Contents

Field Trainer Technical Overview

Version: 0.5.1 **Last Updated:** November 2025 **System Type:** Distributed Athletic Performance Training Platform **Target Platform:** Raspberry Pi Zero W (Field Devices) / Raspberry Pi 5 8GB (Gateway)

Executive Summary

Field Trainer is a sophisticated athletic performance training system designed for amateur coaches and athletic trainers. The system uses a mesh network of wireless sensor devices to track athlete performance through training courses, providing real-time feedback, performance metrics, and comprehensive session management.

Key Features: - Wireless mesh network topology (Batman-adv) supporting up to 6+ field devices - Real-time touch detection and performance tracking - Dual-mode operation: Online (internet-connected) and Offline (Access Point) - Multi-athlete concurrent training support - Team and athlete management with full roster capabilities - Performance history tracking with personal records and achievements - Audio feedback system with TTS (male/female voices) - LED visual feedback on field devices and gateway - Web-based dual interface: Admin (port 5000) and Coach (port 5001) - SQLite database for persistent storage - Automatic network failover for field use

---

System Architecture

High-Level Architecture

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    Gateway (Device 0)                       +
+                    Raspberry Pi 5 8GB                       +
+                                                             +
+  ++++++++++++++++  ++++++++++++++++  ++++++++++++++++    +
+  + Admin UI     +  + Coach UI     +  +  REGISTRY    +    +
+  + Port 5000    +  + Port 5001    +  +  (Shared)    +    +
+  ++++++++++++++++  ++++++++++++++++  ++++++++++++++++    +
+                                                             +
+  ++++++++++++++++  ++++++++++++++++  ++++++++++++++++    +
+  + Database     +  + Network Mgr  +  + LED/Audio    +    +
+  + SQLite       +  + Auto-Switch  +  + MAX98357A    +    +
+  ++++++++++++++++  ++++++++++++++++  ++++++++++++++++    +
+                                                             +
+  ++++++++++++++++  Audio: Ready sounds + Male/Female TTS  +
+  +  Heartbeat   +  LEDs: 15 NeoPixels (status indication) +
+  +   Master     +  Networking: wlan0 (mesh) + wlan1 (AP)  +
+  ++++++++++++++++  bat0: 192.168.99.100                    +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                            +
                    Batman-adv Mesh
                   (192.168.99.0/24)
                            +
    +++++++++++++++++++++++++++++++++++++++++++++++++
    +                       +                       +
++++++++++            ++++++++++++            ++++++++++
+Device 1+            + Device 2 +            +Device N+
+Pi Zero +            + Pi Zero  +            +Pi Zero +
+MPU6050 +            + MPU6050  +            +MPU6050 +
+MAX98357+            + MAX98357 +            +MAX98357+
+15 LEDs +            + 15 LEDs  +            +15 LEDs +
++++++++++            ++++++++++++            ++++++++++
```

Component Responsibilities

1. **Gateway (Device 0)**
   - Web interface hosting (admin + coach)
   - Course deployment and lifecycle management
   - Touch event processing and athlete attribution
   - Performance data storage
   - Audio announcements via MAX98357A (ready sounds + male/female TTS)
   - Server-side LED status indication (15 NeoPixels)
   - Network mode management (online/offline)
   - Heartbeat master (monitors all field devices)
2. **Field Devices (Devices 1-N)**

- Touch detection via MPU6050 accelerometer
- LED visual feedback (15 NeoPixels)
- Audio feedback via MAX98357A (action commands)
- Mesh network communication
- Heartbeat status reporting to gateway
- Remote command execution (deploy, activate, deactivate, shutdown)

Hardware Requirements

Gateway Device (Device 0)

Production Hardware (All Required): - Raspberry Pi 5 8GB RAM - MicroSD card: 32GB+ (Class 10 or better) - Power supply: 5V/5A USB-C - **WiFi #1 (wlan0)**: Built-in WiFi - Mesh network (Batman-adv) - **WiFi #2 (wlan1)**: USB WiFi adapter - Internet client OR Access Point - Recommended: Panda PAU09 or AC600 (MediaTek chipset) - Must support AP mode for offline operation - Avoid: Realtek RTL8188EUS (no AP mode support) - **Audio**: MAX98357A I2S amplifier (onboard audio disabled) - Provides ready notification sounds - Male and female voice TTS for action commands - **LEDs**: WS2812B NeoPixel LED strip (GPIO 18, 15 LEDs) - **HDMI monitor** for direct access - **USB keyboard** for emergency recovery

Network Configuration: - bat0 (mesh): 192.168.99.100/24 - wlan1 (internet/AP): Dynamic or 192.168.10.1 (AP mode)

Field Devices (Devices 1-N)

Per Device (All Required): - Raspberry Pi Zero W - MicroSD card: 32GB (Class 10) - Power supply: 5V/2.5A Micro-USB - **MPU6050 accelerometer** (I2C) - VCC → 3.3V - GND → GND - SDA → GPIO 2 (I2C SDA) - SCL → GPIO 3 (I2C SCL) - **MAX98357A I2S amplifier** for audio feedback - Male and female voice TTS for action commands - **WS2812B NeoPixel LED strip** (GPIO 18, 15 LEDs) - Built-in WiFi for mesh network (wlan0)

Network Configuration: - All devices on same mesh: `bat0` interface (192.168.99.0/24) - Gateway mesh IP: 192.168.99.100 - Field device IPs: 192.168.99.101, .102, .103, etc. - MAC filtering enabled for security

---

File Structure

```
/opt/
+++ field_trainer_main.py         # Main entry point, starts dual Flask apps
+++ field_trainer_web.py          # Admin interface (port 5000) - legacy
+++ coach_interface.py            # Coach interface (port 5001) - primary UI
+++ field_trainer_core.py         # Core field device logic (for Devices 1-N)
+++ field_client_connection.py    # Mesh network client communication
+++ audio_manager.py              # Audio playback and TTS
+++ led_controller.py             # NeoPixel LED control
+++ mpu65xx_touch_sensor.py       # MPU6050 accelerometer interface
+++ athlete_routes.py             # Flask blueprint for athlete management
+++ athlete_helpers.py             # Athlete import/export helpers
+
+++ field_trainer/                 # Core module
+   +++ __init__.py
+   +++ ft_version.py              # VERSION = "0.5.1"
+   +++ ft_registry.py             # REGISTRY singleton - system state
+   +++ ft_config.py               # System configuration constants
+   +++ ft_models.py               # Data models (NodeInfo, etc.)
+   +++ ft_courses.py              # Course loading (legacy JSON)
+   +++ ft_mesh.py                 # Batman-adv mesh utilities
+   +++ ft_heartbeat.py            # Device heartbeat monitoring
+   +++ ft_touch.py                # Touch event processing
+   +++ ft_led.py                  # LED manager for server LEDs
+   +++ ft_audio.py                # Audio manager interface
+   +++ ft_webapp.py               # Flask app utilities
+   +++ db_manager.py              # Database manager (53KB, comprehensive)
+   +++ settings_manager.py        # System settings management
+   +
+   +++ templates/                 # Jinja2 HTML templates (22 files)
+   +   +++ base.html              # Base template with navigation
+   +   +++ index.html             # Admin dashboard
+   +   +++ health.html            # Service health check page
+   +   +++ settings.html          # System settings + network controls
+   +   +++ session_setup.html     # Session creation and management
+   +   +++ athletes.html          # Athlete roster management
+   +   +++ teams.html             # Team management
+   +   +++ ...                    # Performance, history, etc.
+   +
+   +++ static/                    # Static web assets
+   +   +++ css/
+   +   +   +++ custom.css
+   +   +   +++ style.css
+   +   +++ js/
+   +   +   +++ main.js
+   +   +   +++ app.js
+   +   +   +++ settings.js
+   +   +++ vendor/               # Local CDN replacements (offline mode)
+   +   +   +++ bootstrap/        # Bootstrap 5.3.2
+   +   +   +++ bootstrap-icons/  # Icons + fonts
+   +   +   +++ sortable/         # Sortable.js for drag-drop
+   +   +++ audio/                # Audio files
+   +       +++ .placeholder
+   +
+   +++ athletic_platform/         # Performance tracking module
+       +++ bridge_layer.py        # Bridge between sessions and analytics
+       +++ ...
+
+++ scripts/                       # Utility scripts
```

```
+   +++ ft-network-manager.py      # Network auto-switching daemon
+   +++ github_deploy.sh           # Git deployment script
+   +++ init_mac_filter.sh         # MAC filter initialization
+   +++ manage_mac_filter.sh       # MAC filter management
+   +++ vscode_setup.sh            # VSCode remote setup
+
+++ data/                          # Runtime data
+   +++ field_trainer.db           # SQLite database
+   +++ network-config.json        # Network manager configuration
+   +++ network-status.json        # Current network status
+   +++ *.backup*                  # Database backups
+
+++ test_*.py                      # Comprehensive test suites
+   +++ test_attribution_logic.py  # Touch attribution tests
+   +++ test_concurrency.py        # Multi-athlete concurrency
+   +++ test_database_integrity.py # Database validation
+   +++ test_load_stress.py        # Load testing
+   +++ test_touch_sequences.py    # Touch sequence validation
+
+++ backups/                       # System backups
    +++ phase*_*/                  # Timestamped backups
```

Configuration Files (System-wide)

```
/etc/
+++ systemd/system/
+   +++ field-trainer-server.service   # Main service
+   +++ ft-network-manager.service     # Network manager service
+
+++ hostapd/
+   +++ hostapd-ft.conf               # Access Point config (offline mode)
+
+++ dnsmasq.d/
+   +++ ft-ap.conf                    # DHCP/DNS for AP mode
+
+++ avahi/services/
+   +++ fieldtrainer.service          # mDNS (fieldtrainer.local)
+
+++ field-trainer-macs.conf           # MAC filter whitelist

/home/pi/
+++ restore-network.sh                # Emergency network restore
+++ RESTORE_INSTRUCTIONS.txt          # Recovery documentation
+++ QUICK_REFERENCE.txt               # Quick reference card
```

Core Components

1. REGISTRY (Singleton)

File: `/opt/field_trainer/ft_registry.py`

The REGISTRY is the heart of the system - a thread-safe singleton that maintains all runtime state.

Key Responsibilities: - Device management (nodes dictionary with NodeInfo objects) - Course lifecycle (deploy, activate, deactivate) - Touch event routing - System logging (ring buffer, max 200 entries) - LED control (server-side Device 0) - Audio management (TTS and playback) - Database integration - Snapshot generation for UI

Key Data Structures:

```
self.nodes: Dict[str, NodeInfo]          # node_id -> NodeInfo
self.logs: deque(maxlen=200)             # Ring buffer of log entries
self.course_status: str                   # "Inactive", "Deployed", "Active"
self.selected_course: Optional[str]       # Currently loaded course
self.assignments: Dict[str, str]          # node_id -> action
self.courses: Dict                        # Loaded from database
self._touch_handler: Callable            # Set by coach_interface
```

Thread Safety: - Uses `threading.Lock()` for nodes dictionary - All public methods are thread-safe - Used by both Flask apps simultaneously

Key Methods: - `snapshot()` - Returns complete system state for UI - `deploy_course(course_name)` - Deploy course to field devices - `activate_course()` - Start active training mode - `deactivate_course()` - Stop training, clear LEDs - `send_command(node_id, command)` - Send command to field device - `log(msg, level, source, node_id)` - Add structured log entry

2. DatabaseManager

File: `/opt/field_trainer/db_manager.py` (53KB)

Comprehensive SQLite database manager with 15 tables.

Key Responsibilities: - Course CRUD operations - Team and athlete management - Session and run management - Performance tracking - Personal records and achievements - Settings persistence - Contact and medical information

Key Methods: - `get_all_courses()` - List all courses - `get_course(course_id)` - Get course with actions - `create_team(name, age_group)` - Create team - `get_team_athletes(team_id)` - Get team roster - `create_session(team_id, course_id)` - Start new session - `create_run(session_id, athlete_id)` - Queue athlete - `record_touch(run_id, device_id, timestamp)` - Record touch event - `update_personal_record(athlete_id, metric, value)` - Track PRs

Connection Management: - Uses context managers for connection safety - Automatic connection pooling - Foreign key enforcement enabled

3. Coach Interface (Primary UI)

File: `/opt/coach_interface.py` (94KB)

Flask application serving the primary user interface on port 5001.

Key Features: - Session setup and management - Team and athlete management - Live session monitoring - Performance history review - Settings management - Network mode controls

Key Routes:

```
GET  /                          # Dashboard
GET  /health                    # Service health check
GET  /settings                  # Settings page
GET  /session-setup             # Session setup wizard
POST /api/session/create        # Create new session
POST /api/session/{id}/start    # Start session
POST /api/network/force-mode    # Force online/offline mode
GET  /api/network/status        # Network status
```

Touch Attribution Logic: Sophisticated algorithm in `find_athlete_for_touch()`: 1. Identifies athletes at correct sequence position (gap == 1) 2. Handles athletes who skip devices (gap > 1) 3. Ignores repeated touches (gap == 0) and backwards motion (gap < 0) 4. Supports multiple simultaneous athletes

Session State:

```
active_session_state = {
    'session_id': str,
    'active_runs': {run_id: {...}},  # Multiple concurrent athletes
    'device_sequence': [device_ids],
    'total_queued': int
}
```

4. Admin Interface (Legacy)

File: `/opt/field_trainer_web.py`

Flask application on port 5000 - original interface, now supplementary.

Responsibilities: - Course deployment (primary function) - System monitoring - Device status - System logs

Note: Admin and Coach interfaces share the same REGISTRY instance for state synchronization.

5. Network Manager

File: `/opt/scripts/ft-network-manager.py`

Python daemon that automatically switches between online and offline modes.

Operation Modes:

1. **Online Mode (Default)**
    - wlan1 connects to home/office WiFi (wpa_supplicant)
    - Internet access available
    - Dynamic IP via DHCP
    - Monitors internet connectivity every 60 seconds
2. **Offline Mode (Access Point)**
    - wlan1 becomes WiFi hotspot

- SSID: `Field_Trainer`
- Password: `RaspberryField2025`
- Gateway IP: 192.168.10.1
- DHCP range: 192.168.10.10-100
- mDNS: `fieldtrainer.local`

Automatic Failover: - Checks internet every 60 seconds (8.8.8.8, 1.1.1.1) - After 3 consecutive failures → switch to AP mode - When internet returns → wait 5 minutes for stability - Automatically switches back to online mode

Manual Control:

```
sudo python3 /opt/scripts/ft-network-manager.py force-online
sudo python3 /opt/scripts/ft-network-manager.py force-offline
sudo python3 /opt/scripts/ft-network-manager.py status
```

Configuration: `/opt/data/network-config.json`

```
{
    "network_mode": {
        "current": "online",
        "auto_switch": true
    },
    "monitoring": {
        "internet_check_interval": 60,
        "internet_check_retries": 3,
        "failback_delay": 300
    },
    "access_point": {
        "enabled": false,
        "ssid": "Field_Trainer",
        "password": "RaspberryField2025",
        "ip": "192.168.10.1"
    }
}
```

6. Audio Manager

File: `/opt/audio_manager.py`

TTS and audio playback system using `mpg123`, `pico2wave`, and MAX98357A I2S amplifier.

Hardware: - MAX98357A I2S amplifier (onboard audio disabled) - Connected via I2S GPIO pins - Digital audio output for better quality

Audio Types: 1. **Ready Notification Sounds** - Pre-recorded alerts (course ready, session start, etc.) 2. **Male Voice TTS** - Action commands and announcements 3. **Female Voice TTS** - Action commands and announcements

Features: - Male/female voice selection - Volume control (0-100%) - Audio file playback (MP3, WAV) - Text-to-speech conversion - Automatic file caching

Key Methods: - `speak(text, voice=None)` - TTS announcement - `play_audio_file(filename)` - Play ready notification sounds - `set_voice_gender(gender)` - Switch between male/female voice - `set_volume(percent)` - Adjust volume

7. LED Manager

File: `/opt/led_controller.py` and `/opt/field_trainer/ft_led.py`

Controls WS2812B NeoPixel LED strips on gateway and field devices.

LED States:

```
IDLE        # Blue pulse
DEPLOYED    # Cyan solid
ACTIVE      # Green solid
TRIGGERED   # Yellow flash → Green
COMPLETE    # Rainbow sequence
ERROR       # Red flash
```

Hardware: - GPIO 18 (PWM) - **15 LEDs** per device (gateway and field devices) - 5V power supply - Brightness configurable (default: 128/255)

Database Schema

Database File: `/opt/data/field_trainer.db` **Type:** SQLite 3 **Size:** ~500KB typical **Tables:** 15

Core Tables

1. teams Stores team information with metadata.

```
team_id TEXT PRIMARY KEY
name TEXT NOT NULL UNIQUE
age_group TEXT
active INTEGER DEFAULT 1
created_at TIMESTAMP
updated_at TIMESTAMP
```

2. athletes Comprehensive athlete roster with demographics.

```
athlete_id TEXT PRIMARY KEY
team_id TEXT NOT NULL (FK → teams)
name TEXT NOT NULL
jersey_number INTEGER
athlete_number TEXT
age INTEGER
birthdate TEXT
gender TEXT
position TEXT
```

```
created_at TIMESTAMP
updated_at TIMESTAMP
```

3. courses Training course definitions with metadata.

```
course_id INTEGER PRIMARY KEY AUTOINCREMENT
course_name TEXT NOT NULL UNIQUE
description TEXT
category TEXT DEFAULT 'Agility'
mode TEXT DEFAULT 'sequential'
course_type TEXT DEFAULT 'conditioning'
num_devices INTEGER DEFAULT 6
total_devices INTEGER DEFAULT 6
distance_unit TEXT DEFAULT 'yards'
total_distance INTEGER DEFAULT 0
diagram_svg TEXT
layout_instructions TEXT
is_builtin INTEGER DEFAULT 0
version TEXT DEFAULT "1.0"
created_at TIMESTAMP
updated_at TIMESTAMP
```

4. course_actions Sequence of actions within a course.

```
action_id INTEGER PRIMARY KEY AUTOINCREMENT
course_id INTEGER NOT NULL (FK → courses)
sequence INTEGER NOT NULL
device_id TEXT NOT NULL
device_name TEXT
action TEXT NOT NULL
action_type TEXT NOT NULL
audio_file TEXT
instruction TEXT
min_time REAL DEFAULT 1.0
max_time REAL DEFAULT 30.0
triggers_next_athlete BOOLEAN DEFAULT 0
marks_run_complete BOOLEAN DEFAULT 0
UNIQUE(course_id, sequence)
```

5. sessions Training session records.

```
session_id TEXT PRIMARY KEY
team_id TEXT NOT NULL (FK → teams)
course_id INTEGER NOT NULL (FK → courses)
status TEXT CHECK IN ('setup', 'active', 'completed', 'incomplete')
started_at TIMESTAMP
completed_at TIMESTAMP
audio_voice TEXT CHECK IN ('male', 'female')
course_deployed INTEGER DEFAULT 0
```

```
deployment_timestamp TEXT
notes TEXT
created_at TIMESTAMP
```

6. runs Individual athlete run records.

```
run_id TEXT PRIMARY KEY
session_id TEXT NOT NULL (FK → sessions)
athlete_id TEXT NOT NULL (FK → athletes)
course_id INTEGER NOT NULL (FK → courses)
queue_position INTEGER NOT NULL
status TEXT CHECK IN ('queued', 'running', 'completed', 'incomplete', 'dropped', 'absent')
started_at TIMESTAMP
completed_at TIMESTAMP
total_time REAL
UNIQUE(session_id, queue_position)
```

7. segments Per-segment timing data within a run.

```
segment_id INTEGER PRIMARY KEY AUTOINCREMENT
run_id TEXT NOT NULL (FK → runs)
from_device TEXT NOT NULL
to_device TEXT NOT NULL
sequence INTEGER NOT NULL
expected_min_time REAL NOT NULL
expected_max_time REAL NOT NULL
actual_time REAL
touch_detected BOOLEAN DEFAULT 0
touch_timestamp TIMESTAMP
alert_raised BOOLEAN DEFAULT 0
alert_type TEXT CHECK IN ('missed_touch', 'too_slow', 'too_fast')
UNIQUE(run_id, sequence)
```

Performance Tracking Tables

8. performance_history Historical performance records.

```
record_id TEXT PRIMARY KEY
athlete_id TEXT NOT NULL (FK → athletes)
run_id TEXT (FK → runs)
session_id TEXT (FK → sessions)
metric_name TEXT NOT NULL
metric_value REAL NOT NULL
metric_unit TEXT
is_personal_record BOOLEAN DEFAULT 0
course_id INTEGER
segment_data TEXT
recorded_at TIMESTAMP
notes TEXT
```

9. personal_records Current best performances per athlete per metric.

```
pr_id TEXT PRIMARY KEY
athlete_id TEXT NOT NULL (FK → athletes)
metric_name TEXT NOT NULL
current_best REAL NOT NULL
metric_unit TEXT
achieved_at TIMESTAMP NOT NULL
run_id TEXT
previous_best REAL
improvement REAL
UNIQUE(athlete_id, metric_name)
```

10. achievements Badges and milestones.

```
achievement_id TEXT PRIMARY KEY
athlete_id TEXT NOT NULL (FK → athletes)
badge_type TEXT NOT NULL
badge_name TEXT NOT NULL
description TEXT
criteria TEXT
earned_at TIMESTAMP
run_id TEXT
metric_value REAL
```

Configuration Tables

11. settings System-wide settings (key-value store).

```
id INTEGER PRIMARY KEY AUTOINCREMENT
setting_key TEXT UNIQUE NOT NULL
setting_value TEXT NOT NULL
description TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
```

Common settings: - `voice_gender`: "male" or "female" - `system_volume`: "0" to "100" - `deployment_timeout`: seconds - `distance_unit`: "yards" or "meters"

12. coach_preferences Per-coach preferences (future multi-coach support).

```
coach_id TEXT PRIMARY KEY
distance_unit TEXT DEFAULT 'yards'
deployment_timeout INTEGER DEFAULT 300
audio_voice TEXT DEFAULT 'male'
theme TEXT DEFAULT 'light'
created_at TEXT
updated_at TEXT
```

Athlete Extended Data

13. athlete_contacts Emergency contacts and guardians.

```
contact_id INTEGER PRIMARY KEY AUTOINCREMENT
athlete_id TEXT NOT NULL (FK → athletes)
name TEXT NOT NULL
relationship TEXT
phone TEXT
email TEXT
is_primary INTEGER DEFAULT 0
can_pickup INTEGER DEFAULT 1
created_at TIMESTAMP
```

14. athlete_medical Medical information for safety.

```
medical_id INTEGER PRIMARY KEY AUTOINCREMENT
athlete_id TEXT NOT NULL UNIQUE (FK → athletes)
allergies TEXT
allergy_severity TEXT
medical_conditions TEXT
medications TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
```

Network Architecture

Batman-adv Mesh Network

Purpose: Wireless communication between gateway and field devices

Configuration: - Interface: `bat0` - Underlying interface: `wlan0` (built-in WiFi on all devices) - Network: 192.168.99.0/24 - Gateway: 192.168.99.100 (Device 0) - Field devices: 192.168.99.101, .102, .103, etc.

Features: - Self-healing mesh topology - Automatic route optimization - No infrastructure required - Range: ~100m outdoors per hop - MAC filtering for security

Commands:

```
batctl if              # Show mesh interfaces
batctl n               # Show mesh neighbors
batctl o               # Show mesh originators
sudo batctl ping <IP>  # Mesh ping
```

Internet Connectivity (wlan1)

Gateway Only - Field devices don't need internet

Online Mode (Default)

- Interface: wlan1
- DHCP client via `wpa_supplicant`
- Connects to home/office WiFi
- IP: Assigned by router
- Used for: Updates, git access, remote access

Offline Mode (Access Point)

- Interface: wlan1
- Static IP: 192.168.10.1/24
- Services:
  - `hostapd-ft`: Access Point
  - `dnsmasq-ft`: DHCP + DNS
  - `avahi-daemon`: mDNS (fieldtrainer.local)

Isolation: - Mesh network (192.168.99.x) and AP network (192.168.10.x) are separate - No routing between networks - Mesh always available regardless of internet status

Network Topology Diagram

```
Internet
   +
   + (wlan1 online mode)
   +
+++++++++++++++++++++++++++++++++++++++++++
+  Gateway (Device 0) - Raspberry Pi 5    +
+  - eth0: 192.168.7.x (dev only)         +
+  - wlan0: Mesh interface                +
+  - bat0: 192.168.99.100                 +
+  - wlan1: Internet OR Access Point      +
+  - 15 LEDs + MAX98357A Audio            +
+++++++++++++++++++++++++++++++++++++++++++
   +
   + (wlan0 → bat0 mesh)
   +
   +++++++++++++++++++++++++++++++
   +         +         +         +
++++++++ ++++++++ ++++++++ ++++++++
+Dev 1 + +Dev 2 + +Dev 3 + +Dev N +
+Pi 0W + +Pi 0W + +Pi 0W + +Pi 0W +
+.99.  + +.99.  + +.99.  + +.99.  +
+101   + +102   + +103   + +10N   +
+15LED + +15LED + +15LED + +15LED +
+Audio + +Audio + +Audio + +Audio +
++++++++ ++++++++ ++++++++ ++++++++
  wlan0    wlan0    wlan0    wlan0
```

```
Coaches (Offline Mode)
   +
   + (WiFi: Field_Trainer)
   +
+++++++++++++++++++++++
+  wlan1 AP Mode      +
+  192.168.10.1       +
+  SSID: Field_Trainer+
+++++++++++++++++++++++
```

MAC Filtering Security

Purpose: Prevent unauthorized devices from joining mesh

Configuration: `/etc/field-trainer-macs.conf`

```
DEVICE_0_MAC="88:a2:9e:0d:29:98"
DEVICE_1_MAC="b8:27:eb:60:3c:54"
DEVICE_2_MAC="b8:27:eb:bd:c0:8f"
# ... etc
```

Management Script: `/opt/scripts/manage_mac_filter.sh`

```
sudo /opt/scripts/manage_mac_filter.sh enable   # Enable filtering
sudo /opt/scripts/manage_mac_filter.sh disable  # Disable filtering
sudo /opt/scripts/manage_mac_filter.sh status   # Check status
sudo /opt/scripts/manage_mac_filter.sh list     # List MACs
```

Implementation: - Uses `batctl meshif bat0 ap_isolation` and custom iptables rules - Whitelist approach (only approved MACs allowed) - Applies to bat0 interface only (not wlan1 AP)

API Endpoints

Admin Interface (Port 5000)

System Status

```
GET  /status
Returns: {
  "status": "ok",
  "course_status": "Active",
  "devices": 6,
  "version": "0.5.1"
}
```

Course Management

```
POST /deploy
Body: {"course_name": "Course Name"}
Returns: {"status": "success", "message": "Course deployed"}
```

```
POST /activate
Returns: {"status": "success"}

POST /deactivate
Returns: {"status": "success"}
```

Device Commands

```
POST /send_command
Body: {
  "node_id": "device_2",
  "command": "shutdown"
}
Returns: {"status": "success"}
```

Logs

```
GET  /api/logs
Returns: [
  {
    "ts": "2025-11-09T12:34:56Z",
    "level": "info",
    "source": "controller",
    "node_id": null,
    "msg": "Course activated"
  },
  ...
]

POST /api/logs/clear
Returns: {"status": "success"}
```

Coach Interface (Port 5001)

Health Check

```
GET  /health
GET  /health?format=json
Returns: {
  "service_name": "Field Trainer Coach Interface",
  "version": "0.5.1",
  "pid": 12345,
  "uptime": "2h 15m",
  "registry_id": "abc123",
  "port": 5001,
  "courses_loaded": 8,
  "nodes_connected": 6,
  "course_status": "Active",
  "active_session": "Session Name",
  "active_runs": 3,
  "started_at": "2025-11-09T10:19:23Z"
}
```

Network Management

```
GET  /api/network/status
Returns: {
  "mode": "online",
  "auto_switch": true,
  "ssid": "smithhome",
  "ap_ssid": "Field_Trainer",
  "monitoring": {
    "check_interval": 60,
    "retries": 3,
    "failback_delay": 300
  }
}

POST /api/network/force-mode
Body: {"mode": "online" | "offline"}
Returns: {"status": "success", "message": "..."}

POST /api/network/auto-switch
Body: {"enabled": true | false}
Returns: {"status": "success"}
```

Session Management

```
POST /api/session/create
Body: {
  "session_name": "Practice Session",
  "team_id": "team_123",
  "course_id": 1,
  "audio_voice": "male"
}
Returns: {
  "status": "success",
  "session_id": "sess_123"
}

POST /api/session/<session_id>/start
Returns: {"status": "success"}

POST /api/session/<session_id>/add-athlete
Body: {
  "athlete_id": "athlete_456",
  "queue_position": 3
}
Returns: {
  "status": "success",
  "run_id": "run_789"
}

GET  /api/session/<session_id>/status
Returns: {
  "session_id": "sess_123",
  "status": "active",
  "active_runs": [
    {
      "run_id": "run_789",
      "athlete_name": "John Doe",
      "status": "running",
      "current_device": "device_3",
      "elapsed_time": 12.5
    },
    ...
  ]
}
```

Team and Athlete Management

```
GET  /api/teams
Returns: [{"team_id": "...", "name": "...", "age_group": "..."}, ...]

POST /api/teams/create
Body: {"name": "Team Name", "age_group": "U15"}
Returns: {"status": "success", "team_id": "..."}

GET  /api/teams/<team_id>/athletes
Returns: [{"athlete_id": "...", "name": "...", "jersey_number": 10}, ...]

POST /api/athletes/import
Content-Type: multipart/form-data
Body: CSV file with athlete data
Returns: {
  "status": "success",
  "imported": 15,
  "errors": []
}
```

Settings

```
GET  /api/settings
Returns: {
  "voice_gender": "male",
  "system_volume": 60,
  "deployment_timeout": 300
}

POST /api/settings/update
Body: {
  "voice_gender": "female",
  "system_volume": 75
}
Returns: {"status": "success"}
```

Resource Management

Memory Management

Constraints: - Raspberry Pi 3 A+: 512MB RAM total - Typical usage: ~180-250MB - Flask apps: ~40MB each - REGISTRY: ~10-30MB depending on session size - Database: File-based (no RAM-resident)

Optimization Strategies: 1. Ring buffer logs (max 200 entries) 2. Lazy loading of course data 3. Efficient NodeInfo data structures 4. Connection pooling for database 5. Minimal external dependencies

Storage Management

MicroSD Card Usage: - OS + System: ~4GB - Field Trainer: ~50MB - Database: ~500KB (grows with sessions) - Logs: Rotated by systemd (max 100MB) - Audio files: ~10MB - Static assets: ~2MB

Database Growth: - Small team (20 athletes): ~1MB per season - Large team (100 athletes): ~5MB per season - Automatic vacuuming enabled

Backup Strategy: - Manual backups before major updates - Location: `/opt/backups/` - Naming: `phase*_backup_YYYYMMDD_HHMMSS/` - Database backups: `/opt/data/*.backup*`

CPU Management

Load Distribution: - Web UI: Low load, event-driven - Touch processing: Burst load on touches - Network monitoring: Minimal (60s intervals) - Audio: Burst on announcements - LEDs: PWM handled by hardware

Process Management: - Single Python process with two Flask apps (threaded) - systemd handles restart on crash - Graceful shutdown support (SIGTERM, SIGINT)

Network Bandwidth

Mesh Network: - Heartbeats: 1 per device per second (~100 bytes) - Commands: On-demand, small packets - Touch events: Immediate, ~200 bytes - Total: <10 Kbps typical

Internet (when available): - Updates: On-demand - Git: On-demand - Web UI: ~50-200 KB per page load - API calls: <5 KB typical

Logging

System Logs (journalctl)

Main Service:

```
sudo journalctl -u field-trainer-server -f       # Follow
sudo journalctl -u field-trainer-server -n 100   # Last 100 lines
sudo journalctl -u field-trainer-server --since "1 hour ago"
```

Network Manager:

```
sudo journalctl -u ft-network-manager -f
sudo journalctl -u ft-network-manager --since today
```

Log Rotation: - Managed by systemd-journald - Max size: 100MB - Retention: 7 days or until space needed

Application Logs (REGISTRY)

In-Memory Ring Buffer: - Max entries: 200 (LOG_MAX constant) - FIFO with automatic eviction - Structured format (JSON-compatible)

Log Entry Format:

```
{
  "ts": "2025-11-09T12:34:56.789Z",     # UTC timestamp
  "level": "info",                       # info, warning, error, debug
  "source": "controller",                # controller, device_N, system
  "node_id": "device_2",                 # Optional: which device
  "msg": "Touch detected on device 2"    # Human-readable message
}
```

Access: - Web UI: Admin dashboard → System Log - API: `GET /api/logs` - Direct: `REGISTRY.logs` (deque object)

Log Levels: - `debug`: Verbose operational details - `info`: Normal operations (default) - `warning`: Potential issues, degraded function - `error`: Failures requiring attention

Diagnostic Logs

Network Diagnostics:

```
sudo /opt/scripts/ft-network-manager.py status   # Network status
ip addr show                                      # All interfaces
batctl if                                         # Mesh interfaces
batctl n                                          # Mesh neighbors
sudo rfkill list                                  # Radio status
```

Service Status:

```
systemctl status field-trainer-server
systemctl status ft-network-manager
systemctl status hostapd-ft
systemctl status dnsmasq-ft
```

Database Diagnostics:

```
sqlite3 /opt/data/field_trainer.db "PRAGMA integrity_check;"
sqlite3 /opt/data/field_trainer.db ".tables"
sqlite3 /opt/data/field_trainer.db "SELECT COUNT(*) FROM runs;"
```

Deployment Configuration

Systemd Services

field-trainer-server.service **Location:** `/etc/systemd/system/field-trainer-server.service`

```
[Unit]
Description=Field Trainer System (Device 0 - Gateway)
After=network.target batman-mesh.service
Wants=network.target

[Service]
Type=simple
User=pi
Group=pi
WorkingDirectory=/opt
ExecStart=/usr/bin/python3 /opt/field_trainer_main.py --host 0.0.0.0 --port 5000 --debug 0
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
Environment="PYTHONUNBUFFERED=1"

[Install]
WantedBy=multi-user.target
```

Management:

```
sudo systemctl start field-trainer-server
sudo systemctl stop field-trainer-server
sudo systemctl restart field-trainer-server
sudo systemctl status field-trainer-server
sudo systemctl enable field-trainer-server   # Auto-start on boot
```

ft-network-manager.service **Location:** `/etc/systemd/system/ft-network-manager.service`

```
[Unit]
Description=Field Trainer Network Manager
After=network.target

[Service]
```

```
Type=simple
ExecStart=/usr/bin/python3 /opt/scripts/ft-network-manager.py
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Management:

```
sudo systemctl start ft-network-manager
sudo systemctl stop ft-network-manager
sudo systemctl restart ft-network-manager
sudo systemctl status ft-network-manager
```

Network Services (Offline Mode)

hostapd-ft.service Managed by `ft-network-manager.py`:

```
sudo systemctl start hostapd-ft    # Start AP
sudo systemctl stop hostapd-ft     # Stop AP
```

Config: `/etc/hostapd/hostapd-ft.conf`

dnsmasq-ft.service Managed by `ft-network-manager.py`:

```
sudo systemctl start dnsmasq-ft
sudo systemctl stop dnsmasq-ft
```

Config: `/etc/dnsmasq.d/ft-ap.conf`

Boot Sequence

1. Linux kernel loads
2. `network.target` reached
3. `batman-mesh.service` starts (mesh network)
4. `ft-network-manager.service` starts
   - Reads last known mode from config
   - Configures wlan1 appropriately
5. `field-trainer-server.service` starts
   - Initializes REGISTRY
   - Loads courses from database
   - Starts Flask on ports 5000 and 5001
6. System ready

Boot Time: ~45-60 seconds to full operation

Troubleshooting Guides

Problem: Service Not Starting

Symptoms: - Cannot access web interface - `systemctl status field-trainer-server` shows failed

Diagnosis:

```
sudo systemctl status field-trainer-server
sudo journalctl -u field-trainer-server -n 50
```

Common Causes: 1. **Port already in use** - Check: `sudo lsof -i :5000` and `sudo lsof -i :5001` - Fix: Kill conflicting process or reboot

2. **Database locked**
   - Check: `lsof /opt/data/field_trainer.db`
   - Fix: `sudo systemctl restart field-trainer-server`
3. **Missing dependencies**
   - Check: `python3 -c "import flask; import sqlite3"`
   - Fix: `sudo apt install python3-flask python3-sqlite`
4. **Permission issues**
   - Check: `ls -l /opt/data/field_trainer.db`
   - Fix: `sudo chown pi:pi /opt/data/field_trainer.db`

Problem: Different PIDs on Ports 5000 and 5001

Symptoms: - Health page shows different PIDs - State not synchronized between admin and coach - Different registry IDs

Cause: Two separate processes running instead of one unified service

Fix:

```
sudo systemctl restart field-trainer-server
```

This kills all port holders and starts fresh with one process hosting both ports.

Verification:

```
curl http://localhost:5000/health?format=json | jq .pid
curl http://localhost:5001/health?format=json | jq .pid
# Should be identical
```

Problem: WiFi Shows "Not Connected" After AP Mode

Symptoms: - Settings page shows "Not connected" - `ip addr show wlan1` shows no IP - `rfkill list` shows "Soft blocked: yes"

Cause: WiFi interface soft-blocked by rfkill during mode transition

Fix:

```
sudo rfkill unblock wifi
sudo systemctl restart wpa_supplicant@wlan1
sudo systemctl restart dhcpcd
```

Prevention: Network manager now automatically unblocks WiFi (as of v0.5.1)

Problem: Mesh Network Not Working

Symptoms: - Gateway shows 0 devices - `batctl n` shows no neighbors - Field devices not responding

Diagnosis:

```
batctl if                    # Should show wlan0
batctl n                     # Should show neighbors
batctl o                     # Should show originators
sudo batctl ping 192.168.99.101
```

Common Causes:

1. **Batman module not loaded**
   - Check: `lsmod | grep batman`
   - Fix: `sudo modprobe batman-adv`
2. **MAC filtering blocking devices**
   - Check: `sudo /opt/scripts/manage_mac_filter.sh status`
   - Fix: `sudo /opt/scripts/manage_mac_filter.sh disable` (temporary)
3. **WiFi channel mismatch**
   - All devices must be on same channel
   - Check mesh config on all devices
4. **Distance too far**
   - Max ~100m per hop in open space
   - Walls/obstacles reduce range significantly

Problem: Course Not Deploying

Symptoms: - Deploy button pressed, no change - Devices don't show "Deployed" LED state - REGISTRY shows "Inactive"

Diagnosis:

```
sudo journalctl -u field-trainer-server -n 100 | grep -i deploy
batctl n    # Check if devices are reachable
```

Common Causes:

1. **Devices not connected**
   - Check: Admin dashboard → Mesh Network
   - Fix: Ensure devices are powered and mesh is up
2. **Course not found**
   - Check: `sqlite3 /opt/data/field_trainer.db "SELECT course_name FROM courses;"`
   - Fix: Create course in database
3. **Deployment timeout**
   - Default: 300 seconds
   - Fix: Wait or increase timeout in settings
4. **Device MAC mismatch in course**

    - Course specifies device_ids that don't exist
    - Fix: Edit course to match actual device IDs

Problem: Touch Events Not Detected

Symptoms: - Athlete touches device, no response - No LED feedback - Touch not logged in session

Diagnosis:

```
# On field device (SSH to device_N):
sudo journalctl -u field-device -n 50 | grep -i touch
sudo i2cdetect -y 1    # Should show 0x68 (MPU6050)
```

Common Causes:

1. **MPU6050 not connected**
    - Check wiring: VCC, GND, SDA, SCL
    - Fix: Reconnect sensor
2. **I2C not enabled**
    - Check: `ls /dev/i2c-*`
    - Fix: `sudo raspi-config` → Interface Options → I2C → Enable
3. **Course not active**
    - Touches only processed in "Active" mode
    - Fix: Click "Activate" button after deploy
4. **Session not started**
    - Coach interface requires active session
    - Fix: Create session and add athletes to queue
5. **Touch attribution logic**
    - Check logs for "No active runs" or "Device not in sequence"
    - Fix: Ensure athletes are queued and course is correct

Problem: Audio Not Playing

Symptoms: - No TTS announcements - Silent speaker

Diagnosis:

```
aplay -l                        # List playback devices
speaker-test -t wav -c 2        # Test audio
mpg123 --test /opt/field_trainer/audio/test.mp3
```

Common Causes:

1. **Volume muted**
    - Check: `amixer get Master`
    - Fix: `amixer set Master 75%` or Settings page
2. **Wrong audio output**
    - Fix: `sudo raspi-config` → System Options → Audio → Select output
3. **mpg123 not installed**
    - Check: `which mpg123`
    - Fix: `sudo apt install mpg123`
4. **pico2wave not installed**

    - Check: `which pico2wave`
    - Fix: `sudo apt install libttspico-utils`
5. **Audio disabled in config**
    - Check: `/opt/field_trainer/ft_config.py` → `ENABLE_SERVER_AUDIO`
    - Should be `True`

Problem: Offline Mode Not Activating

Symptoms: - No "Field_Trainer" WiFi network visible - Manual force-offline fails

Diagnosis:

```
sudo python3 /opt/scripts/ft-network-manager.py status
sudo systemctl status hostapd-ft
sudo systemctl status dnsmasq-ft
iw list | grep -A 10 "Supported interface modes"
```

Common Causes:

1. **USB WiFi doesn't support AP mode**
    - Check: `iw list` → should show "AP" in supported modes
    - Fix: Replace with compatible adapter (Panda AC600)
2. **hostapd not installed**
    - Check: `which hostapd`
    - Fix: `sudo apt install hostapd dnsmasq`
3. **Configuration error**
    - Check: `/etc/hostapd/hostapd-ft.conf`
    - Verify interface name matches (wlan1)
4. **wpa_supplicant still running**
    - Check: `ps aux | grep wpa_supplicant`
    - Fix: `sudo systemctl stop wpa_supplicant@wlan1`

Problem: Database Corruption

Symptoms: - SQLite errors in logs - "database disk image is malformed" - Crashes on session creation

Diagnosis:

```
sqlite3 /opt/data/field_trainer.db "PRAGMA integrity_check;"
```

Fix (Restore from Backup):

```
cd /opt/data
cp field_trainer.db field_trainer.db.corrupted
cp field_trainer.db.backup_YYYYMMDD_HHMMSS field_trainer.db
sudo systemctl restart field-trainer-server
```

Fix (Dump and Restore):

```
sqlite3 /opt/data/field_trainer.db .dump > dump.sql
mv field_trainer.db field_trainer.db.corrupted
sqlite3 /opt/data/field_trainer.db < dump.sql
```

Emergency Recovery

Situation: Complete system failure, need to restore

Method 1: Network Restore (if SSH accessible)

```
ssh pi@fieldtrainer.local
/home/pi/restore-network.sh
sudo systemctl restart field-trainer-server
```

Method 2: Direct Console (HDMI + Keyboard) 1. Connect HDMI monitor and USB keyboard 2. Login: `pi` / password 3. Run: `/home/pi/restore-network.sh` 4. Follow on-screen prompts

Method 3: Full Reinstall 1. Backup database: `cp /opt/data/field_trainer.db ~` 2. Clone repo: `cd /opt && git pull origin main` 3. Restore database: `cp ~/field_trainer.db /opt/data/` 4. Restart: `sudo systemctl restart field-trainer-server`

Development vs Production

Key Differences

Aspect	Development (Pi 5)	Production (Pi 5)
Hardware	Raspberry Pi 5 8GB RAM	Raspberry Pi 5 8GB RAM
Ethernet	eth0 available	No ethernet used
Internet	eth0 + wlan1 dual-path	wlan1 only (single-path)
Failover Test	Requires unplugging eth0	Automatic on WiFi loss
Memory	Ample headroom	Ample headroom
Boot Time	~30 seconds	~30 seconds
USB Ports	4x USB-A	4x USB-A
Field Devices	Pi Zero W	Pi Zero W

Development Workflow

1. **SSH Access:**

   ```
   ssh pi@192.168.7.116     # eth0 (development)
   ssh pi@fieldtrainer.local    # mDNS
   ```

2. **Code Changes:**

   ```
   cd /opt
   git pull origin <branch>
   sudo systemctl restart field-trainer-server
   ```

3. **Testing:**

   - Admin: http://192.168.7.116:5000
   - Coach: http://192.168.7.116:5001
   - Health: http://192.168.7.116:5001/health

4. **Log Monitoring:**

   ```
   sudo journalctl -u field-trainer-server -f
   ```

5. **Database Changes:**

   ```
   # Backup first
   cp /opt/data/field_trainer.db /opt/data/field_trainer.db.backup_$(date +%Y%m%d_%H%M%S)

   # Make changes
   sqlite3 /opt/data/field_trainer.db

   # Restart to reload
   sudo systemctl restart field-trainer-server
   ```

Production Deployment

1. **Pre-deployment:**
   - Test on dev system
   - Backup production database
   - Review git commit history
2. **Deployment:**

   ```
   ssh pi@fieldtrainer.local
   cd /opt
   cp /opt/data/field_trainer.db /opt/data/field_trainer.db.backup_$(date +%Y%m%d_%H%M%S)
   git pull origin main
   sudo systemctl restart field-trainer-server
   ```

3. **Verification:**
   - Check health page
   - Verify PID matches on both ports
   - Check mesh network status
   - Test basic operations (deploy, activate)
4. **Rollback (if needed):**

   ```
   git log --oneline -5
   git checkout <previous-commit>
   sudo systemctl restart field-trainer-server
   ```

Testing Internet Failover

On Development System (Pi 5 with eth0):

```
# Unplug eth0 to simulate WiFi loss
# OR disable WiFi:
sudo ifconfig wlan1 down

# Monitor:
sudo journalctl -u ft-network-manager -f
```

```
# Should see:
# - Internet check failed (1/3)
# - Internet check failed (2/3)
# - Internet check failed (3/3)
# - Switching to AP mode
# - AP mode active
```

On Production System (Pi 3 A+): - Automatically triggers on real WiFi loss - No ethernet to interfere with testing

---

Known Limitations

Hardware Limitations

1. **Raspberry Pi Zero W (Field Devices)**
   - 512MB RAM total on field devices
   - Limited processing power compared to gateway
   - Single-core CPU
2. **USB WiFi Adapter Compatibility**
   - Not all adapters support AP mode
   - Realtek RTL8188EUS known incompatible
   - MediaTek chipsets (Panda) recommended
3. **Mesh Range**
   - ~100m per hop in open space
   - Walls/obstacles significantly reduce range
   - Max recommended: 6+ devices
4. **Pi Zero W Limitations**
   - No ethernet port
   - Single USB port (Micro-USB OTG)
   - Limited GPIO compared to full-size Pi

Software Limitations

1. **Concurrent Athletes**
   - Tested up to 10 simultaneous runners
   - More may work but not validated
   - Touch attribution complexity increases
2. **Database Size**
   - SQLite performs well up to ~100MB
   - Large historical data may need archival
   - No automatic archiving implemented
3. **Touch Attribution**
   - Assumes athletes follow course sequence
   - Skipped devices detected but may cause attribution issues
   - Backwards motion ignored
4. **Network Failover**
   - 3-minute delay after internet returns before switching back

- No hysteresis on internet loss (immediate after 3 checks)
- Cannot detect “slow” internet vs “no” internet

5. **Multi-User Support**
    - No authentication/authorization system
    - Single-coach assumption
    - No concurrent session editing protection

Operational Limitations

1. **Field Device Updates**
    - Must update each device individually
    - No over-the-air update system
    - Requires SSH access to each device
2. **Course Editing**
    - No visual course designer
    - Manual SQL or JSON editing required
    - No validation of device placement
3. **Performance Analytics**
    - Basic metrics only
    - No advanced statistical analysis
    - No comparative analytics between athletes
4. **Audio Feedback**
    - TTS quality limited by pico2wave
    - No custom audio per athlete
    - Single speaker only (no multi-device audio)
5. **LED Patterns**
    - Fixed patterns, not customizable via UI
    - Same pattern on all devices for same action
    - No per-athlete color coding

Security Limitations

1. **No Authentication**
    - Web interface has no login
    - Anyone on network can access
    - Relies on network isolation for security
2. **MAC Filtering Only**
    - Mesh security via MAC whitelist
    - Spoofing possible but unlikely in this context
    - No encryption on mesh traffic beyond WPA2
3. **No HTTPS**
    - Plain HTTP only
    - Not an issue on isolated networks
    - Would need certificates for production internet use

Future Enhancement Opportunities

High Priority

- Visual course designer
- Over-the-air device updates
- Multi-user authentication
- Advanced performance analytics
- Athlete comparison reports

Medium Priority

- Mobile app (iOS/Android)
- Video integration for form analysis
- Heart rate monitor integration
- GPS tracking for outdoor courses
- Parent portal for performance viewing

Low Priority

- Cloud sync for multi-site deployments
- Machine learning for form coaching
- Integration with third-party platforms
- Custom LED pattern designer
- Multi-language support

Licensing and Commercial Use

Current Status

- Proprietary system
- Developed for amateur athletic training
- All rights reserved

Technical Strengths for Licensing

1. **Proven Architecture**: Mesh network topology with automatic failover
2. **Comprehensive Database**: 15-table schema with full audit trail
3. **Offline Capability**: Full operation without internet connectivity
4. **Scalability**: Tested with multiple simultaneous athletes
5. **Extensibility**: Modular design allows feature additions
6. **Resource Efficiency**: Runs on low-cost Raspberry Pi hardware
7. **Production Ready**: Systemd integration, logging, error handling

Integration Points

- RESTful API for third-party integrations
- SQLite database accessible via standard tools
- Webhook support could be added

- CSV import/export for athlete data
- JSON course format for external tools

Appendix: Quick Reference

Essential Commands

```
# Service Management
sudo systemctl restart field-trainer-server
sudo systemctl status field-trainer-server
sudo journalctl -u field-trainer-server -f

# Network Management
sudo python3 /opt/scripts/ft-network-manager.py status
sudo python3 /opt/scripts/ft-network-manager.py force-online
sudo python3 /opt/scripts/ft-network-manager.py force-offline

# Mesh Network
batctl n                 # Show neighbors
batctl o                 # Show originators
batctl ping <IP>         # Mesh ping

# Database
sqlite3 /opt/data/field_trainer.db
  .tables
  .schema courses
  SELECT * FROM runs LIMIT 10;

# Emergency Restore
/home/pi/restore-network.sh
```

Important URLs

```
Admin Interface:  http://fieldtrainer.local:5000
Coach Interface:  http://fieldtrainer.local:5001
Health Check:     http://fieldtrainer.local:5001/health
Settings:         http://fieldtrainer.local:5001/settings

Offline Mode:
Connect to:       Field_Trainer (WiFi)
Password:         RaspberryField2025
Then access:      http://fieldtrainer.local:5001
Or:               http://192.168.10.1:5001
```

File Locations

```
Main Code:        /opt/coach_interface.py
                 /opt/field_trainer/
```

```
Database:         /opt/data/field_trainer.db
Config:           /opt/field_trainer/ft_config.py
                 /opt/data/network-config.json
Logs:             sudo journalctl -u field-trainer-server
Services:         /etc/systemd/system/field-trainer-server.service
Recovery:         /home/pi/restore-network.sh
```

Version History

- **0.5.1** (Nov 2025): Added offline mode, network auto-switching, local asset serving
- **0.5.0** (Nov 2025): Added team management, athlete roster, CSV import
- **0.4.0** (Oct 2025): Multi-athlete concurrent support, touch attribution
- **0.3.0** (Oct 2025): Performance history, personal records, achievements
- **0.2.0** (Sep 2025): Database integration, session management
- **0.1.0** (Aug 2025): Initial mesh network, course deployment, basic touch detection

---

Document Prepared By: Claude Code Assistant **For:** Field Trainer System Documentation
Last Updated: November 9, 2025