Análisis del Proyecto: Sistema de Identificación de Plantas

Gemini Code Assist

8 de febrero de 2026

1. Resumen del Proyecto

Este proyecto consiste en un sistema IoT diseñado para identificar especies de plantas mediante reconocimiento de imágenes.

1.1. Flujo de funcionamiento

1. **Captura:** Un módulo ESP32-CAM toma una foto de la planta.
2. **Procesamiento:** La imagen se envía a un servidor local desarrollado en Python (Flask).
3. **Identificación:** El servidor consulta la API de PlantNet con la imagen recibida.
4. **Resultado:** La especie identificada se muestra en una interfaz web.

2. Componentes Principales

- **Hardware:** Módulo ESP32-CAM.
- **Backend:** Servidor Flask (Python) que actúa como intermediario.
- **Servicio Externo:** API de PlantNet (requiere registro).

3. Requisitos y Configuración Crítica

Según la documentación analizada (`README.md` y `app.py`), existen pasos manuales importantes:

1. **Credenciales de API:** Es necesario registrarse en PlantNet para obtener una `API_KEY` y configurarla en el archivo `app.py`.
2. **Firmware del ESP32-CAM:**

- Se debe usar el Arduino IDE para cargar el código.
- Es crucial editar el código fuente para incluir el SSID y contraseña de la red Wi-Fi.
- Se debe configurar la IP del servidor Flask en el firmware: `http://192.168.xxx.xxx:33/identify`.

3. **Puerto del Servidor:** El servidor Flask está configurado en el puerto **33**, por lo que la interfaz web será accesible en `http://localhost:33`.

4. Observaciones

- **Dependencia de Red:** El ESP32-CAM apunta a una IP local específica. Si el router asigna IPs dinámicas (DHCP), será necesario reconfigurar el ESP32 o establecer una IP estática en el servidor.
- **Librerías Python:** Se requiere instalar `Flask` y `requests`.

5. Análisis del Código del Servidor (app.py)

Se ha realizado una revisión del archivo `app.py`, identificando la lógica principal y puntos de mejora técnica.

5.1. Lógica de Negocio

- **Endpoint /identify (POST):** Recibe los datos binarios de la imagen (`request.data`), los guarda localmente como `image.jpg` en la carpeta de templates y los reenvía a la API de PlantNet.
- **Interacción con PlantNet:** Construye la consulta usando el proyecto .[a]ll$_7$ la API Key configurada. Espera una respuesta JSON y busca la clave `bestMatch`.
- **Persistencia:** Utiliza una variable global `plant_type` para almacenar el último resultado identificado.

5.2. Problemas Detectados y Deuda Técnica

1. **Gestión de Estado (Global Variable):** El uso de la variable global `plant_type` no es seguro ("thread-safe"). Si múltiples dispositivos envían imágenes, los resultados se sobrescribirán, mostrando datos incorrectos a los usuarios.
2. **Estructura de Directorios:** El código guarda las imágenes subidas en la carpeta `templates`. En Flask, esta carpeta está reservada para plantillas HTML, mientras que los archivos generados deberían ir en `static` o una carpeta `uploads`.

3. **Seguridad:** La `API_KEY` está escrita directamente en el código ("hardcoded"). Se recomienda usar variables de entorno.

4. **Rutas Redundantes:** Existen dos definiciones de ruta idénticas para servir archivos desde `templates`.

6. Refactorización del Servidor

Se han aplicado las siguientes mejoras al archivo `app.py`:

- **Seguridad:** La API Key ahora se busca en las variables de entorno (`PLANTNET_API_KEY`).
- **Organización:** Las imágenes se guardan en la carpeta `static` en lugar de `templates`.
- **Limpieza:** Se eliminaron rutas duplicadas y librerías no utilizadas.
- **Compatibilidad:** Se añadió una ruta para mantener compatibilidad con el frontend si este solicita imágenes a `/templates/`.

7. Análisis de Acceso Remoto: Uso de ngrok

Se ha evaluado la posibilidad de utilizar **ngrok** para exponer el servidor Flask a internet y permitir el acceso desde fuera de la red local.

7.1. Ventajas

- **Facilidad de uso:** No requiere configurar "Port Forwarding."[en] el router ni lidiar con firewalls complejos.
- **HTTPS:** Proporciona automáticamente una conexión segura (SSL/TLS), lo cual es beneficioso para la transmisión de imágenes.

7.2. Desafío Crítico: URL Dinámica

La versión gratuita de ngrok genera una URL aleatoria cada vez que se inicia (ej. `http://a1b2c3d4.ngrok.io`).

- **Impacto en el ESP32:** Dado que la URL del servidor está "quemada"(hardcoded) en el firmware del ESP32-CAM, **cada vez que se reinicie ngrok, la URL cambiará y el ESP32 dejará de conectar**.
- **Solución:** Sería necesario recompilar y volver a subir el código al ESP32 con la nueva URL cada vez, o adquirir una suscripción de pago de ngrok para tener dominios estáticos.

8. Alternativas para URL Fija (Gratuitas)

Ante la limitación de ngrok, se han identificado alternativas que permiten mantener una URL estable sin coste:

8.1. Localtunnel

Es una herramienta de código abierto que permite exponer puertos locales.

- **Ventaja Principal:** Permite solicitar un subdominio personalizado (ej. `https://mi-planta.loca.lt`). Si está disponible, se asigna, permitiendo configurar el ESP32 una sola vez.
- **Requisito:** Requiere tener Node.js instalado.
- **Comando:** `npx localtunnel --port 33 --subdomain nombre-elegido`
- **Consideración:** Puede presentar una página de advertencia inicial que bloquee peticiones automáticas si no se configuran headers específicos en el cliente (ESP32).

8.2. Cloudflare Tunnel

Es la opción más robusta y profesional, pero tiene un requisito importante para obtener una URL fija.

- **Requisito:** Necesitas ser dueño de un **dominio propio** (ej. `midominio.com`).
- **Costo:** Aunque el servicio de túnel es gratuito, el dominio generalmente tiene un costo anual (aunque existen dominios muy económicos).
- **Funcionamiento:** Permite crear subdominios fijos (ej. `planta.midominio.com`) que no cambian, ideal para el ESP32.
- **Nota:** Existe una versión de prueba ("Quick Tunnel") que no requiere dominio, pero genera URLs aleatorias, por lo que no soluciona el problema del ESP32.

9. Gestión del Repositorio en GitHub

Para acceder y actualizar la carpeta del proyecto en GitHub con los cambios recientes:

1. **Subir Cambios:** Utilizar `git add .`, `git commit` y `git push` para sincronizar la refactorización y documentación.
2. **Acceso Web:** La URL del repositorio se puede obtener mediante el comando `git remote -v`.
3. **Permisos:** Si el repositorio fue clonado de un tercero, se requerirá hacer un "Fork"para tener permisos de escritura.