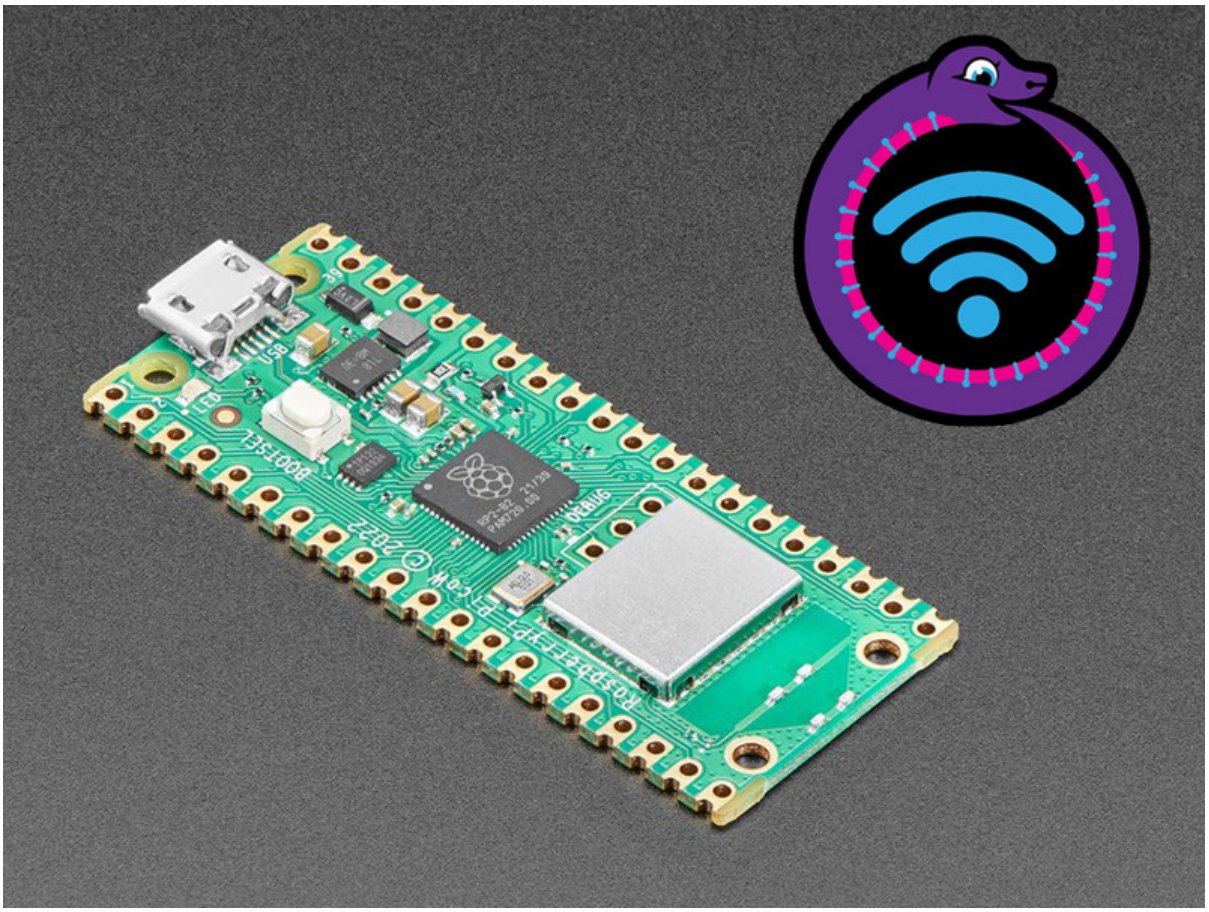
Quick-Start the Pico W WiFi with CircuitPython

Created by Liz Clark

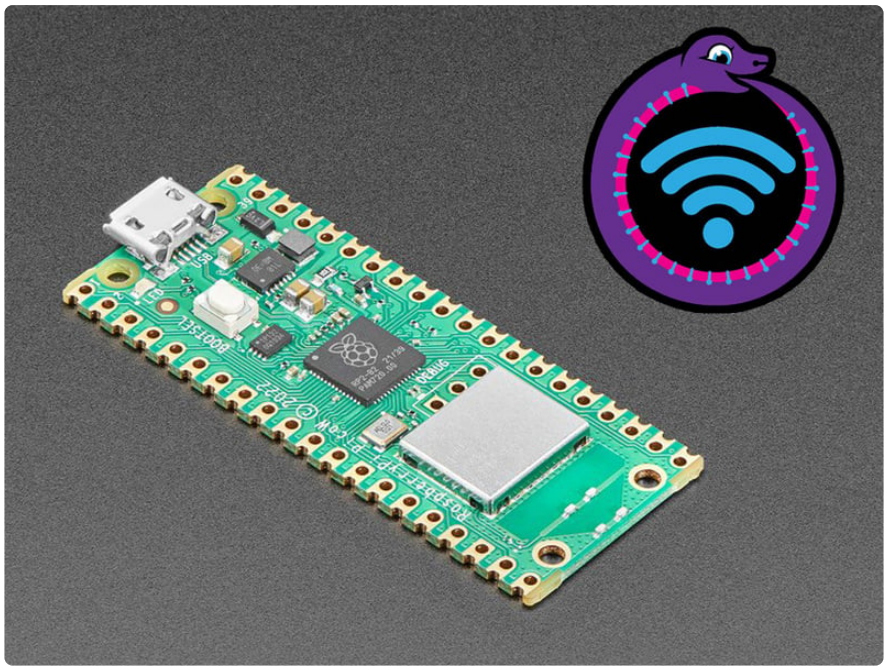


https://learn.adafruit.com/pico-w-wifi-with-circuitpython

Last updated on 2024-03-08 04:09:07 PM EST

Table of Contents

Overview

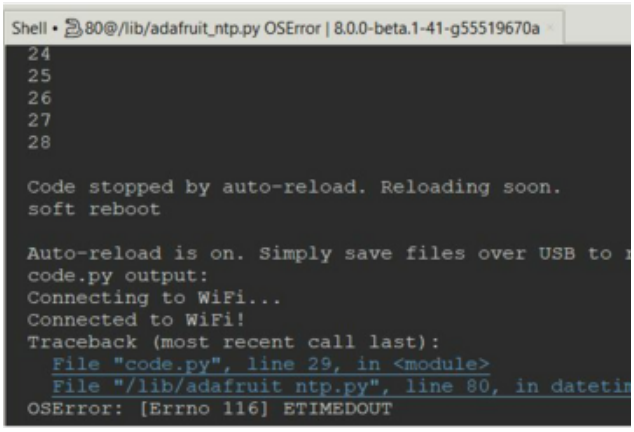


The Raspberry Pi Foundation changed single-board computing when they released the Raspberry Pi computer (https://adafru.it/Qa1), now they're ready to do the same for microcontrollers with the release of the brand new **Raspberry Pi Pico W**. This low-cost microcontroller board features their powerful new chip, the **RP2040**, and all the fixin's to get started with IoT embedded electronics projects at a stress-free price.

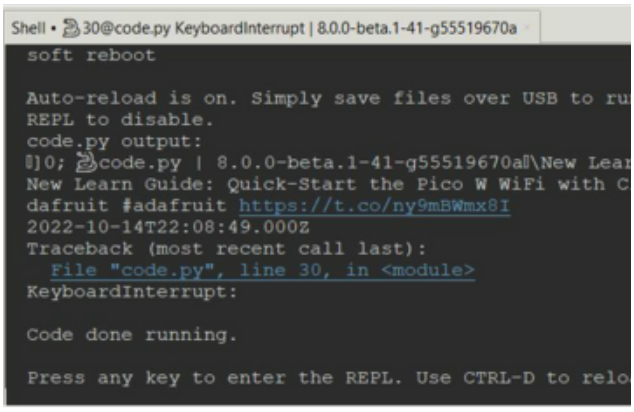
Raspberry Pi Pico W brings WiFi to the Pico platform while retaining complete pin compatibility with its older sibling, and now as of CircuitPython 8.0.0-beta.2 (https://adafru.it/11wd), there is CircuitPython WiFi support for the Pico W! This guide includes examples for testing your WiFi connection, using requests to pull JSON feeds, ping API's and log sensor data for IoT projects; all using CircuitPython!

Status Bar

As of CircuitPython 8.0.0 (https://adafru.it/11fn), if you have a smart terminal program like Thonny (https://adafru.it/Qb6), tio (https://adafru.it/11xF) or Screen (https://adafru.it/wD0), you will see the status of your CircuitPython board in the header bar of the terminal.



If you have an error while running your code, the status bar will tell you what line of code was running when the error occurred, as well as the type of error.



Additionally, if you end the program from the shell with a `KeyboardInterrupt`, that information will be displayed in the status bar.

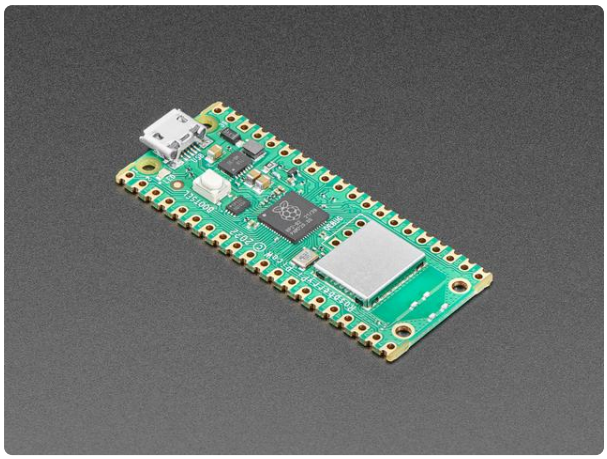
About the Code Examples

WiFi and networking are complicated and have many failure states. Rather than having extensive code to detect and recover from each specific kind of failure, the examples here use `microcontroller.reset()` which fully re-initialize both the microcontroller and the WiFi co-processor and start the code again with a clean slate.

The general pattern is:

```
try:
    your_application_here()
except Exception as e:
    print("Error:\n", str(e))
    print("Resetting microcontroller in 10 seconds")
    time.sleep(10)
    microcontroller.reset()
```

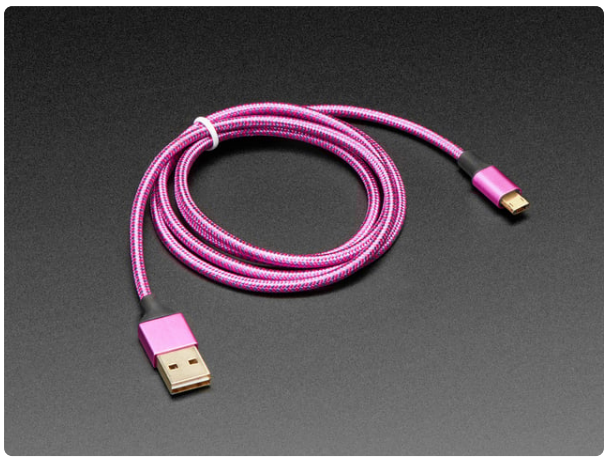
Parts



Raspberry Pi Pico W

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...

https://www.adafruit.com/product/5526



Fully Reversible Pink/Purple USB A to micro B Cable - 1m long

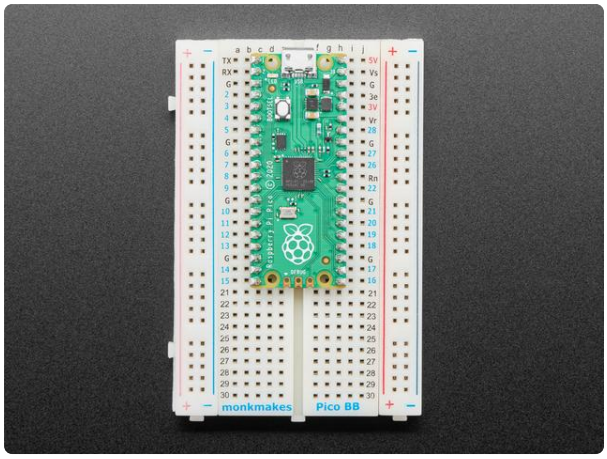
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...

https://www.adafruit.com/product/4111

Adafruit AHT20 - Temperature & Humidity Sensor Breakout Board

The AHT20 is a nice but inexpensive temperature and humidity sensor from the same folks that brought us the DHT22. You can take...

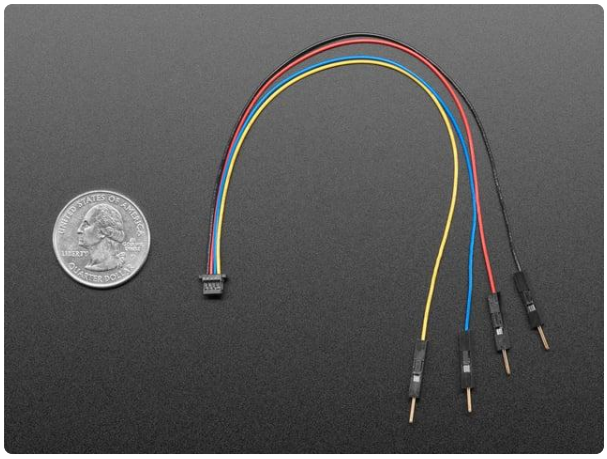
https://www.adafruit.com/product/4566



Solderless Breadboard for Raspberry Pi Pico by Monk Makes

It can be tricky to work out which pin is which when the Raspberry Pi Pico is attached to solderless...

https://www.adafruit.com/product/5422



STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable

This 4-wire cable is a little over 150mm / 6" long and fitted with JST-SH female 4-pin connectors on one end and premium Dupont male headers on the other. Compared with the...

https://www.adafruit.com/product/4209

Installing CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) 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.

CircuitPython Quickstart

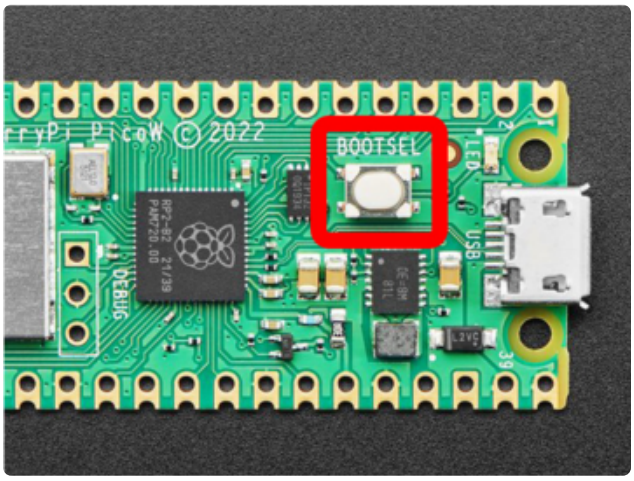
Follow this step-by-step to quickly get CircuitPython working on your board.

Download the latest version of CircuitPython for the Raspberry Pi Pico W from circuitpython.org

https://adafru.it/11xd

Click the link above and download the latest UF2 file.

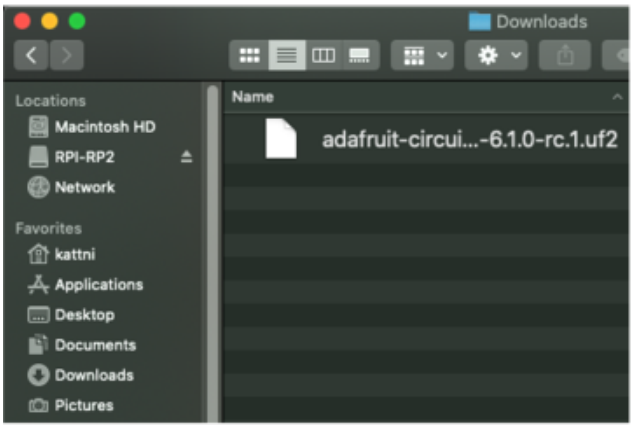
Download and save it to your desktop (or wherever is handy).



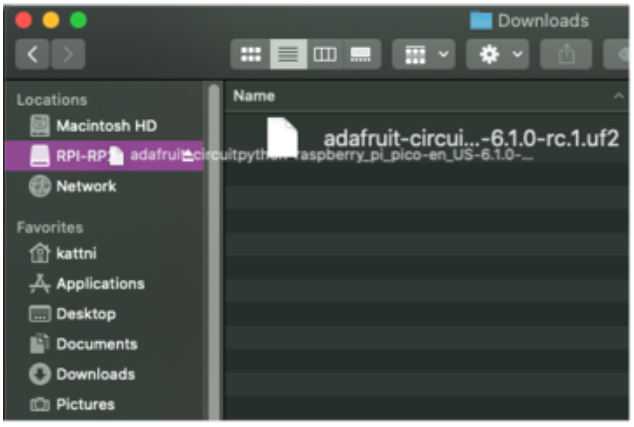
Start with your Pico W unplugged from USB. Hold down the **BOOTSEL** button, and while continuing to hold it (don't let go!), plug the Pico W into USB. **Continue to hold the BOOTSEL button until the RPI-RP2 drive appears!**

If the drive does not appear, unplug your Pico W and go through the above process again.

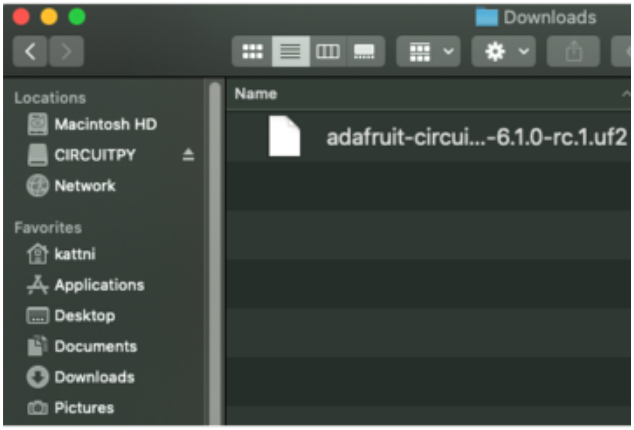
A lot of people end up using charge-only USB cables and it is very frustrating! **So make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **RPI-RP2**.



Drag the **adafruit_circuitpython_etc.uf2** file to **RPI-RP2.**



The **RPI-RP2** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Flash Resetting UF2

If your Pico W ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try installing this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After nuking, re-install CircuitPython

flash_nuke.uf2

Create Your settings.toml File

CircuitPython works with WiFi-capable boards to enable you to make projects that have network connectivity. This means working with various passwords and API keys. As of CircuitPython 8 (https://adafru.it/Em8), there is support for a **settings.toml** file. This is a file that is stored on your **CIRCUITPY** drive, that contains all of your secret network information, such as your SSID, SSID password and any API keys for IoT services. It is designed to separate your sensitive information from your **code.py** file so you are able to share your code without sharing your credentials.

CircuitPython previously used a **secrets.py** file for this purpose. The **settings.toml** file is quite similar.

Your settings.toml file should be stored in the main directory of your CIRCUITPY drive. It should not be in a folder.

CircuitPython **settings.toml** File

This section will provide a couple of examples of what your **settings.toml** file should look like, specifically for CircuitPython WiFi projects in general.

The most minimal **settings.toml** file must contain your WiFi SSID and password, as that is the minimum required to connect to WiFi. Copy this example, paste it into your **settings.toml**, and update:

- `your_wifi_ssid`
- `your_wifi_password`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
```

Many CircuitPython network-connected projects on the Adafruit Learn System involve using Adafruit IO. For these projects, you must also include your Adafruit IO username and key. Copy the following example, paste it into your settings.toml file, and update:

- `your_wifi_ssid`
- `your_wifi_password`
- `your_aio_username`

- `your_aio_key`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
ADAFRUIT_AIO_USERNAME = "your_aio_username"
ADAFRUIT_AIO_KEY = "your_aio_key"
```

Some projects use different variable names for the entries in the **settings.toml** file. For example, a project might use `ADAFRUIT_AIO_ID` in the place of `ADAFRUIT_AIO_USERNAME`. **If you run into connectivity issues, one of the first things to check is that the names in the settings.toml file match the names in the code.**

Not every project uses the same variable name for each entry in the settings.toml file! Always verify it matches the code.

settings.toml File Tips

Here is an example **settings.toml** file.

```
# Comments are supported
CIRCUITPY_WIFI_SSID = "guest wifi"
CIRCUITPY_WIFI_PASSWORD = "guessable"
CIRCUITPY_WEB_API_PORT = 80
CIRCUITPY_WEB_API_PASSWORD = "passw0rd"
test_variable = "this is a test"
thumbs_up = "\U0001f44d"
```

In a **settings.toml** file, it's important to keep these factors in mind:

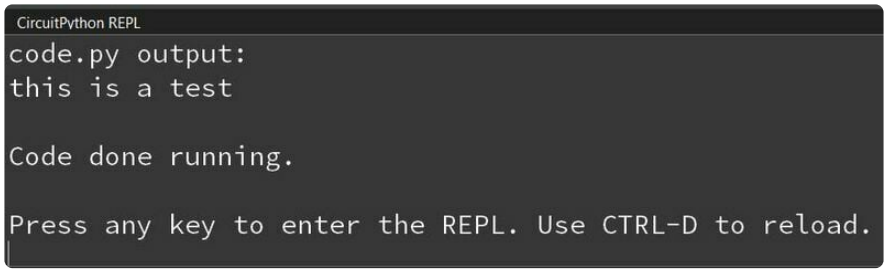
- Strings are wrapped in double quotes; ex: `"your-string-here"`
- Integers are **not** quoted and may be written in decimal with optional sign (`+1`, `-1`, `1000`) or hexadecimal (`0xabcd`).
    - Floats, octal (`0o567`) and binary (`0b11011`) are not supported.
- Use `\u` escapes for weird characters, `\x` and `\ooo` escapes are not available in **.toml** files
    - Example: `\U0001f44d` for (thumbs up emoji) and `\u20ac` for € (EUR sign)
- Unicode emoji, and non-ASCII characters, stand for themselves as long as you're careful to save in "UTF-8 without BOM" format

When your **settings.toml** file is ready, you can save it in your text editor with the **.toml** extension.

Accessing Your **settings.toml** Information in **code.py**

In your **code.py** file, you'll need to `import` the `os` library to access the **settings.toml** file. Your settings are accessed with the `os.getenv()` function. You'll pass your settings entry to the function to import it into the **code.py** file.

```
import os

print(os.getenv("test_variable"))
```



In the upcoming CircuitPython WiFi examples, you'll see how the **settings.toml** file is used for connecting to your SSID and accessing your API keys.

Environment Variables Docs

Environment Variables Docs (https://adafru.it/11wE)

Pico W Basic WiFi Test

In this example, you'll test your Pico W WiFi connection by connecting to your SSID, printing your MAC address and IP address to the REPL and then pinging Google.

Add Your **settings.toml** File

Remember to add your **settings.toml** file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD` in the file. Note that Create Your settings.toml File (https://adafru.it/18f9) uses different names for these values.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"
```

Code the Basic WiFi Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import os
import ipaddress
import wifi
import socketpool

print()
print("Connecting to WiFi")

#  connect to your SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

print("Connected to WiFi")

pool = socketpool.SocketPool(wifi.radio)

#  prints MAC address to REPL
print("My MAC addr:", [hex(i) for i in wifi.radio.mac_address])

#  prints IP address to REPL
print("My IP address is", wifi.radio.ipv4_address)

#  pings Google
ipv4 = ipaddress.ip_address("8.8.4.4")
print("Ping google.com: %f ms" % (wifi.radio.ping(ipv4)*1000))
```

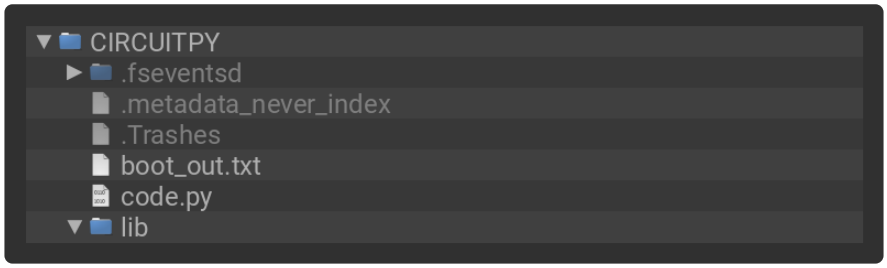
Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

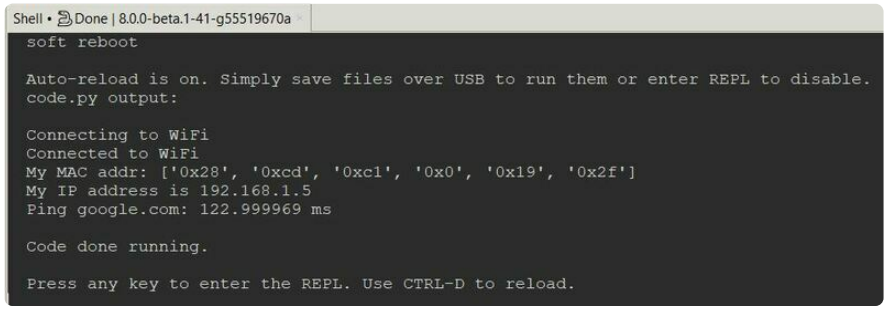
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **code.py** file.

No libraries from the library bundle are used in this example, so the lib folder is empty. All of the libraries are a part of the core.



Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



Having Problems?

If you get the message "TypeError: object with buffer protocol required", likely CircuitPython cannot find the file **settings.toml** on your **CIRCUITPY** drive. Check to be sure the file name is correct and that it contains the `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD` values as noted in the "Create Your settings.toml File" section.

If you get "Connecting to WiFi" but don't get to "Connected to WiFi", check to make sure the `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD` values in **settings.toml** are valid for your 2.4GHz network. There is no retry in this code, you might have to run the program a second time to get a response.

How the Pico W Basic WiFi Test Works

In the basic WiFi test, the Pico W connects to your SSID by importing your SSID and SSID password from the **settings.toml** file.

```
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))
```

Then, your MAC address and IP address are printed to the REPL.

```
#  prints MAC address to REPL
print("My MAC addr:", [hex(i) for i in wifi.radio.mac_address])

#  prints IP address to REPL
print("My IP address is", wifi.radio.ipv4_address)
```

Finally, google.com is pinged. The amount of time it takes to ping is printed to the REPL and the code stops running.

```
#  pings Google
ipv4 = ipaddress.ip_address("8.8.4.4")
print("Ping google.com: %f ms" % (wifi.radio.ping(ipv4)*1000))
```

By successfully running this WiFi test code, you can confirm that your Pico W is connecting to WiFi with CircuitPython successfully and you can move on to more advanced projects.

Pico W Requests Test (Adafruit Quotes)

In this example, you'll use the adafruit_requests (https://adafru.it/11xf) library to fetch a text response from the Adafruit Quotes PHP script (https://adafru.it/11xA). Each time the URL is requested, it returns a new quote from a large database of quotes.

Code the Requests Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

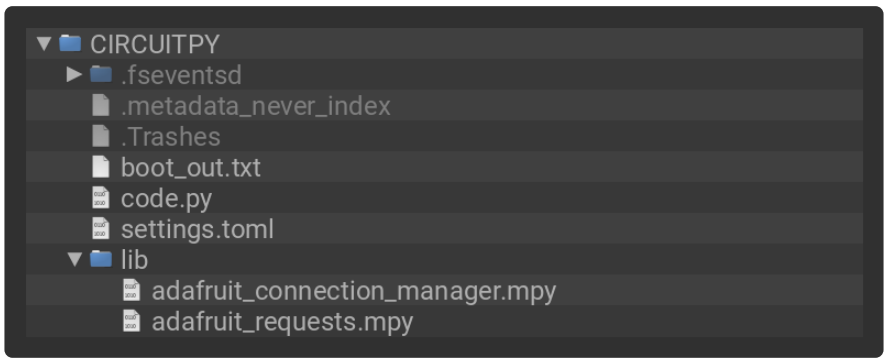
```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import os
import time
import ssl
import wifi
import socketpool
import microcontroller
import adafruit_requests

#  adafruit quotes URL
quotes_url = "https://www.adafruit.com/api/quotes.php"

#  connect to SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

while True:
    try:
        #  pings adafruit quotes
        print("Fetching text from %s" % quotes_url)
        #  gets the quote from adafruit quotes
        response = requests.get(quotes_url)
        print("-" * 40)
        #  prints the response to the REPL
        print("Text Response: ", response.text)
        print("-" * 40)
        response.close()
        #  delays for 1 minute
        time.sleep(60)
    # pylint: disable=broad-except
    except Exception as e:
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
```

Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.



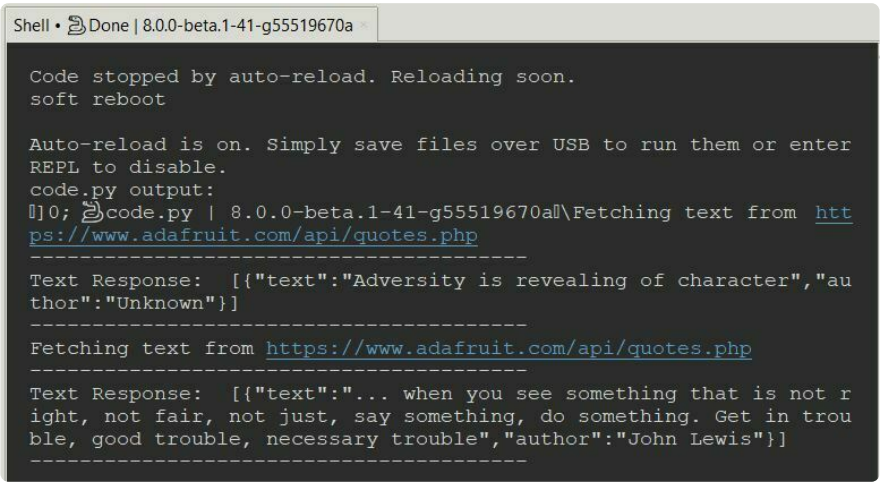
Add Your **settings.toml** File

Remember to add your **settings.toml** file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `WIFI_SSID` and `WIFI_PASSWORD` in the file.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"
```

Run **code.py**

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



Every 60 seconds, a `request` will be made to the Adafruit quotes URL. Then, the response will be printed to the REPL. The response includes the quote text and the author of the quote.

Pico W JSON Feed (OpenWeatherMap)

In this example, you'll use the OpenWeatherMap API (https://adafru.it/Pen) to receive and parse a JSON feed of your location's weather.

You'll need to register for an account with OpenWeatherMap and get your API key. Go to this link (https://adafru.it/EeH) and register for a free account. Once registered, you'll get an email containing your API key. This key will be added to your **settings.toml** file as the 'openweather_token'.

Code the JSON Feed Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import os
import ssl
import wifi
import socketpool
import microcontroller
import adafruit_requests

wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

# Use cityname, country code where countrycode is ISO3166 format.
# E.g. "New York, US" or "London, GB"
location = "Manhattan, US"

# openweathermap URL, brings in your location & your token
url = "http://api.openweathermap.org/data/2.5/weather?q="+location
url += "&appid="+os.getenv('openweather_token')

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

while True:
    try:
        #  pings openweather
        response = requests.get(url)
        #  packs the response into a JSON
        response_as_json = response.json()
        print()
        #  prints the entire JSON
        print(response_as_json)
        #  gets location name
        place = response_as_json['name']
```

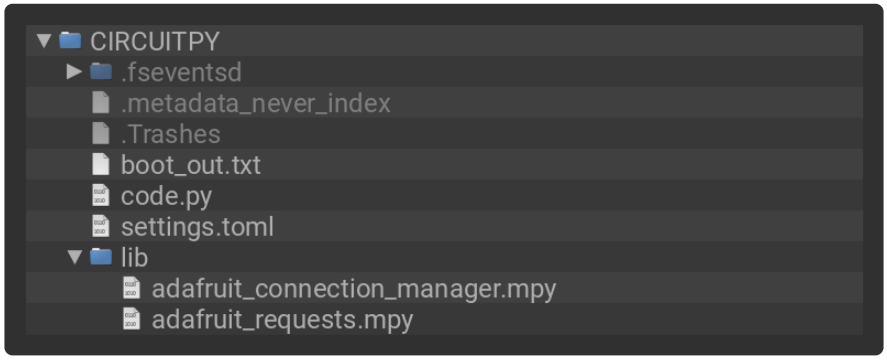
```
        #  gets weather type (clouds, sun, etc)
        weather = response_as_json['weather'][0]['main']
        #  gets humidity %
        humidity = response_as_json['main']['humidity']
        #  gets air pressure in hPa
        pressure = response_as_json['main']['pressure']
        #  gets temp in kelvin
        temperature = response_as_json['main']['temp']
        #  converts temp from kelvin to F
        converted_temp = (temperature - 273.15) * 9/5 + 32
        #  converts temp from kelvin to C
        #  converted_temp = temperature - 273.15

        #  prints out weather data formatted nicely as pulled from JSON
        print()
        print("The current weather in %s is:" % place)
        print(weather)
        print("%s°F" % converted_temp)
        print("%s%% Humidity" % humidity)
        print("%s hPa" % pressure)
        #  delay for 5 minutes
        time.sleep(300)
    # pylint: disable=broad-except
    except Exception as e:
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
```

Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.



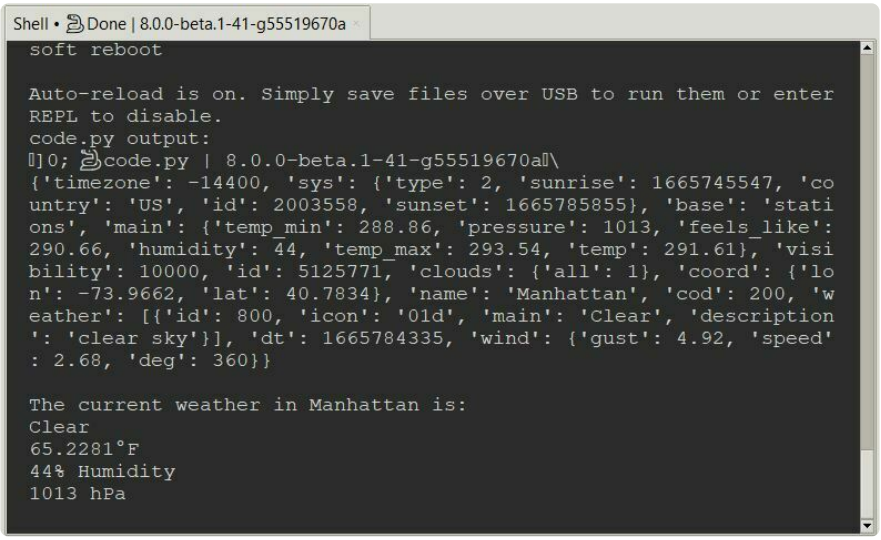
Add Your **settings.toml** File

Remember to add your settings.toml file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `WIFI_SSID`, `WIFI_PASSWORD` and `openweather_token` in the file.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"

openweather_token = "your-openweather-token-here"
```

Run **code.py**

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



The code makes a `request` to the OpenWeatherMap API and receives a JSON feed of your location's weather. That JSON output is printed to the REPL. Then, the location, weather, humidity, pressure and temperature converted to Fahrenheit is printed to the REPL. This call is repeated every five minutes.

Pico W Twitter Feed

The Twitter API is no longer free to use and this example may no longer be useful

In this example, you'll use the Twitter API (https://adafru.it/NFH) to create a query URL to request tweets that fit your search criteria. To get started, the query in the example looks for tweets from Adafruit that have the text "NEW GUIDE". With this query, you'll always know when a new Learn Guide has been tweeted about.

Code the Twitter Feed Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import os
import ssl
import wifi
import socketpool
import microcontroller
import adafruit_requests

#  query URL for tweets. looking for tweets from Adafruit that have the text "NEW
GUIDE"
#  disabling line-too-long because queries for tweet_query & TIME_URL cannot have
line breaks
#  pylint: disable=line-too-long
tweet_query = 'https://api.twitter.com/2/tweets/search/recent?query=NEW GUIDE
from:adafruit&tweet.fields=created_at'

headers = {'Authorization': 'Bearer ' + os.getenv('bearer_token')}

wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

last_id = 0 #  checks last tweet's ID
check = 0 #  time.monotonic() holder

while True:
    try:
        if (check + 30) < time.monotonic():
            #  updates current time
            #  get tweets from rpilocator containing in stock at adafruit
            twitter_response = requests.get(url=tweet_query, headers=headers)
            #  gets data portion of json
            twitter_json = twitter_response.json()
            twitter_json = twitter_json['data']
            #  to see the entire json feed, uncomment 'print(twitter_json)'
            #  print(twitter_json)
            #  tweet ID number
            tweet_id = twitter_json[0]['id']
            #  tweet text
            tweet = twitter_json[0]['text']
            #  timestamp
            timestamp = twitter_json[0]['created_at']
            if last_id != tweet_id:
                print("New Learn Guide!")
                print(tweet)
                print(timestamp)
                last_id = tweet_id
            check = time.monotonic()
    # pylint: disable=broad-except
```

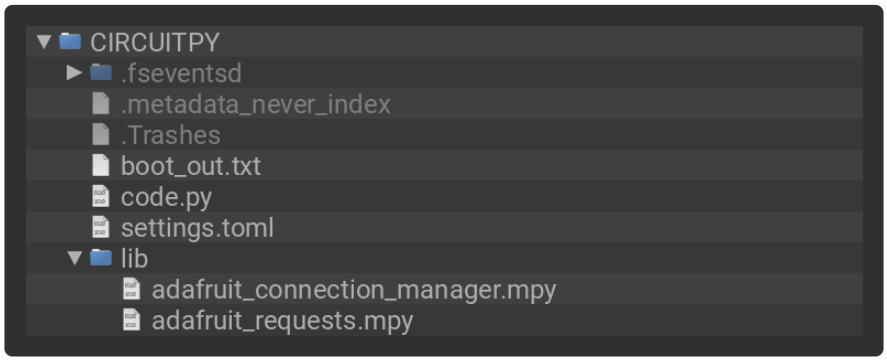
```
    #  any errors, reset Pico W
    except Exception as e:
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
```

Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.



Add Your **settings.toml** File

Remember to add your **settings.toml** file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `WIFI_SSID`, `WIFI_PASSWORD` and Twitter `bearer_token` in the file.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"

bearer_token = "your-twitter-bearer-token-here"
```

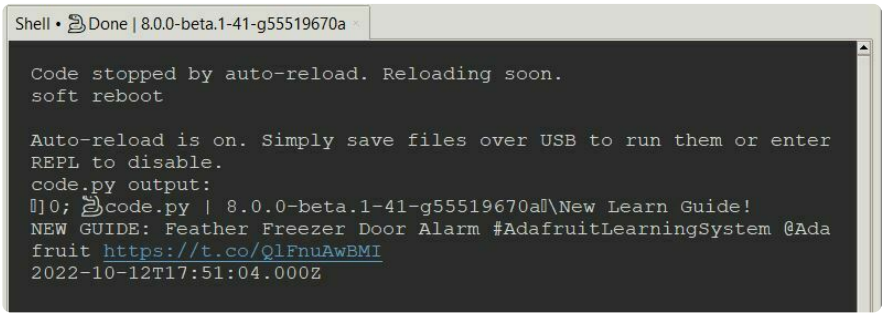
For the Twitter API, you'll need to request a developer account and create a project on the API. Follow along with this guide page (https://adafru.it/11xB) to see how to do this and retrieve your bearer authentication token.

How to Setup the Twitter API

https://adafru.it/11xB

Run **code.py**

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



Every 30 seconds, a request is made to the Twitter API using your query URL. If a new tweet is available that matches the query, its text and timestamp are printed to the REPL. The tweet's tweet ID is used to determine if a new tweet is available.

Pico W with Adafruit IO

In this example, you'll use the AHT20 temperature and humidity sensor (https://adafru.it/11xC) to log temperature and humidity data to Adafruit IO.

Prerequisite Guides

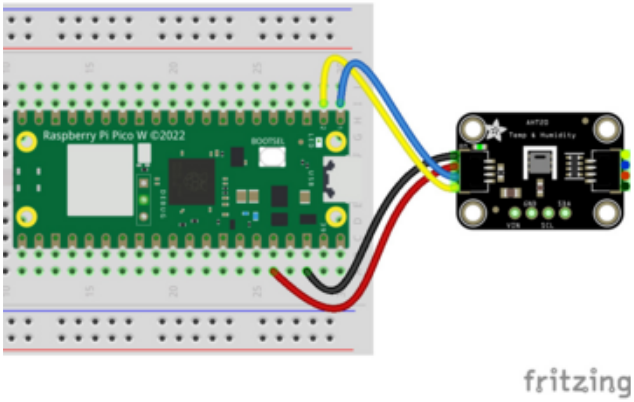
Adafruit AHT20 Temperature & Humidity Sensor

https://adafru.it/11xC

Welcome to Adafruit IO

https://adafru.it/BRB

AHT20 Wiring



Board 3V to **sensor VIN**
Board GND to **sensor GND**
Board GP1 to **sensor SCL**
Board GP0 to **sensor SDA**

Code the Adafruit IO Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import os
import time
import ssl
import wifi
import socketpool
import microcontroller
import board
import busio
import adafruit_requests
import adafruit_ahtx0
from adafruit_io.adafruit_io import IO_HTTP, AdafruitIO_RequestError
```

```
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

aio_username = os.getenv('aio_username')
aio_key = os.getenv('aio_key')

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())
# Initialize an Adafruit IO HTTP API object
io = IO_HTTP(aio_username, aio_key, requests)
print("connected to io")

#  use Pico W's GP0 for SDA and GP1 for SCL
i2c = busio.I2C(board.GP1, board.GP0)
aht20 = adafruit_ahtx0.AHTx0(i2c)

try:
# get feed
    picowTemp_feed = io.get_feed("pitemp")
    picowHumid_feed = io.get_feed("pihumid")
except AdafruitIO_RequestError:
# if no feed exists, create one
    picowTemp_feed = io.create_new_feed("pitemp")
    picowHumid_feed = io.create_new_feed("pihumid")

#  pack feed names into an array for the loop
feed_names = [picowTemp_feed, picowHumid_feed]
print("feeds created")

clock = 300

while True:
    try:
        #  when the clock runs out..
        if clock > 300:
            #  read sensor
            data = [aht20.temperature, aht20.relative_humidity]
            #  send sensor data to respective feeds
            for z in range(2):
                io.send_data(feed_names[z]["key"], data[z])
                print("sent %0.1f" % data[z])
                time.sleep(1)
            #  print sensor data to the REPL
            print("\nTemperature: %0.1f C" % aht20.temperature)
            print("Humidity: %0.1f %%" % aht20.relative_humidity)
            print()
            time.sleep(1)
            #  reset clock
            clock = 0
        else:
            clock += 1
    # pylint: disable=broad-except
    #  any errors, reset Pico W
    except Exception as e:
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
    #  delay
    time.sleep(1)
    print(clock)
```

Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.

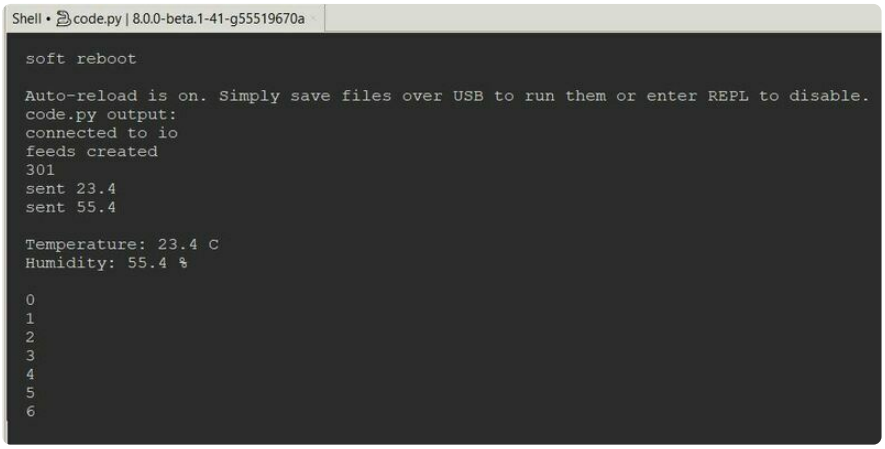
Add Your **settings.toml** File

Remember to add your **settings.toml** file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `CIRCUITPY_WIFI_SSID`, `CIRCUITPY_WIFI_PASSWORD`, `aio_username` and `aio_key` in the file.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"

aio_username = "your-aio-username-here"
aio_key = "your-aio-key-here"
```

Run **code.py**

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



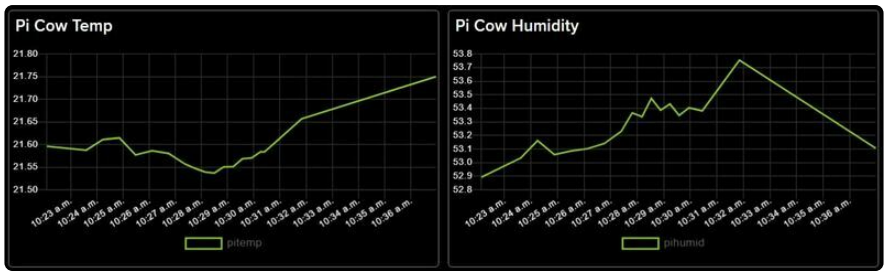
In the code, two feeds for Adafruit IO are created for each of the streams of data from the AHT20.

```
try:
# get feed
    picowTemp_feed = io.get_feed("pitemp")
    picowHumid_feed = io.get_feed("pihumid")
except AdafruitIO_RequestError:
# if no feed exists, create one
    picowTemp_feed = io.create_new_feed("pitemp")
    picowHumid_feed = io.create_new_feed("pihumid")
```

In the loop, every 5 minutes the AHT20's temperature and humidity measures are sent to Adafruit IO. When the data is sent, it is printed to the REPL.

```
#  when the clock runs out..
        if clock &gt; 300:
            #  read sensor
            data = [aht20.temperature, aht20.relative_humidity]
            #  send sensor data to respective feeds
            for z in range(2):
                io.send_data(feed_names[z]["key"], data[z])
                print("sent %0.1f" % data[z])
                time.sleep(1)
            #  print sensor data to the REPL
            print("\nTemperature: %0.1f C" % aht20.temperature)
            print("Humidity: %0.1f %%" % aht20.relative_humidity)
            print()
            time.sleep(1)
            #  reset clock
            clock = 0
```

On Adafruit IO, you can add the feeds to your dashboard to view your data over time.



Pico W with Azure IoT Central

In this example, you'll use the AHT20 temperature and humidity sensor (https://adafru.it/11xC) to log temperature and humidity data to Microsoft Azure IoT Central.

Prerequisite Guides

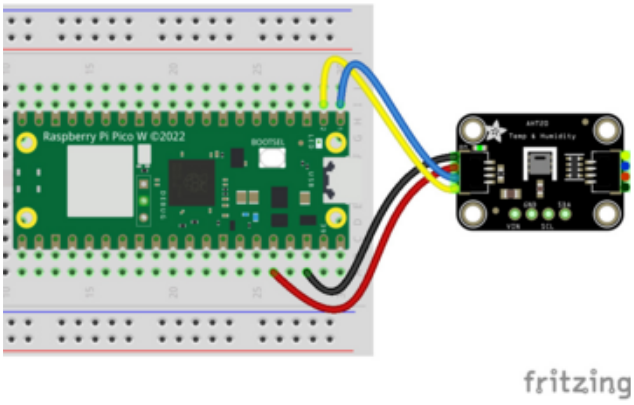
Adafruit AHT20 Temperature & Humidity Sensor

https://adafru.it/11xC

Getting Started with Microsoft Azure and CircuitPython

https://adafru.it/10dY

AHT20 Wiring



Board 3V to **sensor VIN**
Board GND to **sensor GND**
Board GP1 to **sensor SCL**
Board GP0 to **sensor SDA**

Code the Azure IoT Central Test

Once you've finished setting up your Pico W with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import os
import json
import busio
import microcontroller
import board
import rtc
import socketpool
import wifi
import adafruit_ntp
import adafruit_ahtx0
from adafruit_azureiot import IoTCentralDevice

#  use Pico W's GP0 for SDA and GP1 for SCL
i2c = busio.I2C(board.GP1, board.GP0)
aht20 = adafruit_ahtx0.AHTx0(i2c)

print("Connecting to WiFi...")
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

print("Connected to WiFi!")

#  ntp clock
pool = socketpool.SocketPool(wifi.radio)
ntp = adafruit_ntp.NTP(pool)
rtc.RTC().datetime = ntp.datetime

if time.localtime().tm_year < 2022:
    print("Setting System Time in UTC")
    rtc.RTC().datetime = ntp.datetime

else:
    print("Year seems good, skipping set time.")

# Create an IoT Hub device client and connect
esp = None
pool = socketpool.SocketPool(wifi.radio)
device = IoTCentralDevice(
    pool, esp, os.getenv('id_scope'), os.getenv('device_id'),
os.getenv('device_primary_key')
)

print("Connecting to Azure IoT Central...")

device.connect()

print("Connected to Azure IoT Central!")

#  clock to count down to sending data to Azure
azure_clock = 500

while True:
    try:
        #  when the azure clock runs out
        if azure_clock > 500:
            #  pack message
            message = {"Temperature": aht20.temperature,
                       "Humidity": aht20.relative_humidity}
            print("sending json")
```

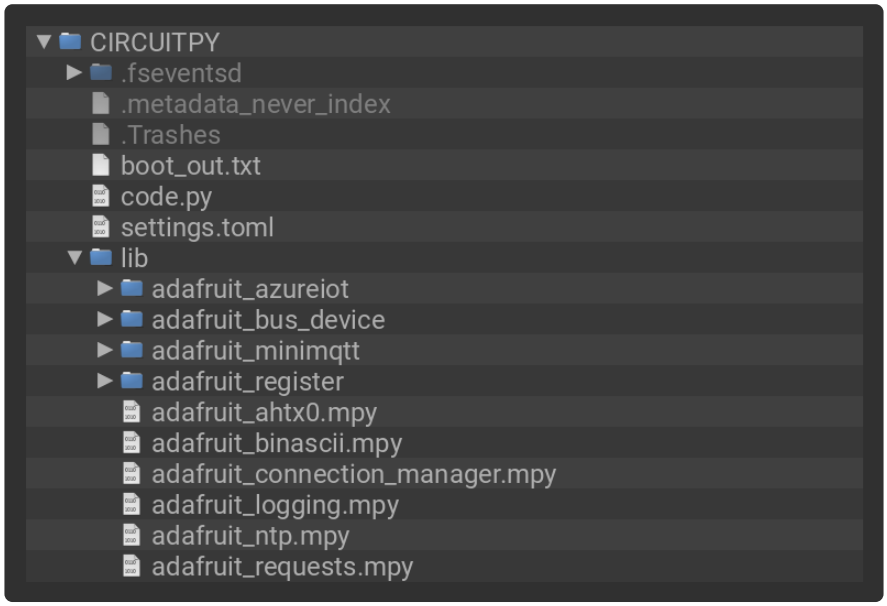
```
            device.send_telemetry(json.dumps(message))
            print("data sent")
            #  reset azure clock
            azure_clock = 0
        else:
            azure_clock += 1
        #  ping azure
        device.loop()
    #  if something disrupts the loop, reconnect
    # pylint: disable=broad-except
    #  any errors, reset Pico W
    except Exception as e:
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
    #  delay
    time.sleep(1)
    print(azure_clock)
```

Upload the Code and Libraries to the Pico W

After downloading the Project Bundle, plug your Pico W into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Pico W's **CIRCUITPY** drive.

- **lib** folder
- **code.py**

Your Pico W **CIRCUITPY** drive should look like this after copying the **lib** folder and the **code.py** file.



Add Your **settings.toml** File

Remember to add your **settings.toml** file as described in the Create Your settings.toml File page (https://adafru.it/18f9) earlier in the guide. You'll need to include your `WIFI_SSID` and `WIFI_PASSWORD` in the file. Additionally, you'll need your Azure IoT Central `id_scope`, `device_id` and `device_primary_key`.

```
CIRCUITPY_WIFI_SSID = "your-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-ssid-password-here"

id_scope = "your-id-scope-here"
device_id = "your-device-id-here"
device_primary_key = "your-device-primary-key-here"
```

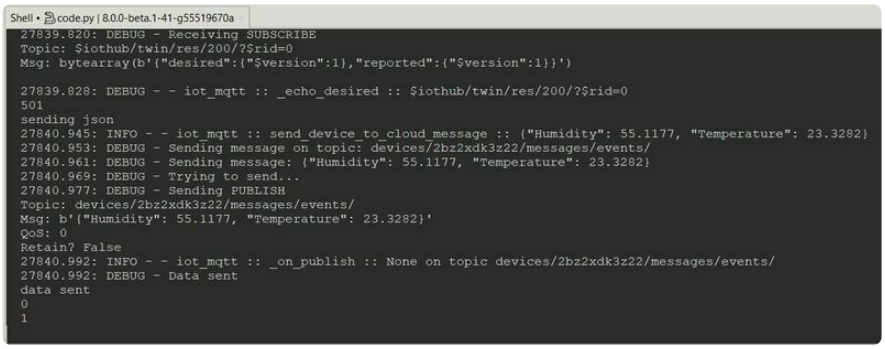
You'll need to setup a Microsoft Azure account and create an Azure IoT Central application to properly use this example. Be sure to reference the getting started with Microsoft Azure and CircuitPython guide (https://adafru.it/10dY) to follow all of the steps for this process successfully.

Getting Started with Microsoft Azure and CircuitPython

https://adafru.it/10dY

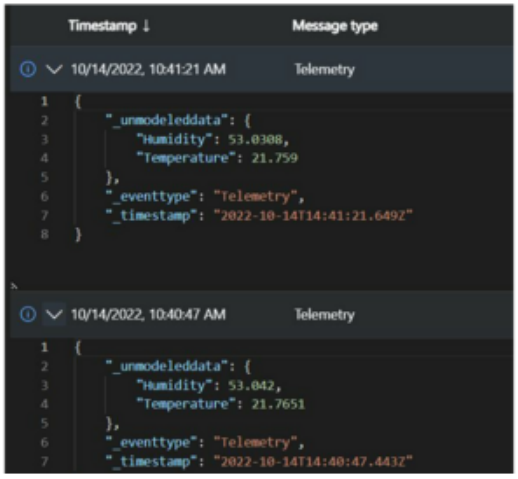
Run **code.py**

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!



Every five minutes, the AHT20's temperature and humidity data will be packed into a JSON message and transmitted to your Azure IoT Central app. In the REPL, you'll see `DEBUG` messages from `adafruit_requests` and messages from the loop letting you know when the JSON message has been sent and the sensor readings from the AHT20.

```
#  when the azure clock runs out
        if azure_clock &gt; 500:
            #  pack message
            message = {"Temperature": aht20.temperature,
                       "Humidity": aht20.relative_humidity}
            print("sending json")
            device.send_telemetry(json.dumps(message))
            print("data sent")
            #  reset azure clock
            azure_clock = 0
```



In your Azure IoT Central app, you'll see your transmitted telemetry under your device.