IoT

MDPI

Article

Local-First Residential Lighting Control with Edge-IoT Integration: Design, Implementation, and Analysis of Remote–Local Interference

Oussama Accouche

College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; oussama.accouche@aum.edu.kw

Abstract

In smart lighting systems, separating wall control from network supervision can preserve residential lighting operation during network outages, but remote processing may still affect local responsiveness. This paper presents the design, implementation, and analysis of a low-cost, local-first lighting controller deployed in a Lebanese residence. An Arduino Mega 2560 interprets single, double, triple, and long presses from 19 wall pushbuttons and controls 32 solid-state relay channels, including 23 loaded lighting circuits. The pushbuttons operate at 5 V DC, with mains switching confined to the electrical panel. An ESP32-C3 bridge running ESPHome supports integration with Home Assistant and provides connectivity to the HomeLab Dashboard (HLDB), a custom web application; the bridge and HLDB form an edge layer inside the residence, while output authority remains with the Mega. Software-in-the-loop evaluation of the firmware examined gesture recognition, command handling, and persistence. A timed simulation identified interference between remote reporting and wall-input processing, blocking button sampling for up to 272 ms. Bench measurements on a hardware replica confirmed the 0.4 s single-click delay and the electrical interface. A 30-day audit of one household recorded 133 remote requests, of which 83.5% requested switching lights off. The firmware, simulation harness, scripts, and results accompany the manuscript.

Keywords: residential lighting; lighting control; local control; pushbutton gestures; Arduino Mega; ESP32-C3; ESPHome; solid-state relay; software-in-the-loop; Internet of Things

Check for updates

Academic Editor: Amiya Nayak

Received: 15 October 2026
Revised: 24 November 2026
Accepted: 27 November 2026
Published: 29 November 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

In residential lighting systems, the wall switch is the primary interface for everyday operation, and occupants expect it to work whenever the lighting circuit has power. Interface accessibility and control logic influence how occupants use building systems [1,2], while preferences for automation vary between individuals [3]. Networked lighting, one of the most common Internet of Things (IoT) applications in the home, introduces additional control paths through smart switches, smart bulbs, and supervisory platforms. Depending on the architecture, remote access, groups, and scenes may rely on a hub, a local network, or a cloud service, while a smart bulb requires its electrical supply to remain connected. Other implementation challenges include compatibility between no-neutral switches and light-emitting diode (LED) lamps [4], fragmented communication protocols [5,6], and security and privacy concerns [7,8]. Meanwhile, smart-lighting research has focused on sensing

and data-driven control for energy savings and comfort in non-residential buildings [9–13], whereas the residential environment has received far less attention [14].

Supply conditions introduce an additional challenge. Lebanese households have long relied on combinations of intermittent utility electricity and private generators [15]. The economic crisis has intensified energy poverty [16], while photovoltaic systems have become increasingly prevalent [17]. In such environments, network equipment, supervisory servers, and lighting circuits may be powered from different sources. Consequently, lighting interfaces dependent on network availability may become unusable even when the lighting circuits remain energized. The residence considered in this study is a two-floor villa supplied by a photovoltaic–battery system and supervised by the HomeLab Dashboard (HLDB), a lightweight custom web application. Its lighting subsystem follows a local-first architecture: an embedded controller independently processes physical wall-button inputs and exclusively manages lighting outputs, while networked services, hosted at the edge inside the residence rather than in a cloud, provide additional supervisory functionality without becoming necessary for local operation.

Local control is itself established practice: earlier systems keep manual pushbuttons operable when the network fails [18–20] (Section 2). However, removing network dependencies from the wall-control path does not by itself establish how that path behaves when local inputs and remote requests share controller resources. Command retries, state reporting, and persistent storage can still affect action ordering and input processing.

This study presents a low-cost, local-first solution for residential lighting control that the author designed, built, and installed. Wall pushbuttons operate individual room circuits through the controller without depending on the network, while remote control is provided through HLDB; Home Assistant, which supervised the installation previously, remains supported through the bridge. Each button uses multiple gestures, and buttons at different doorways act on shared circuit states. This engineering case study describes the design and implementation of this controller and evaluates its behavior with respect to four aspects:

1. Shared state and action ordering: how a common output state coordinates doorway buttons and remote requests, and how retransmitted absolute commands can supersede intervening wall actions when no manual-priority policy is implemented.
2. Gesture classification and response delay: how deferred resolution allows one button to operate up to three circuits while introducing a delay before single- and double-click actions.
3. Interference from remote processing: how blocking state reports interrupt button sampling, cause missed or misclassified gestures, and interact with stale timestamps to trigger premature persistence.
4. Persistence under interrupted supply: how delayed state storage reduces writes to the electrically erasable programmable read-only memory (EEPROM) at the cost of losing recent changes when power is interrupted.

The evidence comprises documentation of the installed prototype, including its electrical design, firmware, protocol and supervisory integration (Section 3); software-in-the-loop (SIL) experiments using idealized timing and a timed model of serial transmission and EEPROM writes (Sections 4 and 5); functional checks of local operation on the installation; bench measurements of the electrical interface and of the button-to-output timing on a hardware replica of the controller (Section 5.6); and a 30-day audit of remote lighting requests (Section 5.5). The contributions are, first, an analysis of the unmodified firmware of how remote reporting interferes with wall-button processing in the controller's loop; second, the multi-gesture, shared-state design of the controller; and third, a reproducible package of firmware, simulation harness, scripts, results, and schematics provided as Supplement

S1. The findings identify design considerations for controllers that process wall inputs and remote commands within the controller's loop.

2. Related Work

Lighting-control research has addressed occupancy-based control, which can reduce office lighting energy by up to 60% [21], the Digital Addressable Lighting Interface (DALI) on low-cost hardware [22,23], and wireless retrofits with Zigbee, Thread, or Matter [24–26]. Other work has explored integration through Message Queuing Telemetry Transport (MQTT) and Node-RED [27,28]. Edge processing has been motivated by the immediacy that users expect from a light control [29].

Low-cost residential systems typically combine a microcontroller, relays, and a remote interface, such as an Android application over Bluetooth [30] or a NodeMCU gateway with a cloud service [31]. Sowah et al. provide a particularly close comparison: their Arduino Mega with a 16-channel relay board is supervised by openHAB on a Raspberry Pi with token-based authentication, encryption, and web and Android interfaces, and debounced pushbuttons act as manual switches when the network is unavailable [18]. Andramuño et al. designed modules that work autonomously or with a central node and a mobile application, and modeled them with the Unified Modeling Language (UML) and Petri nets [19]; Villegas et al. kept pushbuttons on a Siemens LOGO! programmable logic controller (PLC) alongside ESP32-based voice control [20]; and Zagan and Găitan built an STM32 smart switch with a touch interface and an extended Modbus remote terminal unit (RTU) protocol [32]. Local operation also appears in gateway and platform studies, such as an ESP32 gateway evaluated in an educational building [33] and Home Assistant automation of an office [34]. Independence from the Internet is not independence from the network: the local mode of the ENERHOME system, for example, still relies on Wi-Fi [35].

These implementations were evaluated mainly by functional or prototype tests, complemented by penetration testing [18] or formal modeling [19]; Villegas et al. tested manual and voice operation [20], and Zagan and Găitan described an embedded implementation [32]. Chen et al. reported laboratory control and monitoring with a Raspberry Pi host that supervises PLCs over Modbus and MQTT through a web interface [28]. Galván et al. verified the firmware of an ESP32 monitoring node by hardware–software co-simulation before building it, noting that low-cost IoT studies rarely validate their embedded firmware [36]; the present work applies a comparable software-in-the-loop approach to the firmware of an installed controller.

The interface itself involves trade-offs. A review of human–building interfaces called for more research on how their design, context, and underlying logic affect usability and perceived control [1]. For physical buttons, a laboratory study rated a 1 s long press more usable than 2 or 3 s and measured the interval between the two clicks of a double click [37]. These findings motivate attention to gesture timing, although they do not establish suitable thresholds for the household studied here. Local processing does not remove security and privacy risks: attacks on building-automation protocols are documented [38], Zigbee lighting networks have been exploited through their MQTT bridges [7], and smart-home traffic, even encrypted, can reveal occupants' activities [8,39]. At the link level, the error-detection capability of a cyclic redundancy check (CRC) depends on its polynomial and on the message length [40]; a CRC detects accidental corruption but does not authenticate the sender.

The present system differs from the closest predecessors in four respects. Sowah et al. map each debounced pushbutton to one relay [18], and Andramuño et al. coordinate autonomous modules through a central node [19]; here, one button operates up to three circuits through deferred multi-gesture resolution, doorway buttons and remote requests

act on one controller-held state, and remote commands are absolute, allow-listed and CRC-framed, without a toggle command. Moreover, where those systems were evaluated by penetration testing or formal modeling, the unmodified firmware is evaluated here by fault injection and timed simulation, which reveals interference between remote reporting and button sampling that neither study examines.

These studies establish the relevance of local operation, interface timing, and communication integrity. The present work examines their interaction within one installed controller, focusing on how gesture processing, remote-command retries, state reporting, and persistence affect the local control path.

3. System Design

3.1. Site, Circuits and Requirements

The installation serves a two-floor residential villa with an approximate area of 200 m^2. Its lighting and the control side of the main distribution board are fed by a 220 V bus that a 16 A changeover relay transfers between the Main power source and the Backup source. Every lighting circuit has its own switched live conductor from the lighting panel, and every wall pushbutton is wired back to it. The controller has 32 outputs: 23 supply luminaires in 14 rooms or areas, four are configured as reserve circuits without load, and five are spares; the 27 lighting circuits are the controller's logical lights. Rooms with two or three circuits are named by role (bright, dim, or medium); these are separate switched circuits that provide multilevel switching [41] (p. 16.3), not dimming levels. Nineteen momentary, normally open pushbuttons are installed: the two kitchen doorways each have a button acting on the same circuits, and the guest room has a full-room button and a doorway button acting on a subset of its circuits.

Five requirements guided the design:

- **Local operation.** The wall buttons work without the bridge, Wi-Fi, or HLDB, provided that the controller and lighting circuits are powered.
- **Multi-circuit and shared-doorway control.** One button operates the circuits of its room through different gestures, and buttons controlling overlapping circuits act on the same stored states.
- **Output authority and remote addressing.** Only the panel controller writes outputs. Remote requests use absolute target states and address logical circuits, groups, floors, or spare channels through fixed allowlists.
- **Start-up and restoration.** Output latches are set low before the pins are configured as outputs, and a previous state is restored only from a validated record.
- **Audit without activity history.** Remote requests are audited, but no history of light states is retained to limit the recording of household activity patterns.

3.2. Development History

The Arduino Mega with its wall pushbuttons and solid-state relay (SSR) boards has been in daily use for several years, initially as a stand-alone controller without any network connection. In early 2025, an ESP32 running ESPHome was added to the Mega's serial port and gave local network access through the ESP's embedded web page. Home Assistant on a Raspberry Pi then supervised the bridge for several months before the installation moved to HLDB, which added remote access, supervisory control with permissions, and an audit database. The firmware and HLDB were revised along the way; the current firmware pair has been in service since late July 2026. Throughout these changes, the wall buttons never depended on the network layer, which was added and then replaced several times. HLDB

can also run on a Raspberry Pi, so the complete lighting stack can be reproduced at low cost; in this residence, it also supervises the energy system and other devices.

The wall-button interface also changed during this period. During early development, a binary circuit-control scheme was used for several weeks, and the household occupants found it confusing. It was replaced by the present scheme, in which single, double and, where configured, triple clicks select the circuits of a room and a long press switches them off; the occupants received this arrangement positively. The earliest firmware also switched all lights of the two floors off after a 5 s press of the living-room button, and later versions made this threshold and the button-to-circuit assignment adjustable on the bridge's web page. Both were removed: the occupants found the configurability too complex, and the wall all-off was not used deliberately and was triggered playfully by younger household members. The current version fixes the assignment in firmware tables and offers all-off only on the screens.

3.3. Architecture

Figure 1 shows the architecture. An Arduino Mega 2560 reads the 19 pushbuttons and drives four 8-channel SSR boards. An ESP32-C3 Mini running ESPHome [42] connects the Mega's serial port to the home network and exposes the circuits as entities of ESPHome's encrypted native application programming interface (API). HLDB subscribes to entity state updates, serves the lighting pages of the web dashboard and of two wall kiosks, checks permissions and audits every remote request. The local control path from pushbutton to Mega to SSR operates independently of the network components. The arrangement follows the central lighting control panel of commercial practice, in which low-voltage switches operate relays housed in the panel [41] (pp. 16.11–16.13); here the switches are 5 V pushbuttons, the relays are SSRs, and the panel logic runs on a microcontroller.

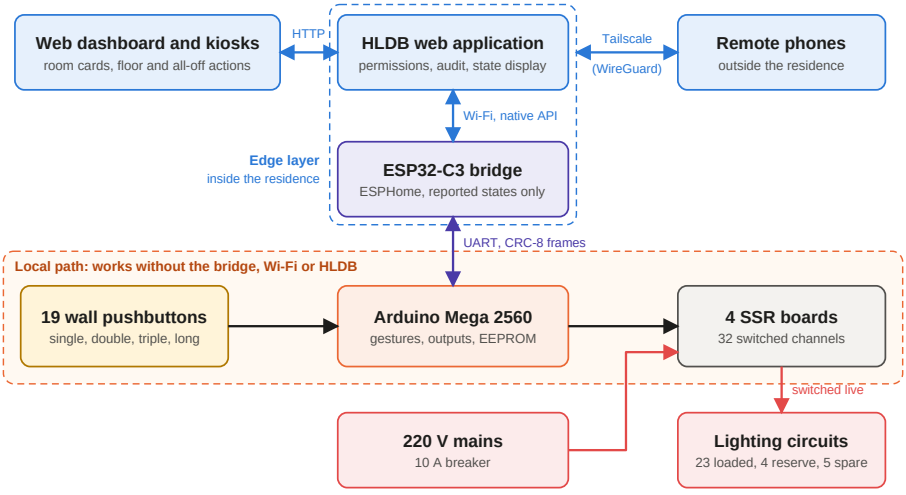

Figure 1. Architecture of the lighting system: local path (buttons, Mega, SSR boards), edge layer (bridge, HLDB), and remote access through Tailscale.

3.4. Electrical Design

This section describes the installed panel and identifies electrical limitations relevant to its reproduction. The KiCad schematic `LightingSSR_OnePage` shows the 5 V control circuitry on the left and the 220 V circuitry on the right of a single sheet (Figure 2).

A 220 V alternating-current (AC) to 5 V direct-current (DC) module (PS1, 5 A) supplies the Mega through its 5 V pin, the ESP32-C3 board, the SSR inputs, and the pushbutton circuits. The same module also supplies the installation's power controller.

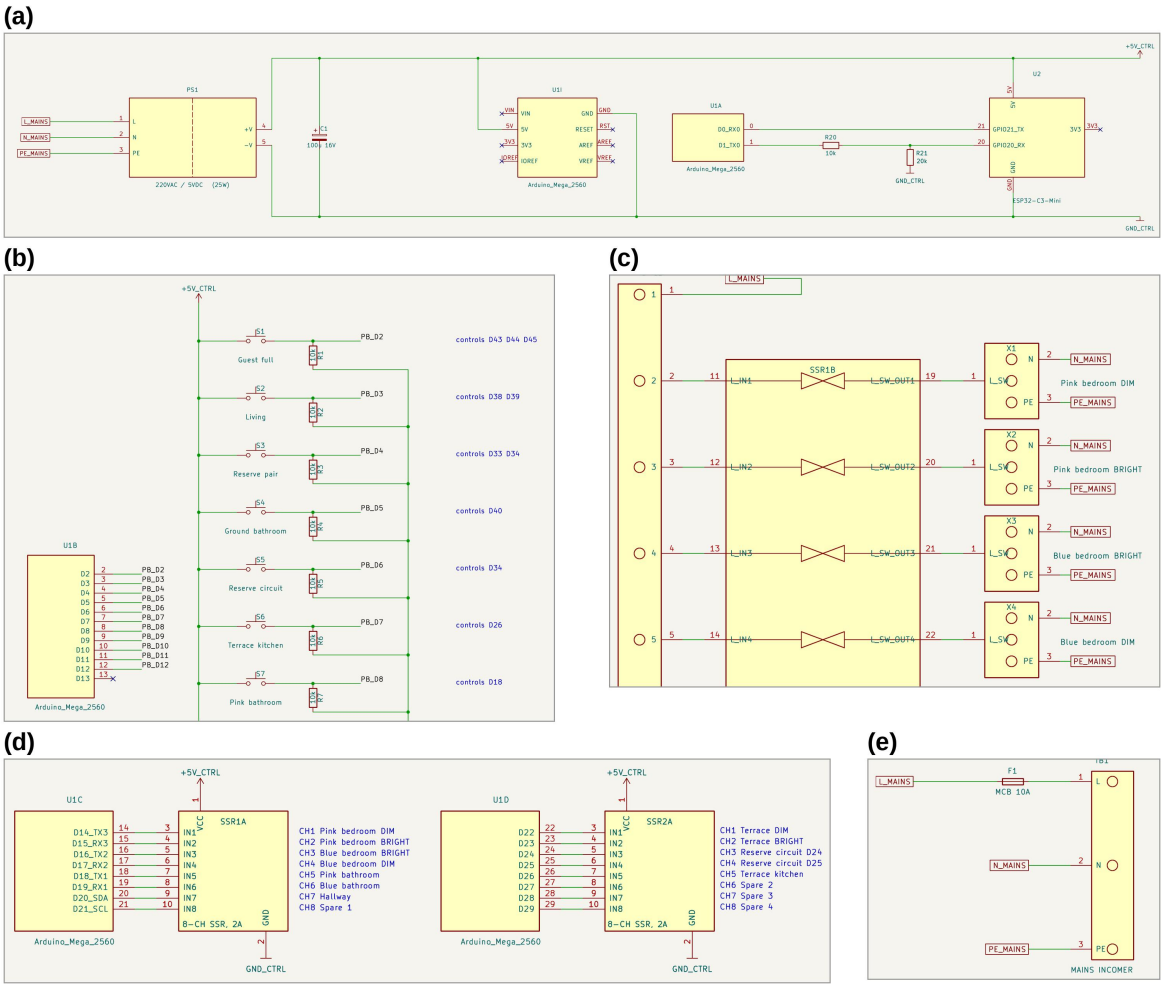

Figure 2. Excerpts of the KiCad schematic of the installed panel (revision B; complete schematics for revisions B and C are in Supplement S1). (**a**) Control supply and serial link with the R20/R21 divider. (**b**) Pushbutton inputs with 10 kΩ pull-downs. (**c**) Live distribution and SSR outputs. (**d**) SSR control inputs. (**e**) Mains incomer and 10 A breaker.

Each pushbutton connects its input to +5 V when pressed, and an external 10 kΩ pull-down resistor holds the input low when released, so the inputs are active high. The serial link uses the Mega's first universal asynchronous receiver–transmitter (UART), UART0, at 19,200 Bd. A 10/20 kΩ divider scales the Mega's 5 V transmit level to 3.33 V for the ESP32-C3; at the Mega's specified high output of 4.2 V the divided level is 2.8 V, still 0.33 V above the ESP32-C3's high-input threshold of 0.75 V_{DD} = 2.475 V [43,44]. In the reverse direction, the ESP32-C3 guarantees a high output of only 0.8 V_{DD} = 2.64 V, below the 3.0 V that the Mega requires at 5 V. On a bench replica (Section 5.6), the link exchanged frames without errors both with the divider and with a direct connection, and the ESP32-C3 drove the Mega's receive pin to 3.32–3.36 V, a margin of 0.32–0.36 V. This explains its operation in the installation since early 2025, but the margin belongs to one sample and is not guaranteed, and a direct connection also applies the Mega's 5 V level to a 3.3 V input. The installed link (revision B) is therefore distinguished from a reference link (revision C, Supplement S1), in which a 74AHCT1G125 buffer supplied at 5 V drives the Mega's receive pin; its transistor–transistor logic (TTL) input threshold of 2.0 V is met by the guaranteed 2.64 V [45]. The divider is kept for the other direction, and a level shifter of this kind is recommended for new builds.

The four identical SSR boards each carry eight G3MB-202P SSRs (input 5 V DC; load 100–240 V AC and 0.1–2 A, with zero-cross switching), an on-board input driver and indicator per channel, and a 2 A fuse on every output [46]. They switch the live conductors behind a single 10 A miniature circuit breaker, which protects the common feed, while the board fuses protect each channel. The input drivers keep the load on the Mega's pins small: on the bench replica, the current into one channel input was 0.42 mA, unchanged with eight channels on, whereas each SSR and its indicator drew about 9 mA from the board's 5 V supply (Section 5.6). Table 1 compares the resulting budget with the ATmega2560 limits [43]. During installation, all 32 SSR outputs were held on for several hours, which is functional evidence only.

Table 1. Current budget of the SSR drive with all 32 outputs on, from the per-channel values measured on the bench replica.

Quantity	Value	Limit
Current from one output pin	0.42 mA	40 mA absolute maximum per pin
Total from 32 output pins	13.5 mA	200 mA through the VCC and GND pins; below any port-group limit
SSR boards from PS1	about 0.28 A	5 A rating of PS1, shared with the other loads

All lighting circuits are LED. The largest single lighting circuit draws 1.47 A at 219 V (233 W), 74% of the SSR rating. With the maximum on-state voltage drop of 1.60 V, its SSR dissipates at most about 2.4 W without a heat sink. The 2 A rating applies up to an ambient temperature of 25 °C and decreases linearly to 0.5 A at 80 °C, so this channel remains within its rating up to about 44 °C [46]. The panel temperature was not measured, so this circuit has the smallest thermal margin; a heat sink or a higher-rated SSR is recommended for comparable circuits in panels that may exceed this temperature. The SSRs are optically isolated, with a dielectric strength of 2500 V AC between input and output, and the control and 220 V wiring occupy separate sides of the panel. Their surge rating is 30 A peak for one cycle and should be halved for repetitive inrush; the inrush current of the LED drivers was not measured. An SSR can fail short or open, leaving its lamp permanently on or off. Because the controller reports commanded rather than measured states, such a failure is noticed only by the occupants, and per-channel current sensing would be needed to detect it automatically. The off-state leakage of up to 1.5 mA at 200 V AC can make some LED lamps glow faintly.

3.5. Mega Firmware

The Mega runs an Arduino sketch and the bridge an ESPHome configuration with embedded C++; they are commissioned as a matched pair. Figure 3 shows the structure of the Mega firmware. Constant tables in flash describe the 27 logical lights, 5 spares, 18 groups of up to three lights, 2 floors and 19 buttons. Because every button and circuit is wired to the panel, these tables, not the wiring, define each button's function. The main loop services the serial receiver, the buttons, persistence, and the periodic snapshot in turn, without calls to `delay()`; all timing uses the millisecond counter with unsigned subtraction. The sketch uses 6% of the Mega's flash memory and 29% of its random-access memory (RAM).

All normal-operation output changes pass through one output manager, which writes the pin, stores the logical state, records the source of the action, marks the state for persistence and, if the state changed, emits a `STATE` report. Toggling a light and switching a group, a floor, or all lights off are built on it. Because the wall path and the serial path act on the same stored state, doorway buttons and remote requests never hold separate toggle histories; there is no manual-override latch, so the last processed action determines the commanded

state. Reported states are the commanded outputs; the system does not measure current or light output to confirm that a lamp follows them. Commands carry light, group, floor, or spare identifiers checked against their tables, never pin numbers. Spares belong to no button, group, floor, or all-off operation.

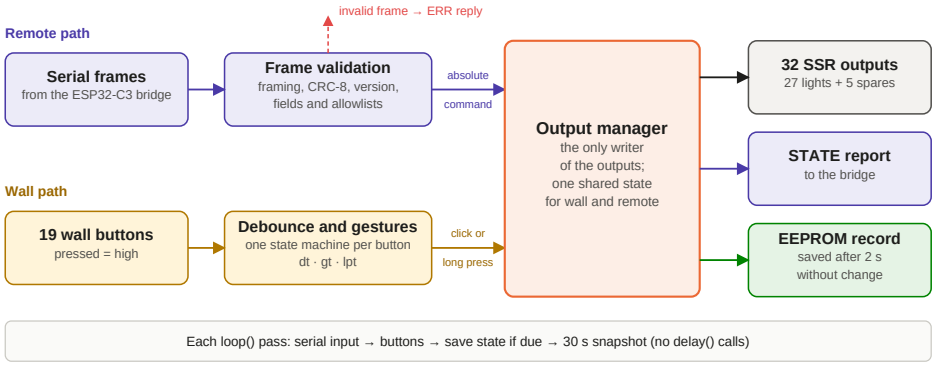

Figure 3. Mega firmware structure (simplified): serial commands and wall gestures converge on one output manager.

3.6. Gesture Recognition

Each input has its own debouncer and gesture state machine (Figure 4). The debouncer accepts a level once it has remained unchanged for the debounce time dt. Single and double clicks are resolved only when the multi-click gap time gt has elapsed after the last debounced release, so that the first click of a double click never toggles the single-click circuit on the way; a third click executes at once and closes the sequence. A long press executes the group-off action once while the button is still held, when the debounced press has lasted the long-press time lpt and the raw input is still active; its release is not counted as a click. After a triple click or long press, the firmware applies a lockout interval of gt, during which new presses are ignored. If an input remains active for 60 s, a diagnostic stuck-input counter is incremented without repeating the control action.

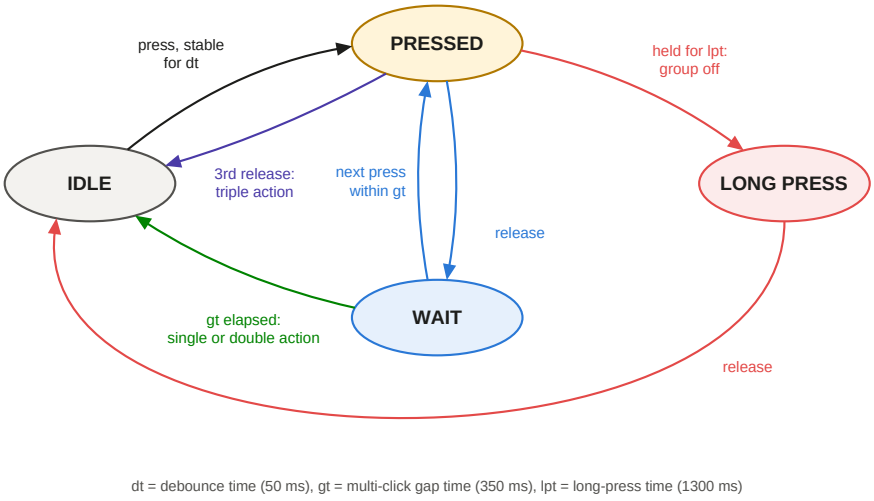

Figure 4. Per-button gesture state machine (simplified) with debounce time dt, multi-click gap gt and long-press time lpt; lockout and stuck-input handling are omitted.

The button behavior depends on the circuits assigned to it. A toggle switches a circuit on if it is off, or off if it is on.

- Single-circuit room: A single click toggles the circuit. A long press switches it off.

- Two-circuit room: A single click toggles the bright circuit, and a double click toggles the dim circuit. A long press switches both off.
- Three-circuit room: A single click toggles the bright circuit, a double click toggles the dim circuit, and a triple click toggles the medium circuit. A long press switches all three off.

Each click gesture affects only its assigned circuit, so multiple circuits can remain on together.

Two buttons in the same room can have identical or different assignments. In the kitchen, the buttons at the two doorways control the same circuits with identical gesture mappings. In the guest room, the assignments differ. One button controls all three circuits using the three-circuit mapping above. The second button controls only the bright and medium circuits, with a single and a double click respectively. Both buttons act on the same controller-held circuit states, so an action at either doorway is reflected in subsequent operation from the other.

The default timing parameters are dt = 50 ms, gt = 350 ms and lpt = 1300 ms. They are adjustable through the protocol within 20–150, 200–600 and 700–2500 ms, respectively, subject to dt < gt < lpt. Changes take effect in RAM and are written to EEPROM only through an explicit save command. These defaults were selected through iterative adjustment during residential use, guided by informal household feedback until the occupants found the settings satisfactory for everyday use; with the final values, the occupants reported no practical difficulties (see Section 6.3).

3.7. Persistence and Start-Up

The EEPROM holds a 13-byte state record (address 64; 27 light and 5 spare bits) and a 12-byte timing record (address 96), each with a magic word, a format version, and a CRC-8. After a state change, the firmware waits until 2 s pass without another change and then writes the record with `EEPROM.update()`, which programs only the bytes that differ [47]. At start-up, the firmware writes a low level to the output latch of every output pin before configuring it as an output, loads the timing record or the defaults, and restores the light states only from a state record whose magic word, version, counts, and CRC are valid; any other content leaves all outputs off and is not rewritten at boot. Restoring the last valid state is deliberate for an interruptible supply: lights that were on come back on when power returns. CRC validation rejects records that are detectably invalid, but an interrupted update can leave old or partial data, and its detection is not guaranteed.

3.8. Serial Protocol and Bridge

The Mega and the bridge exchange American Standard Code for Information Interchange (ASCII) frames `<body*HH>` at 19,200 Bd, 8N1 (Figure 5a). The body lists the protocol version (4), the message type, a transaction number, and the arguments, separated by vertical bars; `HH` is the CRC-8 of the body (polynomial `0x07`, initial value 0, no reflection). A start character always restarts collection, and the Mega accepts at most 120 characters. It validates framing, CRC, field format, version, handshake state, command, and identifier allowlist in that order. The 13 commands set lights, spares, and groups to absolute states, switch a group, a floor, or all lights off, read and change the gesture timing, and request states or diagnostics; there is no toggle command. The Mega answers each command with `ACK` (transaction number and command name) or `ERR` with one of 13 error codes, reports every actual change with `STATE`, and sends light and spare snapshots after the handshake and every 30 s. Until a `HELLO` arrives, all other commands are rejected; the wall buttons never wait for a handshake.

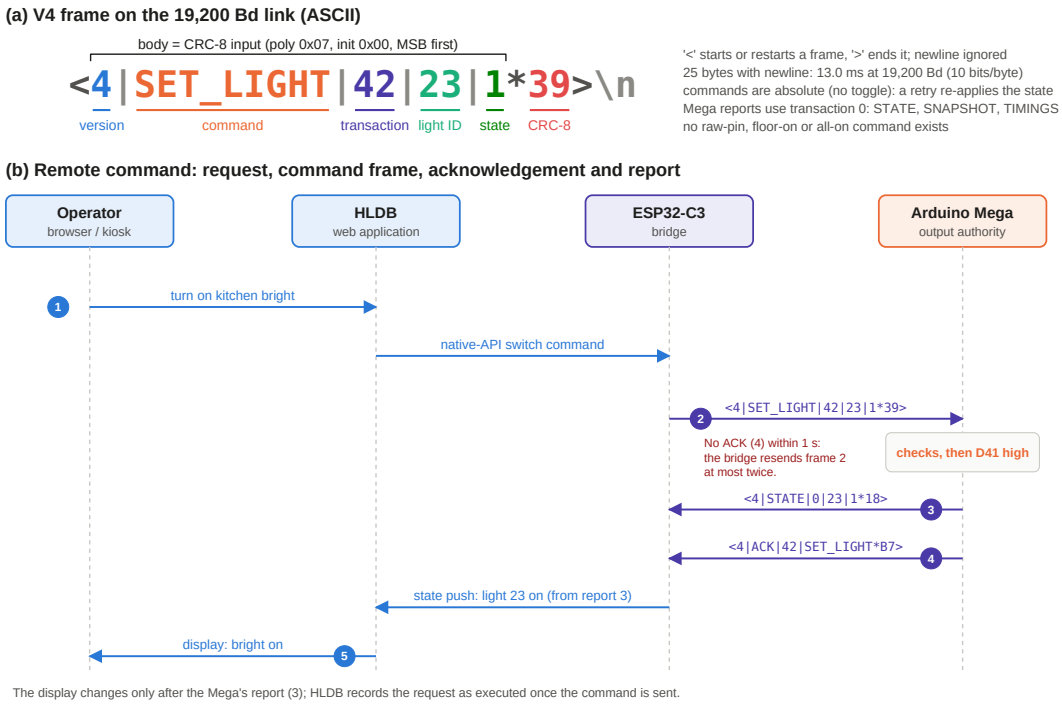

Figure 5. Serial protocol. (**a**) Structure of a version-4 frame. The asterisk (`*`) separates the frame body from its CRC-8 checksum. (**b**) Single-light command path in five numbered messages; the displayed state changes only after the Mega's report (3).

The bridge parses the Mega's frames with a fixed 160-character buffer and publishes only states the Mega has reported; until then, an entity is unknown. Commands leave through one script that allows a single transaction in flight, refuses requests before the handshake and during a 30 s guard after the bridge boots, and retransmits the same frame at most twice when no matching `ACK` arrives within 1 s (checked every 250 ms). The Mega does not de-duplicate transaction numbers, so a retransmission re-applies the same absolute state. If a wall action changes that circuit between a lost acknowledgment and the retransmission, within the 2–3 s retry window, the retransmission reasserts the remote state. The bridge declares the link lost after 45 s without a valid frame, sends `HELLO` every 5 s while incompatible, and requests diagnostics every 60 s. It exposes switches for rooms, circuits, and spares, buttons for the off, resynchronization, diagnostics, and timing actions, and number entities for dt, gt, and lpt.

3.9. HLDB Web Application and User Interfaces

HLDB is a lightweight custom web application, built with FastAPI, that supervises the residence's devices and records them in SQLite. Its ESPHome provider keeps one encrypted connection to the bridge through the `aioesphomeapi` client [48], maps each configured channel to an entity, and subscribes to state messages, so HLDB does not poll the controller. The lighting configuration exposes 14 rooms with 23 channels (reserves and spares are hidden from occupants), the all-lights-off action, and two floor-off actions through four representational state transfer (REST) endpoints. Each request, which names only a configured room, channel, or floor, passes an authentication and permission check and is written to a command audit with its target, action, outcome, and requesting user. Lighting control is a separate permission, so an account can display the lighting states without switching them, a moderated form of shared access that has also been proposed for guest users of domestic IoT systems [49]. The outcome `executed` means that the native-API command was sent without an exception; HLDB does not wait for the Mega's acknowledgment. Displayed channel

states come from controller reports, with one exception: after an accepted all-lights-off request, HLDB immediately marks every channel off, and subsequent reports override that mark. HLDB keeps no history of light states or button presses, because such a record would reveal when rooms are occupied [8,50].

Remote access. HLDB is not exposed to the Internet, and no port is forwarded on the home router. Occupants reach it from outside the residence through Tailscale, a mesh virtual private network built on the WireGuard protocol [51,52], which authenticates each enrolled device and encrypts the traffic between devices. Tailscale runs only on the HLDB host and does not route traffic to other devices on the home network, so the bridge is not reachable remotely. The remote path therefore depends on the residence's Internet connection and on Tailscale's coordination service, whereas the wall path depends on neither.

The Lights page of the dashboard (Figure 6a) groups room cards by floor, with an on/off control per channel that is disabled while the state is unknown, and a panel for the floor-off and all-lights-off actions. It reads HLDB's cached controller state every 5 s and immediately and 1.2 s after a command, so displays can briefly be stale; it is available in English, French, or Arabic. Two wall kiosks (1024 × 600 and 1080p), one for each floor, show ground-floor and first-floor lighting pages with touch chips per circuit and quick-off actions that execute only after a 1.2 s press and hold (Figure 6b). The gesture timing is not exposed in HLDB; it is tuned and saved from the commissioning page served by the bridge's embedded web server, so occupants cannot change the button behavior by accident.

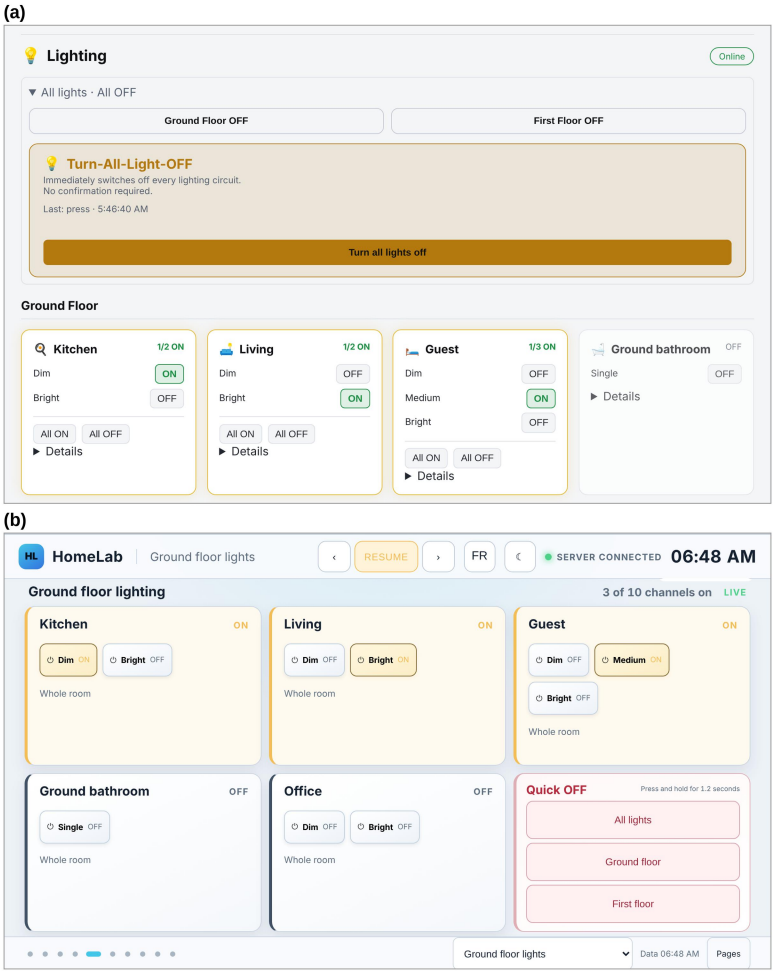

Figure 6. HLDB lighting interfaces with synthetic states. (**a**) Dashboard Lights page. (**b**) Ground-floor page of the wall kiosk.

3.10. Trust Boundaries and Threat Model

The CRC-8 of the serial frames provides integrity against accidental corruption only: it neither authenticates the sender nor prevents replay. The serial link therefore relies on the physical security of the panel and on the bridge being trusted. Table 2 summarizes the boundaries between the components.

Table 2. Trust boundaries of the lighting system.

Boundary	What Crosses It	Protection in Place	Residual Risk
Internet to HLDB host	Remote requests from phones	No forwarded port; Tailscale authenticates enrolled devices and encrypts traffic; it runs only on the HLDB host	A compromised enrolled device or account; dependence on the coordination service
User to HLDB	Lighting requests	Authentication, a separate lighting permission, allowlist of rooms, channels and floors, command audit	A compromised HLDB account or host can send any allowed request
HLDB to bridge	Native-API commands and state reports	Encrypted, key-authenticated ESPHome native API on the home network; the bridge is not reachable through Tailscale	Any device holding the API key on the home network
Bridge to Mega	Command and report frames	CRC-8 (integrity only), framing, version and handshake checks, absolute commands with identifier allowlists, no pin numbers	No authentication or replay protection: the bridge or the wiring can switch any light, replay commands or flood the link
Mega to outputs	Pin writes	One output manager; output latches low before configuration	Physical access to the panel

An attacker who controls the bridge can switch lights and change the gesture timing within its allowed ranges, but cannot change the button-to-circuit assignment, which is fixed in the firmware tables, and cannot disable the wall path. A flood of aggregate commands could, however, cause the reporting stalls predicted in Section 5.4 repeatedly, so the interference identified in this study is also a denial-of-service consideration for the wall path. Message authentication on the serial link, for example, a truncated message authentication code with a sequence number, is a possible extension; no penetration test was performed.

4. Evaluation Methods

Software-in-the-loop harness. The Mega firmware was compiled without modification for a Linux host against a minimal replacement of the Arduino functions it uses (digital input and output, the millisecond counter, the serial port and the EEPROM). A test script sets button levels at chosen times, sends bytes to the serial receiver, simulates a supply interruption by restarting the firmware from the retained EEPROM contents, and records every output change, transmitted frame, and EEPROM write. Time advances in 1 ms steps with one `loop()` call per step, so measured delays include a sampling step of 1 ms. This idealized harness does not model AVR execution time, interrupts, the blocking of a full serial transmit buffer, or EEPROM programming time; a timed variant adds the last two and is used for the fourth question below. Neither variant models electrical behavior, and the bridge firmware is not executed, so the results describe the controller logic rather than measured button-to-light latency. Table 3 summarizes the scope of both variants.

Questions. The tests address four questions, each answered in one subsection of Section 5:

1. Are button gestures recognized correctly, and with what delay? (Section 5.1)
2. How are lighting states saved and restored after a simulated interruption? (Section 5.2)

3. How does the controller handle corrupted commands and competing local and remote actions? (Section 5.3)
4. Can remote processing delay or disrupt wall-button operation? (Section 5.4)

Table 3. Scope of the software-in-the-loop harness.

Aspect	Idealized Variant	Timed Variant
Unmodified firmware logic	Executed	Executed
Millisecond counter, digital inputs and outputs (1 ms steps)	Modeled	Modeled
Serial bytes and EEPROM contents	Modeled	Modeled
Transmit-buffer blocking (64 bytes; 0.521 ms per byte)	Not modeled	Modeled
EEPROM programming time (3.3 ms per programmed byte)	Not modeled	Modeled
AVR execution time, interrupts and their latency	Not modeled	Not modeled
Electrical behavior, bridge firmware and network	Not modeled	Not modeled

The test conditions are given with each result. The scripts, random seeds, raw results, firmware digest, and reproduction commands are provided in Supplement S1, from which every reported number can be regenerated.

Checks on the installation. During development and daily use, the supply was repeatedly interrupted, and single and multiple clicks were exercised on the installation; the wall buttons were also checked with the bridge disconnected and, separately, with HLDB shut down. These checks were qualitative and uninstrumented: they confirm the behavior at human scale, whereas the harness quantifies it with millisecond precision.

Bench replica. To check the electrical interface and the timing on hardware, a low-voltage replica of the controller was built from an Arduino Mega 2560, an ESP32-C3 SuperMini running the bridge configuration and one of the panel's SSR boards, wired as in the installation and supplied at 5 V from a bench supply, without mains loads. Currents were read from the supply display and from a bench multimeter in series with the Mega's supply lead or with one SSR input. Signal levels and timing were recorded with a two-channel oscilloscope on the serial lines and on a button input and its output, and presses were applied with a jumper wire. Repeated attempts to capture a press during an all-lights-off command did not succeed, so the stall of Section 5.4 remains a result of the timed model.

Remote-request audit. The remote requests were counted from HLDB's command audit in a read-only copy of the HLDB database for the 30-day window from 10 August 2026 00:00 to 9 September 2026 00:00 (UTC+03:00). Only aggregated counts are reported.

Versions. Supplement S1 provides the current firmware of the Mega and of the bridge; the Mega sketch executed in the harness was in service throughout the audit window.

Artificial intelligence (AI) tools assisted in writing the simulation (see Acknowledgments); every reported number was regenerated from these scripts and checked against the result files in Supplement S1.

5. Results

5.1. Are Button Gestures Recognized Correctly, and with What Delay?

What was tested. Single, double, and triple clicks and long presses at the default, minimum, and maximum timing settings; sweeps of the press duration and of the interval between two clicks; and synthetic contact bounce, in which every press and release edge bounced three times, with the time between successive bounce transitions drawn at random up to a maximum dwell time (200 single and 200 double clicks per condition).

What happened. Single and double clicks acted dt + gt after the final release and long presses dt + lpt after the press, plus the 1 ms sampling step of the harness: 401 ms and 1351 ms with the defaults, for which dt + gt = 400 ms. Across the allowed settings, the single-click delay ranged from 221 to 751 ms, and a triple click acted dt after its final release. The classification boundaries followed the parameters exactly (Figure 7a,b): a press of 50 ms was ignored and one of 51 ms counted as a click, two clicks separated by less than 350 ms formed a double click, and a press held for 1351 ms or longer was a long press. Under synthetic bounce, all clicks were classified correctly for maximum dwell times up to 25 ms with dt = 50 ms and up to 20 ms with dt = 20 ms (Figure 7c). Errors appear when bouncing leaves less than dt of stable contact or when an unstable interval longer than dt is taken as an extra click, so a long debounce time has its own cost: with dt = 150 ms, errors began at a maximum dwell time of 10 ms.

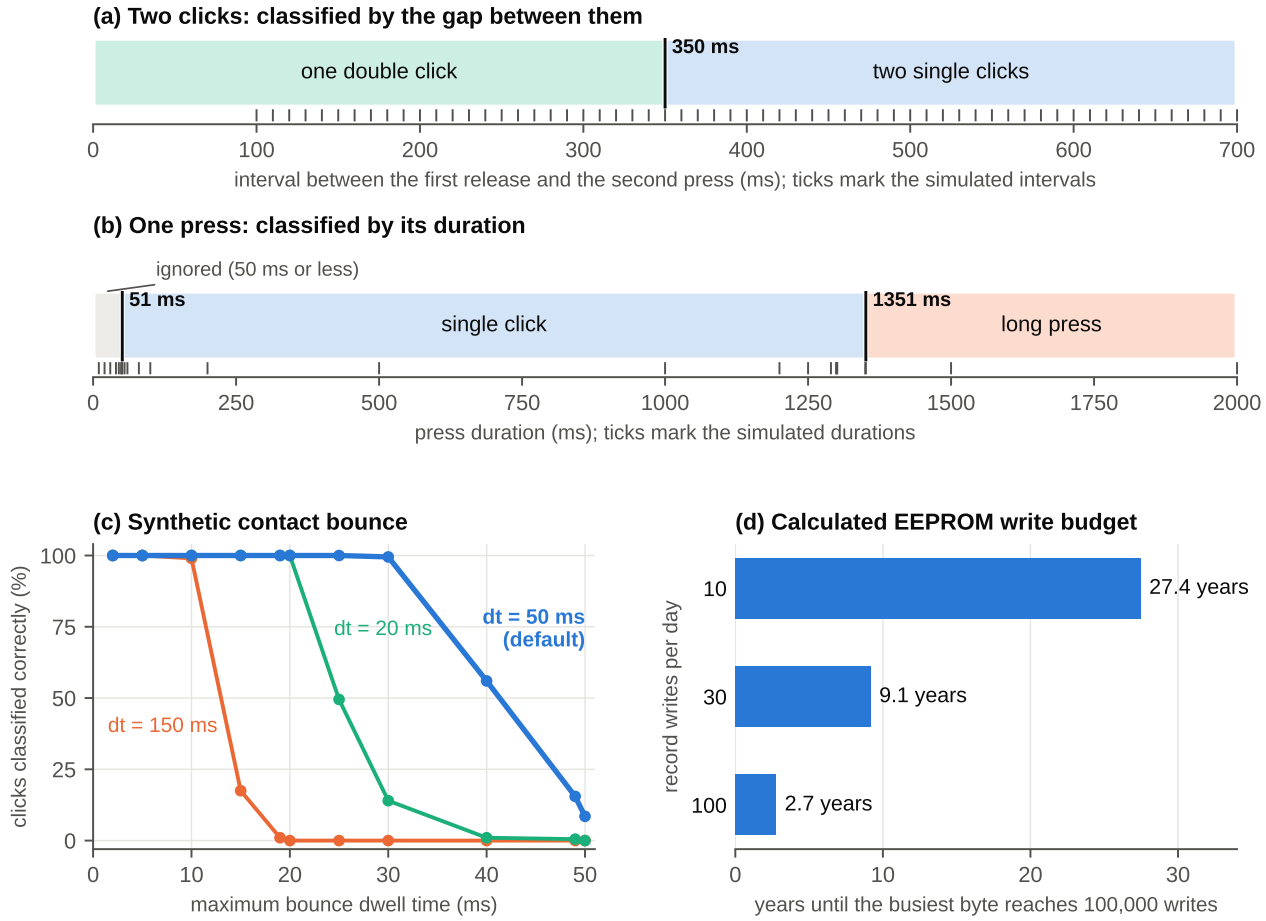

Figure 7. SIL results with default timing. (**a**) Double-click classification against the interval between clicks. (**b**) Press classification against press duration. (**c**) Accuracy under synthetic contact bounce (full sweep in Supplement S1). (**d**) Estimated EEPROM service life at assumed write rates [43].

Why it matters. Gesture classification is the core function of the wall interface. The delay of about 0.4 s on every single and double click is the measurable cost of letting one button serve up to three circuits, and the bounce results show that dt must stay well below the shortest press occupants make. These results agree with the observations on the installation, where clicks and multiple clicks behaved as intended, and with the 398 ms measured on the bench replica (Section 5.6). With the default values, the occupants reported no difficulties and accepted the delay as the cost of selecting several circuits from one button.

5.2. How Are Lighting States Saved and Restored After a Simulated Interruption?

What was tested. How often the EEPROM is written when lights change; what is restored when the supply is interrupted at increasing times after a change, simulated by restarting the firmware from the retained EEPROM contents; and start-up from five memory contents, one valid and four invalid (erased memory, a corrupted CRC, an older record format, and data of the previous firmware generation).

What happened. Ten changes within 2 s produced a single write. A change was lost when the interruption came within 2 s of it and restored when it came later. Only the valid record restored the lights; the four invalid contents started with all outputs off, and nothing was written to the EEPROM at boot. No output pin was written high while it was still configured as an input. The CRC byte of the record changed with 399 of 400 writes, so it wears first: at the specified 100,000 write cycles [43], it would last about 27, 9 and 3 years at assumed rates of 10, 30 and 100 writes per day (Figure 7d). These are scenario values rather than a prediction for this household: one write can cover several light changes, and the actual write rate is unknown because wall actions are not recorded.

Why it matters. For an interruptible supply, the firmware trades persistence latency against write frequency. The 2 s delay used in the reported implementation serves as a baseline operating value rather than a fixed design constraint. Because the record is saved only after 2 s without a further change, continued activity postpones the save, so changes made before the last 2 s can also be lost when the supply is interrupted; an invalid record never switches lights on at start-up. The tests used an idealized EEPROM model and did not include an interruption during a record update. The service-life figures are estimates under assumed write rates, not measurements, and the start-up result concerns the software order of output writes, not electrical start-up or brown-out behavior; Section 5.4 shows a case in which the 2 s wait is skipped. On the installation, repeated supply interruptions showed the expected restoration at human scale.

5.3. How Does the Controller Handle Corrupted Commands and Competing Local and Remote Actions?

What was tested. One valid `SET_LIGHT` command frame was corrupted in about 99,000 ways after a valid handshake: every single-bit and double-bit error, 20,000 random triple-bit errors, and 20,000 frames at each of four bit-error rates. Separate scenarios pressed the two kitchen doorway buttons in turn without any handshake, repeated an absolute remote command, and retransmitted a command after an intervening wall toggle.

What happened. None of the corrupted frames produced an acknowledgment or an output change: the CRC rejected 45.6% of them, the field, version, command, and identifier checks 21.6%, and the framing checks 32.8%. They included the 25 double-bit error patterns that the 8-bit CRC cannot detect in this frame [40], all of which were rejected by the later checks. The doorway buttons switched their shared circuit on and then off, a repeated absolute command produced no further transition, and a retransmission after a wall toggle reasserted the remote state.

Why it matters. The command path rejected every tested corruption of the tested frame, which supports the output-authority requirement for this frame type. The result applies to the tested `SET_LIGHT` frame and to the injected error patterns; other frame types and lengths, burst errors, and deliberate manipulation were not tested, and the CRC provides integrity rather than authentication (Section 3.10).

5.4. Can Remote Processing Delay or Disrupt Wall-Button Operation?

What was tested. In the timed variant, single and double clicks were started 251 times around a remote command, from 150 ms before to 350 ms after it, for four kinds of remote traffic of increasing size (Table 4).

What happened. While the Mega sends serial reports, it waits whenever its 64-byte transmit buffer is full [53], and it does not read its buttons while waiting. A single command (44 bytes) fit in the buffer and caused no stall. In the timed SIL model, an all-lights-off with 27 lights on (561 bytes) stalled the loop for 272 ms, comprising 258.3 ms of waiting for buffer space, as 496 of the 561 bytes could not be buffered at 0.521 ms per byte, and 13.2 ms of EEPROM programming caused by the premature save described below: presses that fell within the stall were lost, clicks overlapping it were recognized up to 270 ms late, and double clicks whose second click fell within it switched the single-click circuit instead. The stall also exposed a timing fault: after it, button edges carried the time at which the loop pass began, so releases appeared up to 38 ms early, and the 2 s save delay was skipped. The full sweep in Supplement S1 shows that shorter presses are more exposed: during a diagnostics request, 5 of 251 single presses of 80 ms were lost, and during a floor-off, 6 of 251 presses of 200 ms.

Table 4. Wall clicks started at 251 times around a remote command in the timed harness (120 ms presses; double clicks 150 ms apart; other durations in Supplement S1). The stall is split into waiting for the serial transmit buffer and EEPROM programming.

Remote Traffic (Bytes Sent by the Mega)	Stall (ms)	UART Wait (ms)	EEPROM Wait (ms)	Single Clicks Lost	Double Clicks Read as Single
One `SET_LIGHT` command (44)	0	0	0	0	0
Diagnostics request (138)	38	38.0	0	0	0
Floor-off, 17 lights on (356)	162	151.6	9.9	46	67
All-lights-off, 27 lights on (561)	272	258.3	13.2	101	151

Why it matters. Independence from the network does not by itself make the wall responsive, because remote traffic shares the controller's loop. The timed SIL model predicts vulnerable intervals of 38, 162, and 272 ms for the three larger messages, and because the bridge requests diagnostics automatically every 60 s (Section 3.8), short windows of this kind could recur without any dashboard use. How often a press falls into such a window in household use is unknown, because wall actions were not recorded; the known trigger frequencies are one diagnostics request per minute and, in the 30-day audit, 36 all-lights-off and two floor-off requests. A window of a few tenths of a second is short on the human scale: on the installation, the wall buttons operated with the bridge disconnected and with HLDB shut down, and no missed press was noticed during dashboard all-off commands, but such uninstrumented checks cannot exclude occasional losses, and repeated attempts to capture a press during an all-lights-off command on the bench replica did not succeed (Section 5.6). The 272 ms value is the maximum stall predicted by the timed model under the tested conditions. Its two components call for separate corrections (Section 6); the evaluated firmware is unchanged in this study. Dashboard commands took about one second to appear, consistent with the page refresh (Section 3.9).

5.5. Remote-Request Audit

The audit describes the remote use of one household over 30 days and is not representative of residential users in general. HLDB recorded 133 remote lighting requests on 27 of the

30 days, a median of three per day and at most 24 (Figure 8). Ninety-five addressed a single channel (73 off, 22 on), 36 requested all lights off, and two requested a floor off. Overall, 111 requests (83.5%) asked to switch lights off. Among the channel requests, 47 addressed a dim circuit, 31 a bright circuit, 15 a single-circuit room, two the medium guest circuit, and 86 of all requests (64.7%) were made between 18:00 and midnight.

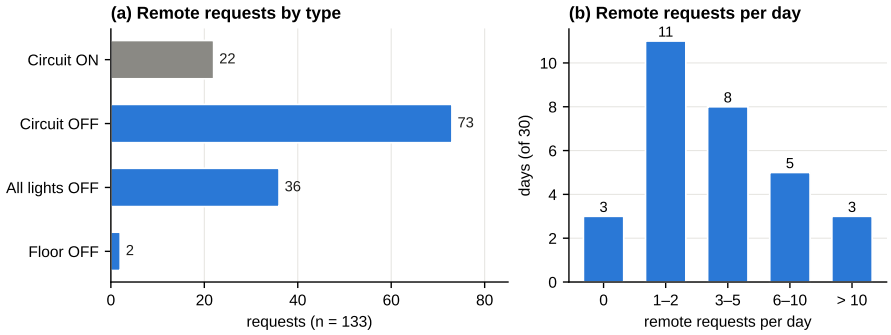

Figure 8. Remote lighting requests recorded by HLDB over 30 days. (**a**) Requests by type (off requests in blue). (**b**) Requests per day.

HLDB recorded 116 requests as dispatched to the bridge without an exception and 17 as errors because the native-API session was unavailable: 12 single-channel off requests, three all-lights-off and two floor-off requests, in four episodes while the bridge was offline. Neither outcome confirms controller acknowledgment or lamp actuation. Requests repeated by an occupant are counted individually, as in one error episode with nine requests within about nine minutes, whereas retransmissions by the bridge occur below HLDB and are not logged. All 133 requests are retained in the analysis of requested actions.

Occupants reported that the remote control was rarely used to turn individual lights on during normal occupancy, because the physical wall buttons were more convenient for that purpose. The remote control was instead used mainly for shutdown actions, particularly the all-off command around bedtime, when reaching a phone was often more convenient than checking multiple rooms individually, and when away from the residence if occupants were uncertain whether lights had been left on. This reported usage pattern is consistent with the command log, in which off requests strongly outnumbered on requests and remote activity was concentrated during the evening.

5.6. Bench Measurements on a Hardware Replica

What was tested. Supply and SSR input currents, serial-link levels, and the button-to-output delay on the replica described in Section 4 (Figure 9).

What happened. With all channels off, one on, and eight on, the 5 V supply delivered 0.08, 0.09, and 0.15 A, whereas the current in the Mega's own supply lead stayed between 62 and 64 mA. The current into one SSR input was 0.42 mA when on (0.005 mA when off) and 0.41 mA with eight channels on. On the serial link, the ESP32-C3 drove the Mega's receive pin to 3.32–3.36 V high and 0.04–0.16 V low; the divider delivered 3.32 V to the ESP32-C3, and frames were exchanged without errors both with the divider and with a direct connection. The Mega began its reply less than 0.5 ms after the end of a command frame. A single click switched the output 398 ms after the release (cursor measurement at 50 ms/div; three further trials gave about 0.4 s), and the contact bounce produced by the jumper wire caused a single output transition.

Why it matters. The measurements confirm on hardware the idealized timing of Section 5.1, for which the parameters set 400 ms and the harness gave 401 ms, and they settle the electrical questions of Section 3.4: the SSR inputs load the Mega's pins far below their

limits, and the installed serial link works with a measured but not guaranteed margin. They concern one sample of each board and the low-voltage side only; the stall of Section 5.4, the remote command-to-output latency at the SSR, and the mains-side quantities were not measured.

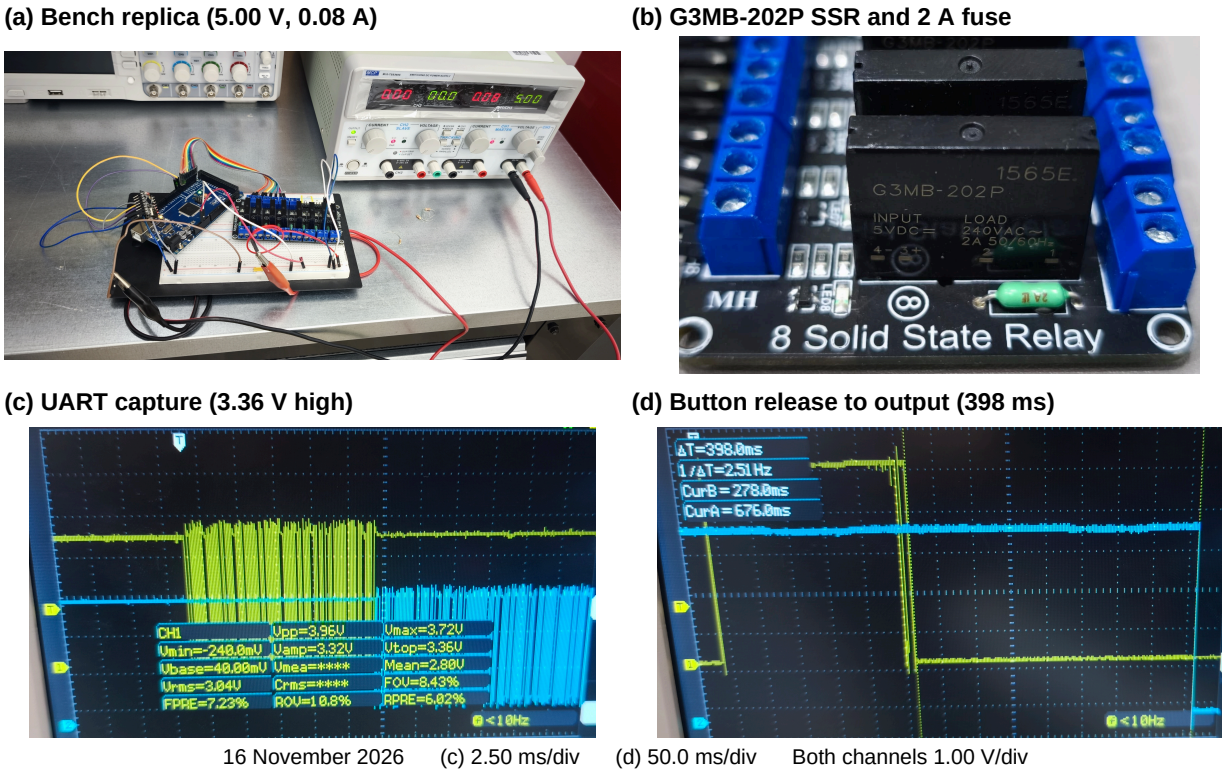

Figure 9. Bench measurements on the low-voltage replica. (**a**) Setup with one SSR board (supply 5.00 V, 0.08 A, all channels off). (**b**) G3MB-202P SSR and 2 A output fuse. (**c**) Command frame at the Mega's receive pin (yellow; high level 3.36 V) and the Mega's reply through the divider (blue). (**d**) Button release (yellow) to output transition (blue): 398 ms. Asterisks (****) in panel (**c**) are instrument readouts without a numeric value and were not used in the analysis.

6. Discussion

6.1. Position and Design Lessons

The present work does not add a sensing or energy-optimization method; it specifies the layer beneath them. Compared with the nearest predecessors [18,19], it adds multi-circuit gestures, shared doorway state, absolute allow-listed commands and defined start-up behavior, and evaluates them on the unmodified firmware rather than by functional demonstration alone. Home Assistant [34] supervised this installation before HLDB (Section 3.2); because the lighting logic stays in the panel, the supervisory layer could be replaced, and HLDB can be unavailable, without losing any wall function.

These findings suggest five design lessons. First, a shared controller state avoids independent toggle histories for doorway buttons and remote requests, but it gives the wall no priority, so a manual-override policy would need explicit arbitration and feedback, for example, a transaction cache that ignores retransmissions after a wall action. Second, the multi-gesture interface came from practice: the early binary scheme confused the household, whereas selecting circuits by the number of clicks was readily adopted (Section 3.2). Its delay of about 0.4 s per single click, confirmed on hardware, ensures that a double click never passes through the single-click circuit; one-circuit buttons could act on release. The household-tuned thresholds suit these occupants only and, as button timing affects perceived usability [37], warrant formal evaluation with users of different motor abilities.

Third, network independence does not guarantee responsiveness: wall operation never waits for the network, but remote traffic shares the controller's loop, so output reporting must be bounded or non-blocking (Section 5.4). The two components of the modeled stall call for separate corrections: reporting an aggregate change as one snapshot frame, or queueing reports without blocking, removes the wait for the transmit buffer, and refreshing the button timestamp after serial processing prevents the premature save and its EEPROM wait. These corrections are recommendations for future firmware; the installed controller runs the evaluated version. Fourth, in this household, remote use was concentrated on switching lights off, often all at once, and aggregate actions moved from the wall to the screens: the early wall all-off was abandoned, whereas 36 all-off requests were sent remotely in 30 days. Fifth, central wiring turns the button-to-circuit mapping into software, so buttons can be reassigned without rewiring.

6.2. Comparison with Commercial Retrofits

Commercial smart lighting is usually retrofitted device by device: a smart switch replaces each wall switch, or a smart bulb replaces each lamp, and each device joins a hub or a vendor cloud. The present design centralizes the decision in the panel instead; Table 5 compares the approaches for a room with three circuits.

Table 5. Qualitative comparison of lighting-control approaches for a room with three circuits.

	Conventional Switches	Smart Switches	Smart Bulbs	This System
Wall devices	Three switches	One to three smart switches	Existing switches, left on	One pushbutton
Voltage at the wall	Mains	Mains	Mains	5 V DC
Operation without network	Full	Local on/off, where supported	On/off at the switch only	Full, all gestures
Changing a button's function	Rewiring	Vendor application	Vendor application	Firmware update
Whole-house functions	None	Vendor hub or cloud	Vendor hub or cloud	Supervisory layer
Wiring	Existing	Existing; neutral often required	Existing	Every button and circuit to the panel

At the wall, one pushbutton replaces up to three switches, so that 19 buttons operate 27 circuits, and only 5 V DC control signals reach the wall plates; mains switching is confined to the panel. The controller components cost about USD 79 (Appendix A), that is, USD 2.5 per output channel or USD 3.4 per loaded circuit, including the supply and the breaker but excluding the pushbuttons, terminals, enclosure, wiring and installation. A Wi-Fi smart switch costs about USD 7 per circuit for the device alone (author-recorded local price, October 2026), also without installation. Both figures exclude wiring and buttons, but they are not a like-for-like comparison: a smart switch is a much easier retrofit, whereas the central design requires every button and circuit to be wired to the panel, which suits new construction and renovation better than retrofits.

Because the Mega accepts absolute, allow-listed commands for lights, groups, and floors, whole-house functions are composed at the supervisory layer without firmware changes. Master off and floor off are native firmware commands exposed by HLDB; scene control in HLDB is under test, and a page on which a local AI model learns patterns of use and suggests scenes is under development. Through ESPHome, the bridge also integrates natively with Home Assistant, which supervised the installation before HLDB (Section 3.2) and provides scenes and automations. HLDB runs here on an old Windows laptop and

can also be hosted on a Raspberry Pi. In the timed SIL model, supervisory commands that switch many lights at once produce the stalls of Section 5.4 unless they are sent light by light or the reporting is made non-blocking.

The audit also suggests what a retrofit most needs. Remote use was dominated by shutdown requests: 111 of the 133 requests (83.5%) asked to switch lights off, 38 of them for all lights or a floor, and in this household the occupants used remote control mainly around bedtime and when away from the residence (Section 5.5). Where a whole-house off is the main need, a single smart relay, contactor, or remotely operated breaker on the supply that feeds the lighting circuits, an established retrofit option that needs no control wiring [41] (p. 17.6), would provide it with much less wiring and at lower cost than per-circuit control, and without replacing any wall switch. It would not reproduce the single-circuit off requests or remote switching on, and while it is open, the wall switches are also without supply, so a light needed at night must first be re-enabled remotely. It also requires the lighting to be fed from dedicated breakers and a relay rated for the total lighting load, including the inrush current of LED drivers [41] (p. 16.11).

6.3. Limitations and Future Work

The evaluation concerns one residential installation and a 30-day remote-request audit. Household feedback informed the gesture mappings and timing parameters, but these observations were informal and do not establish usability across different occupants, visitors, or users with different motor abilities. Double- and triple-click functions may also be difficult to discover without explanation. The audit characterizes remote requests rather than overall lighting use: wall actions were not recorded, and dispatch by HLDB does not confirm controller acknowledgment or lamp actuation. Consequently, the available data cannot establish end-to-end reliability, energy savings, or general preferences for local and remote control, and the audit results are not generalized beyond this household.

The SIL experiments characterize the implemented firmware under specified input sequences and timing assumptions. The timed variant includes serial-transmission and EEPROM programming delays, but neither variant reproduces the complete physical system or executes the bridge firmware. The reported missed gestures are therefore simulation results: the bench replica confirmed the idle single-click timing (Section 5.6) but did not reproduce the stall. The model omits AVR execution time and interrupt latency (Table 3). Similarly, rejection of the tested corruptions of one representative command frame does not establish detection of every possible communication error or protection against malicious commands. Security was not independently evaluated, and no penetration test was performed (Section 3.10).

The electrical measurements concern one sample of each board on the low-voltage side. The installed serial link works with a measured margin that the datasheets do not guarantee, which the reference design (revision C) addresses. SSR temperature, LED-driver inrush, SSR leakage, standby power, and reset and brown-out behavior were not measured, and the thermal margin of the largest circuit depends on the unmeasured panel temperature. The centralized arrangement also requires dedicated wiring from buttons and lighting circuits to the panel, making it more suitable for new construction or major renovation than straightforward retrofits. The system provides separately switched lighting levels rather than continuous dimming.

Future work should first implement the stall corrections of Section 6 and test them on hardware with a synchronized capture of button presses during aggregate commands, and measure the remote command-to-output latency, the panel temperature, and the LED-driver inrush. A usability study involving participants beyond the installation's regular occupants should assess gesture discovery, timing, errors, and perceived control. Additional

supervisory functions, including scenes and occupancy- or daylight-based automation, could subsequently be evaluated while preserving independent wall operation; device-free occupancy sensing on ESP32-class hardware, already demonstrated for managing the lighting and heating, ventilation, and air-conditioning (HVAC) energy of offices [13], would avoid camera-based monitoring. Any feature that learns from household activity would require an explicit decision about data collection and retention, consistent with the system's privacy objectives.

7. Conclusions

This study presented the design, implementation, and analysis of a low-cost residential lighting controller that combines multi-gesture wall operation with replaceable network supervision. Its principal finding is that remote traffic sharing the controller's loop can interfere with wall-button processing even when wall operation is independent of the network. An Arduino Mega 2560 processes 19 wall pushbuttons and controls 32 SSR outputs, including 23 loaded lighting circuits. The pushbuttons operate at 5 V DC, with mains switching confined to the panel. An ESPHome bridge supports Home Assistant integration and connects the controller to HLDB for remote requests and state display. Controller components cost approximately USD 80, including the power supply but excluding buttons, terminals, enclosure, wiring, and installation. Functional checks demonstrated wall operation with the bridge disconnected or HLDB unavailable.

Evaluation of the unmodified firmware showed both the benefits and limitations of sharing one controller between local and remote functions. A common circuit state coordinates doorway buttons and remote requests, although retransmissions can supersede intervening wall actions. Under idealized timing, single and double clicks acted 401 ms after the final release, and the bench replica measured 398 ms. In the timed simulation, an all-lights-off operation interrupted button sampling for 272 ms, causing missed or misclassified gestures; stale timestamps also bypassed the intended persistence delay. After a simulated supply interruption, the controller restored the last completed and validated state record. Because the record is saved only after 2 s without a further change, continued activity postpones the save, so changes made before the last 2 s can also be lost; these tests used an idealized EEPROM model and did not include an interruption during a record update. No corrupted variant of the tested command frame was accepted, a result that applies to the tested frame and error patterns. On the bench, the SSR inputs drew 0.42 mA per channel, and the serial link worked with a measured margin, for which a buffered reference design is provided. These findings demonstrate that network independence alone does not ensure local responsiveness.

The 30-day audit of one household recorded 133 remote requests, 83.5% of which requested switching lights off. These observations describe supervisory use in this household rather than confirmed lamp operation. The documented implementation and accompanying evaluation materials provide a basis for reproducing the control logic and investigating its behavior. The principal design lesson is that independent wall control also requires careful management of reporting delays, timestamps, and competing commands within the controller.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/iot7040105/s1, Supplement S1: Mega firmware, software-in-the-loop harness, experiment scripts and results, protocol specification, bridge configuration, HLDB provider source, KiCad schematics of the installed panel (revision B) and of the reference design (revision C), bench measurements and photographs, file manifest with checksums and aggregated audit counts.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. No participants were recruited, and no intervention was made; the study reports only aggregated operational counts from the automation logs of the residence's occupants, without user identities or individual timestamps.

Informed Consent Statement: Not applicable.

Data Availability Statement: The firmware, harness, scripts, results, schematics, bench measurements, and aggregated audit counts are provided in Supplement S1. The household database from which the counts were derived is not publicly available because it describes the daily routines of a private household.

Acknowledgments: Generative AI tools were used at several stages of this work, including Claude Opus 5.5 (Anthropic), GPT-6.1 (OpenAI), and Gemini 3.8 (Google), primarily to support coding, manuscript refinement, and preparation of the reproducibility and evidence package. All AI-assisted outputs were reviewed and verified by the author. The author takes full responsibility for the accuracy, integrity, and content of this work.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AC	Alternating current	LED	Light-emitting diode
ACK	Acknowledgment	MQTT	Message Queuing Telemetry Transport
AI	Artificial intelligence	PLC	Programmable logic controller
API	Application programming interface	RAM	Random-access memory
ASCII	American Standard Code for Information Interchange	REST	Representational state transfer
CRC	Cyclic redundancy check	RTU	Remote terminal unit
DALI	Digital Addressable Lighting Interface	SHA	Secure Hash Algorithm
DC	Direct current	SIL	Software-in-the-loop
EEPROM	Electrically erasable programmable read-only memory	SSR	Solid-state relay
HLDB	HomeLab Dashboard	TTL	Transistor–transistor logic
HVAC	Heating, ventilation and air-conditioning	UART	Universal asynchronous receiver–transmitter
IoT	Internet of Things	UML	Unified Modeling Language

Appendix A. Implementation Details

Table A1 lists the controller pin allocation. Table A2 lists the bill of materials.

Table A1. Arduino Mega 2560 pin allocation (button-to-output map in Supplement S1).

Pins	Direction	Function
D0 (RX0), D1 (TX0)	UART	Serial link to the ESP32-C3 bridge, 19,200 Bd
D2–D12, D46–D53	Input	19 wall pushbuttons, external 10 kΩ pull-downs, pressed = high
D14–D20, D22–D26, D30–D34, D36–D45	Output	27 lighting circuits (23 loaded, 4 reserves D24, D25, D33, D34), light identifiers 1–27
D21, D27, D28, D29, D35	Output	5 spare channels, spare identifiers 1–5
D13, A0–A15	–	Unused

Table A2. Bill of materials of the controller (USD, author-recorded prices, October 2026; buttons, enclosure and wiring excluded). Bold indicates the total component cost.

Designator	Component	Qty	Cost
U1	Arduino Mega 2560-compatible board	1	12.00
U2	ESP32-C3 Mini board	1	2.00
SSR1–SSR4	8-channel SSR board, G3MB-202P, 5 V active high, 2 A per channel, 2 A fuse per channel	4	48.00
F1	10 A AC miniature circuit breaker, 1 pole	1	6.00
R1–R21, C1	Resistors (10 kΩ, 20 kΩ) and 100 µF capacitor, allowance	1 lot	1.00
PS1	220 V AC/5 V DC 5 A supply, shared with the power controller	1	10.00
	Lighting controller total About USD 80 to run up to 32 lighting circuits, i.e., about USD 2.5 per output channel or USD 3.4 per loaded circuit		**79.00**

References

1. Day, J.K.; McIlvennie, C.; Brackley, C.; Tarantini, M.; Piselli, C.; Hahn, J.; O'Brien, W.; Rajus, V.S.; De Simone, M.; Kjærgaard, M.B.; et al. A review of select human-building interfaces and their relationship to human behavior, energy use and occupant comfort. *Build. Environ.* **2020**, *178*, 106920. [CrossRef]
2. Sadeghi, S.A.; Karava, P.; Konstantzos, I.; Tzempelikos, A. Occupant interactions with shading and lighting systems using different control interfaces: A pilot field study. *Build. Environ.* **2016**, *97*, 177–195. [CrossRef]
3. Ahmadi-Karvigh, S.; Ghahramani, A.; Becerik-Gerber, B.; Soibelman, L. One size does not fit all: Understanding user preferences for building automation systems. *Energy Build.* **2017**, *145*, 163–173. [CrossRef]
4. Hu, Y.; Wen, W.; Yan, W.; Ding, R. Analysis of Smart Wall Switch without Neutral Wire Compatibility Issue. In Proceedings of the 2019 16th China International Forum on Solid State Lighting & 2019 International Forum on Wide Bandgap Semiconductors China (SSLChina: IFWS), Shenzhen, China, 25–27 November 2019; pp. 269–272. [CrossRef]
5. Noura, M.; Atiquzzaman, M.; Gaedke, M. Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mob. Netw. Appl.* **2019**, *24*, 796–809. [CrossRef]
6. Holguin, I.; Errapotu, S.M. Smart Home IoT Communication Protocols and Advances in their Security and Interoperability. In Proceedings of the 2023 7th Cyber Security in Networking Conference (CSNet), Montréal, QC, Canada, 16–18 October 2023; pp. 208–211. [CrossRef]
7. Hussein, N.; Nhlabatsi, A. Living in the Dark: MQTT-Based Exploitation of IoT Security Vulnerabilities in ZigBee Networks for Smart Lighting Control. *IoT* **2022**, *3*, 450–472. [CrossRef]
8. Acar, A.; Fereidooni, H.; Abera, T.; Sikder, A.K.; Miettinen, M.; Aksu, H.; Conti, M.; Sadeghi, A.R.; Uluagac, S. Peek-a-boo: I see your smart home activities, even encrypted! In Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Linz, Austria, 8–10 July 2020; pp. 207–218. [CrossRef]
9. Wagiman, K.R.; Abdullah, M.N.; Hassan, M.Y.; Mohammad Radzi, N.H.; Abu Bakar, A.H.; Kwang, T.C. Lighting system control techniques in commercial buildings: Current trends and future directions. *J. Build. Eng.* **2020**, *31*, 101342. [CrossRef]
10. Chew, I.; Karunatilaka, D.; Tan, C.P.; Kalavally, V. Smart lighting: The way forward? Reviewing the past to shape the future. *Energy Build.* **2017**, *149*, 180–191. [CrossRef]
11. Putrada, A.G.; Abdurohman, M.; Perdana, D.; Nuha, H.H. Machine Learning Methods in Smart Lighting Toward Achieving User Comfort: A Survey. *IEEE Access* **2022**, *10*, 45137–45178. [CrossRef]
12. Chen, J.; An, J.; Yan, D.; Zhou, X. Future trends in intelligent lighting control systems: Integrated technologies, multi-system linkage and AI integration. *Build. Simul.* **2024**, *17*, 1909–1932. [CrossRef]
13. Chen, Y.; Sumiyoshi, D.; Wang, X.; Yamamoto, T.; Ueno, T.; Oh, J. A Foundational Edge-AI Sensing Framework for Occupancy-Driven Energy Management in SMOs. *IoT* **2026**, *7*, 25. [CrossRef]
14. Soheilian, M.; Fischl, G.; Aries, M. Smart Lighting Application for Energy Saving and User Well-Being in the Residential Environment. *Sustainability* **2021**, *13*, 6198. [CrossRef]
15. Abi Ghanem, D. Energy, the city and everyday life: Living with power outages in post-war Lebanon. *Energy Res. Soc. Sci.* **2018**, *36*, 36–43. [CrossRef]
16. Dagher, L.; Jamali, I.; Abi Younes, O. Extreme energy poverty: The aftermath of Lebanon's economic collapse. *Energy Policy* **2023**, *183*, 113783. [CrossRef]
17. Ginzarly, R.; Nachabe, L.; Nohra, C.; Hussein, A.; Malas, H.; Khawla, M. An Automatic Transfer Switch (ATS) System Based on Cost Effectiveness - Lebanese Case Study. In Proceedings of the 2024 International Conference on Smart Systems and Power Management (IC2SPM), Beirut, Lebanon, 28–29 November 2024; pp. 74–79. [CrossRef]

18. Sowah, R.A.; Boahene, D.E.; Owoh, D.C.; Addo, R.; Mills, G.A.; Owusu-Banahene, W.; Buah, G.; Sarkodie-Mensah, B. Design of a Secure Wireless Home Automation System with an Open Home Automation Bus (OpenHAB 2) Framework. *J. Sens.* **2020**, *2020*, 8868602. [CrossRef]
19. Andramuño, J.; Mendoza, E.; Núñez, J.; Liger, E. Intelligent distributed module for local control of lighting and electrical outlets in a home. *J. Physics Conf. Ser.* **2021**, *1730*, 012001. [CrossRef]
20. Villegas, M.D.; Paredes, E.D.; Arévalo, J.A.; Quito Carrión, A.; Pillajo, R.; Cuenca Sánchez, A.; Proaño, P. Smart Automation for Residential Spaces with PLC-ESP32 Architecture. *Eng. Proc.* **2025**, *115*, 7. [CrossRef]
21. de Bakker, C.; Aries, M.; Kort, H.; Rosemann, A. Occupancy-based lighting control in open-plan office spaces: A state-of-the-art review. *Build. Environ.* **2017**, *112*, 308–321. [CrossRef]
22. Adam, G.K. DALI LED Driver Control System for Lighting Operations Based on Raspberry Pi and Kernel Modules. *Electronics* **2019**, *8*, 1021. [CrossRef]
23. Wu, M.L.; Kung, C.M.; Lin, Y.N. DALI-2 Intelligent Lighting Control System. In Proceedings of the 2020 International Symposium on Computer, Consumer and Control (IS3C), Taichung, Taiwan, 13–16 November 2020; pp. 158–161. [CrossRef]
24. Varghese, S.G.; Kurian, C.P.; George, V.; John, A.; Nayak, V.; Upadhyay, A. Comparative study of zigBee topologies for IoT-based lighting automation. *IET Wirel. Sens. Syst.* **2019**, *9*, 201–207. [CrossRef]
25. Sistu, S.; Liu, Q.; Ozcelebi, T.; Dijk, E.; Zotti, T. Performance Evaluation of Thread Protocol based Wireless Mesh Networks for Lighting Systems. In Proceedings of the 2019 International Symposium on Networks, Computers and Communications (ISNCC), Istanbul, Turkey, 18–20 June 2019; pp. 1–8. [CrossRef]
26. Nobile, M.; Palmese, F.; Boiano, A.; Redondi, A.E.C.; Cesana, M. Zigbee Vs. Matter Over Thread: Understanding IoT Protocol Performance in Practice. In Proceedings of the 2026 Mediterranean Artificial Intelligence and Networking Conference (MAIN), Palermo, Italy, 1–3 July 2026; pp. 1–9. [CrossRef]
27. Kumar Arigela, A.; Banapuram, C.; Venu, N. Remote based Home Automation with MQTT: ESP32 Nodes and Node-RED on Raspberry Pi. In Proceedings of the 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Kirtipur, Nepal, 3–5 October 2024; pp. 313–318. [CrossRef]
28. Chen, C.Y.; Wu, S.H.; Huang, B.W.; Huang, C.H.; Yang, C.F. Web-based Internet of Things on environmental and lighting control and monitoring system using node-RED, MQTT and Modbus communications within embedded Linux platform. *Internet Things* **2024**, *27*, 101305. [CrossRef]
29. Putrada, A.G.; Abdurohman, M.; Perdana, D.; Nuha, H.H. EdgeSL: Edge-Computing Architecture on Smart Lighting Control With Distilled KNN for Optimum Processing Time. *IEEE Access* **2023**, *11*, 64697–64712. [CrossRef]
30. Sharkawy, A.N.; Hasanin, M.; Sharf, M.; Mohamed, M.; Elsheikh, A. Development of Smart Home Applications Based on Arduino and Android Platforms: An Experimental Work. *Automation* **2022**, *3*, 579–595. [CrossRef]
31. Jabbar, W.A.; Kian, T.K.; Ramli, R.M.; Zubir, S.N.; Zamrizaman, N.S.M.; Balfaqih, M.; Shepelev, V.; Alharbi, S. Design and Fabrication of Smart Home With Internet of Things Enabled Automation System. *IEEE Access* **2019**, *7*, 144059–144074. [CrossRef]
32. Zagan, I.; Găitan, V.G. BIoT Smart Switch-Embedded System Based on STM32 and Modbus RTU—Concept, Theory of Operation and Implementation. *Buildings* **2024**, *14*, 3076. [CrossRef]
33. Serepas, F.; Papias, I.; Christakis, K.; Dimitropoulos, N.; Marinakis, V. Lightweight Embedded IoT Gateway for Smart Homes Based on an ESP32 Microcontroller. *Computers* **2025**, *14*, 391. [CrossRef]
34. Khomenko, Y.; Babichev, S. Modular IoT Architecture for Monitoring and Control of Office Environments Based on Home Assistant. *IoT* **2025**, *6*, 69. [CrossRef]
35. Dharshini, B.; Divya Dharshini, G.; Hari Kumaaran, R.; Shruthi, S.; Surya Kumar, M.; Packia Sudha, M. Enerhome: IoT-Based Smart Home Management with Rainmaker. In Proceedings of the 6th International Conference on Trends in Material Science and Inventive Materials (ICTMIM), Kanyakumari, India, 8–10 April 2026; pp. 1241–1249. [CrossRef]
36. Galván, J.A.; Mandujano, D.; Canchos, L.; Lujerio, W.; Chauca, M. A Simulated IoT Embedded System for Classroom Indoor Air Quality, Noise, and Lighting Monitoring: Architecture and Firmware Performance Analysis. *IoT* **2026**, *7*, 72. [CrossRef]
37. Ling, J.; Wang, Y.; Feng, Y.; Niu, Y. The Impact of Long-Press Duration and Double-Click Interval on User Experience in Physical Button Interaction. In Proceedings of the 17th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 22–23 August 2025; pp. 68–71. [CrossRef]
38. Morales-Gonzalez, C.; Harper, M.; Cash, M.; Luo, L.; Ling, Z.; Sun, Q.Z.; Fu, X. On building automation system security. *High-Confid. Comput.* **2024**, *4*, 100236. [CrossRef]
39. Copos, B.; Levitt, K.; Bishop, M.; Rowe, J. Is Anybody Home? Inferring Activity From Smart Home Network Traffic. In Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 245–251. [CrossRef]
40. Koopman, P.; Chakravarty, T. Cyclic redundancy code (CRC) polynomial selection for embedded networks. In Proceedings of the International Conference on Dependable Systems and Networks (DSN 2004), Florence, Italy, 28 June–1 July 2004; pp. 145–154. [CrossRef]

41. DiLaura, D.L.; Houser, K.W.; Mistrick, R.G.; Steffy, G.R., Eds. *The Lighting Handbook: Reference and Application*, 10th ed.; Illuminating Engineering Society of North America: New York, NY, USA, 2011.
42. ESPHome. ESPHome: Smart Home Made Simple. 2026. Available online: https://esphome.io (accessed on 8 October 2026).
43. Atmel Corporation, San Jose, CA, USA. ATmega640/V-1280/V-1281/V-2560/V-2561/V 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash, Datasheet 2549Q–AVR–02/2014. 2014. Available online: https://ww1.microchip.com/downloads/en/devicedoc/atmega640-1280-1281-2560-2561-datasheet-ds40002211a.pdf (accessed on 8 October 2026).
44. Espressif Systems. ESP32-C3 Series Datasheet, Version 2.4. 2026. Available online: https://documentation.espressif.com/esp32-c3_datasheet_en.html (accessed on 8 October 2026).
45. Texas Instruments. SN74AHCT1G125 Single Bus Buffer Gate with 3-State Output, Datasheet. 2016. Available online: https://www.ti.com/lit/ds/symlink/sn74ahct1g125.pdf (accessed on 17 November 2026).
46. Omron. G3MB Solid State Relay: Low Cost Subminiature PCB Mounting 2 Amp Single In-Line Package SSR, Datasheet. 1999. Available online: https://docs.rs-online.com/db5e/0900766b8002f991.pdf (accessed on 26 November 2026).
47. Arduino. EEPROM Library Guide. 2026. Available online: https://docs.arduino.cc/learn/programming/eeprom-guide/ (accessed on 8 October 2026).
48. ESPHome. aioesphomeapi: Python Client for the ESPHome Native API. 2026. Available online: https://github.com/esphome/aioesphomeapi (accessed on 8 October 2026).
49. Poulter, A.J.; Cox, S.J. Enabling Secure Guest Access for Command-and-Control of Internet of Things Devices. *IoT* **2021**, *2*, 236–248. [CrossRef]
50. Razavi, R.; Gharipour, A.; Fleury, M.; Akpan, I.J. Occupancy detection of residential buildings using smart meter data: A large-scale study. *Energy Build.* **2019**, *183*, 195–208. [CrossRef]
51. Donenfeld, J.A. WireGuard: Next Generation Kernel Network Tunnel. In Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 26 February–1 March 2017. [CrossRef]
52. Pennarun, A. How Tailscale Works. Tailscale Blog. 2020. Available online: https://tailscale.com/blog/how-tailscale-works (accessed on 15 October 2026).
53. Arduino. ArduinoCore-avr, Version 1.8.6: HardwareSerial Implementation. 2022. Available online: https://github.com/arduino/ArduinoCore-avr/tree/1.8.6/cores/arduino (accessed on 8 October 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.