UNIVERSIDAD MODELO

Escuela de ingeniería

Ingeniería Biomédica

Moreno Suarez, Sebastian Alexander

Domínguez Ugarte, Jose Eduardo

Ramirez Aguilar, Juan Carlos

Carrillo Pantoja, Ana Karen

Barraza Rovira, Arturo

Flores Rodríguez, Sofia Sarahi

Teleoperación Bilateral

8vo semestre

Robótica

Introducción

Los exoesqueletos humanos constituyen una de las principales innovaciones tecnológicas en el ámbito de la asistencia motriz, la rehabilitación y la mejora del rendimiento físico. Diseñados para apoyar la movilidad en personas con discapacidad, reducir la fatiga en tareas repetitivas y potenciar habilidades físicas en distintos contextos, estos dispositivos enfrentan limitaciones importantes en su desempeño, principalmente debido a efectos mecánicos como la fricción, la inercia y la fuerza gravitatoria.

Para alcanzar un funcionamiento eficiente y seguro, es fundamental que los exoesqueletos integren mecanismos de compensación que contrarresten dichas fuerzas. En este sentido, los actuadores, que son los elementos encargados de generar el movimiento en las articulaciones, desempeñan un rol esencial, ya que permiten lograr desplazamientos más suaves y naturales. Por esto, en esta parte de la práctica se estará realizando una conexión bilateral entre dos arduinos para poder copiar el movimiento, el uno del otro, o más específicamente que el controlador maestro, pueda mover al controlador esclavo.

Marco teórico

Motores

Los motores eléctricos son dispositivos que convierten la energía eléctrica en energía mecánica, permitiendo el movimiento en diversas aplicaciones. Entre los tipos más comunes se encuentran los motores de corriente continua (DC), de corriente alterna (AC), los motores paso a paso y los servomotores. En proyectos que involucran microcontroladores como Arduino, los motores DC y los servomotores son frecuentemente utilizados debido a su simplicidad en el control y su eficiencia. La modulación por ancho de pulsos (PWM) es una técnica comúnmente empleada para regular la velocidad y posición de estos motores, ajustando el ciclo de trabajo de una señal periódica para controlar la energía entregada al motor.

Encoders

Los encoders son sensores que proporcionan información sobre la posición, velocidad y dirección de un eje rotativo o lineal. Se clasifican principalmente en encoders incrementales y absolutos. Los encoders incrementales generan pulsos en función del movimiento del eje, permitiendo medir cambios relativos en la posición. Por otro lado, los encoders absolutos ofrecen una referencia única para cada posición del eje, proporcionando datos exactos de la posición en todo momento. En sistemas de control de motores, los encoders son esenciales para implementar estrategias de retroalimentación que aseguren precisión y estabilidad en el movimiento.

Control PID

El control proporcional-integral-derivativo (PID) es una metodología ampliamente utilizada en sistemas de control automático para mantener una variable en un valor deseado. Este controlador combina tres acciones:

- Proporcional (P): Responde al error actual entre la referencia y la medida, ajustando la salida en proporción a este error.

- Integral (I): Considera la suma de errores pasados, corrigiendo desviaciones acumuladas a lo largo del tiempo.
- Derivativa (D): Anticipa errores futuros basándose en la tasa de cambio del error, proporcionando una acción correctiva que mejora la estabilidad del sistema.

La implementación de un controlador PID en sistemas con motores y encoders permite un control preciso de variables como la posición y la velocidad, mejorando el rendimiento y la respuesta dinámica del sistema.

Motorreductor JGB37-3530

El motorreductor JGB37-3530 es un motor de corriente continua que incorpora una caja reductora, diseñada para ofrecer un alto torque a bajas revoluciones por minuto (RPM). Este tipo de motor es ideal para aplicaciones que requieren fuerza y precisión a bajas velocidades, como en proyectos de robótica y automatización. Además, la posibilidad de integrar un encoder en el eje de salida permite implementar sistemas de control de retroalimentación, mejorando la exactitud en el control de posición y velocidad.

Puente H

Un puente H es un circuito electrónico que permite controlar la dirección de giro de un motor de corriente continua, facilitando su movimiento en ambas direcciones. Este circuito se compone de cuatro interruptores (como transistores o MOSFETs) que gestionan el flujo de corriente hacia el motor. Implementaciones comunes de puentes H, como el controlador L298N, permiten manejar motores que requieren corrientes elevadas, siendo esenciales en aplicaciones donde se necesita invertir la dirección del motor de manera eficiente.

Arduino MEGA

El Arduino MEGA 2560 es una placa de desarrollo basada en el microcontrolador ATmega2560, diseñada para proyectos que demandan múltiples entradas y salidas. Sus características principales incluyen 54 pines digitales de I/O, 16 entradas analógicas y 4 puertos de comunicación serial (UART). Estas especificaciones la convierten en una opción

adecuada para proyectos de robótica avanzada y control de motores con retroalimentación de encoders, permitiendo la gestión simultánea de múltiples dispositivos y sensores.

RPM (Revoluciones Por Minuto)

Las revoluciones por minuto (RPM) son una unidad de medida que indica la cantidad de vueltas que un objeto, como un eje de motor, realiza en un minuto. Esta métrica es fundamental para determinar la velocidad de rotación en aplicaciones industriales, automotrices y de robótica, siendo crucial para el diseño y control de sistemas mecánicos que requieren precisión en el movimiento.

INA219

El INA219 es un sensor digital diseñado para medir corriente, voltaje y potencia en sistemas eléctricos. Este dispositivo es especialmente útil en proyectos que requieren un monitoreo preciso del consumo energético, como aplicaciones de energía renovable, sistemas de gestión de baterías y dispositivos electrónicos portátiles.

Características principales del INA219:

Medición de corriente: El INA219 puede medir corrientes de hasta ±3,2 A. Utiliza una resistencia de derivación interna de 0,1 ohmios con una precisión del 1% para detectar la caída de voltaje y calcular la corriente que atraviesa el circuito.

Medición de voltaje: Este sensor es capaz de medir voltajes de hasta 26 V en el bus, lo que lo hace adecuado para una amplia gama de aplicaciones.

Resolución y precisión: Gracias a su convertidor analógico-digital (ADC) interno de 12 bits, el INA219 ofrece una resolución de 0,8 mA en el rango de ±3,2 A, lo que permite mediciones precisas del consumo de corriente.

Comunicación I2C: El dispositivo se comunica con microcontroladores a través de la interfaz I2C, utilizando direcciones de 7 bits que pueden configurarse entre 0x40, 0x41, 0x44 y 0x45 mediante puentes en los pines A0 y A1.

Flexibilidad en la resistencia shunt: Aunque el sensor viene con una resistencia shunt de 0,1 ohmios, es posible reemplazarla por otra de diferente valor para ajustar el rango de medición de corriente según las necesidades específicas del proyecto.

La integración del INA219 en proyectos con microcontroladores, como Arduino, permite a los desarrolladores monitorear en tiempo real el consumo de energía de diversos componentes, facilitando la implementación de sistemas más eficientes y seguros.

Cancelación de Gravedad

La cancelación de gravedad se enfoca en compensar el peso del exoesqueleto y del usuario para minimizar el esfuerzo al moverse. Esta compensación se puede lograr mediante dos enfoques principales:

- **Sistemas pasivos**: Utilizan mecanismos mecánicos como resortes, contrapesos o estructuras elásticas para redistribuir la carga y reducir la demanda energética del usuario.
- **Sistemas activos**: Emplean motores y actuadores controlados en tiempo real para generar fuerzas opuestas a la gravedad, mejorando la asistencia en movimientos complejos y prolongados.

Estos sistemas han sido clave en aplicaciones como la rehabilitación neuromuscular, la asistencia a personas con movilidad reducida y la reducción del esfuerzo físico en entornos industriales y militares

Cancelación de Inercia

Para lograr movimientos más naturales, es fundamental reducir el esfuerzo necesario para cambiar de dirección o velocidad. Esto se consigue con tres enfoques tecnológicos:

- **Modelado dinámico y control predictivo**: Algoritmos que anticipan el movimiento del usuario y generan fuerzas opuestas para reducir la sensación de peso.
- **Sensores y actuadores de alta velocidad**: Motores eléctricos e hidráulicos capaces de responder en tiempo real para compensar cambios de inercia.
- **Aprendizaje adaptativo basado en IA**: Sistemas que analizan los patrones de movimiento del usuario y ajustan la asistencia para mejorar la fluidez de los desplazamientos.

La cancelación efectiva de inercia es crucial en dispositivos diseñados para la asistencia diaria y la rehabilitación, permitiendo que los usuarios se muevan con mayor naturalidad y menor esfuerzo.

Cancelación de Fricción

La fricción es una de las principales fuerzas resistivas en un exoesqueleto, afectando su eficiencia y aumentando el consumo energético del usuario. Para contrarrestarla, se emplean técnicas como:

- **Compensación activa**: Aplicación de fuerzas controladas mediante actuadores para contrarrestar la fricción en articulaciones y motores.
- **Uso de materiales de baja fricción**: Empleo de recubrimientos y lubricantes avanzados para minimizar la resistencia mecánica en los puntos de contacto.

Estas estrategias han permitido mejorar el rendimiento de los exoesqueletos en diversas aplicaciones, optimizando su integración con el movimiento humano y reduciendo la fatiga muscular.

Desarrollo

Materiales

- Arduino MEGA x1
- Puente H x 1
- Motorreductor JGB37-3530 - 333 RPM x 1
- Potenciometro 5 - 10K x 1
- INA219 Modulo x 1
- Software Arduino IDE
- Laptop o computadora de escritorio.

Implementación del circuito

En la Figura 1 podemos observar las conexiones presentadas en las sesiones de clase por el Mtro. Ruben R. Raygosa poder llevar a cabo la elaboración de la práctica.

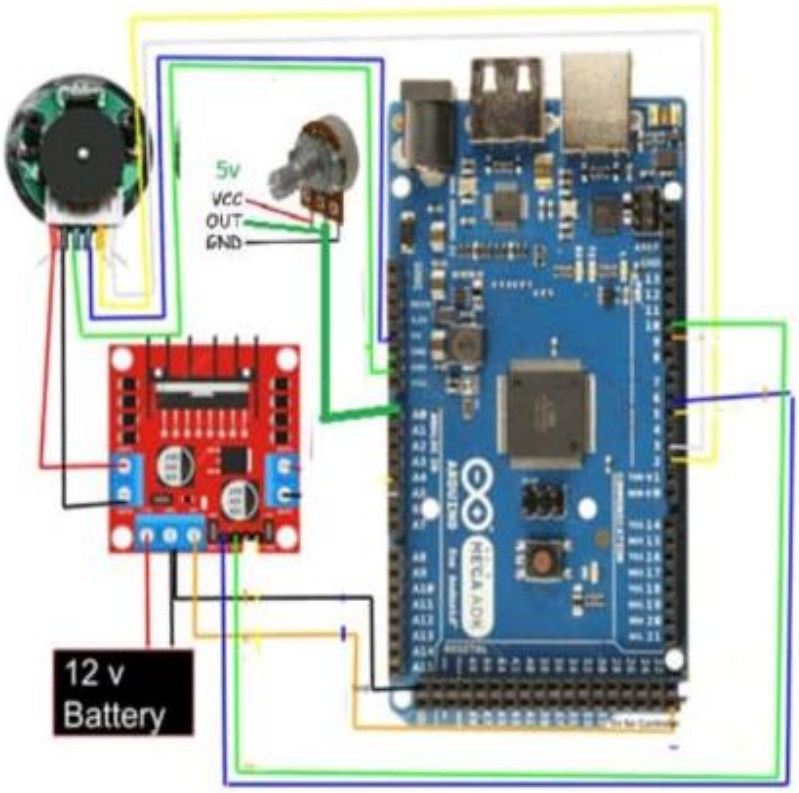

Figura 1. Conexiones

Metodología

Definición de pines:

- Encoder.h: Para leer la posición del encoder rotatorio.
- TimerOne.h: Para ejecutar una rutina (interrupción) periódica cada 10 ms.
- Adafruit_INA219.h: Para medir corriente del motor usando el sensor INA219.
- L298N: Puente H para controlar dirección y velocidad del motor.

1. ***Definiciones y variables***

Encoder encoder(ENCODER_A, ENCODER_B);
Adafruit_INA219 ina219;
Instancia los objetos del encoder y del sensor INA219

volatile long pos_actual, pos_anterior;
volatile float veloc, veloc_anterior, aceleracion;
float fricc, inercia, k, kd, C;
volatile float output;
Estas variables se usan para calcular la dinámica del motor (velocidad, aceleración, fuerza de fricción, etc.).

float corriente_filtrada = 0;
float alpha = 0.1;
Se usa un filtro exponencial para suavizar la lectura de corriente.

2. ***setup()***

- Inicia la comunicación serial con PC y con otro Arduino.
- Configura Timer1 para que llame a calcular Velocidad() cada 10 ms.

3. ***loop()***

Se ejecuta constantemente:
• Envía corriente_filtrada y pos_actual por Serial1 (a otro Arduino).

• Imprime lo mismo por el monitor serial para depuración.

• Recibe datos del otro Arduino:

1. A_recibida: parámetro adicional (no se usa directamente en el cálculo).

2. pos_deseada: la nueva posición que el motor debe alcanzar.

• Llama a motorControl(output) para mover el motor según el valor calculado.

• Espera 10 ms para no saturar el puerto serial.

4. Calcular la velocidad

Se ejecuta cada 10 ms por interrupción del temporizador:

• Calcula la velocidad y aceleración a partir de la posición del encoder.

• Calcula la señal de salida output, que es una mezcla de:

*1. Término por inercia: inercia * aceleración*

*2. Término por fricción: fricc * veloc*

*3. Término proporcional al error de posición: k * (pos_actual - pos_deseada)*

*4. Término derivativo: kd * veloc*

*5. Término por corriente del motor: C * corriente_filtrada*

• Limita el output a valores entre -255 y 255.

5. motorControl(speed)

Controla la dirección y velocidad del motor con el L298N:

• Si speed > 10, gira en una dirección.

• Si speed < -10, gira en la otra.

• Si está en el rango entre -10 y 10, el motor se detiene.

Resultados

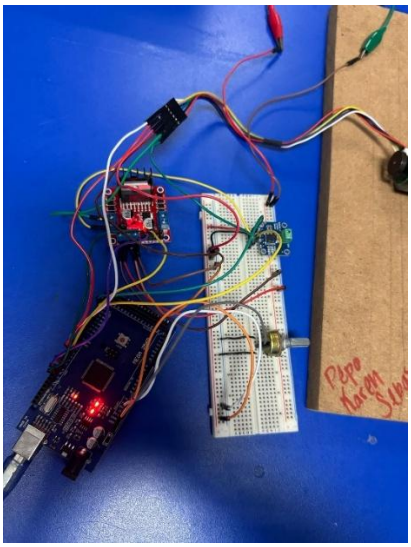

Figura 2. Armado del circuito.

En la figura 2 podemos ver las conexiones realizadas para poder controlar el motor, en conjunto con el INA219 y su puente H correspondiente.

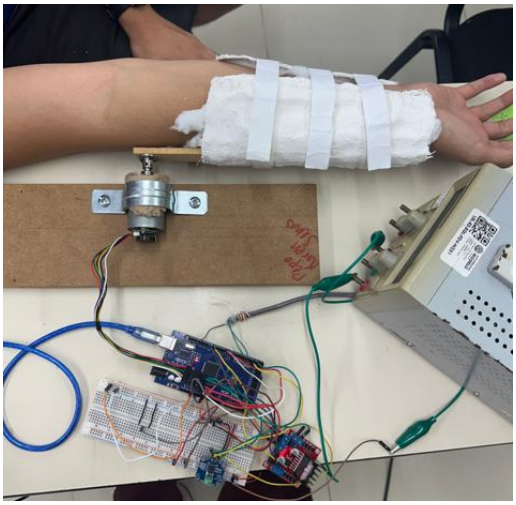

Figura 3. Armado con exoesqueleto

Como resultado final, el siguiente video muestra el funcionamiento adecuado del código, evidenciando la correcta cancelación de fricción e inercia y cumpliendo así con el objetivo establecido para la práctica.

Link: Video Evidencia de funcionamiento del Exoesqueleto de un grado de libertad.

https://drive.google.com/file/d/13BzjPuFdY6t7rwHbACawkiqR1EGgvID/view?usp=sharing

https://drive.google.com/file/d/11OE2YdMZm54VGxVskkTA-S8JzQqBWgce/view?usp=sharing

Las siguientes imágenes demuestran los resultados obtenidos a lo largo de la práctica, desde las primeras pruebas con la comunicación bilateral y las pruebas finales, entregadas al profesor como examen.

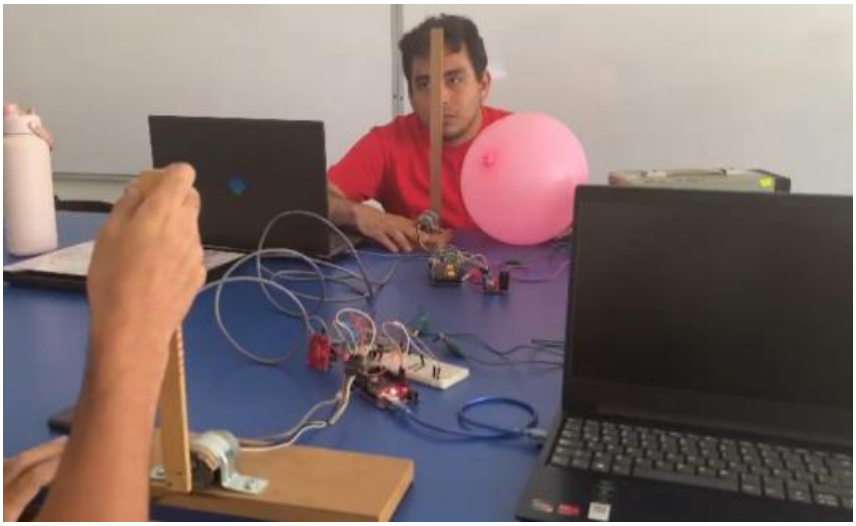

Figura 4. Comunicación bilateral maestro – esclavo / Posición 1

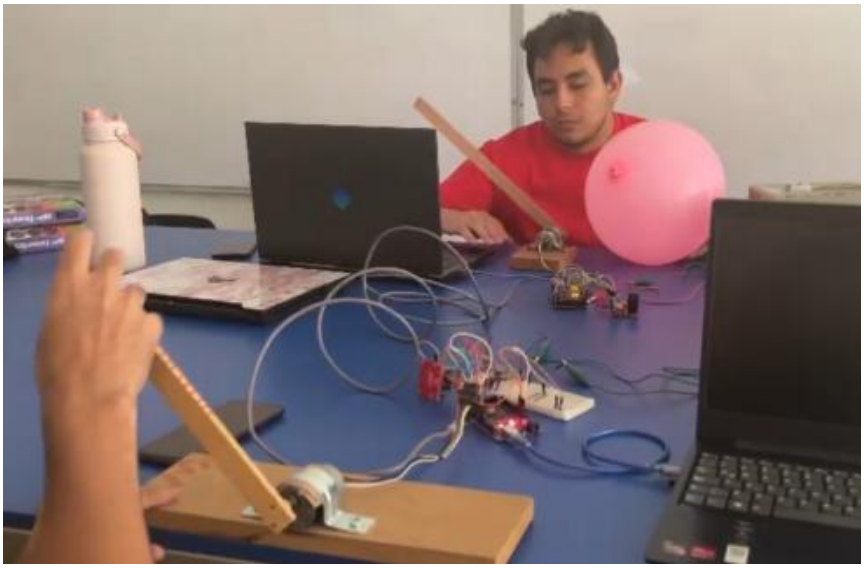

Figura 5. Comunicación bilateral maestro – esclavo / Posición 2

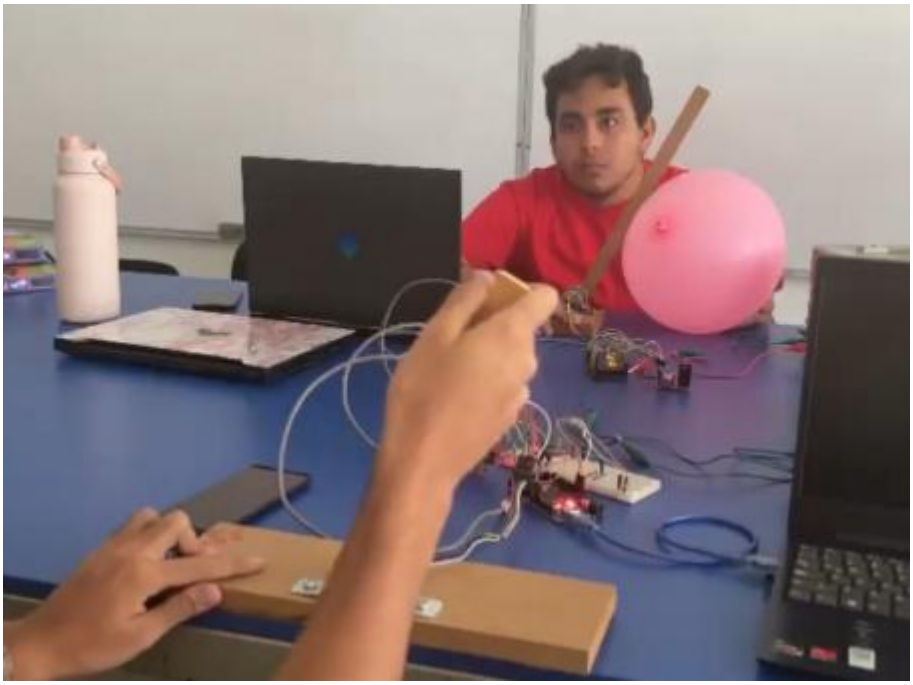

Figura 6. Comunicación bilateral maestro – esclavo / Posición 3

Resultados de entrega final de practica:

Figura 7. Amperaje con fuerza opuesta

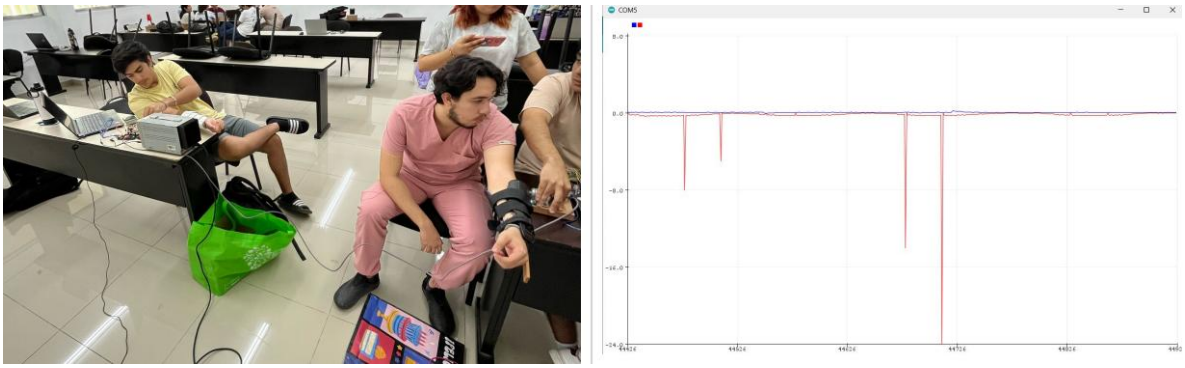

Figura 8. Amperaje sin fuerza opuesta

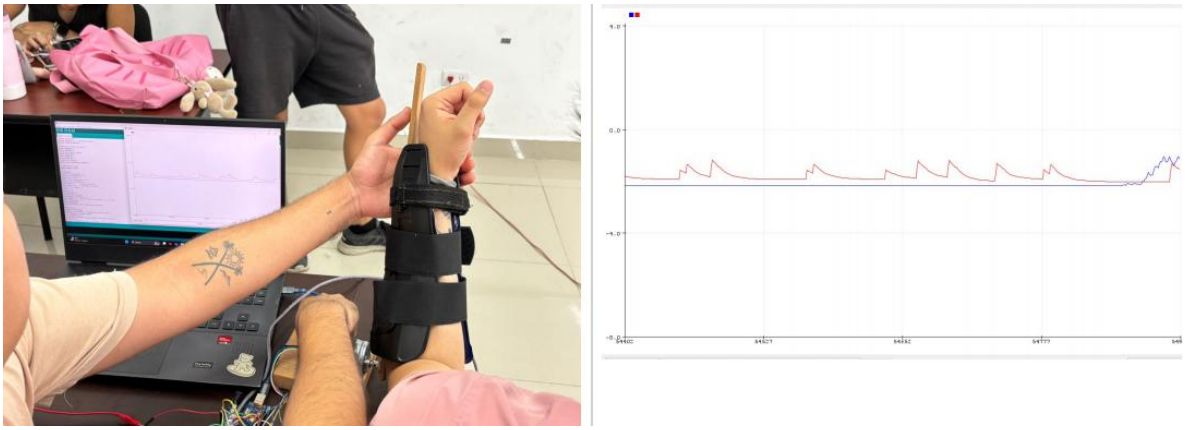

Figura 9. Amperaje con fuerza opuesta (Hacia el globo)

Conclusión

Como conclusión, se puede destacar el éxito en las pruebas realizadas y al desarrollar un control que minimizara los efectos de la fricción y la inercia en un exoesqueleto de un grado de libertad. El proceso se llevó a cabo en dos fases, comenzando con la compensación de la fricción y continuando con la cancelación de la inercia.

En la primera fase, se implementó un control orientado a reducir la resistencia generada por la fricción, lo cual permitió un movimiento más fluido del actuador. Los resultados experimentales mostraron una mejora significativa en la respuesta del sistema en comparación con su comportamiento sin compensación.

Siendo asi una comprobación de funcionamiento bilateral efectiva, ya que, al realizar los movimientos de el brazo del motor maestro, el brazo esclavo instantáneamente seguía su movimiento. Esto nos da a entender las bases solidas que estamos construyendo para poder entender muchos de los procedimientos, procesamientos y la rehabilitación que se puede realizar con este tipo de equipos o comunicaciones, ya que sirve perfectamente para la ayuda en rehabilitación de los movimientos de un brazo o simplemente ayudar a la movilidad de este.

Además, las pruebas no solo fueron realizadas por miembros del equipo, sino también de compañeros de la carrera, los cuales por experiencia propia pudieron experimentar lo que podría ser una rehabilitación asistida electrónicamente, ya que, en todo momento, el maestro puede tener el control del brazo y también de la resistencia que puede llegar a sentir el brazo esclavo.

Bibliografía

Castaño Giraldo, S. A. (2021). *Motor DC con Encoder – Velocidad – Posición*. Control Automático Educación. Recuperado de https://controlautomaticoeducacion.com/sistemas-embebidos/arduino/motor-dc-encoder/

Amazon. (2025). *JGB37-3530 Encoder Reduction Motor, DC 12V Electric Motor Smart Car Gear Motor*. Recuperado de https://www.amazon.com/JGB37-3530-Encoder-Reduction-Motor-Electricn/dp/B0B5GWDRNJ

Arduino. (2025). *Arduino Mega 2560 Rev3*. Recuperado de https://store.arduino.cc/products/arduino-mega-2560-rev3

Mecánica Industrial. (2025). *Revoluciones por minuto (RPM): Definición y cálculo*. Recuperado de https://www.mecanicaindustrial.com/revoluciones-por-minuto-definicion-calculo/

Del Valle Hernández, L. (2022, 13 enero). INA219 mide el consumo en tus proyectos Arduino. *Programarfacil Arduino y Home Assistant*. https://programarfacil.com/blog/arduino-blog/sensor-ina219-arduino-esp8266-esp32/

Ivaldi, S. (2019). *Los exoesqueletos*. The Fabricator. Recuperado de https://www.thefabricator.com/thefabricatorenespanol/article/safety/los-exoesqueletos

Sánchez Rodríguez, S. (2021). *Simulación y control de un exoesqueleto robótico* (Trabajo de fin de máster, Universidad de Alicante). Recuperado de https://rua.ua.es/dspace/bitstream/10045/115952/1/Simulacion_y_control_de_un_exoesqueleto_robotico_Sanchez_Rodriguez_Sheila.pdf

Pérez, J. A. (2013). *Control de seguimiento de un exoesqueleto basado en impedancia con compensación de fricción* (Tesis de maestría, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional). Recuperado de https://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesJAP.pdf

Anexos

Código de Arduinos

```
#include <Encoder.h>
#include <TimerOne.h>  // Biblioteca TimerOne
#include <Adafruit_INA219.h>

// Definición de pines del encoder
#define ENCODER_A 2
#define ENCODER_B 4
// Pines del puente H (L298N)
#define MOTOR_PWM 9
#define MOTOR_IN1 8
#define MOTOR_IN2 7
// Objeto del encoder
Encoder encoder(ENCODER_A, ENCODER_B);

// Instancia del INA219
Adafruit_INA219 ina219;

// Variables
volatile long pos_actual = 0;
volatile long pos_anterior = 0;
volatile float veloc = 0;
volatile float veloc_anterior = 0;
volatile float aceleracion = 0;
float fricc = 3;    // Factor de fricción
float inercia = 2;  // Coeficiente de inercia
volatile float output = 0;
float k = 10;
float kd = 20;
float pos_des = 0;
// Filtro para corriente
float corriente_filtrada = 0;
float alpha = 0.1;  // Valor entre 0 (muy suave) y 1 (sin filtro)
String inputString = "";
float pos_deseada;
float A_recibida;
float C = 150;

void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);  // Comunicación con otro Arduino
  // Inicializar el sensor INA219
  if (!ina219.begin()) {
    Serial.println("No se encontró INA219. Verifique conexiones.");
    while (1)
      ;  // Detener ejecución si el sensor no está disponible
  }
  pinMode(MOTOR_PWM, OUTPUT);
```

```
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  // Configurar TimerOne para que interrumpa cada 10ms (10000 µs)
  Timer1.initialize(10000);
  Timer1.attachInterrupt(calcularVelocidad);
}

void loop() {
  float corriente = ina219.getCurrent_mA() / 1000.0;
  corriente_filtrada = alpha * corriente + (1 - alpha) * corriente_filtrada;
  Serial1.print(corriente_filtrada);
  Serial1.print(",");
  Serial1.println(pos_actual);
  //imprimir en mi pantalla
  Serial.println("Enviando: ");
  Serial.print(pos_actual);
  Serial.print(", ");
  Serial.println(corriente_filtrada);
  // Recibir datos desde otro Arduino
  if (Serial1.available()) {
    inputString = Serial1.readStringUntil('\n');  // Leer hasta salto de
línea
    int pos1 = inputString.indexOf(',');
    int pos2 = inputString.indexOf(',', pos1 + 1);
    A_recibida = inputString.substring(0, pos1).toFloat();
    pos_deseada = inputString.substring(pos1 + 1, pos2).toFloat();
    Serial.println("Recibido");
    Serial.print(pos_deseada);
    Serial.print(",");
    Serial.println(A_recibida);
  }
  motorControl(output);
  delay(10);  // Evita saturar el puerto serie
}

// Función ejecutada por el Timer1 cada 10 ms
void calcularVelocidad() {
  pos_actual = encoder.read();  // Leer posición del encoder
  // Calcular la velocidad y la aceleración
  veloc = pos_actual - pos_anterior;
  aceleracion = veloc - veloc_anterior;
  // Cálculo para la friccion e inercia
  output = (inercia * aceleracion) + (fricc * veloc) + (k * (pos_actual-
pos_deseada)) + (kd * veloc)+(C * corriente_filtrada);
  output = constrain(output, -255, 255);  // Limitar el valor de salida
  // Actualizar las variables de posición y velocidad anterior
  pos_anterior = pos_actual;
  veloc_anterior = veloc;
}

// Función para controlar el motor
void motorControl(float speed) {
```

```
  speed = constrain(speed, -255, 255);
  if (speed > 10) {
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, HIGH);
    analogWrite(MOTOR_PWM, speed);
  } else if (speed < -10) {
    digitalWrite(MOTOR_IN1, HIGH);
    digitalWrite(MOTOR_IN2, LOW);
    analogWrite(MOTOR_PWM, abs(speed));
  } else {
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, LOW);
    analogWrite(MOTOR_PWM, 0);
  }
}
```