ESPHome Integration

note

ESPHome V1.18 is required. This guide also assumes you have prior knowledge using ESPHome with Home Assistant.

1. Add New Device#

Begin by adding a new device in ESPHome. Copy the code below into the YAML file, completely replacing what is there.

Change the "device_name" to what you'd like.

Do not change any of the other values at the moment, we will do that later.

```
substitutions:
device_name: val-1000
pulley_diameter_mm: "12.2" # 1 turn about 47 mm
gear_ratio: "1" # not used yet for simplicity
distance_mm: "600" # about 10 turns
acceleration: 600 steps/s^2
velocity: 600 steps/s # 1 turn per second
open_current: 800ma
open_stall_threshold: "20"
close_current: 800ma
close_stall_threshold: "20"
tcool_threshold: "910"
microsteps: "2"
change_direction: "True"

external_components:
source: github://glmnet/esphome@stepper-tmc2209
components: [tmc2209]

esphome:
name: ${device_name}
platform: ESP32
board: nodemcu-32s
platformio_options:
upload_speed: 921600

on_boot:
- tmc2209.setup:
microsteps: 2
tcool_threshold: ${tcool_threshold}
current: 600mA
stall_threshold: 20

wifi:
ssid: SSID
password: PASSWWORD

# Enable logging
logger:
level: INFO

# verbose logger over uart causes motor artifacts as pulses are generated in main loop
# logger:
# level: VERY_VERBOSE
# logs:
# api: DEBUG
# api.service: DEBUG
# scheduler: DEBUG

ota:

uart:
id: uart_stepper
tx_pin: GPIO17
```

```
rx_pin: GPIO16
baud_rate: 9600

status_led:
pin: GPIO26

api:
services:
- service: control_stepper
variables:
target: int
speed: int
microsteps: int
tcool_threshold: int
stall_threshold: int
rms_current_amps: float
then:
- tmc2209.setup:
id: my_stepper
microsteps: !lambda "return microsteps;"
tcool_threshold: !lambda "return tcool_threshold;"
stall_threshold: !lambda "return stall_threshold;"
current: !lambda "return rms_current_amps;"
- logger.log:
format: moving to %d
args: [target]
- stepper.set_speed:
id: my_stepper
speed: !lambda "return speed;"
- stepper.set_target:
id: my_stepper
target: !lambda "return target;"
- service: set_stepper_zero
then:
- stepper.report_position:
id: my_stepper
position: 0
- stepper.set_target:
id: my_stepper
target: 0
binary_sensor:
- platform: gpio
name: Button1
pin:
number: GPIO23
inverted: true
mode: INPUT
on_press:
then:
- cover.open:
id: template_cov
- platform: gpio
name: Button2
pin:
```

```
number: GPIO34
inverted: true
mode: INPUT
on_press:
then:
- cover.close:
id: template_cov

- platform: gpio
name: Sensor1
pin:
number: GPIO22
inverted: true
- platform: gpio
name: SensorGPIO0
pin:
number: GPIO0
inverted: true
- platform: gpio
name: Sensor2
pin:
number: GPIO32
inverted: true
- platform: gpio
id: stall_guard_sensor
name: StallGuard
pin: GPIO2
on_press:
script.execute: stop_at_current_position

stepper:
- platform: tmc2209
id: my_stepper

step_pin: GPIO13
dir_pin:
number: GPIO14
inverted: ${change_direction}

sleep_pin:
number: GPIO27
inverted: true
acceleration: ${acceleration}
max_speed: ${velocity}

script:
- id: stop_at_current_position
then:
stepper.set_target:
id: my_stepper
target: !lambda "return id(my_stepper).current_position;"

globals:
- id: open_position
```

```
type: float
initial_value: ${distance_mm} / (${pulley_diameter_mm} * PI) * 200 * ${microsteps}

cover:
- platform: template
id: template_cov
name: "${device_name} cover"
open_action:
- tmc2209.setup:
current: ${open_current}
stall_threshold: ${open_stall_threshold}
microsteps: 2
- stepper.set_target:
id: my_stepper
target: !lambda "return id(open_position);"

close_action:
- tmc2209.setup:
current: ${close_current}
stall_threshold: ${close_stall_threshold}
microsteps: 2
- stepper.set_target:
id: my_stepper
# 0 Means closed
target: 0

position_action:
- stepper.set_target:
id: my_stepper
target: !lambda "return id(open_position) * pos;"

stop_action:
- script.execute: stop_at_current_position

interval:
interval: 5s
then:
lambda: |-
static auto operation = COVER_OPERATION_IDLE;
static auto position = id(my_stepper).current_position;
if (operation != id(template_cov).current_operation ||
position != id(my_stepper).current_position)
{
ESP_LOGD("main", "Stepper Position is: %d/%d", id(my_stepper).current_position, (int)id(open_position));
if (id(my_stepper).current_position > id(my_stepper).target_position)
operation = COVER_OPERATION_CLOSING;
else if (id(my_stepper).current_position < id(my_stepper).target_position)
operation = COVER_OPERATION_OPENING;
else
operation = COVER_OPERATION_IDLE;
id(template_cov).current_operation = operation;

position = id(my_stepper).current_position;
id(template_cov).position = position / id(open_position);
```

```
id(template_cov).publish_state();
}
```
Copy

note

Be sure 12V power is plugged in to the board

Compile and upload via USB.

Test the board by pressing one of the buttons and motion should begin.

2. Setting Parameters#

You have the ability to set the following parameters of your device

```
a. pulley_diameter_mm:
b. gear_ratio:
c. distance_mm:
d. acceleration:
e. velocity:
f. open_current:
g. open_stall_threshold:
h. close_current:
i. close_stall_threshold:
j. tcool_threshold:
k. microsteps:
l. inverted
```
Copy

To set the parmeters, you need to update the YAML file and compile and upload it each time you want to change a parameter. This can make it very difficult to dial some of these values in.

To fix this, we've added a service that will allow you to test different values without requiring complilation or upload. Once you dial in the correct value, you can update the YAML file accordingly. This will limit the amount of time you spend on compiling and uploading.

To use the service, navigate to **Developer Tools** --> **Services** tab.

From the Service dropdown list, find ESPHome: <YOUR-DEVICE>_control_stepper and select it.

tip

The service called ESPHome: <YOUR-DEVICE>_set_stepper_zero will set the the current position to 0, which is the home position. Call this to reset the home position.

You will see the following:

 Services

Use the service to update values and test them out.

ⓘnote

These values will not save. You will need to update the YAML file with these values.

As noted above, changing the values will not save them. This tool is only used for dialing in the correct value. Once the values are determined, go back to the YAML file in ESPHome and update the values in the file. Save and upload the file.

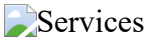
Services Configuration#

To find the correct values for your application, begin by changing the values in the sevices.

Begin by manually moving your device to the home position.

Change your Service to ESPHome: <YOUR-DEVICE>_set_stepper_zero

Press the CALL SERVICE to set your home position

 Services

Now switch the Service to ESPHome: <YOUR-DEVICE>_control_stepper

You will find the following values that can be changed:

target#

This value is in steps. Most stepper motors have 200 steps per revolution. Each step is multiplies by microsteps value below. If using 64 microsteps, that means each motor revolution will require 12800 steps.

If using the small pulley that comes with the string curtain opener, there are 10000 steps per inch when setting the microsteps to 64.

To change the motor direction, change this value to a negative.

speed#

Begin at 5000 which is a slow but silent speed.

microsteps#

Set to 64 unless you have a good reason to change it to something else.

tcool_threshold#

See the StallGuard page to learn more.

stall_threshold#

See the StallGuard page to learn more.

rms_current_amps#

See the StallGuard page to learn more.

YAML Configuration#

Here are the values that you can change in the YAML file

a. pulley_diameter_mm:#

Most motors require a pulley to attach a belt to. Enter the the diameter (in millimeters) of this pulley. A formula will use this value and the distance value to calculate the number of steps to move.

If using a lead screw, set the diameter of the lead screw.

b. gear_ratio:#

If no gear is used, set this to "1".

If using a gear, set the value of the ratio by using an integer. For example, is using a 7:1 gear (requires 7 input rotations to output 1 rotations) set this value to "7".

c. distance_mm:#

What is the total distance, in millimeters, that you need to move.

d. acceleration:#

What is the acceleration value. If using StallGuard, read the StallGuard section before setting this value.

e. velocity:#

What is the maximum velocity value.

f. open_current:#

Set the open current in milliamps. The maximum for the VAL-1000 is 2000.

danger

Be sure to properly match your power supply to the amount of current you are setting.

This value is the Root Mean Square (RMS) current, and is **not** the maximum current. Multiply this value by 1.41 to find the maximum current.

For example, if setting to 2000mA, the max current used by the TMC2209 will be 2000 x 1.41 = 2820mA

In addition, alwasy reserve 500mA for the ESP32 as it may require up to 500mA during WiFi transmission.

In this scenario, you will need at least a 3.4A power supply.

g. open_stall_threshold:#

See the StallGuard page to learn more.

h. close_current:#

Set the close current in miliamps. Note the warnings in the open current section.

i. close_stall_threshold:#

See the StallGuard page to learn more.

j. tcool_threshold:#

See the StallGuard page to learn more.

k. microsteps:#

Set the number of microsteps. This is not as straightforward as it may seem so please read.

Set to 64 if you want it to work well. Keep reading to learn more.

The TMC2209 has a feature called Microplyer that automatically turns low step resolutions into 256 microstepssteps. For example, if you set the above value to 1, each step pulse will automatically get turned into 256 microsteps. The downside is that accuracy also goes down, as you cannot set your position to any of the microsteps that microplyer creates.

On the other hand, setting this value to 256 will limit the top speed of the motor because the ESP32 simply cannot generate pulses fast enough to move the motor very fast.

The value 64 is a good sweet spot that allows the TMC2209 to run at its top speed while also remaining as accurate as possible.

l. invert motor direction#

If you need to change the direction the motor is turning, change the "inverted" value, under the dir_pin to True or False, depending which way it needs to spin.

stepper:

- platform: tmc2209 id: my_stepper

  step_pin: GPIO13 dir_pin: number: GPIO14 inverted: True

Edit this page
Previous
« StallGuard
Next
VALAR W1: Window Opener Setup Instructions »

- 1. Add New Device
- 2. Setting Parameters
- Services Configuration
  - target
  - speed

Docs

- Tutorial

Community

- Stack Overflow
- Discord
- Twitter

More

- Blog
- GitHub