Project Proposal: RFID Access Control (ESP32/TTGO LoRa32)

GitHub `github.com/Silent0Wings`

Project Description

Problem

Campus doors often rely on manual checks or weak access control. This causes unauthorized entry, unreliable attendance data, and difficult rule updates. A secure, traceable, and networked system is required for better safety and auditability.

Proposed Solution

An ESP32 TTGO LoRa32 node integrates an RFID reader, touch input, relay lock, and OLED UI. It connects to a Wi-Fi backend for authentication, logging, and policy updates through bidirectional messages. Two devices will sync locally via ESP-NOW (BLE fallback). Offline mode uses a 24-hour cached allow list with queued logs. Security applies TLS, rotating tokens, and rate limits. Safety includes relay isolation, fuse, and ESD protection. Success is measured by sub-300 ms decision latency, zero missed reads, and consistent policy propagation.

Meets Required Aspects

Aspect	How Addressed
Sensors/Actuators	MFRC522, touch sensor, relay lock. LEDs only for status.
Microcontroller	ESP32 TTGO LoRa32 (Arduino IDE).
Internet	Wi-Fi.
Distributed Logic	Device: scan, checks, actuation. Server: auth, roles, logs, analytics.
Bidirectional Messages	Uplink events; downlink policies/decisions.
UI	OLED; ESP32 web page (login, registration, diagnostics).
Local Wireless (demo)	Two ESP32 nodes via ESP-NOW; BLE fallback.

Security & Offline Policy (Summary)

Aspect	Implementation
Connection	Enforce TLS-secured server link
Sign-ins	Tokens expire every 15–30 min
Abuse Control	Rate limit; lock after retries
Logs	Ordered; tamper-protected
Offline Mode	24h cache; resend logs when online
Secrets	Store securely; rotate; no hard-coding

Hardware List (Bill of Materials)

Category	Items
Controller	TTGO LoRa32 (ESP32)
Sensors	MFRC522; touch; reed switch; tamper switch
Actuators	Relay or lock driver; buzzer; R/G LEDs + resistors
UI	SSD1306 OLED 128x64 (I2C)
Power	5 V adapter; buck; fuse; TVS; decoupling caps
Storage/Timing	microSD; RTC (optional)
Cabling/Enclosure	Jumpers; PCB/perfboard; labeled connectors; case
IDs	RFID cards and key fobs

Architecture & Functional Blocks

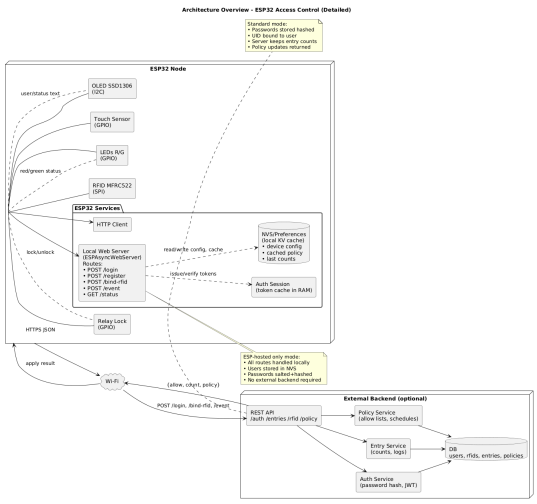

Figure 1: Architecture overview: device, sensors and actuators, Wi-Fi to backend, local wireless.

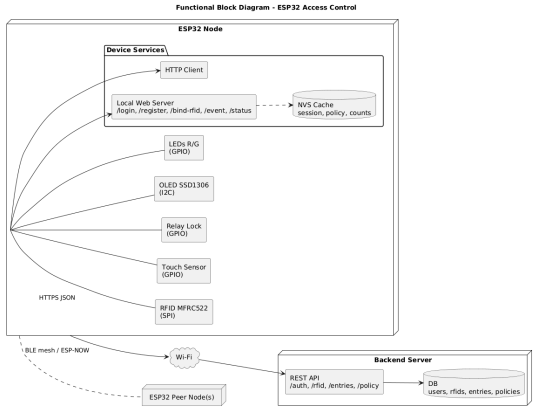

Figure 2: Functional blocks and their relations.

System Design

Validation & Metrics Summary

Check	Target / Method
Decision Latency	$\leq$ 300 ms (LAN)
Read Accuracy	0 misses / 1000 scans
Audit Logs	Ordered & complete
Offline Mode	24h cache; sync queued logs
Security	TLS, token rotation, rate limit
Testing	RFID, relay, Wi-Fi, safety

Milestones

ID	Description
M1	HW bring-up & OLED UI
M2	API v0: auth, events, policy
M3	End-to-end demo with ESP-NOW
M4	Security hardening & logging
M5	Final tests & report

Risks & Mitigations

Risk	Mitigation
Server latency spikes	Local cache + exponential backoff
Token desync	Short TTL + explicit refresh path
RF interference	Retries + RSSI check + wired open override
Power faults	Fuse + TVS + brownout detection

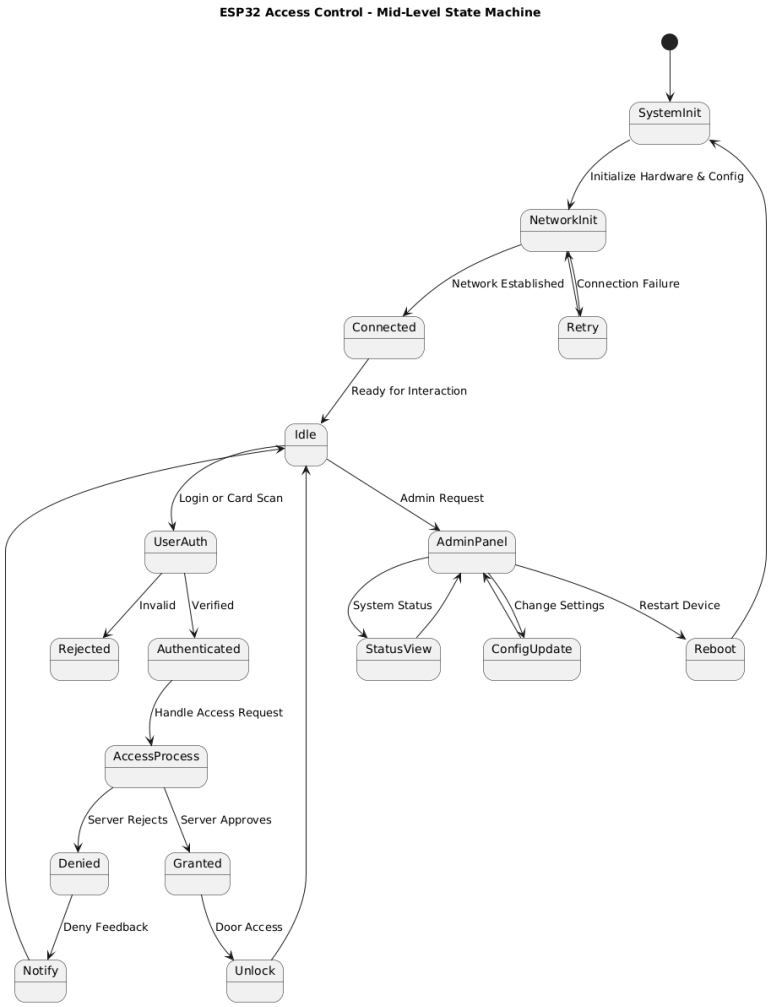

Figure 3: Mid-level state machine: boot, online, offline, decision, and recovery paths.

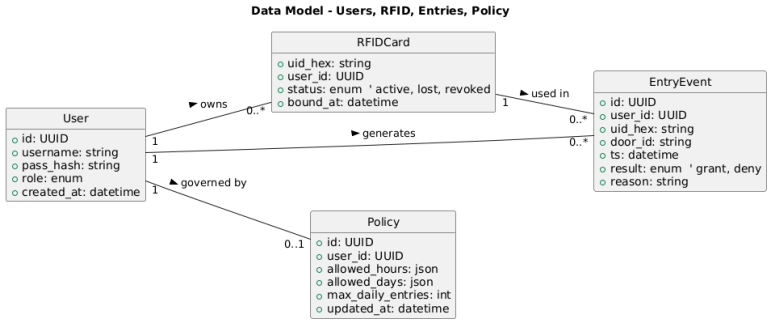

Figure 4: Data model: device events, API endpoints, database entities, policy downlink.

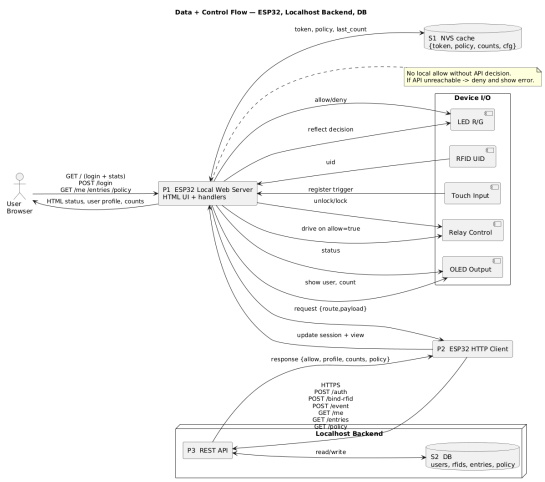

Figure 5: Bidirectional flow: uplink events, downlink policy, offline buffering, enforcement points.

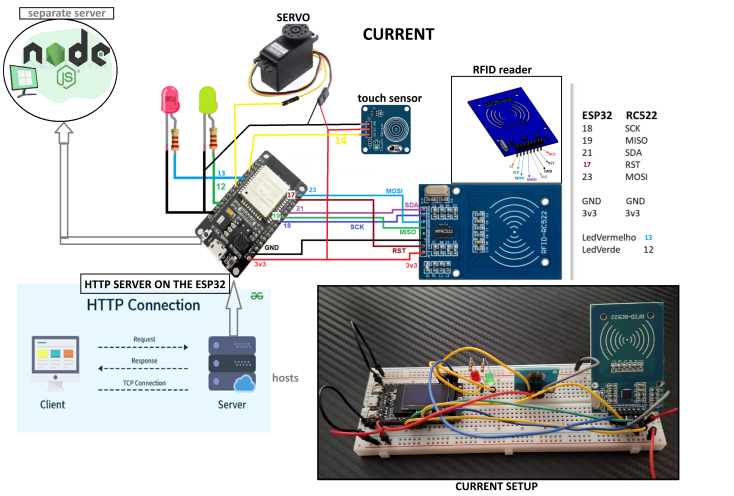

Figure 6: Wiring schematic with protections: relay isolation, fuse, TVS, decoupling.

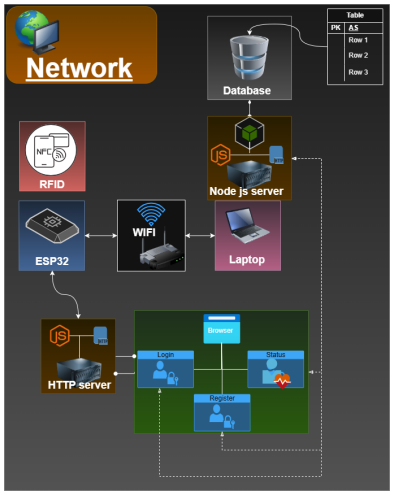

Figure 7: System architecture: RFID reader, ESP32 controller, browser UI, backend services, and database.

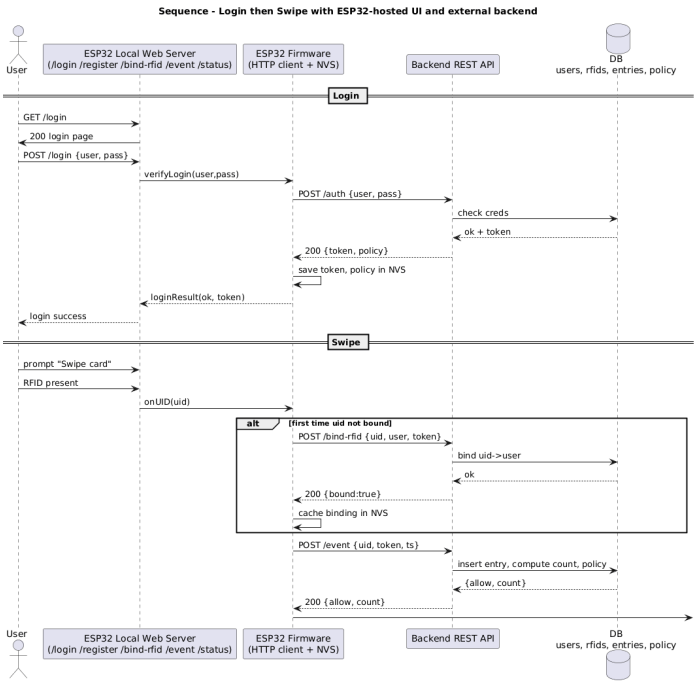

Figure 8: Sequence diagram: card tap → device event → API auth → policy decision → actuation → log persist → acknowledgement.