ESP8266 MQTT Relay Controller

Complete Documentation & Windows Automation Scripts

Version 3.2

IoT Device Control with WiFi Management

MQTT Communication & OTA Updates

Windows Batch Script Automation

Generated: 31/8/2025 8:15:36 am

Table of Contents

1. Overview

This firmware provides a complete IoT relay control solution for ESP8266 microcontrollers with advanced features including WiFi management, MQTT communication, Over-The-Air (OTA) updates, and comprehensive device monitoring.

Key Features

- **WiFi Management**: Auto-configuration portal using WiFiManager
- **MQTT Communication**: Secure, reliable MQTT messaging with JSON payloads
- **OTA Updates**: Remote firmware updates via HTTPS/HTTP
- **Device Monitoring**: Real-time status reporting and heartbeat monitoring
- **Manual Control**: Physical button with multiple functions
- **Safety Features**: Emergency reset, configuration reset, and OTA safety locks
- **Comprehensive Logging**: Detailed serial output for debugging

2. Hardware Requirements

Pinout Configuration

GPIO Pin	Function	Description
D1 (GPIO5)	RELAY_PIN	Controls the relay (active HIGH)
D3 (GPIO0)	BUTTON_PIN	Manual control button (pull-up)
GPIO2	LED_PIN	Built-in LED (inverted logic)

Required Components

- ESP8266 development board (NodeMCU, Wemos D1, etc.)
- 5V/3.3V relay module
- Push button (optional - for manual control)
- Power supply (5V recommended for relay operation)

3. Software Dependencies

Arduino Libraries

```
#include <ESP8266WiFi.h>       // Core WiFi functionality
#include <PubSubClient.h>      // MQTT communication
#include <ArduinoJson.h>       // JSON parsing and generation
#include <WiFiManager.h>       // WiFi configuration portal
#include <EEPROM.h>            // Configuration storage
#include <ESP8266httpUpdate.h> // OTA update functionality
#include <WiFiClientSecure.h> // HTTPS support
#include <ESP8266HTTPClient.h> // HTTP client
```

4. Configuration

Default Settings

- **MQTT Broker**: broker.emqx.io
- **MQTT Port**: 1883
- **WiFi AP Name**: ESP8266_Relay_[ChipID]
- **WiFi AP Password**: relay123
- **Configuration Timeout**: 5 minutes
- **Heartbeat Interval**: 60 seconds
- **OTA Timeout**: 5 minutes

MQTT Topics Structure

All topics are dynamically generated based on device ChipID:

Base Pattern: home/relay/[DEVICE_ID]/[TOPIC]

Topics:
• home/relay/[DEVICE_ID]/command - Receive commands
• home/relay/[DEVICE_ID]/status - Publish status updates
• home/relay/[DEVICE_ID]/heartbeat - Periodic health check
• home/relay/[DEVICE_ID]/connection - Connection events
• home/relay/[DEVICE_ID]/ota - OTA update commands/status

5. Setup and Installation

Initial Setup

1. Flash the firmware to your ESP8266
2. Power on the device
3. Connect to WiFi network ESP8266_Relay_[ChipID] with password "relay123"
4. Navigate to http://192.168.4.1
5. Configure your WiFi credentials and MQTT settings
6. Save configuration and restart

MQTT Configuration

The web portal allows configuration of:

- **MQTT Server**: Broker hostname/IP
- **MQTT Port**: Broker port (default: 1883)
- **Device Name**: Human-readable device identifier

6. Command Reference (Windows)

Note: Replace [DEVICE_ID] with your actual device ID (e.g., a1b2c3)

Basic Relay Commands

Turn Relay ON

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "on" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "1" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "true"
```

Turn Relay OFF

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "off" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "0" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "false"
```

Toggle Relay State

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "toggle"
```

Request Current Status

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "status"
```

OTA Update Commands

Check Current Version

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/ota" -m "{\"command\": \"check_version\"}"
```

Perform Firmware Update

```
mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/ota" -m "{\"command\": \"update\", \"url\": \"https://example.com/firmware.bin\", \"version\": \"3.3\"}"
```

Monitoring Commands

Monitor All Device Topics

```
mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/+"
```

Monitor Status Updates Only

```
mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/status"
```

7. Windows Batch Scripts

Basic Control Script (relay_control.bat)

```
@echo off
setlocal enabledelayedexpansion

:: Configuration
set DEVICE_ID=a1b2c3
set BROKER=broker.emqx.io
set MQTT_PUB=mosquitto_pub.exe

echo ESP8266 Relay Controller
echo Device ID: %DEVICE_ID%
echo Broker: %BROKER%
echo.

:menu
echo Choose an option:
echo 1. Turn Relay ON
echo 2. Turn Relay OFF
echo 3. Toggle Relay
echo 4. Get Status
echo 5. Exit
echo.
set /p choice=Enter choice (1-5):

if "%choice%"=="1" (
    echo Turning relay ON...
    %MQTT_PUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "on"
    echo Command sent.
    goto menu
)
if "%choice%"=="2" (
    echo Turning relay OFF...
    %MQTT_PUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "off"
    echo Command sent.
    goto menu
)
if "%choice%"=="3" (
    echo Toggling relay...
    %MQTT_PUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "toggle"
    echo Command sent.
    goto menu
)
if "%choice%"=="4" (
    echo Getting status...
    %MQTT_PUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "status"
    echo Command sent.
    goto menu
)
if "%choice%"=="5" (
    echo Goodbye!
    exit /b 0
)

echo Invalid choice. Please try again.
goto menu
```

Device Monitor Script (device_monitor.bat)

```
@echo off
setlocal enabledelayedexpansion

:: Configuration
set DEVICE_ID=a1b2c3
set BROKER=broker.emqx.io
set MQTT_SUB=mosquitto_sub.exe

echo ESP8266 Device Monitor
echo Device ID: %DEVICE_ID%
echo Broker: %BROKER%
echo.

:menu
echo Choose monitoring option:
echo 1. Monitor All Topics
echo 2. Monitor Status Only
echo 3. Monitor Heartbeat
echo 4. Monitor with Log File
echo 5. Exit
echo.
set /p choice=Enter choice (1-5):

if "%choice%"=="1" (
    echo Monitoring all topics (Press Ctrl+C to stop)
    %MQTT_SUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/+" -v
    goto menu
)
if "%choice%"=="2" (
    echo Monitoring status updates (Press Ctrl+C to stop)
    %MQTT_SUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/status" -v
    goto menu
)

echo Invalid choice. Please try again.
goto menu
```

8. Status Messages

Device Status (Published to /status)


```
{ "device_id": "a1b2c3", "device_name": "ESP8266_a1b2c3", "relay": "ON", "uptime": 12345, "free_heap": 32768, "rssi": -45, "ip": "192.168.1.100", "ssid": "MyWiFi", "firmware_version": "3.2", "ota_in_progress": false, "timestamp": 67890 }
```


OTA Status Values

Status	Description
OTA_STARTING	Update process initiated
OTA_PROGRESS	Update in progress (includes progress %)
OTA_COMPLETE	Update completed successfully
OTA_FAILED	Update failed
OTA_TIMEOUT	Update timed out
OTA_CANCELLED	Update was cancelled

9. Physical Controls

Button Functions

The physical button provides multiple functions based on press duration:

- **Short Press** (< 300ms): Toggle relay state
- **Long Press** (5+ seconds): Emergency restart
- **Extended Press** (10+ seconds): Reset WiFi configuration

LED Indicators

The built-in LED provides visual feedback:

- **Solid ON**: Relay is OFF, WiFi connected
- **Solid OFF**: Relay is ON, WiFi connected
- **Fast Blink**: Configuration mode active
- **Slow Blink**: OTA update in progress
- **Continuous Blink**: WiFi/MQTT connection issues

10. Safety Features

OTA Safety

- Relay is automatically disabled during OTA updates
- Manual relay control is blocked during OTA
- Low memory detection cancels OTA process
- 5-minute timeout prevents stuck updates
- Progress monitoring with detailed error reporting

Memory Management

- Continuous heap memory monitoring
- Low memory warnings (< 10KB free)
- Automatic OTA cancellation on low memory
- Regular memory usage reporting

Network Resilience

- Automatic WiFi reconnection
- MQTT connection monitoring with auto-reconnect
- Configurable retry limits and timeouts
- Graceful degradation during network issues

11. Troubleshooting

Common Issues

Device Not Connecting to WiFi

1. Check if device is in configuration mode (LED blinking fast)
2. Connect to ESP8266_Relay_[ChipID] network
3. Navigate to http://192.168.4.1
4. Verify WiFi credentials and save configuration
5. Hold button for 10+ seconds to reset if needed

MQTT Connection Issues

```
:: Test MQTT broker connectivity mosquitto_pub.exe -h broker.emqx.io -t "test/topic" -m "test message" :: Check if device is publishing heartbeat timeout 65 mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/heartbeat" -C 1 :: Test command reception mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "status"
```

Memory Issues

```
:: Monitor device status for memory information mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/status" | findstr "free_heap" :: Check for error messages mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/status/error"
```

12. Advanced Configuration

Custom MQTT Broker Commands

```
:: Test custom broker connectivity mosquitto_pub.exe -h your-broker.com -p 1883 -t "test" -m "connection test" :: Monitor with custom broker mosquitto_sub.exe -h your-broker.com -p 1883 -t "home/relay/[DEVICE_ID]/+"
```

Secure MQTT (with authentication)

```
:: Connect with username/password mosquitto_pub.exe -h broker.emqx.io -u username -P password -t "home/relay/[DEVICE_ID]/command" -m "status" :: Monitor with authentication mosquitto_sub.exe -h broker.emqx.io -u username -P password -t "home/relay/[DEVICE_ID]/+"
```

13. API Integration Examples

Home Assistant Integration

```
# configuration.yaml switch: - platform: mqtt name: "ESP8266 Relay" command_topic: "home/relay/a1b2c3/command" state_topic: "home/relay/a1b2c3/status" value_template: "{{ value_json.relay }}" payload_on: "on" payload_off: "off" sensor: - platform: mqtt name: "ESP8266 Uptime" state_topic: "home/relay/a1b2c3/heartbeat" value_template: "{{ value_json.uptime }}" unit_of_measurement: "seconds"
```

PowerShell Integration

```
# ESP8266 PowerShell Control Script
param(
    [string]$DeviceId = "a1b2c3",
    [string]$Broker = "broker.emqx.io",
    [string]$Command = "status"
)

$MQTTPub = "mosquitto_pub.exe"

function Send-RelayCommand {
    param([string]$cmd)
    $topic = "home/relay/$DeviceId/command"
    & $MQTTPub -h $Broker -t $topic -m $cmd
    Write-Host "Command '$cmd' sent to device $DeviceId"
}

# Main execution
switch ($Command) {
    "on" { Send-RelayCommand "on" }
    "off" { Send-RelayCommand "off" }
    "toggle" { Send-RelayCommand "toggle" }
    "status" { Send-RelayCommand "status" }
    default { Write-Host "Invalid command" }
}
```

14. Maintenance Scripts

Health Check Script (health_check.bat)

```
@echo off
setlocal enabledelayedexpansion

:: Configuration
set DEVICE_ID=a1b2c3
set BROKER=broker.emqx.io
set MQTT_PUB=mosquitto_pub.exe
set MQTT_SUB=mosquitto_sub.exe

echo ESP8266 Health Check Utility
echo Device ID: %DEVICE_ID%
echo Broker: %BROKER%
echo Current Time: %date% %time%
echo.

echo === HEALTH CHECK STARTING ===
echo.

echo 1. Testing MQTT Broker Connection...
%MQTT_PUB% -h %BROKER% -t "test/health_check" -m "test" > nul 2>&1
if %errorlevel% equ 0 (
    echo    ✓ MQTT Broker Connection: OK
) else (
    echo    ✗ MQTT Broker Connection: FAILED
    goto end
)

echo 2. Testing Device Responsiveness...
%MQTT_PUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "status"
echo    Waiting for device response...
timeout 10 %MQTT_SUB% -h %BROKER% -t "home/relay/%DEVICE_ID%/status" -C 1 > temp_status.txt 2>&1
if exist temp_status.txt (
    echo    ✓ Device Response: OK
    del temp_status.txt
) else (
    echo    ✗ Device Response: TIMEOUT
)

:end
echo.
echo === HEALTH CHECK COMPLETED ===
echo Report generated at: %date% %time%
pause
```

15. Quick Reference

Essential Windows Commands

Basic Control:

```
:: Turn relay ON/OFF/Toggle mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "on" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "off" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "toggle"
```

Status & Monitoring:

```
:: Get status and monitor mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/command" -m "status" mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/+" mosquitto_sub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/heartbeat"
```

OTA Updates:

```
:: Check version and update mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/ota" -m "{\"command\": \"check_version\"}" mosquitto_pub.exe -h broker.emqx.io -t "home/relay/[DEVICE_ID]/ota" -m "{\"command\": \"update\", \"url\": \"https://example.com/firmware.bin\", \"version\": \"3.3\"}"
```

Batch Script Templates

Quick Control Template

```
@echo off
set DEVICE_ID=your_device_id
set BROKER=broker.emqx.io
mosquitto_pub.exe -h %BROKER% -t "home/relay/%DEVICE_ID%/command" -m "%1"

REM Usage: control.bat on|off|toggle|status
```

Installation Notes

Windows Setup:

1. Download and install Mosquitto MQTT clients for Windows
2. Add Mosquitto installation directory to system PATH
3. Verify installation: `mosquitto_pub.exe --help`
4. Copy provided batch scripts to working directory
5. Edit device IDs and broker settings in scripts

Script Configuration

All batch scripts require these variables:

- **DEVICE_ID** - ESP8266 device ID (chip ID in hex)
- **BROKER** - MQTT broker hostname or IP
- **MQTT_PUB / MQTT_SUB** - Paths to mosquitto executables

Common Pin Assignments

Component	ESP8266 Pin	GPIO	Function
Relay Control	D1	GPIO5	Digital Output (HIGH = ON)
Manual Button	D3	GPIO0	Digital Input (Pull-up)
Status LED	Built-in	GPIO2	Digital Output (Inverted)

MQTT Topic Summary

Topic	Direction	Purpose
home/relay/[ID]/command	Subscribe	Receive control commands
home/relay/[ID]/status	Publish	Device status updates
home/relay/[ID]/heartbeat	Publish	Periodic health check (60s)
home/relay/[ID]/ota	Both	OTA update commands/status
home/relay/[ID]/connection	Publish	Connection events

Troubleshooting Checklist

Device Issues:

- ☐ Check power supply (5V recommended)
- ☐ Verify WiFi credentials in config portal
- ☐ Confirm MQTT broker accessibility
- ☐ Monitor serial output for errors
- ☐ Check device ID matches topic structure

Communication Issues:

- ☐ Test MQTT broker with mosquitto tools
- ☐ Verify topic names and device ID
- ☐ Check network connectivity
- ☐ Monitor heartbeat messages
- ☐ Validate JSON command format

OTA Update Issues:

- ☐ Ensure firmware URL is accessible
- ☐ Check available memory (>10KB)
- ☐ Verify firmware compatibility
- ☐ Monitor OTA status messages
- ☐ Cancel if memory issues occur

Version History

Version	Date	Changes
3.2	Current	Enhanced OTA, improved error handling, memory monitoring, enhanced MQTT structure
3.1	Previous	Added WiFiManager integration, improved button handling
3.0	Previous	Initial release with basic MQTT and OTA support

Appendix A: Complete Batch Script Collection

Multi-Device Manager (multi_device.bat)

```
@echo off
setlocal enabledelayedexpansion

:: Configuration - Add your device IDs here
set DEVICE_COUNT=3
set DEVICE_1=a1b2c3
set DEVICE_2=b2c3d4
set DEVICE_3=c3d4e5
set BROKER=broker.emqx.io
set MQTT_PUB=mosquitto_pub.exe

echo Multiple ESP8266 Device Manager
echo Broker: %BROKER%
echo Configured Devices: %DEVICE_COUNT%
for /l %%i in (1,1,%DEVICE_COUNT%) do (
    echo   %%i. !DEVICE_%%i!
)
echo.

:menu
echo Choose an option:
echo 1. Turn All Relays ON
echo 2. Turn All Relays OFF
echo 3. Toggle All Relays
echo 4. Get All Status
echo 5. Control Single Device
echo 6. Exit
echo.
set /p choice=Enter choice (1-6):

if "%choice%"=="1" (
    echo Turning all relays ON...
    for /l %%i in (1,1,%DEVICE_COUNT%) do (
        echo Sending ON command to !DEVICE_%%i!
        %MQTT_PUB% -h %BROKER% -t "home/relay/!DEVICE_%%i!/command" -m "on"
    )
    echo All commands sent.
    goto menu
)

if "%choice%"=="6" (
    echo Goodbye!
    exit /b 0
)

echo Invalid choice. Please try again.
goto menu
```

Scheduled Tasks Manager (scheduled_tasks.bat)

```
@echo off
setlocal enabledelayedexpansion

:: Configuration
set DEVICE_ID=a1b2c3
set BROKER=broker.emqx.io
set MQTT_PUB=mosquitto_pub.exe

echo ESP8266 Scheduled Tasks Manager
echo Device ID: %DEVICE_ID%
echo.

:menu
echo Choose schedule type:
echo 1. Daily ON/OFF Schedule
echo 2. Weekly Schedule
echo 3. Interval Toggle
echo 4. View Current Time
echo 5. Exit
echo.
set /p choice=Enter choice (1-5):

if "%choice%"=="1" (
    echo Daily Schedule Setup
    set /p on_hour=Enter ON hour (24-hour format):
    set /p on_minute=Enter ON minute:
    set /p off_hour=Enter OFF hour (24-hour format):
    set /p off_minute=Enter OFF minute:

    echo Creating daily schedule
    echo ON at %on_hour%:%on_minute%
    echo OFF at %off_hour%:%off_minute%

    :: Create scheduled task for ON
    schtasks /create /tn "ESP8266_Relay_ON_%DEVICE_ID%" /tr "mosquitto_pub.exe -h %BROKER% -t
home/relay/%DEVICE_ID%/command -m on" /sc daily /st %on_hour%:%on_minute% /f

    :: Create scheduled task for OFF
    schtasks /create /tn "ESP8266_Relay_OFF_%DEVICE_ID%" /tr "mosquitto_pub.exe -h %BROKER% -t
home/relay/%DEVICE_ID%/command -m off" /sc daily /st %off_hour%:%off_minute% /f

    echo Daily schedule created successfully!
    goto menu
)

if "%choice%"=="5" (
    echo Goodbye!
    exit /b 0
)

echo Invalid choice. Please try again.
goto menu
```

Appendix B: JSON Message Examples

Command Messages (Send to /command topic)

```
"on" // Turn relay ON "off" // Turn relay OFF "toggle" // Toggle relay state "status" // Request status "info" // Request device info "restart" // Restart device "reset_wifi" // Reset WiFi config
```

OTA Command Messages (Send to /ota topic)

```
// Check version { "command": "check_version" } // Update firmware { "command": "update", "url": "https://example.com/firmware.bin", "version": "3.3" } // Force update { "command": "force_update", "url": "https://example.com/firmware.bin", "version": "3.3" } // Cancel update { "command": "cancel" }
```

Status Response Messages (Published to /status topic)

```
{ "device_id": "a1b2c3", "device_name": "ESP8266_a1b2c3", "relay": "ON", "uptime": 12345, "free_heap": 32768, "rssi": -45, "ip": "192.168.1.100", "ssid": "MyWiFi", "firmware_version": "3.2", "ota_in_progress": false, "timestamp": 67890 }
```

Device Info Response (Published to /status/info topic)

```
{ "device_id": "a1b2c3", "device_name": "ESP8266_a1b2c3", "chip_id": "A1B2C3", "flash_size": 4194304, "free_heap": 32768, "boot_time": 1234, "uptime": 12345, "wifi_ssid": "MyWiFi", "ip": "192.168.1.100", "mac": "AA:BB:CC:DD:EE:FF", "rssi": -45, "mqtt_server": "broker.emqx.io", "mqtt_port": 1883, "relay_state": "ON", "total_commands": 42, "total_reconnects": 3, "total_ota_updates": 2, "firmware_version": "3.2", "ota_capable": true, "ota_in_progress": false }
```

Appendix C: Wiring Diagrams

Basic Wiring Setup

ESP8266 NodeMCU to Relay Module:

- ESP8266 3.3V → Relay VCC (if 3.3V relay) or use external 5V supply
- ESP8266 GND → Relay GND
- ESP8266 D1 (GPIO5) → Relay IN

Optional Push Button:

- Button Pin 1 → ESP8266 D3 (GPIO0)
- Button Pin 2 → ESP8266 GND
- Internal pull-up resistor is used

Power Supply:

- ESP8266: 3.3V (USB or external regulator)
- Relay: 5V recommended for reliable operation
- Load: According to relay specifications

Safety Considerations

Important Safety Notes:

- ⚠️ Always disconnect power when making connections
- ⚠️ Use appropriate relay ratings for your load
- ⚠️ Ensure proper isolation for AC loads
- ⚠️ Use fuses appropriate for your application
- ⚠️ Follow local electrical codes and regulations
- ⚠️ Consider using optocouplers for additional isolation

End of Documentation

Support Information

Documentation Version: 3.2

Firmware Version: 3.2

Generated: 31/8/2025 8:15:36 am

This documentation provides complete coverage of the ESP8266 MQTT Relay Controller with Windows automation scripts. For the latest updates and additional resources, please refer to the project repository.

Disclaimer: This firmware and documentation are provided as-is for educational and development purposes. Users are responsible for proper implementation, testing, and compliance with local regulations for IoT devices.