UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA
FACULTAD DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Curso: Inteligencia Artificial.

Catedrático: Ing. Jonathan Josué García Cuque.

Proyecto final

Sistema de Reconocimiento de Actividad Física con Seguimiento de Movimiento

HENRY ESTUARDO ALTÚN VARGAS 5390-17-308
STEVEN AMILCAR ALONZO MIRANDA 5390-18-1580
RONAL MAURICIO AGUIRRE MACHIC 5390-17-8306
GLENNALLEN KEITH BROWN LIMA 5390-18-19318

Introducción:

Este proyecto es realizado, para poner en practica los conocimientos obtenidos en el curso de inteligencia artificial, en donde utilizamos el machine Learning, Python, Arduino, en el manual veremos cómo se realizó el proyecto, así como también código fuente para la interconexión de los sensores con la PC.

Manual Técnico del Proyecto

Sistema de Reconocimiento de Actividad Física con Seguimiento de Movimiento

---

1. Metodología del Proyecto

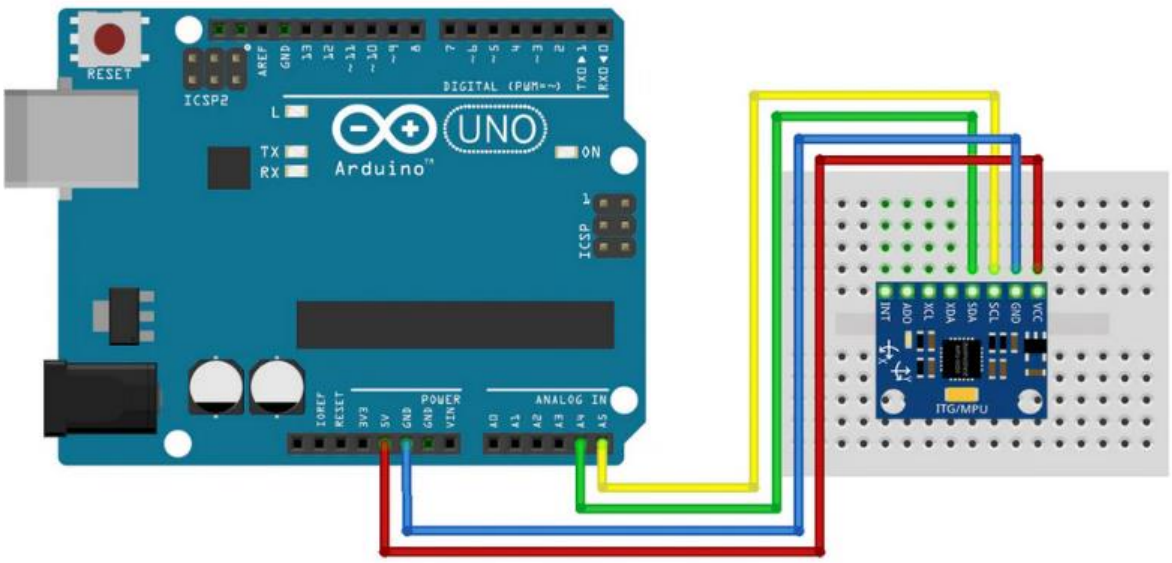
Objetivo:

Desarrollar un sistema que reconozca actividades físicas en tiempo real (caminar, correr, estar parado, caída) usando sensores inerciales (MPU6050), algoritmos de Machine Learning y herramientas de visualización como Power BI.

Fases del Desarrollo:

- **Fase 1:** Recolección de datos del sensor MPU6050 vía Arduino.
- **Fase 2:** Limpieza y preprocesamiento de datos en Python.
- **Fase 3:** Entrenamiento de un modelo de Machine Learning (Decision Tree o Random Forest).
- **Fase 4:** Clasificación en tiempo real desde datos seriales del sensor.
- **Fase 5:** Registro automático de los resultados en un archivo .csv.
- **Fase 6:** Visualización de los datos y eventos críticos en Power BI.

Fase 1. Arduino_sensor.ino



Código cargado en el Arduino que obtiene datos del sensor MPU6050:

Este código se encarga de inicializar el sensor MPU6050 y enviar continuamente por puerto serial los datos del acelerómetro y giroscopio.

Explicación:
#include <Wire.h> y #include <MPU6050.h>: Incluyen las librerías necesarias para la comunicación I2C y para manejar el sensor MPU6050.

MPU6050 mpu;: Crea una instancia del sensor.

En setup():

Se inicializa la comunicación serial a 9600 baudios.
Se inicializa la comunicación I2C.
Se inicia el sensor y se verifica la conexión. Si falla, se muestra un mensaje y se detiene el programa.

En loop():

Se leen los valores de aceleración (ax, ay, az) y giroscopio (gx, gy, gz) en los tres ejes.
Se imprimen por el puerto serial separados por comas.
Hay un retardo de 100 ms entre cada lectura.

Puerto_COM3

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();

  if (!mpu.testConnection()) {
    Serial.println("MPU6050 connection failed");
    while (1);
  }
}

void loop() {
  int16_t ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Imprime los datos en una sola línea separada por comas
  Serial.print(ax); Serial.print(",");
  Serial.print(ay); Serial.print(",");
  Serial.print(az); Serial.print(",");
  Serial.print(gx); Serial.print(",");
  Serial.print(gy); Serial.print(",");
  Serial.println(gz);

  delay(100); // Muestra datos cada 100 ms
}
```

Fase 2. script_Conexión con puerto serial de arduino.py

Este script se utiliza para leer los datos enviados por el Arduino y almacenarlos en un archivo .csv.

Explicación:

- Se abre el puerto serial COM3 a 9600 baudios.

- Se leen continuamente líneas enviadas por el Arduino.
- Cada línea contiene 6 valores (aceleración y giroscopio).
- Se convierten a enteros y se agregan a una lista.
- Cuando el usuario presiona Ctrl + C, se guarda la lista en un DataFrame de pandas.
- Finalmente, se exporta a un archivo datos_actividad.csv en la ruta del proyecto.

C: > Users > Estuardo > Desktop > 2025 > Inteligencia Artificial > Proyecto Final > paso 1 import serial.py > ...

```python
import serial
import pandas as pd
import os

puerto = serial.Serial('COM3', 9600)
datos = []

print("Leyendo datos... Presiona Ctrl+C para detener")

try:
    while True:
        linea = puerto.readline().decode().strip()
        partes = linea.split(",")
        if len(partes) == 6:
            datos.append([int(x) for x in partes])
except KeyboardInterrupt:
    df = pd.DataFrame(datos, columns=["ax", "ay", "az", "gx", "gy", "gz"])

    ruta = r"C:\Users\Estuardo\Desktop\2025\Inteligencia Artificial\Proyecto Final"
    archivo = os.path.join(ruta, "datos_actividad.csv")

    df.to_csv(archivo, index=False)
    print(f"Datos guardados en '{archivo}'")

```

Fase 3. script_entrenamiento.py

Este script entrena un modelo de clasificación de actividades a partir de los datos recolectados.

Explicación:

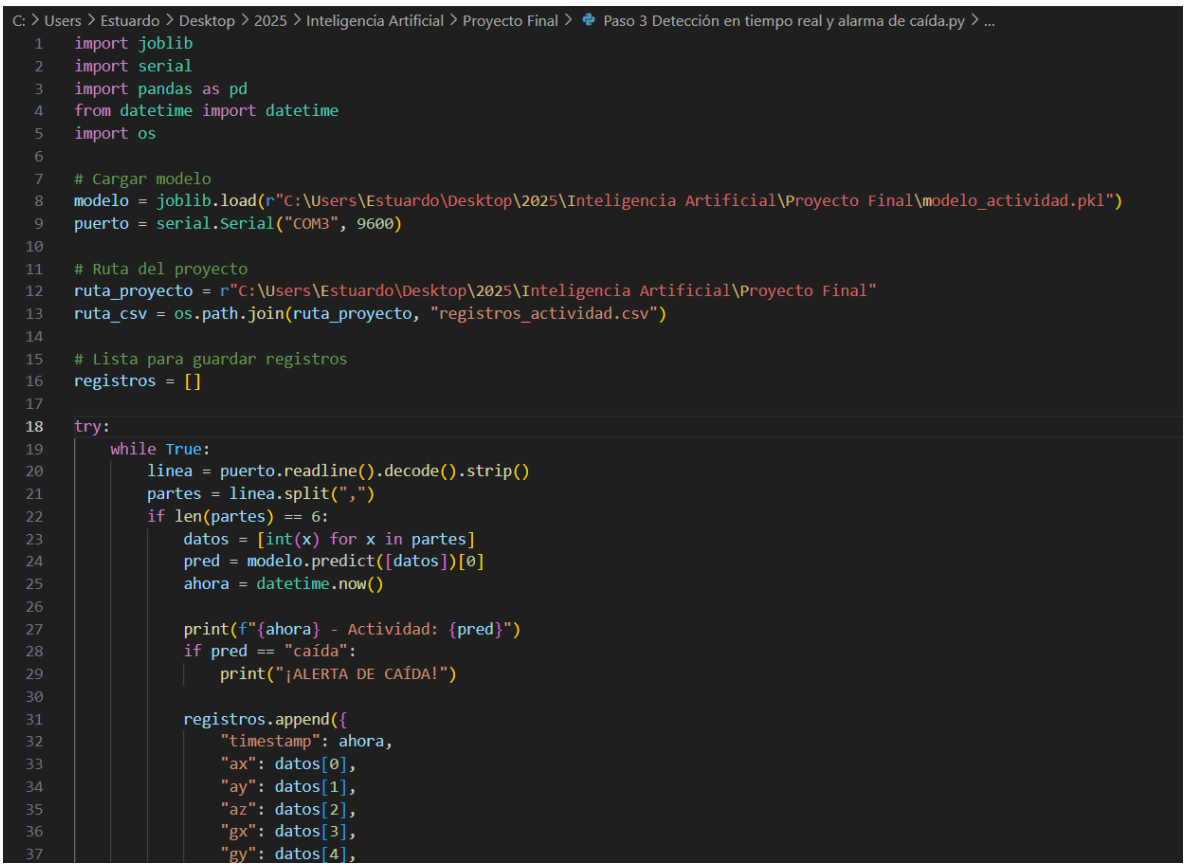
- Se lee el archivo datos_actividad_etiquetado.csv, que contiene los valores del sensor y una columna con la actividad etiquetada.
- Se dividen los datos en características (X: valores del sensor) y etiquetas (y: actividad).
- Se separan en conjuntos de entrenamiento y prueba (80/20).
- Se entrena un modelo RandomForestClassifier con 100 árboles.
- Se evalúa el modelo con métricas de clasificación y se imprime el reporte.
- Se guarda el modelo entrenado en un archivo .pkl llamado modelo_actividad.pkl.

Fase 4. script_lectura_modelo.py

Este script carga el modelo entrenado, lee datos en tiempo real desde el sensor y predice la actividad.

Explicación:

- Se carga el modelo desde el archivo .pkl.
- Se abre el puerto serial y se comienza a leer los datos enviados por Arduino.
- Cada línea es convertida en una lista de enteros y pasada al modelo para predecir la actividad.
- Se muestra la actividad predicha en consola junto con la fecha y hora.
- Si se detecta una caída, se imprime una alerta especial.
- Cada 10 registros, se guarda la información en un archivo registros_actividad.csv en la carpeta del proyecto.
- Cuando se interrumpe el programa con Ctrl + C, se guardan los registros pendientes.



```
                    "gz": datos[5],
                    "actividad": pred
                })

                # Guardar CSV cada 10 registros
                if len(registros) >= 10:
                    df = pd.DataFrame(registros)
                    df.to_csv(ruta_csv, index=False, mode='a', header=not os.path.exists(ruta_csv))
                    registros = []

    except KeyboardInterrupt:
        puerto.close()
        if registros:
            df = pd.DataFrame(registros)
            df.to_csv(ruta_csv, index=False, mode='a', header=not os.path.exists(ruta_csv))
        print(f"Programa finalizado. Datos guardados en: {ruta_csv}")

```

2. Configuración del Sistema

Hardware:

- Arduino UNO o compatible.

Placa microcontroladora basada en el microchip ATmega328P, utilizada para la recolección de datos desde sensores y su transmisión al computador a través del puerto serial (USB).

- Sensor MPU6050 (acelerómetro + giroscopio).

Módulo que integra un acelerómetro de 3 ejes y un giroscopio de 3 ejes, lo que permite obtener datos precisos sobre aceleración y rotación. Es esencial para detectar patrones de movimiento como caminar, correr, estar parado o caer.

- Cables de conexión Dupont.

Se requieren para conectar el sensor MPU6050 al Arduino. Se debe conectar:

- VCC a 3.3V o 5V
- GND a GND
- SDA a A4
- SCL a A5

- USB

Para conectar el Arduino a la computadora, permitiendo la alimentación y la transmisión de datos vía puerto serial.

Software:

- Arduino IDE

Entorno de desarrollo para cargar el sketch (programa) al Arduino. Se utiliza para configurar el sensor MPU6050 y enviar datos en tiempo real al puerto serial.

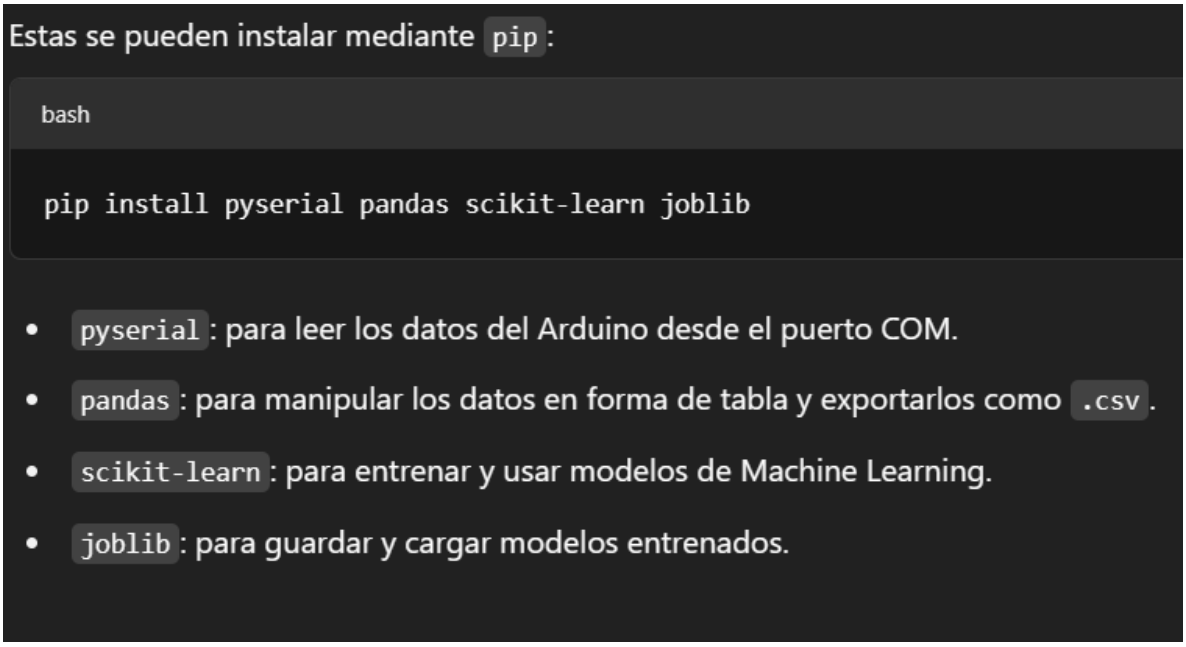
Librerías necesarias en el IDE:

- Wire.h – comunicación I2C.
- MPU6050.h – para inicializar y leer el sensor.

- Python 3.12

Lenguaje de programación utilizado para capturar, procesar, clasificar y guardar los datos provenientes del sensor.

- Librerías de Python necesarias

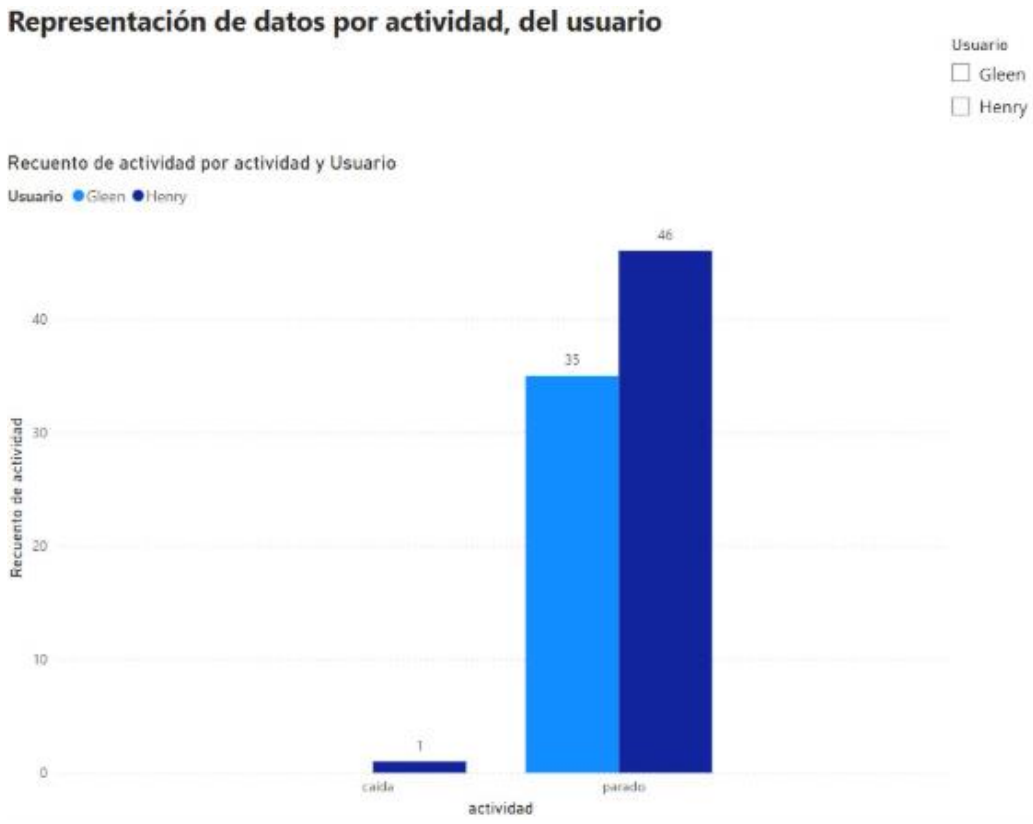


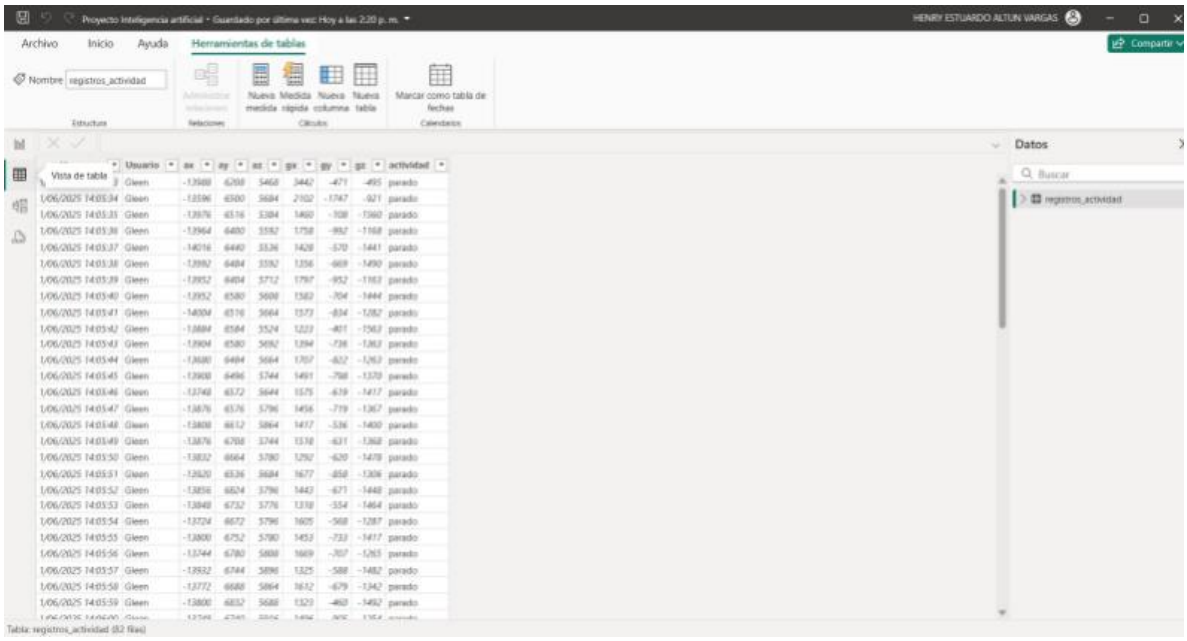
- Power BI Desktop

Power BI es una herramienta de visualización de datos que permite conectar archivos .csv exportados desde Python y crear dashboards con gráficas interactivas.

Funciones clave en este proyecto:

- **Importación de archivos .csv**: Conecta los registros de actividad almacenados desde Python.
- **Transformación de datos**: Filtrado, segmentación por usuario, fecha, hora, etc.
- **Visualización de actividades físicas**:
    - Gráficas por hora o por día.
    - Conteo de eventos críticos como caídas.
    - Comparativas entre múltiples usuarios si se habilita esa funcionalidad.




Proyecto Inteligencia artificial - Guardado por última vez: Hoy a las 2:20 p. m.
HENRY ESTUARDO ALTUN VARGAS
Archivo
Inicio
Ayuda
Herramientas de tablas
Compartir
Nombre registros_actividad
Estructura
Relaciones
Nueva medida
Medida rápida
Nueva columna
Nueva tabla
Cálculos
Marcar como tabla de fechas
Calendarios
Vista de tabla
Datos
Buscar
registros_actividad
Usuario
ax
ay
az
gx
gy
gz
actividad
Tabla: registros_actividad (82 filas)