Alpha1 Latest Firmware Overview

Latest Version: 5.1.4
Platform: ESP32-C3 **Framework:** ESP-IDF

Overview

Alpha1 v5 is an IoT firmware solution designed primarily for the intelligent management of commercial washers and dryers in laundromats, alongside general vending and payment system integration. Built on the ESP32 platform using the ESP-IDF framework, this firmware provides comprehensive connectivity, monitoring, and control capabilities tailored for various laundromat machine types and stacked configurations.

Features

- **Multi-Environment Support**: Configurable for different machine types (BASE, MachineA, MachineB, MachineC, StackA, KioskCAA, GasSensorA)
- **Real-time Monitoring**: Current sensing, pulse detection, and status reporting
- **Secure Connectivity**: WiFi management with captive portal and MQTT communication
- **Payment Processing**: Support for coin and electronic payment systems
- **Remote Management**: OTA updates, configuration management, and remote diagnostics
- **Data Persistence**: Critical data backup and recovery with NVS storage and LittleFS file system
- **Web Interface**: Built-in web server with responsive UI (embedded HTML/JS/CSS) for device configuration
- **Auto-calibration**: Intelligent threshold calibration for optimal performance

Architecture

Core Components

Main Application (`src/main.c`)

The main application orchestrates all system components and provides:

- Environment-specific machine type selection
- Global variable management
- System initialization and task coordination
- Event handling and queue management

Configuration Management (`src/Alpha1_configurations.h`)

Centralized configuration system defining:

- Device configuration structures
- Machine type definitions
- Queue message enumerations
- Auto-threshold calibration parameters

Machine Types

1. BASE Environment

- **Purpose**: Factory default firmware flashed onto every new device before deployment
- **Features**: WiFi connectivity, OTA update capability, and cloud communication. No machine-specific logic. Once the device is registered in the Node Cloud platform, the operator selects the machine type and the correct firmware is pushed to the device over-the-air.
- **Build Flag**: `-D BASE`

2. MachineA / MachineB / MachineC Environments (`src/Machines/MachineX.h/.c`)

- **Build Flags**: `-D MACHINE_A`, `-D MACHINE_B`, `-D MACHINE_C`
- **MachineA** — Standard coin-op washer or dryer. Uses CT current sensing to detect running/idle cycles and a timer-based pulse reader to count coin drops. Lock data records both the start and end of each cycle. Supports local kiosk MQTT and auto-threshold calibration.
- **MachineB** — Detergent dispensers, water vending machines, and similar coin/bill-operated product dispensing equipment. No current sensing or cycle tracking. Counts bill/token pulses using a timer-based reader. Transaction completion is determined by an inactivity timeout (lock-check counter) rather than a CT sensor transition.
- **MachineC** — Commercial washer or dryer using a hardware rising-edge GPIO interrupt for coin pulse detection instead of a timer. CT current sensing and lock data work the same as MachineA. Includes a noise filter on the pulse reader during active cycles and supports local kiosk MQTT.

3. StackA Environment (`src/Machines/StackA.h/.c`)

- **Purpose**: Advanced stacked machine system (shared motherboard)
- **Features**:
  - Multiple payment methods (coin, epay, cash advance)
  - Enhanced current monitoring
  - Extended payment processing capabilities
  - Advanced status reporting
- **Build Flag**: `-D STACK_A`

4. KioskCAA Environment (`src/Machines/KioskCAA.h/.c`)

- **Purpose**: Centralized kiosk management integration
- **Features**: Kiosk-based payment routing and control
- **Build Flag**: `-D KIOSKCA_A`

5. GasSensorA Environment (`src/Sensors/GasSensorA.h/.c`)

- **Purpose**: Dryer cycle monitoring via gas tong pulse detection
- **Features**: Connects to the gas tong (burner assembly) signal line of gas-powered dryers. Counts pulses each time the burner fires to track dryer cycle runs. Tracks both a current pulse count and a lifetime cumulative total, reported periodically to the cloud. No payment processing, no current sensing.
- **Build Flag**: `-D GASSENSOR_A`

Managed Components

Common Core Component

- **`alpha_common`**: Shared operational logic, unified commands, UI templates, and network abstractions utilized across all machine types.

Connectivity Components

- **`wifi_app`**: WiFi connection management and captive portal
- **`mqtt_app`**: MQTT client for cloud communication
- **`wm_app`**: WiFi Manager for network configuration
- **`sntp_app`**: Network time synchronization

Data Management Components

- **`nvs_app`**: Non-volatile storage management
- **`criticalData_app`**: Critical data backup and recovery
- **`ota_app`**: Over-the-air firmware updates

Hardware Interface Components

- **`pulse_app`**: Pulse detection and counting

- `ct_app`: Current transformer interface
- `ui_app`: User interface and LED management

File Structure

```
src/
├── main.c                      # Main application entry point
├── Alpha1_configurations.h     # Global configuration definitions
├── Machines/
│   ├── MachineA.h/.c          # MachineA implementation
│   └── StackA.h/.c            # StackA implementation
├── Sensors/
│   └── GasSensorA.h/.c        # Gas sensor interface
└── webpage/
    ├── index.html             # Web interface HTML
    ├── code.js               # Client-side JavaScript
    └── style.css             # Styling
```

Build Environments

Prerequisites

- PlatformIO
- ESP-IDF framework
- ESP32-C3 development board

Environment Configuration

BASE Environment

```
pio run -e BASE
```

- Minimal feature set
- Basic monitoring capabilities

Machine Environments (MachineA, MachineB, MachineC)

```
pio run -e MachineA
# or MachineB, MachineC
```

- MachineA/C: CT current sensing, coin pulse detection, cycle-based lock data
- MachineB: Bill/token pulse counting for detergent vending, water vending, and similar dispensing machines, inactivity-based transaction completion

StackA Environment

```
pio run -e StackA
```

- Dual-payment system
- Support for coin, epay, and cash advance

Specialized Environments

```
pio run -e KioskCAA
pio run -e GasSensorA
```

- KioskCAA: Central payment routing to machines on the floor
- GasSensorA: Dryer cycle counting via gas tong pulse detection

Build Configuration

The firmware uses environment-specific build flags defined in `platformio.ini`:

```
[env:BASE]
build_flags = ... -D BASE

[env:MachineA]
build_flags = ... -D MACHINE_A

[env:StackA]
build_flags = ... -D STACK_A
```

Custom Firmware Extraction

A custom Python script (`copyFirmware.py`) is embedded in the build process via `extra_scripts`. It copies the resulting binaries into release folders automatically.

Partition Setup

The project defines custom board partitions using `alphaX_partitions.csv`, efficiently structuring the ESP32-C3 flash for Application, NVS, LittleFS, and OTA logic.

Hardware Requirements

- **MCU**: ESP32-C3 DevKit M-1
- **Flash**: Minimum 4MB
- **RAM**: 400KB SRAM
- **Connectivity**: WiFi 802.11 b/g/n
- **ADC**: For current sensing
- **GPIO**: For pulse detection and LED indicators

Pin Configuration

- Current sensing via ADC
- Pulse input detection
- Status LEDs (WiFi, MQTT, Config)
- Configuration and reset buttons

Installation

1. **Clone the repository:**

   ```
   git clone https://github.com/Antlysis/Alpha1-v5.git
   cd Alpha1-v5
   ```

2. **Install dependencies:**

   ```
   pio pkg install
   ```

3. **Build for specific environment:**

   ```
   # For MachineA
   pio run -e MachineA

   # For StackA
   pio run -e StackA

   # For BASE
   pio run -e BASE
   ```

4. **Upload firmware:**

```
pio run -e MachineA -t upload
```

Configuration

Initial Setup

1. Power on the device
2. Connect to the WiFi access point `Laundro:XX:XX:XX`
3. Navigate to `192.168.4.1` in a web browser if configuration portal does not open automatically
4. Configure WiFi credentials and outlet parameters

MQTT Configuration

- **Broker**: Configurable via web interface
- **Topics**: Auto-generated based on device MAC address
- **QoS**: Configurable per message type
- **Last Will**: Automatic disconnect detection

Device Parameters

- **Outlet ID**: Unique identifier for the machine
- **Device IDs**: Primary and secondary device identifiers
- **Pulse Configuration**: Width and interval settings
- **Threshold Values**: Current detection thresholds

API Reference

Machine Interface Functions

MachineA/StackA Common Functions

```
void machine_begin(Alpha1_t *conf);           // Initialize machine
void machine_status(char *status, criticalData_t *lockData, char *time);  // Get status
int machine_pay(char* payload);              // Process payment
```

```
bool machine_command(char *payload, int length, char *response);  // Execute command
void machine_updateConfig(void);             // Update configuration
```

MQTT Topics Structure

```
<modelID>/<outletID>/<deviceID>/status    # Status updates
<modelID>/<outletID>/<deviceID>/config    # Configuration commands
<modelID>/<outletID>/<deviceID>/payment   # Payment notifications
<modelID>/<outletID>/<deviceID>/debug     # Debug messages
```

Supported Chipsets

Supported Chipset	Notes
ESP32-C3	Tested & Fully supported

Troubleshooting

Debug Information

- **Serial Monitor**: 115200 baud rate
- **Log Levels**: Configurable per component
- **Remote Logging**: Available via MQTT debug topic

Testing

Detailed test cases and procedures are maintained in the repository:

- **TESTCASE.md**: Outlines comprehensive unit, integration, and systemic test scenarios across components like OTA, MQTT, and NVS.
- **TESTCASE_MACHINES.md**: Environment-specific validation checklists ensuring successful operational flows for designated machine targets.

Version History

See CHANGELOG.md for detailed version history and deployment information.

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly on target hardware
5. Submit a pull request

License

Support

For technical support and issues:

- Create an issue in the repository
- Contact the development team
- Review documentation and troubleshooting guides

For deeper understanding of the implementation, refer to the comprehensive comments and documentation within the program files in the `src/` and `managed_components/` directories.

Revision #9
Created 15 May 2026 06:42:09 by Rahul
Updated 15 May 2026 10:09:04 by Rahul