

## Mini Stepper Gauge Config

### PREDEFINED CONFIGURATIONS:

Here you find some pre defined gauges, just to give you something to start with. Click the radio button and the values will be loaded to the tables. If you have "saved" your own setup, the first option (Saved Configuration) will load those values back for you.

- Saved Configuration (The data previously saved to the MCU)
- Speedometer 0-260Km/h
- Diesel Rev. counter 0-5500rpm
- Turbo pressure (BAR)



### PINS ON GAUGE STEPPER:

The pin assignment for the Stepper motor is for direct drive of the small "Gauge steppers" used in instrument clusters in modern cars. The ESP32 can not drive any larger steppers without amplifying the output power on the pins. Check the data sheet for your stepper to get the coils connected correctly. If the stepper moves the wrong way you can swap two of the pins to reverse the stepper rotation.

For the D1-mini ESP8266 module from Are-Inc. the pin settings are:

#### Coil:

- A1: 13 (D7)
- A2: 14 (D5)
- B1: 12 (D6)
- B2: 4 (D2)

#### Initial Homing.

Here you add the number of steps the stepper needs to take to be certain that it has reach the most left position (against the mechanical stop).

#### Enable Sweep.

When "checked" the gauge make a run from the lowest value to the max value and back to the lowest value again when powered up, then stopping at the low value. (Mimic the "cool" gauges from Autogauge...) It will use the values that is stored in the "Speed points" further down in the setup page.

#### Sweep Speed.

This is the steppers maximum step speed during homing. Default number is around 800-2000

#### Sweep Acceleration.

This is setting the acceleration of the needle when homing. Default number is around 3-500.

#### Max Steps.

This is the maximum steps the stepper should be allowed to take from its "Zero" position. (Many of the gauge steppers has 600 steps from end to end) low as possible but still handle the full acceleration of the car. Default number is around 800-1000

### Trim Value Speed Plus.

If the stepper reach "Zero" when homing and the needle is to far "down" below "Zero" you can trim this position by adding number of steps to rise the needle to the 0 position on the gauge face. (You can use the "Manual control" at the end of this page to test how many steps you need to add.)

### Stepper Max Speed.

This is the steppers maximum step speed. If the stepper is set to quick you might get problem when doing the "Home" sequence. You want the stepper as slow as possible but still handle the full acceleration of the car. Default number is around 800-1000

### Stepper Acceleration.

This is setting the acceleration of the needle. You can test different settings to get a smooth move of the needle. If you stepper is weak and struggle to move the needle without missing steps, you can try lower this value. Default number is around 3-500.

## INPUT FILTERING:

### Input Filter %.

"The gauge uses an adjustable first-order IIR (Exponential Moving Average) filter to smooth the input signal."

Your gauge reads live data — like speed, RPM, or boost — many times per second. But real signals aren't always calm; they can "jitter" or bounce up and down due to sensor noise or fast data updates. To make the needle move smooth instead of twitchy, the gauge applies a digital filter. The filter works a bit like adding weight to the needle:

- a light needle (low filtering) reacts fast but can wiggle,
- a heavy needle (high filtering) moves smoother but lags slightly behind real data.

### Filter blend %:

Controls the overall stability of the gauge. A higher value (e.g., 80-100%) makes the gauge less responsive but smoother, while a lower value (e.g., 50-70%) makes it more responsive to changes in the input value. Adjust this to find the balance between smoothness and responsiveness that works for your needs.

### Filter Smoothness %:

This fine-tunes how the filter behaves depending on the size of the change. Small input changes can be made more responsive, while big jumps can be dampened for stability. Think of it as a dynamic sensitivity control.

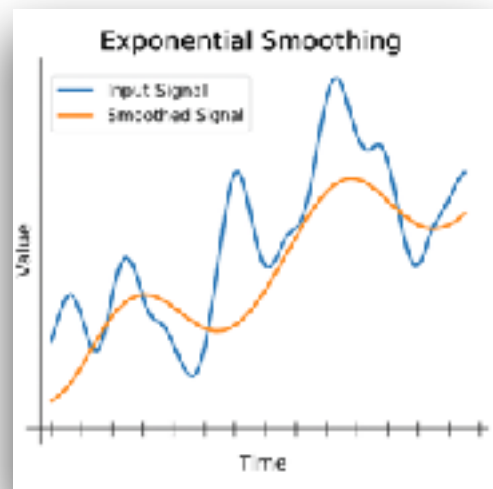
Higher values (1.3–1.8) → more responsive to tiny fluctuations.

Lower values (1.0–1.2) → smoother, slower response, better for steady readings like "temp".

Together, these two parameters let you "tune" the gauge feel:

A fast, twitchy boost gauge might use 60% / 1.5

A slow, elegant temperature gauge might use 90% / 1.1

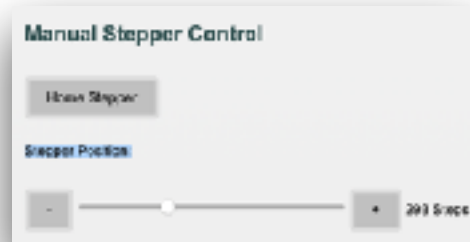


### **Speed Points (comma-separated):**

This is part of a lookup table, converting steps to input data. There is a "interpolation" between the numbers you add here, the more number you add the smoother and precise the needle will be on the dial. The numbers are added with comma separation and are matched to the input data.

The speed points is your input, so for a speed input of 100 then that is what you add in this , separated line.

To get the exact matching step count for that input (100) then you can use the "Stepper Position" slider at the bottom of the page. Move the slider until you are exactly where the needle show your value (100) and read the step count on the slider.



### **( REMEBER TO HOME THE STEPPER FIRST!! )...**

Ex.

# Speed points (comma-separated, in one line)  
0,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100

# Step points (comma-separated, one line)  
0,4,8,12,17,21,26,30,35,39,44,48,52,57,61,65,70,74,78,83

Where Speed points are the value of the incoming data. And for each value here you add a matching "Step" number in the textbox for "Step points". The incoming data values in between your added data will be interpolated by the code to make for a smooth needle movement.

## **Manual Stepper Control.**

### **Button "*Home Stepper*".**

Clicking this button put the stepper motor in the "Zero" position @ 0 degrees. From here the stepper can move the full "315 degrees" for a standard gauge stepper. or whatever amount of freedom the stepper you use can do.

### ***Stepper Position.***

Using the slider moves the stepper "live" so that you can control the needle on the dial manually and on the left side of the slider you find the number of steps it took to reach that position.

Use the number to fill the "Step Points" box for steps and add the accompanying number on your gauge dial in the "Speed Points" box.

Remember to add comma (,) between each number as in the example above.

Use the slider for quick and corse adjustment and then pick "Step by Step" using the (+) and (-) buttons on each side of the slider.

## Gauge(s) RGB Color Picker:

### RGB-strip Output Pin:

Set the pin number for the data out to RGB-strip. We use the (ArduinoIDE numbering), so use the number shown on your dev. Board, lik D2, D4, D8 or alike. But only enter the number Not the letter "D"...

### INSTRUMENT CLUSTER BACKLIGHT

#### Select color:

In the field "Color (RGB): " you enter the RGB number for the color you want your instrumentation backlight to be.

Id you dont know the RGB code , you can click the coloured square and an "Color Picker" will pop up. Click the color you want and the code will be filled in to the field.



#### No. of BCK-leds led(s):

Enter the number of leds in the strip used for backlight. The system is setup to use the first leds in the strip for backlight, (*or the first LED's in the chain*) here you enter the *number of leds you use for the backlight*.

It starts with the first LED in the chain and continues for the amount of LED(s) you enter here.

### INDICATOR LIGHTS.

Indicator Lights continues from the chain of backlights and forward.

EX: .. If you have 8 Back light LED's, the Indicator light starts at 9 —> ...

The Indicator lights can be used for whatever warning/indication you like, ex. BeamLights, Turn signals, Oil-pressure, Charging, water temp a.s.o. ...

- Acc. 1: Color: rgb(255, 38, 0) Led No. 9
- Acc. 2: Color: rgb(255, 38, 0) Led No. 10
- Acc. 3: Color: rgb(255, 147, 0) Led No. 11
- Acc. 4: Color: rgb(4, 51, 255) Led No. 12

In th Example above we assume you got 11 backlights, and therefore the Indicators got the number 9, 10, 11, 12. You st the color of the LED the same way as for the back light. Enter the RGB color code or click the little square and use the color pick that pops up.

As you seen this code is fre to use and can be downloaded at our homepage. It is easy to add more "Indicator Leds if needed., Just copy the code from the earlyer and change a few variable names. (*You will find more on this in the programming manual.*)

#### LED-brightness

Here we set the full power brightness for the LED(s) in the strip/chain.

This settings will be for both the backlights and Indicator lights.

We also set a "dimmed" level that is used for example when the Headlight is turned on in the evening when its dark...

**Full Brightness:**

Enter a value between 0 and 255. Where 255 is full brightness and 0 is “off”.

**Dimmed Brightness:**

Enter a value between 0 and 255. Where 255 is full brightness and 0 is “off”.

## Using the LED strip(chain)

**Connecting the LED's:**

The ESP is not powerful to supply the power for the LED's by it self, so you will need to create your own power source for the LED's. From the selected pin of your choice and to the D-in on your LED-strip, or LED chain.

The code used for controlling the RGB Led's is the “Adafruit\_NeoPixel” library so if you read about that library you can get information what type of LED's that is supported.

**Commands:**

To operate the LED's we use Serial commands.

You will need to create a “HUB” that speak to your systems buttons and inputs, then sends serial commands to the mini-gauge that controls the LED's.

The following commands are used to control the LED's:

bb0 = Turn backlight off

bb1 = Turn backlight on

lbh = Set brightness level to the High setting.

lbl = Set brightness level to the Low setting.

la0 = Turn Indicator LED\_1 off

la1 = Turn Indicator LED\_1 on

lb0 = Turn Indicator LED\_2 off

lb1 = Turn Indicator LED\_2 on

lc0 = Turn Indicator LED\_3 off

lc1 = Turn Indicator LED\_3 on

ld0 = Turn Indicator LED\_4 off

ld1 = Turn Indicator LED\_4 on

To test this you can start a serial terminal and have the USB cable connected to your MCU. Then just send these messages in the terminal to see the LEDs turn on/off..

The baud rate is set to 115200 in the example code.

This is a very simple and “basic” function, and you can alter it to use the ESP-NOW protocol, or SPI, or if you use a ESP MCU with free pins, you could just hook the buttons to pin inputs via a simple voltage divider.