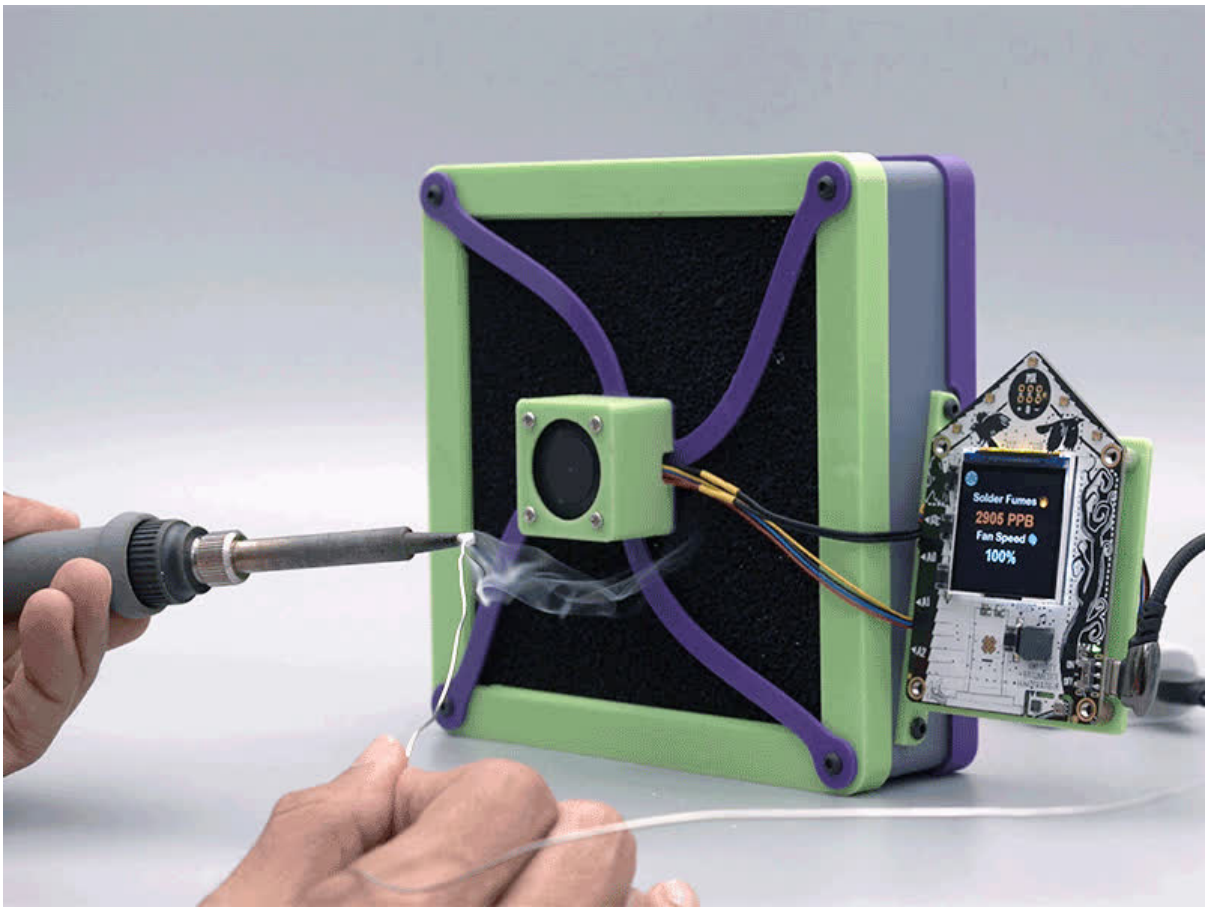
FunHouse IoT Fume Extractor and Air Quality Sensor

Created by Liz Clark



https://learn.adafruit.com/funhouse-fume-extractor-iot-air-quality-sensor

Last updated on 2024-03-23 01:52:44 PM EDT

Table of Contents

Overview 5

- Smart Fume Extractor
- Adafruit IO & Data Logging
- Fun Features
- Parts

CircuitPython 9

- Set Up CircuitPython
- Option 1 - Load with UF2 Bootloader
- Option 2 - Use Chrome Browser To Upload BIN file
- Option 3 - Use esptool to load BIN file

Code the FunHouse Fume Extractor 13

- Installing Project Code

Adafruit IO Setup 17

- Creating the Feeds
- Creating the Dashboard
- Adding the Feeds to Your Dashboard

CircuitPython Code Walkthrough 21

- Libraries
- STEMMA Sensor Setup
- Graphics
- State Machines and Variables
- Adafruit IO Functions
- The Loop
- Pressing the Down Button (Not Connecting to WiFi)
- Pressing the Select Button (Connecting to WiFi)
- The Main Program
- Sending Data to Adafruit IO
- Connecting and Disconnecting from WiFi

Circuit Diagram 28

- Adafruit Library for Fritzing
- Wired Connections

CAD Files 29

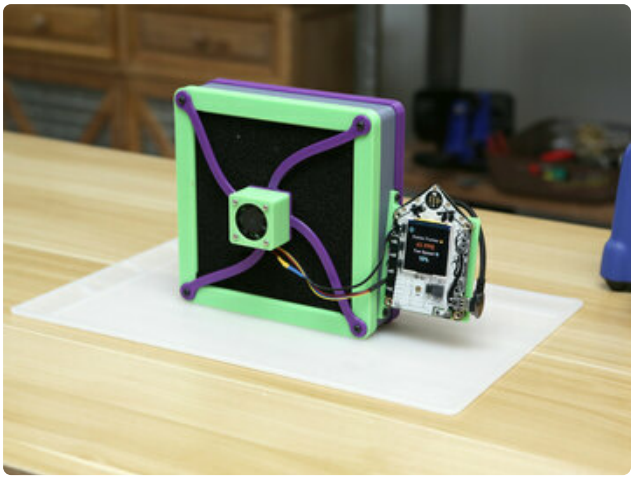
- 3D Printed Parts List
- CAD Assembly
- Slicing Parts
- Slicing Fan Grill
- Design Source Files

Wiring Mini Fan 31

- Wires for 5V Mini Fan
- Wiring JST Cable
- Wiring Mini 5V Fan Cable
- Wired Mini 5V Fan

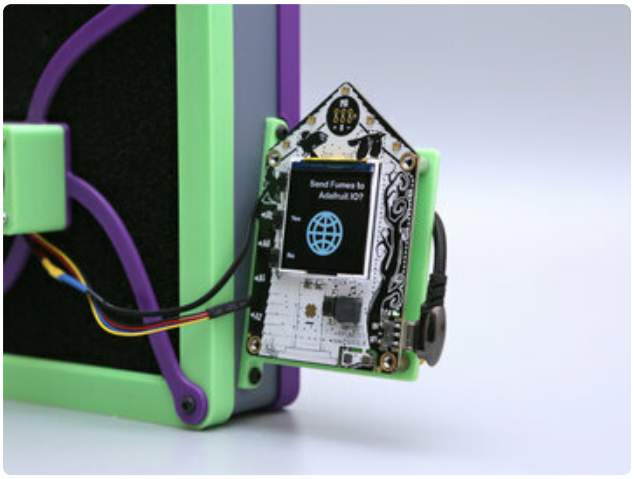
Wiring Noctua Fan 33
• Wires fro Noctua Fan
• Connect 4-Wire Cable
• Solder JST Cable
• Connect EMC2101 to Fan
• Connect Wires to EMC2101
• Wired EMC2101 and Noctua Fan
Assembly 35
• Back Cover and Fan
• Secure Fan to Back Cover
• Secured Fan
• Sensor Mount Hardware
• Secured Sensor
• Mini Fan Hardware
• Connect STEMMA Cable
• Mini Fan Cover Install
• Secure Mini Fan to Sensor Bracket
• Secure PCB Mount to Frame
• Secured PCB Mount
• Install Frame to Back Cover
• Install Carbon Filter
• Secure Sensor Bracket to Front Cover
• Installing Sensor Bracket
• Sensor and Mini Fan Assembled
• Install Front Cover to Frame
• EMC2101 Hardware
• Secure EMC2101 to PCB Mount
• Connect JST Cables to FunHouse
• Connect STEMMA to FunHouse
• Secure FunHouse to PCB Mount
• Secured FunHouse
• Final Build

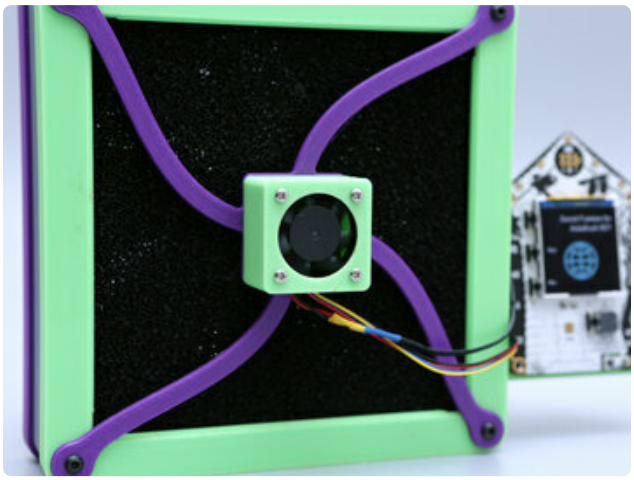
Overview



Smart Fume Extractor

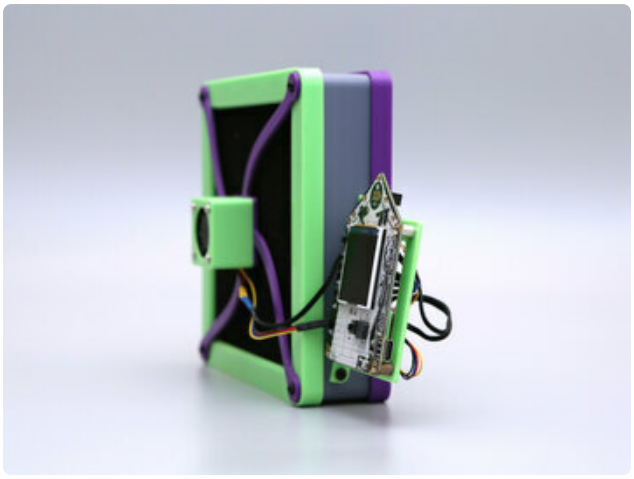
Build a smart DIY fume extractor with an Adafruit FunHouse and CircuitPython. Use a PWM fan and the EMC2101 controller to change the speed based on readings from an SPG30 air quality sensor.

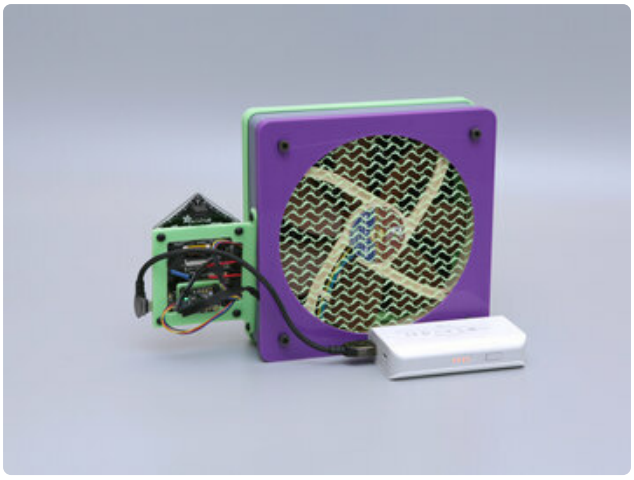




Adafruit IO & Data Logging

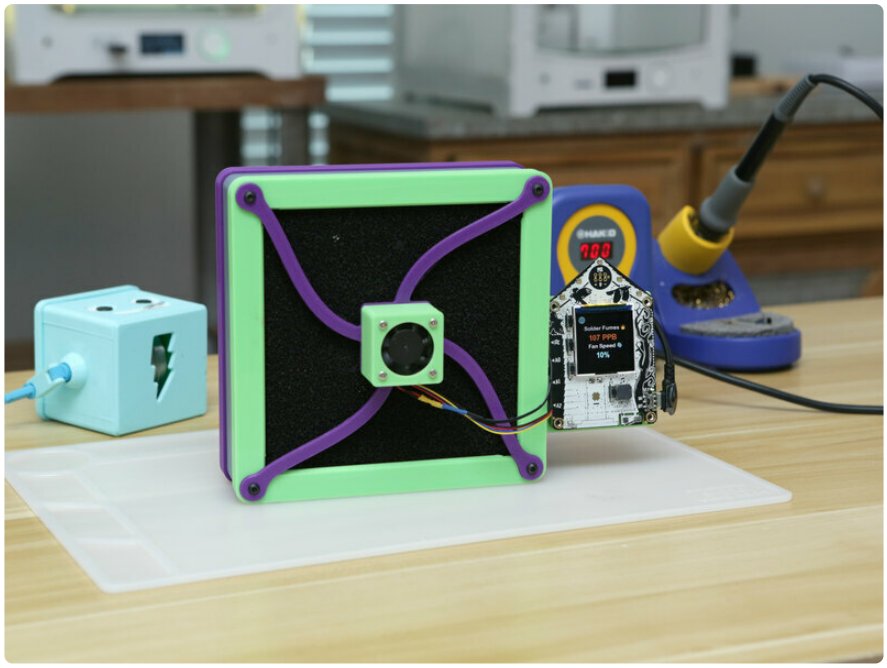
Use CircuitPython libraries and ESP32-S2 to connect to WiFi and log sensor data to a feed and display it on a dashboard with Adafruit IO. Use the FunHouse's built-in TFT to display bitmap graphics with fan speed and air quality. Use the built-in buttons to select options like choosing to log data. CircuitPython makes it easy to customize features, experiment with other hardware and quickly iterate.





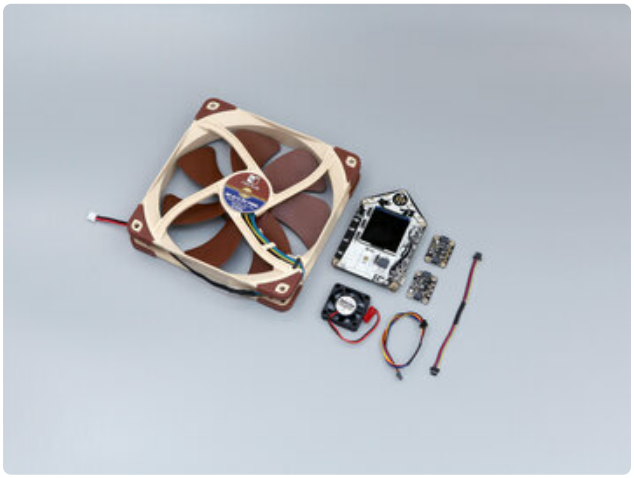
Fun Features

The electronics are housed in a 3D printed enclosure that snap fits together. Use a carbon activated filter to adsorb smoke from solder fumes. The fan controller and air quality sensor are connect with STEMMA QT cables for a plug-and-play circuit. A mini fan mounted in front of the air quality sensor directs fumes for sampling and measuring. The ports on the FunHouse make it easy to power 5V peripheral.

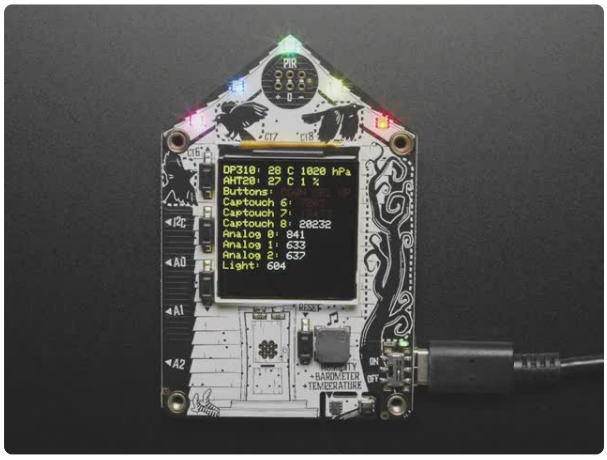


Parts

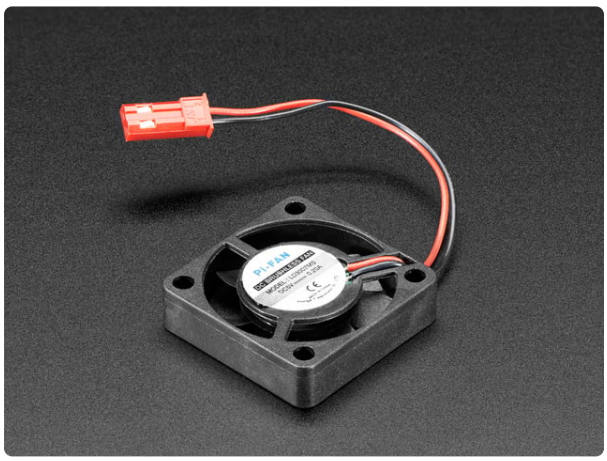
Parts to build this project.



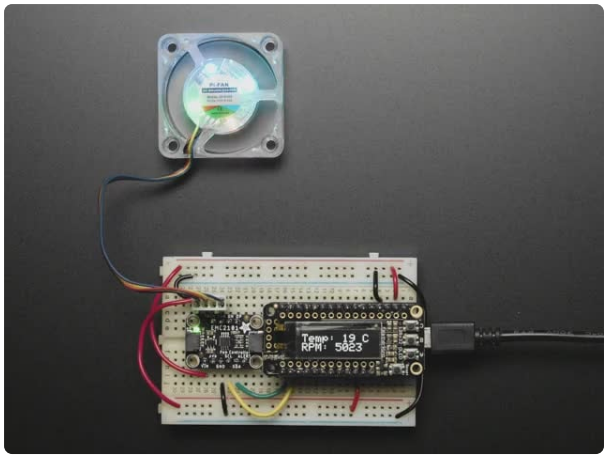
FunHouse ESP32-S2 (http://adafru.it/4985)
Noctua PWM Fan - 140mm (https://adafru.it/Tce)
5V Mini Fan (http://adafru.it/3368)
EMC12101 PWM Fan Controller (http://adafru.it/4808)
SGP30 Air Quality Sensor (http://adafru.it/3709)
STEMMA QT Cable - 200mm (http://adafru.it/4401)
STEMMA QT Cable - 100mm (http://adafru.it/4210)
Carbon Filter (http://adafru.it/3836)
M2.5 Hardware Kit (http://adafru.it/3299)
M3 Hardware Kit (http://adafru.it/4685)



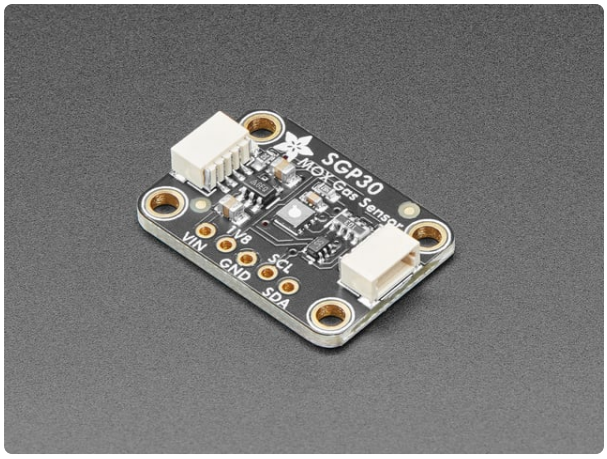
Adafruit FunHouse - WiFi Home Automation Development Board
Home is where the heart is...it's also where we keep all our electronic bits. So why not wire it up with sensors and actuators to turn our house into an electronic wonderland....
https://www.adafruit.com/product/4985



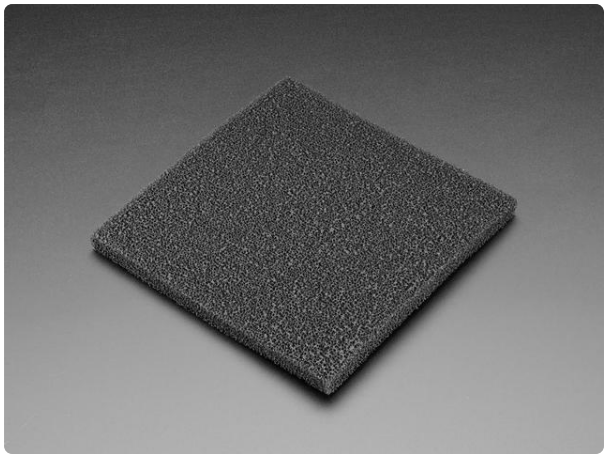
Miniature 5V Cooling Fan for Raspberry Pi (and Other Computers)
Looking for another way to keep your Raspberry Pi cool? Hook up this 5V Mini Cooling...
https://www.adafruit.com/product/3368



[Adafruit EMC2101 I2C PC Fan Controller and Temperature Sensor]
Cooling fans...They're everywhere, and they serve the important purpose of keeping things cool, generally electronics. One might rightfully think: "these fans are pretty good...
https://www.adafruit.com/product/4808



[Adafruit SGP30 Air Quality Sensor Breakout - VOC and eCO2]
Breathe easy with the SGP30 Multi-Pixel Gas Sensor, a fully integrated MOX gas sensor. This is a very fine air quality sensor from the sensor experts...
https://www.adafruit.com/product/3709



[Carbon Filter for Solder Smoke Absorption]
If you do a lot of soldering indoors, a Carbon Filter is essential for absorbing that solder smoke and leaving your air smelling fresh.These measure about...
https://www.adafruit.com/product/3836

[Black Nylon Machine Screw and Stand-off Set – M3 Thread]
Totaling 420 pieces, this M3 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel...
https://www.adafruit.com/product/4685

Black Nylon Machine Screw and Stand-off Set – M2.5 Thread

Totaling 380 pieces, this M2.5 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel your maker...

https://www.adafruit.com/product/3299

STEMMA QT / Qwiic JST SH 4-Pin Cable - 200mm Long

This 4-wire cable is a little over 200mm / 7.8" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4401

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

1 x Noctua PWM Fan

140mm Noctua PWM Fan

https://www.amazon.com/dp/B07DXR1Y73

CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/f9W) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no

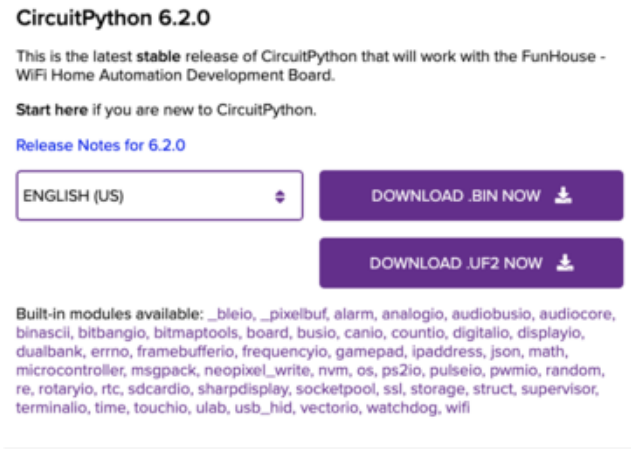
upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

Set Up CircuitPython

Follow the steps to get CircuitPython installed on your FunHouse.

Download the latest CircuitPython for your board from circuitpython.org

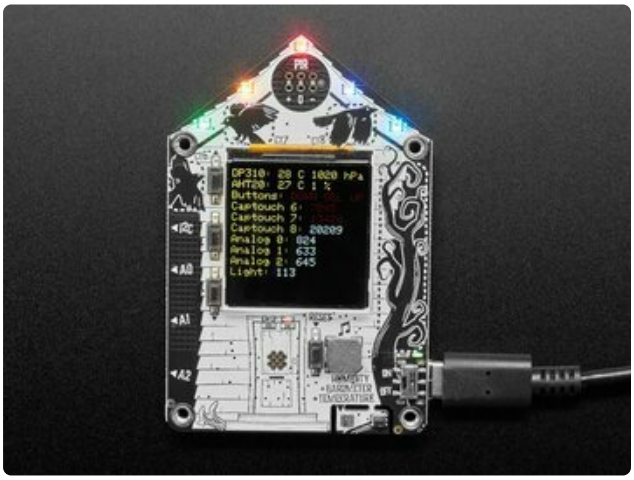
https://adafru.it/RKB



Click the link above and download the latest .BIN and .UF2 file

(depending on how you program the ESP32S2 board you may need one or the other, might as well get both)

Download and save it to your desktop (or wherever is handy).



Plug your FunHouse into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

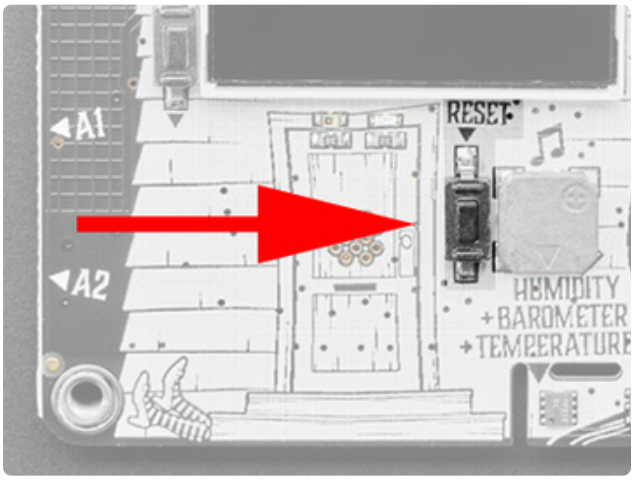
Option 1 - Load with UF2 Bootloader

This is by far the easiest way to load CircuitPython. **However it requires your board has the UF2 bootloader installed. Some early boards do not (we hadn't written UF2 yet!) - in which case you can load using the built in ROM bootloader.**

Still, try this first!

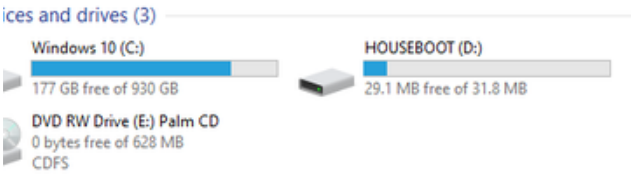
Try Launching UF2 Bootloader

Loading CircuitPython by drag-n-drop UF2 bootloader is the easier way and we recommend it.



Launch UF2 by **double-clicking** the Reset button (the one next to the USB C port). You may have to try a few times to get the timing right.

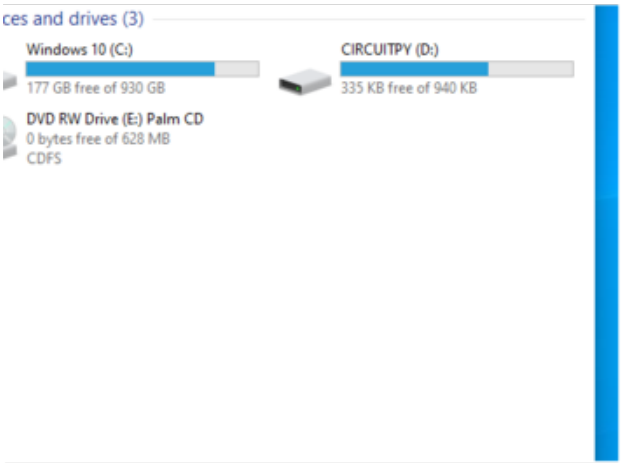
About a half second pause between clicks while the DotStars are purple seems to work well.



If the UF2 bootloader is installed, you will see a new disk drive appear called **HOUSEBOOT**

Copy the **UF2** file you downloaded at the first step of this tutorial onto the **HOUSEBOOT** drive

If you're using Windows and you get an error at the end of the file copy that says **Error from the file copy, Error 0x800701B1: A device which does not exist was specified.** You can ignore this error, the bootloader sometimes disconnects without telling Windows, the install completed just fine and you can continue. If its really annoying, you can also upgrade the bootloader (the latest version of the UF2 bootloader fixes this warning) (https://adafru.it/RLc)



Your board should auto-reset into CircuitPython, or you may need to press reset. A **CIRCUITPY** drive will appear. You're done! Go to the next pages.

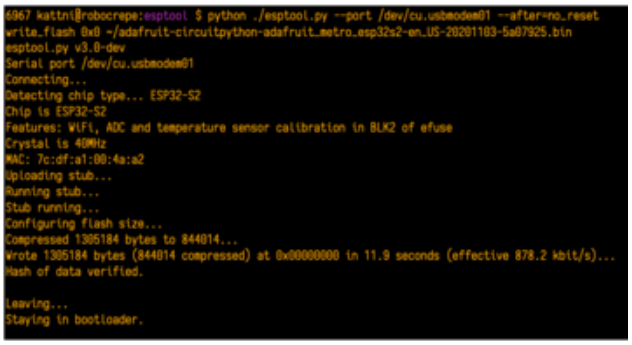
Option 2 - Use Chrome Browser To Upload BIN file

You will need to do a full erase prior to uploading new firmware.

The next best option is to try using the Chrome-browser version of esptool we have written. This is handy if you don't have Python on your computer, or something is really weird with your setup that makes esptool not run (which happens sometimes and isn't worth debugging!) You can follow along on the Install UF2 Bootloader (https://adafru.it/RLc) page and either load the UF2 bootloader and then come back to Option 1 on this page, or you can download the CircuitPython BIN file directly using the tool in the same manner as the bootloader.

Option 3 - Use esptool to load BIN file

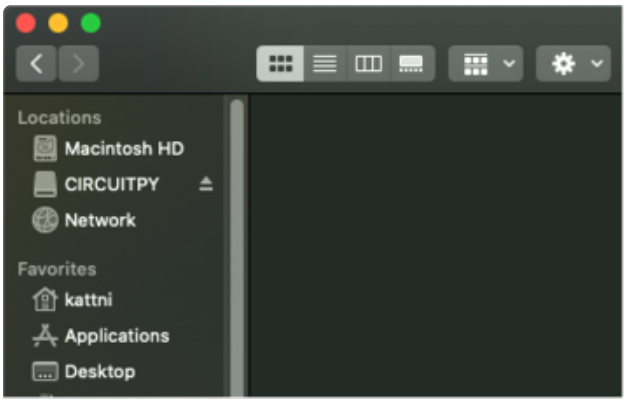
For more advanced users, you can upload with **esptool** to the ROM (hardware) bootloader instead!



Follow the initial steps found in the Run esptool and check connection section of the Install UF2 Bootloader page (https://adafru.it/RLc) to verify your environment is set up, your board is successfully connected, and which port it's using.

In the final command to write a binary file to the board, replace the port with your port, and replace "firmware.bin" with the the file you downloaded above.

The output should look something like the output in the image.



Press reset to exit the bootloader.

Your **CIRCUITPY** drive should appear!

You're all set! Go to the next pages.

Code the FunHouse Fume Extractor

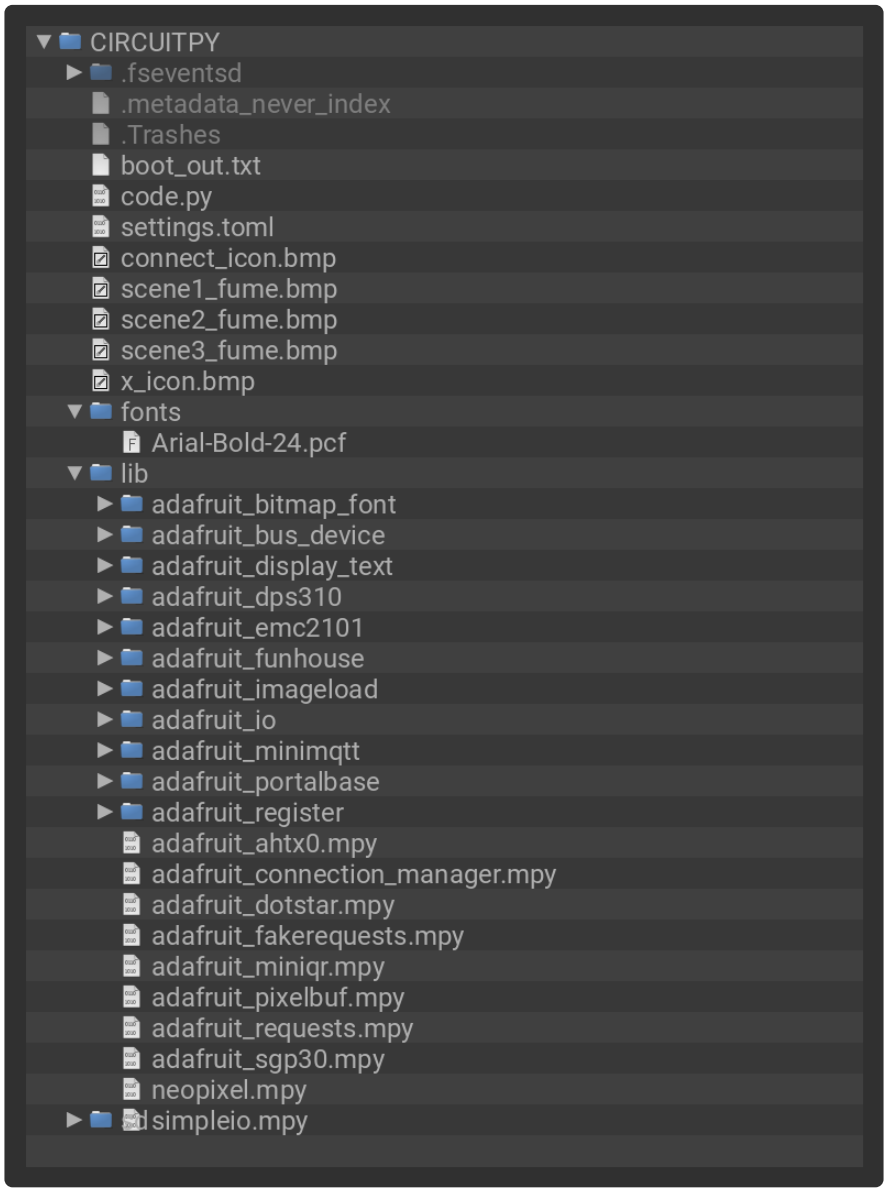
Installing Project Code

To use with CircuitPython, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file

in a zip file. Extract the contents of the zip file, open the directory **FunHouse_Fume_Extractor/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should now look similar to the following image:




```
# SPDX-FileCopyrightText: 2021 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import simpleio
import adafruit_sgp30
import displayio
import adafruit_imageload
from adafruit_emc2101 import EMC2101
from adafruit_funhouse import FunHouse

i2c = board.I2C()  # uses board.SCL and board.SDA
```

```
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller

#  setup for SGP30 sensor
sgp30 = adafruit_sgp30.Adafruit_SGP30(i2c)
#  setup for fan controller
emc = EMC2101(i2c)

print("SGP30 serial #", [hex(i) for i in sgp30.serial])

#SGP30 start-up
sgp30.iaq_init()
sgp30.set_iaq_baseline(0x8973, 0x8AAE)

#  FunHouse setup
funhouse = FunHouse(default_bg=0x0F0F00)
#  start-up bitmap
bitmap, palette = adafruit_imageload.load("/scene1_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
tile_grid = displayio.TileGrid(bitmap, pixel_shader=palette)
#  connecting bitmap
bitmap2, palette2 = adafruit_imageload.load("/scene2_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
grid2 = displayio.TileGrid(bitmap2, pixel_shader=palette2)
#  default background
bitmap3, palette3 = adafruit_imageload.load("/scene3_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
grid3 = displayio.TileGrid(bitmap3, pixel_shader=palette3)
#  internet connection icon
bitmap4, palette4 = adafruit_imageload.load("/connect_icon.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
icon1 = displayio.TileGrid(bitmap4, pixel_shader=palette4, x = 2, y = 2)
#  red x icon
bitmap5, palette5 = adafruit_imageload.load("/x_icon.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
icon2 = displayio.TileGrid(bitmap5, pixel_shader=palette5, x = 2, y = 2)
#  display group
group = displayio.Group()
#  adding start-up bitmap to group
group.append(tile_grid)
funhouse.splash.append(group)
#  text for fume data
fume_text = funhouse.add_text(
    text="    ",
    text_position=(110, 90),
    text_anchor_point=(0.5, 0.5),
    text_color=0xf57f20,
    text_font="fonts/Arial-Bold-24.pcf",
)
#  text for fan RPM data
fan_text = funhouse.add_text(
    text="    ",
    text_position=(110, 165),
    text_anchor_point=(0.5, 0.5),
    text_color=0x7fffff,
    text_font="fonts/Arial-Bold-24.pcf",
)
#  showing graphics
funhouse.display.show(funhouse.splash)

#  state machines
run = False #  state if main code is running
connected = False #  checks if connected to wifi
start_up = False #  state for start-up
```

```
clock = 0 #  time.monotonic() device

#  function for sending fume data to adafruit.io
def send_fume_data(solder_fumes):
    funhouse.network.push_to_io("fumes", solder_fumes)

#  function for sending fan rpm to adafruit.io
def send_fan_data(fan_rpm):
    funhouse.network.push_to_io("fan-speed", fan_rpm)

while True:
    #  if main program has not started
    if not run:
        #  if you press the down button
        if funhouse.peripherals.button_down:
            print("run")
            #  remove start-up bitmap
            group.remove(tile_grid)
            #  add main bitmap
            group.append(grid3)
            #  add red x icon to show not connected to internet
            group.append(icon2)
            #  change state for main program
            run = True
        #  if you press the middle button
        if funhouse.peripherals.button_sel:
            #  remove start-up bitmap
            group.remove(tile_grid)
            #  add connecting... bitmap
            group.append(grid2)
            #  connect to the network
            funhouse.network.connect()
            print("connecting")
            #  change state for network
            connected = True
            #  start main program
            start_up = True
            #  start time.monotonic()
            clock = time.monotonic()
        #  after connecting to the internet
        if start_up:
            #  remove connecting bitmap
            group.remove(grid2)
            #  add main bitmap
            group.append(grid3)
            #  add internet icon
            group.append(icon1)
            #  start main program
            run = True
            #  reset start-up state
            start_up = False

    #  run state for main program after selecting whether or not to connect to wifi
    if run:
        #  print eCO2 and TVOC data to REPL
        print("eCO2 = %d ppm \t TVOC = %d ppb" % (sgp30.eCO2, sgp30.TVOC))
        #  2 second delay
        time.sleep(2)

        #  fumes variable for reading from SGP30
        #  comment out either TVOC or eCO2 depending on data preference
        fumes = sgp30.TVOC
        #  fumes = sgp30.eCO2

        #  mapping fumes data to fan RPM
        #  value for TVOC
        mapped_val = simpleio.map_range(fumes, 10, 1000, 10, 100)
        #  value for eCO2
        #  mapped_val = simpleio.map_range(fumes, 400, 2500, 10, 100)
```

```
#  adding fume text
#  PPB is for TVOC, PPM is for eCO2
funhouse.set_text("%d PPB" % fumes, fume_text)
#  funhouse.set_text("%d PPM" % fumes, fume_text)

#  adding fan's RPM text
funhouse.set_text("%d%s" % (mapped_val, "%"), fan_text)
#  printing fan's data to the REPL
print("fan = ", mapped_val)
#  setting fan's RPM
emc.manual_fan_speed = int(mapped_val)

#  if you're connected to wifi and 15 seconds has passed
if connected and ((clock + 15) < time.monotonic()):
    #  send fume data to adafruit.io
    send_fume_data(fumes)
    #  send fan RPM to adafruit.io
    send_fan_data(mapped_val)
    #  REPL printout
    print("data sent")
    #  reset clock
    clock = time.monotonic()
#  if you're connected to wifi and you press the up button
if connected and funhouse.peripherals.button_up:
    #  the internet icon is removed
    group.remove(icon1)
    #  the red x icon is added
    group.append(icon2)
    #  reset connected state - no longer sending data to adafruit.io
    connected = False
    #  REPL printout
    print("disconnected")
    #  1 second delay
    time.sleep(1)
#  if you're NOT connected to wifi and you press the up button
if not connected and funhouse.peripherals.button_up:
    #  the red x icon is removed
    group.remove(icon2)
    #  the internet icon is added
    group.append(icon1)
    #  the connection state is true - start sending data to adafruit.io
    connected = True
    #  REPL printout
    print("connected")
    #  1 second delay
    time.sleep(1)
```

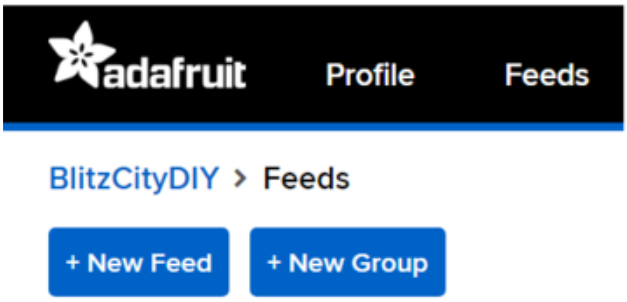
Adafruit IO Setup

Before you begin logging your fume extractor data, you need to do some setup in Adafruit IO.

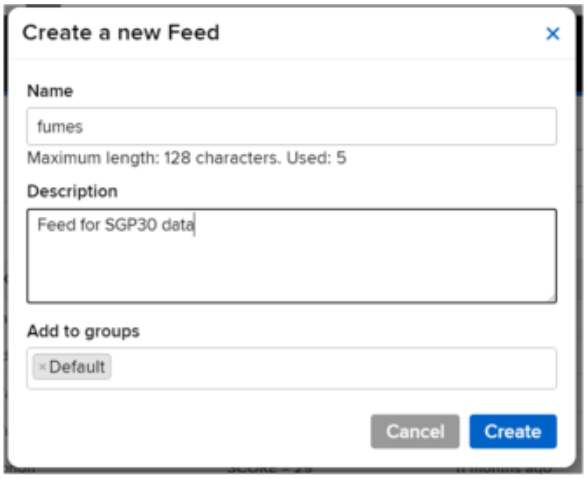
First, check out this Learn guide (https://adafru.it/Ef8) if you've never used Adafruit IO before to see how to get started.

Creating the Feeds

For this project you'll need two feeds: **fumes** and **fan-speed**.

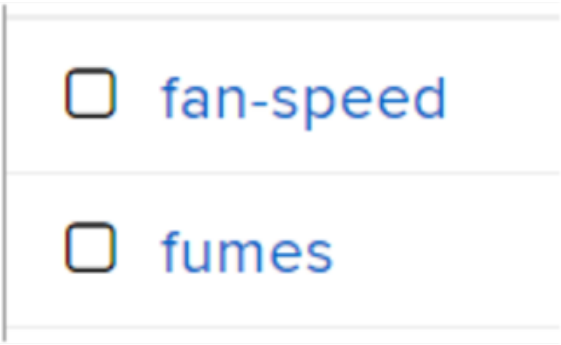


Go to **Feeds** and then click on + **New Feed**.



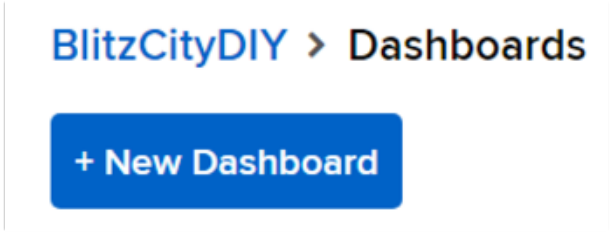
Name the feed **"fumes"** and then click Create.

Repeat this process to create the **"fan-speed"** feed.

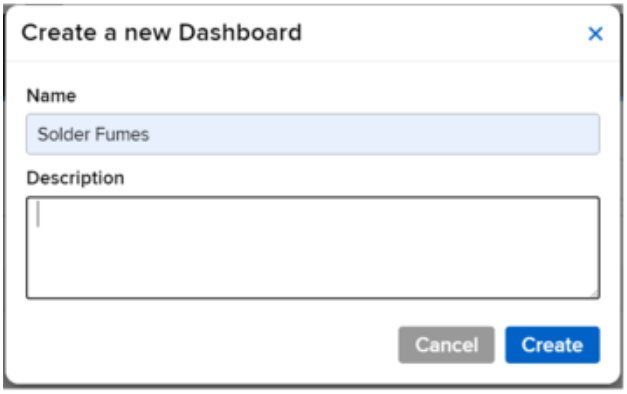


Afterwards, you should see both feeds in your list.

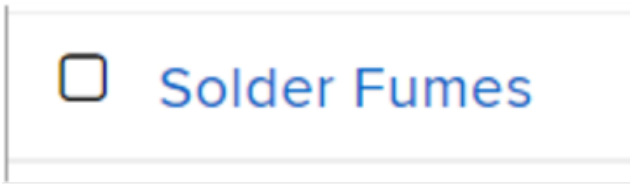
Creating the Dashboard



Go to **Dashboards** and then click on + **New Dashboard**.

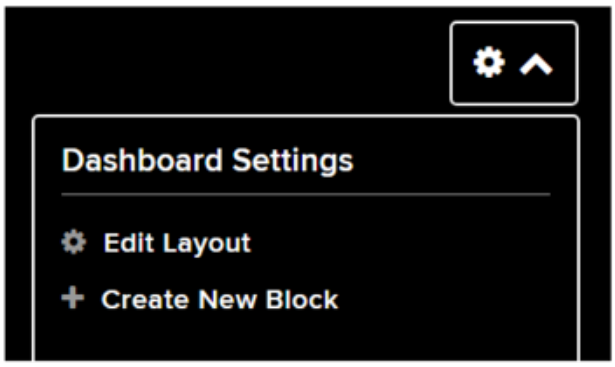


Name the dashboard and then click **Create**.

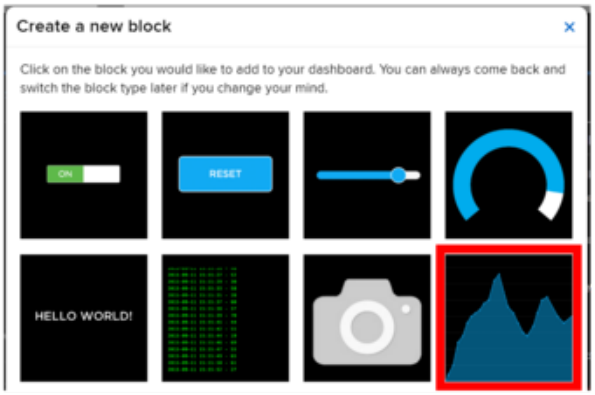


Click on the dashboard to add the feeds.

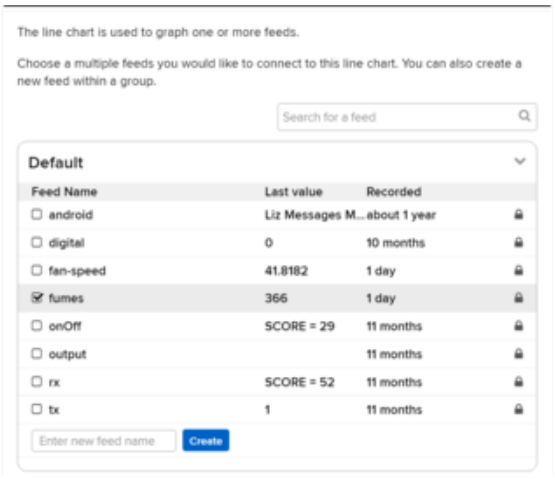
Adding the Feeds to Your Dashboard



On your dashboard page, click on the **cog wheel** on the right-hand side of the screen. Then click + **Create New Block**.



Click on the **Line Chart** block, outlined in red in the picture.

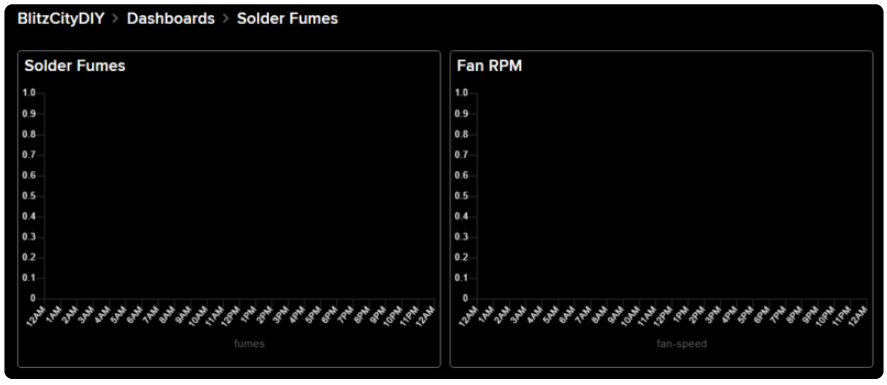


Check-off the **fumes** feed.

Then click **Next step**.

Name the block and then click **Create block**.

Repeat this process for the **fan-speed** feed. Afterwards, you should have two line chart blocks on your dashboard for both feeds.



CircuitPython Code Walkthrough

Libraries

The code begins by importing the libraries.

```
import time
import board
import simpleio
import adafruit_sgp30
import displayio
import adafruit_imageload
from adafruit_emc2101 import EMC2101
from adafruit_funhouse import FunHouse
```

STEMMA Sensor Setup

Next, I2C and the two STEMMA sensors, the **SGP30** and **EMC2101**, are setup.

```
i2c = board.I2C()

#  setup for SGP30 sensor
sgp30 = adafruit_sgp30.Adafruit_SGP30(i2c)
#  setup for fan controller
emc = EMC2101(i2c)

print("SGP30 serial #", [hex(i) for i in sgp30.serial])

#SGP30 start-up
sgp30.iaq_init()
sgp30.set_iaq_baseline(0x8973, 0x8AAE)
```

For more information on the Adafruit SGP30 TVOC/eCO2 Gas Sensor, check out this Learn guide

https://adafru.it/Tch

For more information on the EMC2101 Fan Controller and Temperature Sensor, check out this Learn guide

https://adafru.it/Tci

Graphics

The `adafruit_funhouse` library has a `FunHouse` object that can be used for `displayio`.

Following that import, the five bitmaps are setup with `displayio`. There are three background images that will be shown depending on where you are in the code. The remaining two bitmaps are icons to show whether or not you are sending data to Adafruit IO.

```
#  FunHouse setup
funhouse = FunHouse(default_bg=0x0F0F00)
#  start-up bitmap
bitmap, palette = adafruit_imageload.load("/scene1_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
tile_grid = displayio.TileGrid(bitmap, pixel_shader=palette)
#  connecting bitmap
bitmap2, palette2 = adafruit_imageload.load("/scene2_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
grid2 = displayio.TileGrid(bitmap2, pixel_shader=palette2)
#  default background
bitmap3, palette3 = adafruit_imageload.load("/scene3_fume.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
grid3 = displayio.TileGrid(bitmap3, pixel_shader=palette3)
#  internet connection icon
bitmap4, palette4 = adafruit_imageload.load("/connect_icon.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
icon1 = displayio.TileGrid(bitmap4, pixel_shader=palette4, x = 2, y = 2)
#  red x icon
bitmap5, palette5 = adafruit_imageload.load("/x_icon.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)
icon2 = displayio.TileGrid(bitmap5, pixel_shader=palette5, x = 2, y = 2)
```

Following the bitmap imports, a display `group` is setup and the first bitmap is added to the `group`. Then two text objects are created, `fume_text` and `fan_text`, that will display the data output from the SGP30 and EMC2101 sensors.

```
#  display group
group = displayio.Group()
#  adding start-up bitmap to group
group.append(tile_grid)
funhouse.splash.append(group)
#  text for fume data
fume_text = funhouse.add_text(
    text="    ",
    text_position=(110, 90),
    text_anchor_point=(0.5, 0.5),
    text_color=0xf57f20,
    text_font="fonts/Arial-Bold-24.pcf",
)
#  text for fan RPM data
fan_text = funhouse.add_text(
    text="    ",
    text_position=(110, 165),
    text_anchor_point=(0.5, 0.5),
    text_color=0x7fffff,
    text_font="fonts/Arial-Bold-24.pcf",
)
#  showing graphics
funhouse.display.root_group = funhouse.splash
```

State Machines and Variables

A few state machines and variables are setup for use in the loop. Their functions are commented below.

```
#  state machines
run = False #  state if main code is running
connected = False #  checks if connected to wifi
start_up = False #  state for start-up
clock = 0 #  time.monotonic() device
```

Adafruit IO Functions

Two functions are created to send data to Adafruit IO. `send_fume_data()` sends data from the SGP30 and `send_fan_data()` sends data from the ECM2101.

For these functions to work properly, you need to have feeds setup in Adafruit IO called "`fumes`" and "`fan-speed`". Be sure to follow the steps on the Adafruit IO Setup page (https://adafru.it/Tcj) in this guide.

```
#  function for sending fume data to adafruit.io
def send_fume_data(solder_fumes):
    funhouse.network.push_to_io("fumes", solder_fumes)

#  function for sending fan rpm to adafruit.io
def send_fan_data(fan_rpm):
    funhouse.network.push_to_io("fan-speed", fan_rpm)
```

The Loop

The loop begins by checking if the main program is running with the `run` state. If the main program is not running, then the start-up graphic is shown on the FunHouse and you need to press either the middle or bottom button to begin the fume extraction. The buttons determine whether or not you connect to WiFi and send your fume data to Adafruit IO

Pressing the Down Button (Not Connecting to WiFi)

If you press the down button, the start-up bitmap is removed from `group` and the main program bitmap is added to `group` for display. Additionally, the red x icon is also added to `group` to show that you are not connected to WiFi. Finally, `run` is set to `True` so that the fume extraction and data display can begin.

```
#  if you press the down button
        if funhouse.peripherals.button_down:
            print("run")
            #  remove start-up bitmap
            group.remove(tile_grid)
            #  add main bitmap
            group.append(grid3)
            #  add red x icon to show not connected to internet
            group.append(icon2)
            #  change state for main program
            run = True
```

Pressing the Select Button (Connecting to WiFi)

If you press the select button (in the middle of the FunHouse's three buttons), the start-up bitmap is removed from `group` and the "connecting..." bitmap is added to group.

Then, the function `funhouse.network.connect()` is called. This connects the FunHouse to WiFi.

After a network connection has been established, the `start_up` state is set to `True` and `clock` is reset with `time.monotonic()`.

```
#  if you press the middle button
        if funhouse.peripherals.button_sel:
            #  remove start-up bitmap
            group.remove(tile_grid)
            #  add connecting... bitmap
            group.append(grid2)
            #  connect to the network
            funhouse.network.connect()
            print("connecting")
            #  change state for network
```

```
            connected = True
            #  start main program
            start_up = True
            #  start time.monotonic()
            clock = time.monotonic()
```

The start-up state exists so that after the FunHouse connects to WiFi, the connecting graphic can be removed from `group` and replaced with the main graphic. Additionally, the internet icon is added to show that you're sending data to Adafruit IO. Finally, `run` is set to `True`.

```
#  after connecting to the internet
        if start_up:
            #  remove connecting bitmap
            group.remove(grid2)
            #  add main bitmap
            group.append(grid3)
            #  add internet icon
            group.append(icon1)
            #  start main program
            run = True
            #  reset start-up state
            start_up = False
```

The Main Program

The main program takes readings from the SGP30 to affect the fan's RPM speed. This is done with the `simpleio.map_range()` function. You can either use the SGP30's eCO2 readings or TVOC readings. You'll just need to comment and uncomment certain lines to change this.

The data from the SGP30's readings and the fan's RPM are updated for the `fume_text` and `fan_text` objects to display on the screen.

```
#  fumes variable for reading from SGP30
        #  comment out either TVOC or eCO2 depending on data preference
        fumes = sgp30.TVOC
```

```
        #  fumes = sgp30.eCO2

        #  mapping fumes data to fan RPM
        #  value for TVOC
        mapped_val = simpleio.map_range(fumes, 10, 1000, 10, 100)
        #  value for eCO2
        #  mapped_val = simpleio.map_range(fumes, 400, 2500, 10, 100)

        #  adding fume text
        #  PPB is for TVOC, PPM is for eCO2
        funhouse.set_text("%d PPB" % fumes, fume_text)
        #  funhouse.set_text("%d PPM" % fumes, fume_text)

        #  adding fan's RPM text
        funhouse.set_text("%d%s" % (mapped_val, "%"), fan_text)
        #  printing fan's data to the REPL
        print("fan = ", mapped_val)
        #  setting fan's RPM
        emc.manual_fan_speed = int(mapped_val)
```

Sending Data to Adafruit IO

Every fifteen seconds, the data from the SGP30 and EMC2101 are sent to feeds on Adafruit IO. After the data is sent, `clock` is reset with `time.monotonic()`.

```
#  if you're connected to wifi and 15 seconds has passed
        if connected and ((clock + 15) &lt; time.monotonic()):
            #  send fume data to adafruit.io
            send_fume_data(fumes)
            #  send fan RPM to adafruit.io
            send_fan_data(mapped_val)
            #  REPL printout
            print("data sent")
            #  reset clock
            clock = time.monotonic()
```

Connecting and Disconnecting from WiFi

If you decide after booting up the FunHouse that you want to connect or disconnect from WiFi, you can press the up button on the FunHouse next to the icon in the top left corner of the screen.

The connection is affected by changing the `connected` state. The icon also changes to show the connection status visually.

```
#  if you're connected to wifi and you press the up button
        if connected and funhouse.peripherals.button_up:
            #  the internet icon is removed
            group.remove(icon1)
            #  the red x icon is added
            group.append(icon2)
            #  reset connected state - no longer sending data to adafruit.io
            connected = False
            #  REPL printout
            print("disconnected")
            #  1 second delay
```

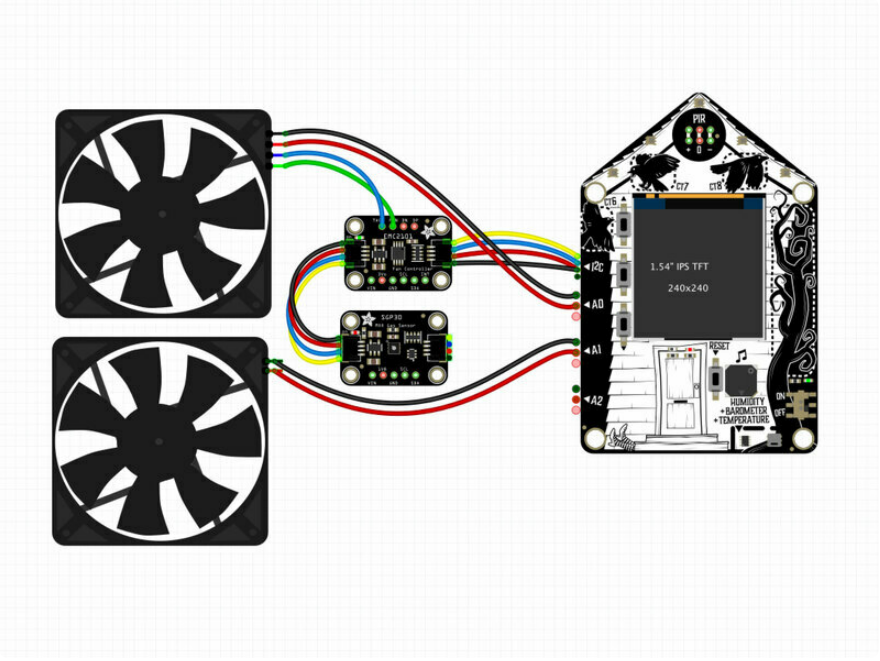
```
        time.sleep(1)
    #  if you're NOT connected to wifi and you press the up button
    if not connected and funhouse.peripherals.button_up:
        #  the red x icon is removed
        group.remove(icon2)
        #  the internet icon is added
        group.append(icon1)
        #  the connection state is true - start sending data to adafruit.io
        connected = True
        #  REPL printout
        print("connected")
        #  1 second delay
        time.sleep(1)
```

Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing (https://adafru.it/oEP).

Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts (https://adafru.it/AYZ).



Wired Connections

The Noctua PWM fan and 5V mini fan are powered by the 5V and ground pins on the FunHouse's **A0** and **A1** ports. The FunHouse can be powered by 5V USB hub or 5V 1A USB battery bank.

5V Mini Fan

- Ground to A1 ground
- Voltage to A1 voltage

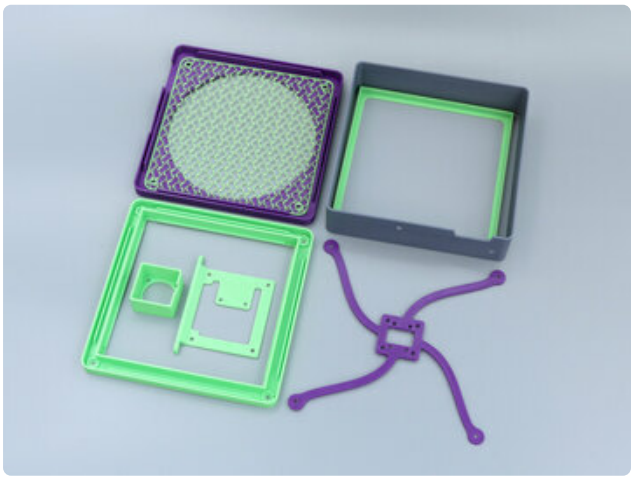
Noctua Fan

- Wire 1 ground to A0 ground
- Wire 2 voltage to A0 voltage
- Wire 3 tach to TACH on EMC2101
- Wire 4 PWM to FAN on EMC2101

FunHouse

- I2C to EMC2101
- SGP30 to EMC2101

CAD Files



3D Printed Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

front-cover.stl
back-cover.stl
minifan-cover.stl
fan-grill.stl
filter-holder.stl
sensor-mount.stl
pcb-mount.stl
minifan-cover.stl
frame.stl

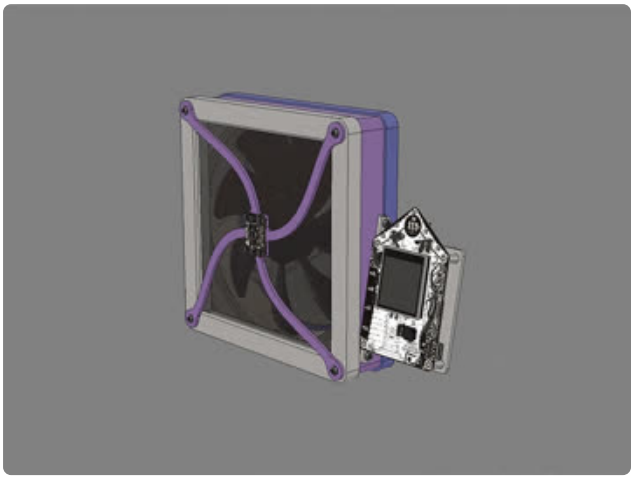
Download STLs.zip

https://adafru.it/Tck

Download CAD-source.zip

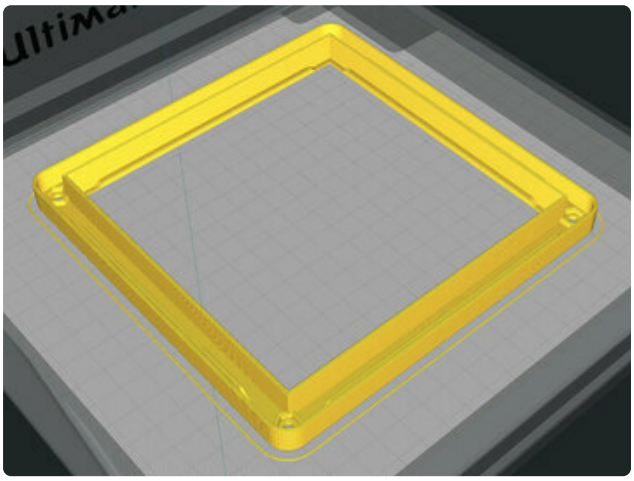
https://adafru.it/Tcl

CAD Assembly



The FunHouse is secured to the **pcb-mount** with M3 screws. The **pcb-mount** is attached to the **frame** using M3 screws and hexnuts. The EMC2101 breakout is secured to the **pcb-mount** with M2.5 screws. The 140mm fan is secured to the **back-cover** with M4 screws and hex nuts. The **back-cover** snap fits over the frame. The carbon filter is press fitted into the **front-cover**. The SGP30 sensor is secured to the center of the **sensor-mount** using M2.5 screws and hex nuts. The **sensor-mount** is secured to the front cover using M3 screws and hex nuts. The **front-cover** snap fits over the frame.

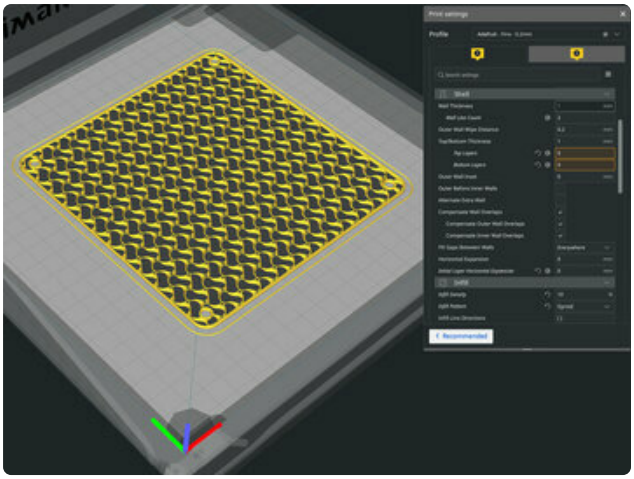
Slicing Parts



Minimum build volume required 154mm x 154mm. No supports are required. Slice with setting for PLA material.

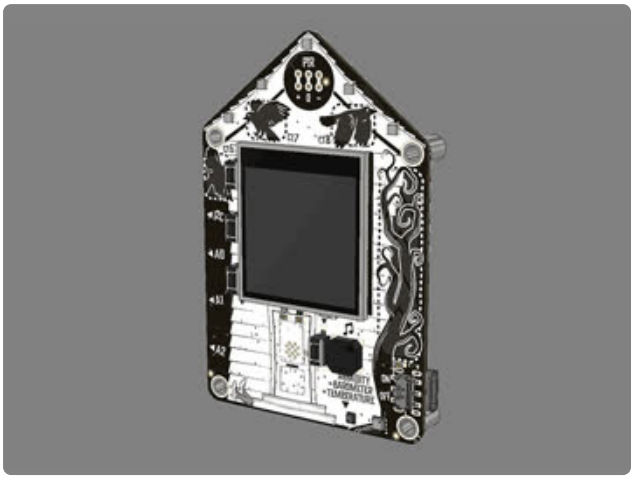
The parts were sliced using CURA using the slice settings below.

PLA filament 220c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed



Slicing Fan Grill

The fan grill mesh is created by using the slicing softwares infill pattern. Set the Top/Bottom layers to 0 and choose your preferred infill settings. Use a minimum of 10% infill. Gyroid patter is used in the screenshot.



Design Source Files

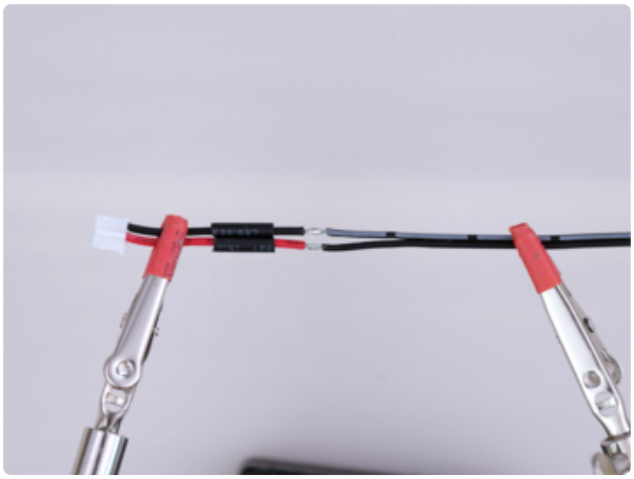
The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo (https://adafru.it/AW8).

Wiring Mini Fan

Wires for 5V Mini Fan

Use a piece of 2-wire ribbon to connect the 3-pin JST connector to the 5V fan. Use the following wire lengths.

2-wire ribbon - 112mm
3-pin JST cable - 38mm
mini fan cable - 38mm

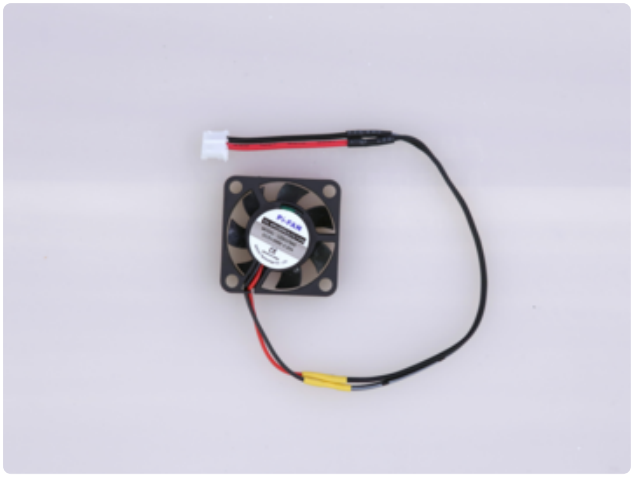


Wiring JST Cable

Solder the 2-wire ribbon cable to the 3-pin JST cable. Be sure to add heat shrink before soldering!

Wiring Mini 5V Fan Cable

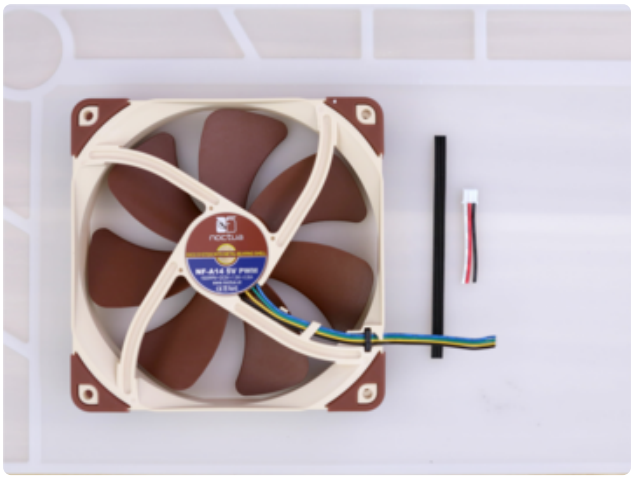
Solder the 2-wire ribbon cable to the cable on the 5V mini fan.



Wired Mini 5V Fan

Double check the ground and voltage wires polarity is correct.

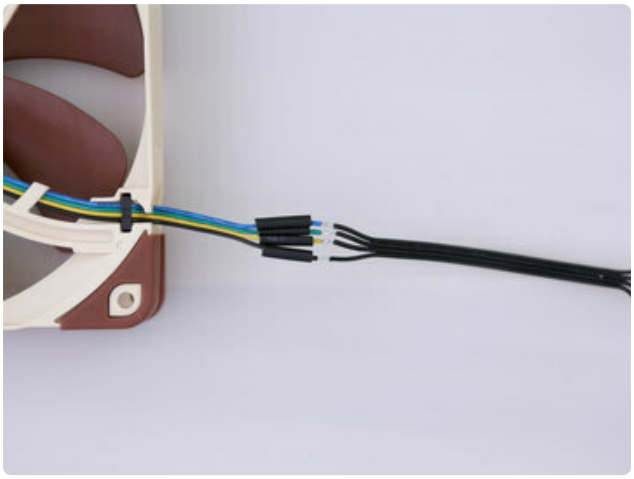
Wiring Noctua Fan



Wires fro Noctua Fan

Cut the cable from the fan short and use a piece of 4-wire ribbon cable to connect to the 3-pin JST cable.

noctua fan cable - 50mm
noctua fan 4-wire ribbon - 106mm
noctua fan 3-wire jst - 38mm

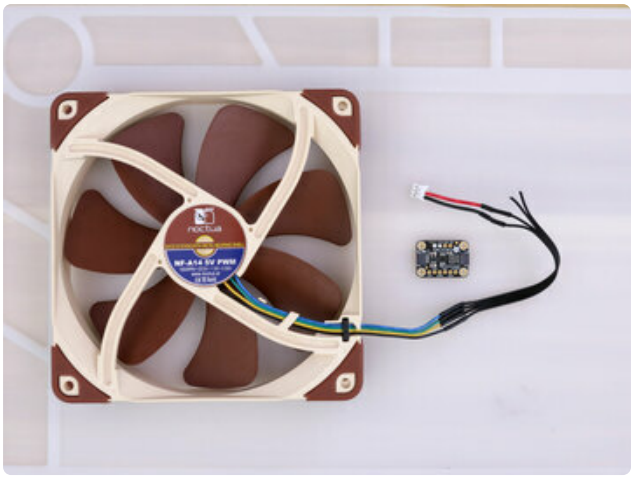


Connect 4-Wire Cable

Solder the 4-wire cable to the wires on the Noctua fan.

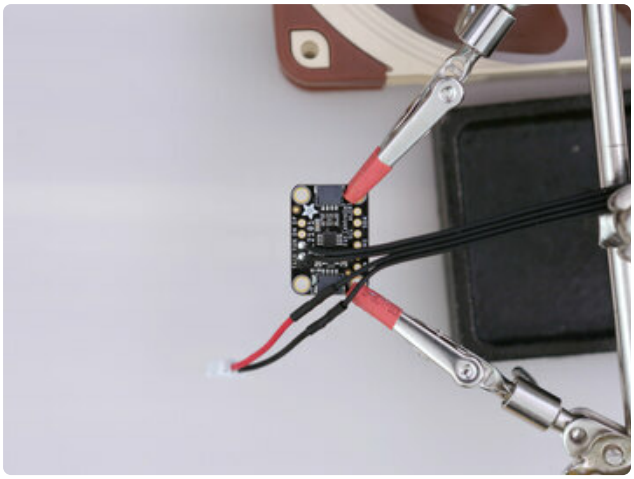
Solder JST Cable

Connect the JST cable to the black and yellow wires on the fan. The ground wire connects to the black wire. The red wire, voltage, connects to the yellow wire on the fan.



Connect EMC2101 to Fan

The remaining wires connect to the Tach and Fan pins on the EMC2101 fan controller.



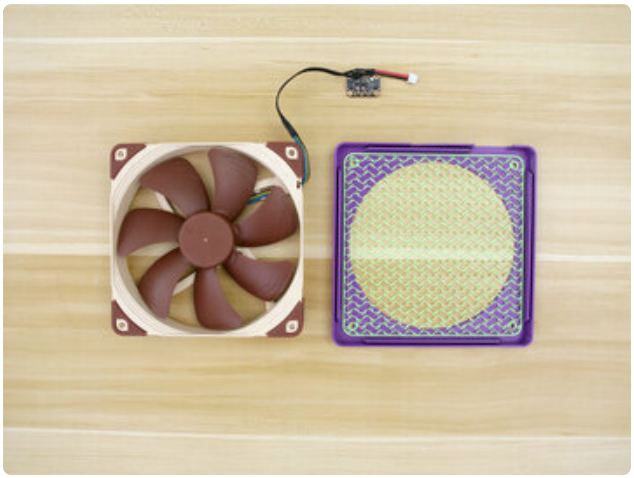
Connect Wires to EMC2101

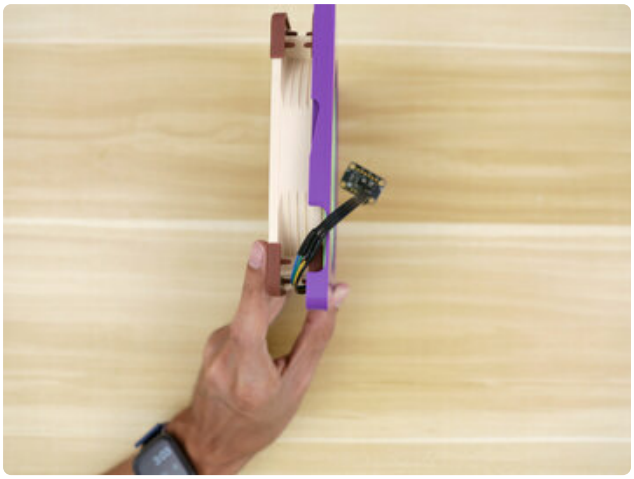
Solder the fourth wire from the fan to the FAN pin. Connect the third wire from the fan to the TACH pin on the EMC2101 board.

Wired EMC2101 and Noctua Fan

Double check the connects are solid.

Assembly





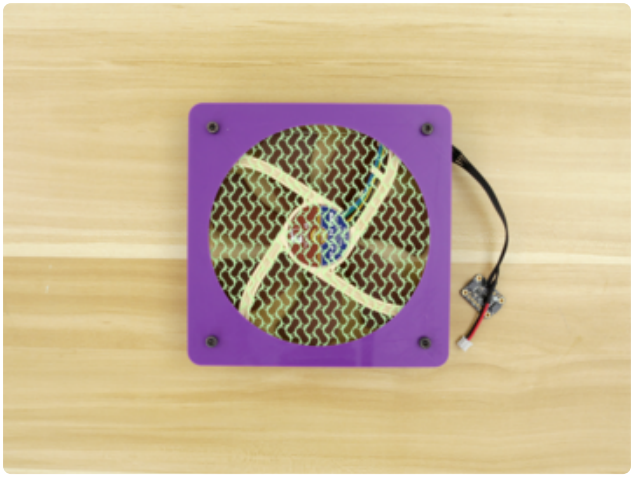
Back Cover and Fan

Place the fan grill over the back cover and line up the mounting holes. The Noctua fan is placed over the grill with the cable lined up with the wire cutout.

Secure Fan to Back Cover

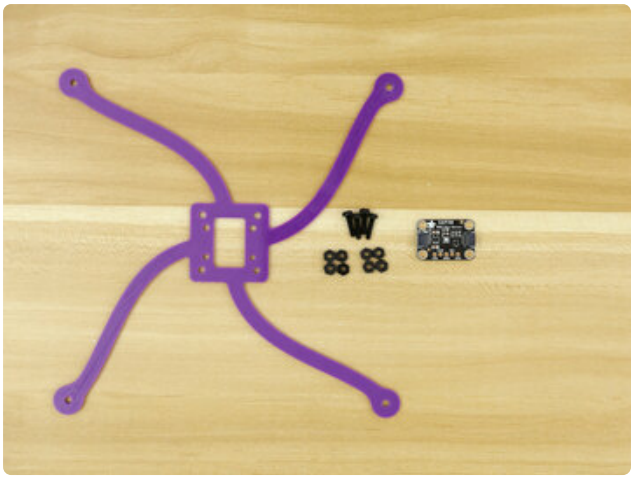
Insert M4 screws through the mounting holes while fastening hex nuts. Repeat for the other screws.

4x M4 x 10mm
4x M4 Hex nuts



Secured Fan

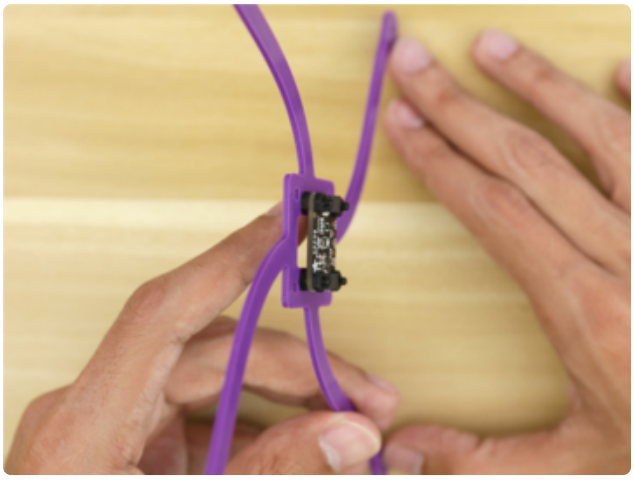
Double check all of screws are tightly fastened.



Sensor Mount Hardware

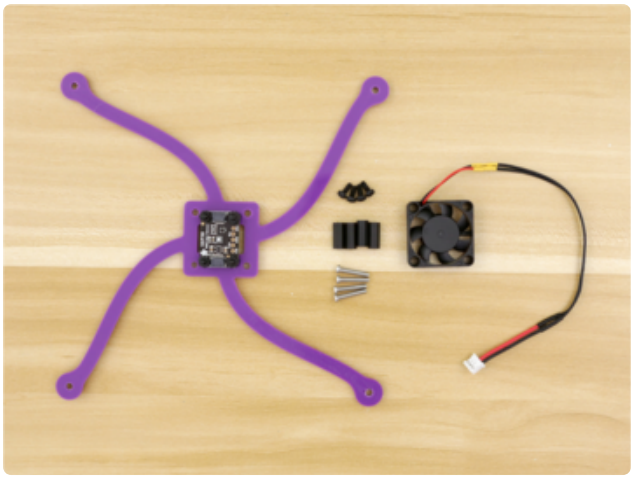
Use the following hardware to secure the sensor to the bracket. Insert M2.5 screws through the four inner holes and fasten hex on the other side of the bracket.

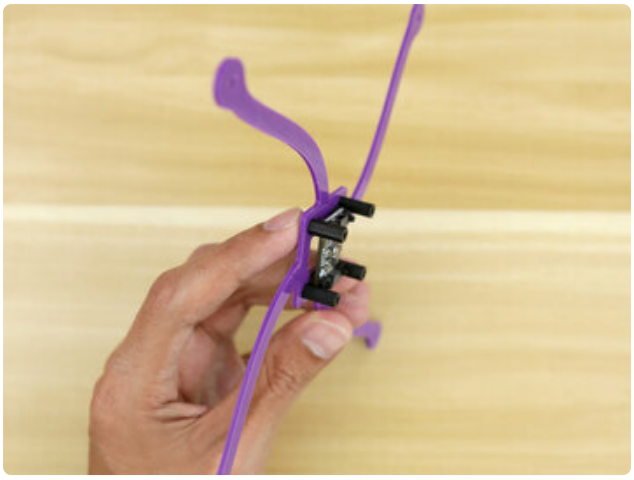
M2.5 x 10mm screws
8x M2.5 hex nuts



Secured Sensor

Place the SPG30 sensor over the screws and fit through the mounting holes. Insert and fasten hex nuts to secure the PCB to the screws.

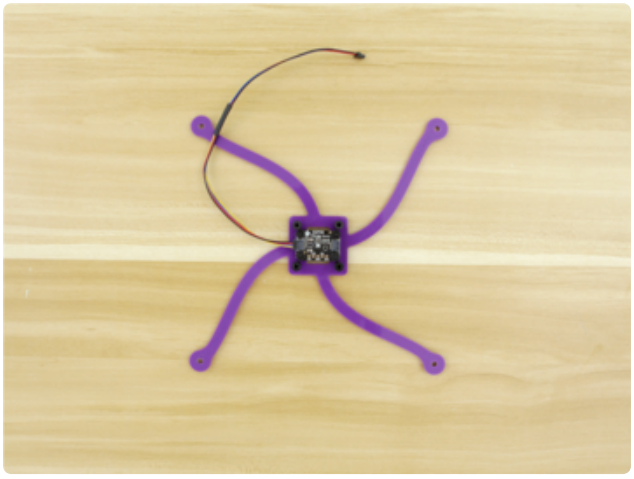




Mini Fan Hardware

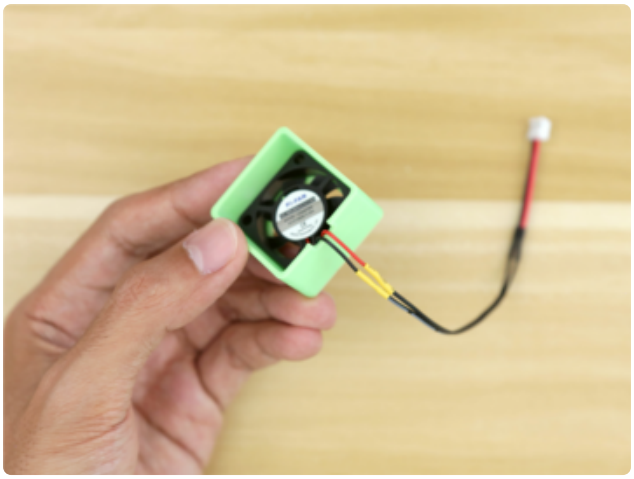
Use the following hardware to secure the mini fan to the sensor bracket. Insert the M2.5 6mm long screws through the outer holes on the sensor bracket and fasten the M2.5 standoffs.

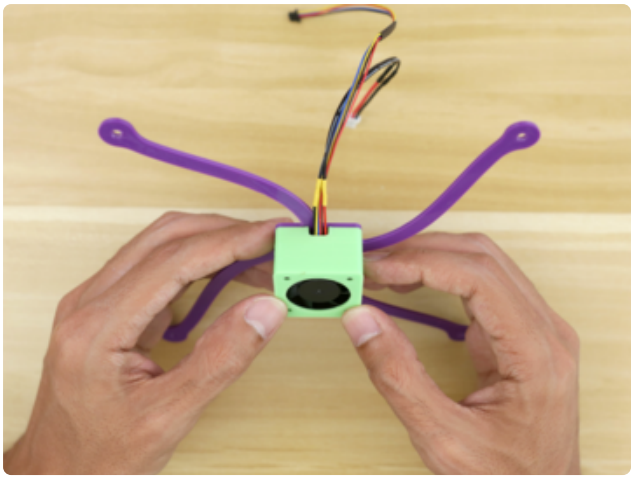
4x M2.5 FF Standoffs – 12mm
4x M2.5 screws – 6mm
4x M2.5 screws – 12mm



Connect STEMMA Cable

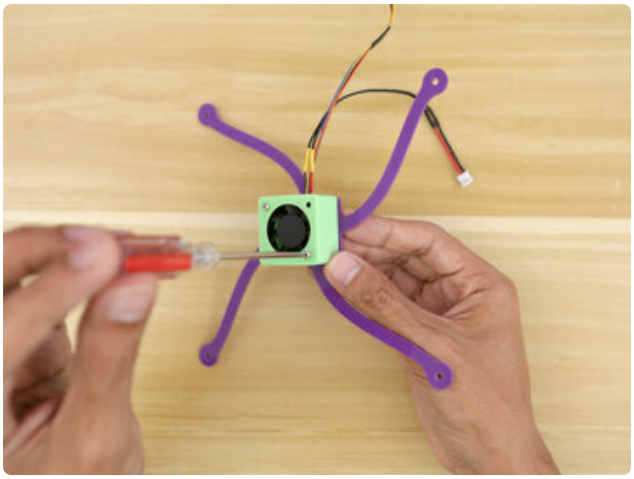
Plug in the 200mm long STEMMA QT cable to the SGP30 sensor.

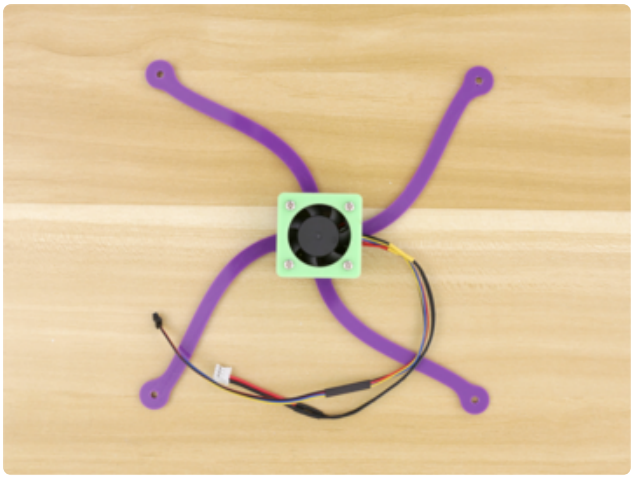




Mini Fan Cover Install

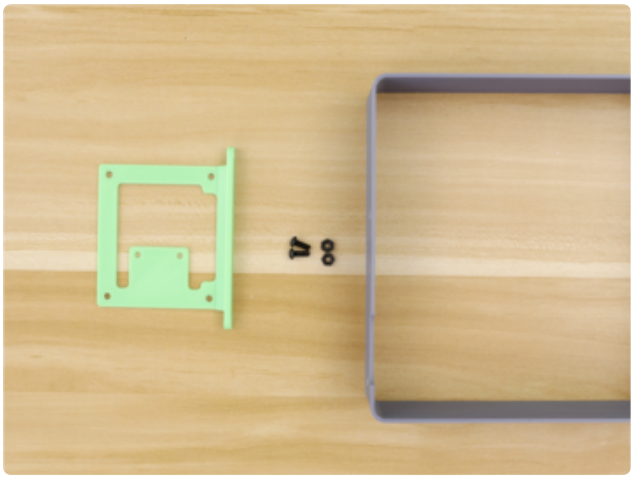
Fit the mini fan into the fan cover with the fan blades facing down. Orient the mini fan so the cable is fitted through the wire cutout. Place the fan cover over the sensor bracket with the cables matching in orientation.

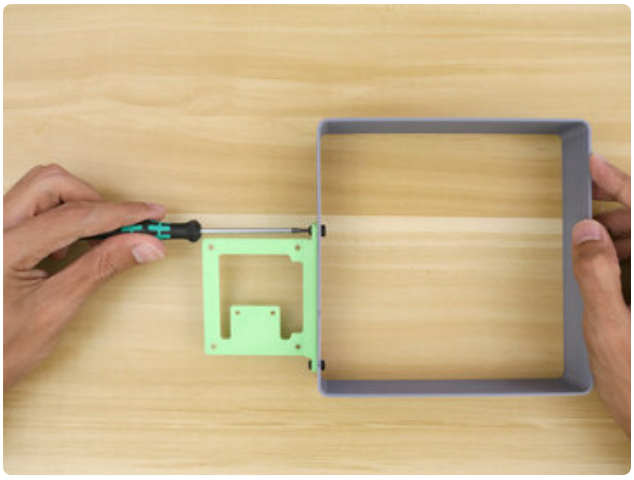




Secure Mini Fan to Sensor Bracket

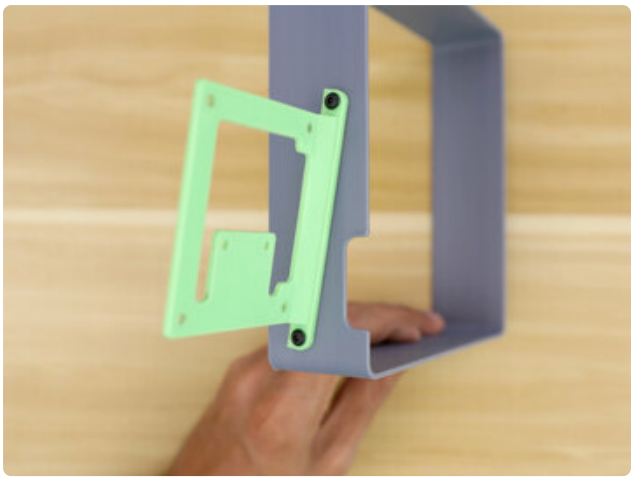
Line up the mounting holes and insert the longer M2.5 screws. Fasten the four screws tightly to secure the fan and cover to the sensor bracket.





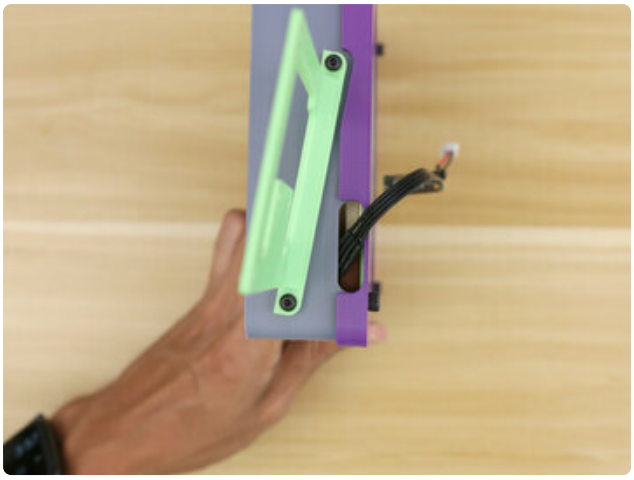
Secure PCB Mount to Frame

Use two M3 x 8mm long screws and hex nuts to secure the PCB mount to the side of the frame.



Secured PCB Mount

Double check the orientation of the PCB mount and the frame. Use the cut out to line up the parts.

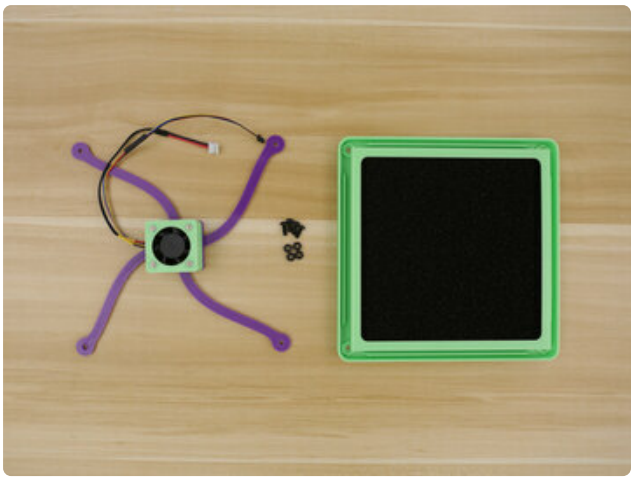


Install Frame to Back Cover

Place the frame over the back cover with the walls snap fitting into the channels. Fit the fan's cable through the wire cutout in the back cover.

Install Carbon Filter

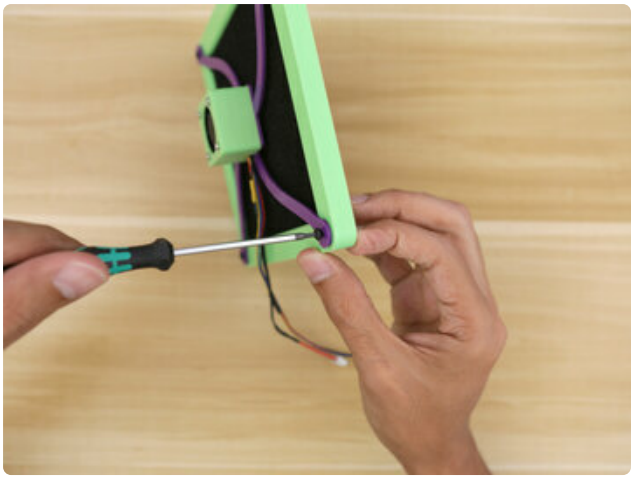
Press fit the carbon filter into recess the back of the cover. The filter cover is snap fitted over the filter to keep it secured and away from the fan blades.



Secure Sensor Bracket to Front Cover

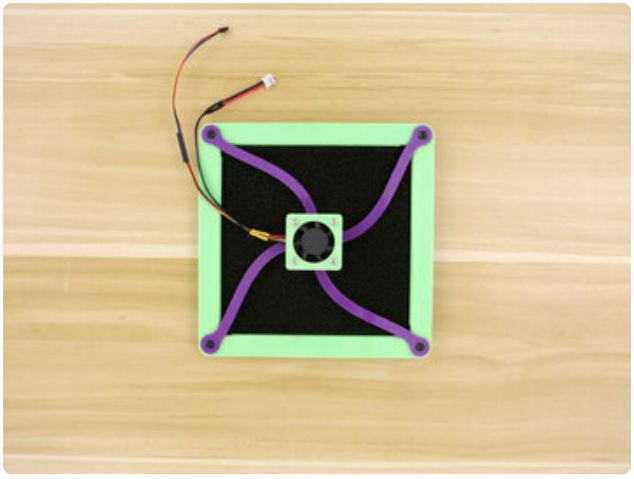
Use the following hardware to attach the sensor bracket to the front cover.

4x M3 x 8mm screws
4x M3 hex nuts



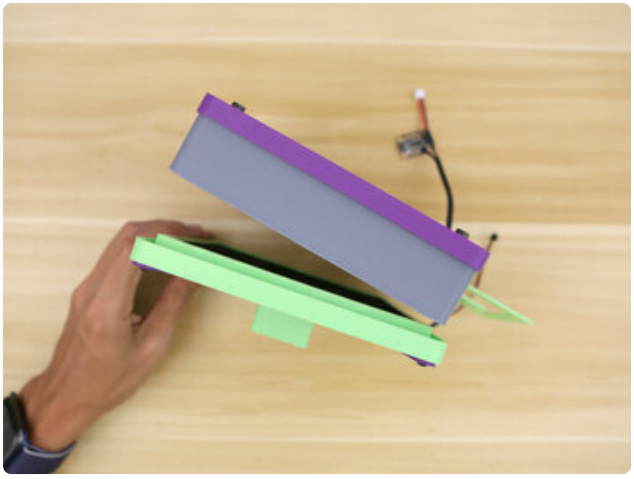
Installing Sensor Bracket

Hex nuts are placed into the recesses on the back of the cover. Place the sensor bracket over the front cover and line up the mounting tabs. Insert and fasten the M3 screws to secure the sensor bracket to the front cover.



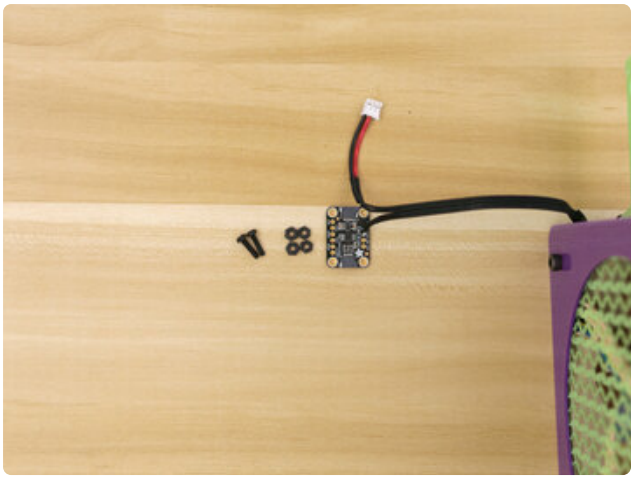
Sensor and Mini Fan Assembled

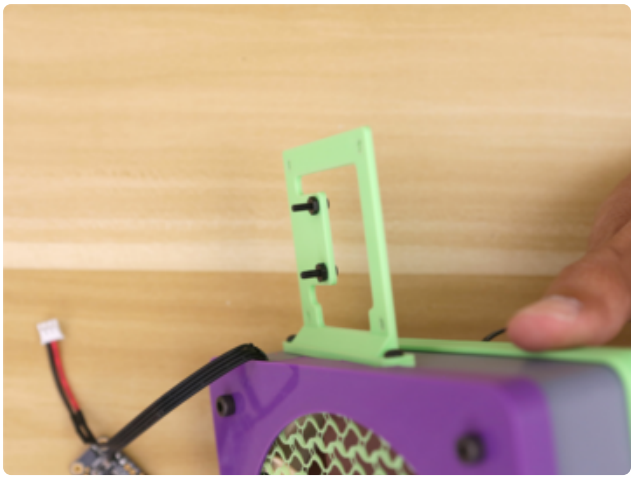
Double check the screws are tightly fastened and secured.



Install Front Cover to Frame

The front cover can now be snap fitted over the frame.

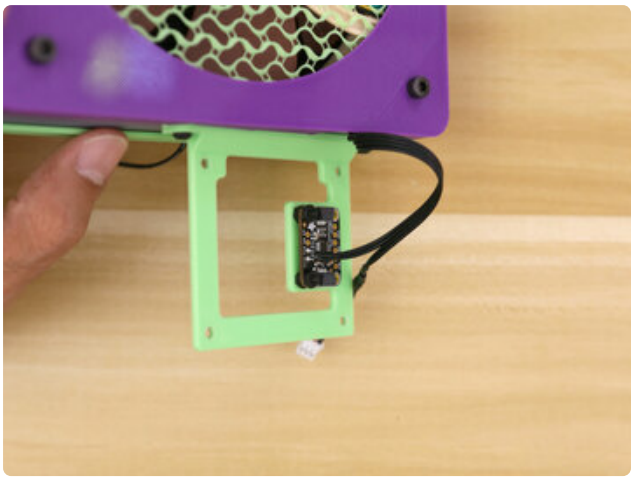




EMC2101 Hardware

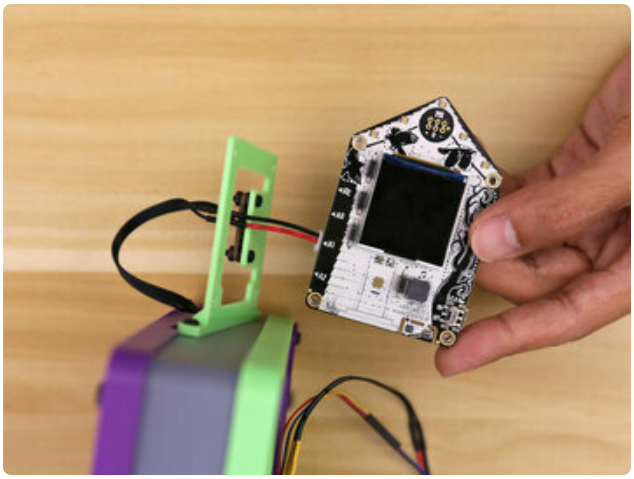
Use the following hardware to secure the fan controller to the PCB mount. Insert the M2.5 screws through the two mounting holes on the PCB mount. Fasten hex nuts over the screws to secure them to the mount.

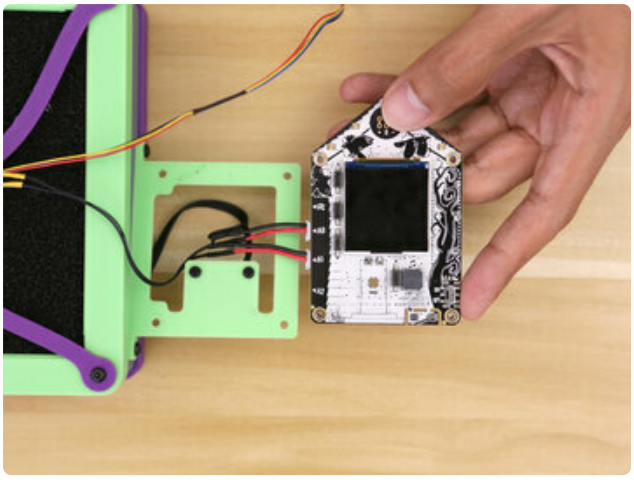
2x M2.5 x 10mm screws
4x M2.5 hex nuts



Secure EMC2101 to PCB Mount

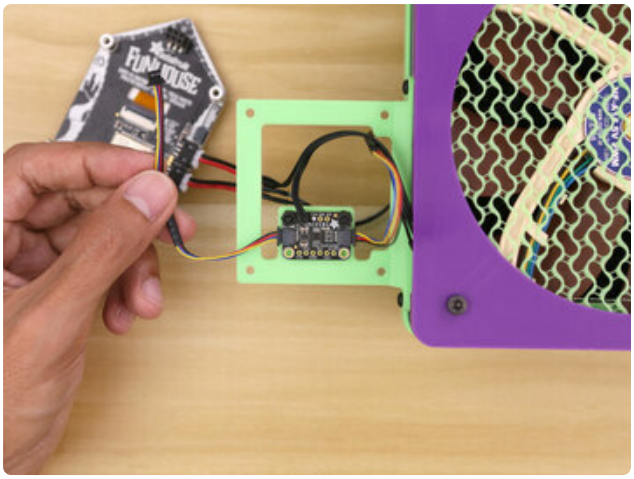
The fan controller is fitted onto the PCB mount and secured with hex nuts.

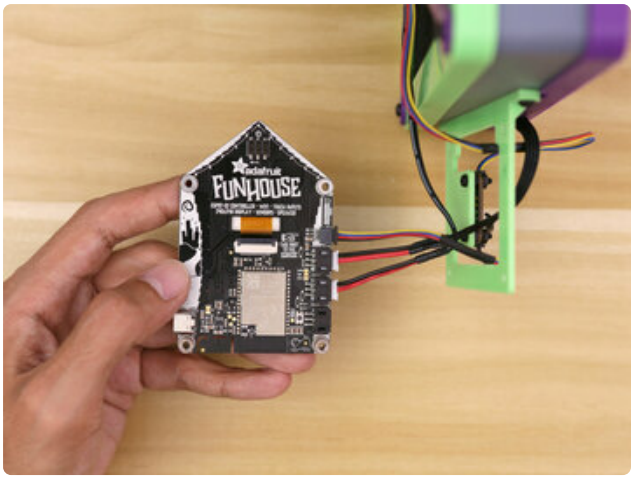




Connect JST Cables to FunHouse

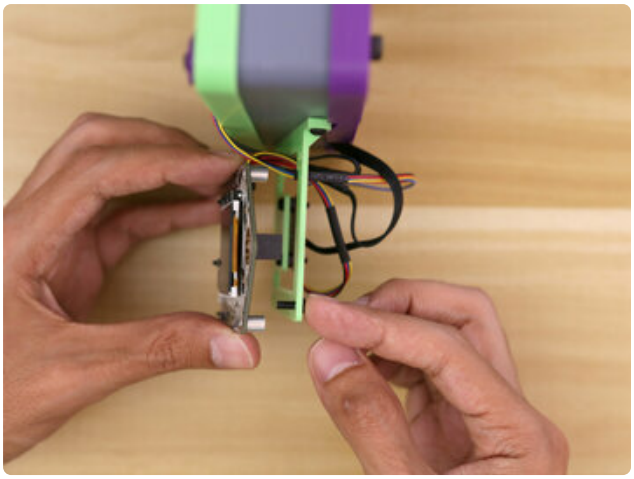
Plug in the JST cable from the mini fan to the A0 port on the FunHouse. Then connect the JST cable from the Noctua fan to the A1 port.

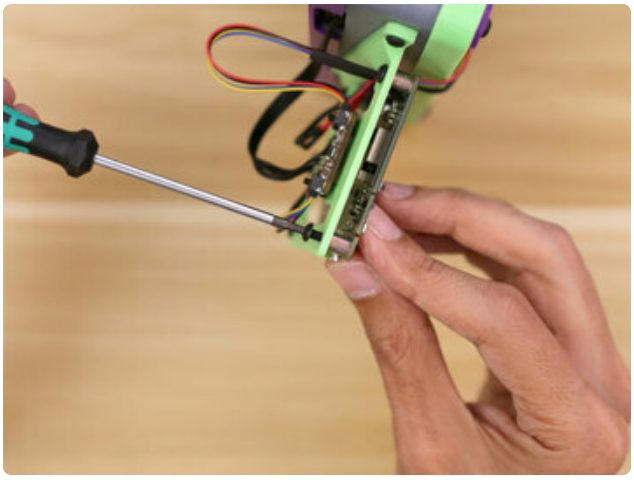




Connect STEMMA to FunHouse

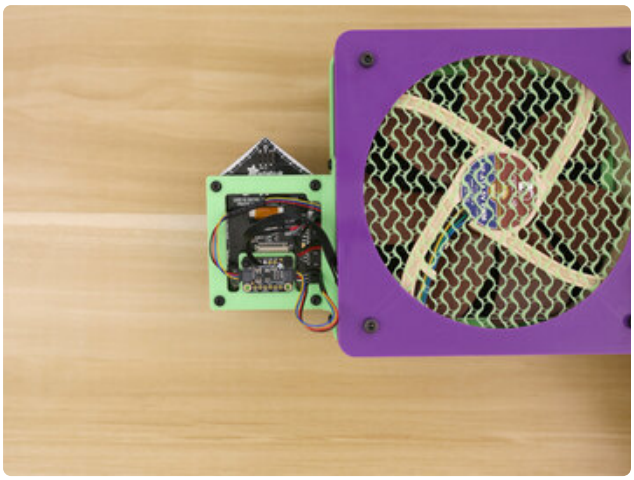
Connect the 200mm STEMMA QT cable to the EMC2101 board. Connect the 100mm STEMMA QT cable to the available connector on the EMC2101. Then, connect the other end to the I2C port on the FunHouse. Double check the daisy chain order is correct.





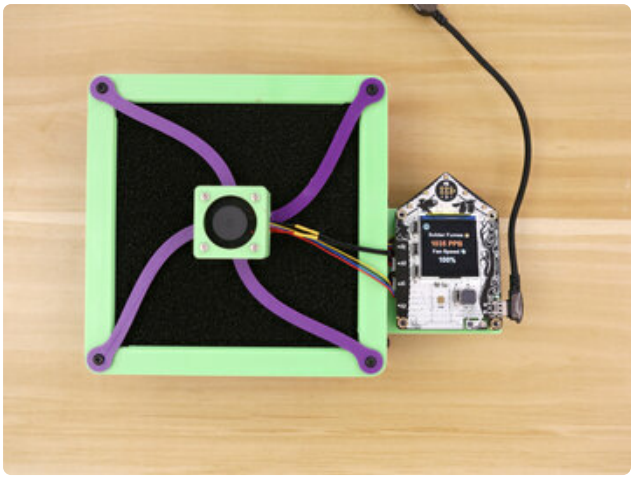
Secure FunHouse to PCB Mount

Place the FunHouse over the PCB mount and line up the mounting holes. Insert and fasten 4x M3 x 6mm screws to secure the PCB.



Secured FunHouse

Adjust the cables and double check the wires are no being pinched or kinking.



Final Build

Congratulations on your build! Use a USB type C cable and a 5V power supply to power on the FunHouse. Don't forget to check the on/off switch.