

Hệ thống phát hiện và cảnh báo té ngã thời gian thực tích hợp cảm biến, xử lý ảnh và định vị

Trần Đức Hảo

Ngày 17 tháng 9 năm 2025

Hệ thống phát hiện và cảnh báo té ngã thời gian thực tích hợp cảm biến, xử lý ảnh và định vị

Trần Đức Hảo

Ngày 17 tháng 9 năm 2025

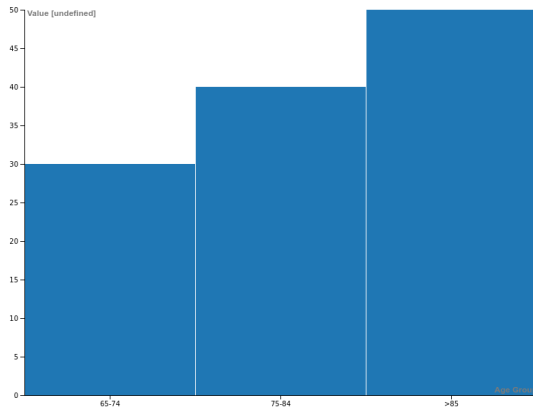
Tác giả: Trần Đức Hảo
Người hướng dẫn: PSG.TS Hà Hoàng Kha

Khoa: Khoa Điện – Điện tử
Trường: Trường Đại học BK.HCM

Ngày 17 tháng 9 năm 2025

Té ngã: Mối đe dọa toàn cầu

- Nguyên nhân chính gây chấn thương và tử vong không cố ý.
- **WHO**: ~ 646,000 ca tử vong/năm;
> 80% ở các nước thu nhập trung bình/thấp.
- Người cao tuổi: **30%** té ngã/năm ở người > 65 tuổi, tăng lên **50%** ở người > 85 tuổi.



Hình: Tỷ lệ té ngã theo nhóm tuổi

Tổng quan các phương pháp phát hiện té ngã

- **Dựa trên thị giác (Vision-based):** Sử dụng camera và thuật toán nhận diện tư thế người (MediaPipe, OpenPose).
- **Dựa trên cảm biến đeo (Wearable-based):** Dùng cảm biến quán tính (IMU, MPU6050) trên thiết bị.
- **Kết hợp đa phương thức (Multi-modal):** Tích hợp dữ liệu từ nhiều nguồn để tăng độ chính xác.

Quốc tế

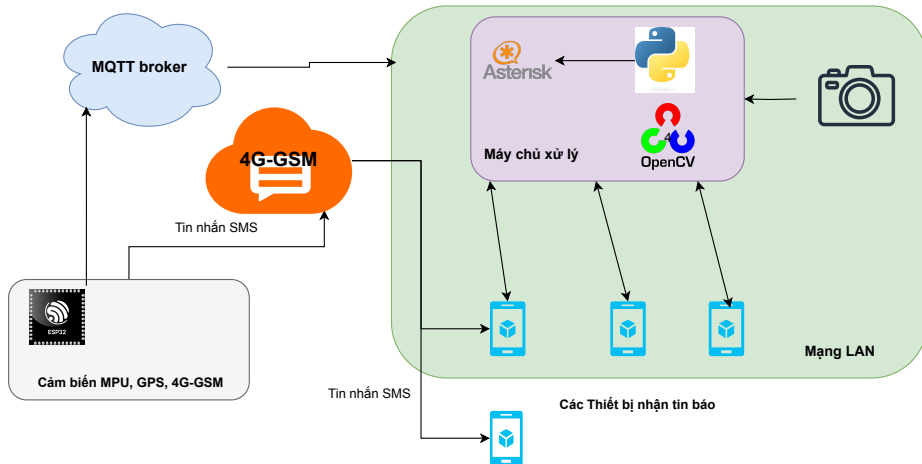
- **Xu hướng:** Sử dụng YOLO, Transformer, AI nhẹ, cảm biến mmWave.
- **Thành tựu:** Giảm false alarm, tối ưu cho thiết bị biên, Sensor Fusion.

Trong nước

- **Thực trạng:** Chủ yếu mô hình thử nghiệm (PoC) với ESP32, Arduino.
- **Hạn chế:** Thiếu dữ liệu lớn, độ chính xác thấp (75-85%), thiếu tích hợp đa phương thức.

- Xây dựng hệ thống **giám sát và cảnh báo té ngã** thông minh, **chi phí thấp**.
- Phát hiện **thời gian thực** (real-time) bằng cách kết hợp dữ liệu cảm biến và hình ảnh.
- Thiết kế kiến trúc phân lớp, **ổn định** và **dễ mở rộng**.

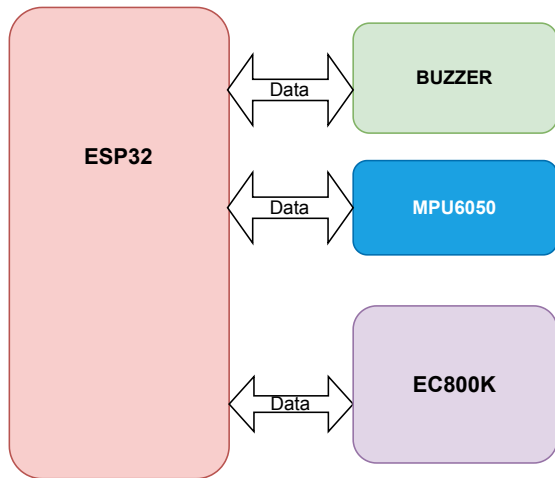
Kiến trúc hệ thống tổng thể



Hình: Sơ đồ hệ thống tổng thể

Hệ thống nhúng (ESP32)

- **Phần cứng:** ESP32, MPU6050, GPS EC800K.
- **Nguyên lý:** Phát hiện té ngã dựa trên ngưỡng động học.
- **Giao tiếp:** Gửi cảnh báo qua **MQTT và SMS**.
- **Ưu điểm:** Thiết bị độc lập, tiết kiệm năng lượng, dễ mở rộng.



Hình: Sơ đồ nhúng ESP32 & truyền thông

Mục tiêu Hiệu năng

- Tổng độ trễ < 5 giây.
- Độ chính xác $> 90\%$, False Alarm $< 8\%$.
- Uptime dịch vụ MQTT $> 99\%$.

Giới hạn nghiên cứu

- Hoạt động trong nhà với điều kiện ánh sáng và mạng ổn định.
- Nguyên mẫu **ESP32** chưa tích hợp học sâu toàn phần.
- Không phát triển app di động/web phức tạp.

Tổng quan các phương pháp phát hiện té ngã

Phương pháp	Cơ chế	Ưu điểm	Nhược điểm
Đeo được	IMU (gia tốc kế, con quay hồi chuyển); phát hiện gia tốc/tư thế bất thường	Phản hồi nhanh; chính xác; chi phí thấp	Cần đeo liên tục; dễ false positive; pin/hiệu chuẩn
Môi trường	Cảm biến cố định: sàn áp suất, PIR, âm thanh; AI phân tích	Không xâm phạm; giám sát nhiều người; tích hợp smart home	Chi phí cao; phạm vi hạn chế; nhầm vật thể
Thị giác	Camera RGB/RGB-D/IR; pose estimation (OpenPose/MediaPipe)	Thông tin trực quan; không cần đeo; tích hợp giám sát	Quyền riêng tư; phụ thuộc ánh sáng; cần phần cứng mạnh
Đa phương thức	Kết hợp IMU + camera + môi trường; data fusion (Kalman/Deep Learning)	Độ chính xác cao; giảm cảnh báo sai; mở rộng phạm vi; kinh tế	Phức tạp; tốn năng lượng; đồng bộ khó

- Kết hợp dữ liệu để xác nhận té ngã, giảm false positive.
- Chế độ linh hoạt: In-situ (cục bộ) + Mobile (edge device).
- Bảo mật: xử lý cục bộ, chỉ gửi dữ liệu tối thiểu, tùy chỉnh khu vực nhạy cảm.

Hệ thống phát hiện ngã với ba giao thức cốt lõi

- **SIP**: Truyền tải âm thanh/video cảnh báo thời gian thực
- **MQTT**: Vận chuyển dữ liệu cảm biến từ thiết bị IoT
- **JSON**: Định dạng cấu trúc dữ liệu trao đổi

Mục tiêu

Xây dựng hệ thống cảnh báo không gián đoạn, độ trễ thấp từ cảm biến đến cuộc gọi VoIP

Giao thức SIP - Khởi tạo Phiên

Chức năng chính:

- Thiết lập cuộc gọi VoIP từ hệ thống cảnh báo
- Kết nối với Asterisk PBX để gọi điện thoại
- Truyền âm thanh cảnh báo qua RTP

Các bước hoạt động:

- 1 REGISTER: Đăng ký thiết bị với server
- 2 INVITE: Khởi tạo cuộc gọi cảnh báo
- 3 ACK: Xác nhận kết nối thành công
- 4 RTP: Truyền dữ liệu âm thanh
- 5 BYE: Kết thúc cuộc gọi

Lưu ý

Sử dụng ICE để xuyên NAT, TLS/SRTP để bảo mật

Đặc điểm:

- Nhẹ, tiết kiệm băng thông cho thiết bị IoT
- Mô hình Publish/Subscribe qua Broker
- Hỗ trợ 3 mức QoS đảm bảo độ tin cậy

Mức QoS:

- **QoS 0**: Gửi một lần (dữ liệu thường)
- **QoS 1**: Ít nhất một lần (có xác nhận)
- **QoS 2**: Đúng một lần (cảnh báo quan trọng)

Ví dụ topic: `sensor/room/temperature, alert/fall/detected`

JSON - Định dạng dữ liệu:

- Nhẹ, dễ đọc, tương thích đa nền tảng
- Cấu hình thiết bị và trao đổi dữ liệu cảm biến
- Tối ưu payload cho MQTT

Luồng tích hợp hoàn chỉnh:

- 1 ESP32 phát hiện ngã → tạo JSON payload
- 2 Gửi qua MQTT topic với QoS phù hợp
- 3 Ứng dụng trung gian nhận và xử lý JSON
- 4 Kích hoạt cuộc gọi SIP qua Asterisk AMI
- 5 Phát cảnh báo âm thanh đến điện thoại

Kết Luận và Tối Ưu Hóa

Lợi ích của việc kết hợp 3 giao thức:

- **MQTT**: Thu thập dữ liệu hiệu quả từ cảm biến
- **JSON**: Cấu trúc dữ liệu linh hoạt, dễ xử lý
- **SIP**: Cảnh báo âm thanh tức thì, đáng tin cậy

Các biện pháp tối ưu:

- Payload JSON nhỏ gọn tiết kiệm năng lượng
- QoS MQTT phù hợp với mức độ quan trọng
- Tự động kết nối lại khi mất kết nối
- Bảo mật TLS cho MQTT và SIP

Kết quả

Hệ thống cảnh báo tự động, tin cậy từ thiết bị nhúng đến cuộc gọi VoIP

Định nghĩa & Mục tiêu

CV cho máy tính **hiểu và phân tích hình ảnh/video** → phân loại, phát hiện, theo dõi, ước lượng tư thế. Mục tiêu: **Nhanh, chính xác, quy mô lớn.**

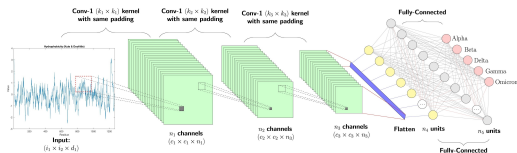


Hình: Pipeline CV: Thu nhận → Tiền xử lý → Trích xuất đặc trưng → Ra quyết định.

- Phân loại ảnh
- Phát hiện đối tượng (bounding box)
- Phân đoạn ảnh: ngữ nghĩa / thể hiện
- Ước lượng Tư thế Người (HPE) → keypoints

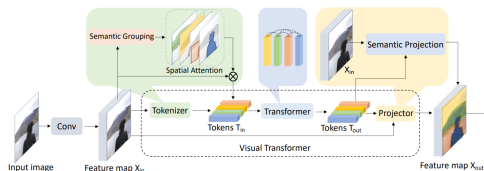
CNN & Vision Transformer

CNN: Tích chập + Pooling → học đặc trưng phân cấp



Hình: CNN Operations

ViT: Chia ảnh thành patches → Self-Attention → quan hệ toàn cục



Hình: Vision Transformer

Tập dữ liệu & Metrics trong CV

Tập dữ liệu chính

- **ImageNet** [?]: >14 triệu ảnh, 20,000 nhãn. Chuẩn cho **phân loại ảnh**.
- **COCO** [?]: Hơn 330k ảnh, có **bounding box, segmentation, keypoints**. Chuẩn cho phát hiện đối tượng phân đoạn.
- **MPII Human Pose**: 25k ảnh, 16 keypoints trên cơ thể người. Chuẩn cho HPE.
- **COCO Keypoints**: Mở rộng COCO cho 17 keypoints, hỗ trợ HPE thời gian thực.

Metrics

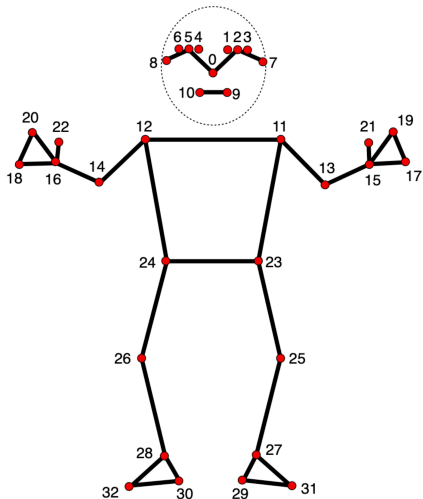
- IoU – Trùng khớp hộp giới hạn
- mAP – Hiệu suất phát hiện đối tượng
- F1-score – Trung bình điều hòa Precision Recall
- OKS – Độ chính xác keypoints HPE

Tổng quan

Hệ thống tích hợp nhận diện tư thế (MediaPipe Pose) và phát hiện té ngã dựa trên đặc trưng động học/tư thế.

- Ứng dụng: Giám sát an toàn, phát hiện té ngã.
- Nền tảng: Thị giác máy tính thời gian thực.

Nhận diện Tư thế Người



Keypoints cơ bản trong HPE (Human

Khái niệm

Ước lượng vị trí khớp từ hình ảnh/video:

$$\mathcal{K} = \{k_i = (x_i, y_i, z_i, c_i)\}$$

(c_i là confidence score cho mỗi keypoint).

Phương pháp

- **Top-down:** Phát hiện người trước, sau đó keypoints (MediaPipe).
- **Bottom-up:** Keypoints trước, nhóm thành người sau (OpenPose).

Kiến trúc BlazePose

BlazePose tối ưu HPE 3D với:

- **Nodes:** Các module xử lý tín hiệu hình ảnh.
- **Edges:** Luồng dữ liệu đồng bộ giữa các module.

(Nodes = các bước tính toán; Edges = kết nối dữ liệu giữa các bước).

Thành phần chính

- **Detection:** ROI từ ảnh RGB, phát hiện người.
- **Landmark:** 33 keypoints 3D, Loss: $\mathcal{L} = \sum \lambda_i \mathcal{L}_i$.
- **Tracking:** Dự đoán vị trí ROI cho khung tiếp theo.

(Landmark 3D giúp đánh giá tư thế và động tác).

Hậu xử lý

- **One Euro Filter:** Làm mịn nhiễu trong dữ liệu keypoints.
- **Chuẩn hóa z :** Dựa trên hông, tăng độ chính xác 3D.

Thuật toán Phát hiện Té ngã

Đặc trưng Động học

- **Vận tốc COM:** $\vec{v}_{\text{COM}} = \frac{\Delta \vec{p}}{\Delta t}$
- **Gia tốc:** $a = \frac{\|\Delta \vec{v}\|}{\Delta t}$

Đặc trưng Tư thế

- **AR (Aspect Ratio):** Tăng khi người nằm ngang
 - θ_{body} : Góc vai-hông
 - Δh_{head} : Giảm chiều cao đầu
- (AR, θ , Δh giúp xác định tư thế bất thường).*

Ba Giai đoạn Phát hiện

- 1 **Sớm:** Tốc độ/gia tốc COM cao
- 2 **Xác nhận:** AR, θ_{body} chỉ nằm ngang
- 3 **Bất động:** Chuyển động $< M_{th}$

Kiến trúc Hệ thống Phát hiện Tệ ngã

Phân loại

- Camera-based
- Wearable-based

Thành phần chính

- 1 Thiết bị thu thập (IMU/Camera)
- 2 Máy chủ xử lý (Deep Learning)
- 3 Truyền thông (Wi-Fi, 4G)

- **ESP32:** lõi kép, FreeRTOS, xử lý song song.
- **IMU:**
 - Gia tốc kế, Con quay, Từ kế
 - Sensor Fusion (Kalman/Madgwick)
- **Ngưỡng phát hiện té ngã:**

$$\|a\| > a_{\text{shock}}, \quad \|a\| \approx 1g$$

- **GPS:** định vị NMEA, hỗ trợ cứu hộ.

Xử lý Máy chủ (Host/Cloud)

Camera Input

ESP32-S3 + OV5640 (5MP)
→ stream JPEG qua Wi-Fi

Máy chủ

- GPU (Jetson Nano, Cloud)
- TensorFlow/PyTorch + OpenCV
- Đồng bộ IMU (MQTT/JSON) + Camera (JPEG)

Kênh truyền

- Wi-Fi: kênh chính
- 4G/LTE (EC800K): dự phòng, SMS/cuộc gọi khẩn

Logic vận hành

- 1 ESP32 phát hiện sơ cấp
- 2 Truyền dữ liệu: Wi-Fi → 4G
- 3 Server xác minh (IMU + HPE)
- 4 Kích hoạt cảnh báo

ESP-IDF Framework

- Hỗ trợ FreeRTOS (đa nhiệm, lõi kép)
- Low-level Access (I2C/SPI, MQTT/HTTP tối ưu)
- Chuyên nghiệp hơn Arduino IDE

Tổng quan Kiến trúc Hệ thống Phát hiện Té ngã

Phân loại hệ thống

Hệ thống phát hiện té ngã hiện đại được chia thành hai nhóm chính:

- **Hệ thống dựa trên Camera**
- **Hệ thống dựa trên Thiết bị đeo**

Ba thành phần cốt lõi

- ❶ **Thiết bị Thu thập Dữ liệu:** Thu thập dữ liệu chuyển động (IMU) và/hoặc hình ảnh (Camera)
- ❷ **Máy chủ Xử lý:** Thực hiện các thuật toán học sâu và logic ra quyết định phức tạp
- ❸ **Hệ thống Truyền thông:** Đảm bảo luồng dữ liệu hai chiều và kích hoạt cảnh báo

Vi điều khiển ESP32

Tại sao chọn ESP32?

- Kiến trúc lõi kép Xtensa LX6
- Xử lý song song hiệu quả
- Tích hợp Wi-Fi, Bluetooth
- Hỗ trợ giao thức MQTT, HTTP

Phân công nhiệm vụ

- **Lỗi 1:** Xử lý thời gian thực (IMU, Kalman Filter)
- **Lỗi 2:** Truyền thông không dây

Lỗi 1

Lỗi 2

ESP32

Cảm biến IMU

IMU (Inertial Measurement Unit)

Tích hợp ba loại cảm biến quan trọng:

Gia tốc kế

- Đo gia tốc tuyến tính
- Hiệu chuẩn từ số nguyên sang đơn vị g
- Vector 3D:
 $\mathbf{a} = [a_x, a_y, a_z]$

Con quay hồi chuyển

- Đo tốc độ góc
- Dựa trên hiệu ứng Coriolis
- Vector 3D:
 $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$

Từ kế

- Tham chiếu hướng từ trường
- Hiệu chỉnh sai số trôi
- Sensor Fusion (Madgwick, Kalman)

Thuật toán Phát hiện Té ngã

Phát hiện dựa trên ngưỡng IMU

Phân tích sự thay đổi đột ngột của gia tốc và tốc độ góc

Shock Event

Gia tốc tổng vượt ngưỡng cao:

$$\|a\| > a_{\text{shock}}$$

Với:

$$\|a\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Post-fall State

- Gia tốc tổng giảm về gần 1g
- Biểu thị trạng thái nằm ngang
- Tốc độ góc có thay đổi lớn

Cảm biến GPS

Module GPS (u-blox NEO-6M) sử dụng **đo tam giác** từ ít nhất 4 vệ tinh, cung cấp tọa

Hệ thống Camera và Máy chủ

Camera Input

- ESP32-S3 + OV5640 5MP
- Nén JPEG
- Truyền qua Wi-Fi (RTSP/HTTP)
- Xác minh hình ảnh

Luồng dữ liệu

- Luồng JSON/MQTT (IMU)
- Luồng JPEG (Camera)
- Đồng bộ hóa dữ liệu
- Giảm báo động giả

Kiến trúc Máy chủ

Phần cứng:

- AWS, Google Cloud
- NVIDIA Jetson Nano
- GPU cho tính toán Tensor

Phần mềm:

- TensorFlow/PyTorch
- OpenCV
- Thuật toán HPE
- Học sâu

Wi-Fi (ESP32) - Kênh chính

- Truyền tải dữ liệu dung lượng lớn
- Video/hình ảnh
- Giao tiếp MQTT với máy chủ
- Độ trễ thấp

4G/LTE (EC800K) - Dự phòng

- Hoạt động khi Wi-Fi không khả dụng
- Hỗ trợ định vị GPS
- Cuộc gọi khẩn cấp tự động
- SMS cảnh báo qua AT commands

1 Thu thập/Xử lý Sơ cấp

- ESP32 thu thập dữ liệu IMU/Camera
- Phát hiện té ngã dựa trên ngưỡng IMU

2 Quyết định Truyền thông

- Nếu phát hiện té ngã → truyền dữ liệu lên máy chủ
- Ưu tiên Wi-Fi, dự phòng 4G

3 Xác minh Máy chủ

- Phân tích hình ảnh (HPE)
- Kết hợp dữ liệu IMU
- Multi-stage Fall Detection Logic

4 Kích hoạt Cảnh báo

- Xác nhận té ngã → ra lệnh cho ESP32
- Kích hoạt Module EC800K gửi SMS/Cuộc gọi

Tại sao chọn ESP-IDF thay vì Arduino IDE?

Hỗ trợ RTOS

- Tích hợp FreeRTOS
- Tận dụng kiến trúc lõi kép
- Đa nhiệm thực sự
- Tác vụ IMU song song với Wi-Fi

Truy cập Cấp thấp

- Truy cập trực tiếp thanh ghi phần cứng
- Cấu hình chi tiết cảm biến (I2C/SPI)
- Tối ưu hóa giao thức mạng
- Hiệu suất thời gian thực

Nội dung chính ở đây...