Sistema de registro, hardware y software de un dispositivo para entrenar el movimiento fino de la muñeca

Díaz Carmona Javier Axel Martínez Hernández Erick

Universidad Nacional Autónoma de México
Facultad de Ingeniería
9 de febrero de 2026

Resumen

Este documento describe el diseño y desarrollo de un dispositivo mecatrónico para la rehabilitación y entrenamiento del movimiento fino de la muñeca. Se detallan los componentes de hardware (Arduino Nano, MPU6050), el diseño estructural en impresión 3D, y la plataforma de software desarrollada en Python para el registro de variables cinemáticas (Pitch, Yaw, Roll) en tiempo real.

1. Objetivo General

Describir detenidamente los componentes utilizados, así como la plataforma de registro, incluyendo los tiempos de muestreo y las variables que se almacenarán en un dispositivo para entrenar el movimiento fino de la muñeca.

2. Justificación y Contexto

La presente propuesta de protocolo de investigación responde a la necesidad de comprender y mejorar la rigidez en el movimiento de la muñeca, una problemática que afecta significativamente la calidad de vida. La muñeca, una articulación compleja, es esencial para actividades diarias como la flexión, extensión, abducción y aducción [1].

Patologías como fracturas del radio distal, artritis reumatoide y el síndrome del túnel carpiano limitan el movimiento fino, fundamental para tareas de precisión. Para abordar esto, se propone un dispositivo mecatrónico interactivo que evalúa la motricidad fina a través de trazos gráficos, movilizando selectivamente los ejes de movimiento [2].

3. Hardware del Sistema

3.1. Selección del Microcontrolador

Los sistemas de rehabilitación modernos incorporan microcontroladores para lograr precisión. Se realizó una comparativa de las placas más populares en el mercado (Ver Cuadro 1).

Tabla 1: Comparativa de Placas de Desarrollo

Característica	Arduino Nano	Arduino Uno	ESP32 DEVKIT V1	R-Pi Pico
Microcontrolador	ATmega328P (8-bit)	ATmega328P (8-bit)	ESP32 (32-bit)	RP2040
Velocidad Reloj	16 MHz	16 MHz	240 MHz	133 MHz
Flash Memory	32 KB	32 KB	4 MB	16 MB
SRAM	2 KB	2 KB	520 KB	264 KB
Pines I/O	14 (6 PWM)	14 (6 PWM)	34	30
Dimensiones	45 x 18 mm	68.6 x 53.4 mm	Variable	Variable
Peso	7 g	25 g	Variable	Variable

Fuente: UNIT Electronics, 2025.

Se seleccionó el **Arduino Nano** por su tamaño reducido (45 x 18 mm), bajo consumo y facilidad de integración, a pesar de requerir una aproximación basada en comunicación serial por no ser compatible nativamente con librerías HID como ‘Mouse.h‘.

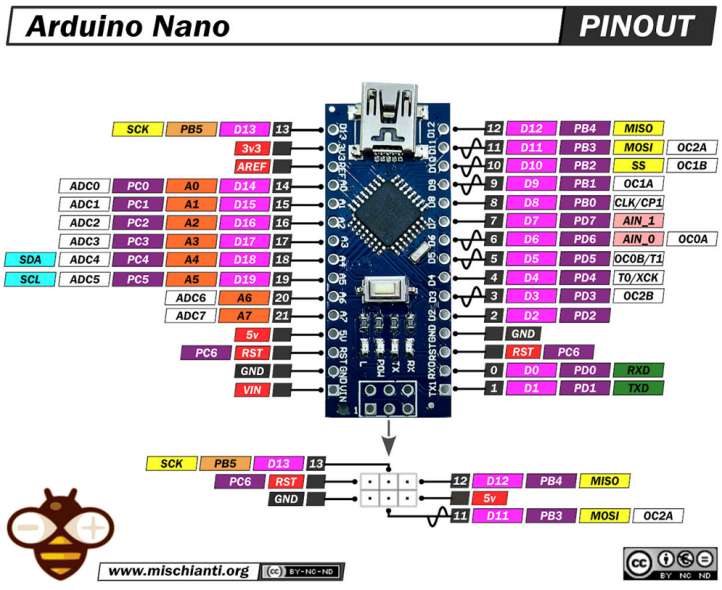


Figura 1: Pinout del Arduino Nano. Fuente: Mischianti.org

3.2. Selección del Sensor de Movimiento (IMU)

Para medir el rango de movimiento activo, se optó por sensores inerciales debido a su portabilidad. Tras comparar opciones como el MPU9250 y GY-80, se seleccionó el **MPU6050**

(6 grados de libertad) por ser ligero, confiable y ampliamente documentado [4].

- **Giroscopio:** 3 ejes, rango hasta $\pm 2000^\circ/s$.
- **Acelerómetro:** 3 ejes, rango hasta $\pm 16g$.
- **Comunicación:** Protocolo I2C.

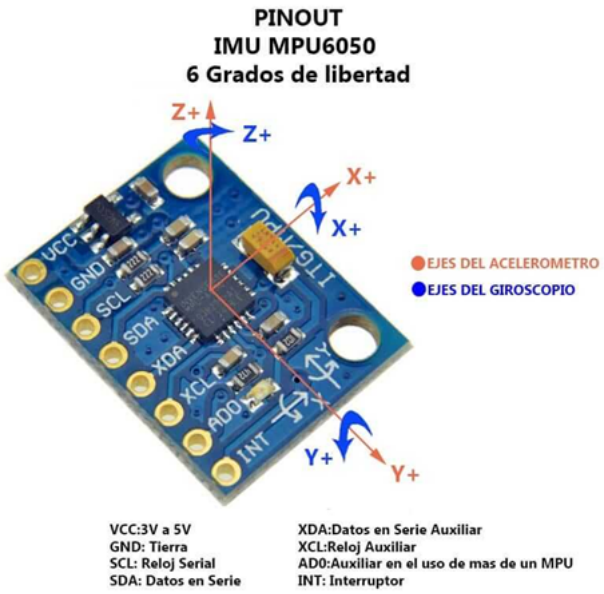
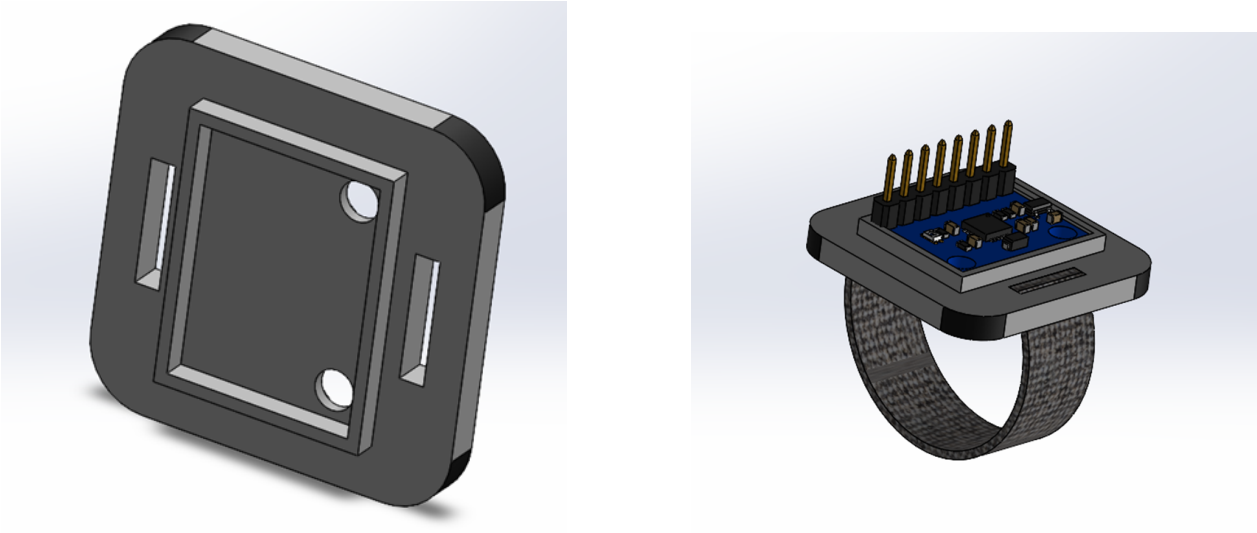


Figura 2: Pinout del sensor MPU6050.

4. Diseño y Estructura del Dispositivo

4.1. Diseño Mecánico y Materiales

El diseño prioriza la ergonomía y la no interferencia con la biomecánica natural. Se descartó una carcasa esférica en favor de una solución tipo "pulseraübicada en el dorso de la mano. El soporte del sensor (aprox. 25 x 30 mm) se fabrica mediante impresión 3D utilizando **PLA (ácido poliláctico)** debido a su rigidez, estabilidad dimensional y origen biodegradable [5].



(a) Soporte del sensor (b) Dispositivo ensamblado con correa

Figura 3: Diseño CAD del dispositivo.

4.2. Conexiones del Circuito

La comunicación entre el MPU6050 y el Arduino Nano se realiza vía I2C:

- **VCC** → 3.3V
- **GND** → GND
- **SCL** → A5
- **SDA** → A4

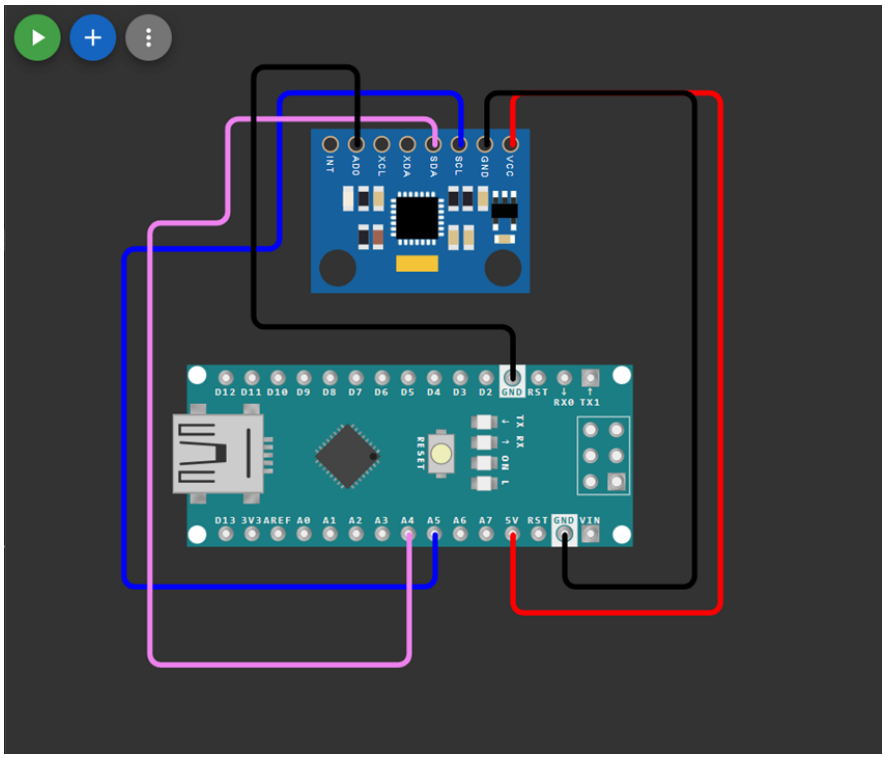


Figura 4: Diagrama de conexiones físicas.

5. Software y Plataforma de Registro

5.1. Firmware (Arduino C++)

El código lee los datos del sensor, calcula los ángulos de inclinación (Roll, Pitch) y rotación (Yaw) mediante integración, y envía los datos vía Serial a 115200 baudios con un muestreo de ~ 50 Hz.

```
#include <Wire.h>
const int MPU = 0x68;
int16_t accX, accY, accZ;
int16_t gyroX, gyroY, gyroZ;
float pitch, roll, yaw;
float gyroYaw = 0;
unsigned long prevTime;

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission();
  Serial.begin(115200);
  prevTime = millis();
}

```

```
void loop() {
  // Lectura del Acelerometro y Giroscopio (resumido)
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6);
  if (Wire.available() == 6) {
    accX = Wire.read() << 8 | Wire.read();
    accY = Wire.read() << 8 | Wire.read();
    accZ = Wire.read() << 8 | Wire.read();
  }

  // Calculos de Pitch, Roll y Yaw (Integracion)
  // ... (Ver codigo completo en anexo)

  // Transmision Serial
  Serial.print(roll, 2);  Serial.print(",");
  Serial.print(pitch, 2);  Serial.print(",");
  Serial.println(yaw, 2);
  delay(20);
}
```

Listing 1: Código de adquisición en Arduino

5.2. Aplicación de Registro (Python)

Se desarrolló una interfaz gráfica (GUI) en Python utilizando ‘tkinter‘, ‘pyserial‘ y ‘csv‘. La aplicación permite:

- Visualizar la trayectoria del puntero controlada por la muñeca.
- Seleccionar trazos guía (Líneas, Círculos, Triángulos).
- Registrar datos con timestamp en archivos CSV.
- Validar velocidad de ejecución (20 - 400 px/s).

```
import serial
import tkinter as tk
import csv
import math

# Configuraciones
PUERTO = 'COM5'
BAUDRATE = 115200
SENSIBILIDAD = 20
ARCHIVO_CSV = "ID_Sesion_TipoTrazo.csv"

def guardar_dato(x, y):
    timestamp = datetime.now().strftime("%d/%m/%Y %I:%M %p")
    with open(ARCHIVO_CSV, mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([timestamp, tipo_trazo, repeticiones, x, y])
```

```
17
18 # Logica principal de lectura serial y actualizacion de puntero
19 # ...
```

Listing 2: Fragmento de la App en Python

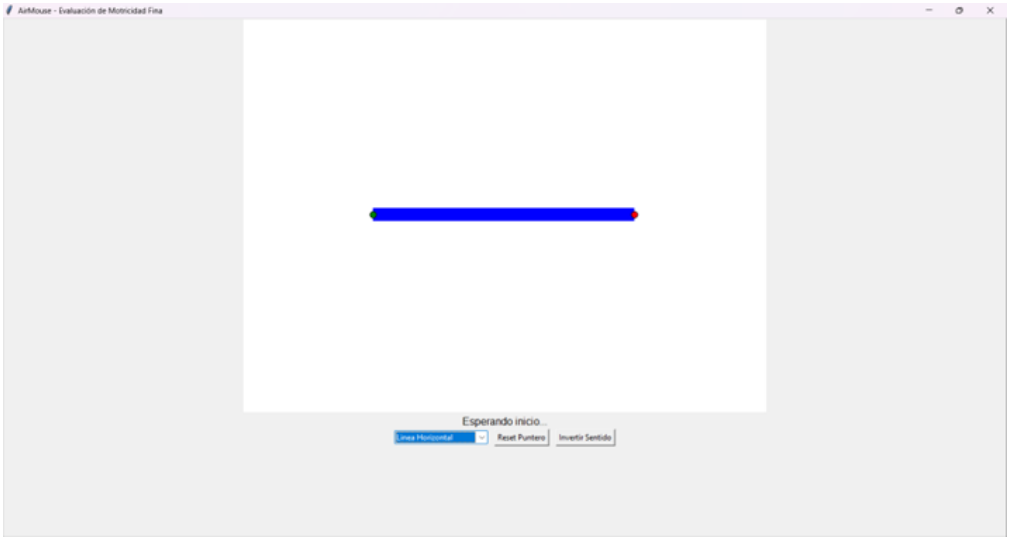


Figura 5: Interfaz gráfica de evaluación de motricidad fina.

6. Resultados y Variables Registradas

El sistema genera archivos CSV estructurados que permiten el análisis posterior de la destreza del usuario.

Tabla 2: Ejemplo de datos registrados

Timestamp	Figura	Repetición	X	Y
18/03/2025 09:03 p. m.	Triángulo	1	432	278
18/03/2025 09:03 p. m.	Triángulo	1	435	279

Referencias

[1] Carreño García, A. L., & Gutiérrez López, I. (2025). *Investigación de las patologías más comunes en la muñeca....*

[2] Suarez Jiménez, B. (2025). *Análisis de trazos y pruebas a realizar para el estudio del movimiento fino....*

[3] Costa, V., et al. (2020). Validity and reliability of inertial sensors for elbow and wrist range of motion assessment. *PeerJ*, 8, e9687.

[4] DISPOSITIVOS IMU: UNA COMPARACIÓN CON PERSPECTIVA MECATRÓNICA. (2023). *Pistas Educativas*, No. 146, TecNM Celaya.

[5] Torrado, et al. (2015). Estudio comparativo de propiedades mecánicas PLA vs ABS.

[6] UNIT Electronics. (2025). IMU MPU6050 6 Grados de libertad.

[7] Rupava. (s. f.). Air-Mouse-Using-Arduino-and-MPU6050. GitHub.