ESPHome

Overview

Integrate ESPHome-based devices into Control4. ESPHome is an open-source system that transforms common microcontrollers, like ESP8266 and ESP32, into smart home devices through simple YAML configuration. ESPHome devices can be set up, monitored, and controlled using a web browser, Home Assistant, or other compatible platforms. This driver enables seamless monitoring and control of ESPHome devices directly from your Control4 system.

Index

System Requirements

- Control4 OS 3.3+

Features

- Local network communication requiring no cloud services
- Real-time updates from all supported entities exposed by the device
- Supports encrypted connections using the device encryption key
- Variable Programming Support

Compatibility

Verified Devices

This driver will generically work with any ESPHome device, but we have tested extensively with the following devices:

- ratgdo - Configuration Guide

If you try this driver on a product listed above, and it works, let us know!

Supported Bluetooth Devices

When used as a Bluetooth proxy, this driver supports the following BLE device types through sub-drivers:

Protocol	Sub-Driver	Example Devices
SwitchBot	ESPHome SwitchBot	Bot, Plug Mini, Relay Switch, Meter, Motion, Contact
BTHome	ESPHome BTHome	Shelly BLU Button/Door/Motion/H&T, DIY sensors
Govee	ESPHome Govee	Temperature/humidity monitors, meat thermometers

Protocol	Sub-Driver	Example Devices
Yale/August	ESPHome Yale	Yale and August smart locks

See the individual sub-driver documentation for device-specific details.

Supported ESPHome Entities

Entity Type	Supported
Alarm Control Panel	❌
API Noise	✅
Binary Sensor	✅
Bluetooth Proxy	✅
Button	✅
Camera	❌
Climate	✅
Cover	✅
Date	✅
Datetime	✅
Event	✅
Fan	✅
Light	✅
Lock	✅
Media Player	❌
Number	✅
Select	✅
Sensor	✅
Siren	❌
Switch	✅
Text	✅
Text Sensor	✅
Time	✅
Update	❌
Valve	❌
Voice Assistant	❌*
Water Heater	✅

> * Voice Assistant requires a speech-to-text and intent processing pipeline (e.g. Home Assistant Assist). Control4 does not natively provide voice intent handling, so this entity type is not supported.

Installer Setup

> ⚠️ Only a ***single*** driver instance is required per ESPHome device. Multiple instance of this driver connected to the same device will have unexpected behavior. However, you can have multiple instances of this driver connected to ***different*** ESPHome devices.

DriverCentral Cloud Setup

> If you already have the DriverCentral Cloud driver installed in your project you can continue to Driver Installation.

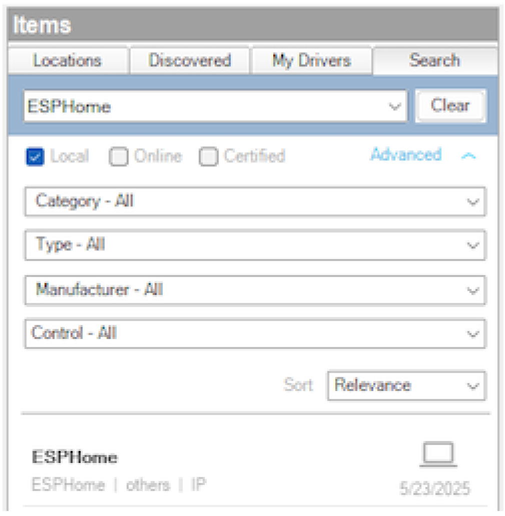
This driver relies on the DriverCentral Cloud driver to manage licensing and automatic updates. If you are new to using DriverCentral you can refer to their Cloud Driver documentation for setting it up.

Driver Installation

Driver installation and setup are similar to most other ip-based drivers. Below is an outline of the basic steps for your convenience.

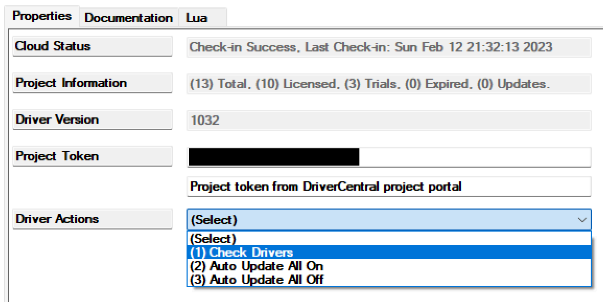
1. Download the latest `control4-esphome.zip` from DriverCentral.
2. Extract and install all `.c4z` files.
3. Use the "Search" tab to find the "ESPHome" driver and add it to your project.

   > ⚠️ A ***single*** driver instance is required per ESPHome device.



4. Select the newly added driver in the "System Design" tab. You will notice that the `Cloud Status` reflects the license state. If you have purchased a license it will show `License Activated`, otherwise `Trial Running` and remaining trial duration.

5. You can refresh license status by selecting the "DriverCentral Cloud" driver in the "System Design" tab and perform the "Check Drivers" action.



Cloud Status	Check-in Success. Last Check-in: Sun Feb 12 21:32:13 2023
Project Information	(13) Total. (10) Licensed. (3) Trials. (0) Expired. (0) Updates.
Driver Version	1032
Project Token	[redacted]
	Project token from DriverCentral project portal
Driver Actions	(Select)

6. Configure the Device Settings with the connection information.

7. After a few moments the `Driver Status` will display `Connected`. If the driver fails to connect, set the `Log Mode` property to `Print` and re-set the `IP Address` field to reconnect. Then check the lua output window for more information.

8. Once connected, the driver will automatically create variables and connection bindings for each supported entity type.

9. To control climate, lights, fans, locks, and/or water heaters, use the "Search" tab to find the "ESPHome Climate", "ESPHome Light", "ESPHome Fan", and/or "ESPHome Lock" driver. For fans, choose the speed variant that matches your fan (e.g., "ESPHome Fan (3 Speed)"). Water heater entities use the "ESPHome Climate" driver. Add one driver instance for each exposed entity in your project. In the "Connections" tab, select the "ESPHome" driver and bind the entities to the newly added drivers.

Driver Setup

Driver Properties

Cloud Settings

Cloud Status (read-only)

Displays the DriverCentral cloud license status.

Automatic Updates [Off | *On*]

Enables or disables automatic driver updates via DriverCentral.

Driver Settings

Driver Status (read-only)

Displays the current status of the driver.

Driver Version (read-only)

Displays the current version of the driver.

Log Level [0 - Fatal | 1 - Error | 2 - Warning | *3 - Info* | 4 - Debug | 5 - Trace | 6 - Ultra]

Sets the logging level. Default is `3 - Info`.

Log Mode [*Off* | Print | Log | Print and Log]

Sets the logging mode. Default is `Off`.

Device Log Forwarding [*Off* | On]

Forward ESPHome device logs to the driver's Lua output at the current Log Level. Changing Log Level or disabling Log Mode will reconnect to apply the new settings.

Device Settings

IP Address

Sets the device IP address (e.g. `192.168.1.30`). Domain names are allowed as long as they can be resolved to an accessible IP address by the controller. HTTPS is not supported.

> ⚠️ If you are using an IP address, you should ensure it will not change by assigning a static IP or creating a DHCP reservation.

Port [1 - 65535, default: *6053*]

Sets the device port. The default port for ESPHome devices is `6053`.

Authentication Mode [*None* | Password | Encryption Key]

Selects the authentication method for connecting to the ESPHome device.

- **None**: No authentication required.
- **Password**: Use a password for authentication (see below).
- **Encryption Key**: Use an encryption key for secure communication (see below).

> **Tip:** For ESPHome devices used primarily as Bluetooth proxies, consider configuring the firmware without API encryption. In busy BLE environments, the

> volume of proxy traffic combined with the controller's limited cryptographic performance can cause significant CPU load. BLE traffic is inherently over-the-air, and sensitive protocols (such as Yale/August lock commands) use their own end-to-end encryption between the driver and the device, making it safe to relay through an unencrypted proxy. To disable encryption, omit the `encryption` block from the ESPHome `api:` configuration and set Authentication Mode to `None`.

Password

Shown only if Authentication Mode is set to `Password`. Sets the device password. This must match the password configured on the ESPHome device.

Encryption Key

Shown only if Authentication Mode is set to `Encryption Key`. Sets the device encryption key for secure communication. This must match the encryption key configured on the ESPHome device.

Use OpenSSL [*Yes* | No]

Use OpenSSL for encryption. This should typically be left at the default value of `Yes` for better performance and compatibility.

Bluetooth Proxy Settings

> The Bluetooth Proxy feature requires an ESP32 device with the `bluetooth_proxy` component configured in ESPHome. See the ESPHome Bluetooth Proxy documentation for firmware configuration.

Bluetooth Proxy Status (read-only)

Shows the current state of the Bluetooth proxy. The format is a pipe-separated list of status components:

Standalone Mode: `Standalone Mode | Scanning (Passive) | 1/4 Active`

Coordinator Mode: `Coordinator Mode | Scanning (Passive) | 0/3 Active | MAC Filter: 5`

Components explained:

- **Mode** - "Standalone Mode" or "Coordinator Mode" depending on whether the driver is connected to a Bluetooth Coordinator
- **Scanner State** - Current state (Idle, Starting, Running, Stopping, Stopped, Failed) and mode (Passive or Active)

- **Connection Slots** - Shows "used/total Active" slots (e.g., "1/4 Active" means 1 slot in use out of 4 available). If you select more active devices than available slots, " (Oversubscribed)" is appended (see Oversubscription)
- **MAC Filter** - (Coordinator mode only) Shows how many device MACs are being filtered for, or "none" if forwarding all advertisements

Bluetooth Proxy Capabilities (read-only)

Displays a comma-separated list of capabilities supported by this ESPHome Bluetooth proxy:

- **Scan** - Can receive BLE advertisements (passive scanning)
- **Connect** - Can establish GATT connections to devices (active connections)
- **Cache** - Caches GATT service data remotely
- **Pair** - Can pair with devices requiring authentication
- **Raw** - Can receive raw advertisement data

Select Bluetooth Devices

> Only visible in **Standalone Mode** (hidden when connected to a Bluetooth Coordinator, as device selection is done there instead).

A dropdown list showing discovered BLE devices. Select "Refresh List" to start a new scan.

During scanning:

- The dropdown displays "-- Scanning..." while the scan is in progress
- Select "-- Stop Scan" to stop early and keep newly discovered devices
- Select "-- Abort Scan" to stop early and discard newly discovered devices
- When complete, the dropdown repopulates with discovered devices

Device list shows:

- MAC Address
- Device Name (if available)
- Device Type (BTHome, SwitchBot, Govee, etc.)
- Connection Type (Active/Passive)

After selecting a device, a connection binding is automatically created for the appropriate sub-driver.

> **Tip:** Some BLE devices (buttons, sensors with long advertisement intervals) may be in sleep mode. Wake them by pressing buttons, triggering motion sensors, or

> opening/closing contact sensors during the scan to ensure they appear in the device list.

Bluetooth Scan Duration [5 - 60, default: *30*]

> Only visible in **Standalone Mode**.

Sets how long (in seconds) to scan for BLE devices when refreshing the device list. Longer scans may discover more devices with long advertisement intervals.

Bluetooth Proxy Room

> Only visible in **Coordinator Mode** (when connected to a Bluetooth Coordinator).

Sets the room where this Bluetooth proxy is physically located. This is used for presence tracking to determine which room a device is in based on signal strength.

Minimum Room RSSI Override (dBm) [-100 - -40, default: *-100*]

> Only visible in **Coordinator Mode** (when connected to a Bluetooth Coordinator).

Overrides the coordinator's global "Minimum Room RSSI" setting for this proxy's room. Use this when a room has different size or characteristics than others.

- **-100 (default)** - Use the coordinator's global setting
- **-85** - More permissive; allow detection from further away (large rooms)
- **-60** - More restrictive; require closer proximity (small rooms)

Examples:

- Large living room: Set to `-85` to allow detection from further away
- Small bathroom: Set to `-60` to require closer proximity
- Leave at `-100` to use the coordinator's global setting

> **Note:** Only affects room assignment for this proxy. The value is sent to the Bluetooth Coordinator when this proxy connects or when the setting changes.

Device Info

Name (read-only)

Displays the name of the connected ESPHome device.

Model (read-only)

Displays the model of the connected ESPHome device.

Manufacturer (read-only)

Displays the manufacturer of the connected ESPHome device.

MAC Address (read-only)

Displays the MAC address of the connected ESPHome device.

Firmware Version (read-only)

Displays the firmware version of the connected ESPHome device.

Driver Actions

Reset Driver

> ⚠️ This will reset all connection bindings and delete any programming associated with the variables.

Resets the driver to its initial state, removing all dynamically created connections and variables. This is useful if you change the connected ESPHome device or there are stale connections or variables.

Parameters:

- **Are You Sure?** [***No*** | Yes] - Confirmation to reset the driver.

Programming Reference

Once connected, the driver automatically creates variables and bindings for each supported ESPHome entity. Use this reference for Control4 programming.

Driver Variables

In addition to per-entity variables, the driver publishes the following device-level variables sourced from the ESPHome device info. They mirror the matching read-only Device Info properties.

Variable	Type	Description
Name	STRING	Friendly name reported by the ESPHome device
Model	STRING	Device model string reported by ESPHome
Manufacturer	STRING	Manufacturer string reported by ESPHome
MAC Address	STRING	ESPHome device MAC address

Variables by Entity Type

Entity Type	Variable Name	Type	Notes
Binary Sensor	`{name} State`	BOOL	"1" = triggered, "0" = clear
Button	(none)	-	Use "Press Button" command (see below)
Climate	(none)	-	State via Thermostat proxy
Cover	`{name} State`	STRING	"open", "closed", "opening", "closing"
Date	`{name}`	STRING	Writable, formatted as YYYY-MM-DD
Datetime	`{name}`	STRING	Writable, formatted as YYYY-MM-DD HH:MM:SS
Event	`{name} Last Event`	STRING	Last event type (e.g., "single_press")
Fan	(none)	-	State via Fan proxy
Light	(none)	-	State via Light proxy
Lock	(none)	-	State via Lock proxy
Number	`{name}`	NUMBER	Writable, 1 decimal precision
Select	`{name}`	STRING	Writable, current option
Sensor	`{name}`	NUMBER	Read-only, 1 decimal precision
Switch	`{name} State`	BOOL	"1" = on, "0" = off (writable)
Text	`{name}`	STRING	Writable
Text Sensor	`{name}`	STRING	Read-only
Time	`{name}`	STRING	Writable, formatted as HH:MM:SS
Water Heater	(none)	-	State via Thermostat proxy

> **Note:** `{name}` is replaced with the entity's display name from ESPHome (e.g., a sensor named "Temperature" creates a variable called "Temperature").

Bindings by Entity Type

Entity Type	Binding Class	Purpose
Binary Sensor	`CONTACT_SENSOR`	Integrates with Contact Sensor proxy
Switch	`RELAY`	Control via Relay proxy
Cover	`CONTACT_SENSOR`	Open/closed state contacts
Cover	`RELAY`	Open/close/stop control relays
Button	`BUTTON_LINK`	Allows other devices to trigger button
Climate	`ESPHOME_CLIMATE`	Bind to ESPHome Climate sub-driver **
Fan	`ESPHOME_FAN_N_SPEED[_REVERSE]`	Bind to ESPHome Fan sub-driver
Light	`ESPHOME_LIGHT`	Bind to ESPHome Light sub-driver
Lock	`ESPHOME_LOCK`	Bind to ESPHome Lock sub-driver
Water Heater	`ESPHOME_CLIMATE`	Bind to ESPHome Climate sub-driver

> **Note:** Sensor, Number, Select, Text, Text Sensor, Date, Time, Datetime, and Event entities do not create bindings. They expose data only through variables and events.

> **Note:** Water Heater entities share the `ESPHOME_CLIMATE` binding class and are controlled via the ESPHome Climate sub-driver. Device modes (such as Eco, Heat Pump, Gas) are exposed as presets.

> ** **Climate remote sensor:** The Climate sub-driver supports using an external Control4 temperature sensor instead of the climate device's built-in sensor. This feature requires the ESPHome device YAML to define user-defined API services (e.g., `set_remote_temperature` and `use_internal_temperature`). See the ESPHome Climate sub-driver documentation for full details and YAML examples.

Bluetooth Proxy Bindings

When the connected ESPHome device exposes a Bluetooth proxy, additional dynamic bindings are created separately from the entity bindings above:

Source	Binding Class	Purpose
Bluetooth Coordinator link	`ESPHOME_BLUETOOTH`	Connects the ESPHome driver to the Bluetooth Coordinator
Selected BTHome device	`ESPHOME_BTHOME`	Bind to ESPHome BTHome sub-driver
Selected Govee device	`ESPHOME_GOVEE`	Bind to ESPHome Govee sub-driver
Selected SwitchBot device	`ESPHOME_SWITCHBOT`	Bind to ESPHome SwitchBot sub-driver
Selected Yale/August lock	`ESPHOME_YALE`	Bind to ESPHome Yale sub-driver

> **Note:** Per-device Bluetooth bindings are created automatically when a device is chosen via the `Select Bluetooth Devices` property (standalone mode) or via the Bluetooth Coordinator's device selector (coordinator mode). Binding display names include the device MAC address suffix for easy identification.

Events by Entity Type

Entity Type	Event Name	Description
Event	`{name}: {event_type}`	One C4 event per event type for programming

> **Note:** Event entities are stateless triggers (button presses, gestures, doorbell rings). Each discovered event type creates a Control4 event that can be used in programming. The `{name} Last Event` variable tracks the most recent event type.

Commands

Command	Parameters	Description
Press Button	Button	Triggers an ESPHome button entity by name
Set Select	Select, Option	Sets a select entity to the specified option

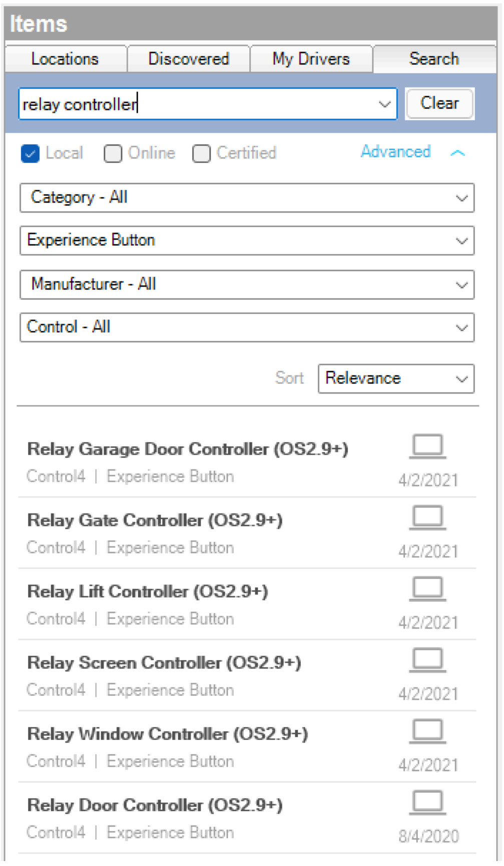
> **Note:** The Button parameter is a dynamic list populated with discovered ESPHome button entities. The Select and Option parameters are dynamic lists populated with discovered ESPHome select entities and their available options.

ratgdo Configuration Guide

This guide provides instructions for configuring the ESPHome driver to work with ratgdo devices for garage door control via relays in Control4 Composer Pro.

Add Relay Controller Driver

Add the desired relay controller driver to your Control4 project in Composer Pro.



Relay Garage Door Controller (OS2.9+)
Control4 | Experience Button 4/2/2021

Relay Gate Controller (OS2.9+)
Control4 | Experience Button 4/2/2021

Relay Lift Controller (OS2.9+)
Control4 | Experience Button 4/2/2021

Relay Screen Controller (OS2.9+)
Control4 | Experience Button 4/2/2021

Relay Window Controller (OS2.9+)
Control4 | Experience Button 4/2/2021

Relay Door Controller (OS2.9+)
Control4 | Experience Button 8/4/2020

Relay Controller Properties

The ratgdo device exposes a "Cover" entity in ESPHome, which maps to the relay controller functionality in Control4.

Number of Relays

The ratgdo device uses a multi-relay configuration to control the garage door. In Composer Pro, you should configure the relay settings as follows:

- Set to **2 Relays** (Open/Close) or **3 Relays** (Open/Close/Stop)
    - The ratgdo device uses separate commands for opening and closing the garage door
    - If your ratgdo firmware supports the "stop" command, configure for 3 relays to enable the stop functionality. If you are not sure, you can look at the ratgdo connections in Composer Pro to see if the "Stop Door" relay is available.

Relay Configuration

- Set to **Pulse**
    - ratgdo uses momentary pulses to trigger the garage door opener, similar to a wall button press

Pulse Time

- Set all relay pulse times to **500** (default)
    - This is the duration the relay will be activated

Invert Relay

- Set all invert relay properties to **No** (default)

Contact Debounce

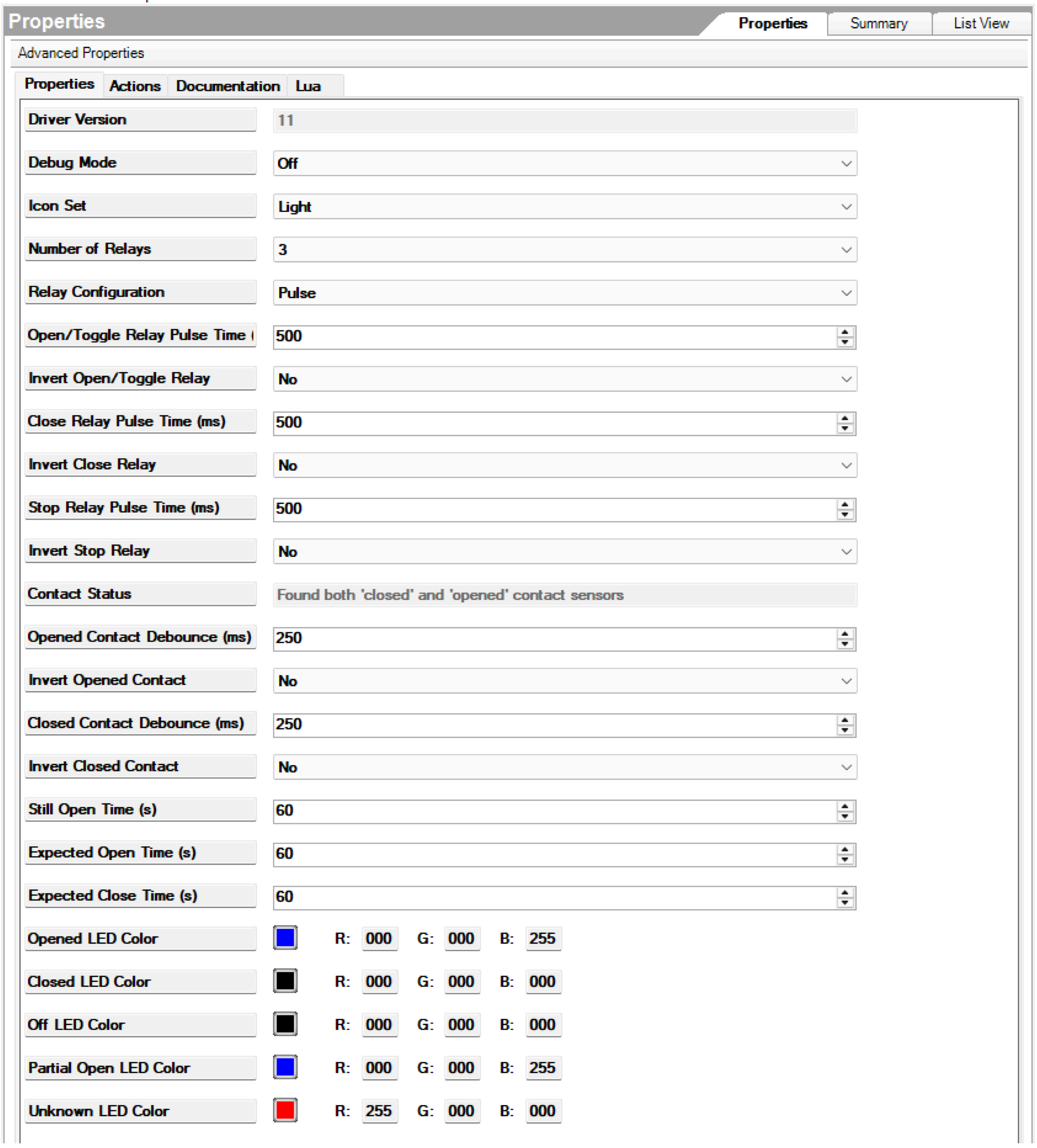
- Set all contact debounce times to **250** (default)
    - This helps prevent false flapping of the garage door state sensors

Invert Contact

- Set all invert contact properties to **No** (default)

Example Properties

For reference, here is an example of the relay controller properties in Composer Pro:



Driver Version	11
Debug Mode	Off
Icon Set	Light
Number of Relays	3
Relay Configuration	Pulse
Open/Toggle Relay Pulse Time (	500
Invert Open/Toggle Relay	No
Close Relay Pulse Time (ms)	500
Invert Close Relay	No
Stop Relay Pulse Time (ms)	500
Invert Stop Relay	No
Contact Status	Found both 'closed' and 'opened' contact sensors
Opened Contact Debounce (ms)	250
Invert Opened Contact	No
Closed Contact Debounce (ms)	250
Invert Closed Contact	No
Still Open Time (s)	60
Expected Open Time (s)	60
Expected Close Time (s)	60
Opened LED Color	🟦 R: 000 G: 000 B: 255
Closed LED Color	⬛ R: 000 G: 000 B: 000
Off LED Color	⬛ R: 000 G: 000 B: 000
Partial Open LED Color	🟦 R: 000 G: 000 B: 255
Unknown LED Color	🟥 R: 255 G: 000 B: 000

Relay Controller Connections

Relays

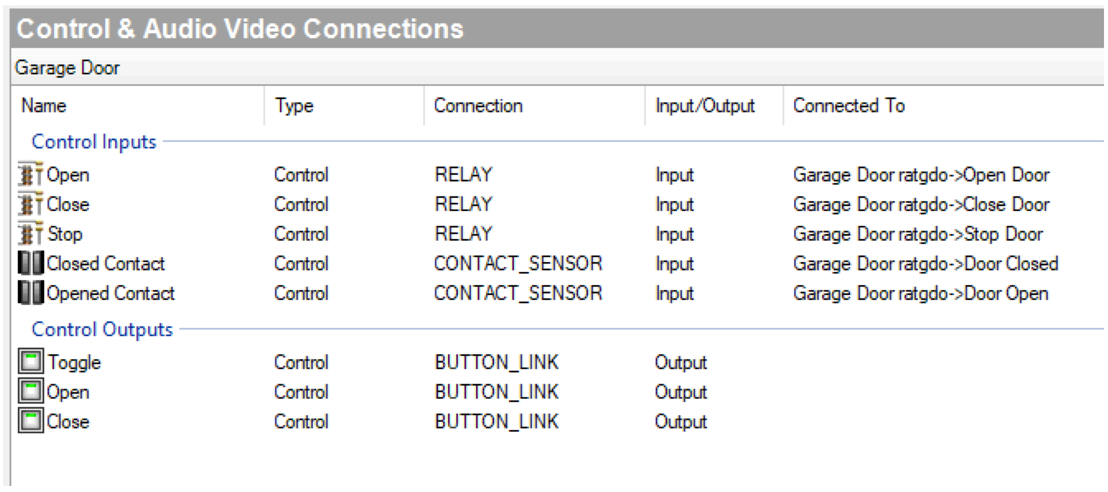
- **Open**: Connect to the ratgdo's "Open Door" relay
- **Close**: Connect to the ratgdo's "Close Door" relay
- **Stop**: Connect to the ratgdo's "Stop Door" relay, if available

Contact Sensors

- **Closed Contact**: Connect to the ratgdo's "Door Closed" contact
- **Opened Contact**: Connect to the ratgdo's "Door Open" contact

Example Connections

For reference, here is an example of how the connections should look in Composer Pro:



Control & Audio Video Connections

Garage Door

Name	Type	Connection	Input/Output	Connected To
Control Inputs				
Open	Control	RELAY	Input	Garage Door ratgdo->Open Door
Close	Control	RELAY	Input	Garage Door ratgdo->Close Door
Stop	Control	RELAY	Input	Garage Door ratgdo->Stop Door
Closed Contact	Control	CONTACT_SENSOR	Input	Garage Door ratgdo->Door Closed
Opened Contact	Control	CONTACT_SENSOR	Input	Garage Door ratgdo->Door Open
Control Outputs				
Toggle	Control	BUTTON_LINK	Output	
Open	Control	BUTTON_LINK	Output	
Close	Control	BUTTON_LINK	Output	

Programming

You can create programming in Control4 to:

- Open/close the garage door based on events
- Monitor the garage door state
- Set up notifications for garage door status changes
- Create custom buttons on touchscreens and remotes

Example: Creating a "Still Open" Alert

Using the "Still Open Time" property from the relay controller driver:

1. Set the "Still Open Time" to your desired duration (e.g., 10 minutes)
2. Create a programming rule that triggers when the "Still Open" event fires
3. Add actions to send notifications or perform other tasks

Additional Entities

Depending on your ratgdo device, firmware, and its capabilities, there may be additional entities exposed by the ESPHome driver. These can come as additional connections or driver variables.

Please refer to ratgdo's documentation for more information on specific entities:

https://ratgdo.github.io/esphome-ratgdo/webui_documentation.html

Bluetooth Proxy Configuration Guide

This guide explains how to use the ESPHome Bluetooth Proxy feature to integrate BLE devices into Control4.

Prerequisites

- ESP32 device with ESPHome firmware and `bluetooth_proxy` component enabled
- BLE devices within range of the ESP32

Recommended Hardware:

The following POE-powered Bluetooth proxies are excellent choices with 4 active connection slots:

- Seeed Studio XIAO ESP32C6 with POE expansion board
- Olimex ESP32-POE or ESP32-POE-ISO

Firmware Installation:

- **Quick Start:** Use web.esphome.io for one-click firmware installation directly in your browser - no YAML configuration needed.
- **Advanced:** For more control over settings, create a custom YAML configuration. See the ESPHome Bluetooth Proxy documentation for configuration options.

Understanding Connection Types

BLE devices use one of two connection modes:

Passive Mode (No Slot Required)

Passive devices broadcast their data in advertisements. The proxy listens without establishing a connection. These devices include:

- **Shelly BLU devices** - Button, Door/Window, Motion, H&T sensors (native BTHome support)
- **BTHome sensors** - Temperature, humidity, motion, door sensors
- **Govee sensors** - Temperature/humidity monitors, meat thermometers

- **SwitchBot sensors** - Meters, motion sensors, contact sensors, water leak

Advantage: Unlimited passive devices can be monitored simultaneously.

Active Mode (Uses Connection Slot)

Active devices require a GATT connection to send commands. The ESP32 has limited connection slots (typically 3-4). These devices include:

- **SwitchBot Bot** - Requires connection to send press/on/off commands
- **SwitchBot Switch** - Plug Mini, Relay switches (encrypted commands)
- **Yale/August Locks** - Requires connection for encrypted lock/unlock commands

Oversubscription

You can select more active devices than available connection slots. This is called **oversubscription** and works well when devices only need brief connections to exchange data (e.g., sending a command to a SwitchBot Bot). The device connects, sends the command, and immediately frees the slot.

Oversubscription becomes problematic when multiple devices need simultaneous connections. If all slots are in use, commands to additional devices will queue and retry until a slot becomes available.

> **Tip:** If you see "Allocation failed" errors in the logs, you have too many concurrent active connections. Consider reducing the number of active devices or adding another proxy.

Step-by-Step Setup

1. **Add the ESPHome driver** and configure it to connect to your ESP32 device.

2. **Verify Bluetooth Proxy Status** shows "Ready" with available slots.

3. **Scan for devices:**

    - Set "Bluetooth Scan Duration" (30 seconds recommended)
    - Select "Refresh List" from the "Select Bluetooth Devices" dropdown
    - Wait for the scan to complete

4. **Select a discovered device** from the dropdown. A connection will be automatically created.

5. **Add the appropriate sub-driver:**

    - Search for the driver matching your device type (e.g., "ESPHome BTHome")
    - Add it to your project

6. **Bind the sub-driver** to the connection created in step 4.

7. **Configure the sub-driver** properties as needed.

Supported Device Types

Device Protocol	Sub-Driver	Connection
BTHome	ESPHome BTHome	Passive
Govee	ESPHome Govee	Passive
SwitchBot	ESPHome SwitchBot	Active/Passive
Yale/August	ESPHome Yale	Active

Performance Considerations

Device Limits

The ESP32 Bluetooth proxy has practical limits on device capacity:

Connection Type	Recommended Limit	Notes
Passive devices	20-30	Sensors broadcasting advertisements
Active connections	3-5	Devices requiring GATT connections

Exceeding these limits may cause:

- Missed advertisements from passive devices
- "Allocation failed" errors for active connections
- Delayed or failed commands to active devices

> **Warning:** If you see "Too many BLE events to process" in the logs, the proxy is overwhelmed. Reduce the number of tracked devices or add another proxy.

Busy BLE Environments

In environments with many BLE devices (smart home hubs, fitness equipment, wireless speakers, neighbors' devices), performance may degrade:

- **2.4 GHz congestion** - BLE and WiFi share spectrum; heavy WiFi traffic affects BLE reception
- **Advertisement flooding** - Many devices broadcasting simultaneously can overwhelm the scanner
- **Interference sources** - Microwaves, USB 3.0 devices, and poorly shielded electronics cause interference

Mitigation strategies:

1. Use Ethernet-connected ESP32 boards (Olimex ESP32-POE) to avoid WiFi/BLE contention
2. Position proxies away from WiFi routers (at least 2-3 meters)
3. Use the Bluetooth Coordinator to distribute load across multiple proxies
4. Reduce scan duration if not actively discovering new devices

Proxy Placement Tips

For optimal BLE reception:

- **Height matters** - Place at chest height or higher, not on the floor
- **Avoid metal** - Keep away from refrigerators, filing cabinets, and metal shelving (metal causes reflections and unstable signals)
- **Avoid enclosed spaces** - Don't place inside cabinets or behind furniture
- **Distance from electronics** - Keep 2-3 meters from routers, switches, and other network equipment

Coverage expectations:

Environment	Typical Range
Open indoor space	10-15 meters (30-50 ft)
Through 1-2 walls	5-10 meters (15-30 ft)
Concrete/brick walls	3-5 meters (10-15 ft)

Bluetooth Coordinator Setup

For advanced setups with **multiple ESPHome Bluetooth proxies**, use the **ESPHome Bluetooth Coordinator** driver to aggregate them. The coordinator provides:

- **RSSI-based routing -** Commands are routed to the proxy with the best signal strength for each BLE device
- **Automatic failover -** If a connection fails, the coordinator retries through alternate proxies
- **Room-level presence tracking -** Track which room a BLE device (like a phone) is in based on signal strength from each proxy

> **Note:** Apple devices (iPhone, iPad, Apple Watch) and Android devices with MAC randomization enabled cannot be tracked for presence. Use dedicated BLE beacons or devices with static MAC addresses instead.

When to Use the Coordinator

Setup	Recommendation
Single ESP32 proxy	Use ESPHome driver directly
Multiple proxies	Use Bluetooth Coordinator
Want presence tracking	Use Bluetooth Coordinator
Want failover/redundancy	Use Bluetooth Coordinator

Coordinator Setup Steps

1. **Add the Bluetooth Coordinator driver** to your project
2. **Connect your ESPHome drivers** to the coordinator's proxy bindings (Connections tab)
3. **Set the "Bluetooth Proxy Room" property** on each ESPHome driver to indicate where that proxy is physically located
4. **Select devices** via the coordinator's "Select Bluetooth Devices" property
5. **For presence tracking**, select devices via "Select Presence Devices"

When an ESPHome driver is connected to the coordinator:

- The "Select Bluetooth Devices" property is hidden (selection is done in the coordinator)

- The "Bluetooth Proxy Room" property becomes visible for presence tracking configuration

See the **ESPHome Bluetooth Coordinator** driver documentation for full details on presence tracking settings and events.

Developer Information

Support

If you have any questions or issues integrating this driver with Control4 or ESPHome, you can contact us at driver-support@finitelabs.com or call/text us at +1 (949) 371-5805.

Changelog

v20260418 - 2026-04-18

Added

- Added Event entity support: stateless triggers (button presses, gestures, doorbell rings) now create Control4 events for programming and track the last event type in a variable
- Added Date, Time, and Datetime entity support: configurable date/time values on the device are exposed as writable string variables (YYYY-MM-DD, HH:MM:SS, YYYY-MM-DD HH:MM:SS)
- Added Climate entity support: ESPHome climate devices are exposed as thermostatV2 sub-drivers with HVAC mode, setpoints, fan mode, presets, and humidity control
- Added Select entity support: STRING variable with the current option, writable via programming or variable writes
- Added "Set Select" programming command with dynamic Select and Option dropdowns

v20260326 - 2026-03-26

v20260325 - 2026-03-25

Fixed

- Fixed cover contact sensors sending duplicate notifications during open/close operations
- Fixed Yale DoorSense contact sensor sending duplicate "Closed" notifications on every poll cycle by tracking the last known door status and only reporting on actual state changes

v20260319 - 2026-03-19

Fixed

- Fixed an issue where entities were no longer being detected reliably on connection

v20260318 - 2026-03-18

Fixed

- Fixed Bluetooth Coordinator failing to connect to active BLE devices (SwitchBot, Yale locks) through proxies

v20260314 - 2026-03-14

Added

- Added fan support with on/off, speed control (1-6 speed variants), direction, and oscillation
- Added ESPHome Yale sub-driver for Yale/August BLE smart locks with lock/unlock control, door sense, and battery monitoring

v20260217 - 2026-02-17

Added

- Added Bluetooth proxy support with scanner infrastructure, advertisement parsing, and GATT connection management
- Added ESPHome Bluetooth Coordinator driver for multi-proxy aggregation with RSSI-based routing and connection failover
- Added room presence tracking with RSSI-based detection, anti-flapping, and contact sensor bindings
- Added ESPHome BTHome sub-driver for Shelly BLU and BTHome v1/v2 sensors
- Added ESPHome Govee sub-driver for temperature, humidity, and meat thermometer sensors
- Added ESPHome SwitchBot sub-driver for Bot, Plug Mini, Meter, Motion, and Contact devices
- Added device log forwarding to the ESPHome driver

v20251031 - 2025-10-31

Fixed

- Fixed compatibility with ESPHome 2025.10.0 for devices configured without passwords
- Improved password authentication failure detection and error reporting

v20251022 - 2025-10-22

Fixed

- Fixed an issue with parsing unknown fields in protobuf messages

v20251019 - 2025-10-19

Added

- Added support for OpenSSL with "Encryption Key" authentication mode across all applicable algorithms

Fixed

- Fixed a bug with the authentication flow in the latest 2025.10.0 firmware

v20250811 - 2025-08-11

Fixed

- Fixed switch entities not responding to bound relay proxies

v20250715 - 2025-07-14

Fixed

- Fixed bug causing entities to not be discovered on connect

v20250714 - 2025-07-14

Added

- Added support for encrypted connections using the device encryption key

v20250619 - 2025-06-19

Added

- Added ratgdo specific documentation

v20250606 - 2025-06-06

Added

- Initial Release