PWM Fan Control Logic

Apex TV Enclosures | Firmware Specification | Version 2.0

1. Control System Overview

The Apex thermal management system uses a microcontroller-based PWM fan controller to maintain optimal internal temperature while minimizing noise. Unlike simple on/off thermostat control, this system provides smooth, proportional fan speed adjustment.

Competitor Limitation: Nearest competitor uses a simple on/off thermostat at 30°C. This causes:

- Sudden fan noise when threshold is crossed (0% → 100% instantly)
- Constant on/off cycling near the setpoint
- Full-speed fans even when 20% would suffice
- No proportional control - binary only

Apex PWM Advantages: Same 30°C start point, but with intelligent ramping:

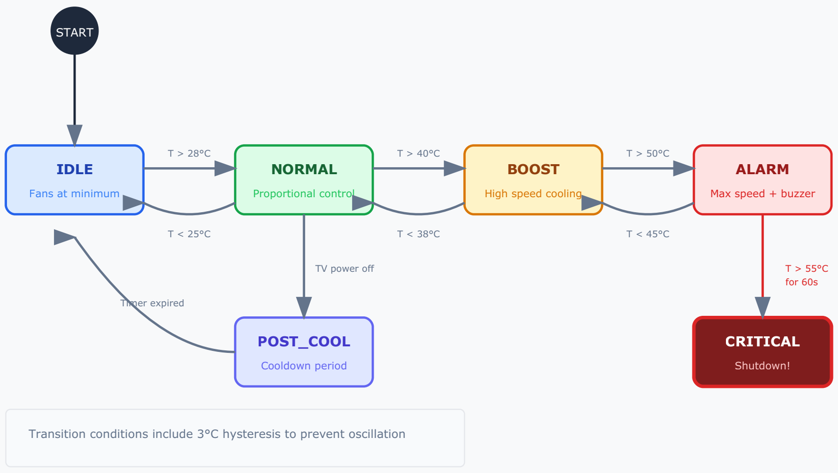
- Fans start at 30°C at **20% PWM** (near-silent)
- Gradual ramp: 50% at 35°C, 100% at 45°C
- Protects TV (rated 0-35°C) with headroom
- No sudden speed changes - smooth adjustment
- Post-cooling after TV power-off

2. System Parameters

Parameter	Value	Description	vs Competitor
`TEMP_START`	30°C	Fans begin at 20% PWM (near-silent)	Same threshold, but 20% vs 100%
`TEMP_MID`	35°C	Fans at 50% PWM	TV's max rated ambient temp
`TEMP_FULL`	45°C	Fans reach 100% PWM	Linear ramp 30-45°C
`TEMP_ALARM`	50°C	Warning LED, maintain 100% fans	TV at risk - alert user
`TEMP_CRITICAL`	55°C	Emergency - signal TV shutdown	Protect TV from damage
`PWM_MIN`	20%	Minimum fan speed (quiet operation)	
`PWM_MAX`	100%	Maximum fan speed	
`RAMP_RATE`	5%/sec	Maximum PWM change per second	
`SAMPLE_INTERVAL`	1000ms	Temperature sampling rate	
`POST_COOL_TIME`	300 sec	Cooling time after TV off	

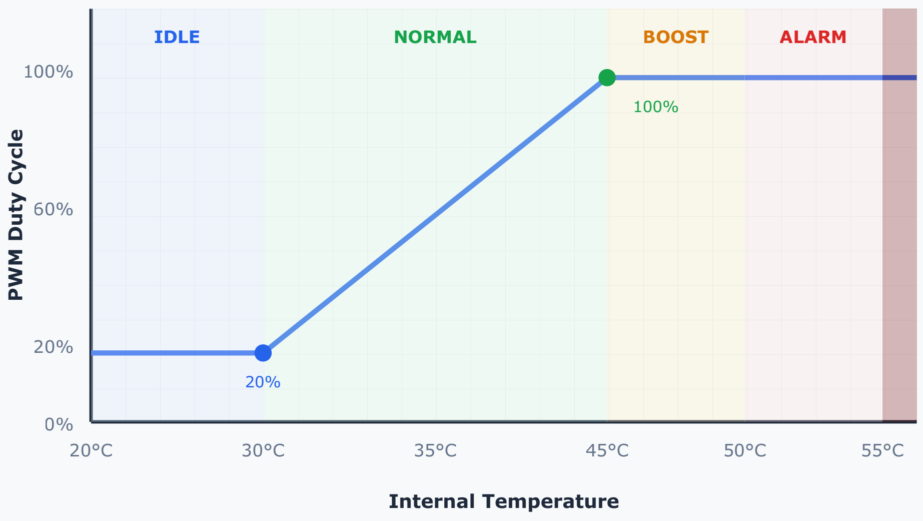
3. State Machine

Figure 1: Controller State Machine



4. PWM Calculation Algorithm

Figure 2: PWM vs Temperature Curve



PWM Calculation Formula

```
// Calculate PWM duty cycle based on temperature
function calculatePWM(temp) {
    if (temp ≤ TEMP_MIN) {
        return PWM_MIN;  // 20%
    }
    else if (temp ≥ TEMP_MAX) {
        return PWM_MAX;  // 100%
    }
    else {
        // Linear interpolation between MIN and MAX
        let range = TEMP_MAX - TEMP_MIN;  // 45 - 28 = 17°C
        let offset = temp - TEMP_MIN;
        let pwm = PWM_MIN + (offset / range) * (PWM_MAX - PWM_MIN);
        return pwm;
    }
}
```

5. Main Control Loop

```
// Main control loop - runs every SAMPLE_INTERVAL (1000ms)
function controlLoop() {
    // 1. Read temperature sensors
    let tempInternal = readSensor(SENSOR_INTERNAL);
    let tempExhaust = readSensor(SENSOR_EXHAUST);

    // 2. Use highest temperature for control
    let tempControl = max(tempInternal, tempExhaust);

    // 3. Check for critical condition
    if (tempControl ≥ TEMP_CRITICAL) {
        criticalCounter++;
        if (criticalCounter ≥ 60) {  // 60 seconds at critical
            enterCriticalState();
            return;
        }
    } else {
        criticalCounter = 0;
    }

    // 4. Update state machine
    updateState(tempControl);

    // 5. Calculate target PWM based on state
    let targetPWM;
    switch (currentState) {
        case STATE_IDLE:
            targetPWM = PWM_MIN;
            break;
        case STATE_NORMAL:
            targetPWM = calculatePWM(tempControl);
            break;
        case STATE_BOOST:
        case STATE_ALARM:
            targetPWM = PWM_MAX;
            break;
        case STATE_POST_COOL:
            targetPWM = 50;  // Medium speed cooldown
            break;
    }

    // 6. Apply smooth ramping
    currentPWM = rampTo(currentPWM, targetPWM, RAMP_RATE);

    // 7. Set fan speed
    setFanPWM(FAN1, currentPWM);
    setFanPWM(FAN2, currentPWM);

    // 8. Handle alarm
    if (currentState == STATE_ALARM) {
        activateBuzzer(true);
    } else {
        activateBuzzer(false);
    }
}

// Smooth ramping function - prevents sudden speed changes
function rampTo(current, target, rate) {
    let diff = target - current;
    if (abs(diff) <= rate) {
        return target;
    }
    return current + sign(diff) * rate;
}
```

6. Failsafe Behavior

Critical Safety Feature: The controller includes hardware watchdog and failsafe logic.

Condition	Detection	Action
Sensor failure	Reading out of range (-40 to +85°C)	Set fans to 100%, activate alarm
Fan failure	Tachometer reads 0 RPM when PWM > 30%	Activate alarm, log error
Over-temperature	Internal temp > 55°C for 60 seconds	Optional: Cut TV power relay
MCU crash	Watchdog timer expires (2 seconds)	Hardware reset, fans default to 100%

7. Hardware Requirements

7.1 Microcontroller

- **Recommended:** ATmega328P or ESP32-C3
- 2+ PWM outputs (25kHz capable for silent operation)
- 1-Wire interface for DS18B20 sensors
- Digital I/O for alarm output
- Optional: WiFi/BLE for remote monitoring (ESP32)

7.2 PWM Frequency

Important: Use 25kHz PWM frequency for silent fan operation. Lower frequencies (e.g., 490Hz Arduino default) cause audible whine.

7.3 Suggested Components

Component	Part Number	Notes
MCU	ATmega328P-AU	Arduino compatible
Voltage Regulator	AMS1117-3.3	3.3V for sensors
MOSFET Driver	IRLZ44N	For PWM output (if needed)
Pull-up Resistor	4.7kΩ	For 1-Wire bus
Decoupling Caps	100nF + 10µF	Per IC

8. Testing Requirements

1. Verify PWM output at 25kHz ±5%
2. Test temperature reading accuracy ±1°C
3. Verify smooth fan ramp (no audible steps)
4. Test all state transitions
5. Verify failsafe activates on sensor disconnect
6. Test watchdog reset functionality
7. Measure noise at each PWM level
8. 48-hour burn-in test