valores obtenidos de los diferentes ejes. Estos son configurados en el registro DATA_FORMAT, y se detallan en el siguiente cuadro:

Rango	Rango de medición	Sensibilidad (LSB/g)	Resolución	Campo registro	Funcionalidad
±2g	-512 a 511	256	13 bits	0x08	Movimientos pequeños y alta precisión
±4g	-1024 a 1023	128	12 bits	0x09	Movimientos moderados
±8g	-2048 a 2047	64	11 bits	0x0A	Movimientos grandes y baja precisión
±16g	-4096 a 4095	32	10 bits	0x0B	Grandes cambios de aceleración y caídas fuertes

Este valor obtenido (de la columna Rango de medición) es un número entero que representa una porción de la aceleración en unidades de g (fuerza de gravedad) ya que surge como respuesta a las fuerzas de aceleración de cada eje de forma independiente. Por lo tanto, si se quiere tener la muestra de la aceleración física real, debe ser convertido a la unidad física correspondiente. Esta conversión se realiza según el rango elegido y su sensibilidad predeterminada, según el siguiente cálculo:

$$\text{Aceleración en g} = \text{Valor entero} / \text{Sensibilidad en LSB/g}$$

Obtenidas las lecturas de cada coordenada se pueden calcular los ángulos de cada eje en un rango de -90° a 90° mediante las siguientes operaciones trigonométricas:

Eje x:

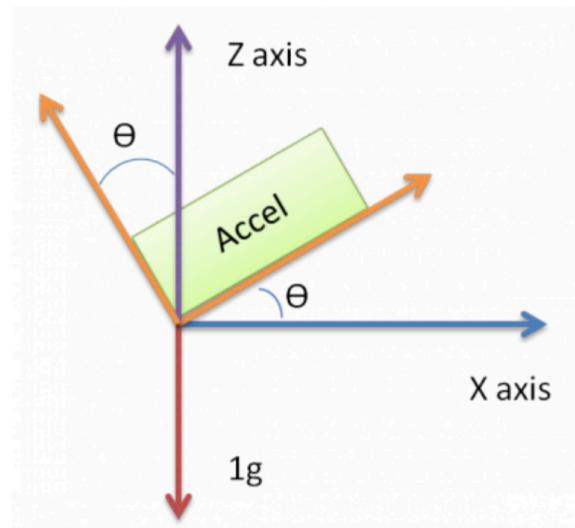
$$\theta_X = \arctan \left(\frac{X}{\sqrt{Y^2 + Z^2}} \right)$$

Eje y:

$$\theta_Y = \arctan \left(\frac{Y}{\sqrt{X^2 + Z^2}} \right)$$

Eje z:

$$\theta_Z = \arctan \left(\frac{Z}{\sqrt{X^2 + Y^2}} \right)$$



3.2 Implementación en el proyecto

Inicialmente, para lograr la comunicación I2C con el microcontrolador, se requirió implementar I2C por software con la librería SoftwareWire ya que los pines de I2C por hardware del NodeMCU (D1 y D2) se encontraban en uso con el controlador de servos PCA9685. Por lo tanto, se conectaron los pines SCL/SDA del ADXL345 con los pines D3 y D4 del microcontrolador y se los configuró para permitir la comunicación I2C. En este proyecto, únicamente se conectaron los pines 3V3, GND, SDA y SCL, ya que las funciones de los demás pines no eran requeridas para este caso.

Para la obtención de medidas del sensor se leyeron directamente los registros en donde se almacenan los valores enteros de cada eje por separado y luego manualmente se convirtieron a su unidad física correspondiente de fuerza de gravedad. Este paso se pudo haber evitado al existir la biblioteca Adafruit ADXL345 que mediante sus funciones ya implementadas retorna el valor convertido directamente a g, pero por problemas de compatibilidad con la biblioteca SoftwareWire no se pudo llevar a cabo.

Al empezar las pruebas, se seleccionó el rango $\pm 2g$ para mayor precisión en las lecturas realizadas ya que los movimientos que realiza el robot son pequeños y fluidos. A medida que se avanzó en su desarrollo, se notó que habían ciertos desfasajes en las medidas obtenidas de los diferentes ejes debido a la existencia de un offset interno del sensor que puede llegar a producir pequeños cambios en la lectura de los datos. Este error es especificado por el fabricante en el datasheet propio del sensor y puede ser de hasta ± 60 de diferencia en cada eje. Por lo tanto, se procedió a calibrar el sensor en una posición conocida, como es de manera plana y estable en una mesa, en donde los valores deberían tender a $X = 0$, $Y = 0$ y $Z = 256$. Esto debido a que los ejes X e Y no están experimentando aceleración debido a la gravedad, por lo que tienden a 0. El eje Z se encuentra alineado con la fuerza gravitacional de la tierra por lo cual da un valor cercano a 256 (equivalente a 1g). Partiendo de esto, se llegó al valor de los diferentes offsets que influían en la medición de cada eje, estos eran +12 en X, -6 en Y, y +46 en Z. De esta forma se corrigieron los valores para obtener medidas más exactas y precisas.

Se decidió colocar el sensor boca abajo en la parte superior del Micro Quadruple por cuestiones de diseño y conexión a los demás componentes. Por lo tanto, las medidas obtenidas del sensor del eje Z tienen el signo opuesto al de su posición normal (ya que indican la orientación vertical del sensor), por lo que se invirtió el signo de cada lectura para simular que el sensor se encuentra en su posición normal al leer los valores. Las medidas de los ejes X e Y no se modificaron ya que miden en el plano horizontal y no son afectadas por este cambio.

Se incorporó en la interfaz web un recuadro en donde se detalla la posición del robot a partir de los ángulos medidos en cada eje, para que el usuario tenga una guía visual de cómo cambian estos ángulos a medida que el robot camina o se inclina.

Posición del Robot

Eje X: 0°

Eje Y: 0°

Eje Z: 0°

Cuando el robot se encuentra en su posición inicial de arranque, los ángulos medidos son aproximadamente $X = 0^\circ$, $Y = -2^\circ$ y $Z = 88^\circ$. El sensor está casi alineado perfectamente en el plano horizontal (por lo que tiende a $X = 0^\circ$ y $Y = 0^\circ$) y en el plano vertical (que tiende a $Z = 90^\circ$) pero por una ligera orientación del cuerpo del robot se notan estas diferencias.

Para realizar la inclinación del robot, se realizó el mismo procedimiento que en el proyecto anterior con el sensor SW520D, pero en este caso con los ángulos límites que retorna el sensor ADXL345. Cuando se presiona el botón correspondiente para iniciar la inclinación, el robot comienza a desnivelarse con las patas delanteras hasta los ángulos límites que permitan a los servomotores descender sin perder la estabilidad del robot. Mediante varias pruebas, se determinaron que éstos ángulos son aproximadamente $X = 11^\circ$, $Y = -3^\circ$, y $Z = 79^\circ$, y cuando se llega a este límite el robot vuelve automáticamente a su posición inicial. Este mecanismo asegura que el robot sea capaz de adaptarse y estabilizarse sin intervención manual ante el estado de inclinación iniciada.

Estos ángulos son consistentes con el movimiento del robot, ya que al desplazarse hacia adelante en su proceso de inclinación, genera una aceleración mayor en el eje X, incrementando su valor. El valor negativo en el eje Y se incrementa, ya que detecta una mayor componente de aceleración en la dirección negativa del eje. Esto refleja una inclinación hacia adelante en el eje X, provocando un cambio en la magnitud de Y. Luego, el eje Z disminuye su valor ya que al inclinarse hacia adelante, reduce la aceleración medida en este eje ya que ya no se encuentra perfectamente alineado con la gravedad de forma vertical.

4. Documentación del Software del proyecto

Organización del Código Fuente

El proyecto está organizado de forma modular, utilizando PlatformIO y Visual Studio Code, lo que facilita la gestión del código fuente y las dependencias. A continuación, se detallan las principales funciones implementadas, cómo cumplen con los requisitos del sistema y los métodos utilizados para validar su funcionalidad.

Estructura de directorios

Carpeta/Archivo	Descripción
platformio.ini	Archivo principal de configuración de PlatformIO. Contiene los parámetros de plataforma, librerías utilizadas y configuración de compilación.
src/main.cpp	Archivo principal del proyecto que implementa todas las funcionalidades del robot cuadrúpedo, incluyendo el control de los servos, la comunicación con el servidor web y el manejo del sensor ADXL345.
ib/	Directorio destinado a librerías privadas específicas. No se utilizó en este caso
include/	Carpeta opcional para declarar encabezados, si se separan funciones adicionales. No se utilizó en este caso.
.pio/	Carpeta generada automáticamente por PlatformIO para la compilación del código.

Instalación y Configuración del Entorno de Desarrollo

Para la instalación y configuración del entorno se comenzó por importar el proyecto del grupo anterior [tpII/2022-A.3-Micro-Quadruple: 2022 A.3-Micro-Quadruple](#).

A partir de este se decidió continuar el uso de la misma plataforma, Visual Studio en conjunto con la herramienta PlatformIO. Para lograr utilizar PlatformIO, luego de instalar la extensión, se configuró de la siguiente forma el archivo 'platformio.ini' de la siguiente manera:

```
[env:nodemcuv2]
platform = espressif8266
board = nodemcuv2
framework = arduino
monitor_speed = 115200
lib_deps =
    esphome/ESPAsyncWebServer-esphome@^3.2.2
    esphome/ESPAsyncTCP-esphome@^2.0.0
    adafruit/Adafruit PWM Servo Driver Library@^2.4.0
    testato/SoftwareWire@^1.6.0
    adafruit/Adafruit ADXL345@^1.3.4
    adafruit/Adafruit Unified Sensor@^1.1.14
```

Donde se especificó la plataforma, en nuestro caso el ESP8266, la placa 'nodemcu2', el framework 'arduino' y el BaudRate de 115200.

Además, en este archivo se configuran las librerías utilizadas para el proyecto:

- **ESPAsyncWebServer:** Esta librería permite la creación de un servidor web asincrónico en ESP8266. Permite manejar solicitudes simultáneamente sin bloquear la ejecución del programa principal. Es el encargado de la gestión de solicitudes HTTP (GET en nuestro caso).
- **ESPAsyncTCP:** Es una librería base requerida por ESPAsyncWebServer. Proporciona las herramientas necesarias para manejar conexiones TCP asincrónicas en el ESP8266. Actúa como el núcleo y permite la comunicación entre el servidor web y el/los clientes.

- **Adafruit PWM Servo Driver:** se encarga de facilitar la comunicación de servos por medio del PCA9685 mediante el protocolo I2C. Permite controlar hasta 16 salidas PWM (en nuestro caso solo se necesitan 12). Se encarga de generar las señales PWM precisas para controlar los servomotores, configurando la frecuencia y el ancho del pulso PWM.
 - **SoftwareWire:** Esta librería es una implementación del protocolo I2C utilizando software. Permite la comunicación con dispositivos I2C en plataformas que no tienen soporte nativo para I2C, o cuando se desean utilizar pines diferentes a los predeterminados en el hardware.
 - **Adafruit_ADXL345:** Esta librería facilita la comunicación con el sensor de aceleración ADXL345 a través del protocolo I2C. Permite leer los valores de aceleración en los tres ejes (X, Y, Z) y configurar parámetros como la sensibilidad y el rango de medición del sensor.
 - **Adafruit Unified Sensor:** Proporciona una interfaz estándar para leer datos de diversos sensores. Es utilizada como base para diferentes librerías de sensores de Adafruit, permitiendo un formato unificado y sencillo para interactuar con varios tipos de sensores.
 - **Wire:** Esta librería proporciona soporte para la comunicación I2C en Arduino. Ya está incluida en el entorno de desarrollo de Arduino y permite gestionar la comunicación con dispositivos I2C sin necesidad de instalar nada adicional.
 - **Math:** Proporciona funciones matemáticas comunes necesarias para realizar cálculos de ángulos y otros procesos matemáticos. También viene incluida con el entorno de Arduino y no necesita instalación.
-

Configuración WiFi

La placa NodeMCU está configurada como Access Point (AP), lo que permite que otros dispositivos se conecten directamente al microcontrolador mediante WiFi, sin necesidad de un router externo. Para la configuración del mismo se tuvo en cuenta:

- **Configurar las credenciales del AP** mediante el comando ‘WiFi.softAP(ssid, password)’
 - SSID: "NodeMCU_AP"
 - Contraseña: "12345678"

- **Obtención de la dirección IP del Access Point:** La línea WiFi.softAPIP() obtiene la dirección IP asignada al Access Point, la cual es 192.168.4.1. Esta dirección IP se utiliza como entrada en el navegador web del usuario para acceder a la interfaz web y controlar el robot.
 - **Servidor Web Asíncrono:** Utilizando la librería ESPAsyncWebServer, el microcontrolador ejecuta un servidor web en el puerto 80. Este servidor gestiona las solicitudes HTTP (GET) enviadas desde el cliente (usuario) para realizar acciones como mover el robot, cambiar su posición inicial, recibir las lecturas del sensor ADXL345, etc.
-

Funciones implementadas

El archivo principal del proyecto ‘main.cpp’ implementa las siguientes funciones para controlar el comportamiento del robot:

Función	Objetivo
initPCA9685()	Inicializa el controlador PCA9685 y establece la frecuencia de los pulsos PWM de 60Hz para los servos. Se llama al inicio en setup() para inicializar el hardware del controlador de servos.
moveServo(uint8_t, uint16_t)	Mueve un servo específico a una posición determinada, indicando el ángulo o posición mediante señales PWM.
position() y position2()	Lleva el robot a la posición inicial. position() para acomodar las patas delanteras y position2() para acomodar las patas traseras.
walkSequence()	Ejecuta la lógica de caminata del robot. Divide el movimiento en pasos (sequenceStep) para mover cada pata en el orden correcto y realiza hasta 3 secuencias del mismo.
tiltRobot()	Inclina el robot hasta que el sensor ADXL345 retorna los ángulos límites de inclinación.

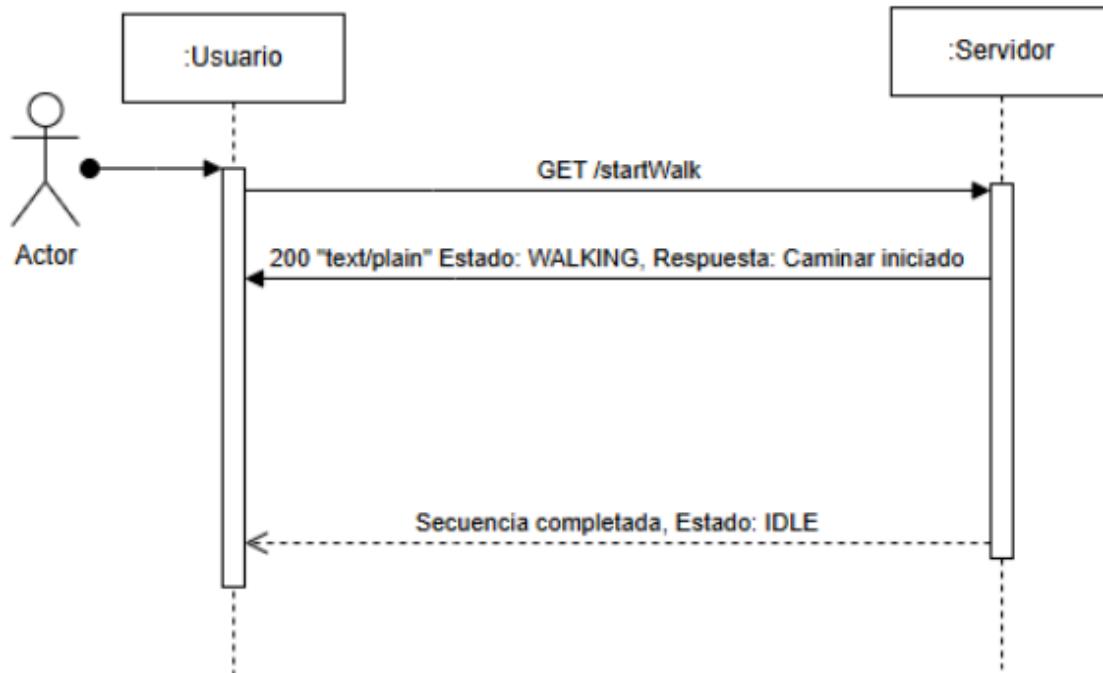
setup()	Configura el Access Point, el servidor web y el PCA9685.
loop()	Ejecuta el estado actual del robot, como caminar o inclinarse.
server.on()	Define las rutas HTTP para cada acción que puede realizar el robot.
initializeADXL345()	Configura el sensor ADXL345 para que entre en modo de medición, verifica la conexión y establece el rango de medición en ±2g.
readAcceleration(int16_t , int16_t , int16_t)	Lee los valores de aceleración de los tres ejes (X, Y, Z) del sensor ADXL345 y aplica un ajuste de offset.
calculateAngles (int16_t, int16_t, int16_t ,int, int, int)	Calcula los ángulos de inclinación respecto a los ejes X, Y y Z utilizando los valores de aceleración leídos y las funciones trigonométricas.

Relación con los Requisitos

- **Control de Servos:** Las funciones moveServo() y walkSequence() permiten controlar el movimiento preciso de los servos según lo requerido para la caminata y las posiciones iniciales.
- **Comunicación WiFi:** Utilizando las bibliotecas ESPAsyncWebServer y ESPAsyncTCP, se implementó un servidor web que permite la interacción remota con el robot.
- **Manejo del sensor ADXL345:** La función tiltRobot() utiliza los valores leídos por el sensor para ajustar la inclinación del robot, garantizando su estabilidad en todo momento.

Caso de Uso: Iniciar Caminar

Este caso de uso describe el flujo de acciones involucradas cuando un usuario interactúa con la interfaz web del sistema para iniciar el movimiento del robot cuadrúpedo. El objetivo principal es permitir que el usuario envíe un comando desde la interfaz HTML, el cual es procesado por el servidor web alojado en el NodeMCU, para activar la secuencia de caminata del robot. Durante este proceso, se verifican estados del sistema, se controlan los servos mediante comandos I2C y se proporciona retroalimentación al usuario en tiempo real.



Se describe el proceso cuando el usuario presiona el botón "Iniciar caminar" en la interfaz HTML:

1. El usuario envía una solicitud HTTP GET /startWalk desde la interfaz HTML al servidor web en el NodeMCU.
2. El servidor verifica el estado del robot (IDLE o WALKING).
3. Si el estado es IDLE, el servidor:
 - a. Cambia el estado a WALKING.
 - b. Llama a la función walkSequence() para iniciar el movimiento.
4. La función walkSequence() envía comandos I2C al PCA9685 para controlar los servos en la secuencia programada.

5. El servidor responde al cliente con un mensaje de confirmación: "Caminar iniciado".
 6. Al completar la secuencia, el estado del robot vuelve a IDLE.
-

Pruebas

Para garantizar la funcionalidad del robot cuadrúpedo, el proceso de validación y pruebas se realizó en varias etapas, incrementando progresivamente la complejidad:

Pruebas individuales de servos:

- Inicialmente, cada servo fue probado de manera individual. Se implementó una interfaz básica con un botón dedicado para controlar cada servo.
- Esto permitió verificar la correcta conexión eléctrica, respuesta a los comandos PWM enviados desde el PCA9685 y el rango de movimiento de cada servo.

Pruebas de patas:

- Una vez validados los servos individuales, las pruebas se ampliaron para abarcar las patas completas.
- Se creó una interfaz en la que un botón controlaba todos los servos correspondientes a una pata específica.
- Estas pruebas ayudaron a identificar problemas de sincronización o ajustes de posiciones iniciales para cada pata.

Prueba completa de la Caminata:

- En la etapa final, se integraron todas las funcionalidades en un único comando de caminata accesible desde la interfaz web.
- Al presionar un botón dedicado, se ejecutaba la secuencia de movimiento completa del robot, validando la coordinación de las patas y la estabilidad durante el desplazamiento.

5. Documentación relacionada

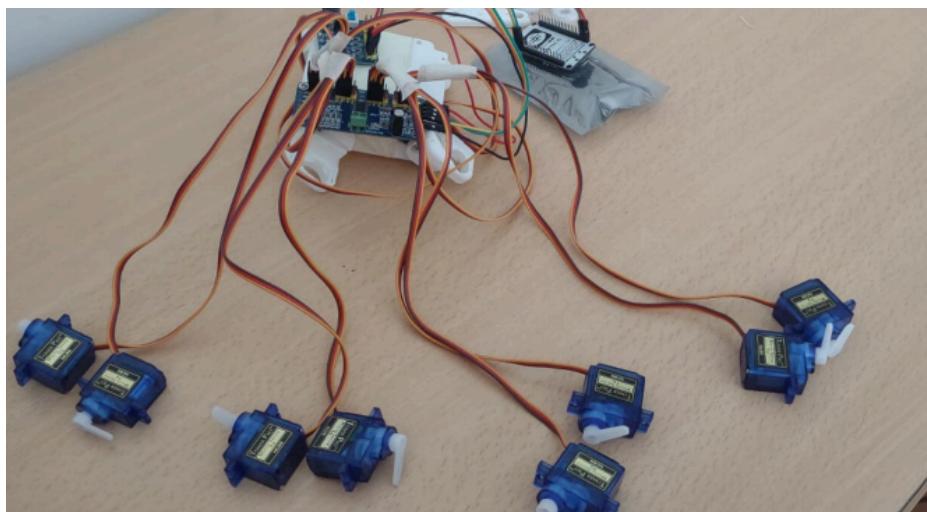
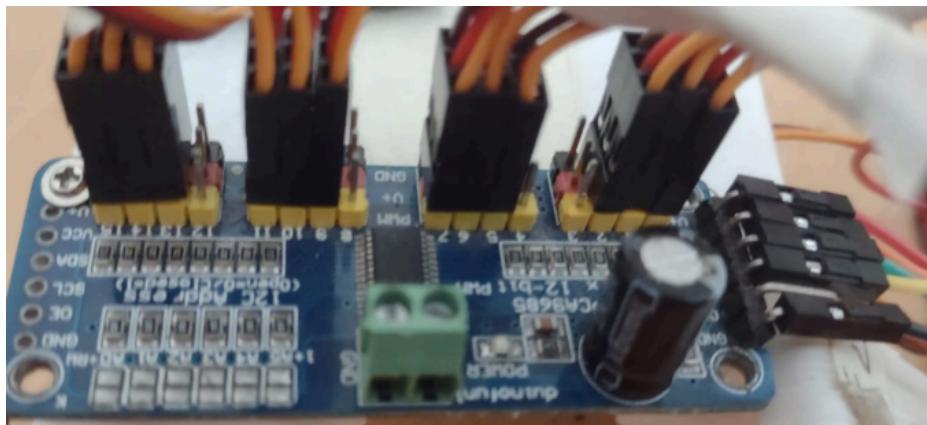
Enlaces del proyecto:

- Sistema de almacenamiento y versionado: [tpII/2024-A3-MicroQuadruple: Continuación de Micro Quadruple](#)
- Bitácora: Bitácora Proyecto A3: [Home · tpII/2024-A3-MicroQuadruple Wiki](#)

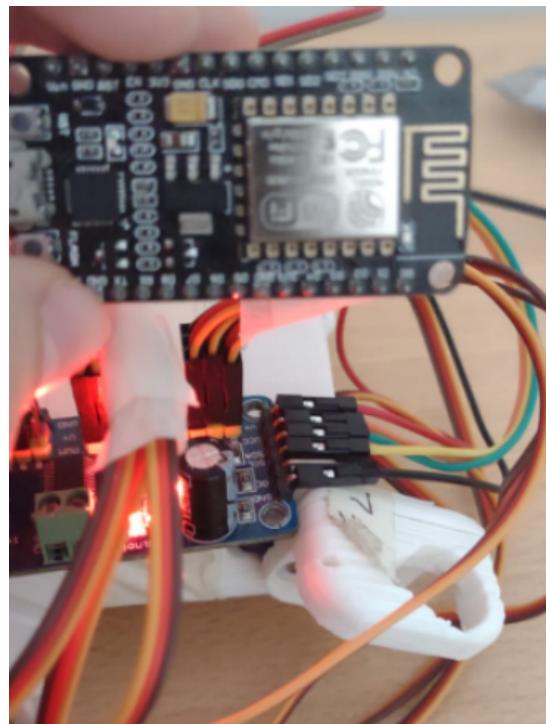
Videos finales del funcionamiento del robot

Fotos de las partes físicas

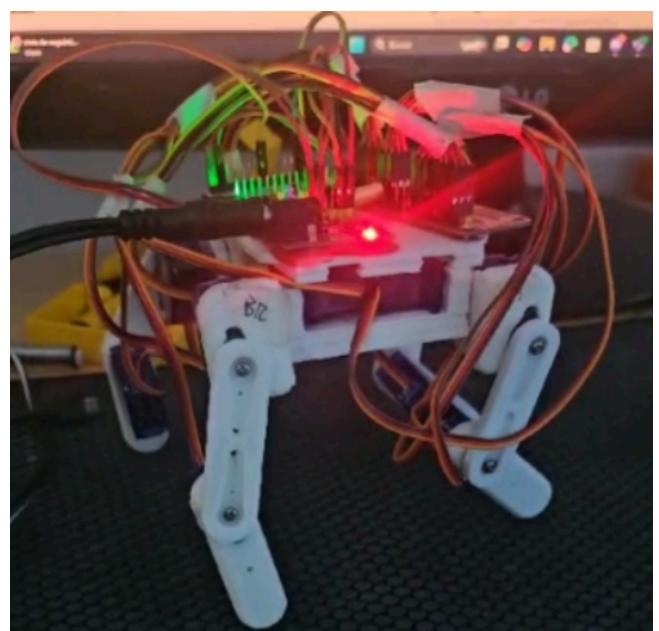
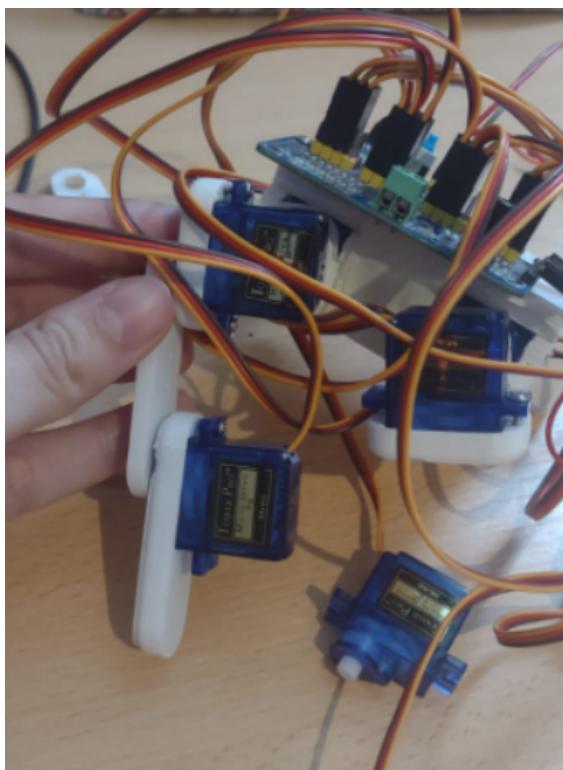
Inicialmente, se comenzó por el montaje de partes del robot como se puede observar en la siguiente imagen, esto se hizo así para realizar el testeo inicial de cada servo.



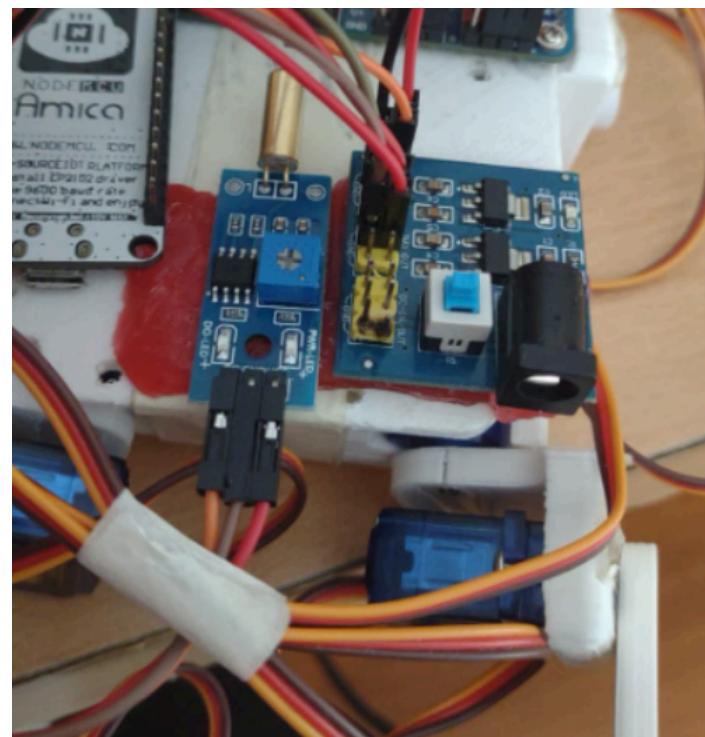
Con este test se encontraron dos servos que no funcionaban y uno que estaba defectuoso. Luego de informar a la cátedra y obtener el cambio de servos, se continúo con los testeos del NodeMCU y del sensor.



Una vez completados los diferentes tests, se comenzó con el ensamblado de las patas y finalmente se terminó de armar el robot completo, como puede observarse a continuación:



A continuación, se verán fotos que muestran más a detalle como se encuentran los componentes adheridos al “lomo” del robot:



Y por último, se muestra una foto del robot cuando se ejecuta el proceso de inclinación, justo antes de que retome su posición original:

A continuación se comparte una foto de la interfaz web que implementa el proyecto:

La imagen presentada corresponde a la interfaz web desarrollada para el control del robot del proyecto. Esta interfaz permite al usuario interactuar de manera sencilla con las funcionalidades principales del robot. Está diseñada para ser accesible desde cualquier dispositivo conectado al Access Point (AP) del NodeMCU.

En la sección superior se encuentra el botón “iniciar Caminar”, este botón envía una solicitud HTTP GET al servidor para ejecutar la función walkSequence() en el NodeMCU. Activa la secuencia de caminata del robot, coordinando las patas para lograr un movimiento fluido.

Luego se puede ver el botón de "Posición de arranque", este envía comandos para ejecutar las funciones position() y position2(), llevando las patas del robot a su posición inicial predefinida. y por último, se encuentra el botón "Inclinarse", el cual permite al robot inclinarse hasta detectar los ángulos límites obtenidos por el sensor ADXL345. Esta funcionalidad es útil para validar la inclinación y estabilidad del robot.

En la sección del centro se encuentra detallada la posición del robot a partir de los ángulos en sus ejes X, Y y Z respectivamente. Esto sirve de guía visual al usuario para poder saber cómo se encuentra posicionado el robot en todo momento.

En la sección inferior se encuentra la cuadrícula de servos. Los cuadros representan cada uno de los servos del robot, numerados de acuerdo con su asignación en el controlador PCA9685. Cada cuadro debería poder actualizar su estado dinámicamente según el servo activo, proporcionando información visual sobre qué servo está en movimiento, pero no se logró completar la visualización de esto.

6. Bibliografía

[Mini Servo Tower Pro Sg90 9g Robotica Arduino Servomotor | MercadoLibre](#)

[How to find offset for the 3 -axis using the ADXL 345 digital accelerometer and how do i convert the data i get into G? - Q&A - MEMS Inertial Sensors - EngineerZone](#)

[Nodemcu Wifi Lua V2 Esp8266 Gpio Esp12f 4mb Uart | MercadoLibre](#)

[Controlador Servos 16 Canales I2c Pca9685 Pwm Arduino Nubeo | Cuotas al mismo precio que publicaste](#)

7. Apéndice: Materiales y presupuesto

En esta etapa final del proyecto, se consolidaron los elementos necesarios para construir el robot cuadrúpedo "Micro Quadruple". Inicialmente, se había previsto utilizar tanto un microcontrolador Arduino UNO como un módulo Wi-Fi ESP8266. Sin embargo, tras evaluar las capacidades del ESP8266, se determinó que este era capaz de cumplir tanto las funciones de procesamiento como de conectividad Wi-Fi. Esto permitió simplificar el diseño y optimizar los recursos del proyecto. A continuación, se detalla la lista actualizada de materiales y costos para la construcción del sistema:

Elemento	Cantidad	Costo unitario (ARS)	Costo total (ARS)
Servomotor es SG90	12 + 3 extra	3.390	50.850
Sensor acelerómetro ADXL345	1	4.500	4.500
Módulo Wi-Fi ESP8266	1	8.239	8.239
Controlador de servos PCA9685	1	12.999	12.999
Fuente de alimentación 5V	1	8.944	8.944
Estructura del robot	1	18.000	18.000

- **Pruebas y ajustes:** Durante las pruebas, se detectaron diferencias en el tamaño de los servomotores respecto a los utilizados en versiones previas del proyecto. Esto ocasionó que algunas piezas de la estructura, particularmente las patas, fueran incompatibles. Se realizaron nuevas mediciones y se reimprimieron las piezas mediante impresión 3D para ajustarlas a las especificaciones actuales.
- **Servomotores adicionales:** La cátedra proporcionó tres servomotores SG90 adicionales para garantizar el correcto funcionamiento del robot, especialmente durante las pruebas, donde algunos servos presentaron fallos debido a un desgaste prematuro.

- **Actualización del hardware:** Todos los elementos necesarios fueron proporcionados por la cátedra en la semana del 19/9, incluyendo los servomotores adicionales y el PCA9685.