Universidad Marianao Gálvez de Guatemala

Facultad de Ingeniería en Sistemas de Información y ciencias de la Computación

“Inteligencia Artificial”

Proyecto Final

Manual técnico

Lester Orlando Martínez Hernández 5390-21-1245

Eduardo Josué Galindo Monzón 5390-19-11356

Mervin Erín Blanco Olivares 5390-21-1541

Marco Fernando Basilio García 5090-21-1248

1. Metodología

¿Qué buscamos hacer?

La idea principal de este proyecto es crear un sistema que pueda identificar lo que una persona está haciendo, por ejemplo: caminar, estar quieta o si se cayó. Para lograrlo, usamos un sensor de movimiento (MPU6050) conectado a un Arduino, el cual envía datos a la computadora. Ahí, un programa hecho en Python analiza esos datos y, si detecta una caída, manda una señal para activar una alarma (buzzer).

¿Cómo funciona todo junto?

1. El Arduino recoge datos del sensor y los manda por cable USB (puerto serial).
2. Python recibe esos datos, los limpia y los pasa por un modelo de machine learning ya entrenado.
3. Se identifica la actividad que se está haciendo.
4. Esa información se guarda en un archivo Excel para tener un registro con fecha y hora.
5. Si se detecta una caída, se prende un buzzer (alarma). Si después la persona ya no está caída, se apaga.

Herramientas y librerías que usamos

- **Arduino + MPU6050**: para captar los movimientos.
- **Python** con las librerías:
    - pandas, numpy: para manejar los datos.
    - serial: para leer los datos que manda el Arduino.
    - sklearn: para entrenar y usar el modelo que detecta la actividad.
    - joblib: para guardar/cargar el modelo.
    - datetime: para registrar la hora de cada actividad.
    - openpyxl: para escribir en Excel.

2. Explicación del Código

Paso 1: Configuración

python

CopiarEditar

puerto_serial = 'COM3'

baudrate = 115200

tiempo_lectura = 7

archivo_modelo = 'modelo_actividad.pkl'

archivo_scaler = 'scaler.pkl'

archivo_excel = "usuarios_actividades.xlsx"

Se define el puerto, la velocidad de conexión, cuánto tiempo se van a leer los datos, y los nombres de los archivos donde se guarda el modelo y los registros.

Paso 2: Preguntar el nombre del usuario

python

CopiarEditar

usuario = input(" Ingresa tu nombre de usuario: ").strip()

Esto sirve para que a cada persona se le cree su propia hoja en el archivo Excel.

Paso 3: Verificar si ya hay un modelo entrenado

Si ya existen los archivos del modelo y del "scaler", se cargan. Si no, se entrena uno nuevo con un archivo CSV llamado dataset_manual.csv que contiene datos anteriores.

Paso 4: Conexión al Arduino

python

CopiarEditar

ser = serial.Serial(puerto_serial, baudrate, timeout=1)

Se establece la comunicación entre la compu y el Arduino. Si no se puede conectar, se muestra un error y se cierra el programa.

Paso 5: Captura y análisis de datos

Cada 7 segundos:

- Se reciben datos en tiempo real del sensor.
- Se convierten en una tabla con columnas: ax, ay, az, gx, gy, gz.

- Se normalizan esos datos y se hacen las predicciones.
- Se identifica cuál fue la actividad más común en ese período.

Paso 6: Guardar la actividad en Excel

Si el usuario ya tenía hoja, se le agregan los nuevos datos. Si es nuevo, se le crea una desde cero.

Paso 7: Activar el buzzer si hay caída

Si se detecta que la persona se cayó, se manda una señal al Arduino para encender el buzzer. Cuando se detecta que la persona ya no está caída, se apaga.

3. Configuración del Sistema

Hardware necesario

- Arduino UNO, Nano o Mega
- Sensor MPU6050 (acelerómetro y giroscopio)
- Buzzer
- Cables de conexión
- Puerto USB disponible en la PC

Software necesario

- Python 3 instalado
- Las siguientes librerías (se instalan así):

bash

CopiarEditar

pip install pandas numpy scikit-learn joblib openpyxl pyserial

Archivos que necesitás

- dataset_manual.csv: datos para entrenar el modelo.
- modelo_actividad.pkl: modelo entrenado guardado.
- scaler.pkl: escalador de datos guardado.
- usuarios_actividades.xlsx: archivo de registro.
- Script en Python y código cargado en el Arduino.

Código Arduino (resumen)

- Usa Wire.h y MPU6050.h para leer el sensor.
- En el loop(), manda los datos de los sensores por serial cada poco tiempo.
- Escucha los comandos:
    - "buzzer_on": prende el buzzer.
    - "buzzer_off": lo apaga.