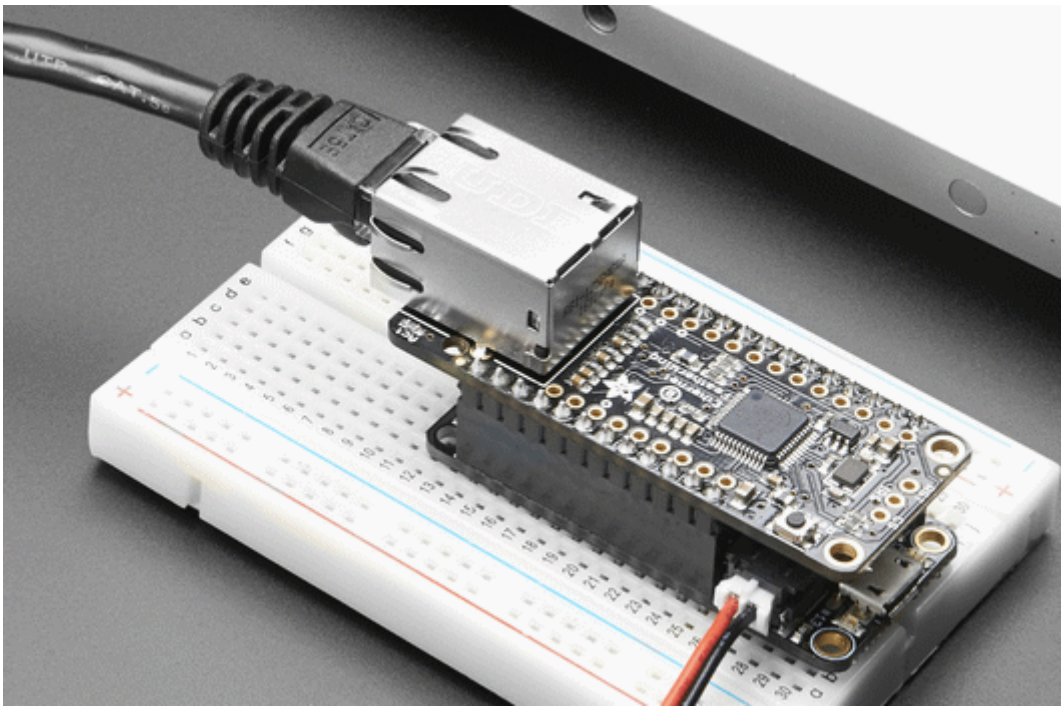
Networking in CircuitPython

Created by Anne Barela

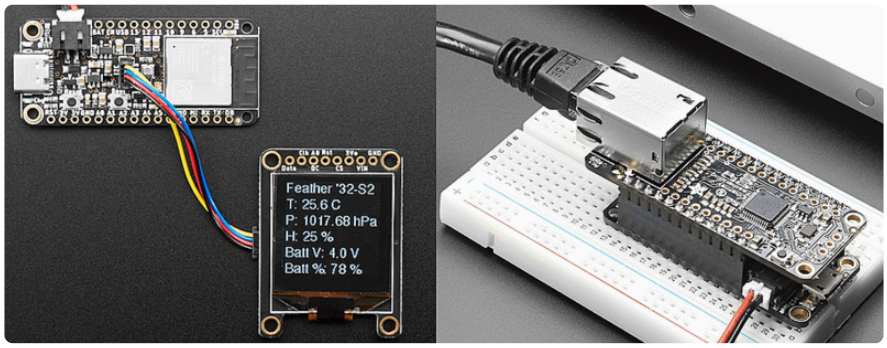


https://learn.adafruit.com/networking-in-circuitpython

Last updated on 2025-05-20 03:29:04 PM EDT

Table of Contents

Overview



The other day, the CircuitPython Team was talking about a new network feature and looking to add it to a guide. "Do we have a networking guide?" No, such a guide, while thought of, never materialized. No more!

This guide is aimed at helping to demonstrate wired and wireless networking using CircuitPython.

The options for wired connections are currently overshadowed by the wireless options, but the use is the same for both.

This guide will cover the following topics:

- networking hardware choices
- connecting to a local network
- typical network operations
- more esoteric things (UDP, mDNS, ...)

Hardware Choices

Networking with CircuitPython is constrained to hardware that supports CircuitPython. While the hardware platforms compatible with CircuitPython contiues to grow, generally there is hardware only from several manufacturers.

This page lists Wifi-capable based on current products with CircuitPython support.

Espressif Microcontrollers

Espressif makes several WiFi-capable microcontrollers. The processors include the original ESP32, the ESP32-S2, ESP32-S3, ESP32-C2 (aka ESP8584), ESP32-C3, and ESP32-C6.

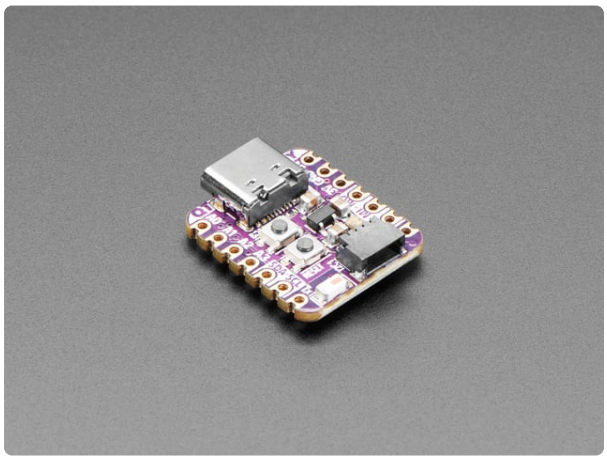
CircuitPython does not support the earlier Espressif ESP8266 chip because of its memory and hardware limitations (although it is supported by MicroPython).

The CircuitPython `wifi` module is the primary interface with Espressif microcontrollers.

Products

The products listed throughout are representative. Much of the time there are many more. See CircuitPython.org for all the compatible boards (https://adafru.it/Em8).

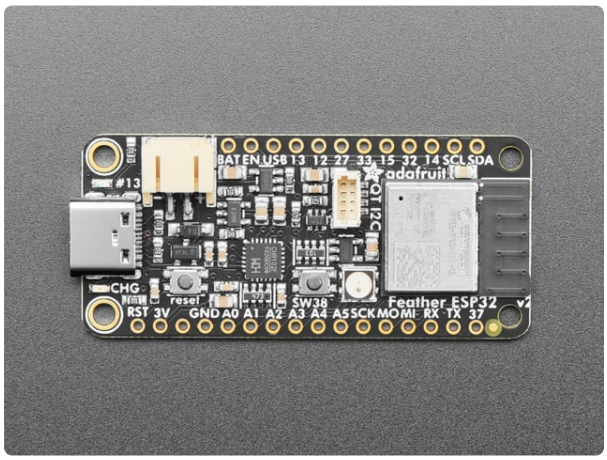
ESP32



Adafruit QT Py ESP32 Pico - WiFi Dev Board with STEMMA QT

This dev board is like when you're watching a super-hero movie and the protagonist shows up in a totally amazing costume in the third act and you're like 'OMG! That's...

https://www.adafruit.com/product/5395

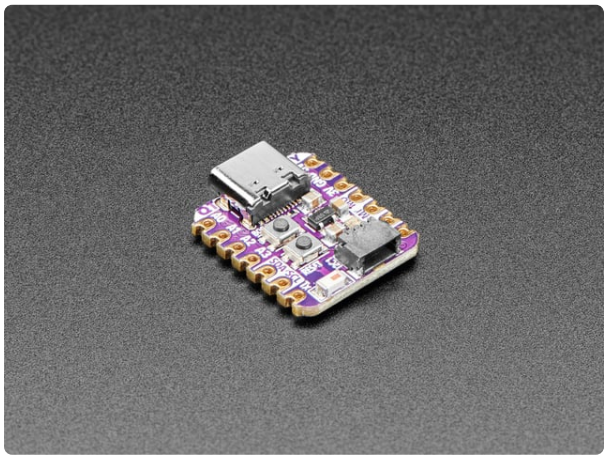


Adafruit ESP32 Feather V2 - 8MB Flash + 2 MB PSRAM

One of our star Feathers is the Adafruit HUZZAH32 ESP32 Feather - with the fabulous ESP32 WROOM module on there, it makes quick work...

https://www.adafruit.com/product/5400

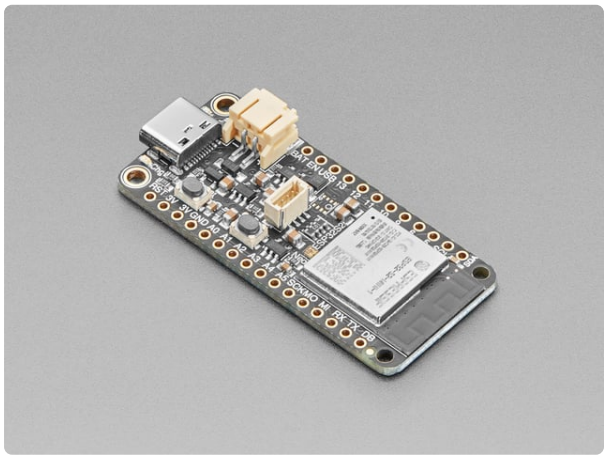
ESP32-S2



Adafruit QT Py ESP32-S2 WiFi Dev Board with STEMMA QT

What has your favorite Espressif WiFi microcontroller, comes with our favorite connector - the STEMMA QT, a chainable I2C port, and has...

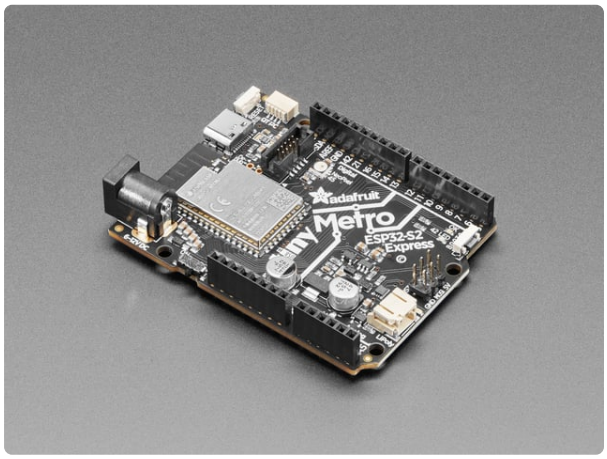
https://www.adafruit.com/product/5325



Adafruit ESP32-S2 Feather - 4 MB Flash + 2 MB PSRAM

What's Feather-shaped and has an ESP32-S2 WiFi module? What has a STEMMA QT connector for I2C devices? What has your favorite Espressif WiFi microcontroller and lots of Flash and...

https://www.adafruit.com/product/5000

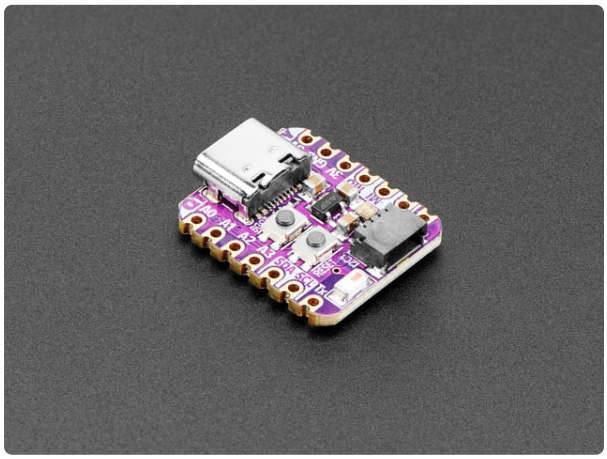


Adafruit Metro ESP32-S2

What's Metro shaped and has an ESP32-S2 WiFi module? What has a STEMMA QT connector for I2C devices, and a Lipoly charger circuit? What has your favorite Espressif WiFi...

https://www.adafruit.com/product/4775

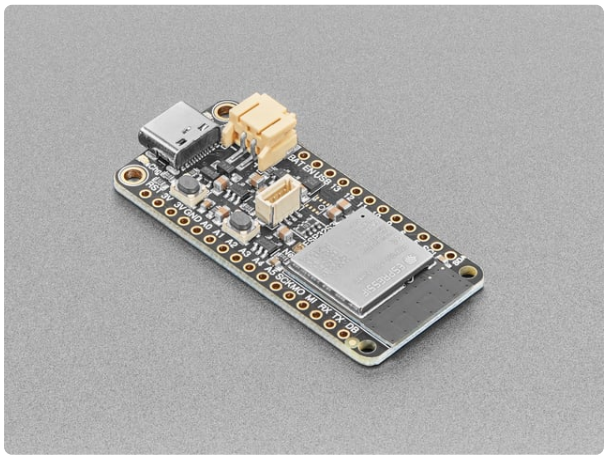
ESP32-S3



Adafruit QT Py ESP32-S3 WiFi Dev Board with STEMMA QT

The ESP32-S3 has arrived in QT Py format - and what a great way to get started with this powerful new chip from Espressif! With dual 240 MHz cores, WiFi and BLE support, and native...

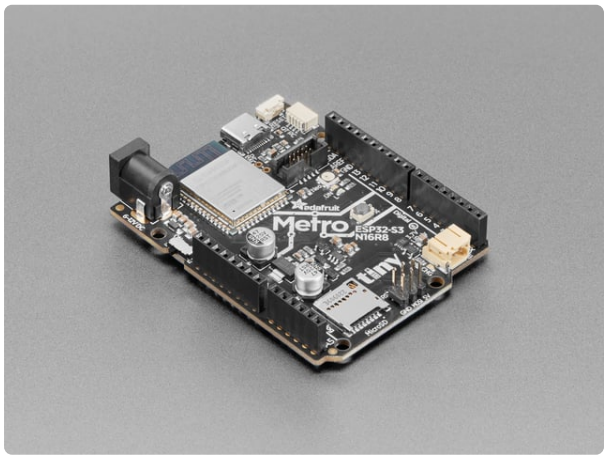
https://www.adafruit.com/product/5426



Adafruit ESP32-S3 Feather with STEMMA QT / Qwiic

The ESP32-S3 has arrived in Feather format - and what a great way to get started with this powerful new chip from Espressif! With dual 240 MHz cores, WiFi and BLE support, and native...

https://www.adafruit.com/product/5323

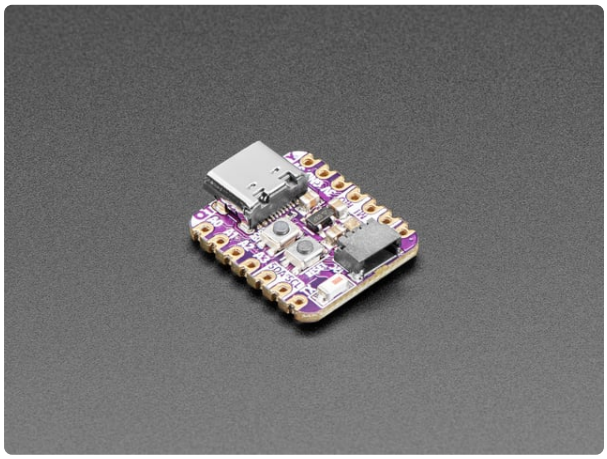


Adafruit Metro ESP32-S3 with 16 MB Flash 8 MB PSRAM

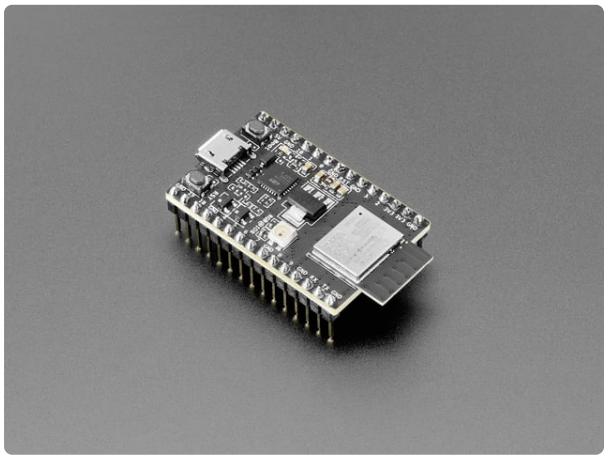
What's Metro-shaped and has an ESP32-S3 WiFi module? What has a STEMMA QT connector for I2C devices and a Lipoly charger circuit? What has your favorite Espressif WiFi...

https://www.adafruit.com/product/5500

ESP32-C3

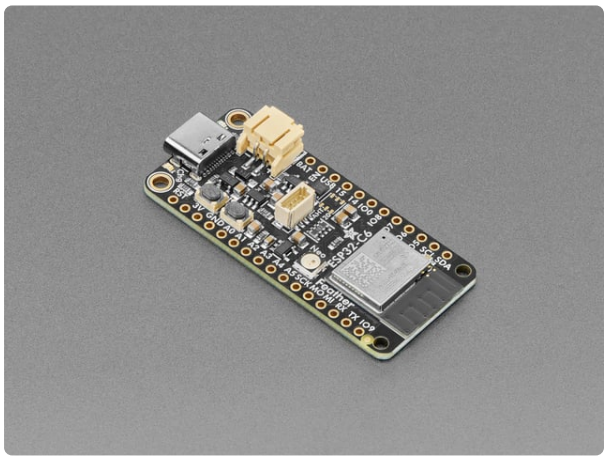


[Adafruit QT Py ESP32-C3 WiFi Dev Board with STEMMA QT]
What's life without a little RISC? This miniature dev board is perfect for small projects: it comes with our favorite connector - the...
https://www.adafruit.com/product/5405

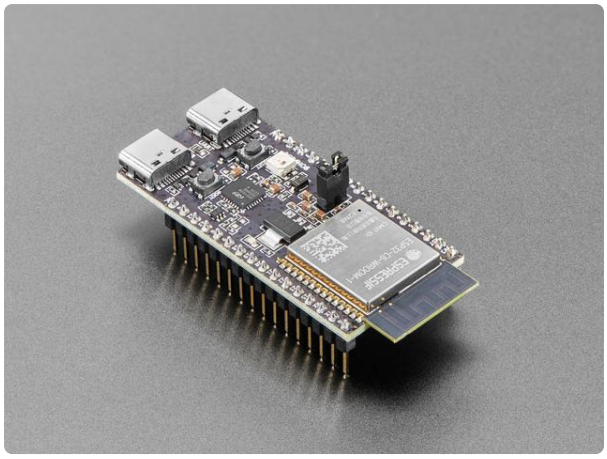


[ESP32-C3 DevKitM-01 - 4 MB SPI Flash]
The ESP32-C3-DevKitM-01 is an entry-level development board equipped with the ESP32-C3-MINI-01, a powerful, generic Wi-Fi + Bluetooth LE MCU module that features...
https://www.adafruit.com/product/5337

ESP32-C6



[Adafruit ESP32-C6 Feather - STEMMA QT]
The ESP32-C6 is Espressif's first Wi-Fi 6 SoC integrating 2.4 GHz Wi-Fi 6, Bluetooth 5 (LE) and the 802.15.4 protocol. It brings the goodness you know from the
https://www.adafruit.com/product/5933



ESP32-C6-DevKitC-1-N8 - 8MB SPI Flash
The ESP32-C6-DevKitC-1-N8 is an entry-level development board equipped with ESP32-C6-WROOM-1, a general-purpose Wi-Fi + Bluetooth LE RISC-V MCU module...
https://www.adafruit.com/product/5672

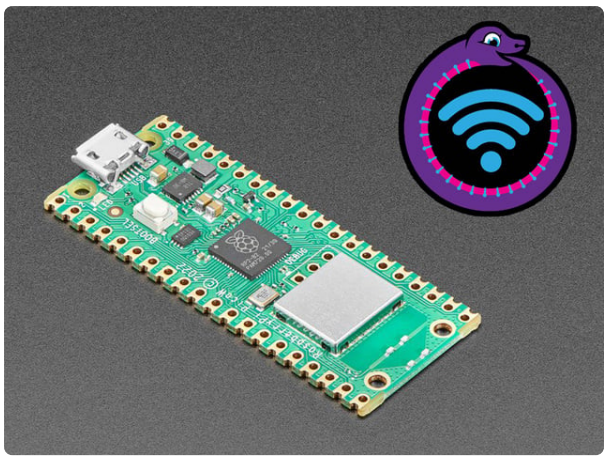
Raspberry Pi Pico W

Raspberry Pi Pico W brings WiFi to the Pi Pico platform, adding on-board a single-band 2.4GHz wireless interface (802.11n) using the Infineon CYW43439 radio module, while retaining complete pin compatibility with its older sibling, the original Pi Pico.

The CircuitPython `wifi` module is the primary interface for WiFi on the Pico W.

Note that while the Pico W has more memory than many microcontroller boards, the WiFi software takes up a great deal of flash space and uses a lot of RAM at runtime, limiting the size of programs that can be created (which use WiFi) on the Pico W.

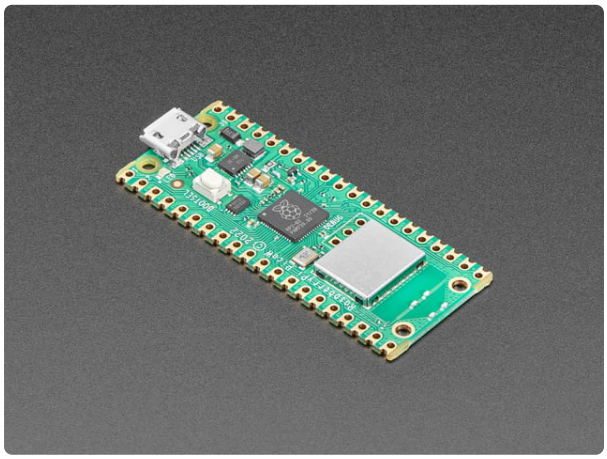
Guide



Quick-Start the Pico W WiFi with CircuitPython
By Liz Clark
Overview

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

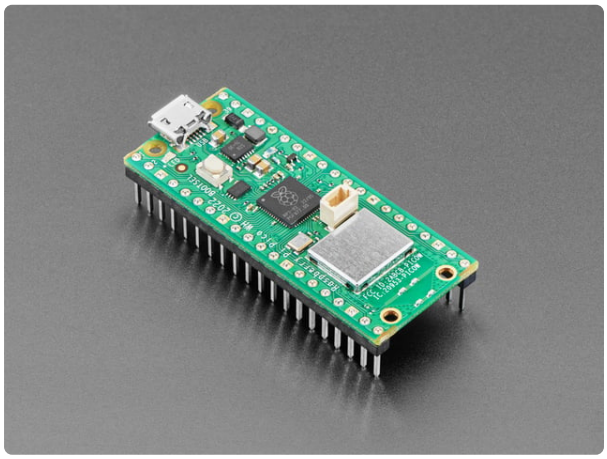
Products



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



Raspberry Pi Pico WH - Pico Wireless with Headers Soldered

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/5544

Airlift

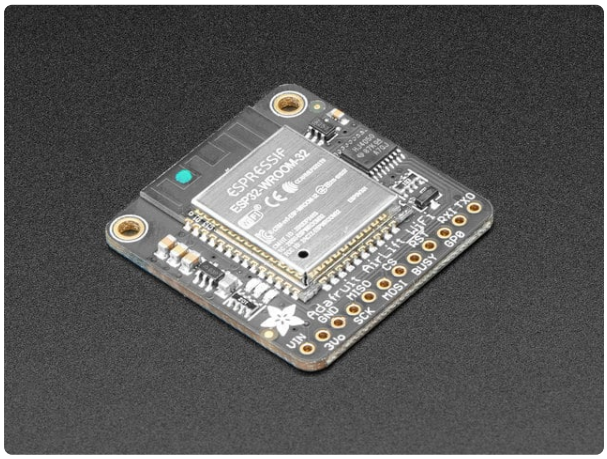
Airlift is the term Adafruit uses for using an ESP32 microcontroller module as a WiFi coprocessor, connected to another microcontroller running CircuitPython. The ESP32 runs a modified version of the **NINA-FW** firmware developed by Arduino.

Your program controls the AirLift coprocessor using the `adafruit_esp32spi` library (aka **ESP32SPI**), which is written in Python.

Note that ESP32SPI requires the CircuitPython microcontroller to have at least 128kb of memory, which rules out smaller CircuitPython-compatible microcontrollers like SAMD21.

The AirLift coprocessor is integrated onto boards such as the Adafruit PyPortal and the Adafruit Metro M4 AirLift Lite, and is also available as a separate breakout board.

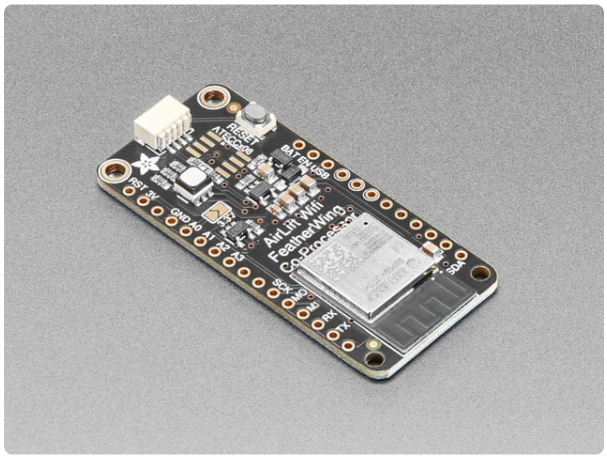
Guide



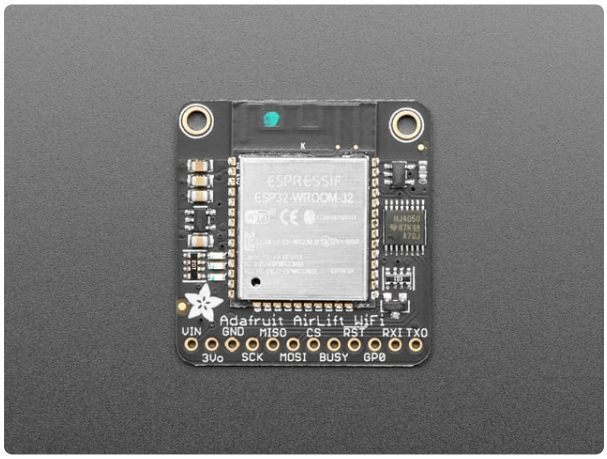
Adafruit AirLift - ESP32 WiFi Co-Processor Breakout
By Kattni Rembor
CircuitPython WiFi

https://learn.adafruit.com/adafruit-airlift-breakout/circuitpython-wifi

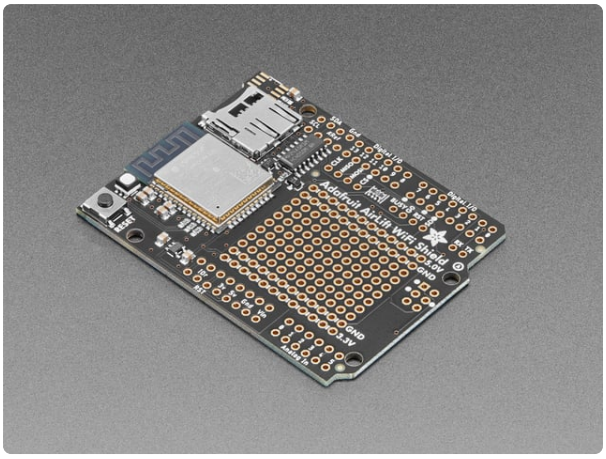
Products



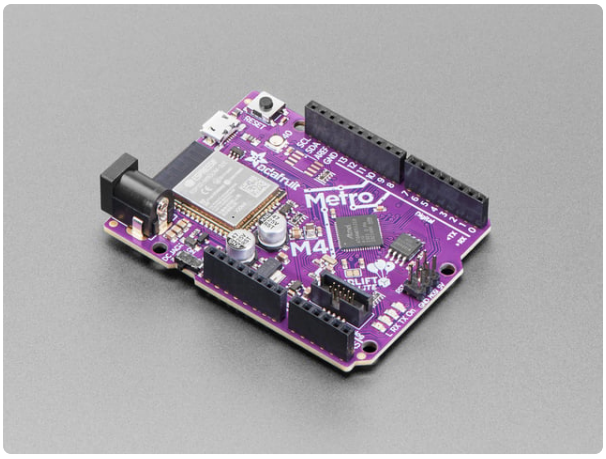
Adafruit AirLift FeatherWing – ESP32 WiFi Co-Processor
Give your Feather project a lift with the Adafruit AirLift FeatherWing - a FeatherWing that lets you use the powerful ESP32 as a WiFi co-processor. You probably have your...
https://www.adafruit.com/product/4264



Adafruit AirLift – ESP32 WiFi Co-Processor Breakout Board
Give your plain ol' microcontroller project a lift with the Adafruit AirLift - a breakout board that lets you use the powerful ESP32 as a WiFi co-processor. You probably...
https://www.adafruit.com/product/4201



[Adafruit AirLift Shield - ESP32 WiFi Co-Processor]
Give your Arduino project a lift with the Adafruit AirLift Shield - a shield that lets you use the powerful ESP32 as a WiFi co-processor. You probably have your favorite...
https://www.adafruit.com/product/4285

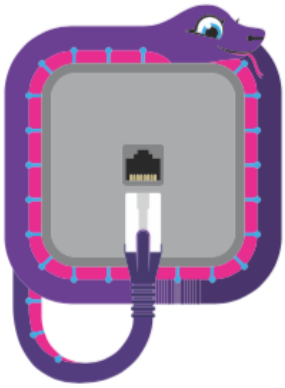


[Adafruit Metro M4 Express AirLift (WiFi) - Lite]
Give your next project a lift with AirLift - our witty name for the ESP32 co-processor that graces this Metro M4. You already know about the Adafruit Metro...
https://www.adafruit.com/product/4000

WizNet 5k Library

WizNet makes a number of chips for doing hardwired Ethernet through an SPI bus to a microcontroller. Their W5000 series chips are supported in CircuitPython through the Adafruit Wiznet5k Library module `adafruit_wiznet5k`.

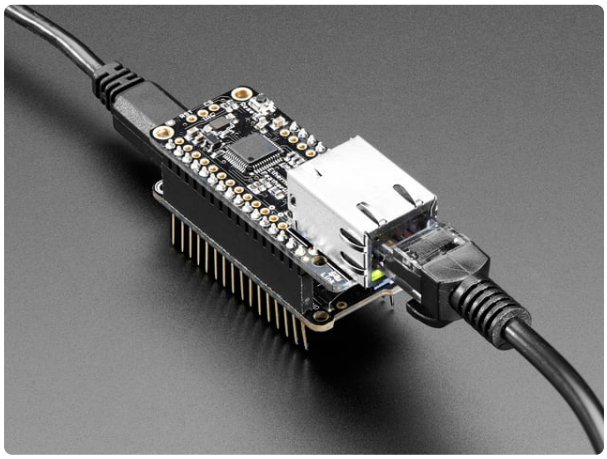
Guide



Ethernet for CircuitPython with Wiznet5K
By Brent Rubell
Overview

https://learn.adafruit.com/ethernet-for-circuitpython/overview

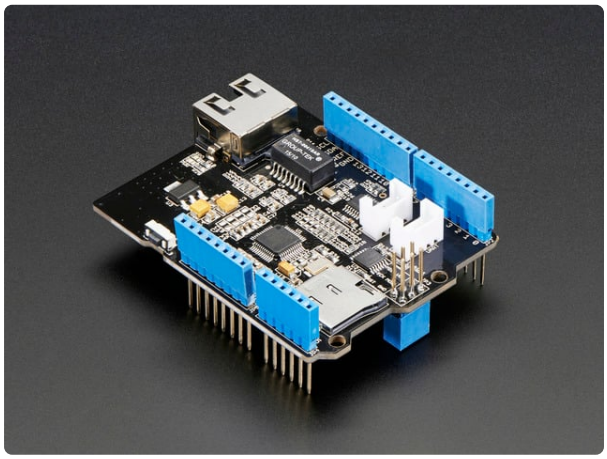
Products



Adafruit Ethernet FeatherWing

Wireless is wonderful, but sometimes you want the strong reliability of a wire. If your Feather board is going to be part of a permanent installation, this Ethernet...

https://www.adafruit.com/product/3201



Ethernet Shield for Arduino - W5500 Chipset

The W5500 Ethernet Shield for Arduino from Seeed Studio is a great way to set up your projects with internet connectivity with just a single chip. Similar to the

https://www.adafruit.com/product/2971

Network Settings

Following good code security practices, network name and security credentials should not be "hardcoded" into CircuitPython programs. Rather they are placed in Python environment variables.

As there is no operating system used on most CircuitPython devices, the values are placed in a separate file named **settings.toml**.

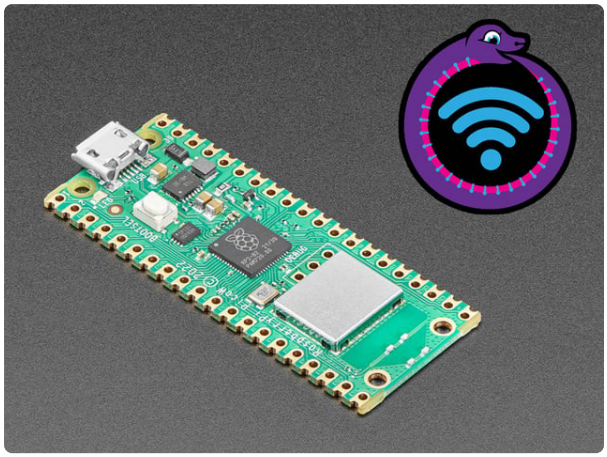
Using **settings.toml** replaces using **secrets.py** in modern CircuitPython code.

For those platforms presenting as a thumb drive (most microcontrollers), the file is placed in the root directory of the **CIRCUITPY** drive.

For microcontrollers not presenting as a thumb drive, the file should be uploaded along with the code and libraries using a compatible tool, such as the CircuitPython Web Workflow Code Editor (see below).

Putting Your Networking Settings in **settings.toml**

There is a handy guide page for how to set up a **settings.toml** file for CircuitPython networking:



Quick-Start the Pico W WiFi with CircuitPython
By Liz Clark
Create Your settings.toml File

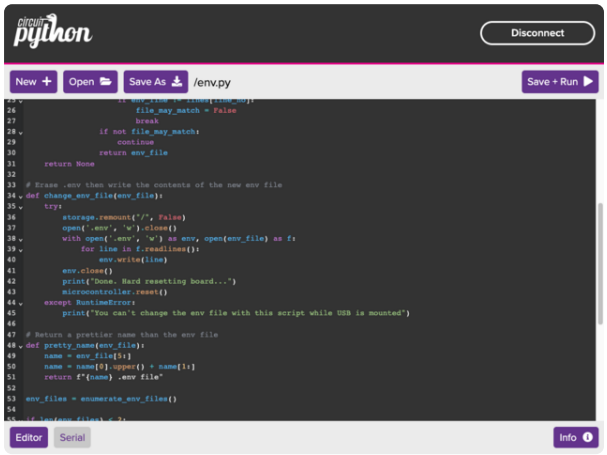
https://learn.adafruit.com/pico-w-wifi-with-circuitpython/create-your-settings-toml-file

Defining the values for `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD` provide an automatic way for the network name and password to be used in CircuitPython programs.

It is also possible to store the values in **settings.toml** and not have them used automatically by CircuitPython. You can still use **settings.toml** to store your credentials, say in `WIFI_SSID` and `WIFI_PASSWORD`, then use those values in your own code.

Adafruit Web Workflow

Adafruit Web Workflow is an in-browser code editor and environment for CircuitPython using WiFi connections. See the following guide for setup and use of Web Workflow.



CircuitPython Web Workflow Code Editor Quick Start
By M. LeBlanc-Williams
Overview

https://learn.adafruit.com/getting-started-with-web-workflow-using-the-code-editor/overview

Terminology

Here are some terms you'll see in this documentation referring to how networking is used in Python and CircuitPython.

What is TCP vs. UDP?

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are both internet protocol suite methods for sending information across the internet.

The main difference between the two is that TCP is connection-based, while UDP is connectionless. This means that TCP requires the two ends of the communication link to remain connected throughout communication, while UDP does not.

What is a Socket?

A socket (network socket) is established within your CircuitPython program to serve as an endpoint for sending and receiving data from/to your local network or the Internet.

A socket's address is defined by its protocol, IP address and port number. The protocol is usually TCP, which is a reliable connection-based protocol with acknowledgments and hand-shaking. Other protocols might also be available such as UDP, which is a connectionless "send and forget" protocol.

For HTTP, a typical socket specifies TCP, the server IP address, and port (https://adafru.it/1a5x) 80. For HTTPS, port number 443 is usually used. UDP sockets are generally 1024 or higher.

What is a Socket Pool?

The `socketpool` module provides sockets through a pool of available sockets. When you are finished using a socket, it is returned to the pool. The pools themselves act like CPython's `socket` (https://adafru.it/1a5y) module.

Only one socket pool can be created for each radio.

Due to the smaller memory size of most microcontrollers and single board computers, the amount of memory available for sockets is limited. Depending on the memory usage, the number of available sockets in the pool may be exhausted. You will need to use proper management of available sockets in a socketpool.

Detailed information on functions and parameters may be found in the CircuitPython documentation (https://adafru.it/1a5y).

What is Secure Sockets Layer (SSL or SSL/TLS)?

Secure Sockets Layer (SSL) is a way of encrypting data that is transmitted over a network, to make the connection secure. SSL evolved into TLS (Transport Layer Security), and the mechanism is now often called SSL/TLS.

One of the most common uses for SSL/TLS is HTTPS, a secure way of making HTTP web requests. SSL/TLS in CircuitPython helps establish a secure HTTPS connection between a CircuitPython device and a secure internet server running HTTPS, now used by most of the web.

In the early days of microcontroller WiFi use, only insecure HTTP requests could be made. When the internet switched over to HTTPS (https://adafru.it/1a5z) due to threats around 2016, it left those older implementations without connections.

SSL/TLS provides authentication and encryption by using public-key cryptography. The public keys are known as certificates (https://adafru.it/1a5A). The public keys have corresponding private keys that are kept secret. Trusted certificate providers issue root certificates. Other certificates are derived from the limited number of root certificates. A set of root certificates is usually stored in the WiFi firmware to allow connection to HTTPS servers whose certificates are based on those roots. You can also supply your own certificates.

Espressif discusses the certificates for ESP products here (https://adafru.it/1a5B).

What is JSON?

JSON (https://adafru.it/1a5C) (JavaScript Object Notation) array is a method of enncoding data in a standard format for files or data interchange (Wikipedia (https://adafru.it/BYZ)).

CircuitPython has the `json` (https://adafru.it/1a5D) module to assist in converting between Python objects and the JSON data format.

You may see examples of the `response` Module using JSON to send and receive data.

Networking with the wifi module

The `wifi` Module

The `wifi` module provides a simple interface between CircuitPython and the internet using WiFi. It is a built-in module on Espressif and Pico W boards.

Here are a couple of short examples from the guide Todbot's CircuitPython Tricks (https://adafru.it/1a5E):

Scan Local WiFi Networks

```
import wifi
networks = []
for network in wifi.radio.start_scanning_networks():
    networks.append(network)
wifi.radio.stop_scanning_networks()
networks = sorted(networks, key=lambda net: net.rssi, reverse=True)
for network in networks:
    print("ssid:",network.ssid, "rssi:",network.rssi)
```

Displaying Your Local IP Address

This short program uses the `wifi` module to connect to the local network, using credentials you set up in a **settings.toml** file, and then gets the internet protocol (IP) address of your device and prints it out.

```
# settings.toml
CIRCUITPY_WIFI_SSID = "PrettyFlyForAWiFi"
CIRCUITPY_WIFI_PASSWORD = "mysecretpassword"
```

```
# code.py
import os, wifi
print("connecting...")
wifi.radio.connect(ssid=os.getenv('CIRCUITPY_WIFI_SSID'),
```

```
                    password=os.getenv('CIRCUITPY_WIFI_PASSWORD'))
print("my IP addr:", wifi.radio.ipv4_address)
```

Using `adafruit_connection_manager`

The `adafruit_connection_manager` library provides a simple way to get a socket pool or an SSL context (used for HTTPS requests). It supports using the `wifi` module , the ESP32SPI library, and can also work on the desktop using CPython ("regular" Python).

Example:

```
import wifi
import adafruit_connection_manager
import adafruit_requests

radio = wifi.radio

# Add code to make sure your radio is connected

pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)
requests.get("http://wifitest.adafruit.com/testwifi/index.html")

# Do something with response
```

The `adafruit_requests` Library

The `adafruit_requests` library provides functions similar to the CPython `requests` module, used for HTTP(S) commands.

Example (https://adafru.it/1a5F):

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
# Updated for CircuitPython 9.0
"""WiFi Simpletest"""

import os

import adafruit_connection_manager
import wifi

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "https://httpbin.org/get"
JSON_POST_URL = "https://httpbin.org/post"

# Initalize Wifi, Socket Pool, Request Session
pool = adafruit_connection_manager.get_radio_socketpool(wifi.radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(wifi.radio)
requests = adafruit_requests.Session(pool, ssl_context)
rssi = wifi.radio.ap_info.rssi
```

```
print(f"\nConnecting to {ssid}...")
print(f"Signal Strength: {rssi}")
try:
    # Connect to the Wi-Fi network
    wifi.radio.connect(ssid, password)
except OSError as e:
    print(f"  OSError: {e}")
print("  Wifi!")

print(f" | GET Text Test: {TEXT_URL}")
with requests.get(TEXT_URL) as response:
    print(f" |   GET Response: {response.text}")
print("-" * 80)

print(f" | GET Full Response Test: {JSON_GET_URL}")
with requests.get(JSON_GET_URL) as response:
    print(f" |   Unparsed Full JSON Response: {response.json()}")
print("-" * 80)

DATA = "This is an example of a JSON value"
print(f" |   JSON 'value' POST Test: {JSON_POST_URL} {DATA}")
with requests.post(JSON_POST_URL, data=DATA) as response:
    json_resp = response.json()
    # Parse out the 'data' key from json_resp dict.
    print(f" |   JSON 'value' Response: {json_resp['data']}")
print("-" * 80)

json_data = {"Date": "January 1, 1970"}
print(f" |   JSON 'key':'value' POST Test: {JSON_POST_URL} {json_data}")
with requests.post(JSON_POST_URL, json=json_data) as response:
    json_resp = response.json()
    # Parse out the 'json' key from json_resp dict.
    print(f" |   JSON 'key':'value' Response: {json_resp['json']}")
print("-" * 80)

print("Finished!")
```

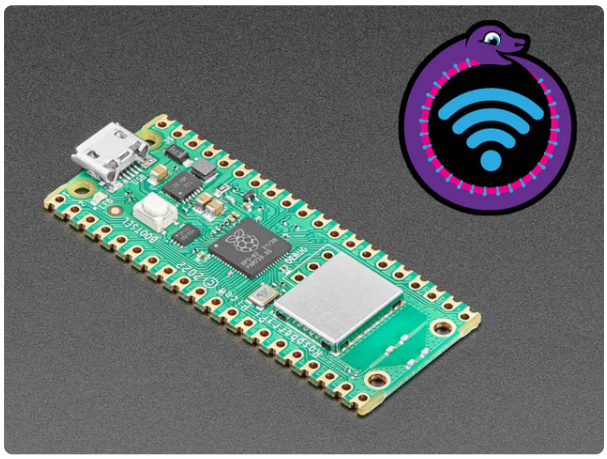
Using MQTT

MQTT is a messaging protocol for communicating between two nodes on the internet. It is often used for Internet of Things (IoT) devices to pass data.

```
# Simple demo of MQTT client in CircuitPython with native WiFi (ESP32-S Series)
# 9 Oct 2021 - @todbot / Tod Kurt
# 31 July 2024 Anne Barela for Adafruit Industries
#
# This will connect to WiFi, then connect to an MQTT broker (shiftr.io was tested)
# and then listen to one MQTT feed while periodically publishing to another MQTT
feed.
#
# Your settings.toml file contains something like:
#   CIRCUITPY_WIFI_SSID = "myWiFiName"
#   CIRCUITPY_WIFI_PASSWORD = "mywifipassword"
#   mqtt_broker ="test.mosquitto.org"
#   mqtt_port = 1883  # unencrytped, use 8883 for TLS encrypted
#   mqtt_username = ""
#   mqtt_password = ""
#
import os
import time
import ssl, socketpool, wifi
import adafruit_minimqtt.adafruit_minimqtt as MQTT

my_mqtt_topic_hello = "me/feeds/hello"  # the topic we send on
```

```
my_mqtt_topic_light = "me/feeds/light"  # the topic we receive on (could be the
same)

# Connect to WiFi
print(f"Connecting to {os.getenv('CIRCUITPY_WIFI_SSID')}")
wifi.radio.connect(os.getenv("CIRCUITPY_WIFI_SSID"),
os.getenv("CIRCUITPY_WIFI_PASSWORD"))

# Set up a MiniMQTT Client
mqtt_client = MQTT.MQTT(
    broker=os.getenv("mqtt_broker"),
    port=os.getenv("mqtt_port"),
    username=os.getenv("mqtt_username"),
    password=os.getenv("mqtt_password"),
    socket_pool=socketpool.SocketPool(wifi.radio),
    ssl_context=ssl.create_default_context(),
)

# Called when the client is connected successfully to the broker
def connected(client, userdata, flags, rc):
    print("Connected to MQTT broker!")

    client.subscribe( my_mqtt_topic_light) # say I want to listen to this topic

# Called when the client is disconnected
def disconnected(client, userdata, rc):
    print("Disconnected from MQTT broker!")

# Called when a topic the client is subscribed to has a new message
def message(client, topic, message):
    print("New message on topic {0}: {1}".format(topic, message))
    val = 0
    try:
        val = int(message)  # attempt to parse it as a number
    except ValueError:
        pass
    print("setting LED to color:",val)
    # led.fill(val)  # if we had leds

# Set the callback methods defined above
mqtt_client.on_connect = connected
mqtt_client.on_disconnect = disconnected
mqtt_client.on_message = message

print("Connecting to MQTT broker...")
mqtt_client.connect()

last_msg_send_time = 0

while True:
    print("waiting")

    mqtt_client.loop(timeout=1)  # see if any messages to me

    if time.monotonic() - last_msg_send_time > 3.0:  # send a message every 3 secs
        last_msg_send_time = time.monotonic()
        msg = "hi there! time is "+str(time.monotonic())
        print("sending MQTT msg..", msg)
        mqtt_client.publish( my_mqtt_topic_hello, msg )
```

Companion Guides



Quick-Start the Pico W WiFi with CircuitPython
By Liz Clark
Overview

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

Further Reading

ReadTheDocs

- Adafruit CircuitPython ConnectionManager Library (https://adafru.it/1a5p)
- Adafruit Requests Library (https://adafru.it/1a5G)

Third Party Guides

- Connect to Multiple WiFi Networks with your Raspberry Pi Pico W (https://adafru.it/1a5H)
- CircuitPython WiFi Manager (https://adafru.it/1a5I) - opens an access point to allow the user to configure the device to configure the device to connect to available WiFi networks. When the device is configured, it then connects to the first available matching network and hands over the control to your code.

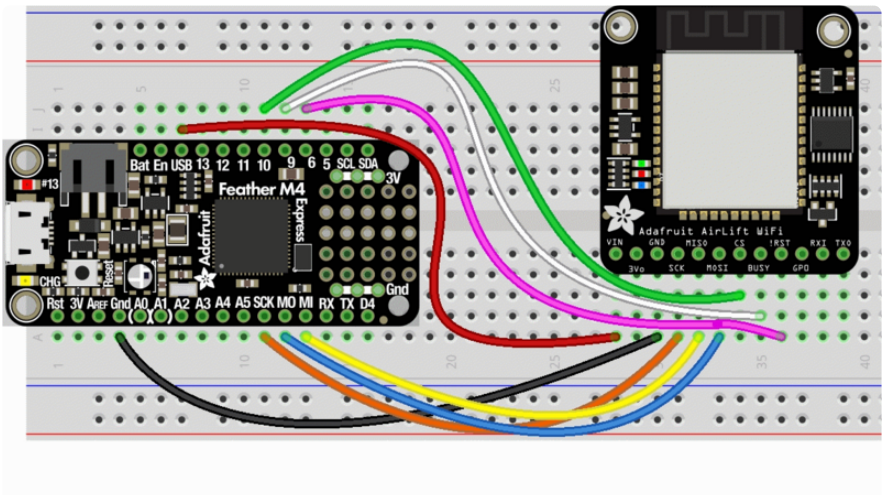
Networking with ESP32SPI on Airlift

Using an Adafruit Airlift module or Airlift capable development board with WiFi is not difficult, but it does require some code that other networking solutions do not require, mainly to set up the SPI bus communications between the main microcontroller and the ESP32 running NINA firmware.

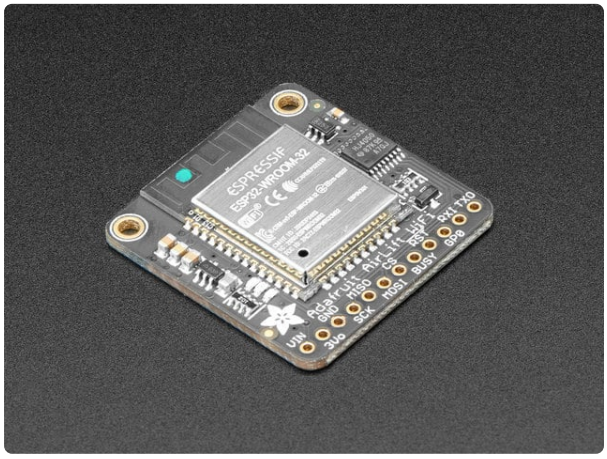
Connections to Enterprise WiFi are not supported by Airlift.

Airlift Board Wiring and Basic Code

Check out the page below for basic wiring with an Airlift breakout board connected to a Feather microcontroller board. The example scans for WiFi access points within range.



```
import board
import busio
from digitalio import DigitalInOut

from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests

print("ESP32 SPI hardware test")

esp32_cs = DigitalInOut(board.D10)
esp32_ready = DigitalInOut(board.D9)
esp32_reset = DigitalInOut(board.D7)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

if esp.status == adafruit_esp32spi.WL_IDLE_STATUS:
    print("ESP32 found and in idle mode")
print("Firmware vers.", esp.firmware_version)
print("MAC addr:", [hex(i) for i in esp.MAC_address])

for ap in esp.scan_networks():
    print("\t%s\t\tRSSI: %d" % (str(ap['ssid'], 'utf-8'), ap['rssi']))

print("Done!")
```



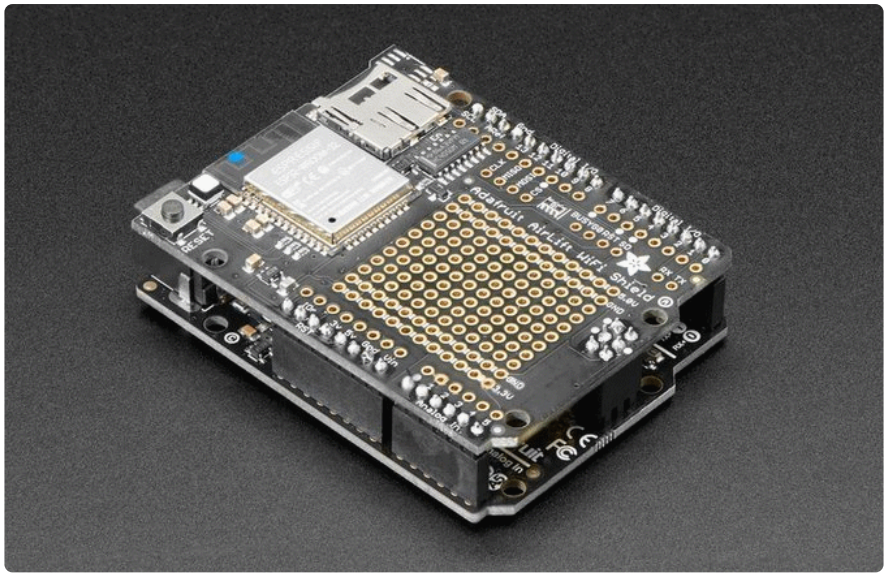
Adafruit AirLift - ESP32 WiFi Co-Processor Breakout

By Kattni Rembor

CircuitPython WiFi

https://learn.adafruit.com/adafruit-airlift-breakout/circuitpython-wifi

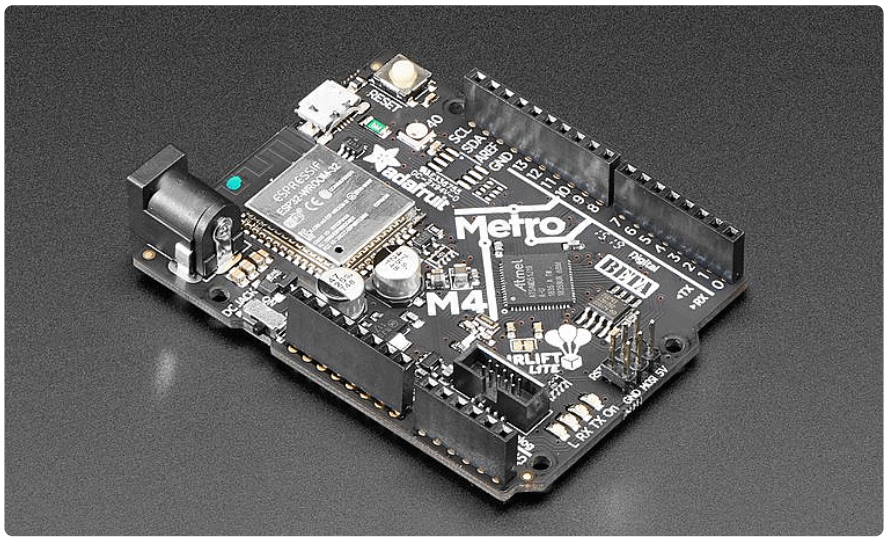
Airlift on the Airlift Shield



The Adafruit Airlift Shield provides an Airlift ESP32 coprocessor on an Arduino shield form factor. The pins for the Airlift are as follows:

- `esp32_cs = DigitalInOut(board.D10)`
- `esp32_ready = DigitalInOut(board.D7)`
- `esp32_reset = DigitalInOut(board.D5)`

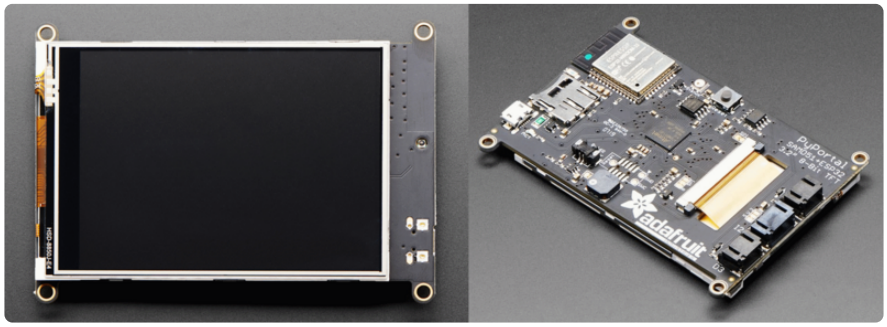
Airlift on the Metro M4 Express Airlift



The ESP32 coprocessor is on the following Cortex M4 pins in CircuitPython:

- CS Pin - `board.ESP_CS`
- Ready/Busy - `board.ESP_BUSY`
- Reset - `board.ESP_RESET`

Airlift on the Adafruit PyPortal



The ESP32 coprocessor is on the following Cortex M4 pins in CircuitPython:

- CS Pin - `board.ESP_CS`
- Ready/Busy - `board.ESP_BUSY`

- Reset - `board.ESP_RESET`

Connection Manager Example

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

import os

import adafruit_connection_manager
import board
import busio
from adafruit_esp32spi import adafruit_esp32spi
from digitalio import DigitalInOut

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
radio = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not radio.is_connected:
    try:
        radio.connect_AP(ssid, password)
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(radio.ap_info.ssid, "utf-8"), "\tRSSI:",
radio.ap_info.rssi)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)

JSON_GET_URL = "https://httpbin.org/get"

# Define a custom header as a dict.
headers = {"user-agent": "blinka/1.0.0"}

print("Fetching JSON data from %s..." % JSON_GET_URL)
with requests.get(JSON_GET_URL, headers=headers) as response:
    print("-" * 60)

    json_data = response.json()
    headers = json_data["headers"]
    print("Response's Custom User-Agent Header: {0}".format(headers["User-Agent"]))
```

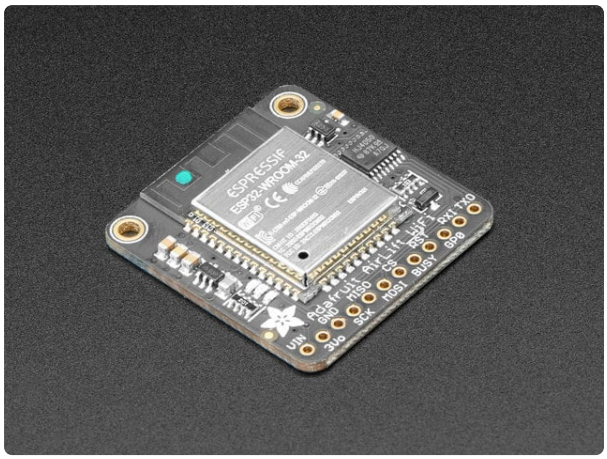
```
    print("-" * 60)

    # Read Response's HTTP status code
    print("Response HTTP Status Code: ", response.status_code)
    print("-" * 60)
```

Requests and Connection Manager Example

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

import os

import adafruit_connection_manager
import board
import busio
from adafruit_esp32spi import adafruit_esp32spi
from digitalio import DigitalInOut

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
radio = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not radio.is_connected:
    try:
        radio.connect_AP(ssid, password)
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(radio.ap_info.ssid, "utf-8"), "\tRSSI:",
radio.ap_info.rssi)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "https://httpbin.org/get"
JSON_POST_URL = "https://httpbin.org/post"

print("Fetching text from %s" % TEXT_URL)
with requests.get(TEXT_URL) as response:
    print("-" * 40)
    print("Text Response: ", response.text)
    print("-" * 40)
```

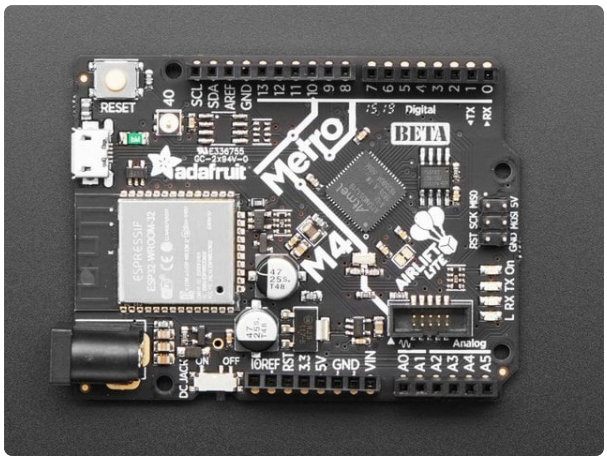
```
print("Fetching JSON data from %s" % JSON_GET_URL)
with requests.get(JSON_GET_URL) as response:
    print("-" * 40)
    print("JSON Response: ", response.json())
    print("-" * 40)

data = "31F"
print(f"POSTing data to {JSON_POST_URL}: {data}")
with requests.post(JSON_POST_URL, data=data) as response:
    print("-" * 40)
    json_resp = response.json()
    # Parse out the 'data' key from json_resp dict.
    print("Data received from server:", json_resp["data"])
    print("-" * 40)

json_data = {"Date": "July 25, 2019"}
print(f"POSTing data to {JSON_POST_URL}: {json_data}")
with requests.post(JSON_POST_URL, json=json_data) as response:
    print("-" * 40)
    json_resp = response.json()
    # Parse out the 'json' key from json_resp dict.
    print("JSON Data received from server:", json_resp["json"])
    print("-" * 40)
```

Companion Guides



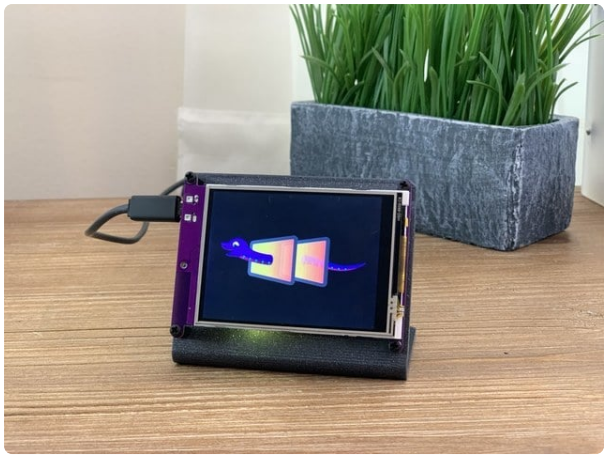
Adafruit AirLift - ESP32 WiFi Co-Processor Breakout
By Kattni Rembor
Overview

https://learn.adafruit.com/adafruit-airlift-breakout/overview



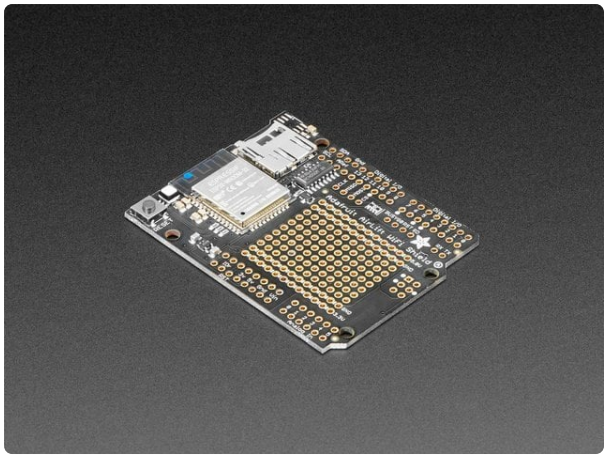
Adafruit Metro M4 Express AirLift (WiFi)
By Brent Rubell
Overview

https://learn.adafruit.com/adafruit-metro-m4-express-airlift-wifi/overview



Adafruit PyPortal - IoT for CircuitPython
By Kattni Rembor
Overview

https://learn.adafruit.com/adafruit-pyportal/overview



Adafruit AirLift Shield - ESP32 WiFi Co-Processor
By Brent Rubell
Overview

https://learn.adafruit.com/adafruit-airlift-shield-esp32-wifi-co-processor/overview

Resources

ReadTheDocs

- `adafruit_esp32spi` Module (https://adafru.it/1a5J)
- `adafruit_esp32spi` examples (https://adafru.it/1a5K)

Networking with WizNet Ethernet

Wireless is wonderful, but sometimes you want the strong reliability of a wired connection. If your project is going to be part of a permanent installation, you may want to add Ethernet wired networking to your project.

Ethernet is incredibly easy to use - there's no network configuration or device pairing. Just plug a standard Ethernet cable into an Ethernet FeatherWing or Ethernet Shield and use the CircuitPython Wiznet5k (https://adafru.it/JBC) library for quick and reliable networking.

Setup

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Egk) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx) matching your version of CircuitPython. The Wiznet5k Library (https://adafru.it/JxE) requires at least CircuitPython version 4.0.0. The latest version (https://adafru.it/Em8) is recommended.

Before continuing, make sure your board's **lib** folder has at least the following files and folders copied over:

- **adafruit_wiznet5k**
- **adafruit_bus_device**
- **adafruit_requests.mpy**
- **adafruit_connection_manager**

Requests and Connection Manager Example

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

import adafruit_connection_manager
import adafruit_requests
import board
import busio
import digitalio

from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K

print("Wiznet5k WebClient Test")

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_URL = "http://api.coindesk.com/v1/bpi/currentprice/USD.json"

# For Adafruit Ethernet FeatherWing
cs = digitalio.DigitalInOut(board.D10)
# For Particle Ethernet FeatherWing
# cs = digitalio.DigitalInOut(board.D5)
spi_bus = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# Initialize ethernet interface with DHCP
eth = WIZNET5K(spi_bus, cs)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(eth)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(eth)
requests = adafruit_requests.Session(pool, ssl_context)

print("Chip Version:", eth.chip)
print("MAC Address:", [hex(i) for i in eth.mac_address])
print("My IP address is:", eth.pretty_ip(eth.ip_address))
print("IP lookup adafruit.com: %s" %
eth.pretty_ip(eth.get_host_by_name("adafruit.com")))
```

```
# eth._debug = True
print("Fetching text from", TEXT_URL)
r = requests.get(TEXT_URL)
print("-" * 40)
print(r.text)
print("-" * 40)
r.close()

print()
print("Fetching json from", JSON_URL)
r = requests.get(JSON_URL)
print("-" * 40)
print(r.json())
print("-" * 40)
r.close()

print("Done!")
```

Simple Server Example

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-FileCopyrightText: 2021 Adam Cummick
#
# SPDX-License-Identifier: MIT

import board
import busio
import digitalio

import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K

print("Wiznet5k SimpleServer Test")

# For Adafruit Ethernet FeatherWing
cs = digitalio.DigitalInOut(board.D10)
# For Particle Ethernet FeatherWing
# cs = digitalio.DigitalInOut(board.D5)
spi_bus = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# Initialize ethernet interface
eth = WIZNET5K(spi_bus, cs, is_dhcp=True)

# Initialize a socket for our server
pool = socketpool.SocketPool(eth)
server = pool.socket()  # Allocate socket for the server
server_ip = eth.pretty_ip(eth.ip_address)  # IP address of server
server_port = 50007  # Port to listen on
server.bind((server_ip, server_port))  # Bind to IP and Port
server.listen()  # Begin listening for incoming clients

while True:
    print(f"Accepting connections on {server_ip}:{server_port}")
    conn, addr = server.accept()  # Wait for a connection from a client.
    print(f"Connection accepted from {addr}, reading exactly 1024 bytes from
client")
    with conn:
        data = conn.recv(1024)
        if data:  # Wait for receiving data
            print(data)
            conn.send(data)  # Echo message back to client
    print("Connection closed")
```

`adafruit_httpserver` Example

```
# SPDX-FileCopyrightText: 2023 Tim C for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import digitalio
from adafruit_httpserver import Request, Response, Server

import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K

print("Wiznet5k HTTPServer Test")

# For Adafruit Ethernet FeatherWing
cs = digitalio.DigitalInOut(board.D10)
# For Particle Ethernet FeatherWing
# cs = digitalio.DigitalInOut(board.D5)
spi_bus = board.SPI()

# Initialize ethernet interface with DHCP
eth = WIZNET5K(spi_bus, cs)

# Create a socket pool
pool = socketpool.SocketPool(eth)

# initialize the server
server = Server(pool, "/static", debug=True)


@server.route("/")
def base(request: Request):
    """
    Serve a default static plain text message.
    """
    return Response(request, "Hello from the CircuitPython HTTP Server!")


server.serve_forever(str(eth.pretty_ip(eth.ip_address)))
```

Network Time Protocol (NTP) Example

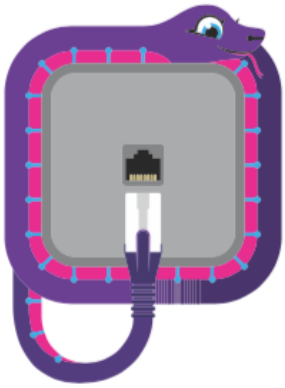
The following code looks for a WiFi capable chip. If it doesn't find one, it looks for the WizNet 5k library and sets up the microcontroller for an Ethernet SPI connection.

This code uses the `adafruit_ntp` and `adafruit_connection_manager` modules.

```
# SPDX-FileCopyrightText: 2024 Justin Myers for Adafruit Industries
# SPDX-FileCopyrightText: 2024 anecdata for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

"""Print out time based on NTP, using connection manager"""

import adafruit_connection_manager

import adafruit_ntp

# determine which radio is available
try:
    import os

    import wifi
```

```
    # adjust method to get credentials as necessary...
    wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
    wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
    radio = wifi.radio
    while not radio.connected:
        radio.connect(wifi_ssid, wifi_password)
except ImportError:
    import board
    from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
    from digitalio import DigitalInOut

    # adjust with busio.SPI() as necessary...
    spi = board.SPI()
    # adjust pin for the specific board...
    eth_cs = DigitalInOut(board.D10)
    radio = WIZNET5K(spi, eth_cs)

# get the socket pool from connection manager
socket = adafruit_connection_manager.get_radio_socketpool(radio)

# adjust tz_offset for locale, only ping NTP server every hour
ntp = adafruit_ntp.NTP(socket, tz_offset=-5, cache_seconds=3600)

print(ntp.datetime)
```

Companion Guide



Ethernet for CircuitPython with Wiznet5K
By Brent Rubell
Overview

https://learn.adafruit.com/ethernet-for-circuitpython/overview

Resources

- Ethernet for CircuitPython with Wiznet5K Guide (https://adafru.it/1a5L)
- `Adafruit_CircuitPython_Wiznet5k` (https://adafru.it/1a5M) examples on GitHub
- Adafruit Connection Manager Adafruit Playground Note (https://adafru.it/1a5q)

Making HTTP and HTTPS Requests

The `requests` module allows you to send HTTP requests using regular Python on the desktop. The `adafruit_requests` library does the same in CircuitPython.

The HTTP request returns a `Response` object with all the response data (content, encoding, status, etc).

A Simple Example Using `wifi`

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
# Updated for Circuit Python 9.0
"""WiFi Simpletest"""

import os

import adafruit_connection_manager
import wifi

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "https://httpbin.org/get"
JSON_POST_URL = "https://httpbin.org/post"

# Initalize Wifi, Socket Pool, Request Session
pool = adafruit_connection_manager.get_radio_socketpool(wifi.radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(wifi.radio)
requests = adafruit_requests.Session(pool, ssl_context)
rssi = wifi.radio.ap_info.rssi

print(f"\nConnecting to {ssid}...")
print(f"Signal Strength: {rssi}")
try:
    # Connect to the Wi-Fi network
    wifi.radio.connect(ssid, password)
except OSError as e:
    print(f"  OSError: {e}")
print("  Wifi!")

print(f" | GET Text Test: {TEXT_URL}")
with requests.get(TEXT_URL) as response:
    print(f" |   GET Response: {response.text}")
print("-" * 80)

print(f" | GET Full Response Test: {JSON_GET_URL}")
with requests.get(JSON_GET_URL) as response:
    print(f" |   Unparsed Full JSON Response: {response.json()}")
print("-" * 80)

DATA = "This is an example of a JSON value"
print(f" |   JSON 'value' POST Test: {JSON_POST_URL} {DATA}")
with requests.post(JSON_POST_URL, data=DATA) as response:
    json_resp = response.json()
    # Parse out the 'data' key from json_resp dict.
    print(f" |   JSON 'value' Response: {json_resp['data']}")
print("-" * 80)

json_data = {"Date": "January 1, 1970"}
print(f" |   JSON 'key':'value' POST Test: {JSON_POST_URL} {json_data}")
with requests.post(JSON_POST_URL, json=json_data) as response:
    json_resp = response.json()
    # Parse out the 'json' key from json_resp dict.
    print(f" |   JSON 'key':'value' Response: {json_resp['json']}")
print("-" * 80)
```

```
print("Finished!")
```

Advanced `wifi` Example

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
# Updated for Circuit Python 9.0

"""WiFi Advanced Example"""

import os

import adafruit_connection_manager
import wifi

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# Initalize Wifi, Socket Pool, Request Session
pool = adafruit_connection_manager.get_radio_socketpool(wifi.radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(wifi.radio)
requests = adafruit_requests.Session(pool, ssl_context)
rssi = wifi.radio.ap_info.rssi

# URL for GET request
JSON_GET_URL = "https://httpbin.org/get"
# Define a custom header as a dict.
headers = {"user-agent": "blinka/1.0.0"}

print(f"\nConnecting to {ssid}...")
print(f"Signal Strength: {rssi}")
try:
    # Connect to the Wi-Fi network
    wifi.radio.connect(ssid, password)
except OSError as e:
    print(f"  OSError: {e}")
print("  Wifi!")

# Define a custom header as a dict.
headers = {"user-agent": "blinka/1.0.0"}
print(f" | Fetching URL {JSON_GET_URL}")

# Use with statement for retreiving GET request data
with requests.get(JSON_GET_URL, headers=headers) as response:
    json_data = response.json()
    headers = json_data["headers"]
    content_type = response.headers.get("content-type", "")
    date = response.headers.get("date", "")
    if response.status_code == 200:
        print(f" |   Status Code: {response.status_code}")
    else:
        print(f" |   Status Code: {response.status_code}")
    print(f" |  | Custom User-Agent Header: {headers['User-Agent']}")
    print(f" |  | Content-Type: {content_type}")
    print(f" |  | Response Timestamp: {date}")
```

Simple Example for Airlift / ESP32SPI

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

```
import os

import adafruit_connection_manager
import board
import busio
from adafruit_esp32spi import adafruit_esp32spi
from digitalio import DigitalInOut

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
radio = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not radio.is_connected:
    try:
        radio.connect_AP(ssid, password)
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(radio.ap_info.ssid, "utf-8"), "\tRSSI:",
radio.ap_info.rssi)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "https://httpbin.org/get"
JSON_POST_URL = "https://httpbin.org/post"

print("Fetching text from %s" % TEXT_URL)
with requests.get(TEXT_URL) as response:
    print("-" * 40)
    print("Text Response: ", response.text)
    print("-" * 40)

print("Fetching JSON data from %s" % JSON_GET_URL)
with requests.get(JSON_GET_URL) as response:
    print("-" * 40)
    print("JSON Response: ", response.json())
    print("-" * 40)

data = "31F"
print(f"POSTing data to {JSON_POST_URL}: {data}")
with requests.post(JSON_POST_URL, data=data) as response:
    print("-" * 40)
    json_resp = response.json()
```

```
    # Parse out the 'data' key from json_resp dict.
    print("Data received from server:", json_resp["data"])
    print("-" * 40)

json_data = {"Date": "July 25, 2019"}
print(f"POSTing data to {JSON_POST_URL}: {json_data}")
with requests.post(JSON_POST_URL, json=json_data) as response:
    print("-" * 40)
    json_resp = response.json()
    # Parse out the 'json' key from json_resp dict.
    print("JSON Data received from server:", json_resp["json"])
    print("-" * 40)
```

Using Wiznet5k Example

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

import adafruit_connection_manager
import board
import busio
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
from digitalio import DigitalInOut

import adafruit_requests

cs = DigitalInOut(board.D10)
spi_bus = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# Initialize ethernet interface with DHCP
radio = WIZNET5K(spi_bus, cs)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "http://httpbin.org/get"
JSON_POST_URL = "http://httpbin.org/post"

print("Fetching text from %s" % TEXT_URL)
with requests.get(TEXT_URL) as response:
    print("-" * 40)
    print("Text Response: ", response.text)
    print("-" * 40)

print("Fetching JSON data from %s" % JSON_GET_URL)
with requests.get(JSON_GET_URL) as response:
    print("-" * 40)
    print("JSON Response: ", response.json())
    print("-" * 40)

data = "31F"
print(f"POSTing data to {JSON_POST_URL}: {data}")
with requests.post(JSON_POST_URL, data=data) as response:
    print("-" * 40)
    json_resp = response.json()
    # Parse out the 'data' key from json_resp dict.
    print("Data received from server:", json_resp["data"])
    print("-" * 40)

json_data = {"Date": "July 25, 2019"}
print(f"POSTing data to {JSON_POST_URL}: {json_data}")
with requests.post(JSON_POST_URL, json=json_data) as response:
    print("-" * 40)
    json_resp = response.json()
```

```
    # Parse out the 'json' key from json_resp dict.
    print("JSON Data received from server:", json_resp["json"])
    print("-" * 40)
```

Resources

ReadTheDocs

- `adafruit_requests` (https://adafru.it/1a5N)

Examples

- `Adafruit_CircuitPython_Requests` GitHub repo Examples (https://adafru.it/1a5O)

HTTP Server Examples

Using `wifi` with `adafruit_httpserver`

```
# SPDX-FileCopyrightText: 2023 Michał Pokusa
#
# SPDX-License-Identifier: Unlicense

from asyncio import create_task, gather, run
from asyncio import sleep as async_sleep

import board
import microcontroller
import neopixel
import socketpool
import wifi

from adafruit_httpserver import GET, Request, Response, Server, Websocket

pool = socketpool.SocketPool(wifi.radio)
server = Server(pool, debug=True)

pixel = neopixel.NeoPixel(board.NEOPIXEL, 1)

websocket: Websocket = None

HTML_TEMPLATE = """
<html lang="en">
    <head>
        <title>Websocket Client</title>
    </head>
    <body>
        <p>CPU temperature: <strong>-</strong>&deg;C</p>
        <p>NeoPixel Color: <input type="color"></p>
        <script>
            const cpuTemp = document.querySelector('strong');
            const colorPicker = document.querySelector('input[type="color"]');

            let ws = new WebSocket('ws://' + location.host + '/connect-websocket');

            ws.onopen = () => console.log('WebSocket connection opened');
            ws.onclose = () => console.log('WebSocket connection closed');
```

```
            ws.onmessage = event => cpuTemp.textContent = event.data;
            ws.onerror = error => cpuTemp.textContent = error;

            colorPicker.oninput = debounce(() => ws.send(colorPicker.value), 200);

            function debounce(callback, delay = 1000) {
                let timeout
                return (...args) => {
                    clearTimeout(timeout)
                    timeout = setTimeout(() => {
                    callback(...args)
                  }, delay)
                }
            }
        </script>
    </body>
</html>
"""


@server.route("/client", GET)
def client(request: Request):
    return Response(request, HTML_TEMPLATE, content_type="text/html")


@server.route("/connect-websocket", GET)
def connect_client(request: Request):
    global websocket

    if websocket is not None:
        websocket.close()  # Close any existing connection

    websocket = Websocket(request)

    return websocket


server.start(str(wifi.radio.ipv4_address))


async def handle_http_requests():
    while True:
        server.poll()

        await async_sleep(0)


async def handle_websocket_requests():
    while True:
        if websocket is not None:
            if (data := websocket.receive(fail_silently=True)) is not None:
                r, g, b = int(data[1:3], 16), int(data[3:5], 16), int(data[5:7], 16)
                pixel.fill((r, g, b))

        await async_sleep(0)


async def send_websocket_messages():
    while True:
        if websocket is not None:
            cpu_temp = round(microcontroller.cpu.temperature, 2)
            websocket.send_message(str(cpu_temp), fail_silently=True)

        await async_sleep(1)


async def main():
    await gather(
        create_task(handle_http_requests()),
```

```
        create_task(handle_websocket_requests()),
        create_task(send_websocket_messages()),
    )


run(main())
```

Return CPU Information Example

```
# SPDX-FileCopyrightText: 2022 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import microcontroller
import socketpool
import wifi

from adafruit_httpserver import JSONResponse, Request, Server

pool = socketpool.SocketPool(wifi.radio)
server = Server(pool, debug=True)

# (Optional) Allow cross-origin requests.
server.headers = {
    "Access-Control-Allow-Origin": "*",
}


@server.route("/cpu-information", append_slash=True)
def cpu_information_handler(request: Request):
    """
    Return the current CPU temperature, frequency, and voltage as JSON.
    """

    data = {
        "temperature": microcontroller.cpu.temperature,
        "frequency": microcontroller.cpu.frequency,
        "voltage": microcontroller.cpu.voltage,
    }

    return JSONResponse(request, data)


server.serve_forever(str(wifi.radio.ipv4_address))
```

Simple Example with Requests

```
# SPDX-FileCopyrightText: 2024 DJDevon3
#
# SPDX-License-Identifier: MIT

import wifi
from adafruit_connection_manager import get_radio_socketpool

from adafruit_httpserver import Request, Response, Server

pool = get_radio_socketpool(wifi.radio)
server = Server(pool, "/static", debug=True)


@server.route("/")
def base(request: Request):
    """
    Serve a default static plain text message.
    """
```

```
    return Response(request, "Hello from the CircuitPython HTTP Server!")


server.serve_forever(str(wifi.radio.ipv4_address))
```

Example for Wiznet5K

```
# SPDX-FileCopyrightText: 2023 Tim C for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import digitalio
from adafruit_httpserver import Request, Response, Server

import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K

print("Wiznet5k HTTPServer Test")

# For Adafruit Ethernet FeatherWing
cs = digitalio.DigitalInOut(board.D10)
# For Particle Ethernet FeatherWing
# cs = digitalio.DigitalInOut(board.D5)
spi_bus = board.SPI()

# Initialize ethernet interface with DHCP
eth = WIZNET5K(spi_bus, cs)

# Create a socket pool
pool = socketpool.SocketPool(eth)

# initialize the server
server = Server(pool, "/static", debug=True)


@server.route("/")
def base(request: Request):
    """
    Serve a default static plain text message.
    """
    return Response(request, "Hello from the CircuitPython HTTP Server!")


server.serve_forever(str(eth.pretty_ip(eth.ip_address)))
```

Resources

ReadTheDocs

- Adafruit CircuitPython HTTPServer Library Examples (https://adafru.it/1a5P)

Additional Examples

- `Adafruit_CircuitPython_HTTPServer` examples on GitHub (https://adafru.it/1a5Q)

NTP Time Example

Network Time Protocol allows for getting the time from specific time servers on a local network or internet.

Example CircuitPython Code

The example code below assumes you have a **settings.toml** file in the **CIRCUITPY** root directory which contains the SSID and password for the local WiFi network, as discussed earlier in this guide.

```
# SPDX-FileCopyrightText: 2022 Scott Shawcroft for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Print out time based on NTP."""

import os
import time

import socketpool
import wifi

import adafruit_ntp

# Get wifi AP credentials from a settings.toml file
wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
if wifi_ssid is None:
    print("WiFi credentials are kept in settings.toml, please add them there!")
    raise ValueError("SSID not found in environment variables")

try:
    wifi.radio.connect(wifi_ssid, wifi_password)
except ConnectionError:
    print("Failed to connect to WiFi with provided credentials")
    raise

pool = socketpool.SocketPool(wifi.radio)
ntp = adafruit_ntp.NTP(pool, tz_offset=0, cache_seconds=3600)

while True:
    print(ntp.datetime)
    time.sleep(1)
```

Example for Wiznet5k

The following code looks for a WiFi capable chip. It it doesn't find one, it looks for the WizNet 5k library and sets up the microcontroller to Ethernet SPI connection.

This code uses the `adafruit_ntp` and `adafruit_connection_manager` modules.

```
# SPDX-FileCopyrightText: 2024 Justin Myers for Adafruit Industries
# SPDX-FileCopyrightText: 2024 anecdata for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

"""Print out time based on NTP, using connection manager"""
```

```
import adafruit_connection_manager

import adafruit_ntp

# determine which radio is available
try:
    import os

    import wifi

    # adjust method to get credentials as necessary...
    wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
    wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
    radio = wifi.radio
    while not radio.connected:
        radio.connect(wifi_ssid, wifi_password)
except ImportError:
    import board
    from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
    from digitalio import DigitalInOut

    # adjust with busio.SPI() as necessary...
    spi = board.SPI()
    # adjust pin for the specific board...
    eth_cs = DigitalInOut(board.D10)
    radio = WIZNET5K(spi, eth_cs)

# get the socket pool from connection manager
socket = adafruit_connection_manager.get_radio_socketpool(radio)

# adjust tz_offset for locale, only ping NTP server every hour
ntp = adafruit_ntp.NTP(socket, tz_offset=-5, cache_seconds=3600)

print(ntp.datetime)
```

Resources

ReadTheDocs

- adafruit_ntp documentation (https://adafru.it/1a5R)

Examples

- Adafruit_CircuitPython_NTP repository (https://adafru.it/106c)
  - Examples (https://adafru.it/1a5S)

Troubleshooting

Here are some issues and solutions regarding networking:

General

Can I put the network credentials in my code instead of in a **settings.toml** file?

Yes, of course. Code like `wifi.radio.connect(ssid="mynetwork", password="12345")` is valid, but is highly discouraged.

If you save the code to GitHub or another online repository or publish a guide or Playground Note, you will be handing your WiFi credentials to the world.

Placing the values in **settings.toml** and using `os.getenv()` allows you to separate the values from the code.

Can CircuitPython use IPv6 addressing?

There is active work on the ability to use IPv6 addresses in addition to IPv4. The work is incomplete as of mid-2024. Please keep an eye on CircuitPython version release notes for when the work will be complete. There is no estimated time of arrival (ETA) for this code to be ready.

Wireless Networking

My device fails to connect or `os.getenv()` returns an error

Current CircuitPython implementations use a file called **settings.toml** to store the WiFi SSID ("network name") and password. See the Network Settings page in this guide on how to create this file and format content in it. The values must be in double quotes ("). Typical entries are similar to the ones below:

```
# settings.toml
CIRCUITPY_WIFI_SSID = "MyLocalNet"
CIRCUITPY_WIFI_PASSWORD = "mysecretpassword"
```

Also note: the values in **settings.toml** must match those in the code. In **settings.toml** if the SSID is specified in the value `CIRCUITPY_WIFI_SSID` then use

`ssid=os.getenv("CIRCUITPY_WIFI_SSID")` to get that value and not `ssid=os.getenv("SSID")` as there is a mismatch in the names which will not result in what you want.

How do I get the IP address for a board on my local network?

You can print it using `print("my IP addr:", wifi.radio.ipv4_address)` or save it to a variable with `my_address = wifi.radio.ipv4_address`. Generally the libraries abstract the address such that you do not have to use its value explicitly. But there are times when it is handy to print it to verify there is a connection, re. if the value is `0.0.0.0` or another nonsense value when it should be similar to other devices on your network (example: `192.168.1.87`) then there may be an issue.

Wired Networking

Does CircuitPython support any hardwired networking other than WizNet 55xx?

Not at this time. This can be revisited as new technologies come on the scene and are adopted by the community.

If you are using a single board computer like Raspberry Pi, you'll be using CPython ("regular Python") which has extensive networking support which you can find documentation in standard Python texts.

Advanced Topics: Ping and UDP

Ping

Ping is a method of measuring the round trip time for messages sent from a host to a network destination and echoed back to the source on an IP network. Pinging involves sending an ICMP echo request to the target host and waiting for an ICMP echo reply.

In CircuitPython, the wifi Module's radio function provides ping functionality. `ping = wifi.radio.ping(ip=ping_ip)` where ping is the round trip time in seconds a request took place.

> Limitations: On Espressif, calling `ping()` multiple times rapidly exhausts available resources after several calls. Rather than failing at that point, `ping()` will wait two seconds for enough resources to be freed up before proceeding.

The following shows how a ping can be sent and printed out.

```
import os
import ipaddress
import ssl
import wifi

print(f"Connecting to {os.getenv('CIRCUITPY_WIFI_SSID')}")
wifi.radio.connect(os.getenv("CIRCUITPY_WIFI_SSID"),
os.getenv("CIRCUITPY_WIFI_PASSWORD"))
print(f"Connected to {os.getenv('CIRCUITPY_WIFI_SSID')}")
print(f"My IP address: {wifi.radio.ipv4_address}")

ping_ip = ipaddress.IPv4Address("8.8.8.8")  # Google.com
ping = wifi.radio.ping(ip=ping_ip)

# retry once if timed out
if ping is None:
    ping = wifi.radio.ping(ip=ping_ip)

if ping is None:
    print("Couldn't ping 'google.com' successfully")
else:
    # convert s to ms
    print(f"Pinging 'google.com' took: {ping * 1000} ms")
```

UDP

TCP (Transmission Control Protocol), used for HTTP, HTTPS, and many other kinds of internet connections, is not the only method of transmitting data over a network connection. TCP is a "reliable delivery" protocol: it keeps trying until the data is delivered or gives up with an error. It sets up a persistent connection between two points and uses a sequence of protocol acknowledgments and "handshakes" to ensure reliable delivery.

By contrast, UDP (User Datagram Protocol) is a "connectionless" protocol. It simply tries to deliver packets of data from one point to another. The packets may be dropped along the way if there is congestion or other problems, and their ordering is not guaranteed. It is useful for streaming data such audio, video, or periodic data reporting, where loss of data is not fatal or corrupting.

In general, you request a socket supporting UDP with the following:

```
wifi.radio.connect(ssid=os.getenv("CIRCUITPY_WIFI_SSID"),
password=os.getenv("CIRCUITPY_WIFI_PASSWORD"))
pool = socketpool.SocketPool(wifi.radio)
sock = pool.socket(pool.AF_INET, pool.SOCK_DGRAM)
```

You'd then create your packet to send (a `bytearray`) and send it like this:

```
sock.sendto(packet, (URL, port))
```

> The port used for UDP for your own use should be 1024 or greater, as lower ports (0-1023) are typically reserved for specific services and protected by the operating system.

Below is an example of using UDP for getting the time from an NTP server:

```
import wifi
import socketpool
import struct
import time

# connect to wifi
print("Connecting to Wifi")
wifi.radio.connect("mySSID", "myPASS")
pool = socketpool.SocketPool(wifi.radio)

# make socket
print("Creating socket")
sock = pool.socket(pool.AF_INET, pool.SOCK_DGRAM)

# Fill packet
packet = bytearray(48)
packet[0] = 0b00100011  # Not leap second, NTP version 4, Client mode
NTP_TO_UNIX_EPOCH = 2208988800  # 1970-01-01 00:00:00

print("Sending packet")
sock.sendto(packet, ("pool.ntp.org", 123))

size, address = sock.recvfrom_into(packet)
print("Received packet")

seconds = struct.unpack_from("!I", packet, offset=len(packet) - 8)[0]
print("Address:", address)
print("Time:", time.localtime(seconds - NTP_TO_UNIX_EPOCH))
```

Below is an example posted by Tod Kurt (@todbot):

```
# udp_recv_code.py -- receive UDP messages from any receiver, can be another
CircuitPython device
# 24 Aug 2022 - @todbot / Tod Kurt
# cribbing from code at https://github.com/adafruit/circuitpython/blob/main/tests/
circuitpython-manual/socketpool/datagram/ntp.py

import time, wifi, socketpool, os
print("Connecting to WiFi...")
wifi.radio.connect(ssid=os.getenv("CIRCUITPY_WIFI_SSID"),
```

```
password=os.getenv("CIRCUITPY_WIFI_PASSWORD"))
print("my IP addr:", wifi.radio.ipv4_address)
pool = socketpool.SocketPool(wifi.radio)

# I, @PaulskPt, used for udp_host erroneously: os.getenv("MULTICAST_GROUP")
udp_host = str(wifi.radio.ipv4_address) # my LAN IP as a string
udp_port = int(os.getenv("MULTICAST_PORT"))  # a number of your choosing, should be
1024-65000
udp_buffer = bytearray(64)  # stores our incoming packet

sock = pool.socket(pool.AF_INET, pool.SOCK_DGRAM) # UDP socket
sock.bind((udp_host, udp_port))  # say we want to listen on this host,port

print("waiting for packets on",udp_host, udp_port)
while True:
    size, addr = sock.recvfrom_into(udp_buffer)
    msg = udp_buffer.decode('utf-8')  # assume a string, so convert from bytearray
    print(f"Received message from {addr[0]}:", msg)
```

Resources

ReadTheDocs

- wifi Module (https://adafru.it/1a5T) (Ping)

Examples

- anecdata's Socket Examples (https://adafru.it/1a5U) - GitHub
- DJDevon's Web APIs & You (https://adafru.it/1a60) - Adafruit Playground

---

MQTT in CircuitPython

MQTT in CircuitPython (https://adafru.it/18rF)

Adafruit IO

Adafruit IO (https://adafru.it/mEi)