**Complete Deployment Guide**

**Table of Contents**

1. [Prerequisites](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#prerequisites)
2. [Initial Setup](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#initial-setup)
3. [Flashing Instructions](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#flashing-instructions)
4. [API Configuration](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#api-configuration)
5. [Build Instructions](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#build-instructions)
6. [Troubleshooting](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#troubleshooting)
7. [Advanced Configuration](https://claude.ai/chat/1bfc3d37-3020-4c47-990d-c201155661f1#advanced-configuration)

**Prerequisites**

**Hardware Requirements**

* **Microcontroller**: ESP32/ESP8266 or compatible board
* **Memory**: Minimum 4MB flash storage
* **Power**: 3.3V stable power supply
* **Connections**: USB-to-serial adapter (if not built-in)

**Software Dependencies**

* **Python 3.8+** with pip
* **Node.js 16+** and npm
* **Git** for version control
* **Serial terminal** (PuTTY, screen, or Arduino IDE Serial Monitor)

**Development Tools**

# Install esptool for flashing

pip install esptool

# Install platformio (recommended)

pip install platformio

# Alternative: Arduino IDE with ESP32 board package

**Initial Setup**

**1. Environment Preparation**

# Clone the repository

git clone https://github.com/yourusername/your-project.git

cd your-project

# Create virtual environment

python -m venv venv

source venv/bin/activate # Linux/Mac

# or

venv\Scripts\activate # Windows

# Install dependencies

pip install -r requirements.txt

**2. Configuration Files**

Create your local configuration:

cp config/config.example.json config/config.json

cp .env.example .env

Edit config/config.json:

{

"wifi": {

"ssid": "YOUR\_WIFI\_NETWORK",

"password": "YOUR\_WIFI\_PASSWORD"

},

"api": {

"endpoint": "https://your-api-endpoint.com",

"key": "your-api-key-here"

},

"device": {

"name": "MyDevice-001",

"location": "Office"

}

}

**Flashing Instructions**

**Method 1: Using esptool (Recommended)**

# Put device in flash mode (hold BOOT button while connecting)

# Find your serial port

esptool.py --port COM3 flash\_id # Windows

esptool.py --port /dev/ttyUSB0 flash\_id # Linux

# Erase flash (optional but recommended)

esptool.py --port [YOUR\_PORT] erase\_flash

# Flash the firmware

esptool.py --port [YOUR\_PORT] --baud 460800 write\_flash \

--flash\_size=detect 0 firmware.bin

**Method 2: Using PlatformIO**

# Build and upload

pio run --target upload

# Monitor serial output

pio device monitor --port [YOUR\_PORT] --baud 115200

**Method 3: Using Arduino IDE**

1. Open the .ino file in Arduino IDE
2. Select your board: **Tools > Board > ESP32 Dev Module**
3. Select your port: **Tools > Port > [YOUR\_PORT]**
4. Click **Upload** button

**Verification**

After flashing, you should see output similar to:

[INFO] Device starting...

[INFO] Connecting to WiFi: YOUR\_NETWORK

[INFO] WiFi connected! IP: 192.168.1.100

[INFO] API connection established

[INFO] Device ready for operation

**API Configuration**

**Setting Up API Endpoints**

**1. Authentication**

The device uses API key authentication. Set your key in the configuration:

{

"api": {

"key": "sk-your-secret-key-here",

"timeout": 30000

}

}

**2. Endpoint Configuration**

Configure your API endpoints:

{

"endpoints": {

"status": "/api/v1/device/status",

"data": "/api/v1/device/data",

"config": "/api/v1/device/config",

"firmware": "/api/v1/device/firmware"

}

}

**3. Testing API Connection**

# Test endpoint accessibility

curl -H "Authorization: Bearer YOUR\_API\_KEY" \

https://your-api-endpoint.com/api/v1/device/status

# Expected response:

{

"status": "ok",

"version": "1.0.0",

"timestamp": "2025-06-29T10:30:00Z"

}

**API Rate Limits**

* **Status updates**: 1 request per minute
* **Data transmission**: 10 requests per minute
* **Configuration changes**: 5 requests per hour

**Build Instructions**

**Frontend Build (if applicable)**

cd web-interface

npm install

npm run build

# For development

npm run dev

**Firmware Build from Source**

# Using PlatformIO

pio run

# Using Arduino CLI

arduino-cli compile --fqbn esp32:esp32:esp32 src/

# Custom build with optimizations

make BOARD=esp32 OPTIMIZE=size build

**Creating Custom Firmware**

1. **Modify source code** in src/ directory
2. **Update version** in version.h
3. **Test locally** with hardware-in-loop
4. **Build release** version:

pio run --environment release

**Build Outputs**

After successful build, you'll find:

* firmware.bin - Main firmware file
* bootloader.bin - Bootloader (if separate)
* partitions.bin - Partition table
* manifest.json - Build metadata

**Troubleshooting**

**Common Issues**

**Flash Errors**

**Problem**: Failed to connect to ESP32 **Solution**:

* Check USB cable and connections
* Hold BOOT button while connecting
* Try different baud rates: 115200, 460800, 921600

**WiFi Connection Issues**

**Problem**: Device can't connect to WiFi **Solution**:

* Verify SSID and password in config
* Check WiFi signal strength
* Try 2.4GHz network (not 5GHz)

**API Authentication Failures**

**Problem**: 401 Unauthorized errors **Solution**:

* Verify API key is correct
* Check key hasn't expired
* Ensure proper header format

**Debug Mode**

Enable verbose logging:

#define DEBUG\_LEVEL 3 // 0=none, 1=error, 2=info, 3=debug

**Serial Monitor Commands**

While connected via serial:

* status - Show device status
* restart - Restart device
* config - Show current configuration
* factory - Factory reset

**Advanced Configuration**

**Over-the-Air (OTA) Updates**

Enable OTA in configuration:

{

"ota": {

"enabled": true,

"port": 3232,

"password": "your-ota-password"

}

}

Upload via OTA:

pio run --target upload --upload-port 192.168.1.100

**Custom Certificates**

For HTTPS API endpoints, add your certificate:

const char\* rootCACertificate = \

"-----BEGIN CERTIFICATE-----\n" \

"YOUR\_CERTIFICATE\_HERE\n" \

"-----END CERTIFICATE-----\n";

**Performance Tuning**

Optimize for your use case:

{

"performance": {

"cpu\_frequency": 240,

"sleep\_mode": "light",

"watchdog\_timeout": 30

}

}

**Monitoring and Logging**

Set up remote logging:

{

"logging": {

"level": "info",

"remote\_endpoint": "https://logs.yourserver.com",

"local\_buffer\_size": 1024

}

}

**Support and Maintenance**

**Regular Maintenance**

* **Weekly**: Check device status and logs
* **Monthly**: Update firmware if available
* **Quarterly**: Review and rotate API keys

**Getting Help**

* **Documentation**: Check the wiki for detailed guides
* **Issues**: Report bugs on GitHub issues page
* **Community**: Join our Discord server for real-time help

**Version History**

* **v1.2.0** - Added OTA support, improved stability
* **v1.1.0** - Enhanced API integration, bug fixes
* **v1.0.0** - Initial release