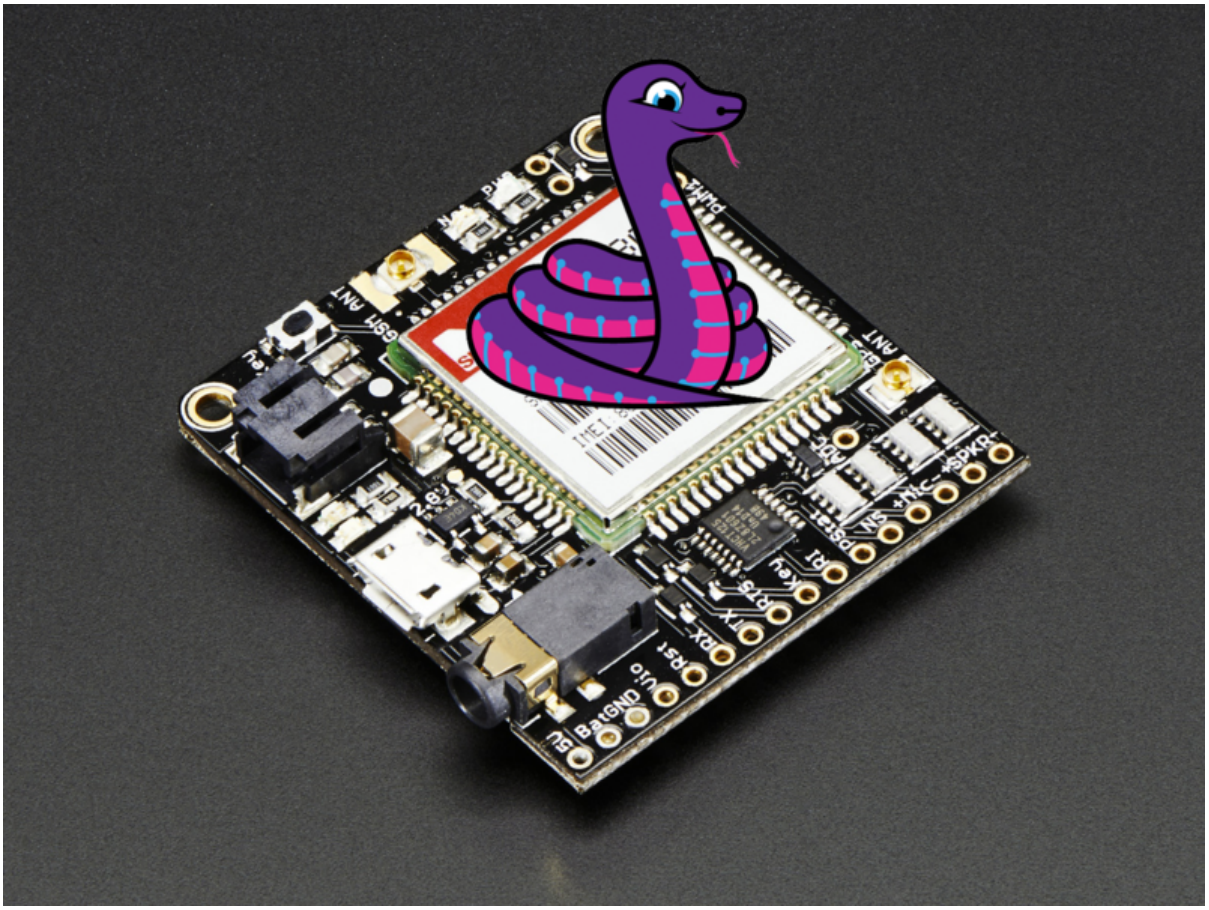
Cellular Data for CircuitPython with FONA

Created by Brent Rubell

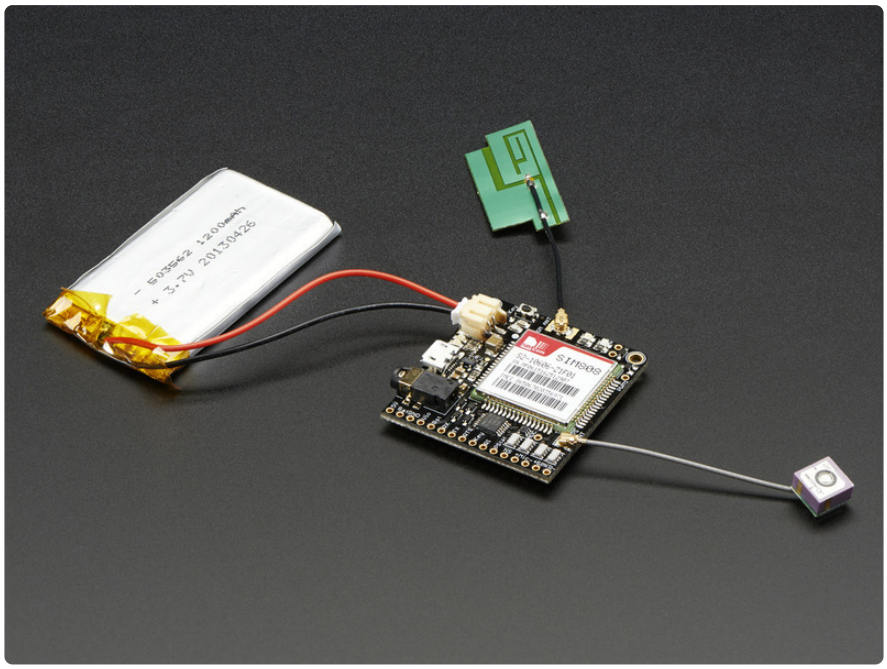


https://learn.adafruit.com/cellular-data-for-circuitpython-with-fona

Last updated on 2024-06-03 03:06:56 PM EDT

Table of Contents

Overview



Build projects that can transmit and receive data from just about anywhere using cellular data! Unlike transports that require a router, wires, or a gateway, cellular towers are located just about anywhere people are.

- **If you'd like to learn more about using cellular data in your project** - check out the cellular section in the All The Internet of Things: Transports guide (https://adafru.it/BIZ).

We designed a CircuitPython library compatible with the Adafruit FONA cellular modem. This module handles all the complicated modem interfacing for you, so you can bring your IoT projects online quickly.

In this guide, you will set up and configure a FONA module with a CircuitPython board to connect to the internet over a cellular network. We've included examples for using this module to send HTTP requests and connect to MQTT brokers.

Parts

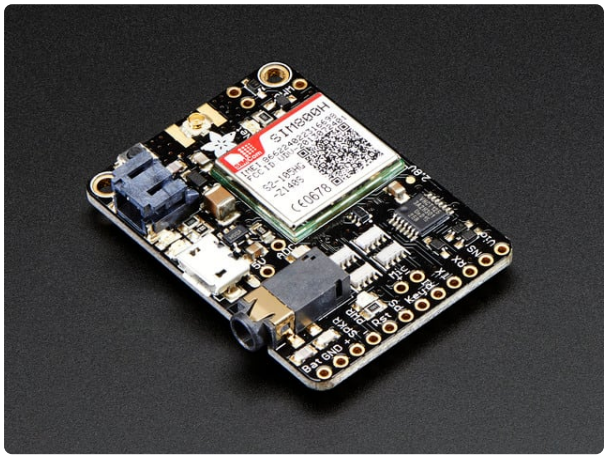
Adafruit currently offers two types of cellular data transports: 2G/GSM and 3G.

First up are the FONA808 and FONA800 modules which use **2G/GSM** networks. These networks use smaller modules, use less power and will work in any country

("quad-band"). A downside of these networks is they may be shut down in the United States by the end of 2020.

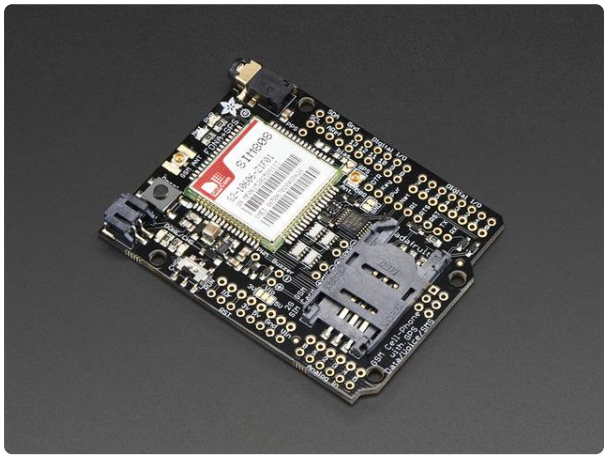
[Adafruit FONA 808 - Mini Cellular GSM + GPS Breakout]

Cellular + GPS tracking, all in one? Oh yes! Introducing Adafruit FONA 808 MiniGSM + GPS, an all-in-one cellular phone module with that lets you add location-tracking, voice, text, SMS...
https://www.adafruit.com/product/2542



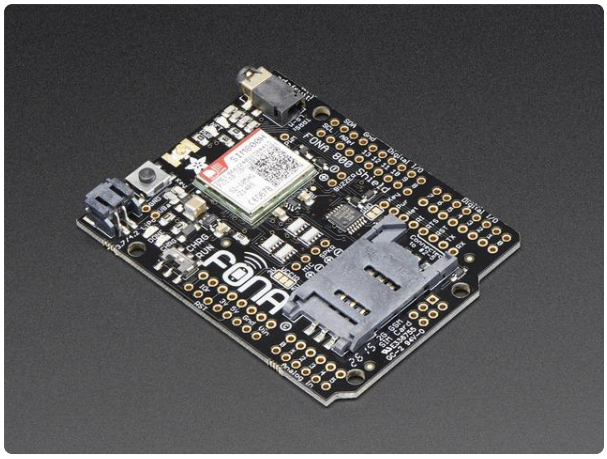
[Adafruit FONA - Mini Cellular GSM Breakout uFL Version]

Ring, Ring! Who's that callin'? It's your breadboard! Introducing Adafruit FONA MiniGSM, an adorable all-in-one cellular phone module that lets you add voice, text, SMS and...
https://www.adafruit.com/product/1946



[Adafruit FONA 808 Shield - Mini Cellular GSM + GPS for Arduino]

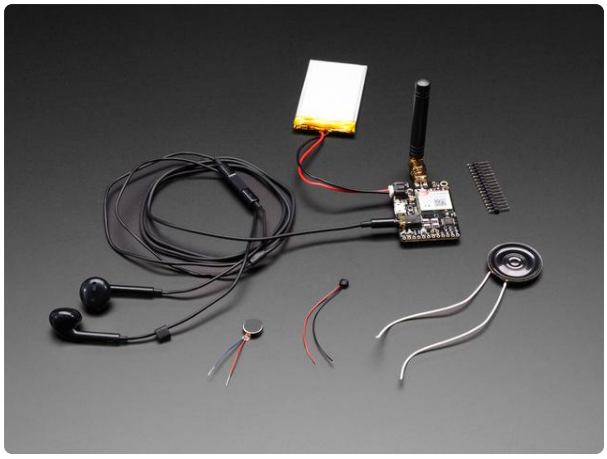
Cellular + GPS tracking, all in one, for your Arduino? Oh yes! Introducing Adafruit FONA 808 GSM + GPS Shield, an all-in-one cellular phone module with that lets you add...
https://www.adafruit.com/product/2636



[Adafruit FONA 800 Shield - Voice/Data Cellular GSM for Arduino](https://www.adafruit.com/product/2468)

Ring, Ring! Who's that callin'? It's your Arduino! Introducing Adafruit FONA 800 Shield, an adorable all-in-one cellular phone shield that lets you add voice, text, SMS and...

https://www.adafruit.com/product/2468



[Adafruit FONA 800 Breakout Board Starter Pack - SMA Version](https://www.adafruit.com/product/2522)

Build your own cellular project and get off the grid with FONA. This pack comes with an SMA-antenna type FONA and paired with hand-picked accessories. It's the Adafruit...

https://www.adafruit.com/product/2522

The **Fona3G** module uses a SIM5320 **3G** module which is physically larger than the SIM808 modules, use more power and have an increased data-rate. Unlike 2G/GSM, major carries have not announced any plan to shut down 3G services and you may need to purchase a module specific to the region where you will be operating your FONA projects.

[Adafruit FONA 3G Cellular Breakout](https://www.adafruit.com/product/2696)

For those who want to take it to the next level we now have a 3G Cellular Modem breakout! The FONA 3G has better coverage, GSM backwards-compatibility and even...

https://www.adafruit.com/product/2696

You will also need some required accessories to make the FONA work. **These are not included with the FONA shield or breakout!**

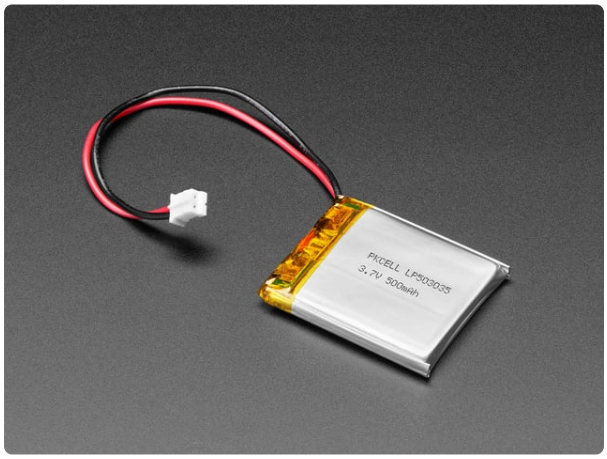
You will need a Mini SIM card to do anything on the cellular network. If you're in the USA, we suggest picking up the SIM Card from Ting.

- If you're not in the US, or want to use a different cellular network provider, please see this page for more information about obtaining a FONA-compatible SIM card. (https://adafru.it/KVE)

GSM SIM Card from Ting & Adafruit - Data/Voice/Text

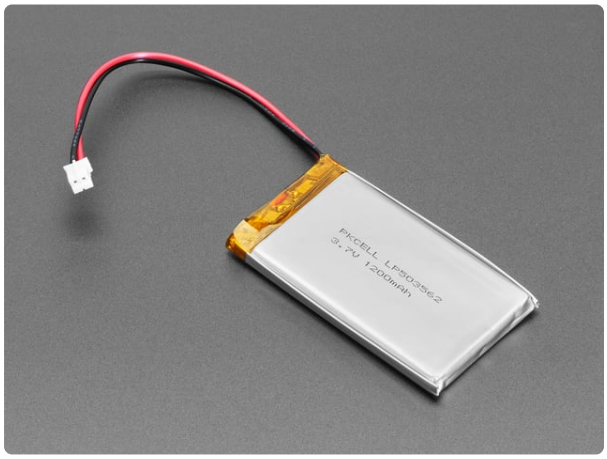
Adafruit is now a phone company :) Or, well, we've sold DIY cell phones for awhile now but you've never been...
https://www.adafruit.com/product/2505

You will need a LiPoly battery (500mAh or larger) to run the FONA module.



Lithium Ion Polymer Battery - 3.7v 500mAh

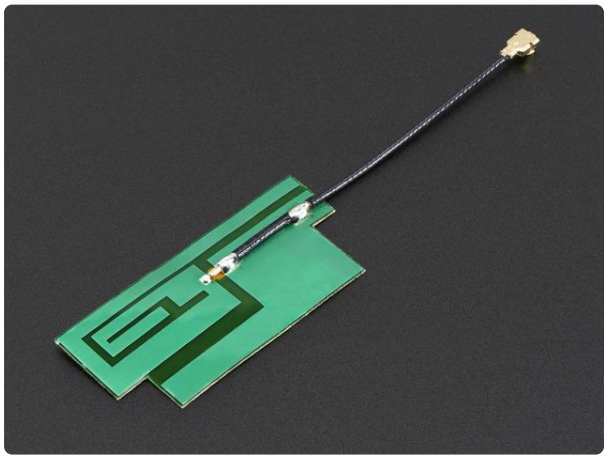
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/1578



Lithium Ion Polymer Battery - 3.7v 1200mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/258

You will need a external uFL GSM Antenna, we like this slim sticker-type antenna:



Slim Sticker-type GSM/Cellular Quad-Band Antenna - 3dBi uFL
That's one slim cellular antenna! At just 75mm long from tip to tip and and with a thickness of just 2mm, this 3dBi GSM antenna is slim, compact and sensitive, with a 3dBi...
https://www.adafruit.com/product/1991

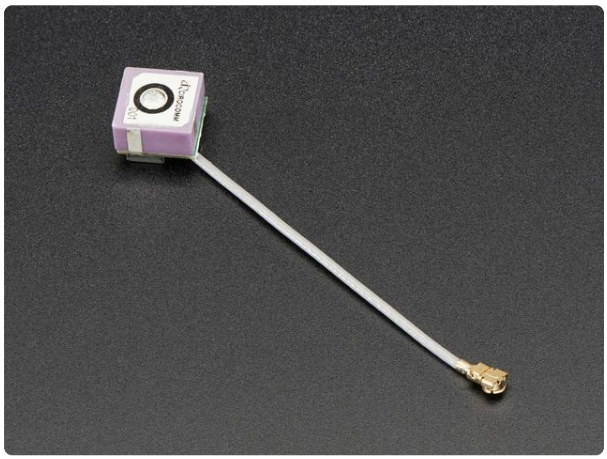
If you want to use a SMA antenna instead, you'll want to pick up a uFL to SMA adapter cable.

1 x uFL to SMA Adapter Cable https://www.adafruit.com/product/851
SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable

You will need a MicroUSB cable for charging the FONA's battery.

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

You will also need an external passive GPS antenna.



Passive GPS Antenna uFL - 9mm x 9mm -2dBi gain
Wow that's a tiny GPS antenna! This passive antenna is only 9mm x 9mm x 6.5mm in size, with a 50mm long uFL cable. Great for when you want to keep things small.Comes with a...
https://www.adafruit.com/product/2460

CircuitPython Setup

CircuitPython Installation

Some CircuitPython compatible boards come with CircuitPython installed. Others are CircuitPython-ready, but need to have it installed. As well, you may want to update the version of CircuitPython already installed on your board. The steps are the same for installing and updating.

- To **install (or update) your CircuitPython board**, follow this page and come back here when you've successfully installed (or updated) CircuitPython. (https://adafru.it/Amd)

Install the Mu Editor

This guide requires you to edit and interact with CircuitPython code. While you can use any text editor of your choosing, **Mu** is a simple code editor that works with the Adafruit CircuitPython boards. It's written in Python and works on Windows, MacOS, Linux and Raspberry Pi. The serial console is built right in, so you get immediate feedback from your board's serial output!

Before proceeding, if you'd like to use Mu, **click the button below to install the Mu Editor**. There are versions for PC, mac, and Linux.

Install Mu Editor

https://adafru.it/ANO

Secrets File Setup

Mobile devices like the Adafruit FONA need to be configured with the access point name (APN) identification with your wireless carrier.

We designed this library to use a **secrets.py** file which is in your **CIRCUITPY** drive to hold secret data including the APN configuration.

Your **secrets.py** file should look like this:

```
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    "apn": "your_apn_name",
    "apn_username": "your_apn_username",
    "apn_password": "your_apn_password",
}
```

In the **secrets.py** file, set the `apn` to the apn name provided by your cellular network carrier.

- For example, the APN for the Ting network is `wholesale`.

`apn_username` and `apn_password` are optional parameters for authentication. Leave these as-is unless otherwise directed by your network carrier

- If you are unsure about any of these settings, please contact your network carrier for more information.

CircuitPython Library Installation

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 FONA Library requires at least CircuitPython version 4.0.0.

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

- **adafruit_fona**
- **adafruit_bus_device**
- **adafruit_requests.mpy**
- **adafruit_simpleio.mpy**

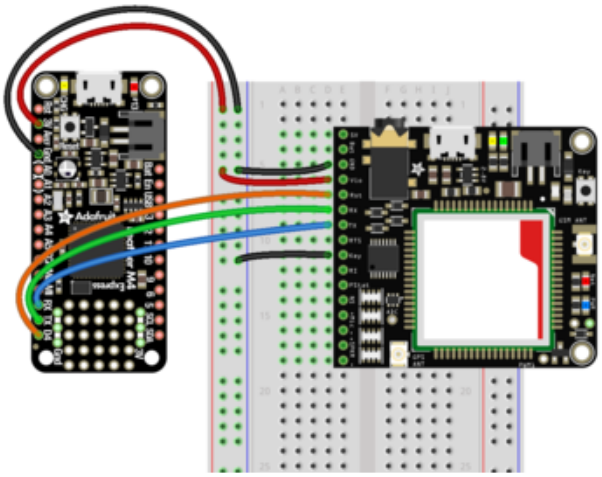
Once all the files are copied, your **CIRCUITPY** drive should look like the following screenshot:

Usage

You will want to use a chip with more memory with FONA CircuitPython such as the M4, STM32, or nRF52

FONA808 or FONA800 Wiring

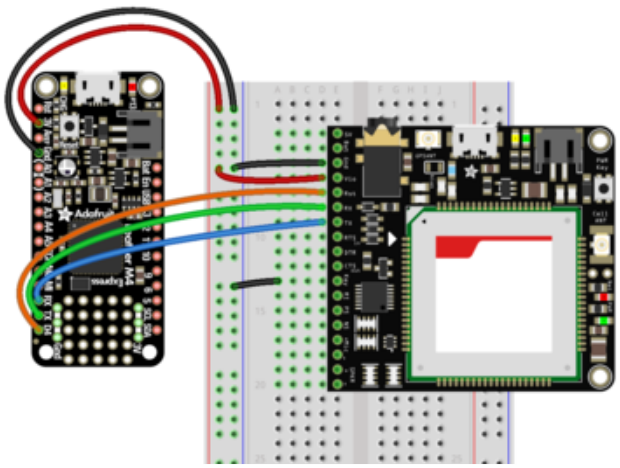
If you're using a FONA808 or FONA800 breakout, use the following wiring so that the hardware UART pins are used. Here's an example with the Feather M4:



Board 5V (or **3.3v**) to **FONA808 VIO**
Board GND to **FONA808 GND**
Board GND to **FONA808 KEY**
Board Serial TX to **FONA808 RX**
Board Serial RX to **FONA808 TX**
Board Digital Pin 4 to **FONA808 RST**

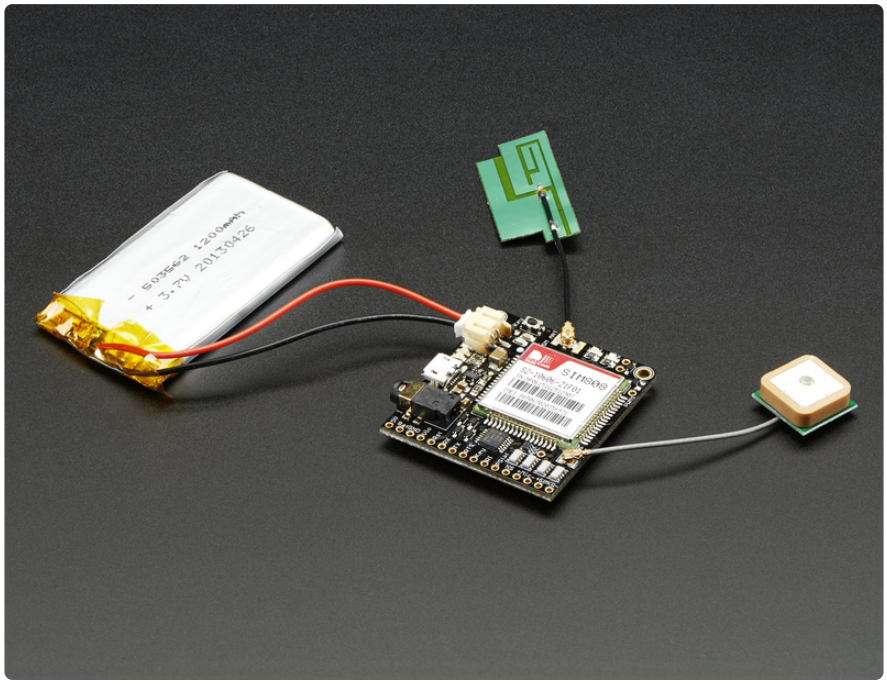
FONA 3G Wiring

If you're using a FONA3G breakout, use the following wiring so that the hardware UART pins are used. Here's an example with the Feather M4:



Board 5V (or **3.3V**) to **Fona3G Vio**
Board GND to **FONA3G GND**
Board GND to **FONA3G Key**
Board RX to **FONA3G TX**
Board TX to **FONA3G RX**
Board Digital 4 Pin to **Fona3G RST**

Assembly



Before using the FONA with CircuitPython, **make sure you've attached a LiPoly battery, GPS antenna and GSM antenna to the FONA**.

- If you haven't done this yet, navigate to this page and come back when you've connected the battery and antennas (https://adafru.it/KVF).

You must also insert a SIM card to use the FONA with cellular data.

- If you haven't done this yet, navigate to this page and come back when you've inserted the SIM card (https://adafru.it/KVF).

Make sure the battery you're using is charged before running the code below.

All of the above are REQUIRED! Flaky behavior is often a low battery, no SIM, no GSM antenna, etc!

Code Usage

Copy the following code to the **code.py** file on your microcontroller.

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

# pylint: disable=unused-import
import time
import board
import busio
import digitalio
import adafruit_connection_manager
import adafruit_requests
from adafruit_fona.adafruit_fona import FONA
from adafruit_fona.fona_3g import FONA3G
import adafruit_fona.adafruit_fona_network as network
import adafruit_fona.adafruit_fona_socket as pool

print("FONA Webclient Test")

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

# Get GPRS details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("GPRS secrets are kept in secrets.py, please add them there!")
    raise

# Create a serial connection for the FONA connection
uart = busio.UART(board.TX, board.RX)
rst = digitalio.DigitalInOut(board.D4)

# Use this for FONA800 and FONA808
fona = FONA(uart, rst)

# Use this for FONA3G
# fona = FONA3G(uart, rst)

# Initialize cellular data network
network = network.CELLULAR(
    fona, (secrets["apn"], secrets["apn_username"], secrets["apn_password"])
)

while not network.is_attached:
    print("Attaching to network...")
    time.sleep(0.5)
print("Attached!")

while not network.is_connected:
    print("Connecting to network...")
    network.connect()
    time.sleep(0.5)
```

```
print("Network Connected!")

print("My IP address is:", fona.local_ip)
print("IP lookup adafruit.com: %s" % fona.get_host_by_name("adafruit.com"))

# create requests session
ssl_context = adafruit_connection_manager.create_fake_ssl_context(pool, fona)
requests = adafruit_requests.Session(pool, ssl_context)

# fona._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!")
```

If you're using a FONA808, you do not need to modify the code below.

If you're using a FONA3G, you'll need to comment out the FONA800/FONA808 initialization and un-comment the FONA3G initialization in the code:

```
# Use this for FONA800 and FONA808
# fona = FONA(uart, rst)

# Use this for FONA3G
fona = FONA3G(uart, rst)
```

Save the code.py file and **open the REPL**.

```
code.py output:
FONA WebClient Test
Attaching to network...
Connecting to network...
Connecting to network...
My IP address is: 51.48.46.57
IP lookup adafruit.com: "104.20.38.240"
Fetching text from http://wifitest.adafruit.com/testwifi/index.html
----------------------------------------
This is a test of Adafruit WiFi!
If you can read this, its working :)

----------------------------------------

Fetching json from http://api.coindesk.com/v1/bpi/currentprice/USD.json
----------------------------------------
{'time': {'updated': 'May 7, 2020 15:49:00 UTC', 'updatedISO': '2020-05-07T15:49:00+00:00',
'updateduk': 'May 7, 2020 at 16:49 BST'}, 'disclaimer': 'This data was produced from the
CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion
rate from openexchangerates.org', 'bpi': {'USD': {'code': 'USD', 'description': 'United
States Dollar', 'rate_float': 9514.15, 'rate': '9,514.1521'}}}
----------------------------------------
Done!
```

In order, the example code:

Imports APN details from the **secrets.py** file.

```
# Get GPRS details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("GPRS secrets are kept in secrets.py, please add them there!")
    raise
```

Creates a serial UART connection for the FONA and sets up the RST pin.

NOTE: You may need to change these pins if you're using a FONA breakout instead of a FONA shield.

```
# Create a serial connection for the FONA connection
uart = busio.UART(board.TX, board.RX)
rst = digitalio.DigitalInOut(board.D4)
```

Initializes the FONA module. This step may take a few seconds since the module attempts to communicate with and bring up the module.

```
# Use this for FONA800 and FONA808
fona = FONA(uart, rst)

# Use this for FONA3G
# fona = FONA3G(uart, rst)
```

We use a module called fona_network to configure the modem. This module automatically determines if the modem connected is a 2G/GSM or 3G/CDMA modem and initializes the modem appropriately.

```
# Initialize cellular data network
network = network.CELLULAR(
    fona, (secrets["apn"], secrets["apn_username"], secrets["apn_password"])
)
```

The code waits for the FONA to attach to the cellular network using the carrier settings.

```
while not network.is_attached:
    print("Attaching to network...")
    time.sleep(0.5)
print("Attached!")
```

Once attached, it attempts to connect to the network and bring up the modem. This step may take a while to connect, depending on your cellular reception.

```
while not network.is_connected:
    print("Connecting to network...")
    network.connect()
```

```
    time.sleep(0.5)
print("Network Connected!")
```

Prints out the local IP and attempts to perform an IP address lookup for adafruit.com.

```
print("My IP address is:", fona.local_ip)
print("IP lookup adafruit.com: %s" % fona.get_host_by_name("adafruit.com"))
```

OK now we're getting to the really interesting part. With a SAMD51 or other large-RAM (well, over 32 KB) device, we can do a lot of neat tricks. Like, for example, we can implement an interface a lot like requests (https://adafru.it/E9o) - which makes getting data really really easy.

Calling `requests.set_socket` passes requests our fona interface and a socket-like implementation for the FONA.

```
# Initialize a requests object with a socket and cellular interface
requests.set_socket(cellular_socket, fona)
```

To read in all the text from a web URL, call `requests.get` -

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

Or, if the data is in structured JSON, you can get the json pre-parsed into a Python dictionary that can be easily queried or traversed. (Again, only for nRF52840, M4 and other high-RAM boards)

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

Python Docs

Python Docs (https://adafru.it/KPD)