GotekBackend & ESP32 Firmware Project Overview

This project implements a complete system to remotely manage disk images for a Gotek USB floppy drive running FlashFloppy firmware. It consists of two major components: (1) an ASP.NET Core backend that serves floppy disk images wrapped inside a FAT12 volume and (2) an ESP32-S2/S3 firmware that provides USB Mass Storage emulation, communicates securely with the backend using mutual TLS, and integrates RFID-based disk image selection.

Backend (ASP.NET Core 8)

The backend provides two primary API endpoints: - GET /api/volume?id=: Generates a FAT12-formatted image on-the-fly containing the requested floppy image and an FF.CFG configuration file. The FF.CFG is loaded from disk, allowing runtime customization. - POST /api/volume?id=: Receives a modified FAT12 volume, extracts the floppy disk image (excluding FF.CFG), and writes it back to the Images folder. Technical Details: - Requires client certificates (mutual TLS) for all requests. - Uses Kestrel with server certificate defined in appsettings.json or via docker-compose. - FAT12 builder and extractor implemented manually to keep images minimal (two files only). - Stores only raw floppy images (.adf, .img, .st, .dsk, .ima, .adl) in the Images/ directory. - Uploaded modified volumes are archived into Uploaded/ for traceability.

ESP32 Firmware

The ESP32-S2/S3 firmware implements: - Wi-Fi setup via a minimal web interface (/setup), storing SSID, password, and backend URL in LittleFS. - Mutual TLS (mTLS) using CA, client certificate, and private key embedded as PEM strings. - USB Mass Storage emulation via TinyUSB, exposing the FAT12 volume retrieved from the backend. - LOAD_PIN input to trigger mounting/ejecting of disk image: * High: Reads 64-bit RFID tag ID, fetches FAT12 volume from backend, mounts USB MSC, pulses Gotek SELECT pin. * Low: Pulses Gotek EJECT pin, unmounts USB MSC, and POSTs modified volume back to backend. - Additional APIs: /api/status (JSON system info), /api/remount (force re-fetch & mount). Technical Details: - Uses JA pins wired to Gotek's BTN2 (SELECT) and BTN3 (EJECT). - RFID reader connected via UART, transmitting 64-bit identifiers as hex strings. - LittleFS used to store configuration and temporary stick.img file. - Implements debounce logic on LOAD_PIN for reliable transitions.

Configuration Files

- FF.CFG: FlashFloppy configuration served inside each FAT12 volume. Stored at project root, mounted into container, editable without rebuild. - appsettings.json: Configures logging and Kestrel server certificates for ASP.NET Core. - Dockerfile: Multi-stage build (dotnet publish -> runtime image). Exposes ports 8080 (HTTP) and 8443 (HTTPS). - docker-compose.yml: Defines two services: * certgen: Runs a script to generate a dev CA, server cert, and client certs. * gotek-backend: Runs the ASP.NET backend with certs mounted, volumes mapped (Images, Uploaded, FF.CFG). - certs/gen-certs.sh: Script to generate CA, server, and client certificates for development use. Produces server.pfx for Kestrel, ca.crt for ESP32, and client certs for testing.

Workflow

1. User inserts RFID tag; ESP32 reads 64-bit ID. 2. ESP32 requests /api/volume?id= from backend over HTTPS (mTLS). 3. Backend builds FAT12 image containing requested floppy and FF.CFG,

returns to ESP32. 4. ESP32 mounts the FAT12 via USB MSC and pulses Gotek SELECT (load). 5. When LOAD_PIN goes low, ESP32 pulses Gotek EJECT, unmounts MSC, and POSTs updated FAT12 back to backend. 6. Backend extracts updated floppy image and replaces the one in Images/.

Technical Notes

- FAT12 implementation is minimal: 512-byte sectors, 2 FATs, 32 root entries, cluster size = 1 sector. - Only two files in root: and FF.CFG. - ESP32 uses TinyUSB callbacks for read/write/test-ready/capacity handling. - Backend enforces client certificates; ESP32 must embed CA and device cert. - Sound effects: FF.CFG enables speaker/fake drive noise, disables OLED/LED display. - Security: Replace PEM placeholders in firmware with actual CA, client cert, and private key.