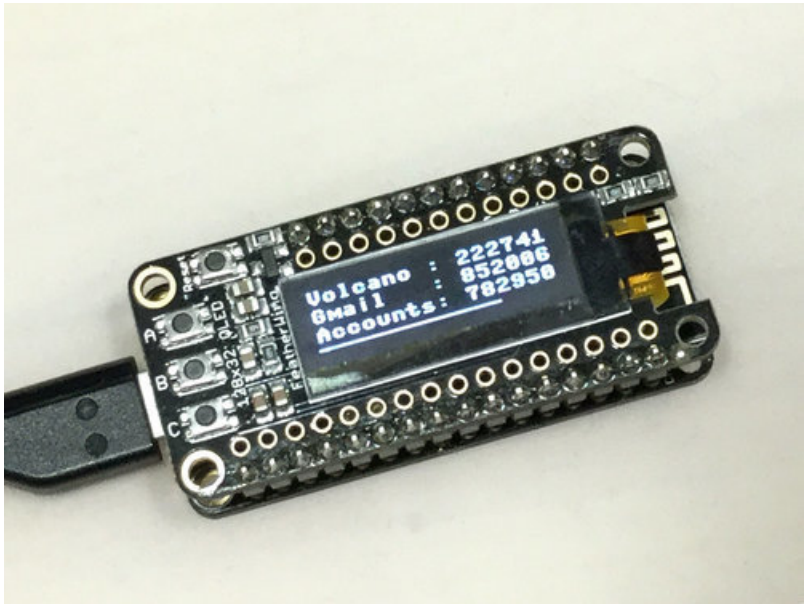
adafruit learning system

CircuitPython 2FA TOTP Authentication Friend

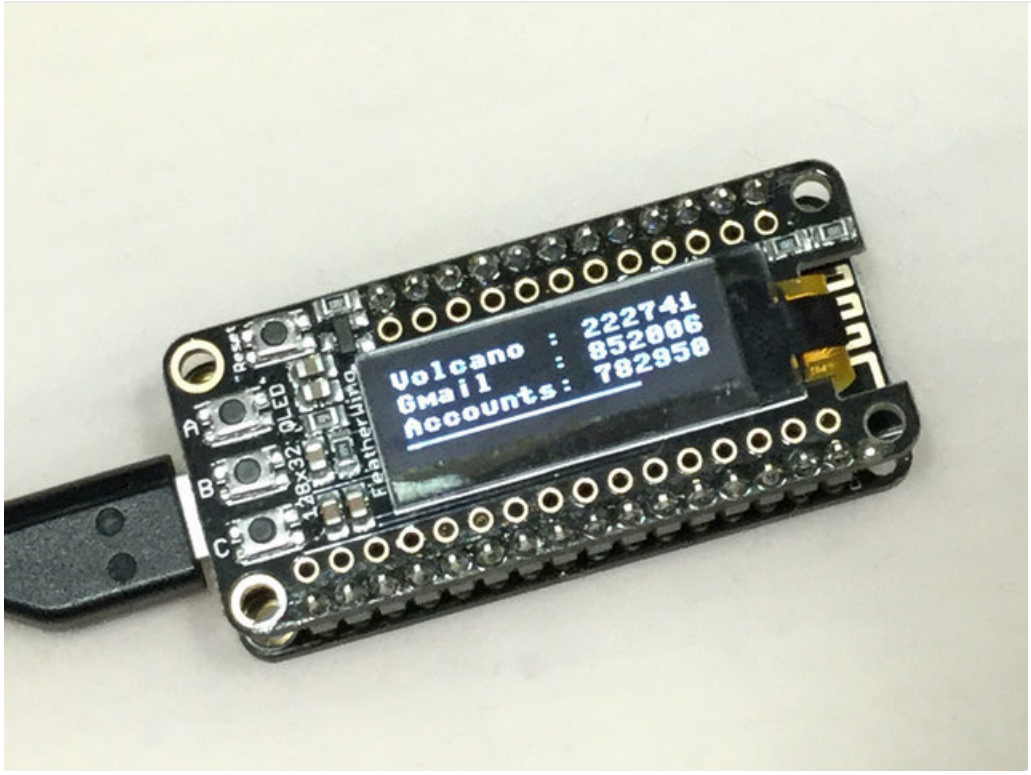
Created by lady ada



Last updated on 2018-08-22 04:05:03 PM UTC

Guide Contents

Introduction



What is TOTP?

Having 2 Factor Authentication on all your accounts is a good way to keep your data more secure. With 2FA logins, not only is a username and password needed, but also a one-time-use code. There's a few different ways to get that code, such as by email, phone or SMS. But my favorite way is to do it is via a 'Google Authenticator' time-based OTP (**o**ne **t**ime **p**assword), also known as a **TOTP**.

Using an app on your phone like Authy or Authenticator, you set up a secret given to you by the service, then every 30 seconds, a new code is generated for you. What's extra nice is that the Google Authenticator protocol is supported by just about *every* service and phone/tablet

So What's The Problem?

I don't own a phone! So I have to ask Mr. Ladyada for an authenticator code. Or I can use my tablet, but it's not always at my desk. And I don't want to buy a phone just for using 2FA!

A Solution!

Luckily for us, the Google Authenticator protocol is really simple - You just need to be able to know the current time, and run a SHA1 hash.

I decided to build a simple device that all it does is generate TOTP's for me, using CircuitPython - my favorite programming language! It uses a **Feather ESP8266** which has WiFi so it can connect to NTP to get the current time on startup, and a **Feather OLED** to display text nice and clearly.

Every time I need a new code, I just click the reset button and within 2 seconds I've got my 3 most common TOTP's on hand (yes its that fast!)

FAQ

THIS IS NOT A QUESTION MORE OF A COMMENT. YOU ARE PROGRAMMING THE TOTP SECRET INTO THE FLASH OF THE MICROCONTROLLER AND ITS NOT ENCRYPTED OR PROTECTED AT ALL ANYONE COULD BREAK INTO YOUR APARTMENT, GO TO YOUR BEDROOM, LOOK ON YOUR DESK, FIND THIS AND THEN CONNECT IT UP TO THEIR HACKER LAPTOP TO GRAB YOUR SECRET KEY THEN IF THEY HAD YOUR USERNAME AND PASSWORD THEY WOULD BE ABLE TO LOG IN AS YOU AND THIS IS REALLY INSECURE ITS SO IRRESPONSIBLE TO CONSIDER PUBLISHING A PROJECT LIKE THIS BY THE WAY DID YOU SEE THAT SNOWDEN APP? MAYBE YOU CAN RUN THAT ON A PHONE SO YOU CAN WATCH YOUR DESK REMOTELY AND MAKE SURE NOBODY BROKE IN TO STEAL YOUR FEATHER? OH WAIT YOU JUST SAID YOU DON'T HAVE A PHONE. OK I DONT KNOW WHAT MY QUESTION IS

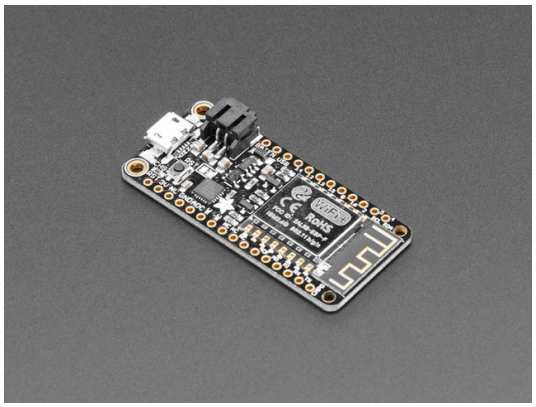
This project is probably not for you

Hardware

Parts Required

Easy! You only need two parts - a **Feather Huzzah** and an **OLED FeatherWing**

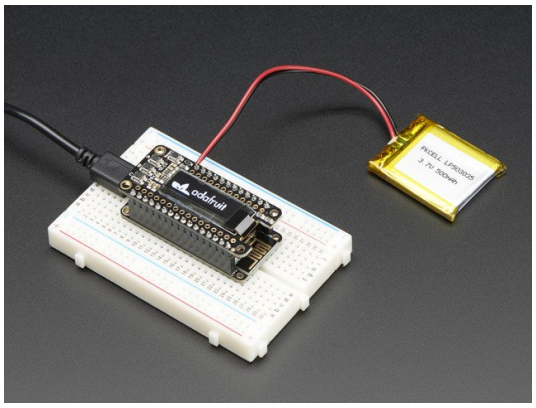
Your purchases from the Adafruit shop help support us writing up these awesome guides, libraries and CircuitPython development and are appreciated!!!



Adafruit Feather HUZZAH with ESP8266 - Loose Headers

$16.95
IN STOCK

ADD TO CART



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

$14.95
OUT OF STOCK

OUT OF STOCK

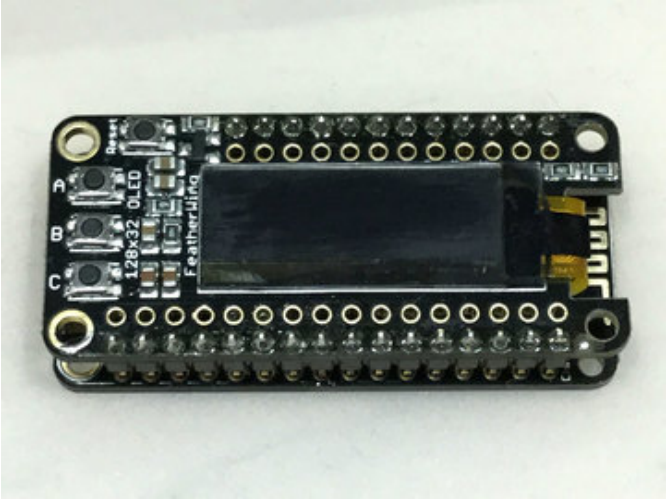
If you happen to be an AdaBox subscriber (what? you should be (https://adafru.it/tNC)!) You can find these parts in your **Adabox 003** kit!

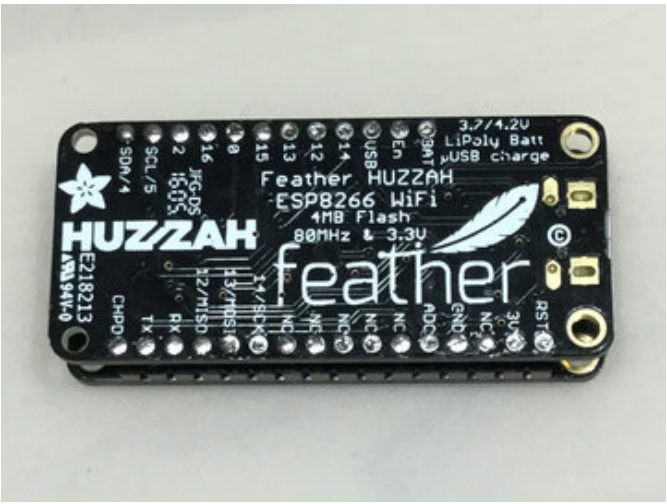
AdaBox003 – The World of IoT – Curated by Digi-Key

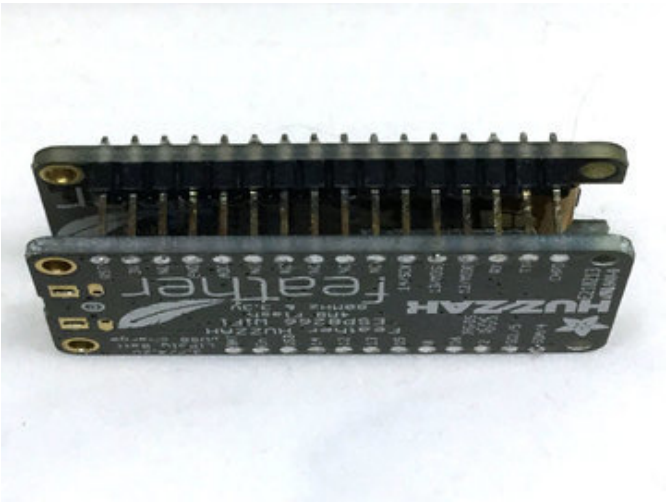
$79.95
IN STOCK

ADD TO CART

Assembly







While I started on a breadboard, I ended up making a cute little sandwich without socket headers at all by connecting the OLED directly to the Feather HUZZAH

If you press the OLED headers against a table you can use a little solder to wick into each hole and have a perfectly flat bottom side. That will keep it from scratching your desk!

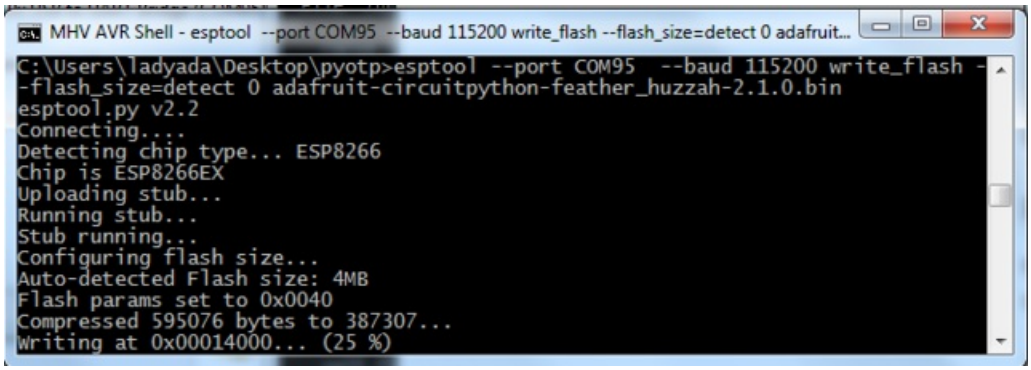
Software

Follow our handy getting-started guide on CircuitPython and especially the ESP8266 installation page/guide to learn how to install CircuitPython on your ESP8266 Feather (https://adafru.it/CiG)

https://adafru.it/cpy-welcome

https://adafru.it/cpy-welcome

Flash the latest version of CircuitPython (you'll need v 2.2 or higher) and continue to the next step!



Installing and using ampy

We're using the ESP8266 Feather which means it has lots of memory and Internet capability. We use the Internet part to get the current time. However, this Feather is not as easy to use as the SAMD series, as it *does not show up as a disk drive!*

You'll need to use ampy to install the circuitpython scripts!

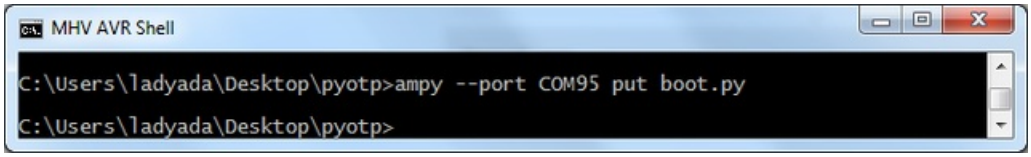
https://adafru.it/q9C

https://adafru.it/q9C

Install boot.py

Once you've gotten ampy working save the following to your computer as **boot.py** and upload it so that you don't have to turn off the os debug output via REPL anymore

```
import esp
esp.osdebug(None)
```

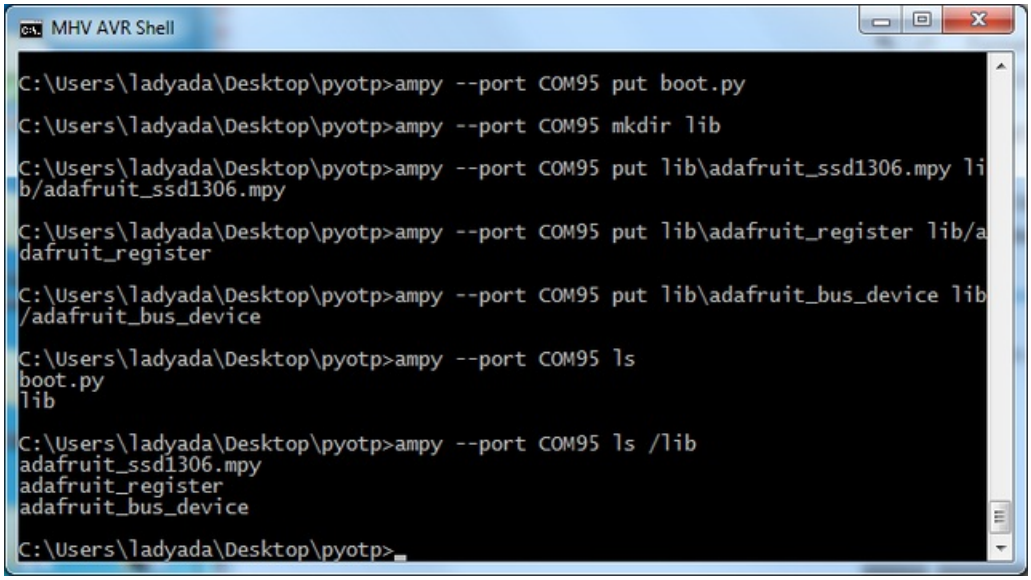


Install libraries

You'll need a bunch of libraries to get the OLED working. Use ampy to create a directory called **lib**

Then download the latest library bundle (https://adafru.it/ABU)

You'll need to upload **adafruit_ssd1306.mpy**, and the **adafruit_bus_device** and **adafruit_register** folders to the **lib** folder. Then check with ampy's `ls` command to verify all your files are in place!



Main Sketch

Now you can download the following script to your computer and save it as **main.py**

Don't upload it via ampy yet! The current file has fake tokens in it that need to be set!

```
import time

import adafruit_ssd1306
import bitbangio as io
import board
import network
import ntptime
import ubinascii
import uhashlib

# pylint: disable=broad-except

# https://github.com/pyotp/pyotp example
totp = [("Discord ", 'JBSWY3DPEHPK3PXP'),
        ("Gmail   ", 'abcdefghijklmnopqrstuvwxyz234567'),
        ("Accounts", 'asfdkwefoaiwejfa323nfjkl')]
ssid = 'my_wifi_ssid'
password = 'my_wifi_password'

TEST = False  # if you want to print out the tests the hashers
ALWAYS_ON = False  # Set to true if you never want to go to sleep!
ON_SECONDS = 60  # how long to stay on if not in always_on mode

i2c = io.I2C(board.SCL, board.SDA)
oled = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

# Gimme a welcome screen!
```

```
# Gimme a welcome screen!
oled.fill(0)
oled.text('CircuitPython', 0, 0)
oled.text('PyTOTP Pal!', 0, 10)
oled.text(' <3 adafruit <3 ', 0, 20)
oled.show()
time.sleep(0.25)

EPOCH_DELTA = 946684800  # seconds between year 2000 and year 1970
SECS_DAY = 86400

SHA1 = uhashlib.sha1

if TEST:
    print("========================================")
    print("SHA1 test: ", ubinascii.hexlify(SHA1(b'hello world').digest()))
    # should be 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed


# HMAC implementation, as hashlib/hmac wouldn't fit
# From https://en.wikipedia.org/wiki/Hash-based_message_authentication_code
def HMAC(k, m):
    SHA1_BLOCK_SIZE = 64
    KEY_BLOCK = k + (b'\0' * (SHA1_BLOCK_SIZE - len(k)))
    KEY_INNER = bytes((x ^ 0x36) for x in KEY_BLOCK)
    KEY_OUTER = bytes((x ^ 0x5C) for x in KEY_BLOCK)
    inner_message = KEY_INNER + m
    outer_message = KEY_OUTER + SHA1(inner_message).digest()
    return SHA1(outer_message)


if TEST:
    KEY = b'abcd'
    MESSAGE = b'efgh'
    print("========================================")
    print("HMAC test: ", ubinascii.hexlify(HMAC(KEY, MESSAGE).digest()))
    # should be e5dbcf9263188f9fce90df572afeb39b66b27198


# Base32 decoder, since base64 lib wouldnt fit


def base32_decode(encoded):
    missing_padding = len(encoded) % 8
    if missing_padding != 0:
        encoded += '=' * (8 - missing_padding)
    encoded = encoded.upper()
    chunks = [encoded[i:i + 8] for i in range(0, len(encoded), 8)]

    out = []
    for chunk in chunks:
        bits = 0
        bitbuff = 0
        for c in chunk:
            if 'A' <= c <= 'Z':
                n = ord(c) - ord('A')
            elif '2' <= c <= '7':
                n = ord(c) - ord('2') + 26
            elif n == '=':
                continue
```

```
            else:
                raise ValueError("Not base32")
            # 5 bits per 8 chars of base32
            bits += 5
            # shift down and add the current value
            bitbuff <<= 5
            bitbuff |= n
            # great! we have enough to extract a byte
            if bits >= 8:
                bits -= 8
                byte = bitbuff >> bits  # grab top 8 bits
                bitbuff &= ~(0xFF << bits)  # and clear them
                out.append(byte)  # store what we got
    return out


if TEST:
    print("========================================")
    print("Base32 test: ", bytes(base32_decode("IFSGCZTSOVUXIIJB")))
    # should be "Adafruit!!"


# Turns an integer into a padded-with-0x0 bytestr


def int_to_bytestring(i, padding=8):
    result = []
    while i != 0:
        result.insert(0, i & 0xFF)
        i >>= 8
    result = [0] * (padding - len(result)) + result
    return bytes(result)


# HMAC -> OTP generator, pretty much same as
# https://github.com/pyotp/pyotp/blob/master/src/pyotp/otp.py


def generate_otp(int_input, secret_key, digits=6):
    if int_input < 0:
        raise ValueError('input must be positive integer')
    hmac_hash = bytearray(
        HMAC(bytes(base32_decode(secret_key)),
             int_to_bytestring(int_input)).digest()
    )
    offset = hmac_hash[-1] & 0xf
    code = ((hmac_hash[offset] & 0x7f) << 24 |
            (hmac_hash[offset + 1] & 0xff) << 16 |
            (hmac_hash[offset + 2] & 0xff) << 8 |
            (hmac_hash[offset + 3] & 0xff))
    str_code = str(code % 10 ** digits)
    while len(str_code) < digits:
        str_code = '0' + str_code

    return str_code


print("========================================")

# Set up networking
```

```
# Set up networking
sta_if = network.WLAN(network.STA_IF)

oled.fill(0)
oled.text('Connecting to', 0, 0)
oled.text(ssid, 0, 10)
oled.show()

if not sta_if.isconnected():
    print("Connecting to SSID", ssid)
    sta_if.active(True)
    sta_if.connect(ssid, password)
    while not sta_if.isconnected():
        pass
print("Connected! IP = ", sta_if.ifconfig()[0])

# Done! Let them know we made it
oled.text("IP: " + sta_if.ifconfig()[0], 0, 20)
oled.show()
time.sleep(0.25)

# Get the latest time from NTP
t = None
while not t:
    try:
        t = ntptime.time()
    except Exception:
        pass
    time.sleep(0.1)

# NTP time is seconds-since-2000
print("NTP time: ", t)

# But we need Unix time, which is seconds-since-1970
t += EPOCH_DELTA
print("Unix time: ", t)

# Instead of using RTC which means converting back and forth
# we'll just keep track of seconds-elapsed-since-NTP-call
mono_time = int(time.monotonic())
print("Monotonic time", mono_time)

countdown = ON_SECONDS  # how long to stay on if not in always_on mode
while ALWAYS_ON or (countdown > 0):
    # Calculate current time based on NTP + monotonic
    unix_time = t - mono_time + int(time.monotonic())
    print("Unix time: ", unix_time)

    # Clear the screen
    oled.fill(0)
    y = 0
    # We can do up to 3 per line on the Feather OLED
    for name, secret in totp:
        otp = generate_otp(unix_time // 30, secret)
        print(name + " OTP output: ", otp)  # serial debugging output
        oled.text(name + ": " + str(otp), 0, y)  # display name & OTP on OLED
        y += 10  # Go to next line on OLED
    # Display a little bar that 'counts down' how many seconds you have left
    oled.framebuf.line(0, 31, 128 - (unix_time % 30) * 4, 31, True)
    oled.show()
```

```
    # We'll update every 1/4 second, we can hash very fast so its no biggie!
    countdown -= 0.25
    time.sleep(0.25)

# All these hashes will be lost in time(), like tears in rain. Time to die
oled.fill(0)
oled.show()
```

Set Up Networking

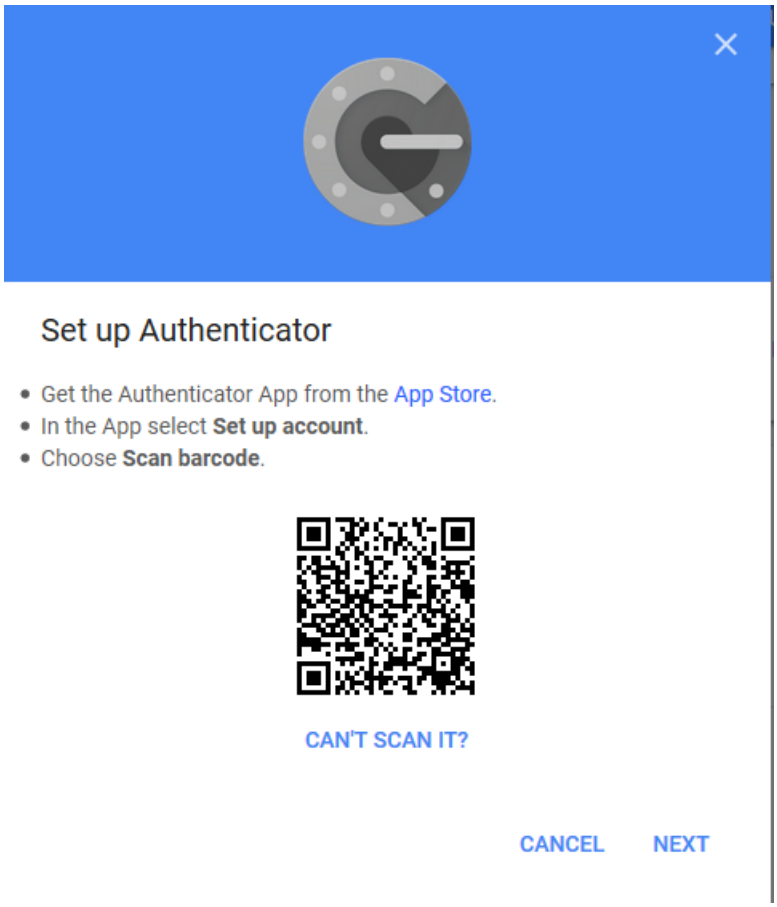
Before uploading, change these two lines to your network SSID and password

```
ssid     = 'my_wifi_ssid'
password = 'my_wifi_password'
```

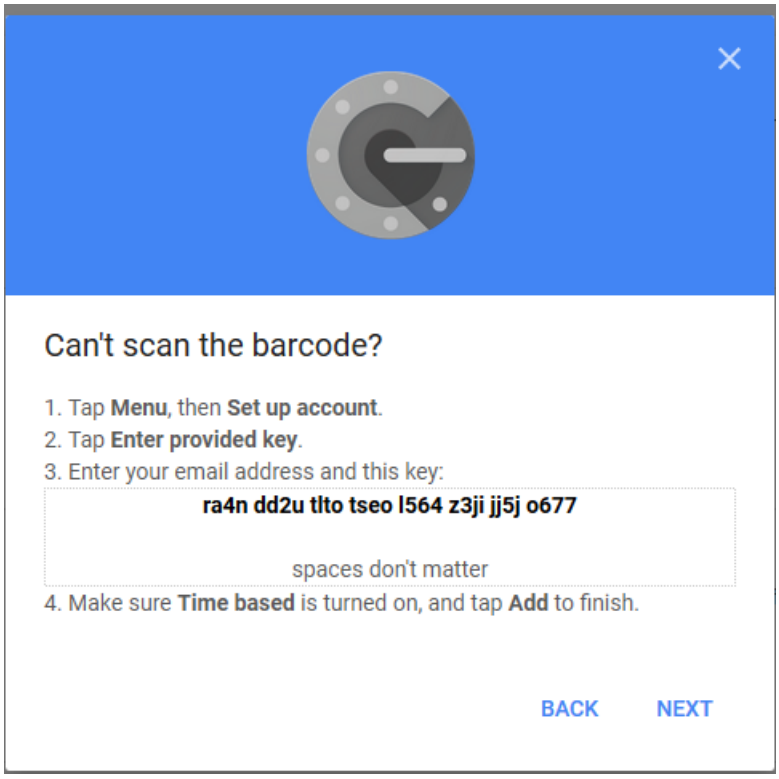
Set Up Tokens

You'll also need to get 2 factor "authenticator tokens/secrets". Each site is a little different about how it does this.

For example, when you set up GMail for 2FA it will show you a QR code like this:



Which is great for phones. For us, we need the base32-encoded token. Click the **Can't Scan It?** link or otherwise request the text token. You'll get a page like this



That string of letters and numbers may be uppercase or lower case, it may also be 16 digits or 24 or 32 or some other qty. It doesn't matter! Grab that string, and remove the spaces so its one long string like `"ra4ndd2utltotseol564z3jijj5jo677"` Note that the number 0 and number 1 never appear so anything that looks like an `O`, `l` or an `I` is a letter.

Now edit this section of the code, you can display up to 3 accounts on a Feather OLED. If you pad the name with spaces the numbers will be right-justified but its not important, I'm just picky

```
totp = [("Discord ", 'JBSWY3DPEHPK3PXP'), # https://github.com/pyotp/pyotp exmple
        ("Gmail   ", 'abcdefghijklmnopqrstuvwxyz234567'),
("Accounts", 'asfdkwefoaiwejfa323nfjkl')]
```

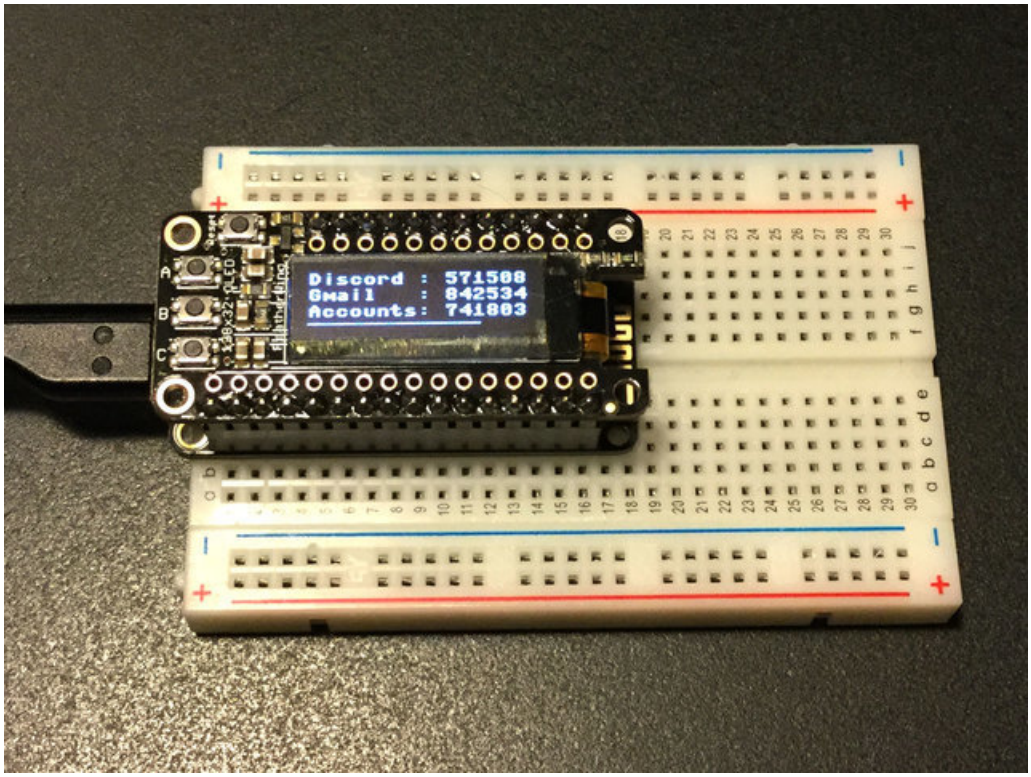
If you want to test the setup first, you can keep the Discord entry which is the "PyOTP" example token. Then scan this with your phone in Authy or Google Authenticator

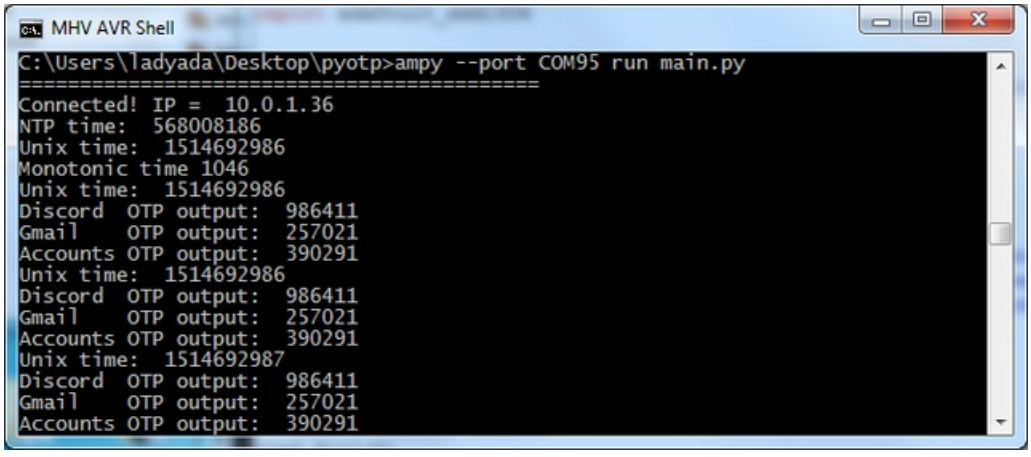
Test It Out!

OK once you've set everything up lets test!

Run the program directly on the Feather with OLED attached `using ampy --port portname run main.py`

You'll see it connect to your local network, get the time via NTP, then calculate and display OTP codes both on the OLED and on the serial port (you'll need to wait till the program is done to see the serial output)





Check against your phone to make sure the codes are correct. Once you're satisfied, tweak the two lines to change the behavior

```
ALWAYS_ON = False  # Set to true if you never want to go to sleep!
ON_SECONDS = 60 # how long to stay on if not in always_on mode
```

Then finalize by uploading main.py with ampy's `put` command