Textable Sensor with FONA and CircuitPython

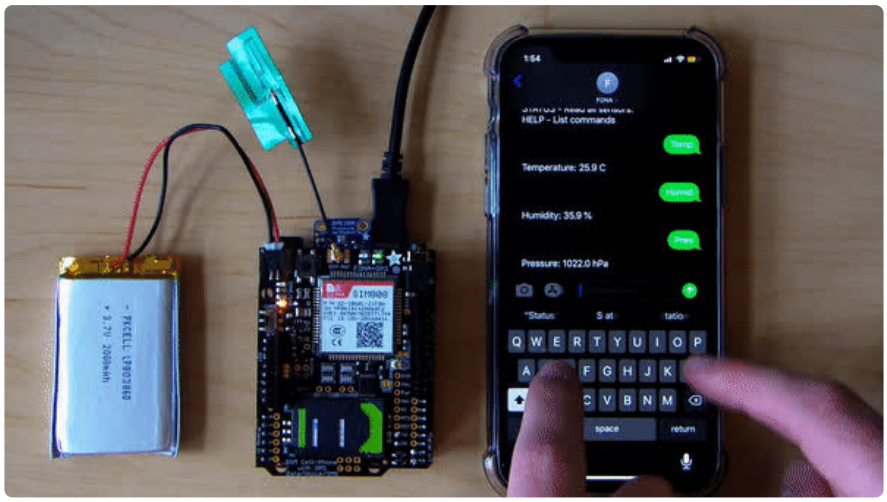
Created by Brent Rubell

https://learn.adafruit.com/textable-sensor-with-fona-and-circuitpython

Last updated on 2024-06-03 03:07:19 PM EDT

Table of Contents

Overview



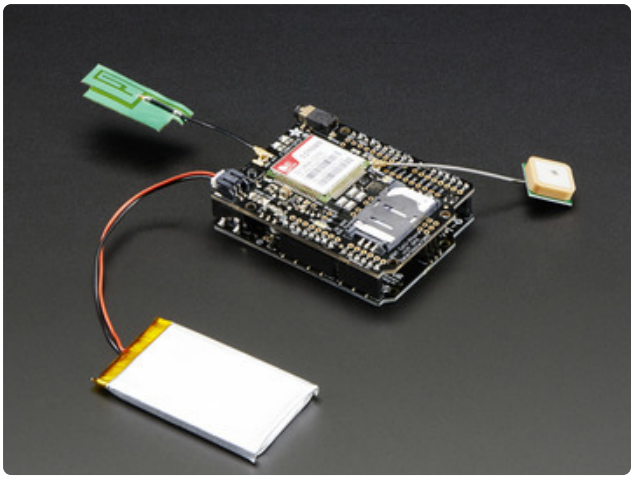
How humid is my apartment right now? **Let me text my FONA!**

While smart-home applications like as Apple's HomeKit are great - they use a large amount of cellular data and take time to load information you may not care about. Instead of using a WiFi connection, an Adafruit FONA cellular module lets you **obtain sensor readings from anywhere with cell reception**.

In this guide, you will build a text-able environmental monitor using the Adafruit FONA shield, a Metro M0/M4 development board and a Bosch BME280 precision sensor. Using CircuitPython, the Feather can send and receive SMS messages using the CircuitPython FONA library (https://adafru.it/LdF) and read environmental data from the BME280 sensor with the CircuitPython BME280 library (https://adafru.it/BfX).

Code with CircuitPython

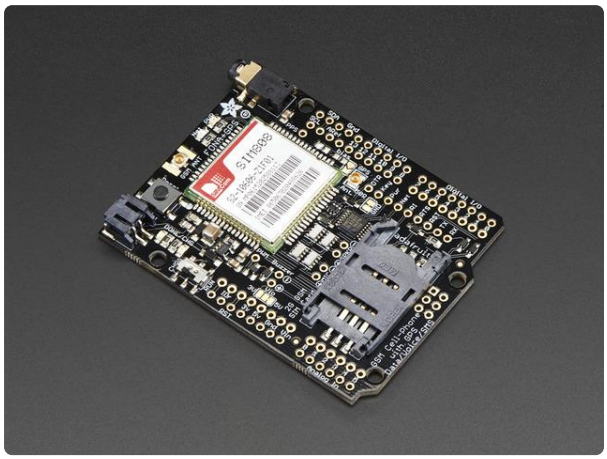
CircuitPython is the easiest way to program an Internet of Things project. We've built a helper library, Adafruit_CircuitPython_FONA, (https://adafru.it/LdF) to make interfacing with the FONA module's SMS capabilities incredibly simple.



Adafruit FONA

The Adafruit FONA (https://adafru.it/LdG) is an all-in-one cellular phone module that lets you add voice, cellar data, location-tracking (FONA-808 and FONA-3G only) and SMS to your project. The FONA shields fit right over shield-compatible CircuitPython boards.

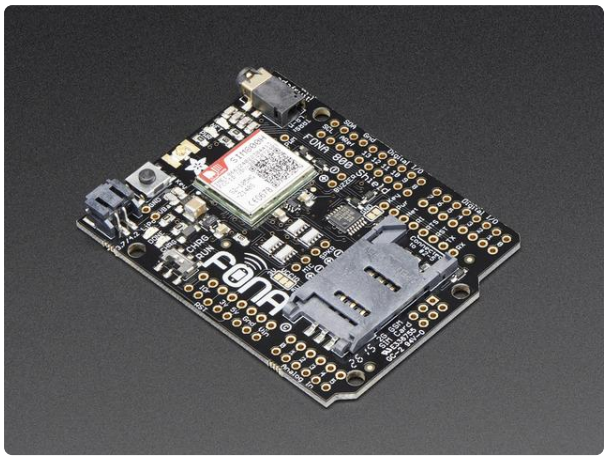
Parts



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...

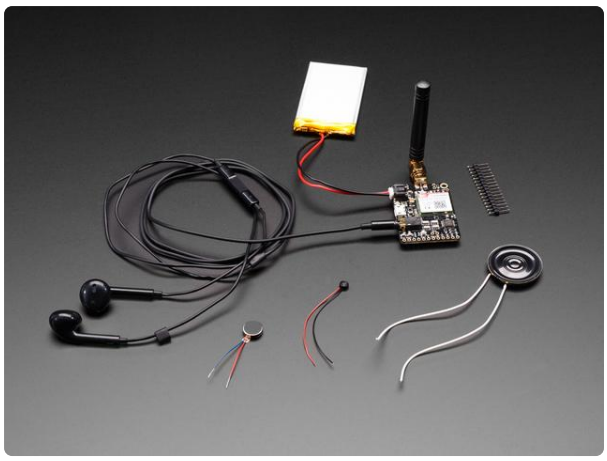
https://www.adafruit.com/product/2636



Adafruit FONA 800 Shield - Voice/Data Cellular GSM for Arduino

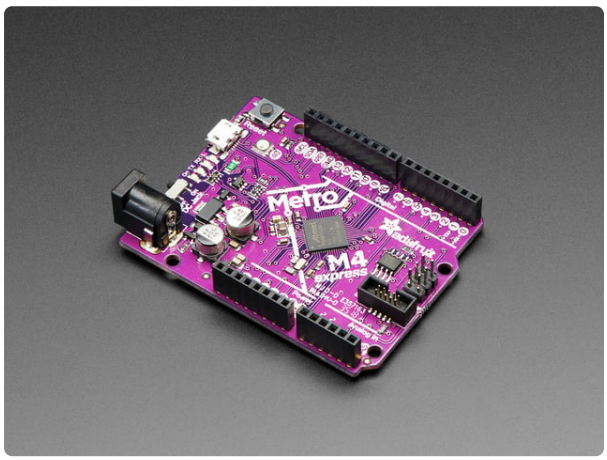
Ring, Ring! Who's that callin'? It's your Arduino! Introducing Adafruit FONA 800 Shield, an adorable all-in-one cellular phone shield that lets you add voice, text, SMS and...

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

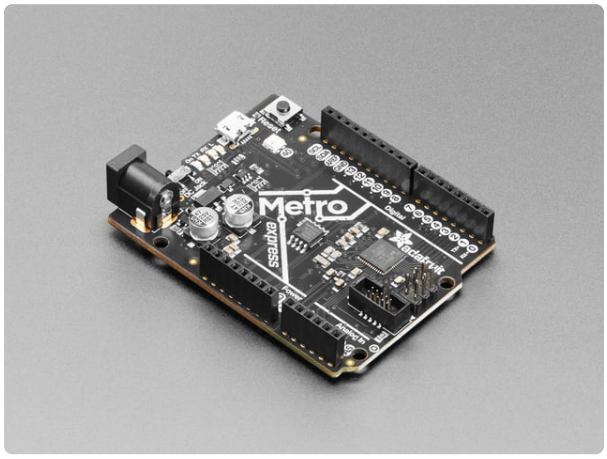


[Adafruit FONA 800 Breakout Board Starter Pack - SMA Version]
Build your own cellular project and get off the grid with FONA. This pack comes with an SMA-antenna type FONA and paired with hand-picked accessories. It's the Adafruit...
https://www.adafruit.com/product/2522

On its own, this shield can't do anything. It requires a microcontroller like a Metro M4 or Metro M0 to drive it!

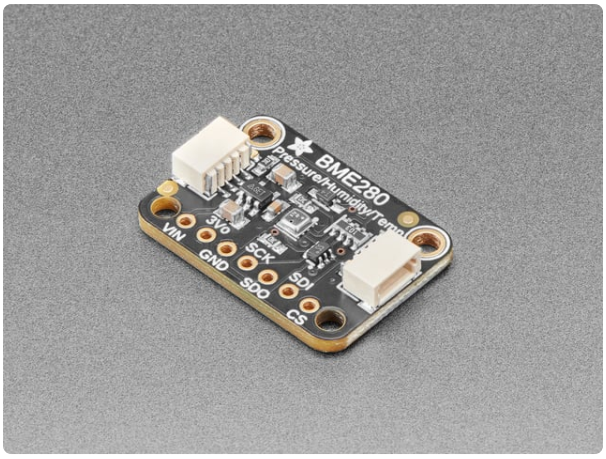


[Adafruit Metro M4 feat. Microchip ATSAMD51]
Are you ready? Really ready? Cause here comes the fastest, most powerful Metro ever. The Adafruit Metro M4 featuring the Microchip ATSAMD51. This...
https://www.adafruit.com/product/3382



[Adafruit METRO M0 Express - designed for CircuitPython]
Metro is our series of microcontroller boards for use with the Arduino IDE. This new Metro M0 Express board looks a whole lot like our
https://www.adafruit.com/product/3505

The Bosch BME280 precision sensor can measure humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy.



Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor

Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...

https://www.adafruit.com/product/2652

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 2G SIM Card from Ting.

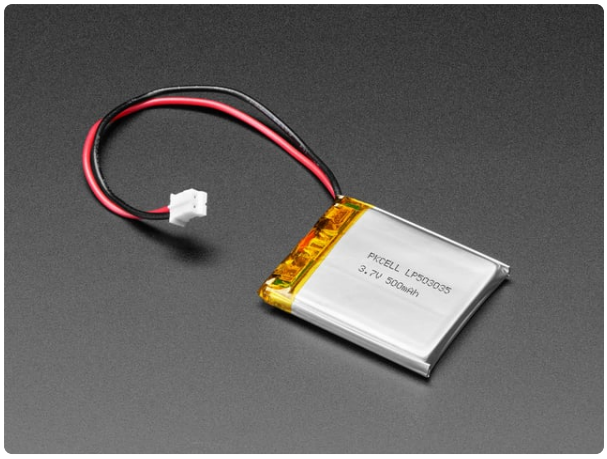
- If you're not in the US, or want to use a different cellular network provider, please see this page for more information about obtaining a FONA-compatible SIM card. (https://adafru.it/KVE)

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

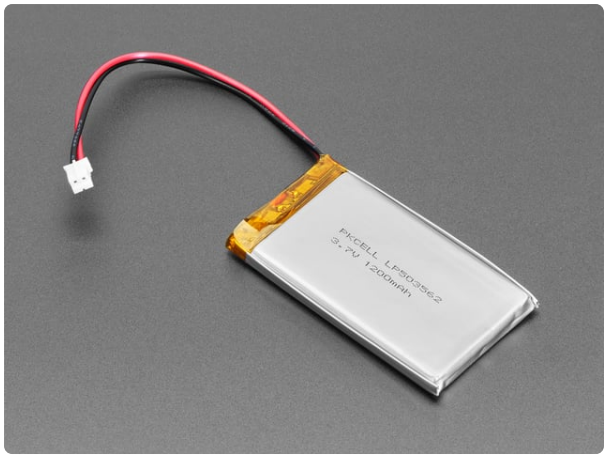
Adafruit is now a phone company :) Or, well, we've sold DIY cell phones for awhile now but you've never been...

https://www.adafruit.com/product/2505

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

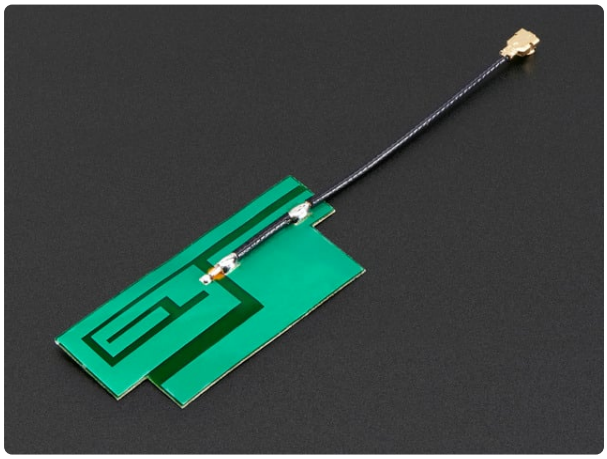


[Lithium Ion Polymer Battery - 3.7v 500mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/1578



[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

Materials

The supplies listed below are both helpful and necessary for completing this project.

1 x MicroUSB Cable — https://www.adafruit.com/product/592
USB cable - USB A to Micro-B - 3 foot long

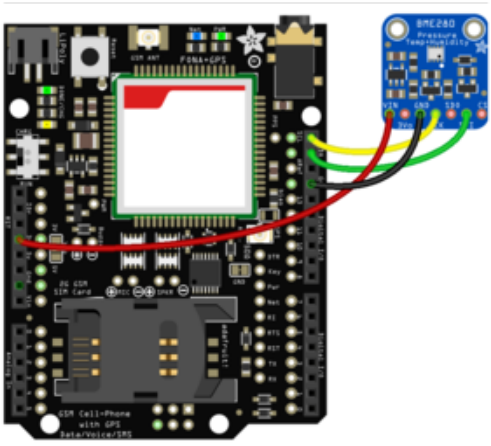
---

1 x Shield Stacking Headers — https://www.adafruit.com/product/85
Shield stacking headers for Arduino (R3 Compatible)

Assembly

Wiring

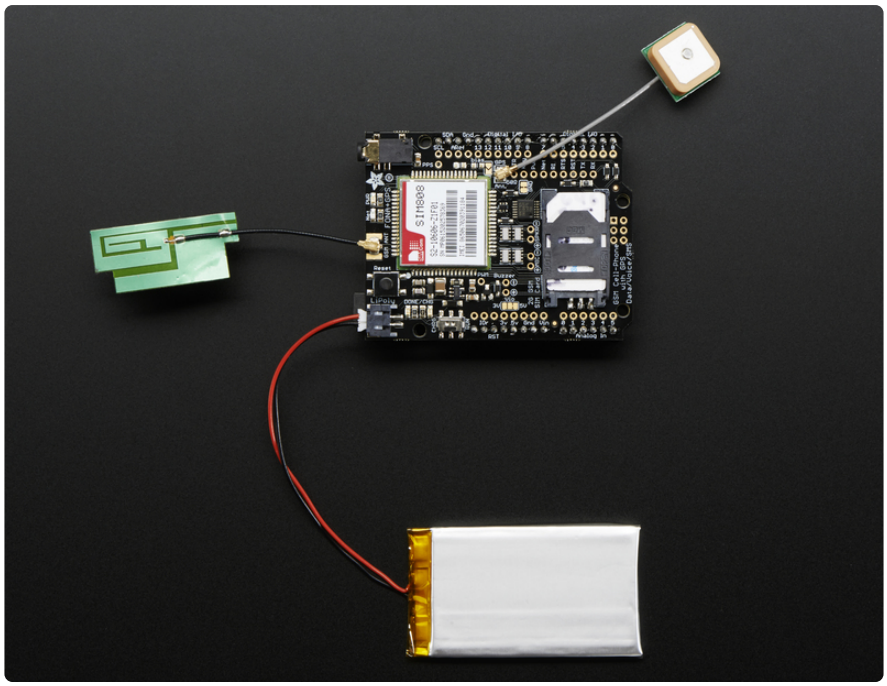
Next, wire up a BME280 to your board. If you'd like to avoid soldering directly to the FONA shield, you can pick up some shield stacking headers (http://adafru.it/85) and some jumper cables (http://adafru.it/153).



Board 3V to **sensor VIN**
Board GND to **sensor GND**
Board SCL to **sensor SCK**
Board SDA to **sensor SDI**

Attach Antennas and Battery

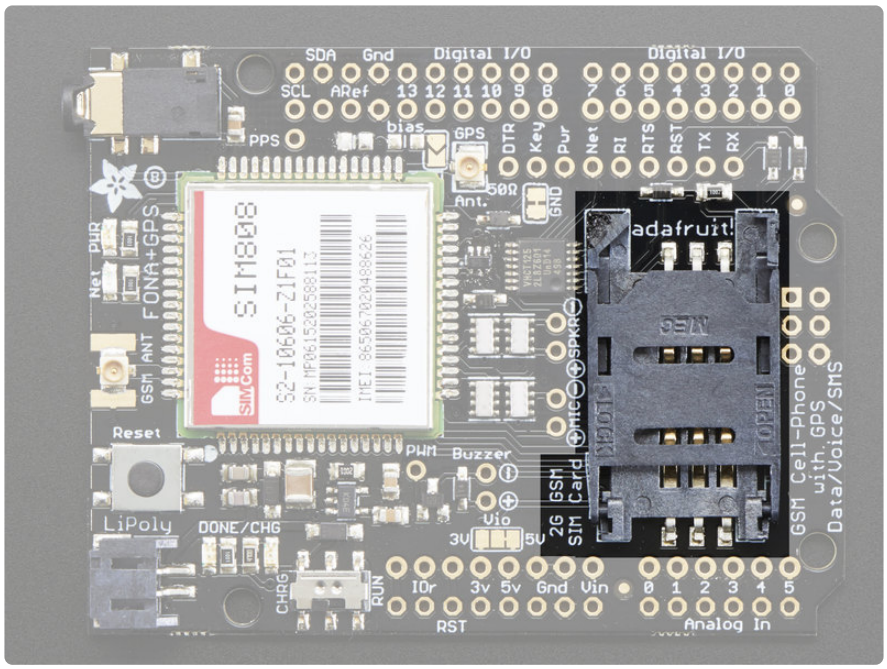
A battery and GSM antenna is required to use the Cellular module. If you want to use GPS as well, a passive GPS antenna is also required



Check polarity for the battery!
Snap the uFL connector on, it will click when placed properly

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

Insert SIM Card



You **must** insert a SIM card to do anything but the most basic tests. The shield and GPS does work without a SIM but of course you cannot send or receive texts, calls, etc!

- If you have not yet inserted a SIM card into your FONA, follow the instructions on this page and come back to this guide when you're done. (https://adafru.it/Lec)

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

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 CircuitPython version 4.0.0 or higher.

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_simpleio.mpy**
- **adafruit_bme280.mpy**

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

Code Usage

Install CircuitPython Code

In the embedded code element below, click on the **Download: Project Zip** link, and save the .zip archive file to your computer.

Then, uncompress the .zip file, it will unpack to a folder named **FONA_SMS_Sensor**.

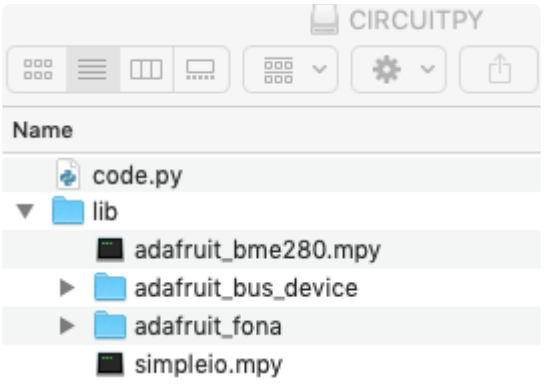
```
# SPDX-FileCopyrightText: 2020 Brent Rubell for Adafruit Industries
#
```

```
# SPDX-License-Identifier: MIT

# pylint: disable=unused-import
import time
import board
import busio
import digitalio
from adafruit_fona.adafruit_fona import FONA
from adafruit_fona.fona_3g import FONA3G
import adafruit_bme280

print("FONA SMS Sensor")

# 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 BME280 Sensor
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# Initialize Network
while fona.network_status != 1:
    print("Connecting to network...")
    time.sleep(1)
print("Connected to network!")
print("RSSI: %ddB" % fona.rssi)

# Enable FONA SMS notification
fona.enable_sms_notification = True

print("Listening for messages...")
while True:
    sender, message = fona.receive_sms()
    if message:
        print("New Message!")
        print("FROM: ", sender)
        print("MSG: ", message)

        # Read BME280 sensor values
        temp = bme280.temperature
        humid = bme280.humidity
        pres = bme280.pressure

        # Sanitize message
        message = message.lower()
        message = message.strip()

        if message in ["temp", "temperature", "t"]:
            response = "Temperature: %0.1f C" % temp
        elif message in ["humid", "humidity", "h"]:
            response = "Humidity: %0.1f %%" % humid
        elif message in ["pres", "pressure", "p"]:
            response = "Pressure: %0.1f hPa" % pres
        elif message in ["status", "s"]:
            response = "Temperature: {0:.2f}C\nHumidity: {1:.1f}%Pressure: {2:.1f}
hPa".format(
                temp, humid, pres
            )
        elif message in ["help"]:
            response = "I'm a SMS Sensor - txt me with a command:\
                        TEMP - Read temperature\
                        HUMID - Read humidity\
```

```
                    PRES - Read pressure\
                    STATUS - Read all sensors.\
                    HELP - List commands"
else:
    response = "Incorrect message format received. \
                    Text HELP to this number for a list of commands."

# Send a response back to the sender
print("Sending response...")
if not fona.send_sms(int(sender), response):
    print("SMS Send Failed")
print("SMS Sent!")
```

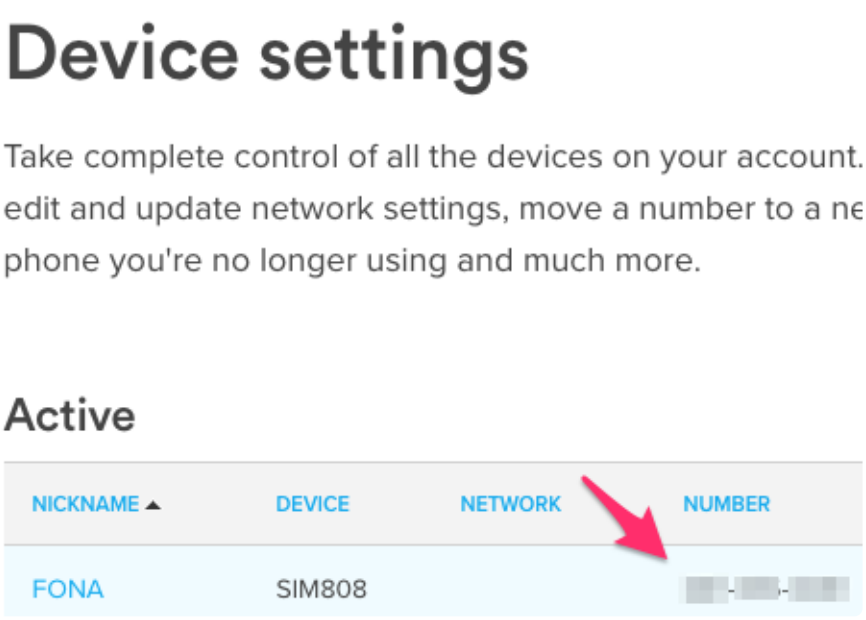
Copy the contents of the **FONA_SMS_Sensor** directory to your PyPortal's **CIRCUITPY** drive.

This is what the final contents of the **CIRCUITPY** drive will look like:



Obtain FONA's Number

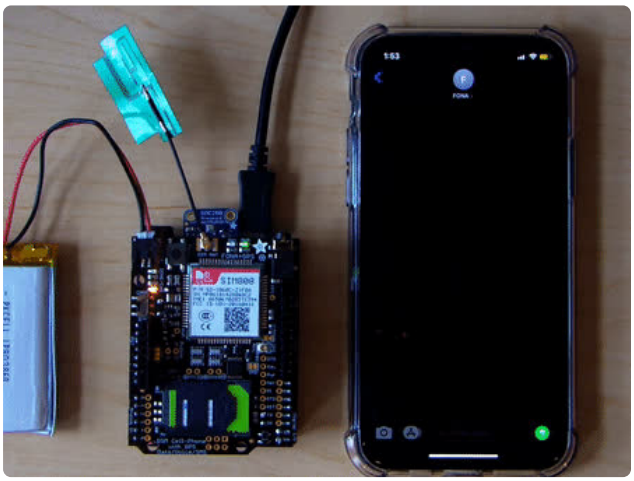
Next, you'll need your SIM card's phone number. This should be available from your carrier's website. Here's an example of the Ting devices page displaying the FONA's number



Code Usage

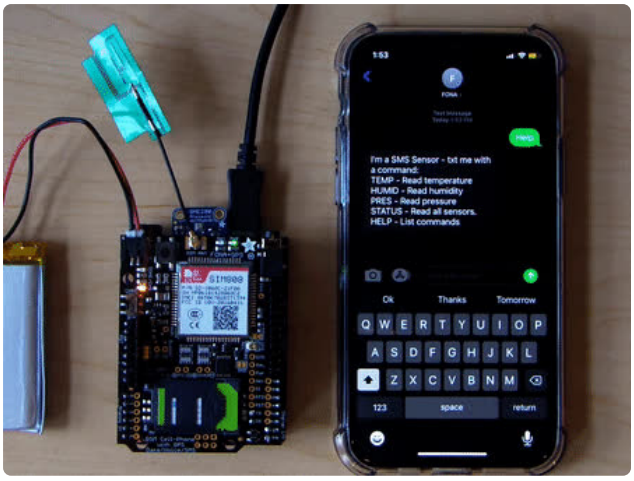
After saving the **code.py** to your board, **open the REPL**. After the FONA registers with the network, it will output its received signal strength indicator (RSSI) level and display that the FONA is ready to receive messages from your phone.

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
FONA SMS Sensor
Connecting to network...
Connecting to network...
Connecting to network...
Connecting to network...
Connected to network!
RSSI: -84dB
FONA Ready!
```



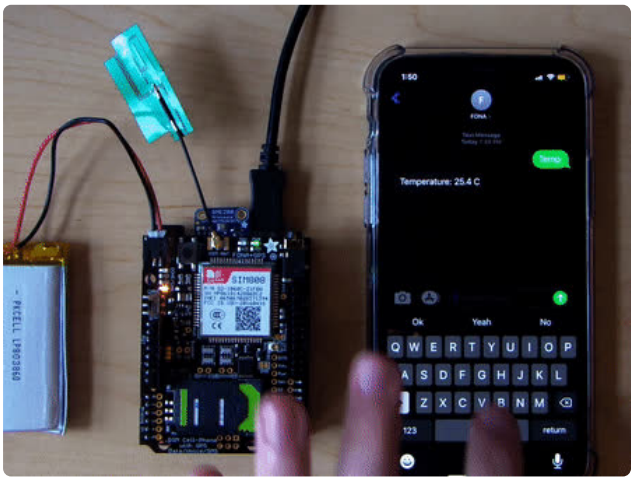
Text HELP to your device's phone number.

The FONA should reply with a SMS listing all possible commands.

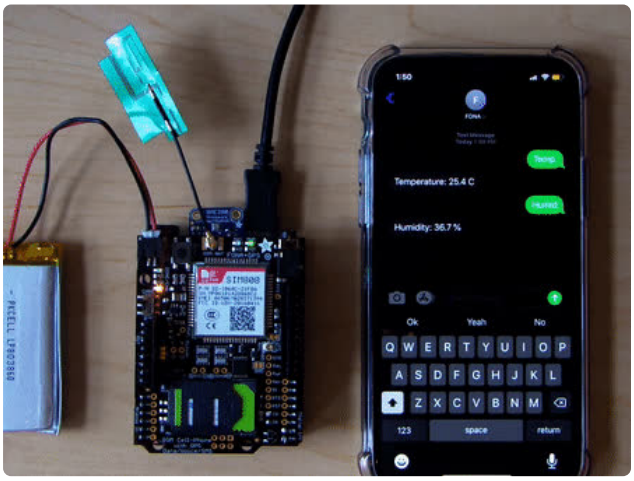


Let's try texting one of the commands. You can read the temperature by texting TEMP to the FONA.

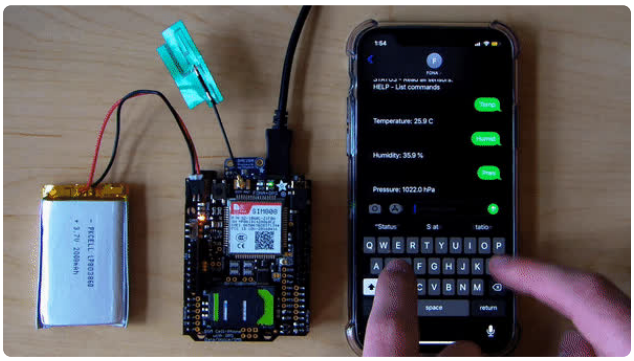
Try this by texting TEMP to the FONA. The FONA will reply with the BME280's temperature reading.



Text HUMID to the FONA. The FONA will reply with the the current relative humidity reading.



Text PRES to the FONA. The FONA will reply with the current pressure.



Want to read all the sensors on the BME280 breakout at once? **Text STATUS to the FONA module**.

The FONA will reply back with readings from all the sensors on the BME280.

Code Walkthrough

The code first creates a serial UART connection with the FONA module and a DigitalInOut object to digitally control IO Pin D4. Then, it initializes the FONA module.

```
# 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)
```

Then an I2C connection is established with the BME280 sensor.

```
# Initialize BME280 Sensor
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
```

The FONA takes some time to register itself with the cellular network. Depending on your location relative to cellular towers, this step may take some time.

Once connected, the received signal strength indicator (RSSI) will be printed to the REPL.

```
# Initialize Network
while fona.network_status != 1:
    print("Connecting to network...")
    time.sleep(1)
print("Connected to network!")
print("RSSI: %ddB" % fona.rssi)
```

By default, the FONA module does not raise any notifications when it receives a text message. Setting `enable_sms_notification` tells the FONA module to send a message to the microcontroller whenever it receives a new message.

```
# Enable FONA SMS notification
fona.enable_sms_notification = True
```

On each iteration of the `while True` loop, the code checks if data is available to be read from the FONA module using the `receive_sms()` method.

```
while True:
    sender, message = fona.receive_sms()
```

The code reads the sender and message from the SIM card's memory slot.

```
if message:
        print("New Message!")
        print("FROM: ", sender)
        print("MSG: ", message)
```

The BME280 sensor's values are read and stored in temp, humid, and pres.

```
# Read BME280 sensor values
temp = bme280.temperature
humid = bme280.humidity
pres = bme280.pressure
```

The code selects a response based on the contents of the SMS message.

```
if message in ['temp', 'temperature', 't']:
    response = "Temperature: %0.1f C" % temp
elif message in ['humid', 'humidity', 'h']:
    response = "Humidity: %0.1f %%" % humid
elif message in ['pres', 'pressure', 'p']:
    response = "Pressure: %0.1f hPa" % pres
elif message in ['status', 's']:
    response = "Temperature: {0:.2f}C\nHumidity: {1:.1f}% \
               Pressure: {2:.1f}hPa".format(temp, humid, pres)
elif message in ['help']:
    response = "I'm a SMS Sensor - txt me with a command:\
               TEMP - Read temperature\
               HUMID - Read humidity\
               PRES - Read pressure\
               STATUS - Read all sensors.\
               HELP - List commands"
else:
    response = "Incorrect message format received. \
               Text HELP to this number for a list of commands."
```

The response is sent back to the sender's phone number.

```
# Send a response back to the sender
print("Sending response...")
if not fona.send_sms(int(sender), response):
  print("SMS Send Failed")
  print("SMS Sent!")
```

That's it! The code will wait to receive the next text message.