1. Overview of System Functionality

- Purpose:
- The Arduino sketch controls a multi-channel relay system (20 channels) with up to 8 I2C-connected switch plates, each supporting multiple switches (up to 20 total switches).
- It manages physical switch inputs to toggle Solid State Relays (SSRs) or adjust dimming, controls LED indicators, and supports many-to-many mappings between switches and channels.
- The system includes timers, persistent storage (EEPROM), and serial communication for configuration and monitoring.

- System Role:
- Designed to replace an aging and frequently faulty C-Bus system, reusing the existing Cat5 wiring structure that daisy-chains from the switchboard to each switch plate and back to the switchboard.
- Controls multiple devices (e.g., lights, fans) via physical switches or serial commands, with customizable mappings and timed operations.

- Key Components:
- Arduino Board: Mega (due to high pin count and memory needs).
- Hardware:
- Up to 8 I2C switch plates (e.g., MCP23017 I/O expanders) for switch inputs and LED outputs.
- 20 SSRs connected to digital pins for controlling external devices.
- PWM-capable pins for dimming (e.g., pin 9 for channel 5).
- Zero-crossing interrupt pin for dimming timing.
- EEPROM for persistent storage of mappings, names, and timers.
- Interactions:
- Reads switch states via I2C, debounces inputs, and detects short press (<230ms, toggle) or hold (>=230ms, dimming; >=3s, Global Timer).
- Toggles SSRs or adjusts dimming based on switch presses or serial commands.
- Updates LEDs to reflect channel states.
- Dimming requires channel to be toggled ON via SSR before dimming is available (240V 50Hz wiring: SSR to dimmer to light).
- Supports Program Mode for serial configuration of mappings, names, and timers.

2. Detailed Behavior Breakdown

2.1 Inputs
- Switch Inputs (via I2C):
- Source: Up to 8 I2C plates (MCP23017 I/O expanders, addresses 0x21-0x27) with 1-6 switches per plate, totaling 20 switches. Example configuration: const int switchPlateI2CAddresses[MAX_PLATES] = {0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0}; const int numSwitchesPerPlate[MAX_PLATES] = {6, 1, 2, 2, 4, 4, 1, 0};
- Hardware Configuration: All plates are wired identically:
- Switches: Connected to GPA pins (GPA0-GPA5 for 6 switches). One side of each switch is grounded, the other connects to a GPA pin, using internal pull-up resistors (enabled via register 0x0C, GPPUA).
- LEDs: Connected to GPB pins (GPB0-GPB5 for 6 switches). Each LED is wired from a GPB pin through an external resistor to the LED, then to ground (common cathode, COMMON_ANODE = false). Controlled via register 0x13 (OLATB).
- For plates with fewer switches (e.g., 1 switch), only corresponding GPA/GPB pins are used (e.g., GPA0, GPB0); unused pins are configured as inputs to avoid interference.
- Processing:

- Reads switch states from register 0x12 (GPIOA), where LOW indicates a pressed switch (grounded) and HIGH indicates unpressed (pulled up).
- Debounces using millis() and debounceStartTime[] with 50ms period (DEBOUNCE_DELAY) non-blockingly.
- Detects short presses (<230ms, TOGGLE_HOLD_THRESHOLD) for toggling, holds (>=230ms) for dimming, or 3s for Global Timer.
- Serial Input:
- Source: Serial port at 9600 baud, parsed non-blockingly using serialBuffer (128-byte char array).
- Processing:
- Commands are case-insensitive (e.g., "CH1_1 ON" or "pendant off").
- Partial/malformed commands (e.g., "CH1_1", "CH1_1 ON 150") are ignored, logging verbose errors in Normal and Program Modes (e.g., "Error: Incomplete command. Use '<channel> ON', '<channel> OFF', or '<channel> ON <dim level>'", "Error: Invalid dim level. Use 30-100").
- Invalid channel names log "Error: Channel not found. Use valid channel name (e.g., 'CH1_1' or 'Pendant')."

2.2 Outputs
- Relay Boards:
- Pins: const int relayControlPins[TOTAL_CHANNELS] = {22, 23, 24, 25, 26, 27, 28, 29, 10, 11, 12, 13, 30, 31, 32, 33, 34, 35, 36, 37}; for 20 channels (2A SSRs for lights on 22-29, 30-37; 10A mechanical relays for fans on 10-13).
- Control Logic: Set HIGH/LOW based on channelStates[], updated by switch presses or serial commands.
- Dimming (PSM):
- Pins: PWM-capable pins in psmPins[] (e.g., psmPins[4] = 9 for channel 5, 0 for non-dimmable). Currently only channel 5 is dimmable, but supports multiple channels (non-zero psmPins[c]).
- Control Logic:
- Outputs 125us pulses (pulseWidth) after zero-crossing interrupt (pin 2, INPUT_PULLUP, rising edge via attachInterrupt()), timed by dimLevel (30-100%) and T8_MULTIPLIER (102.0us/%).
- Hold (>=230ms) decrements dimLevel every 100ms (time-based) for smooth dimming, stopping at release. Resets to 99% on toggle ON.
- Requires channel ON via SSR (240V 50Hz wiring: SSR to dimmer to light).
- Only one dimmable channel per group, enforced in processMappingCommand().
- LEDs (via I2C):
- Control: Written to register 0x13 (OLATB), HIGH to light (common cathode, COMMON_ANODE = false).
- Logic:
- Normal Mode: ON if any mapped channel is ON, OFF if all are OFF or switch is unmapped.
- Program Mode: Flash 1s ON/OFF for all LEDs.
- Timer Mode: Flash 250ms ON/OFF, 1500ms pause if any mapped channel has an active timer.
- Precedence: Timer > Program > Normal.
- Serial Output:
- Normal Mode (Homebridge-friendly):
- Non-verbose: "<name> ON/OFF [<dim level>]" (e.g., "Pendant ON 50", "Channel1_5 OFF"), using custom channelNames[] if set, else defaultChannelNames[] (e.g., "CH1_1"). Dim level (30-100) included only for dimmable channels when ON.
- Startup: "System Ready!".
- Triggered by switch presses or serial commands.
- Program Mode (Verbose):

- Detailed logging for all actions (e.g., "Switch5_1 toggled Channel1_5 OFF", "Mapping saved: Channel1_1 to Switch1_3. Previous mappings cleared.", "System reset complete.").
- Uses custom names if set, else default names.
- Entry: "Program Mode Entered!".

2.3 State Changes and Logic
- Switch Press:
- Short Press (<230ms): Toggles all mapped channels ON (dimLevel = 99% for dimmable) or OFF (dimLevel = 0). Starts per-channel timers if set (channelTimerDurations[c] > 0). Cancels all timers (per-channel and Global) when OFF. Updates channelStates[], SSRs, LEDs.
- Hold (>=230ms): For ON dimmable channels (psmPins[c] != 0), decrements dimLevel (100% to 30%) every 100ms, stops at release, maintains dimLevel. Resets to 99% on toggle ON. Ignored if no channels are ON/dimmable.
- Long Hold (>=3s): If all mapped channels are OFF and timerDuration > 0, turns channels ON and sets timerEndTime[c] to millis() + timerDuration * 60000UL. Does not cancel per-channel timers; all timers run concurrently, with the longest controlling turn-off.
- Unmapped Switches: Do nothing.
- Timer Logic:
- Global Timer (timerDuration, 0-65535 minutes):
- Set via "GlobalTimer <duration>" in Program Mode. Defaults to 0 on startup/reset.
- Applied by 3s hold (all mapped channels OFF), setting timerEndTime[c] for each channel. Runs concurrently with per-channel timers.
- Canceled when any channel is toggled OFF (switch or serial); requires another 3s hold to reapply.
- Stored in EEPROM.
- Per-Channel Timers (channelTimerDurations[], 0-65535 minutes):
- Set via "<channel> TIMER <duration>" in Program Mode. Defaults to 0 on startup/reset.
- Activate when channel is turned ON, canceled when turned OFF.
- Stored in EEPROM.
- Longest Timer Rule: For grouped channels, the longest timer (Global or per-channel) controls turn-off. Timers run concurrently. Example: Channel1_1 (10-minute timer), Channel1_2 (5-minute timer), Global Timer (15 minutes) on Switch1_1 turns off after 15 minutes.
- Cancellation: Turning off a channel (switch or serial) cancels its per-channel and Global Timer effects (channelTimerDurations[c] = 0, timerEndTime[c] = 0). For partial group turn-off (e.g., "CH1_1 OFF"), only the affected channel's timers are canceled; other group members' timers continue.
- Expiration: When any timerEndTime[c] <= millis(), the channel turns OFF (channelStates[c] = false, dimLevel[c] = 0).
- Program Mode:
- Entered via "EnterProgramMode", exited via "ExitProgramMode" (saves mappings if dirtyMappings).
- Mapping commands (e.g., "CH1_1 SW1_3") warn if multiple timers exist: "Warning: Multiple timers in group. Longest timer will control turn-off."
- ClearMappings and SystemReset prompt "Confirm (Y/N)?", waiting 10 seconds for Y/N. No response logs "Error: No response received. Command canceled."; invalid responses log "Error: Invalid response. Enter 'Y' or 'N'." and reset timeout.
- Default Mappings:
- On startup or SystemReset with 1:1 mapping option, map channels to switches 1:1 (e.g., Channel1_1 to Switch1_1) up to 20 switches/channels. Unmapped channels/switches have no mappings.

- Mapping Behavior:
- New mapping (e.g., "CH1_1 SW1_3"):
- Clears existing mappings for specified channels (channelToSwitchMap[][], numSwitchesPerChannel[]) and switches (switchToChannelMap[][], numChannelsPerSwitch[]).
- Creates new mappings for specified channels/switches.
- Breaks groups if not included in new mapping.
- Sets dirtyMappings for EEPROM save on ExitProgramMode.
- Logs "Mapping saved: Channel1_1 to Switch1_3. Previous mappings cleared."
- Constraints:
- One dimmable channel per group, enforced with "Error: Only one dimmable channel allowed per group."
- Multiple timers allowed, with warning: "Warning: Multiple timers in group. Longest timer will control turn-off."
- Serial Commands:
- Channel names (custom or default) are interchangeable, case-insensitive, resolved via findChannelIndex(). Invalid names return "Error: Channel not found."
- Normal Mode:
- <channel> ON/OFF (e.g., "CH1_1 ON", "Pendant OFF").
- <channel> ON <dim level> (e.g., "CH1_1 ON 50").
- EnterProgramMode.
- Program Mode:
- ExitProgramMode: Saves mappings if dirtyMappings.
- <channel/s> <switch/s>: Maps up to 5 channels/switches.
- RENAME <oldName> <newName> (e.g., "RENAME CH1_1 Pendant").
- <channel> TIMER <duration> (e.g., "CH1_1 TIMER 10").
- GlobalTimer <duration> (e.g., "GlobalTimer 15").
- <channel> STATUS.
- STATUS: Displays channel states, mappings, timers, uptime, I2C plate status (see 2.6).
- ClearMappings: Clears all mappings after confirmation.
- SystemReset: Resets variables after confirmation and mapping choice (None or 1:1).
- Commands use short names (CHX_Y, SWP_S), outputs use full names (ChannelX_Y, SwitchP_S).

2.4 Timing-Related Behaviors
- Debouncing: 50ms period (DEBOUNCE_DELAY) using millis() and debounceStartTime[] non-blockingly.
- Toggle/Hold Detection: 230ms threshold (TOGGLE_HOLD_THRESHOLD) for short press (<230ms, toggle) vs. hold (>=230ms, dimming). 3s threshold for Global Timer, detected only if all mapped channels are OFF and timerDuration non-zero. Dimming hold detected only if at least one channel is ON and dimmable.
- Dimming: Decrements dimLevel every 100ms (DIM_STEP_INTERVAL), with 125us pulses (pulseWidth) triggered by zero-crossing interrupt.
- Timers: Set in minutes, converted to milliseconds (duration * 60000UL). Default to 0 on startup/reset.
- Uptime Tracking: Stored in volatile memory using uptimeSeconds (seconds since last 24-hour reset) and uptimeDays (days since startup). Every 24 hours, uptimeDays increments, uptimeSeconds resets to prevent overflow. Resets on power-off.
- Dropout Reporting: Every 10s (DROPOUT_REPORT_INTERVAL) for offline plates.
- Timing Method: Uses millis() for non-blocking timing, micros() for dimming precision.

2.5 Communication Protocols
- I2C (Wire):
- Role: Communicates with switch plates (MCP23017 I/O expanders).
- Configuration: 50kHz clock, 5000us (5ms) timeout, addresses 0x21-0x27.
- Operations:
- Reads switch states (register 0x12).
- Writes LED states (register 0x13).
- Configures ports in setup() (registers 0x00, 0x0C, 0x01).
- UART (Serial):
- Role: User interface for configuration, monitoring, and Homebridge integration.
- Configuration: 9600 baud.
- Operations: Parses commands non-blockingly, sends status updates, logs events.

2.6 Serial Output Formatting
- The STATUS command in Program Mode generates a formatted table displaying all channels' states, mappings, timers, system uptime, and I2C plate status. The output uses fixed-width fields to ensure aligned, readable rows, with no staggering due to varying name lengths or states.
- Channel Table Format:
- Columns:
- Default Name: 12 characters, left-aligned (e.g., Channel1_1).
- Renamed Name: 16 characters, left-aligned (e.g., Living Room, or (Unnamed) if not renamed).
- State: 8 characters, left-aligned (e.g., ON 50%, OFF).
- Mappings: 20 characters, left-aligned (e.g., SW1_1,SW1_2, or (None) if no mappings).
- Timer: 7 characters, right-aligned (e.g., 10m, None for no timer).
- Header row and dashed line separate columns for clarity.
- Global Timer: Displayed below the channel table, 7 characters, right-aligned (e.g., 30m, None).
- System Uptime: Displayed as a single row labeled System Uptime, with value in 16 characters, left-aligned (e.g., 5d 12h 34m 56s for 5 days, 12 hours, 34 minutes, 56 seconds).
- Plate Status Table:
- Columns:
- Plate ID: 12 characters, left-aligned (e.g., Plate 1).
- Address: 8 characters, left-aligned (e.g., 0x27, 0x00 for unused).
- Status: 20 characters, left-aligned (e.g., Online, Offline (2 retries), Not Configured).
- Switch Count: 7 characters, right-aligned (e.g., 6, 0 for unused).
- Lists all 8 possible plates (per MAX_PLATES), with header row and dashed line.
- Example Output:

Default Name	Renamed Name	State	Mappings	Timer
Channel1_1	Living Room	ON 50%	SW1_1,SW1_2	10m
Channel1_2	(Unnamed)	OFF	SW1_3	None
Channel1_3	Kitchen Light	ON	(None)	5m
Channel1_4	(Unnamed)	OFF	SW1_4,SW2_1,SW3_1	None
Channel2_1	Ceiling Fan	ON	SW2_2	20m
Channel3_1	(Unnamed)	OFF	(None)	None
...				
Channel3_8	Porch Light	ON 75%	SW8_1	15m

Global Timer: 30m

```
System Uptime  5d 12h 34m 56s

Plate ID      Address  Status                Switch Count
------------  -------  --------------------  ------------
Plate 1       0x27     Online                6
Plate 2       0x26     Online                1
Plate 3       0x25     Offline (2 retries)   2
Plate 4       0x24     Online                2
Plate 5       0x23     Online                4
Plate 6       0x22     Online                4
Plate 7       0x21     Online                1
Plate 8       0x00     Not Configured        0
```
- Rebuild Requirement: The new Arduino sketch must implement this formatted output for the STATUS command, using fixed-width fields (e.g., via sprintf or string padding) to ensure alignment. Uptime must be calculated from uptimeDays and uptimeSeconds, displayed as days, hours, minutes, and seconds. Plate status must reflect I2C communication state, including retry counts for offline plates.

3. Code Structure Analysis

3.1 Key Variables
- EEPROM_MODE - const String - Set to "LTS" (0-1999) or "TESTING" (2000-4000) to separate production and testing data.
- EEPROM_START_ADDRESS - const int - 0 for LTS, 2000 for TESTING.
- MAX_SWITCHES - const int - Maximum number of switches (20).
- REG_SWITCH_STATE - const int - I2C register for switch states (0x12).
- REG_LED_STATE - const int - I2C register for LED states (0x13).
- relayControlPins[] - const int[TOTAL_CHANNELS] - {22, 23, 24, 25, 26, 27, 28, 29, 10, 11, 12, 13, 30, 31, 32, 33, 34, 35, 36, 37} for SSRs.
- switchPlateI2CAddresses[] - const int[MAX_PLATES] - {0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0} for I2C plates.
- numSwitchesPerPlate[] - const int[MAX_PLATES] - {6, 1, 2, 2, 4, 4, 1, 0} for switch counts.
- totalSwitches - int - Total switches across all plates (calculated in setup(), up to 20).
- psmPins[] - const int[TOTAL_CHANNELS] - PWM pins for dimming (e.g., {0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} for channel 5). Non-zero indicates dimmable channel.
- zcPin - const int - Pin 2 for zero-crossing interrupt.
- channelStates[] - bool[TOTAL_CHANNELS] - Current state of each channel (true = ON).
- dimLevel[] - int[TOTAL_CHANNELS] - Dimming level per channel (30-100, 0 if OFF or non-dimming).
- ledStates[] - bool[MAX_SWITCHES] - LED states for each switch (static array).
- lastSwitchStates[] - bool[MAX_SWITCHES] - Previous switch states for debouncing.
- lastPressTime[] - unsigned long[MAX_SWITCHES] - Timestamps for switch press events.
- lastReleaseTime[] - unsigned long[MAX_SWITCHES] - Timestamps for switch release events.
- pressCount[] - int[MAX_SWITCHES] - Tracks press counts per switch.
- holdTriggered[] - bool[MAX_SWITCHES] - Tracks hold events per switch.
- holdState[] - int[MAX_SWITCHES] - Tracks hold states (0-1 for toggle/hold).
- switchNames[] - String[MAX_SWITCHES] - Switch names (static array).

- switchToChannelMap[][] - int[MAX_SWITCHES][MAX_CHANNELS_PER_SWITCH] - Maps switches to channels (up to 5 channels per switch). Cleared for specified switches during re-mapping, updated with new mappings.
- numChannelsPerSwitch[] - int[MAX_SWITCHES] - Number of channels per switch (max 5). Reset to 0 for specified switches during re-mapping, updated for new mappings.
- channelToSwitchMap[][] - int[TOTAL_CHANNELS][MAX_SWITCHES_PER_CHANNEL] - Maps channels to switches (up to 5 switches per channel). Cleared for specified channels during re-mapping, updated with new mappings.
- numSwitchesPerChannel[] - int[TOTAL_CHANNELS] - Number of switches per channel (max 5). Reset to 0 for specified channels during re-mapping, updated for new mappings.
- timerDuration - uint16_t - Global Timer duration (0-65535 minutes, 0 on startup/reset, stored in EEPROM).
- channelTimerDurations[] - uint16_t[TOTAL_CHANNELS] - Per-channel timer durations (0-65535 minutes, 0 on startup/reset, stored in EEPROM).
- timerEndTime[] - uint32_t[TOTAL_CHANNELS] - Timer expiration times for channels.
- flashState, timerFlashState - bool, int - Control LED flashing in Timer mode.
- uptimeSeconds - uint32_t - Seconds since last 24-hour reset, stored in volatile memory, resets on power-off.
- uptimeDays - uint32_t - Days since startup, stored in volatile memory, resets on power-off.
- serialBuffer - char[128] - Fixed-size buffer for non-blocking serial input commands, sized for mapping up to 5 channels to 5 switches.
- serialBufferIndex - int - Tracks current position in serialBuffer for appending characters.
- channelNames[], defaultChannelNames[] - char[TOTAL_CHANNELS][16] - Custom and default channel names (max 15 chars).
- dirtyMappings - bool - Flags pending mapping changes for batched EEPROM writes on ExitProgramMode.
- dirtyLEDs - bool - Flags channel state or mode changes for LED updates.
- switchStates[] - SwitchState[MAX_SWITCHES] - State machine data for each switch (state, timestamps).
- loopsPerStep - const int - Loops before decrementing dim level (70, replaced by time-based DIM_STEP_INTERVAL).
- pulseWidth - const int - PWM pulse width for dimming (125us).
- T8_MULTIPLIER - const float - Timing multiplier for dimming (102.0us/%).
- TOGGLE_HOLD_THRESHOLD - const unsigned long - Threshold (230ms) for toggle (<230ms) vs. hold (>=230ms, dimming).
- COMMON_ANODE - const bool - false for common cathode LEDs.
- CONFIRM_TIMEOUT - const unsigned long - 10000ms (10s) for ClearMappings/SystemReset prompts.
- DIM_STEP_INTERVAL - const unsigned long - 100ms for time-based dimming steps.
- EEPROM data (842 bytes):
- timerDuration (2 bytes, offset 0)
- channelTimerDurations[] (40 bytes, offset 2)
- channelNames[] (320 bytes, offset 42)
- switchToChannelMap[][] (200 bytes, offset 362)
- channelToSwitchMap[][] (200 bytes, offset 562)
- numChannelsPerSwitch[] (40 bytes, offset 762)
- numSwitchesPerChannel[] (40 bytes, offset 802)

3.2 Functions

- getGlobalSwitchIndex(plateIdx, switchIdx) - Calculates global switch index from plate and switch indices - Inputs: plateIdx, switchIdx - Output: Global index (int).
- findChannelIndex(channelName) - Finds channel index by name (custom or default, case-insensitive) for all command processing (e.g., toggling, mapping, renaming) - Input: channelName - Output: Channel index (int, -1 if not found).
- updateLEDsForPlate(plateIdx) - Updates LEDs for a plate based on mode (Normal, Program, Timer) - Input: plateIdx - Output: None (writes to I2C).
- updateSSRs() - Sets SSR pins based on channelStates[] - Input: None - Output: None (writes to pins).
- updateDimming() - Outputs PWM pulses for dimming based on dimLevel[] and zero-crossing - Input: None - Output: None (writes to psmPins[]).
- updateLEDsBasedOnChannels() - Updates all LEDs based on channel states (Normal Mode only) - Input: None - Output: None (calls updateLEDsForPlate()).
- toggleChannelByName(channelName, state) - Toggles a channel and its mapped group, cancels timers if OFF, starts per-channel timers if ON - Inputs: channelName, state - Output: None (updates states, SSRs, LEDs).
- toggleSwitch(switchIdx) - Toggles all channels mapped to a switch, cancels timers if OFF, starts per-channel timers if ON - Input: switchIdx - Output: None (updates states, SSRs, LEDs).
- dimSwitch(switchIdx) - Adjusts dimming for mapped dimmable channels - Input: switchIdx - Output: None (updates dimLevel[]).
- setGlobalTimer(switchIdx) - Applies Global Timer to mapped channels if all OFF and timerDuration > 0 - Input: switchIdx - Output: None (updates timerEndTime[]).
- processSerialInput() - Parses serial commands non-blockingly, executes Normal/Program Mode commands - Input: None - Output: None (updates states, mappings).
- processMappingCommand(command) - Processes Program Mode mapping commands, validates inputs, updates mappings - Input: command - Output: None (updates mappings, dirtyMappings).
- displayStatus() - Generates formatted STATUS output (channels, timers, uptime, plates) - Input: None - Output: None (serial output).
- saveMappingsToEEPROM() - Saves mappings, names, timers to EEPROM if dirtyMappings - Input: None - Output: None (writes to EEPROM).
- loadMappingsFromEEPROM() - Loads mappings, names, timers from EEPROM, validates data - Input: None - Output: None (updates variables).
- clearEEPROM() - Resets EEPROM data to defaults (empty mappings, names, timers)
- Input: None - Output: None (writes to EEPROM).
- updateTimers() - Checks timer expirations, turns off channels as needed - Input: None - Output: None (updates channelStates[], dimLevel[]).
- setChannelTimer(channelIdx, duration) - Sets per-channel timer duration - Inputs: channelIdx, duration - Output: None (updates channelTimerDurations[]).
- cancelChannelTimer(channelIdx) - Cancels per-channel timer - Input: channelIdx
- Output: None (resets channelTimerDurations[], timerEndTime[]).
- applyGlobalTimer(switchIdx) - Applies Global Timer to mapped channels - Input: switchIdx - Output: None (updates timerEndTime[]).
- retryI2COperation(address, registerAddr, operation) - Retries I2C read/write on failure - Inputs: address, registerAddr, operation - Output: Success/failure (bool).

3.3 Libraries
- Wire.h: For I2C communication with switch plates.
- EEPROM.h: For persistent storage of mappings, names, and timers.

3.3.1 MCP23017 Configuration

- MCP23017 I/O Expander (via Wire.h):
- Role: Manages switch inputs (GPA0-GPA5) and LED outputs (GPB0-GPB5) for each plate.
- Configuration in setup():
- Register 0x00 (IODIRA): 0xFF (GPA inputs).
- Register 0x01 (IODIRB): 0x00 (GPB outputs).
- Register 0x0C (GPPUA): 0xFF (GPA pull-ups).
- Only numSwitchesPerPlate[plateIdx] pins are used per plate; unused pins are inputs.
- Operations:
- Read switches from register 0x12 (GPIOA, LOW for pressed).
- Write LEDs to register 0x13 (OLATB, HIGH to light, common cathode).
- Notes: Uniform wiring simplifies logic; COMMON_ANODE = false.

3.4 Pin Configurations
- I2C Pins:
- SDA, SCL: Blue pair in Cat5 cable, with 4.7kOhm pull-ups to 5V. Configured via Wire.begin().
- Switches: GPA pins (register 0x12), grounded when pressed, internal pull-ups.
- LEDs: GPB pins (register 0x13), resistor to LED to ground (common cathode).
- SSR Pins: relayControlPins[], OUTPUT, HIGH/LOW per channelStates[].
- PWM Pins: psmPins[] (e.g., pin 9 for channel 5), OUTPUT, LOW initially.
- Zero-Crossing Interrupt: Pin 2, INPUT_PULLUP, attachInterrupt() on RISING.

3.5 Memory Management
- Static Allocation: Uses fixed-size arrays (e.g., serialBuffer, channelNames) to avoid dynamic memory fragmentation.
- Persistent Storage:
- EEPROM for mappings, names, timers (842 bytes).
- LTS mode (0-1999) for production, TESTING mode (2000-4000) for development.
- Volatile Storage: uptimeSeconds, uptimeDays reset on power-off.

4. Finite State Machine for Switch Inputs
- States: IDLE, PRESSED, HELD_230MS, HELD_3S
- Transitions:
- IDLE: On press, moves to PRESSED, records pressTime.
- PRESSED:
- If held >=230ms, moves to HELD_230MS.
- If released <230ms, calls toggleSwitch(), returns to IDLE.
- HELD_230MS:
- If held >=3s and all mapped channels OFF, moves to HELD_3S, calls setGlobalTimer().
- If held and mapped channels ON/dimmable, calls dimSwitch().
- If released, returns to IDLE.
- HELD_3S: On release, returns to IDLE.
- Implementation: Per-switch state machine in switchStates[] (enum and struct).

5. Optimizations and Recommendations for Rebuild

5.1 Modularity
- Separate modules (.h/.cpp) for switch handling, serial processing, EEPROM management, LED/SSR control, timers.
- Example: SwitchHandler.h/.cpp for debouncing, FSM, toggle/dim/Global Timer actions.

5.2 Non-Blocking Design

- Use millis() for debouncing, timers, LED flashing.
- Use micros() for dimming precision.
- Avoid delay() to maintain responsiveness.

5.3 Memory Efficiency
- Use char arrays instead of String for serialBuffer, channelNames to prevent fragmentation.
- Static arrays for mappings (switchToChannelMap[][], channelToSwitchMap[][]).
- Minimize EEPROM writes by saving only when dirtyMappings is true.

5.4 Error Handling
- I2C Robustness: retryI2COperation() retries failed operations; 5000us timeout via Wire.setWireTimeout(5000, true). Log Wire.endTransmission() error codes.
- Input Validation:
- Serial commands: Validate tokens (non-empty, valid channel names). Log verbose errors (e.g., "Error: Invalid dim level. Use 30-100").
- Mappings: Enforce one dimmable channel per group, log "Error: Only one dimmable channel allowed per group."
- EEPROM Validation:
- On load, reset corrupted values to defaults, preserve valid data:
- timerDuration > 65535: Reset to 0.
- channelTimerDurations[c] > 65535: Reset to 0.
- channelNames[c] invalid (length >15, non-ASCII): Reset to defaultChannelNames[c].
- switchToChannelMap[s][i] invalid (channel >=20): Clear mapping.
- channelToSwitchMap[c][i] invalid (switch >=20): Clear mapping.
- numChannelsPerSwitch[s] > 5: Reset to 0, clear mappings.
- numSwitchesPerChannel[c] > 5: Reset to 0, clear mappings.
- Log errors (e.g., "Error: Invalid timerDuration, reset to 0") via serial.

5.6 Documentation
- Inline comments for all functions, explaining purpose, inputs, outputs.
- External documentation:
- README: Overview, hardware requirements (Arduino Mega, MCP23017, SSRs, dimmers), wiring (Cat5 daisy-chain, I2C SDA/SCL), serial command reference.
- Command reference: List Normal/Program Mode commands (e.g., "CH1_1 ON", "RENAME CH1_1 Pendant").
- I2C register details (0x12 for switches, 0x13 for LEDs).

5.7 Constraints
- Support up to 8 I2C plates (0x21-0x27).
- Maximum 20 switches, 20 channels.
- One dimmable channel per group.
- Timers in minutes (0-65535), converted to milliseconds.
- Serial buffer size (128 bytes) for mapping commands (up to 5 channels/switches).
- 4KB EEPROM, with LTS (0-1999) and TESTING (2000-4000) modes.

5.8 Hardware Setup
- Wiring:
- Cat5 Cable: Daisy-chained from switchboard to each MCP23017 plate (0x21-0x27) and back.
- Blue pair (Blue, Blue/White): SDA/SCL, 4.7kOhm pull-ups to 5V.
- Orange pair (Orange, Orange/White): 5V VCC, Ground.
- Brown/Green pairs: Spares.
- Hardware Constraints:

- Assume generic 2A SSRs for lights (pins 22-29, 30-37), 10A SSRs for fans (pins 10-13), TRIAC-based dimmer for channel 5 (pin 9).
- MCP23017 plates powered at 5V, typical ~50mA per plate (TBD).
- Setup Guide:
- Configure MCP23017 address jumpers (A0-A2) for 0x21-0x27 (TBD).
- No cable length limits specified; assume standard I2C constraints (~100m max with 4.7kOhm pull-ups, TBD).
- Detailed schematic and instructions to be added later if needed.

6. Challenges and Considerations
- I2C Reliability: Long Cat5 cables may introduce noise or capacitance issues. Mitigated by 4.7kOhm pull-ups, 50kHz clock, and retryI2COperation().
- Timer Management: Concurrent Global/per-channel timers require careful handling to ensure longest timer rule is applied correctly.
- Serial Parsing: Robust parsing for case-insensitive commands and verbose error messages to guide users.
- Homebridge Integration: Clean Normal Mode outputs (e.g., "Pendant ON 50") for compatibility.

7. TODO
- Include hardware schematic in documentation (Arduino Mega, MCP23017 wiring, SSRs, dimmers, Cat5 daisy-chain).
- Add detailed setup guide (MCP23017 jumper settings, recommended Cat5 cable lengths).
- Test timer-based dimming implementation for multiple channels.
- Validate EEPROM error handling for partial corruption scenarios.