# TỔNG KẾT DÒNG CHẢY DỮ LIỆU HỆ THỐNG PIPELINE V5 (VỚI LOOP)

## 1. BẮT ĐẦU TỪ DỮ LIỆU THÔ (DOCX)

- Pipeline khởi đầu bằng file `.docx` chứa định nghĩa tuyến (route) cho từng robot.  
- Gọi module luồng core để xử lý:  
 + `extract\_routes\_by\_luong\_from\_docx(docx\_path)` → lấy route thô.  
 + `get\_standardized\_routes\_for\_luong(luong, raw\_routes)` → chuẩn hóa thành route\_steps\_by\_robot.

## 2. TRUYỀN DỮ LIỆU VÀO LOOP

- Biến `route\_steps\_by\_robot` được truyền vào hàm `run\_pipeline\_v5(route\_steps\_by\_robot)`.  
- Đây là trung tâm điều phối toàn bộ quá trình xử lý marker và tạo timeline.

## 3. STEP 1 – QUÉT MARKER

- Gọi hàm `scan\_markers(route\_steps\_by\_robot)` để xác định robot nào còn marker `"timer time lưu"`.  
- Kết quả: `timer\_marker\_dict = { robot: {vị trí: marker}, ... }`

## 4. PHÂN NHÁNH ROBOT SẠCH / CÓ MARKER

- Loop phân nhánh ngay sau Step 1:  
 + Nếu robot KHÔNG có marker: gọi `generate\_timeline\_rb\_v4\_3()` để xử lý hoàn chỉnh.  
 + Nếu robot CÓ marker: giữ lại để xử lý tiếp Step 2 → Step 6.  
- Kết quả timeline robot sạch được lưu vào `timeline\_by\_robot`.

## 5. XỬ LÝ TRƯỜNG HỢP TOÀN BỘ HỆ SẠCH

- Nếu:  
 + Không còn marker trong toàn hệ.  
 + Và toàn bộ robot đã có trong `timeline\_by\_robot`.  
→ Hệ thống tự động dừng vòng lặp (`break`) và trả về kết quả hoàn chỉnh.

## 6. TIẾP TỤC XỬ LÝ MARKER (NẾU CÓ)

- Nếu còn robot có marker:  
 + Loop sẽ tiếp tục thực hiện Step 2 → Step 6 cho từng marker.  
 + Mỗi vòng xử lý một marker cho một robot cụ thể.  
 + Khi không còn marker → dừng vòng lặp.

## 7. TỔNG KẾT KẾT QUẢ

- Sau khi vòng lặp kết thúc:  
 + `timeline\_by\_robot` sẽ chứa timeline hoàn chỉnh cho tất cả robot.  
 + Có thể truyền trực tiếp sang bước xuất file hoặc xử lý tiếp.