Build your own CO2 sensor as a COVID-19 aerosol risk estimator

Arnold Niessen, 4 January 2021, arnold.niessen@gmail.com.

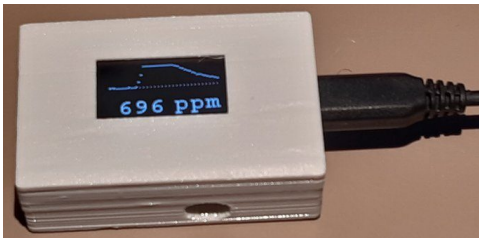
Project files (design files, firmware images, shopping list with links) are available on github [1].

Summary

This guide explains how to build an affordable and portable CO2 sensor.

CO2 sensors can be useful to measure CO2 levels which can be used as a proxy to estimate the risk of COVID-19 infection in indoor environments. Proper ventilation is required to keep CO2 levels low. A CO2 sensor may help to balance cold and risk in winter.

This CO2 sensor has:
-an OLED display for CO2 measurement, and a graph of the past hour (default graph range is 400-1000ppm, dashed indicator line at 600ppm); the graph doesn't shift, but cycles like an ECG graph),
-power over micro-USB,
-a LED with blinking pattern depending on CO2 level (thresholds at 600, 800, and 1000ppm),
-battery version (shown below) has on/off switch and voltage indicator for battery, and
-is light and portable: ca 20x35x55mm without battery, or ca 20x35x84mm with battery.



Picture left: battery version, used on a poorly ventilated toilet, showing CO2 level reaching ~1000ppm in a few minutes when occupied, and reaching ~500ppm in a 40 minute time frame afterwards.

Picture right: minimal version without battery or LED

Its measurements may not be very accurate, but good enough for making quick and approximate CO2 measurements, and making ventilation decisions based on the CO2 status and/or CO2 trend.

It uses an ESP8266 board, an 0.96" OLED display, and an NDIR CO2 sensor (either Winsen MH-Z19B (preferably 0-2000ppm), Senseair S8-LP, or a dual-channel SCD30). Depending on sensor and other choices, total cost is €25-80. It also has wifi remote configuration and read-out capability and it can integrate using http or mqtt with Domoticz, OpenHAB, EmonCMS, and others, but such customization is optional and beyond the scope of this guide.

In its simplest form (without battery) it requires only 3 elements: MCU board, OLED, and sensor, connected with 4-6 pin headers and 4 wires, and can be built in half an hour. If you do not have the skills to build one yourself, perhaps you can find someone who can. If you can build one yourself, perhaps you can build devices for others who cannot.

There are already many DIY instructions for building MH-Z19B based CO2 sensors. This particular design aims to minimize size and build time, and provides options from simpler to more complex versions.

Credits

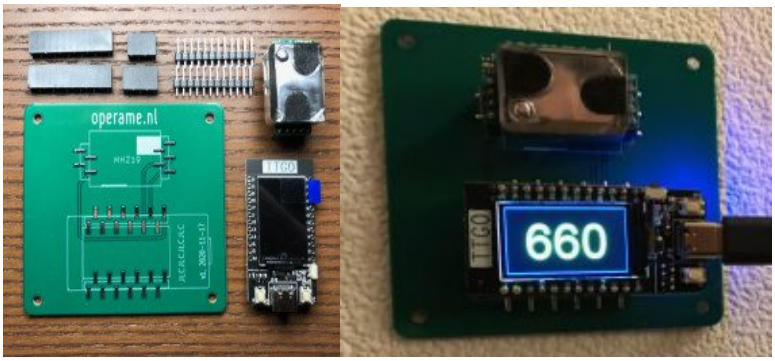
This work was inspired by the build guide from Stefan van der Eijk [2] and the aerosol research work by many qualified researchers, including the authors of [3] and [4], and further motivated by (or triggered by the frustration of) the total lack of adequate government strategy and the continuing denial of aerosol risks or face mask efficacy in several countries, including the Netherlands.

Disclaimer

No guarantees provided! Only follow this guide if you know what your are doing and understand all of the risks involved, in particular the risks of Li-ion battery-powered devices.

Alternatives

Another interesting affordable design aimed at school class wall mounting is the Dutch project “Operame” [5] (sounds as “OpenWindows”) aiming at school classes. A DIY kit sells for €28.



Operame wall-mounted CO2 sensor.

Yet another approach technically similar to this guide is [6].

Alternatively you can buy one of the commercial sensors, in the price range of € 100-200 [7]. For example, the well-known Aranet4 is optimized for power consumption and runs a long time on a set of batteries. The Aranet4 also uses the Senseair S8-LP sensor.

Transmission modes and risk of aerosols in SARS-CoV-2 Coronavirus transmission

I am not an aerosol or Coronavirus expert myself, but I do read and listen carefully to the experts. Depending on who you ask, the main transmission modes of the SARS-CoV-2 coronavirus are close contact (by droplets and/or more likely by aerosols), and/or not-so-close contact (>1.5m / > 6 feet) by aerosols, especially in closed environments with poor ventilation. Aerosols are micro-droplets, up to 100 µm diameter, which can linger in air for hours and remain infectious, and which are generated by talking, singing, shouting, and also by just breathing. For transmission by aerosols, you do not even need to be in the same room. If you enter a room, a bus or a toilet, previously used by an infectious person, you are also at risk. If you breath air which originates from another room, you are at risk.

Few scientists, unfortunately some of them determining government strategy, still believe that aerosols cannot be a relevant transmission mode because of two reasons: they state that R0 is only around 3, unlike airborne measles having a much higher R0. They also mention that physical distancing has

reduced R0 and conclude that larger droplets must be the main transmission mode. Personally, following many expert scientists in this field [3][4], I believe this reasoning to be flawed. Aerosols disperse and are more concentrated near a person, so aerosols could be and are likely responsible for close-contact transmission. Droplets are produced only by coughing or sneezing (or incidentally by "talking with production"), so mainly by symptomatic persons, and hardly by pre-/asymptomatic persons. Droplets may thus in certain circumstances contribute to close contact transmission. The observation that R0 for SARS-CoV-2 is not as high as for measles is not a valid argument to deny aerosols as an important transmission route. Meanwhile R0 seems to increase with newer virus variations. Especially the large variation in viral load between persons may explain why aerosols could provide an important role in transmission: many people do not infect anyone, whereas a subset of people infect many by both aerosols and droplets.

In addition, newer versions of the SARS-CoV-2 coronavirus seem to have higher R factors, which also undermines the "but R is low" argument.

While the debate on the relevance of these transmission modes continues, it is better to be safe than sorry, and to assume that, now that we physically distance and limit group sizes, there is potentially a substantial aerosol-based transmission over distances larger than 1.5m/6 feet.

As a side note, given that aerosols may dominate close-contact transmission by pre-/asymptomatic persons too, please use a proper facemask (preferably FFP2), preferably together with a face shield, but never use a face shield without a facemask.

Most aerosol-based transmission boils down to breathing someone's (unfiltered) exhaled air. The risk of doing so can be estimated by: (1) measuring how long you do so, and (2) measuring the CO2 content in the air you breathe. There are three limitations with CO2 measurements: (1) the measurement includes the CO2 produced by yourself, (2) HEPA filters may reduce aerosols but not CO2, and (3) the risk of catching the virus depends not on CO2 per se but on how many virus particles someone exhales. Different persons may produce different amounts of virus particles based on their viral load and their behaviour. Nevertheless, it is clear that the aerosol risk increases with higher CO2 values. The natural background level of CO2 is around 400ppm. When entering a room with 800 ppm CO2, around 1% of the air is second-hand exhaled by someone else. CO2 measurements can be used as an additional fencing mechanism to reduce your risk of catching this virus especially from a-/presymptomatic persons.

Licenses

The legal but, for some, boring stuff…

upon request, access to copy the corresponding Tasmota source code from a network server at no charge. To ensure you can provide such access later, we advise to make a fork of the CO2 branch of the Tasmota project at [1] on your own GitHub account (just in case my account would disappear). If you accompany the CO2 device with a link to your or my GitHub repository it is unlikely that anyone will take you up on the written offer.

Disclaimer

Any use of this guide and the CO2 sensor device is entirely at your own risk. It contains a Li-ion battery, which needs to be handled carefully to avoid any risks. The safety of the device relies only on the current-limiting circuit included in the battery. Don't proceed if you don't understand or accept the risks.

Which sensor to buy?

The MH-Z19B sensor is slower and less accurate, but auto-calibration can be switched off, so it does not need regular fresh air. However, long-term stability without auto-calibration may be a concern.

The S8 LP sensor seems to be more accurate but auto-calibration cannot be switched off, so regular fresh air is required in operation.

The Sensirion SCD30 seems to be the best choice but it is more expensive.

A more detailed first impression of these sensors is available as Appendix.

Requirements for this project

More details on purchasing options are provided in a separate shopping list in [1].

This project requires

-soldering skills and equipment, tools, some small wires (for convenience, breadboard jumpers can be used here)

-an NDIR CO2 sensor: a Winsen MH-Z19B (0-2000ppm without cable, bare sensor only), a Senseair S8-LP (004-0-0053), or a Sensirion SCD30 (the SCD30 is a dual-channel NDIR sensor)
-a 0.96"OLED I2C (white) display; unfortunately the life time of OLEDs is limited, image quality of this display will degrade after 1-2 years of use
-an ESP8266 MCU programmed with Tasmota; this project uses the NodeMCU V2 Lua 4MB wifi module (the small version, with the ESP8266 mounted directly on the PCB, not cased)
-pin headers, usually included with the MCU board and OLED display
-3D-printed case (bottom and top), optionally 3D-printed solder tool.

-optional: a small PCB to strengthen the micro-USB connector and 2x2 additional header pins; a 3mm LED can be mounted on this PCB; and for the battery version a switch can be mounted on this PCB
-optional: a 3D-printed casing (different versions for with/without battery and for different CO2 sensors)
-optional: a Li-ion battery 25x30x8 mm (make sure it has a protection circuit against over-current, over-discharge, and over-charge included), and a battery charger/DC-DC circuit, and optionally a 120k Ohm resistor to monitor the battery voltage
-optional: a 1k resistor and a 3mm LED; LED may indicate when CO2 exceeds a settable threshold

-a micro USB cable with a power supply or a power bank to power the sensor and/or to charge the battery

-software: Tasmota for the MCU, and a script to run under Tasmota; the script is included in the preconfigured flash firmwares

Suppliers

Please support your local suppliers, if possible, even if they are sometimes a bit more expensive than aliexpress. Local suppliers provide employment and pay taxes, thereby contributing to society, and deliver much faster. However not too many local suppliers seem to sell the CO2 sensors or MCUs [9]. Alternatively, you can buy parts from aliexpress and similar channels. However it is strongly advised to buy the ESP8266 from a reputable source such as Robotdyn or one of their resellers. Clones can be non-functional, have fake memory chips, or have a misconfigured WiFi antenna. Some Wemos-stamped clones and especially HW-628-stamped clones are to be avoided. Some boards have solder bridges and can be repaired, others cannot. These boards are cheap, so consider buying a spare one. But do start programming and configuring the device before you solder the sensor on it (yes, I learned from experience)!

1. Flash Tasmota to the MCU

The ESP8266 requires a good USB cable and power supply. Many people report issues with bad USB cables.

For Windows or Mac you may need to install the CH340G driver. Links are included in [2]. To flash, follow the instructions on [10]. On Linux I use esptool: “esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash -fm dio -fs 32m 0x00000 tasmota_SENSORNAME.bin”.

The Tasmota software is hosted at [1]. If you use any of the pre-compiled binaries, all configurations are ready for immediate operation, including a script to read the sensor value and display on the OLED. There are four binaries available, select the right firmware for your sensor: Winsen MH-Z19B (two versions: self-calibration on or off), SenseAir S8, or Sensirion SCD30.

If you want to change any of the settings, you will need to either build your own binary with your settings preconfigured and flash that one instaed, or to flash one of the provided images (use the compressed versions) and then configure Tasmota over WiFi.

For compiling your own Tasmota image, [11] is very useful.

2. Optional: configure Tasmota for/over WiFi

There is no need to configure Tasmota if you are fine with the default settings and do not want WiFi functionality. In that case, you may skip to step 3.

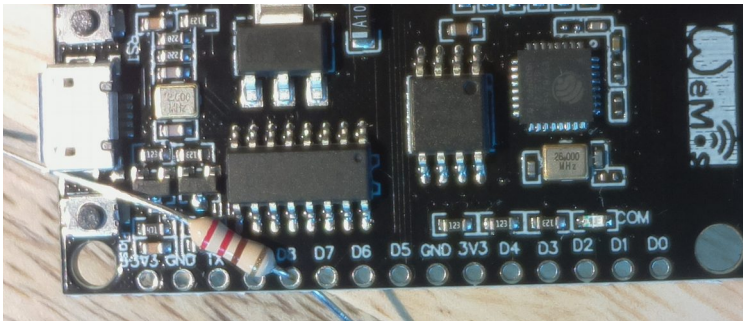
If you did not recompile and flash the MCU with your WiFi credentials preset, the MCU will set up an access point (SSID: Tasmota_*) to enable configuration of wifi settings, but only for a few minutes. Switch your phone or PC to wifi access point Tasmota_*, connect to http://192.168.4.1, and enter your wifi credentials. Setting up wifi can be a hassle and fail – just try a few times and use a decent USB cable and supply. When successful, the MCU will connect to your wifi network and the Tasmota_* access point will disappear. You may find the MCU’s IP address using an app like Fing and find the Tasmota’s

web interface. The device allows a.o. reconfiguration of WiFi and MQTT settings, and editing of the main script.

The device can also be reflashed over WiFi. Due to flash memory limitations, an intermediate minimal-size firmware must be flashed before a newer image can be flashed [12].

3. Optional: prepare a resistor for optional LED

Assuming you add the optional PCB in step 6 below, this is the proper time to insert a 1k resistor into pin D8 and bend it towards the USB connector. If you want maximal brightness, use a 220 Ohm resistor instead.

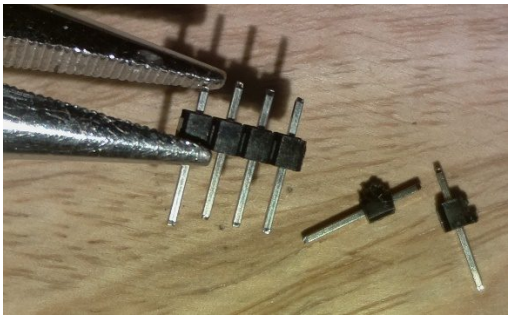


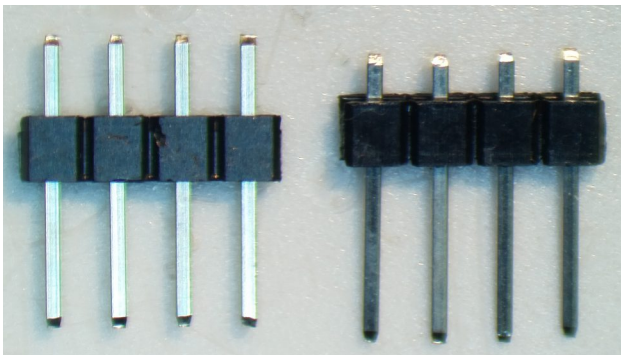
4. Adding OLED display

This guide solders the OLED display directly on the MCU board. This reduces build height to approximately 3.5mm between both boards. Unfortunately this makes it difficult to replace the OLED if it is broken or past its lifetime (10000 hours = ~1 year full-time use). Alternatively you can use pins+headers to make the OLED display removable and replaceable, but this would increase building height by 8mm and require modification of the 3D-printed case.

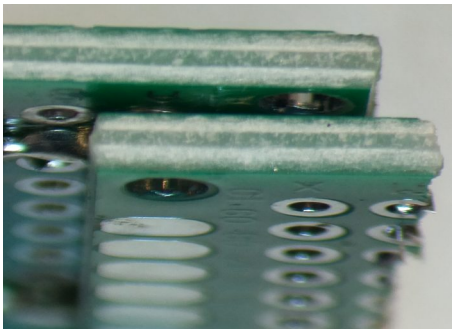
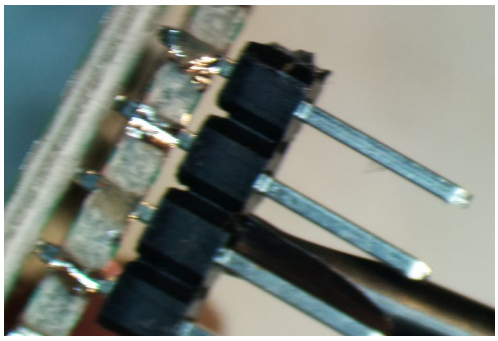
Separate and modify 4+1+1 header pins using some pliers, such that they will be flush with the top of the display PCB after soldering (or they will conflict with the case).

If you use the Sensirion SCD30 CO2 sensor, positioning is critical: the SCD30 sensor uses the same 4 pins as the OLED display; the pins are 11.3mm long and the pins should extend not more than2 mm above the black block to allow for as much as possible extruding length for the SCD30 sensor. If you can find slightly longer pins this would actually be preferred.

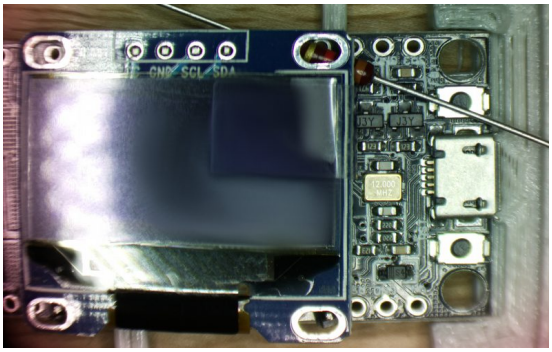


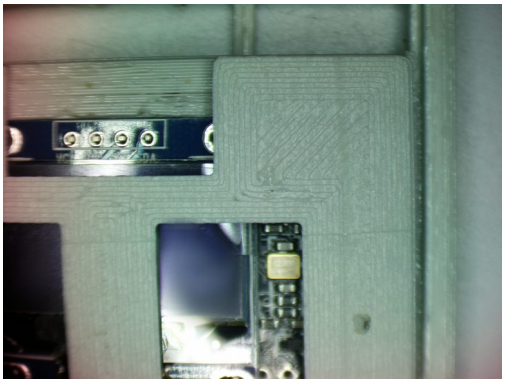
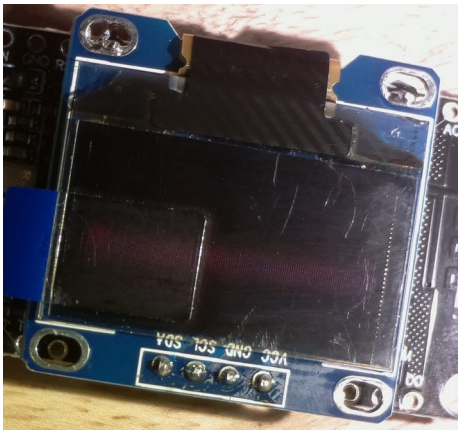


I made a small tool of 2 PCBs, shifted such that the holes mis-align, with a minor gap in. Using a screw driver or your thumbs the pin header can be modified easily and more accurately without the use of pliers.

Now put the long part of the header pins into the MCU board on locations [3v3 GND D5 D6], [NC], [EN], and put the OLED display on top of it. Be careful not to break the thin glass from the OLED display (it is not supported near the lower corners). The pins should not be much higher than the display itself. Try to keep the display parallel to the MCU PCB (some components of the OLED display may interfere with components of the MCU PCB – make sure they don't make electric contact). There is a 3D-printable support tool that is helpful for this step. Solder all 6 pins to both boards (OLED side first). Afterwards, cut the extruding pins on the MCU bottom side (but do NOT do this if you use the Sensirion CO2 sensor, which requires the extruding pins!).



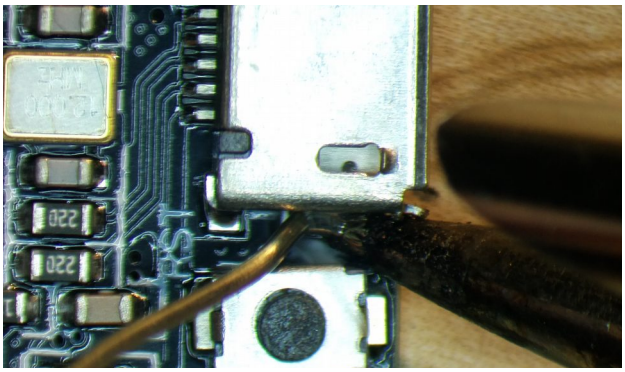



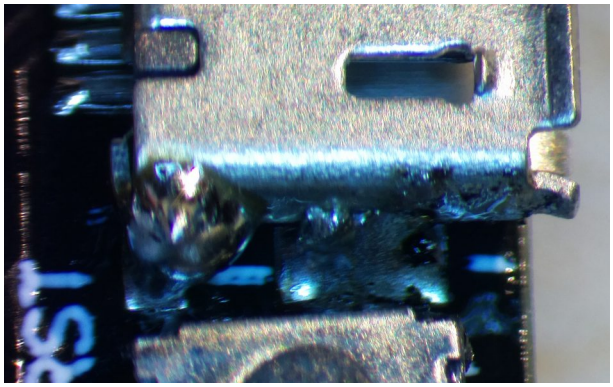
5. Optional: strengthen the USB connector, part 1

Unfortunately the micro-USB is soldered on top of the MCU board and not through-hole. The connector breaks off easily, so be gentle when (dis)connecting!

Its strength can be slightly improved by adding a bit more solder on the sides of the micro-USB connector. Be careful not to use too much solder as solder on the inside of the socket may block the insertion of a USB plug.

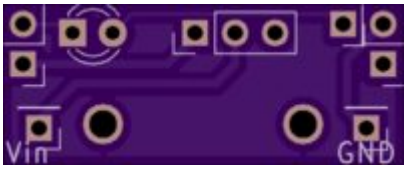
For the battery-powered version this micro-USB connector is used only once for flashing – afterwards it is not needed anymore so this step can be skipped.



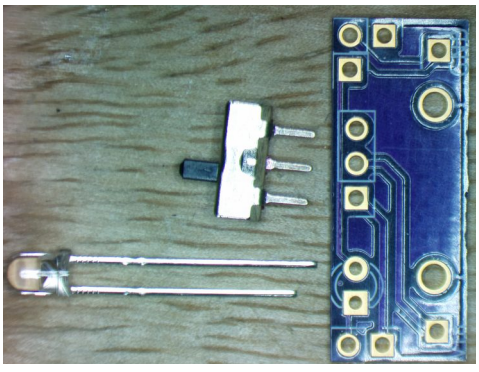


6. Optional (but required for battery version): strengthen the USB connector, part 2, using a PCB which also mounts an optional LED and an optional power switch (switch is required for battery version)

A small PCB can be soldered immediately on top of the micro-USB socket for additional mechanical support; this reduces the risk that the micro-USB socket breaks off. This PCB also provides for a 3mm-LED and, for the battery-powered version, a small switch. The kicad file is available on GitHub.



Optionally, solder a 3mm LED directly on top of the PCB. The LED is mounted straight on the PCB (no distance!). Solder quickly to avoid damage to the LED. The longer wire is pin 2, the shorter wire is pin 1.



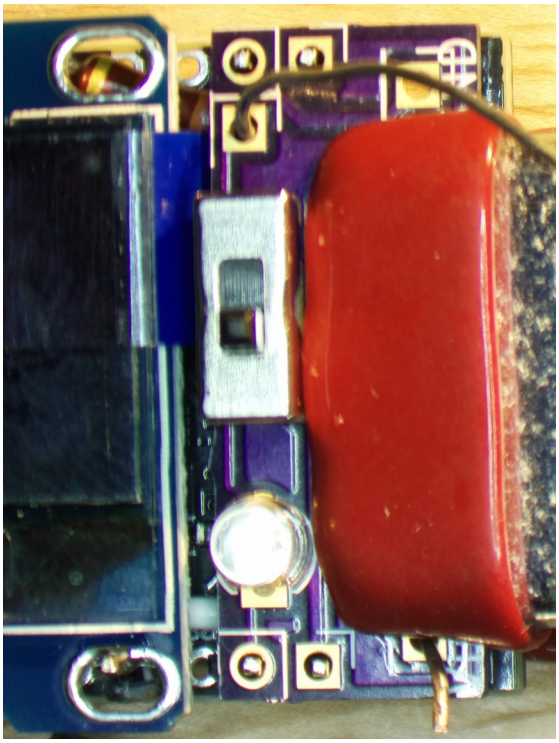
For the battery version solder a switch too – try to solder it straight and flush to ensure it fits in the holes of the 3D case. Make sure the wires don't extend too much, cut or angle them.

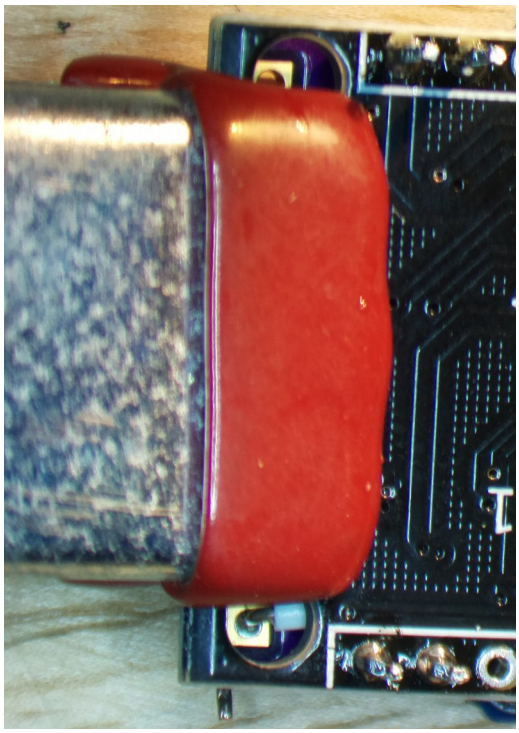
6. Optional: solder PCB over the micro-USB plug, optionally with resistor and resistor

It is a bit tricky to mount the small PCB using 2x2 pin headers (modified in the same way as for the OLED), and the resistor from step 3. The resistor should be motivated to fit the tight space in between; some convincing may be necessary using the force of pliers (drawing the resistor into place).

For the battery version, a wire should be soldered to Vin; this 2" wire is to be inserted between the MCU board and the OLED display.

A clamp may be very useful here.



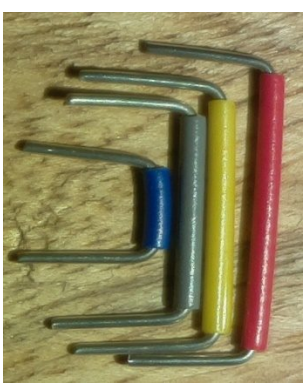


7a. Add CO2 sensor (Winsen MH-Z19B version)

Optional: gently break the socket from the MH-Z19B CO2 sensor using pliers. This makes soldering easier.

Prepare 4 wires for the CO2 sensor. It is easy to use breadboard jumpers of 0.2", 0.4", 0.5", and 0.6". The colors used here are related to the jumper size and are very confusing:

-GND = red = GND
-Vin = blue = VN
-D1 = yellow = Rx
-D2 = grey = Tx



Reduce the size of the blank part on one side and solder these 4 wires to the CO2 sensor. Soldering should be done quickly to avoid heat damage to the sensor device.

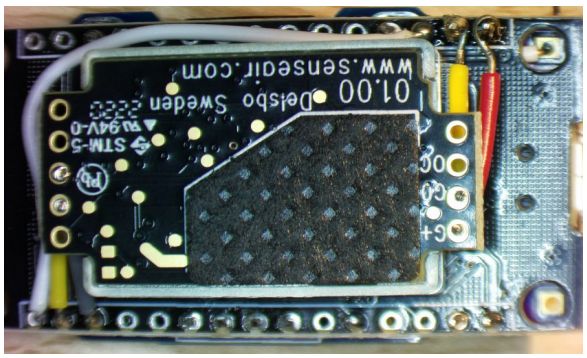
Mount the CO2 sensor on the MCU PCB, using some double-sided adhesive, and solder the 4 wires to GND, Vin (either on the extruding pinheaders, or into the open holes, again, red=GND, blue=Vin), and to D2 and D1.

7b. Soldering the CO2 sensor (Senseair S8 version)

Prepare 4 wires for the CO2 sensor. It is easy to use breadboard jumpers of 0.4”, 0.5” (twice), and 0.6”. The colors used here are related to the jumper size:
-GND = yellow = G0
-Vin = red = G+
-D1 = yellow = Tx
-D2 = grey = Rx

Reduce the size of the blank part on one side and solder these 4 wires to the CO2 sensor. Soldering should be done quickly to avoid heat damage to the sensor device.

Mount the CO2 sensor on the MCU PCB upside-down, symmetrical with respect to the 4 connections on the MCU board (the solder tool may provide some assistance here), and solder the 4 wires to GND, Vin (either on the extruding pinheaders, or into the open holes), and to D2 and D1.



To avoid physical stress on the wires, a bit of hot-glue can be used to blue the sensor to the MCU board.

7c. Soldering the CO2 sensor (Sensirion SCD30 version)

In theory the Sensirion sensor (using pins 1-4) can be soldered directly on the extruding pins below the MUC board from the OLED display. However, the SCD30 specification recommends 1.5mm clearance between the sensor and the MCU board. The result doesn't fit nicely: the sensor extends 0.2mm to the side and 5mm in length direction, and the sensor needs to be soldered at an angle as the extruding pins are just not long enough.

A better solution, work-in-progress, is to add a small additional PCB to improve positioning of this sensor.

8. Test subcomponent

It is time to test the subcomponent of OLED+MCU+sensor: gently insert a micro-USB cable connected to a proper USB power supply. After ca 6 seconds the OLED display should draw a dashed line. The S8-LP or SCD30 sensor will provide a first measurement within a few seconds. The MH-Z19B sensor takes a minute or longer before a valid measurement is available. It will first report 0 ppm, after a minute 410 ppm, and only after another minute start showing actual PPM readings. The initial readings are suppressed by the script and the OLED will show "---" during initialization.

9. Calibration (MZ-Z19B only)

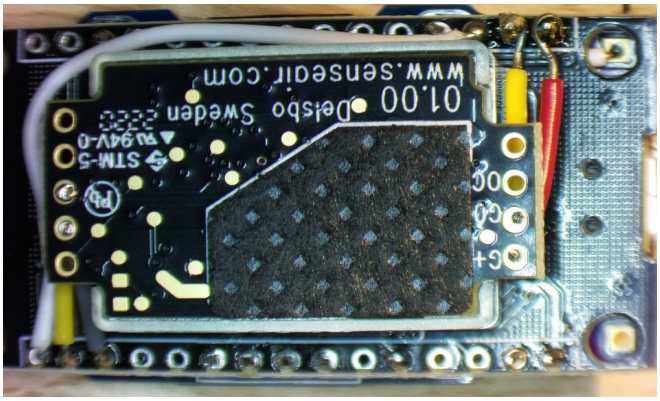
Unless you flashed the self-calibration image, set-up a WiFi connection, put it outside for half an hour preferably around noon, and manually calibrate it from the Tasmota console with the command "Sensor15 2".

10. Decision point: add battery?

If you don't want a battery-powered version, you may skip to step 17.

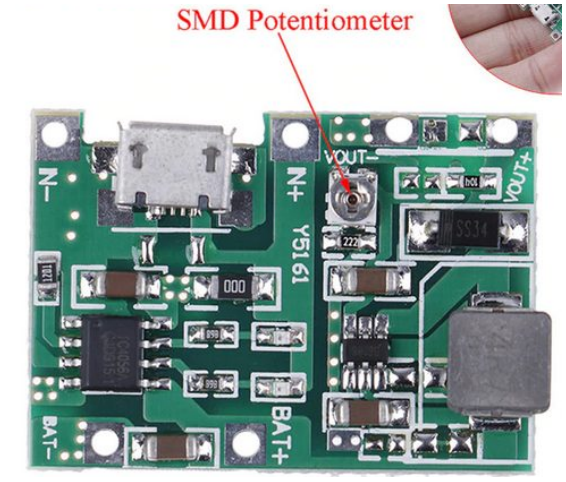
11. Optional: wire RST to D0

The CO2 sensor runs a few hours on a battery. To save battery, Tasmota provides a deepsleep option. Deepsleep requires a wire between D0 and RST. Deepsleep is not supported by the current script, but anticipating any future support for it, such a wire could be added. The (white) wire is already shown in step 6b above.

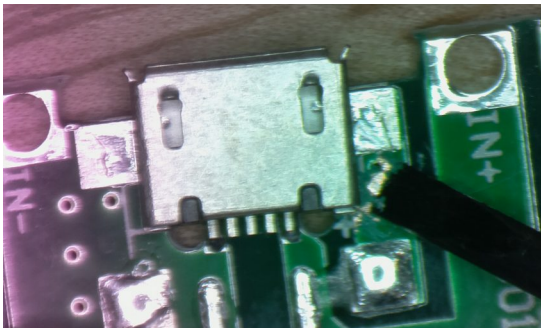


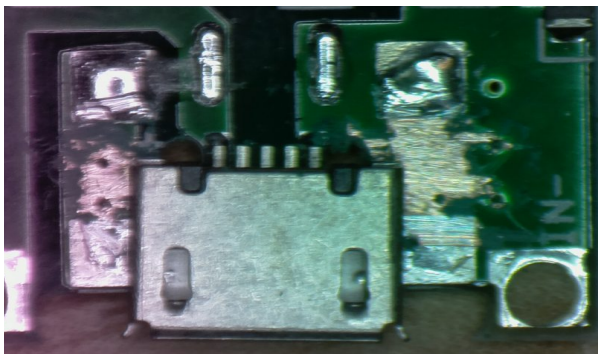
12. Battery version: prepare charge circuit/DC-DC converter

For this project I selected a small and cheap PCB which combines a TP4056 Li-ion charger IC and a MT3608 DC-DC step-up converter. It appeared there are two problems with this board (and many other MT3608 designs). First, this board also has a poorly mounted micro-USB socket (not through-hole). Second, the output voltage specification is stated to be 4.3-27V, but this is wrong – it is 5.6V-27.9V due to wrong component values. The minimum output voltage is too high for the CO2 sensor!

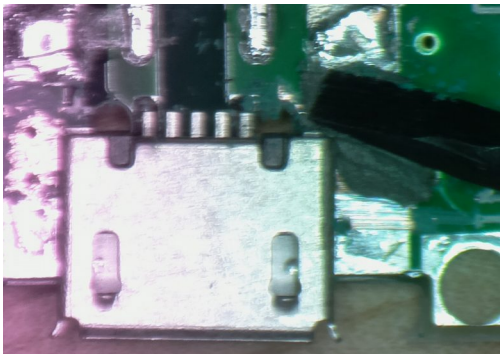


The micro-USB socket can be improved by manual soldering and/or reflowing. First, scratch the soldermask layer near the socket. Then, manually solder the socket.

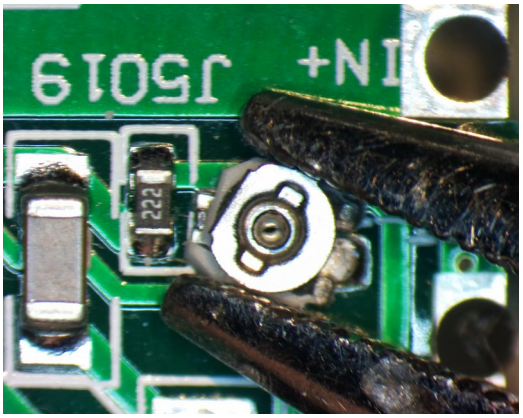


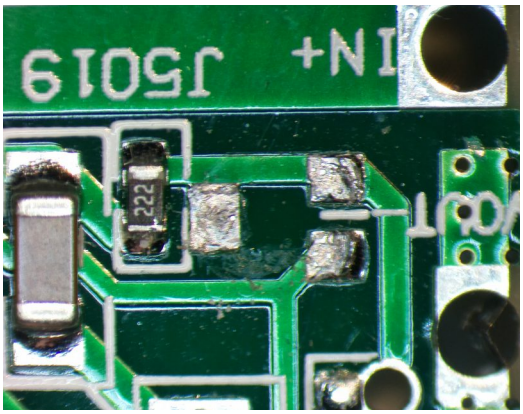


Alternatively, and giving a better result, you can add some solderpaste and use a reflowr (but take care to put only half of the circuit on the reflowr – or disconnect the thick ugly solder blob near the inductor will disconnect).

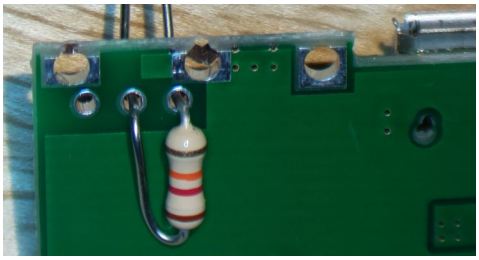


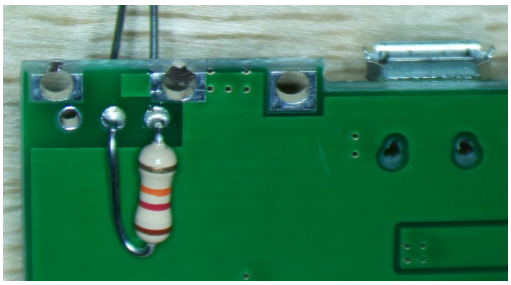
The incorrect output voltage can be repaired by replacing the SMD potentiometer by a 12kOhm resistor. This results in a 4V8 output voltage. The SMD potentiometer can be de-soldered. Alternatively, it can be removed using force (sideways!), with pliers. Afterwards, check the traces for any damage.





A 12 kOhm resistor should then be soldered below the board.





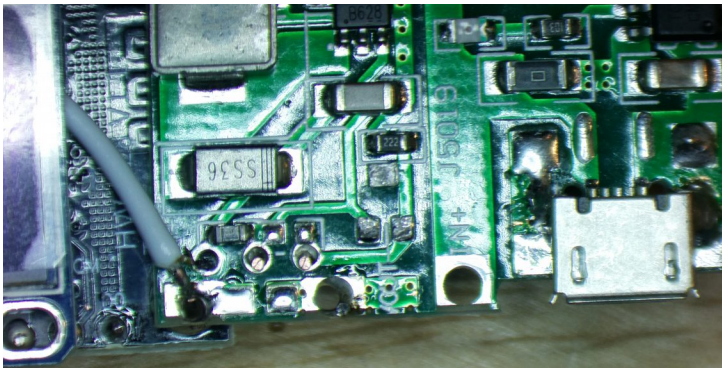
13. Verify output voltage

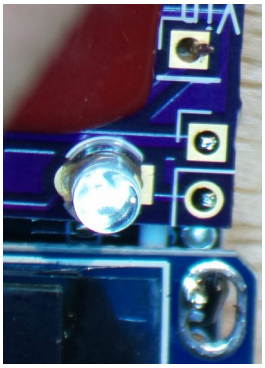
Before connecting the charge circuit to the MCU, verify that it outputs 4.8V (on when powered by the micro-USB plug. If the voltage is wrong, the wires may have been damaged when removing the potentiometer and should be repaired.

14. Connect Vin and GND from MCU to DC-DC converter, and optionally a voltage monitoring resistor

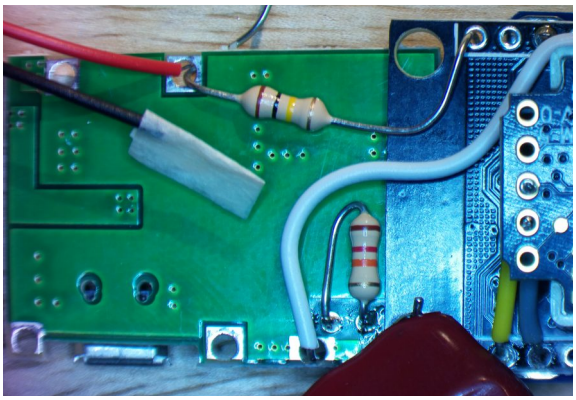
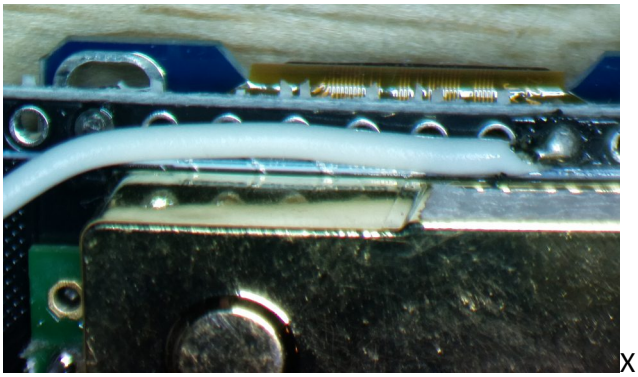
Disconnect the micro-USB plug before continuing.

The Vin wire runs on top of the charge board and in between the MCU board and the OLED board. It connects Vout+ to Vin on the small PCB to connect to the switch.





Connect GND from the charge PCB to the MCU board. The GND wire (white in this picture) runs below both boards, between Vout- on the charge board to GND connection (between 3V3 and CLK) on the MCU board (the white wire between RST and D0 has been omitted in these pictures). Both boards overlap ca 5 mm and can be clamped together.

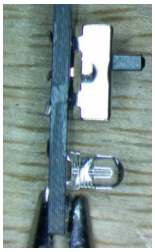
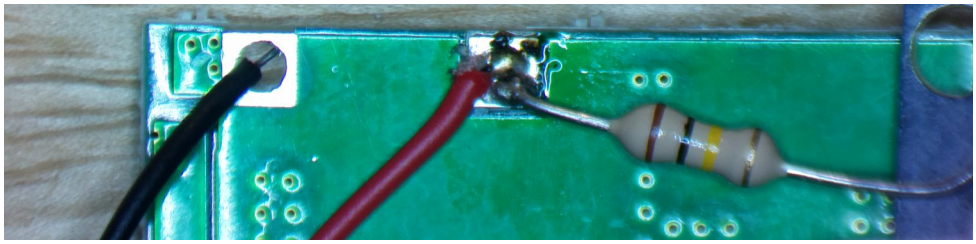

X

X

To enable monitoring of the battery voltage, a 120k resistor (and not 100k as shown in the picture above) can be added between BAT+ and pin A0 of the MCU. Solder the resistor to the MCU board, but not yet to the charge board – this must be done together with the red wire from the battery in the next step.

15. Connect the battery

Ensure that the switch is set to off (or up, so away from the LED).

Isolate the black wire from the battery, and solder the red wire together with the 120k Ohm resistor (both bottom side PCB) to BAT+. Then remove the isolation from the black wire and solder it also from the bottom side to the BAT- terminal. Be careful as the battery cannot be switched off when soldering.

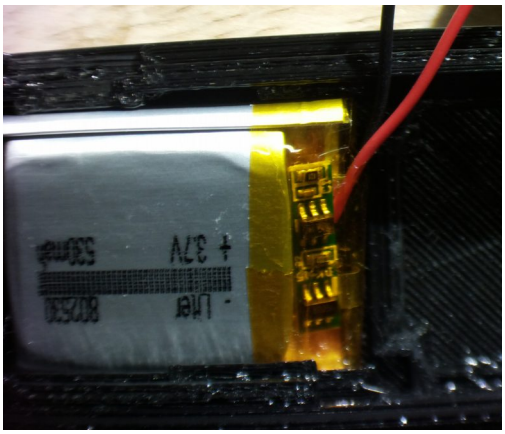



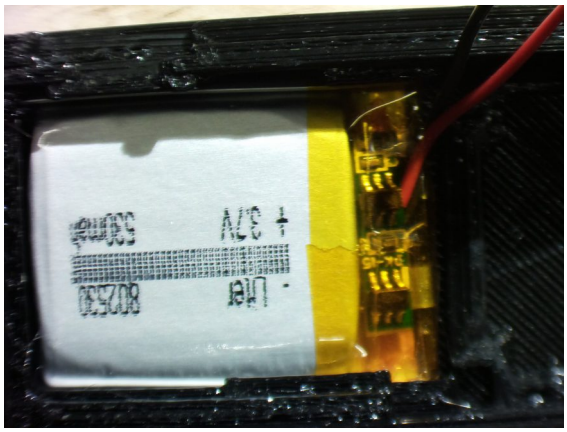
16. Fixate battery wires

For safety purposes fixate the battery wires using hot-glue.

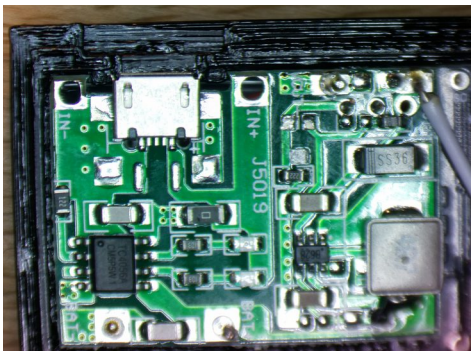
17. 3D case (bottom part)

If applicable move the battery (first one side, then the other side) gently into their locations.





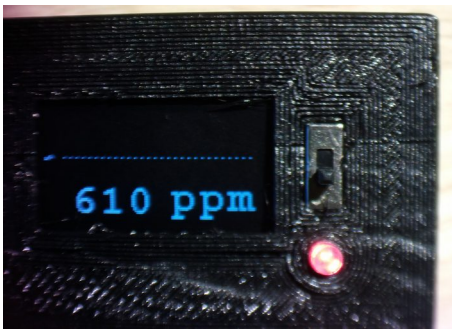
Then, move the PCB or PCBs into position.



18. 3D case (top part)

3D-printed holes are usually printed slightly smaller than designed – so first drill the LED hole to ensure it is at least 3mm.

Then, gently push the top part of the case on the bottom part and check whether it works.



19. Feedback?

Feedback is welcome!

References

[1] CO2 sensor design (PCB, STL files, building guide, firmware images) are located at https://github.com/Arnold-n/Covid-CO2. The firmware images are based on configuration files and a slightly modified version of Tasmota at https://github.com/Arnold-n/Tasmota/tree/CO2. The original Tasmota code is located at https://www.github.com/arendst/Tasmota.

[2] DIY CO2 building guide from Stefan van der Eijk at https://www.thingiverse.com/thing:3465447

[3] "Putting a balance on the aerosolization debate around SARS-CoV-2", by S.J. Dancer, J.W. Tang, L.C. Marr, S. Miller, L. Morawska, and J.L. Jimenez. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7219351/

[4] "FAQs on protecting yourself from COVID-19 Aerosol Transmission" https://tinyurl.com/FAQ-aerosols

[5] https://operame.nl (images in text from operame.nl are licensed under CC-BY-4.0)

[6] https://www.instructables.com/CO2-Sensor-for-Schools/ using Sensirion SCD30 sensor

[7] CO2 devices for sale at https://www.co2indicator.nl/co2-meter.html

[8] https://creativecommons.org/licenses/by-sa/4.0/legalcode

[9] NL https://www.tinytronics.nl/shop/nl, DE www.reichelt.nl, robotdyn https://robotdyn.com or its resellers https://robotdyn.com/how-to-buy/where-else-to-buy.html, see also separate ShoppingList

[10] https://www.letscontrolit.com/wiki/index.php/Basics:_Connecting_and_flashing_the_ESP8266

[11] Quickly set up a build environment for Tasmota using Docker https://github.com/tasmota/docker-tasmota

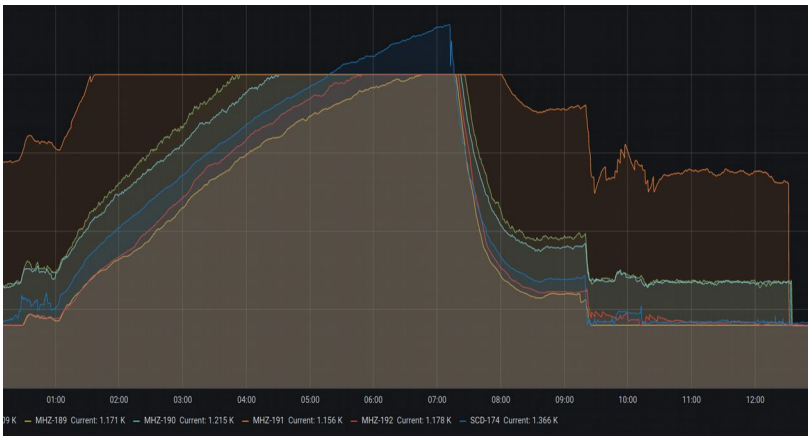
[12] http://ota.tasmota.com/tasmota/release-9.1.0/tasmota-minimal.bin.gz

Appendix: first experiences with the CO2 sensors.

Accuracy and calibration of MH-Z19B sensors

The MH-Z19B sensor has self-calibration (or “ABC” , Automatic Baseline Correction)– every 24 hours since the module is powered on, it sets the lowest CO2 level seen to 400ppm. This only works if the sensor sees enough fresh air, otherwise the readings will be too low. For that reason, I prefer to switch self-calibration off in the Tasmota firmware image (it is default on in Tasmota). It can still be switched on from the Tasmota console using command “Sensor15 1”.

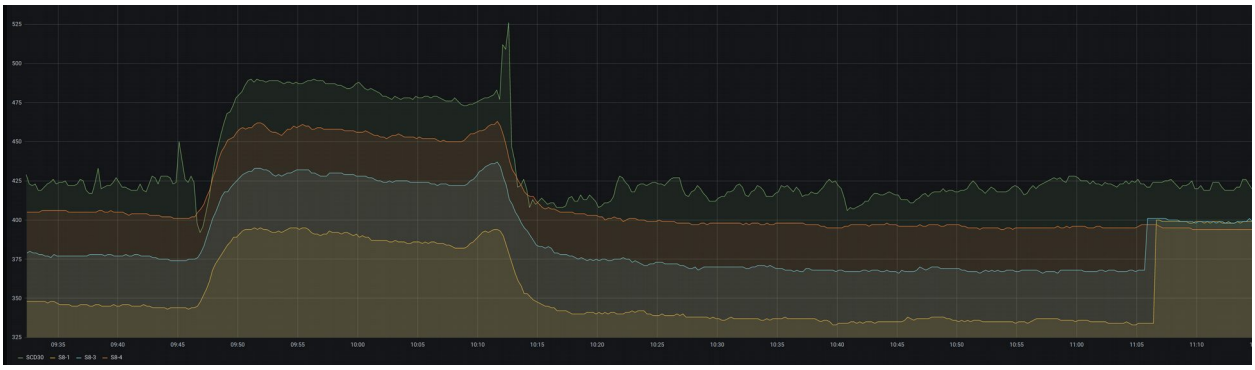
It is strongly advised to manually calibrate the MH-Z19B when it is outside (during day, in an open environment where plants or cars don’t produce too much CO2) for at least 20 minutes, and then manually calibrate from the Tasmota console using command “Sensor15 2”. This requires WiFi to be set up. Alternatively, the firmware image with self-calibration enabled may be used but in that case the sensor should see clean air at least every 24 hours.



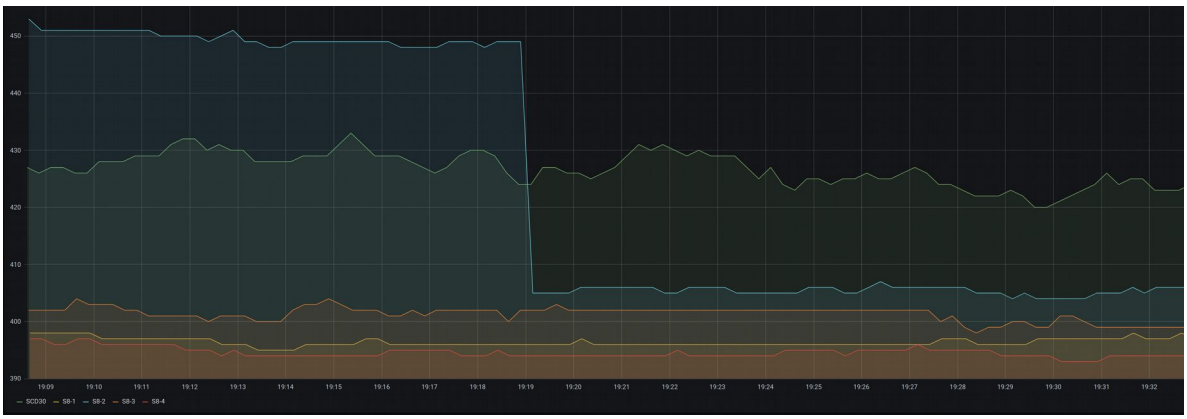
The images below show 5 brand-new MH-Z19B sensors obtained from different suppliers, together with a Sensirion SCD30 for comparison. Initially, in an empty room with an open window, the SCD30 and 2 MHZ-19B sensors were around 450 ppm, 2 others around 650ppm, and 1 sensor even 1400ppm. For this experiment the bedroom was not ventilated, and all MH-Z19B sensors reached their max of 2000ppm. Only at 12:30, with proper ventilation, all sensors were manually reset. Still, the difference between the sensors remains 100-200ppm after calibration, in the example below the SCD30 reads 1432ppm while the MHZ-19B sensors read 1241-1348ppm. The MH-Z19B sensors also tend to average measurements more than the SCD30, which is more responsive to quick changes in CO2.

Accuracy and calibration SenseAir S8 LP

The S8-LP sensors (type 004-0-0053) implement ABC (Automatic Baseline Correction) in smaller steps of max 30-50ppm per 8-day ABC period, although calibrations seem to start at the first day of operation. An example is shown here, where 2 sensors recalibrate in response to readings being below 400ppm.



Another example is a sensor which down-calibrates already 7 hours after its first power-up – without knowledge whether it has seen outside air in those 7 hours:



Even though the graphics don't show it well because of the automatic Y-axis scaling, the S8 sensor readings differ much less than the MH-Z19B readings: for example when SCD30 reported at 1432ppm, the four S8 sensors reported 1418-1444ppm.

Manual calibration of the S8 sensor is possible but requires that the (unconnected) input bCAL_in/CAL of the sensor is pulled up at processor reset for a minimum of 4, and a maximum of 8 seconds and this is not supported in this project.

Accuracy and calibration Sensirion SCD30

The Sensirion SCD30 sensor is a dual-channel NDIR sensor, and should thus be more stable over time than the other sensors. It also has several advanced auto-calibration options, including barimetric pressure compensation, but these options seem to be switched off afaik when running Tasmota.