¡vamos por partes! te dejo un “mapa de ruta” claro y corto + qué pinta tiene cada comando. También te explico exactamente para qué sirve --mode raw y --mode entregable y **cuándo** usarlos.

**Flujo completo (un solo main.cpp)**

[1] Captura de dataset (modo CAPTURA del main)

↓ python data\_acquisition.py COM6 --mode raw

→ data\_raw.csv

[2] Entrenamiento en PC

↓ python train\_tfl.py

→ model\_f32.tflite, model\_int8.tflite, feat\_mean.npy, feat\_std.npy

[3] Empaquetado para ESP32

↓ python export\_to\_c.py

→ include/model\_data.h, include/feat\_norm.h, src/model\_data.cc

[4] Flasheo del clasificador (mismo main.cpp, ahora detecta el modelo)

↓ pio run -t upload

→ “Modo: CLASIFICACIÓN (TFLM activo)” en Serial

[5] Protocolo de pruebas y archivo para entregar

↓ python data\_acquisition.py COM6 --mode entregable

→ data.csv (con Clase Real, Clase Predicha y Resultado de inferencia)

**¿Qué hago justo después de capturar?**

1. **Capturar tu dataset crudo** (manual por defecto: te pide Enter):

python data\_acquisition.py COM6 --mode raw

# (si quieres que arranque solo, añade --auto)

* Esto crea **data\_raw.csv** con columnas: ax,ay,az,gx,gy,gz,temp,label.
* Debes completar el protocolo controlado y los bloques aleatorios con tu movimiento correcto: el script etiqueta “Clase Real” por bloque.

1. **Entrenar el modelo y convertir a TFLite**:

python train\_tfl.py

* Salida:
  + model\_f32.tflite, model\_int8.tflite
  + feat\_mean.npy, feat\_std.npy (normalización idéntica a la que usará el main).

1. **Empaquetar el modelo para la ESP32**:

python export\_to\_c.py

* Copia los archivos generados a tu proyecto PlatformIO:
  + include/model\_data.h
  + include/feat\_norm.h
  + src/model\_data.cc

1. **Compilar y flashear el mismo main.cpp**:

pio run -t upload

* En el monitor serie verás: Modo: CLASIFICACIÓN (TFLM activo) si el modelo fue detectado.

1. **Correr el protocolo de pruebas “final” y generar el archivo de entrega**:

python data\_acquisition.py COM6 --mode entregable

# usa --auto si quieres cuenta regresiva sin Enter

* Salida: **data.csv** con columnas:
  + Feature1..Feature7 (las 7 señales crudas de la última muestra del hop),
  + Clase Real (según bloque),
  + Clase Predicha (p.ej. Mov02),
  + Resultado Inferencia (= probabilidad de la clase predicha).

Ese data.csv + tu main.cpp (el mismo, dual-mode) son lo que te piden para entregar.

**¿Qué hace cada modo del script?**

* --mode raw
  + **Para entrenar**. Guarda **data\_raw.csv** (8 columnas) a partir de lo que llega por serial.
  + Se usa cuando la ESP32 está **en modo CAPTURA** (no hay modelo cargado → el main imprime 7 campos).
  + Bloque de código donde se ve:
  + if args.mode == "raw":
  + # ... construye df con ax..temp,label
  + df.to\_csv("data\_raw.csv", index=False)
* --mode entregable
  + **Para la fase final**. Guarda **data.csv** con la estructura que pide la rúbrica (Clase Real, Clase Predicha y probabilidad).
  + Se usa cuando la ESP32 está **en modo CLASIFICACIÓN** (ya flasheaste el modelo → el main imprime 14 campos).
  + Si aún no tienes modelo y lo usas igual, “Clase Predicha” queda vacía y la probabilidad en 0 (no pasa nada, pero no sirve para la entrega final).
  + Bloque de código donde se ve:
  + else: # modo entregable
  + # ... arma df con Feature1..7, Clase Real, Clase Predicha, Resultado Inferencia
  + df.to\_csv("data.csv", index=False)

Si no pones --mode, el **default es raw** (captura para entrenamiento).

**Mini-checklist en cada paso**

**Antes de entrenar**

* Verifica que data\_raw.csv existe y tiene ~miles de filas (mejor si repites la captura 2–3 veces).

**Después de entrenar (train\_tfl.py)**

* Revisa en consola la **accuracy** y la matriz de confusión.
* Confirma que se generaron model\_int8.tflite, feat\_mean.npy, feat\_std.npy.

**Después de empaquetar (export\_to\_c.py)**

* Asegúrate de tener: include/model\_data.h, include/feat\_norm.h, src/model\_data.cc.

**Al iniciar la ESP32**

* En el monitor serie debe decir **“Modo: CLASIFICACIÓN (TFLM activo)”**.
  + Si dice “Modo: CAPTURA”, el proyecto no está encontrando los headers o el .cc.

**Para el archivo de entrega**

* Ejecuta data\_acquisition.py --mode entregable siguiendo el protocolo.
* Asegúrate de ver **Clase Predicha** y **Resultado Inferencia** con valores (no vacíos ni cero plano).