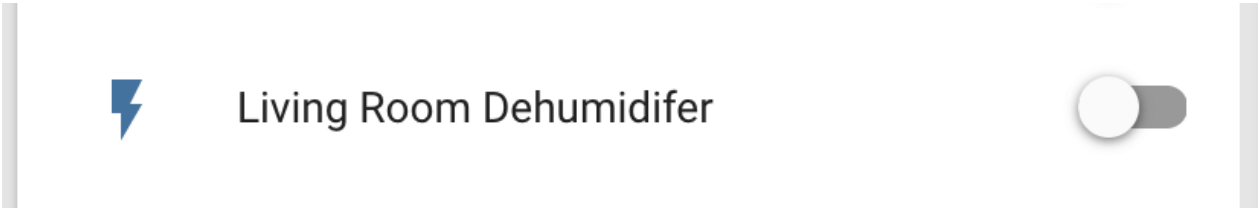
GPIO Switch

The `gpio` switch platform allows you to use any pin on your node as a switch. You can for example hook up a relay to a GPIO pin and use it through this platform.




```yaml
# Example configuration entry
switch:
  - platform: gpio
    pin: 25
    name: "Living Room Dehumidifier"
```

Configuration variables:

- **pin** (**Required**, Pin Schema): The GPIO pin to use for the switch.
- **name** (**Required**, string): The name for the switch.
- **id** (*Optional*, ID): Manually specify the ID used for code generation.
- **interlock** (*Optional*, list): A list of other GPIO switches in an interlock group. See Interlocking.
- **interlock_wait_time** (*Optional*, Time): For interlocking mode, set how long to wait after other items in an interlock group have been disabled before re-activating. Useful for motors where immediately turning on in the other direction could cause problems.
- All other options from Switch.

Active Low Switch

To create an active-low switch (one that is turned off by default), use the Pin Schema:

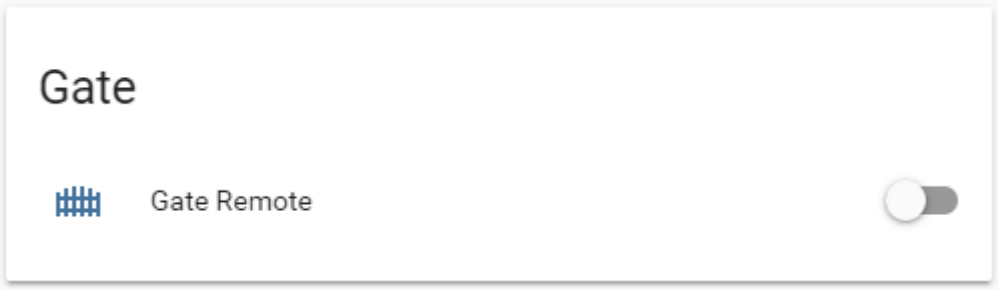
```yaml
# Example configuration entry
switch:
  - platform: gpio
    pin:
      number: 25
      inverted: true
```

Momentary Switch

To create momentary switches, for example switches that toggle a pin for a moment, you can use *on_turn_on* trigger.

An example that uses a single relay to activate a remote control button. The button can only start or stop the motor of the gate. In itself, the button or remote can not know if it opens or closes the gate. The relay simulates the button press for 500ms.

```yaml
# Example configuration entry
switch:
  - platform: gpio
    pin: 25
    id: relay
    name: "Gate Remote"
    icon: "mdi:gate"
    on_turn_on:
    - delay: 500ms
    - switch.turn_off: relay
```



Interlocking

In some cases it is necessary to ensure that two outputs are never active at the same time. ESPHome has a feature to prevent two GPIO Switches from being active at the same time called interlocking. Just give *each switch* in the "interlocking group" an `interlock` option with a list of all the switches in the group.

```yaml
# Example configuration entry
# Prevent relay #1 and relay #2 from being activated at the same time.
switch:
  - platform: gpio
    pin: GPIO25
    name: "Relay #1"
    id: relay1
    interlock: [relay2]

  - platform: gpio
    pin: GPIO26
    name: "Relay #2"
    id: relay2
    interlock: [relay1]
```

Or with some YAML anchors you can further simplify the config:

```yaml
# Example configuration entry
switch:
  - platform: gpio
    # etc
    id: relay1
    interlock: &interlock_group [relay1, relay2]
  - platform: gpio
    # etc
    id: relay2
    interlock: *interlock_group
```

Warning

These are software interlocks. As such, a software bug (which can *always* happen) can still activate both switches at the same time. Similarly, at reset time (before any of ESPHome's code runs) the relay GPIO pins may have pull-ups active, so the relay may be active before ESPHome can manually deactivate them.

So it is **highly** recommended to use hardware interlocks (like SPDT-type relays) that ensure that two GPIOs are never active at the same time.

See also `interlock_wait_time` to make interlocks group wait some amount of time before activating a switch.

See Also

- Switch Component
- GPIO Output
- Template Cover
- Simple Garage Door
- API Reference
- Edit this page on GitHub