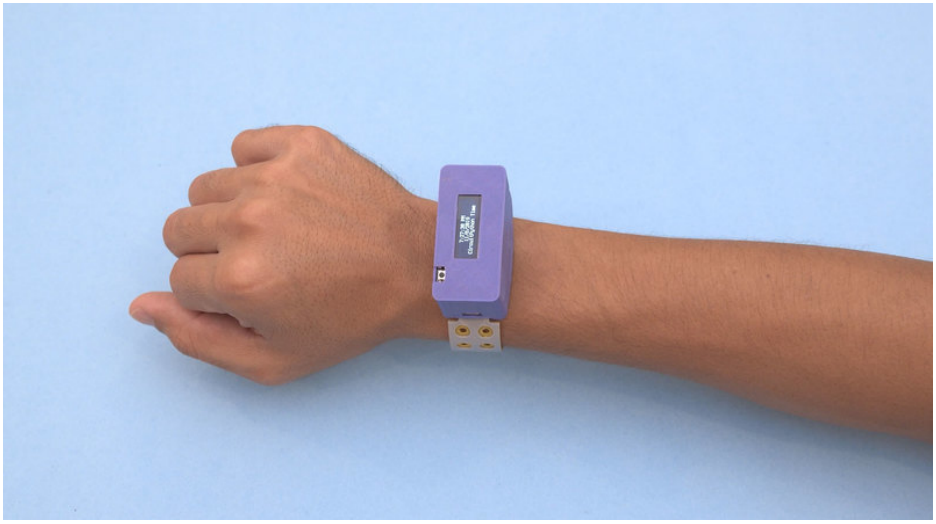
adafruit learning system

CircuitPython OLED Watch Clock

Created by Ruiz Brothers

Last updated on 2020-02-18 09:16:16 PM UTC

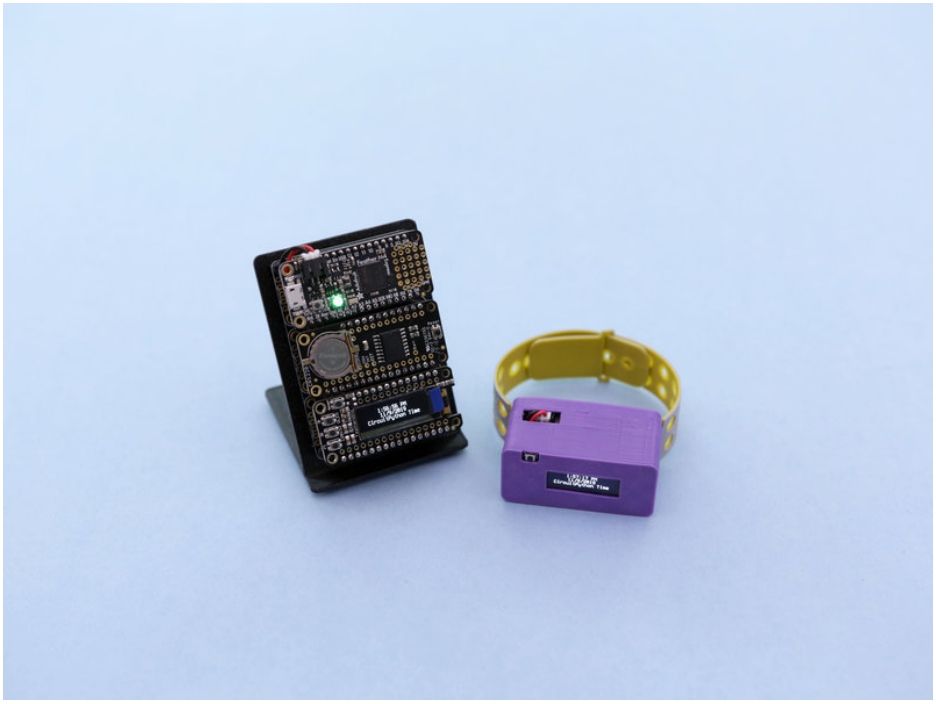
Overview



It's CircuitPython Time!

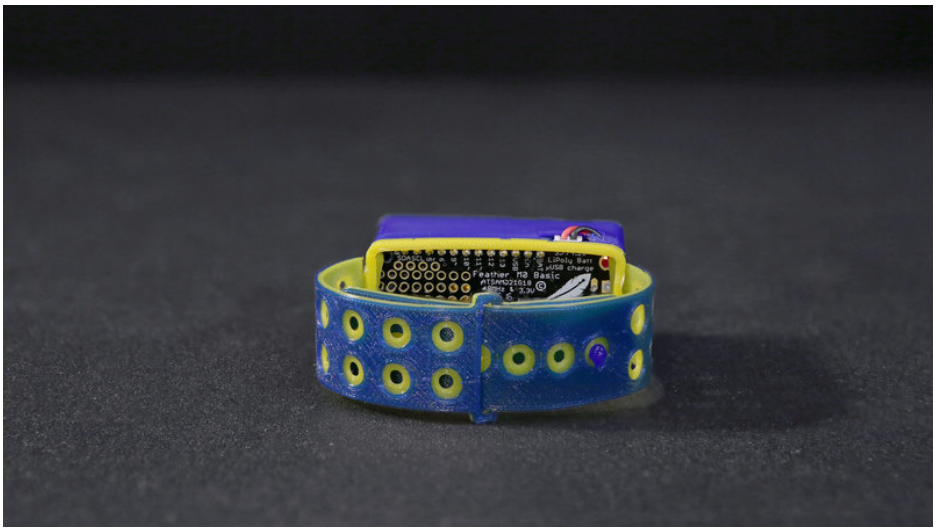
In this project we'll show you how to make a watch using an Adafruit Feather M4 Express, an RTC module and an OLED display.

This uses CircuitPython, which is a version of Python designed to run on microcontrollers.



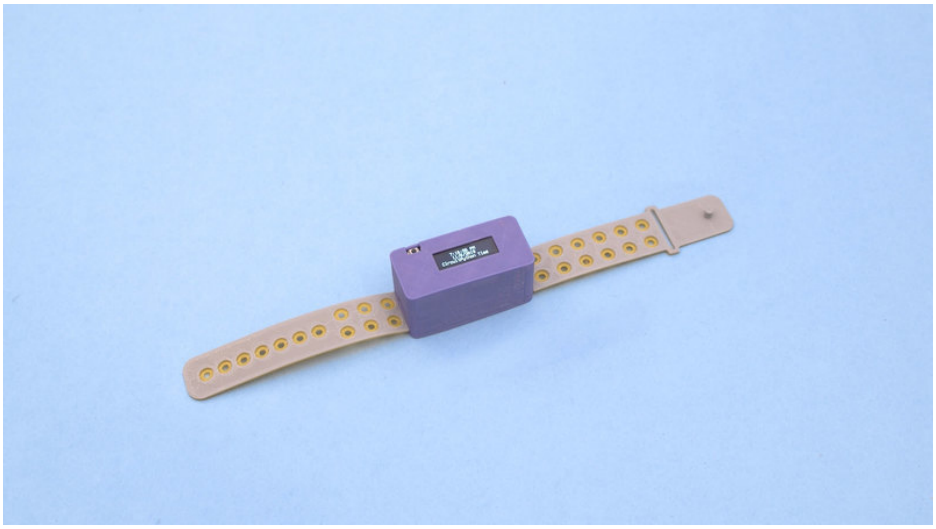
3D Printed Watch Band

The enclosure and wristband are 3D printed in ninjaflex, so it's both strong and comfortable to wear.



The whole circuit fits into the 3D printed case and has a nice snug fit.

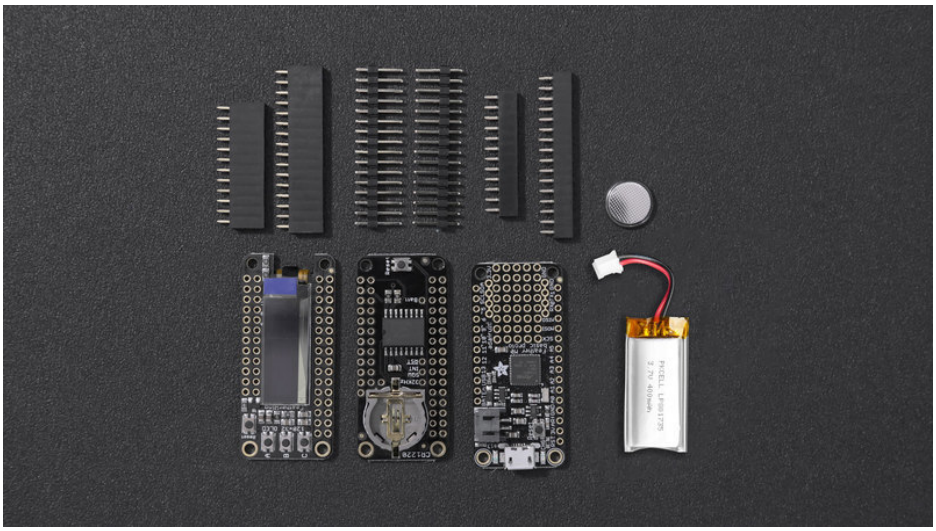
The two straps are fitted into the slots on the side and can be customized to fit different sized wrists.

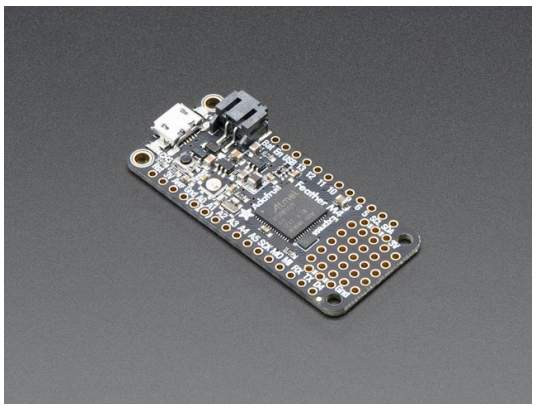


Prerequisite Guides

We suggest walking through the following guides to get a better understanding of the components and CircuitPython. We also have great tutorials on learning how to solder.

- Adafruit Guide to Excellent Soldering (https://adafru.it/sCa)
- Adafruit Feather M4 Express (https://adafru.it/CJN)
- Welcome to CircuitPython (https://adafru.it/cpy-welcome)

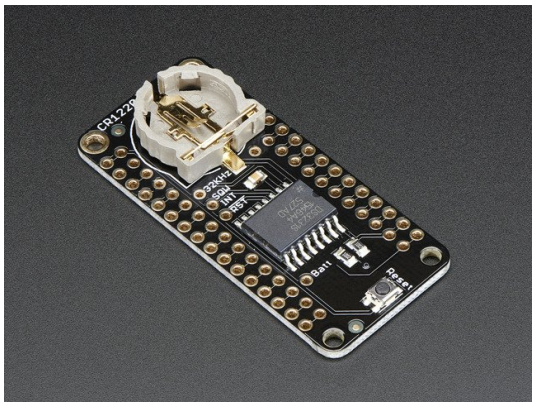




Adafruit Feather M4 Express - Featuring ATSAMD51

OUT OF STOCK

Out Of Stock

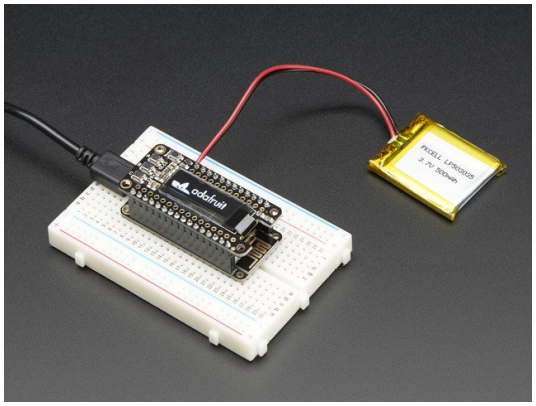


DS3231 Precision RTC FeatherWing - RTC Add-on For Feather Boards

$13.95

IN STOCK

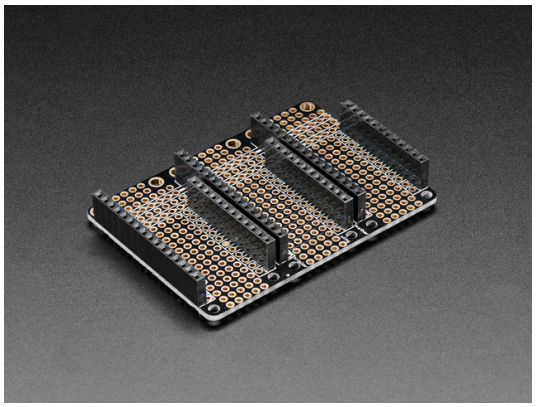
Add To Cart



Adafruit FeatherWing OLED - 128x32 OLED Add-on For Feather

$14.95
IN STOCK

Add To Cart



FeatherWing Tripler Mini Kit - Prototyping Add-on For Feathers

$8.50
IN STOCK

Add To Cart

1 x Coin Cell

CR1220 coin cell batteries

Add To Cart

1 x 400mAh Battery

3.7v idle for Feathers

Out Of Stock

1 x Feather Headers

Short Male Headers

Add To Cart

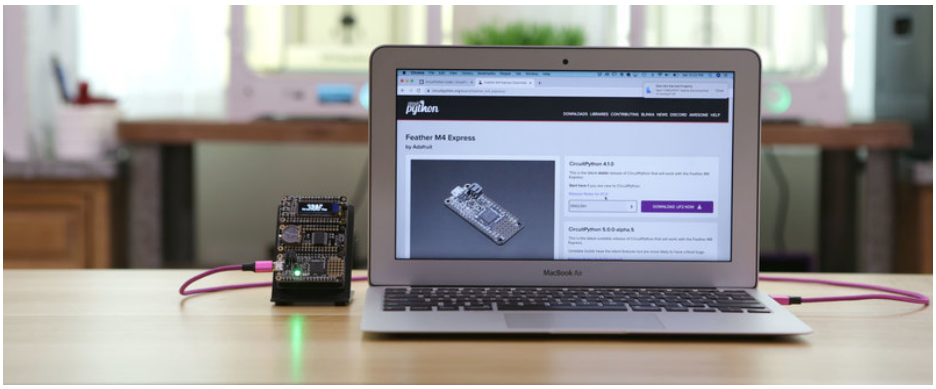
Parts, Tools and Supplies

You'll need the following tools, parts and supplies to complete this build.

- Feather M4 Express (https://adafru.it/Cmy)
- RTC FeatherWing (http://adafru.it/3028)
- OLED FeatherWing (http://adafru.it/2900)
- Tripler FeatherWing (https://adafru.it/wfw)
- CR1220 Coin Cell (http://adafru.it/380)
- 400mAh 3.7v Battery (https://adafru.it/D7i)
- Short Male Headers for Feathers (http://adafru.it/2940)
- Female Header set for Feathers (http://adafru.it/2886)
- 3D Printer (https://adafru.it/diH)

- Flexible Filament (https://adafru.it/sCc)
- Soldering Iron & Solder (https://adafru.it/vME)
- Diagonal Flush Snips (https://adafru.it/dxQ)

CircuitPython Code



CircuitPython makes it easy to get your watch up and running. You'll set the time and date, and you can change the text string displayed under the time and date. This page will walk you through getting your Feather M4 Express set up with CircuitPython, and getting the right libraries installed. Then, it will walk you through the code section by section to explain what's going on.

Let's get started!

Installing CircuitPython

You'll want to make sure you're running the latest version of CircuitPython.

This project requires CircuitPython 5.x or higher. When visiting https://circuitpython.org/board/feather_m4_express/ choose 5.x or higher, which may mean choosing the "unstable" release.

https://adafru.it/Emh

https://adafru.it/Emh

For instructions on installing CircuitPython, check out the CircuitPython page in the Feather M4 Express guide (https://adafru.it/CVs). Once you've got it installed, you're all set to continue.

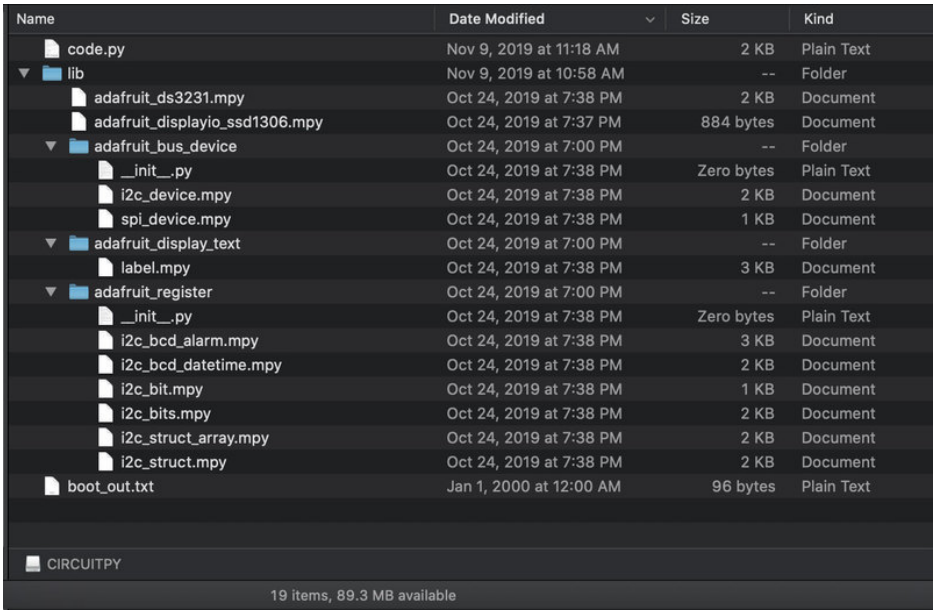
CircuitPython Libraries

This code requires some CircuitPython libraries to function. These libraries are not included with CircuitPython, so you'll need to load them yourself before the code will work. For more information, check out the Welcome To CircuitPython: CircuitPython Libraries (https://adafru.it/ABU) page.

Click the button below to download the CircuitPython library bundle. **Download the version that matches the version of CircuitPython that you are using, e.g. if you are using CircuitPython 5.x, download the 5.x bundle.**

https://adafru.it/ENC

https://adafru.it/ENC



The libraries needed for this project are:

- **Adafruit CircuitPython Bus Device**
- **Adafruit Display Text**
- **Adafruit Displayio SSD1306**
- **Adafruit DS3231**
- **Adafruit Register**

Download the latest library bundle, open the resulting zip file, and find the enclosed **lib** folder. Create a **lib** folder on your **CIRCUITPY** drive if there is not already one present. Copy the following files and folders to **CIRCUITPY/lib**:

- **adafruit_bus_device**
- **adafruit_display_text**
- **adafruit_displayio_ssd1306**
- **adafruit_ds3231**
- **adafruit_register**

Before continuing, ensure you have these files and folders in the **lib** folder on your **CIRCUITPY** drive (similar to the picture above). Once copied, you're ready to continue!

The Code

Copy the following code as **code.py** on your **CIRCUITPY** drive.

```
import board
import displayio
import adafruit_displayio_ssd1306
import terminalio
import adafruit_ds3231
from adafruit_display_text import label


font = terminalio.FONT
```

```
displayio.release_displays()

i2c = board.I2C()
display_bus = displayio.I2CDisplay(i2c, device_address=0x3c)
oled = adafruit_displayio_ssd1306.SSD1306(display_bus, width=128, height=32)

rtc = adafruit_ds3231.DS3231(i2c)


# The first time you run this code, you must set the time!
# You must set year, month, date, hour, minute, second and weekday.
# struct_time order: year, month, day (date), hour, minute, second, weekday , yearday, isdst
# yearday is not supported, isdst can be set but we don't do anything with it at this time

# UNCOMMENT THE FOLLOWING FOUR LINES THE FIRST TIME YOU RUN THE CODE TO SET THE TIME!
# import time
# set_time = time.struct_time((2019, 8, 16, 23, 59, 45, 4, -1, -1))
# print("Setting time to:", set_time)
# rtc.datetime = set_time

# Comment out the above four lines again after setting the time!


while True:
    current = rtc.datetime

    hour = current.tm_hour % 12
    if hour == 0:
        hour = 12

    am_pm = "AM"
    if current.tm_hour / 12 >= 1:
        am_pm = "PM"

    time_display = "{:d}:{:02d}:{:02d} {}".format(hour, current.tm_min, current.tm_sec, am_pm)
    date_display = "{:d}/{:d}/{:d}".format(current.tm_mon, current.tm_mday, current.tm_year)
    text_display = "CircuitPython Time"

    clock = label.Label(font, text=time_display)
    date = label.Label(font, text=date_display)
    text = label.Label(font, text=text_display)

    (_, _, width, _) = clock.bounding_box
    clock.x = oled.width // 2 - width // 2
    clock.y = 5

    (_, _, width, _) = date.bounding_box
    date.x = oled.width // 2 - width // 2
    date.y = 15

    (_, _, width, _) = text.bounding_box
    text.x = oled.width // 2 - width // 2
    text.y = 25

    watch_group = displayio.Group()
    watch_group.append(clock)
    watch_group.append(date)
    watch_group.append(text)

    oled.show(watch_group)
```

```
    oled.show(watch_group)
```

Let's take a look at the code!

Setup

First, import all the necessary libraries into the code. Note that you're not importing all the libraries you copied to your **CIRCUITPY** drive - the libraries you do import rely on the ones that you don't.

```
import board
import displayio
import adafruit_displayio_ssd1306
import terminalio
import adafruit_ds3231
from adafruit_display_text import label
```

Immediately following the `import`s is the line where the font is set. This program uses a font built into `terminalio`. This is great because it eliminates the need for separate font files.

```
font = terminalio.FONT
```

Next, you'll set up the hardware.

First, you must include the `displayio.release_displays()` because the displays do not automatically reset like everything else does. Without this line, each time the code runs, it creates another instance of the display.

Then you create the OLED display and the real time clock objects.

```
displayio.release_displays()

i2c = board.I2C()
display_bus = displayio.I2CDisplay(i2c, device_address=0x3c)
oled = adafruit_displayio_ssd1306.SSD1306(display_bus, width=128, height=32)

rtc = adafruit_ds3231.DS3231(i2c)
```

Setting the Time

The next section of the code allows you to set the date and time. You must set the date and time manually initially. You'll use `time.struct_time` to set the real time clock time.

`time.struct_time` expects you to set, in order:

- **year, month, day, hour, minute, second, weekday, yearday, isDST**

To set the time, you'll change the data provided to `time.struct_time` to match a time slightly in the future. Uncomment the four lines of code under `# UNCOMMENT THE FOLLOWING FOUR LINES THE FIRST TIME YOU RUN THE CODE TO SET THE TIME!` Then, when it is that time, save the file.

For example, if you wanted to set the date and time to August 16, 2019, 11:59:45pm, you would change the code to the following:

```
# UNCOMMENT THE FOLLOWING FOUR LINES THE FIRST TIME YOU RUN THE CODE TO SET THE TIME!
import time
set_time = time.struct_time((2019, 8, 16, 23, 59, 45, 4, -1, -1))
print("Setting time to:", set_time)
rtc.datetime = set_time

# Comment out the above four lines again after setting the time!
```

Once you've set the date and time, comment out the four lines again or it will change the time back to the manual time you set every time the code runs.

Main Loop

Next up, we start the main loop. First, set a variable to be the real time clock date-time.

```
while True:
    current = rtc.datetime
```

The next section has two parts. The first handles taking the 24-hour time provided by `rtc.datetime` and turning it into 12-hour time. The second part uses the 12-hour converted time to determine whether to display AM or PM.

```
    hour = current.tm_hour % 12
    if hour == 0:
        hour = 12

    am_pm = "AM"
    if current.tm_hour / 12 >= 1:
        am_pm = "PM"
```

The next section is everything needed to get text displaying on the OLED. Let's take a look!

First, you assign variables to the text strings displayed on the OLED.

```
    time_display = "{:d}:{:02d}:{:02d} {}".format(hour, current.tm_min, current.tm_sec, am_pm)
    date_display = "{:d}/{:d}/{:d}".format(current.tm_mon, current.tm_mday, current.tm_year)
    text_display = "CircuitPython Time"
```

You can customise this section. You can change `text_display` string to any string that fits on the display. You can also change the order of the `current` time values, e.g. if you wanted the date order to be day/month/year, you would change the `date_display` line to:

`date_display = "{:d}/{:d}/{:d}".format(current.tm_mday, current.tm_mon, current.tm_year)`

The next section creates instances of `label` which is used to display the text. `label` requires providing a font and a string of text. Here you are using the default font specified earlier in the code, and the text strings found immediately above this section.

```
    clock = label.Label(font, text=time_display)
    date = label.Label(font, text=date_display)
    text = label.Label(font, text=text_display)
```

Now that you've created the `label` instances, you can begin to manipulate things about them like location of each string on the display.

`bounding_box` allows you to determine the `x, y` location of the label, and the `width` and `height` of the string. The `width` and `height` are the amount of space taken up by the font being used on the display. This example does not use `height`, so it is ignored.

Next you set the location of the text. The `x` value is the location left to right on the display, the `y` value is the location top to bottom. The `y` value is simply a value to position the lines of text on the display from top to bottom. The `x` value, however, requires some math to ensure that no matter how long the string is, it is always centered.

For the `x` value, you take the width of the display (in pixels) and dividing it by 2 to determine the center of the display. You then divide the width of the bounding box (the width of the string of text) by 2 in the same manner to determine the center of the bounding box. You then subtract that value from the width of the display to center the text on the display. Using `//` ensures that you end up with whole numbers, as you cannot place something on half of a pixel.

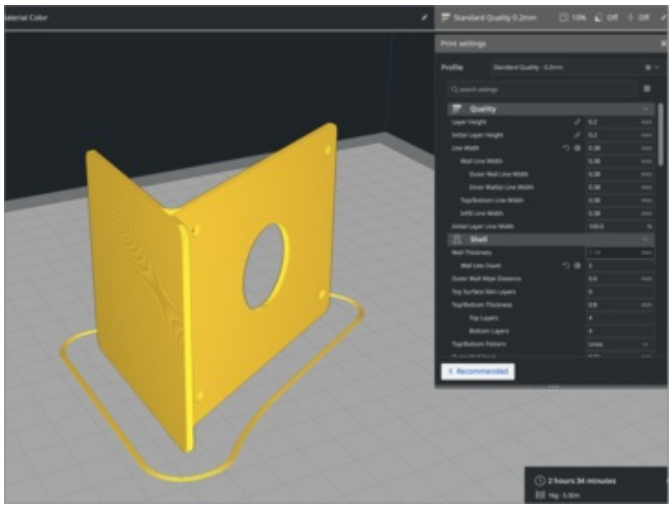
```
(_, _, width, _) = clock.bounding_box
    clock.x = oled.width // 2 - width // 2
    clock.y = 5

    (_, _, width, _) = date.bounding_box
    date.x = oled.width // 2 - width // 2
    date.y = 15

    (_, _, width, _) = text.bounding_box
    text.x = oled.width // 2 - width // 2
    text.y = 25
```

Finally, you create a group to hold all of the lines of text to display them. You create the main group, `watch_group`, and then append the three labels into the group.

Lastly, to display the `watch_group`, you must call `oled.show()`. `show()` requires you to provide the group you're expecting to display, e.g. `oled.show(watch_group)`.

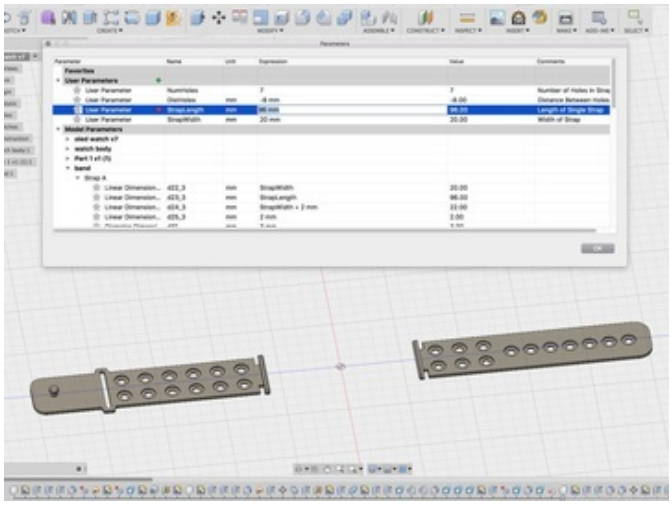
```
    watch_group = displayio.Group()
    watch_group.append(clock)
    watch_group.append(date)
    watch_group.append(text)

    oled.show(watch_group)
```

That's it! Your watch is set up and running, and now you're ready to get it assembled!



Slice Stand

Use PLA or other rigid material to print the stand. The stand is oriented to print on its side. This helps to prevent the use of support material. The first layer requires considerably strong adhesion. A large brim could be used to provide more adhesion to the bed of the 3D printer.

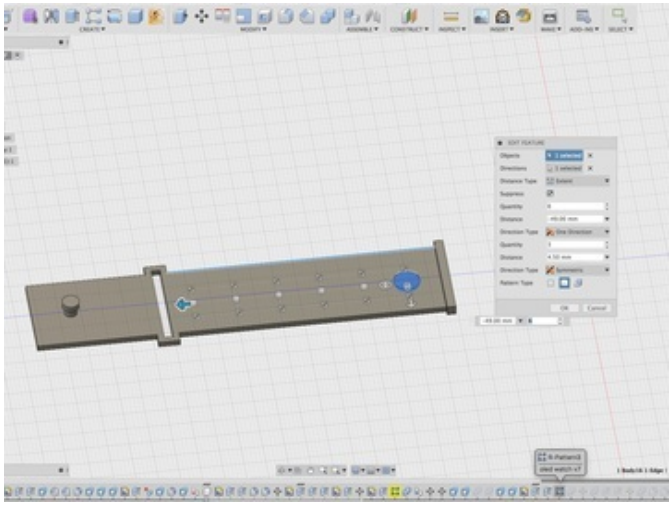


Customize Watch Band

The design of the watch band can be adjusted using Autodesk Fusion 360. Change the parameters to adjust the width, length and number of holes on the band.

Patterns

We can change the patterns for each band by editing the features in the timeline.



Download and 3D Print

The 3D printed parts can be downloaded with the link below.

https://adafru.it/sCd

https://adafru.it/sCd

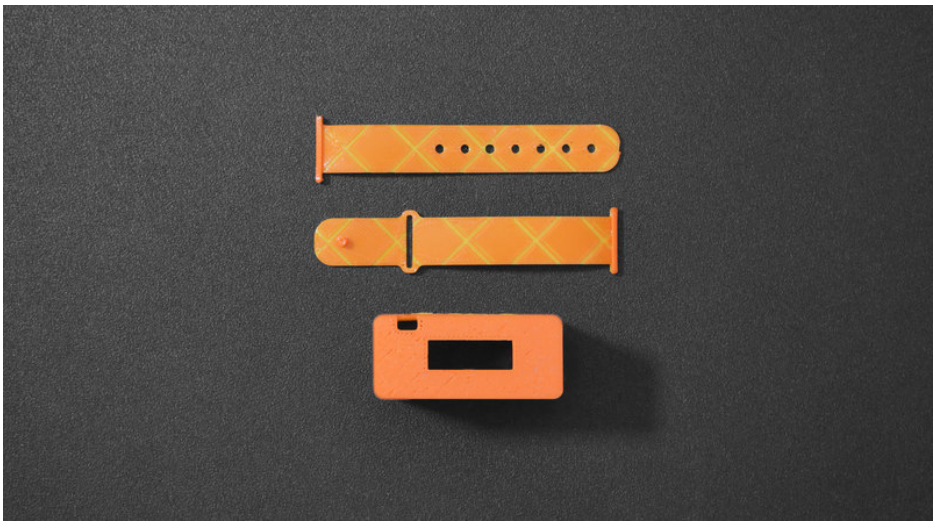
https://adafru.it/GHD

https://adafru.it/GHD

https://adafru.it/sDp

https://adafru.it/sDp

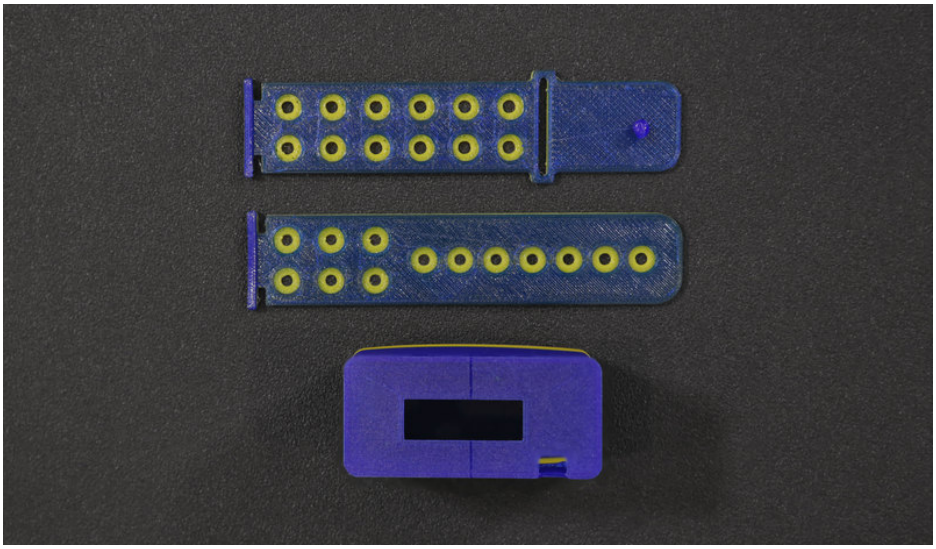
Parts are optimized for flexible materials, rigid material will not allow the components to fit.

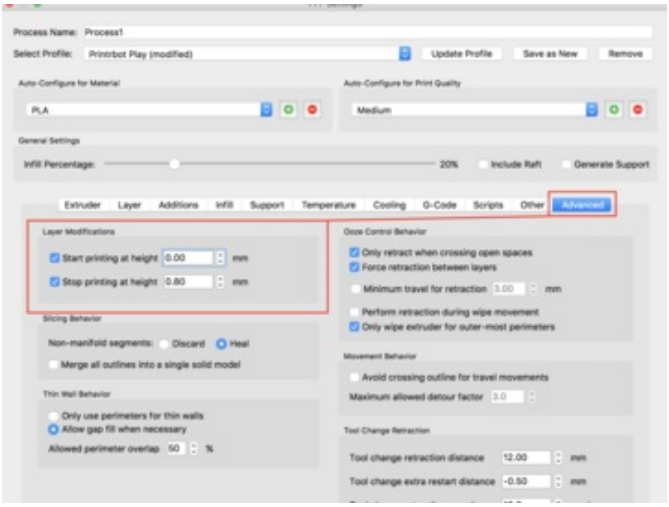


Materials & Slice Settings

This design requires an extruder capable of printing with flexible materials such as the Flashforge Creator (http://adafru.it/2742). If you're using standard Ninjaflex material (85A shore hardness), we recommend printing slow, around 20 to 40mm/s with the extruder temperature set to 240c.

You can also use Cheetah Ninjaflex (https://adafru.it/sCe), which has a higher shore hardness (95A). The only difference will be in how flexible and how grippy the texture is.





Dual Color Printing

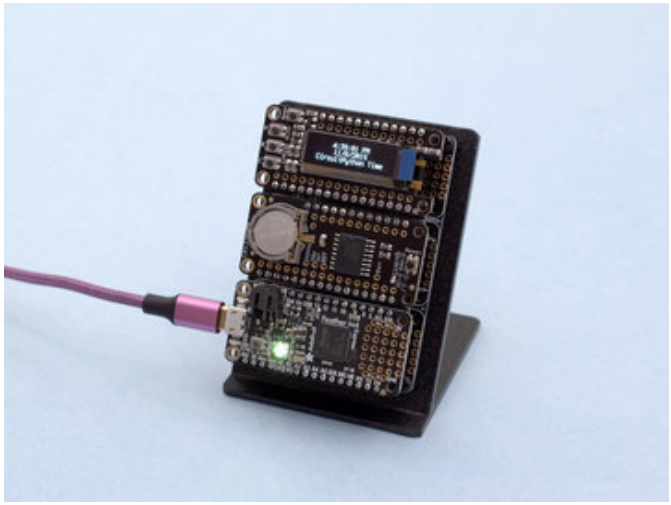
You can achieve the two tone colors using a single extruder by using 2 different gcode files. We'll swap filaments in between the print jobs. You'll need to leave the print on the bed after the first job is completed and then print the second gcode file.

For printing the two wrist bands, the first process will print from **0mm to 0.8mm** – you can set this up in **Simplify3D** by enabling the start/stop options in the **Layer Modifications** section, in the **Advanced** tab. The second process will start at layer 0.8mm and can print the rest of the part.

oled-body.stl oled-starpA.stl oled-starpB.stl	PLA 245c / 0c bed No supports 10% infill 40mm/s print speed No Retraction 120mm/s travel speed	Multiple gcode processes are created to get a dual color print for the bands. Print from 0mm to 0.8mm to print the first bottom color. Then print from 0.8mm to 6mm to print the top colors of the band. The body was printed from 0mm to 23mm when swapping colors.

Retraction clean up

Flexible materials may need retraction disabled to print successfully, so we'll need to clean any left over material around the slots and grooves on the **GP-body.stl** part.

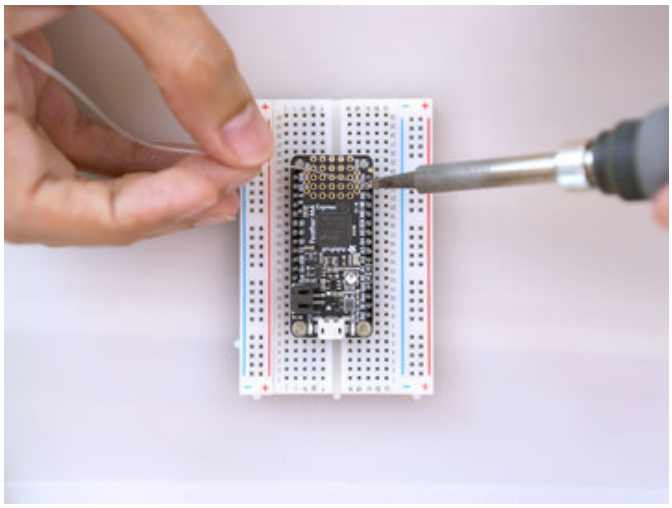


Clock Stand

You can create a desktop clock using the same CircuitPython code and FeatherWings. A 3D printed stand can house an Adafruit Tripler FeatherWing. The Feather M4, RTC module and OLED FeatherWing snaps on via female headers.

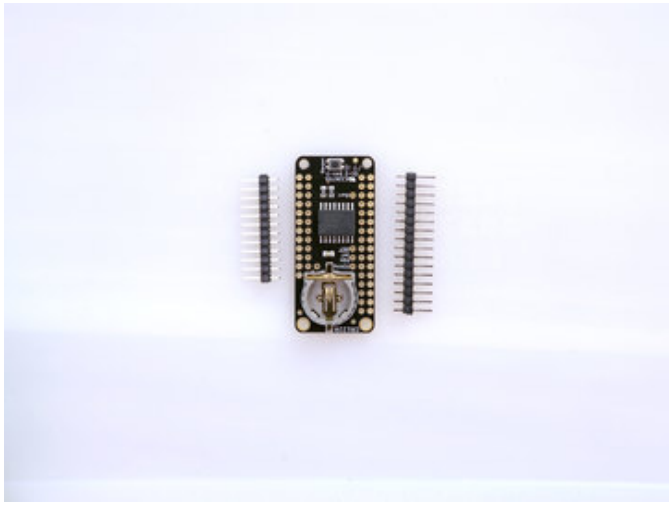
Install Feather M4 Headers

Use the male headers that came with the Feather M4. Use flush diagonal cutters to cut the plastic strip so there's a 12-pin and 16-pin.



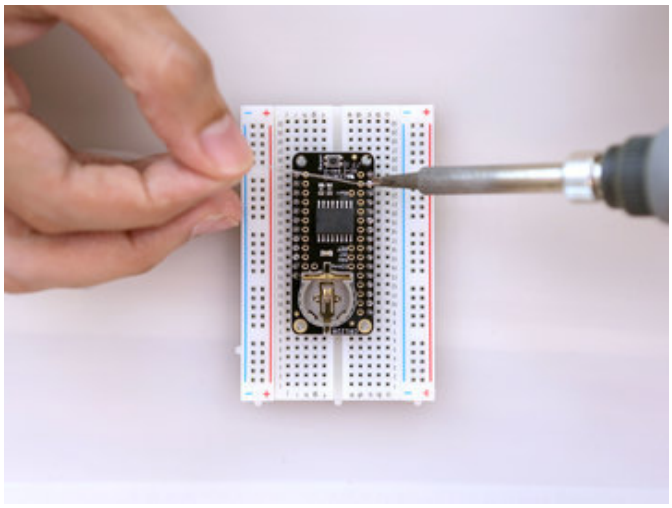
Solder Headers

Insert the two header pins through the bottom of the PCB. A breadboard can assist you by holding them in place while soldering. Firmly press the header pins into the breadboard, making sure the are inserted straight. Apply solder to the top of the pins. Solder all of the pins to the headers.



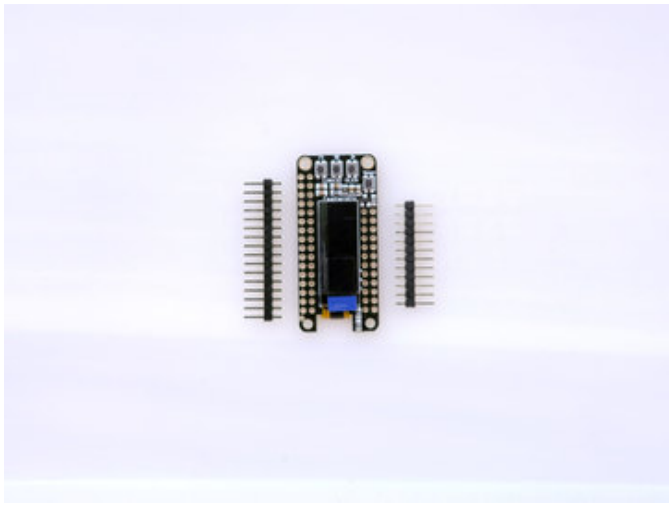
Install RTC FeatherWing Headers

Use the two sets of headers that came with the RTC FeatherWing. Use flush diagonal cutters to trim the headers so you have a 12 and 16-pin strip of headers.



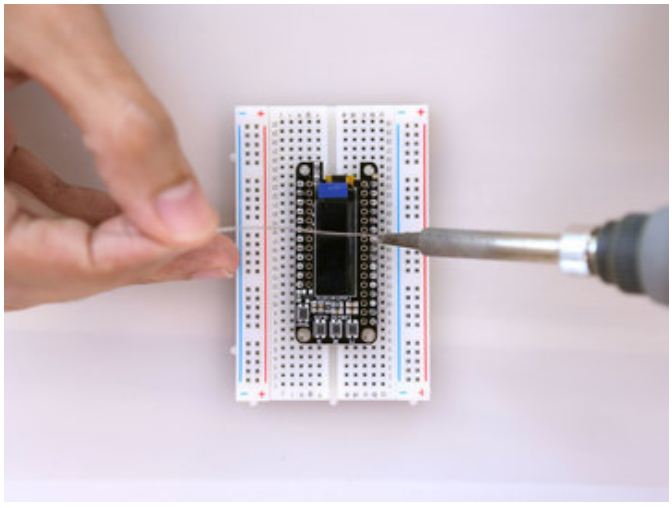
Solder Headers

Using the same breadboard technique as the Feather M4, install the headers to the bottom of the PCB and insert them into the breadboard. Apply solder to all of the pins. Make sure the strips of headers are flush with the PCB.



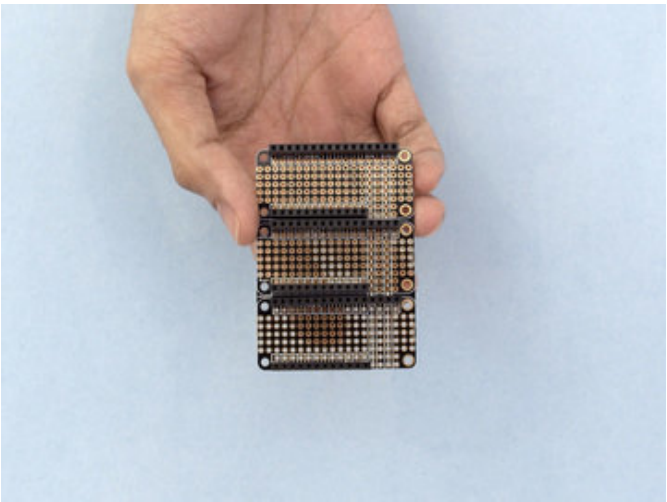
Install OLED FeatherWing Headers

Get the two strips of header pins ready for the OLED FeatherWing.



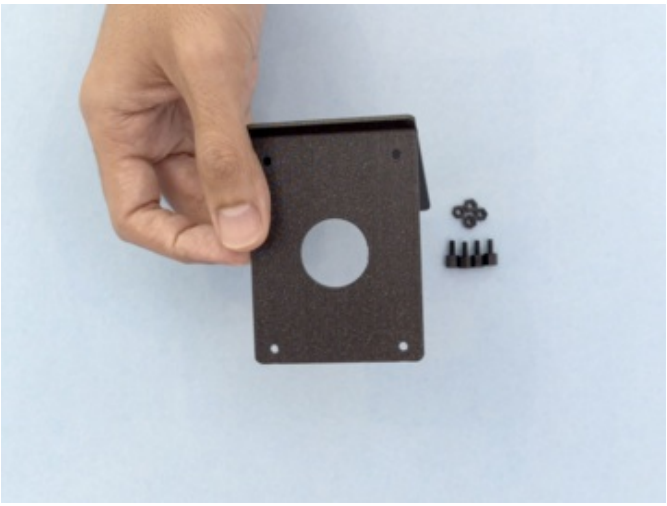
Solder Headers

Repeat the same process using the breadboard to hold the strips of headers in place while soldering.



Tripler FeatherWing

The Tripler FeatherWing is designed to house up to three Feathers / FeatherWings. You need to use three sets of female header pins, a 12-pin and a 16-pin strip. These are installed to the designated spots on the PCB. Follow the markings and labels on the PCB to insert the headers correctly. Reference the photo for correct placement.



Stand Hardware

Use the following hardware to secure the Tripler FeatherWing PCB to the stand.

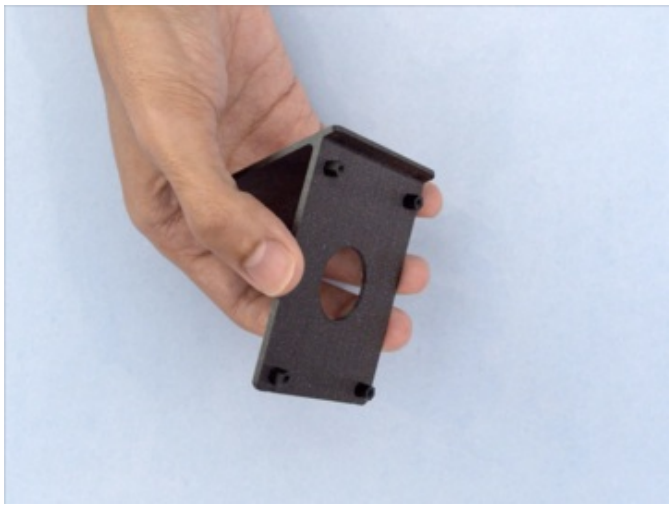
- 4x M2.5 x 6mm male-to-female standoffs
- 4x M2.5 hex nuts
- 4x M2.5 machine screws

You can optionally use tall/shorter standoffs. The black nylon standoff kit includes quite a few different sizes to choose from.

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

$16.95
IN STOCK

Add To Cart

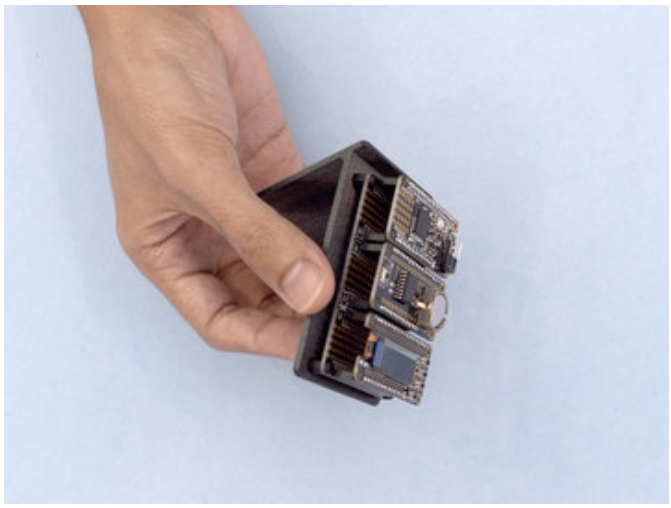


Install Hardware

The male threads on the standoffs are inserted through the four holes on the standoffs. They're M3 sizes so they should easily fit through. Use the hex nuts to secure the standoffs. Finger tighten the hex nuts, do not over tighten.

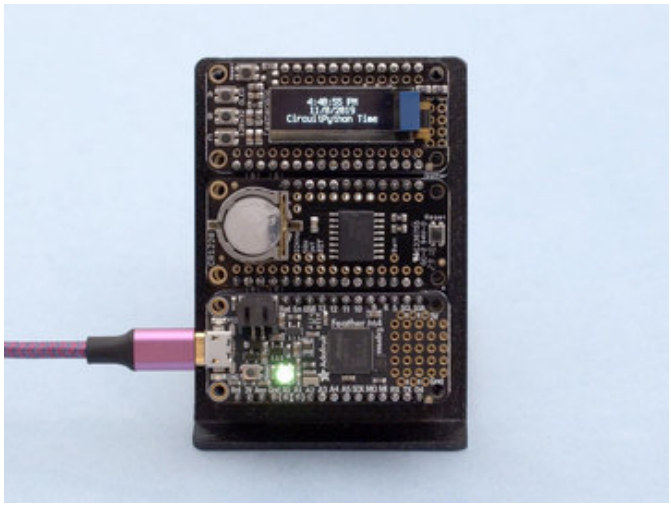
Install Tripler FeatherWing

Place the Tripler FeatherWing on top of the standoffs and line up the mounting holes. Insert and fasten the M2.5 machine screws through the top of the PCB.



Install FeatherWings

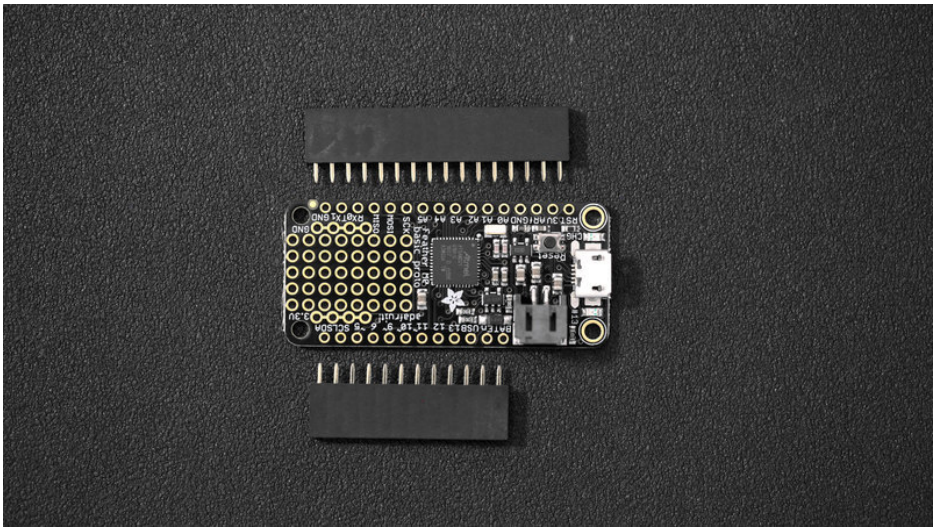
With the Tripler FeatherWing PCB secured to the stand, install the Feather M4 and FeatherWings to the female sockets. Place them in which ever order best suits your project and setup. You can easily swap them out by carefully pulling them out.



Use It

Plug-in a known good micro USB cable with a good data connection to the Feather M4. Access to the reset button and the FeatherWings allow you to test out different Feathers and FeatherWings. Have fun!

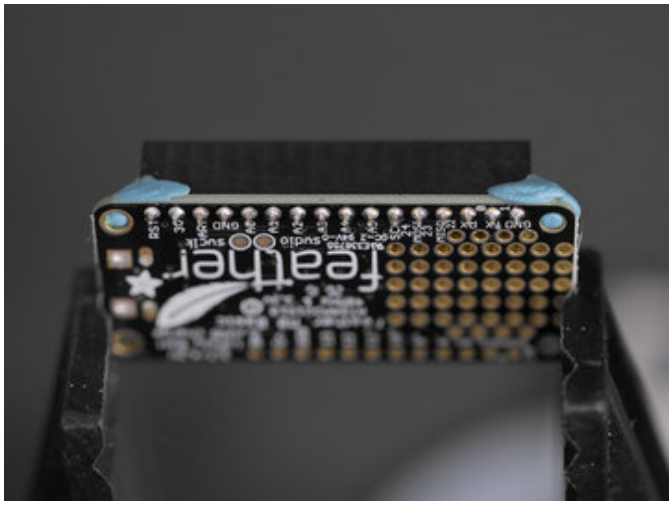
Watch Build



Headers

Add 12 and 16 female pin headers with the female side facing upwards on the Feather M4 Express board. This will let us plug the RTC above.

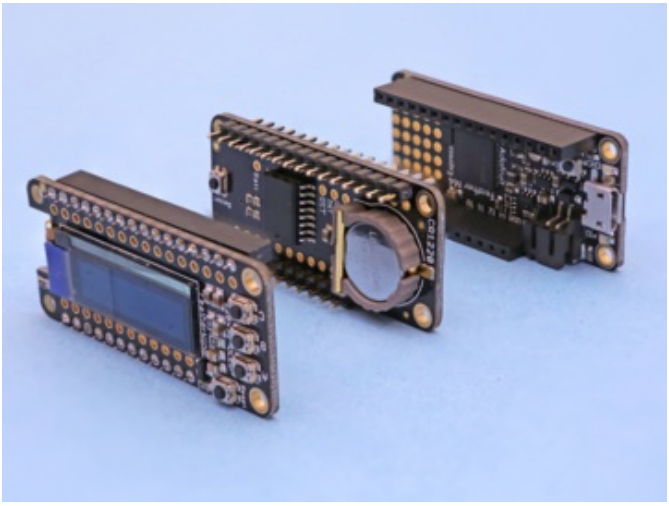
The large headers will allow more room for the JST connector and lipo battery to fit between the RTC and Feather board.



Solder Headers

Use a small amount of tac to hold the headers in place. Make sure the headers are flat against the board and then solder each pin.

We can use a Panavise to hold
the boards while soldering.



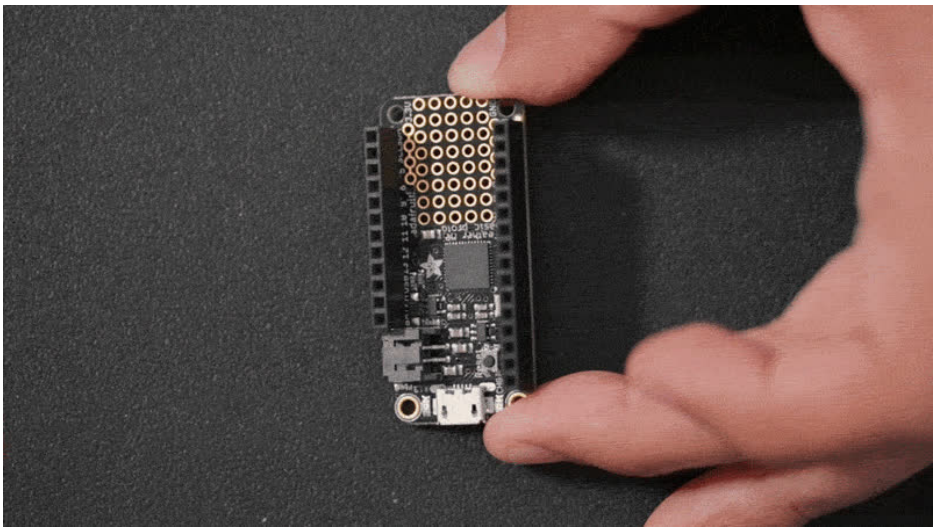
RTC

Use the header pins that came with RTC. Insert the headers with the longer pins facing downwards like shown in the picture.

You can use flush diagonal cutters to cut one of the headers to fit inside on the 12 pin side of the board.

OLED

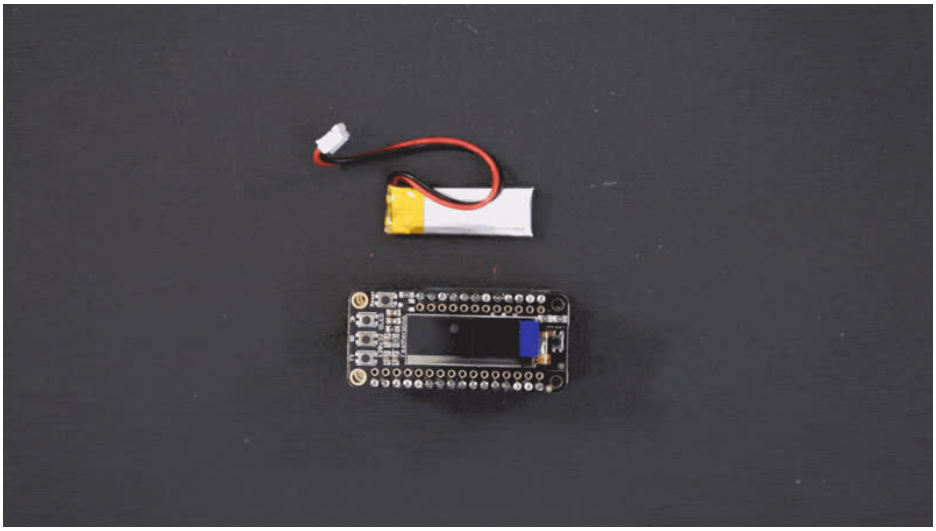
Solder the short feather headers underneath the board with the female ends facing downwards like shown in the picture.



Stacking

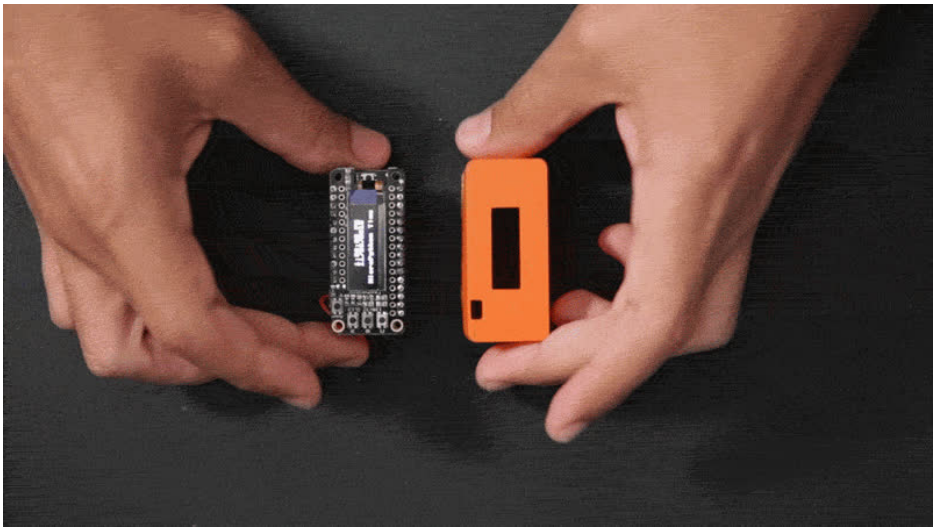
Once all of the headers are soldered on, we can stacking them on top of each other.

Add a coin cell battery to the RTC board by sliding it in with the + side up and at an angle.



Fitting Lipo Battery

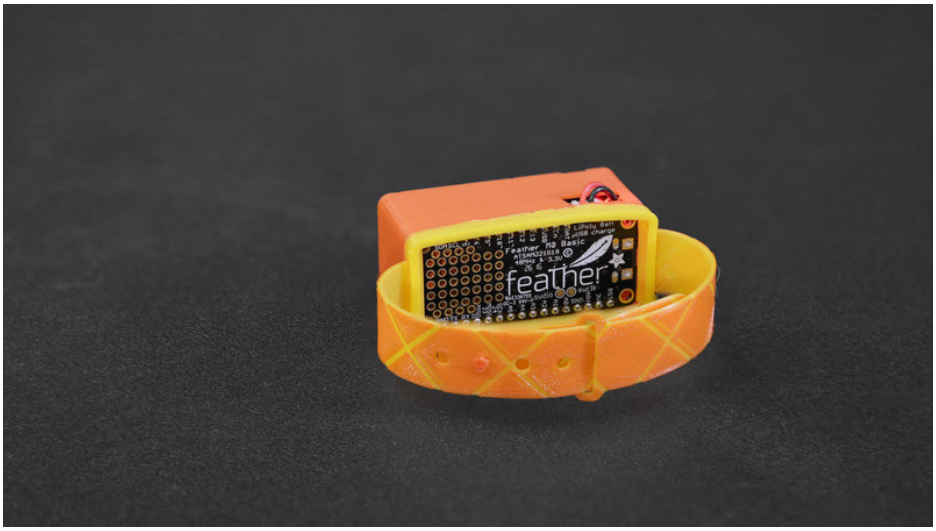
There is just enough room to insert the lipo battery between the Feather M4 and the RTC board. Carefully coil the wires so they fit between the boards.



Watch case

Align the stack of boards with the cutouts on the 3d printed case. Slip the board stack inside the case and attach the watch bands.

The end stoppers on the bands fit inside the slots on each side of the case.



Wear!

Putting the watch on with one hand can be tricky if you don't clean off the retraction left behind after 3d printing the case. Use a flush diagonal cutter to clean the adjustment holes and buckle tongue.

If the watch doesn't fit, you'll only need to adjust the watch bands by editing the length parameters available in the source files.