How to make a GSM Location Tracker with the AdaFruit FONA and Arduino

by **kinasmith** on March 7, 2015

Table of Contents

Intro: How to make a GSM Location Tracker with the AdaFruit FONA and Arduino

Introduction

In this tutorial we are going to make a web connected location logger using the Adafruit FONA Board, an Arduino, and the Sparkfun Data Service. It will get its location using triangulation and post it to an online database with a GET request over a cellular (GPRS) connection.

We will be using CoolTerm to communicate with the FONA, so please install it here. There is a link to a Sparkfun Tutorial on how to use it in Resources.

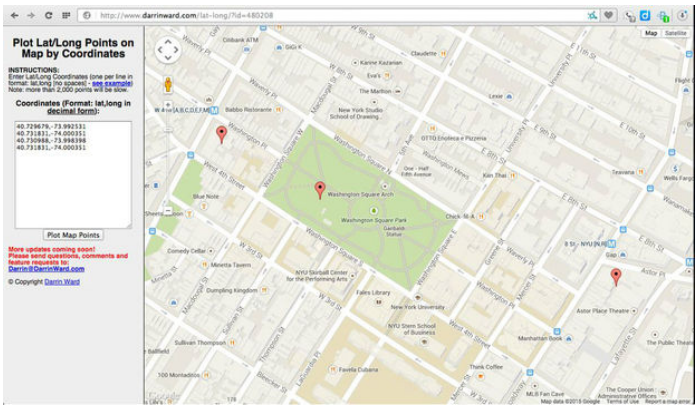
We will also be using the Sparkfun Data Service for the online database portion of this tutorial. Any "internet of things" service should work, however, with some modification to the code.

Resources

- Adafruit FONA page. Great resource for pinouts/SIM card info, etc
- The Datasheets for the GSM module, These are a good reference for commands and more technical info.
  - IP Application Notes
  - Serial Port Application Notes
  - FM Application Notes
- How to Use CoolTerm
- data.Sparkfun Tutorial

Parts List

- Adafruit FONA board
  - I have the uFL antenna connector version for its small size, but either version should work fine
- GSM Antenna
  - I have a uFL antenna, but any antenna of the right frequency that fits should be ok. Look at the Adafruit documentation and be sure to get the right one
- Lipo Battery
- Arduino Uno
- A breadboard
- The Internet
- SIM Card
- Hookup Wire



Step 1: Hookup the Fona

You will need your Lipo Battery, the Arduino, the FONA + Antennae, a breadboard, your SIM card, and some hookup wire. I like to attach my breadboards and Arduino's together like in the picture, it makes for a more reliable connection and less troubles trying to keep the two in close proximity while prototyping. I just used a piece of blue corrugated plastic and some hot glue, but a piece of cardboard will work just as well.

- Insert your SIM Card into the SIM Slot on the back of the Board
- Attach the Antenna

Make the following Connections

Arduino to Breadboard

- Arduino GND -> breadboard Ground
- Arduino 5v -> breadboard Power

FONA to Breadboard

- Solder on the header pins if you haven't already
- Plug the FONA like in the picture, with the Bat PIN in ROW 1
- FONA Vio -> Power Bus (This is VERY important. Communication won't work without this)
- FONA GND -> Ground Bus

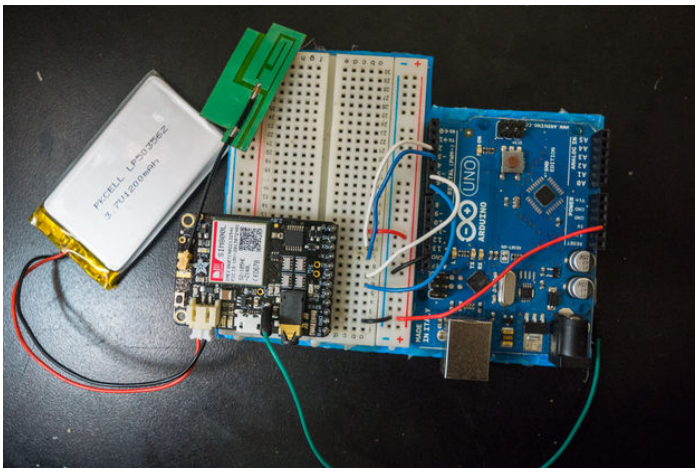
FONA to Arduino

- FONA RX -> PIN 3
- FONA TX -> PIN 4
- FONA PS -> PIN 6 ('Power State' pin. It is HIGH when the FONA is on, and LOW with the FONA is off)
- FONA KEY -> PIN 7 (Pull this pin low for 2 seconds and the FONA will turn off/on)

Your setup should look somewhat like it does in the picture.

- Double check all of the connections
- Plug the Battery into the Connector on the FONA

The FONA is powered from the battery it is attached to. It will not work without that battery there. The USB connector on the FONA charges the Battery, that is all. The FONA will not work with just the USB plugged in, as the power comes from the Battery, NOT the USB connection. If you want to charge the Battery from an external 5v source, there is a hole next to the USB port labeled '5v'. I have a green wire soldered into it, so I can charge the battery from the Arduino. This is useful for embedding but not needed for prototyping.



Step 2: Test the FONA

Upload the following code to test the FONA - Arduino Connection:

```
#include <SoftwareSerial.h><softwareserial.h><br></softwareserial.h>

#define FONA_RX 3 //comms
#define FONA_TX 4 //comms
#define FONA_KEY 6 //powers board down
#define FONA_PS 7 //status pin. Is the board on or not?

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX); //initialize software serial
char inChar = 0;

void setup() {
   pinMode(FONA_PS, INPUT);
   pinMode(FONA_KEY,OUTPUT);
   digitalWrite(FONA_KEY, HIGH);
   Serial.begin(9600);
   Serial.println("Serial Ready");
   fonaSS.begin(9600);
   Serial.println("Software Serial Ready");
}

void loop() {
   if (fonaSS.available()){
       inChar = fonaSS.read();
       Serial.write(inChar);
       delay(20);
   }
   if (Serial.available()>0){
       fonaSS.write(Serial.read());
   }
}
```

Turn on FONA

- to turn on the the FONA press and hold the button by the Battery Connector for 2 seconds and release.
- The blue power light should turn on and if the SIM connects, a red light will start to blink slowly.
- To turn off the board press and hold the same button

Connect Coolterm

Once the code is uploaded, connect to the Arduino Serial Port using Coolterm.

- Select the correct Serial Port
- Baud Rate is set to 9600
- Under Terminal on the left make sure Line Mode is selected
- Click OK and then Connect
- Then in the command bar type: AT and press Enter. It is not case sensitive.
- If the FONA module is active and listening it will respond with OK

If everything has gone OK up to here, Congrats!

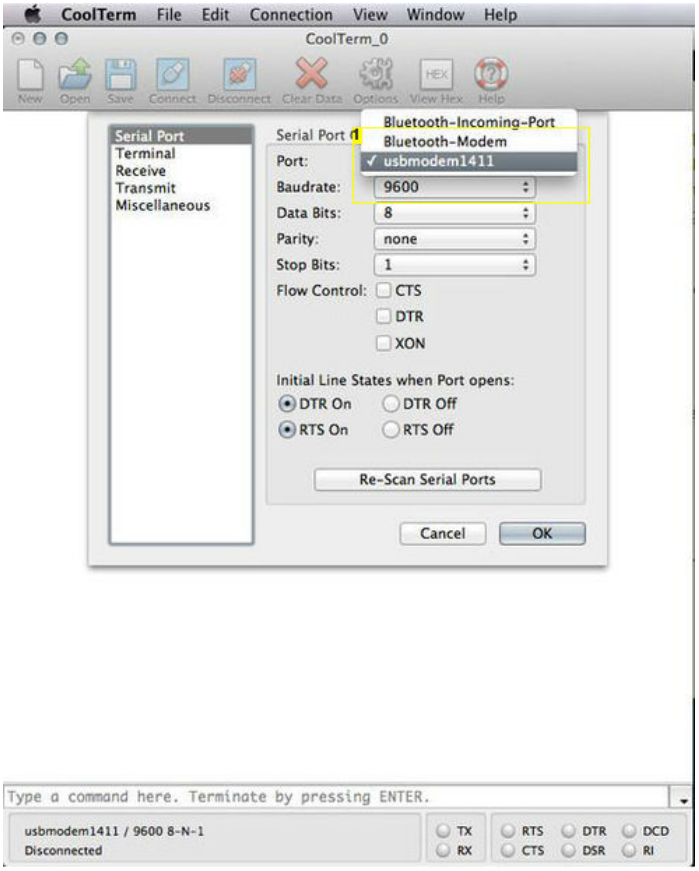


Image Notes
1. Select Arduino Serial Port

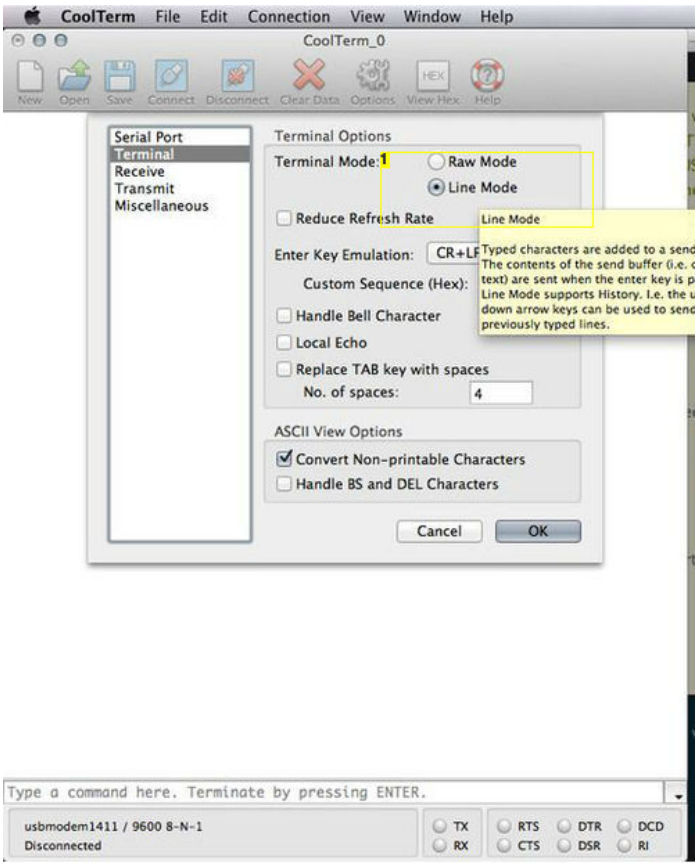


Image Notes
1. Select Line Mode

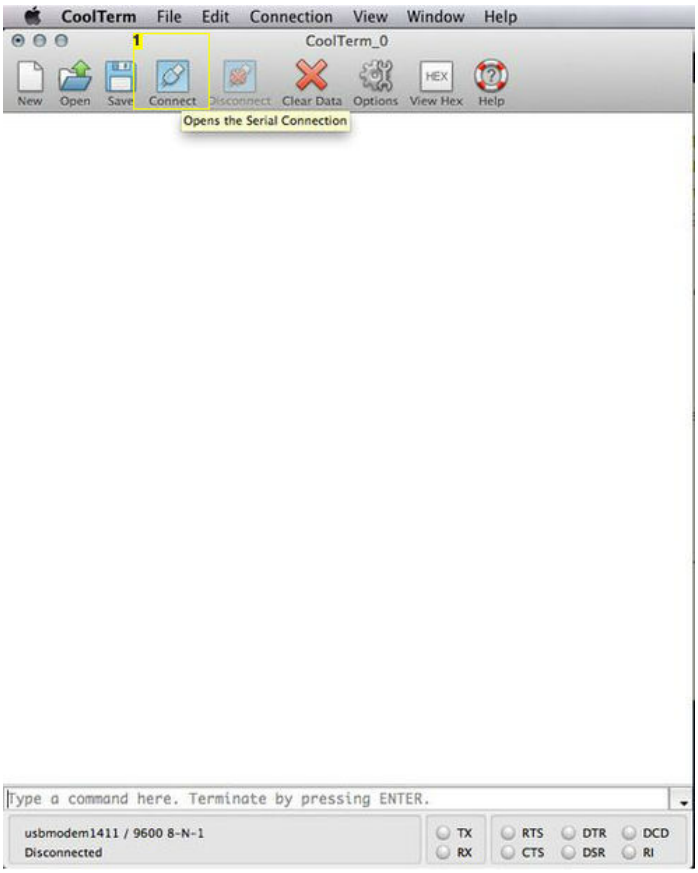


Image Notes
1. Open Serial Connection

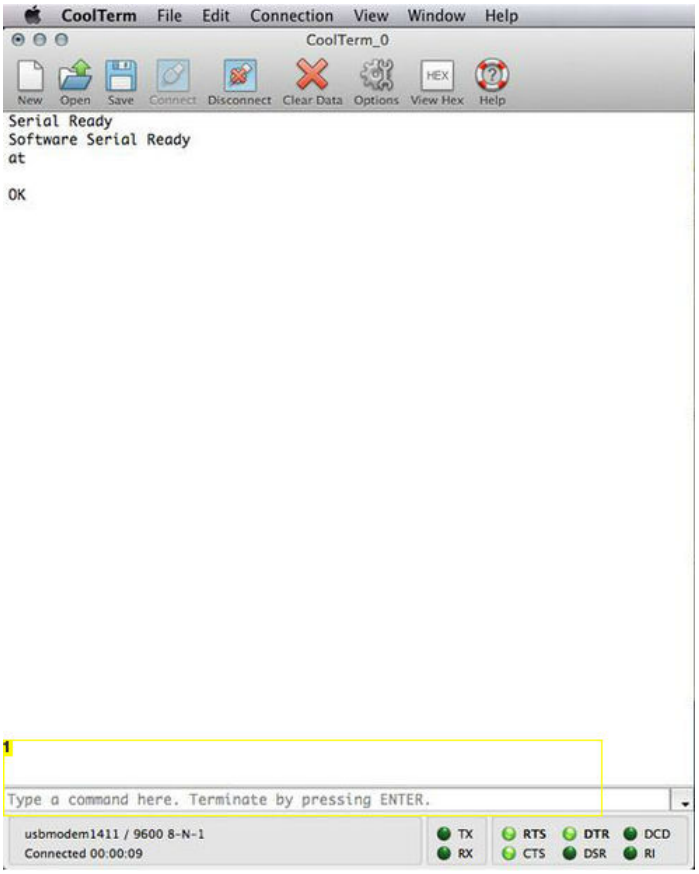


Image Notes
1. Type Commands down here

File Downloads

FONA_Test.ino (708 bytes)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'FONA_Test.ino']

Step 3: The AT Commands

Working with these GSM modules is basically just sending them commands over the Serial Port and parsing the responses. These modules have quite a bit of smarts built into them, which makes this process easier.

To send a command, type it into the command line in Coolterm and press Enter.

Command Syntax

- Test Command: AT+=?
  - returns a list of parameters or value ranges that you can set with the command
- Read Command: AT+?
  - Returns the current set value of the parameters of that command
- Write Command: AT+=<....>
  - This command sets user definable parameter values
- Execution Command: AT+
  - Executes a command without user definable parameter values
- For a short list of useful commands.
  - Try out all of these commands. It's pretty cool! Try making a phone call or sending a text!
- For a compendium of all the commands the datasheet is here.

Get your Location!

Type in the following commands to get your (approximate) location

- AT+CMGF=1
  - response: OK
- AT+CGATT=1
  - response: OK
- AT+SAPBR=3,1,"CONTYPE","GPRS"
  - response: OK
- AT+SAPBR=3,1,"APN","your apn here"
  - response: OK

- AT+SAPBR=1,1 -> note: the red light should start blinking faster
    - response: OK
- AT+CIPGSMLOC=1,1
    - response: +CIPGSMLOC:
    - followed by: OK

Make a GET Request!

To make a GET request, it's a similar process at the beginning. You should type all of these out and make sure your SIM card connects and does a GET request manually before you continue. It is actually very important in these complicated systems to check functionality at each step so if it stops working you know exactly at which point it stopped working. It makes the debugging process much easier.

- First we setup the GPRS:
    - AT+CMGF=1
    - AT+CGATT=1
    - AT+SAPBR=3,1,"CONTYPE","GPRS"
    - AT+SAPBR=3,1,"APN","your apn here"
    - AT+SAPBR=1,1
- The we setup HTTP and make the request
    - AT+HTTPINIT
    - AT+HTTPPARA="CID",1
    - AT+HTTPPARA="URL","your url here"
    - AT+HTTPACTION=0
    - AT+HTTPREAD
- Then we close the HTTP and the GPRS
    - AT+HTTPTERM
    - AT+SAPBR=0,1

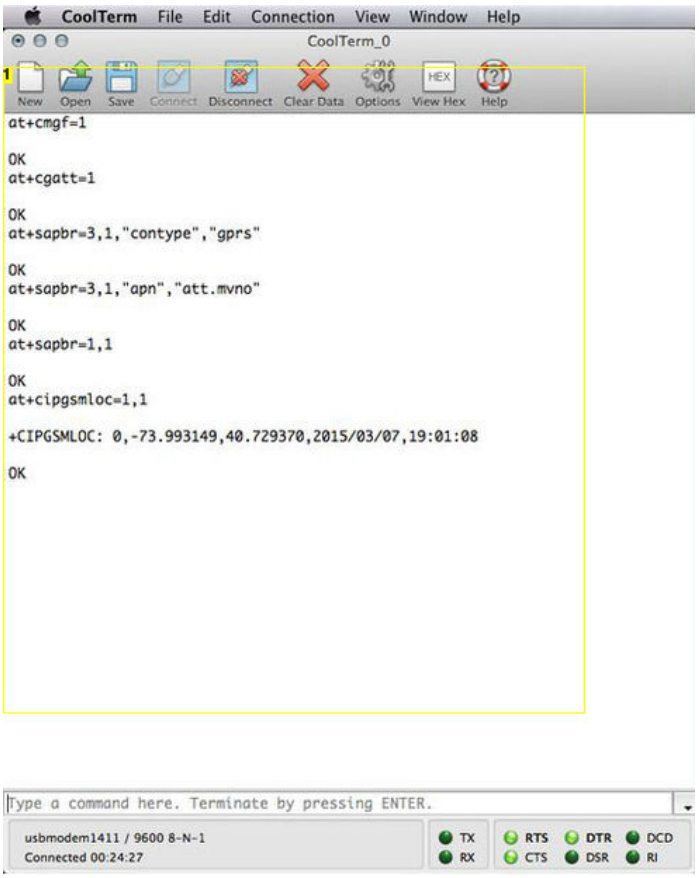


Image Notes
1. Commands and responses for getting Location

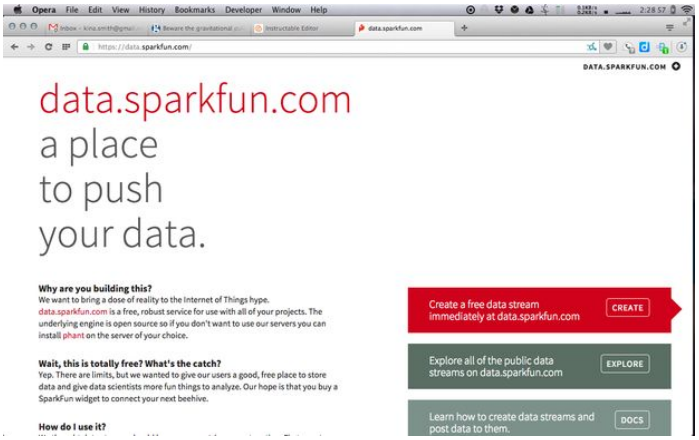
Step 4: Set up a Sparkfun Data Stream

The next step is to create a place for us to send the location data we just received. We are going to use a service that Sparkfun provides for free: data.sparkfun.com. There is a nice set of tutorials that they have put together for it, so I won't be redundant.

For the included code to work right there needs to be 4 columns labeled:

- latitude
- longitude
- time
- date

Once that is done, make sure you have the Private and Public keys and move on to the next step.



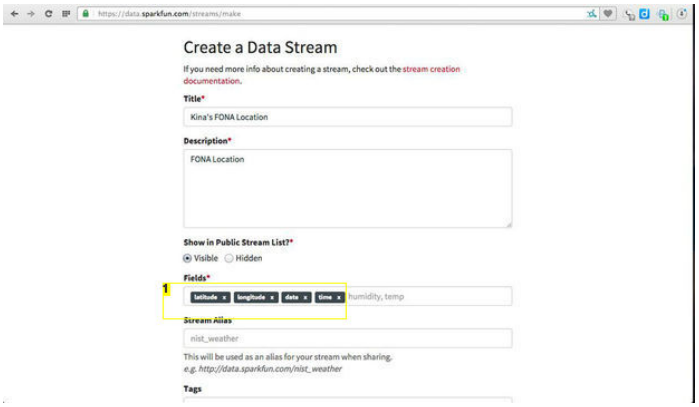
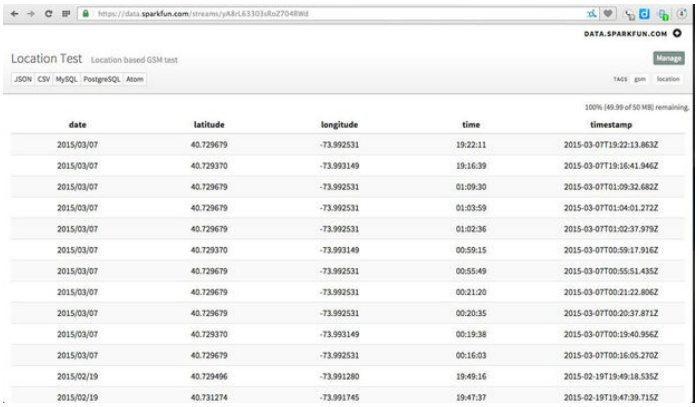


Image Notes
1. These need to match your code



date	latitude	longitude	time	timestamp
2015/03/07	40.729679	-73.992531	19:22:11	2015-03-07T19:22:13.863Z
2015/03/07	40.729370	-73.993149	19:16:39	2015-03-07T19:16:41.946Z
2015/03/07	40.729679	-73.992531	01:09:30	2015-03-07T01:09:32.682Z
2015/03/07	40.729679	-73.992531	01:03:59	2015-03-07T01:04:01.272Z
2015/03/07	40.729679	-73.992531	01:02:36	2015-03-07T01:02:37.979Z
2015/03/07	40.729370	-73.993149	00:59:15	2015-03-07T00:59:17.916Z
2015/03/07	40.729679	-73.992531	00:55:49	2015-03-07T00:55:51.435Z
2015/03/07	40.729679	-73.992531	00:21:20	2015-03-07T00:21:22.806Z
2015/03/07	40.729679	-73.992531	00:20:35	2015-03-07T00:20:37.871Z
2015/03/07	40.729370	-73.993149	00:19:38	2015-03-07T00:19:40.956Z
2015/03/07	40.729679	-73.992531	00:16:03	2015-03-07T00:16:05.270Z
2015/02/19	40.729496	-73.991280	19:49:16	2015-02-19T19:49:18.535Z
2015/02/19	40.731274	-73.991745	19:47:37	2015-02-19T19:47:39.715Z

Step 5: Putting it into Code

We have learned what commands we need to get our location, to send a get request, and we have created an online database to send our data to. The next step is creating code to automate the process of getting our location and sending up to the database.

- Download the attached: FONA_Location.ino
- Copy your APN, Public, and Private keys into the correct spots at the top of the code
- Upload
- Connect with CoolTerm and watch the Serial Monitor for any Errors. It should look like the attached image.
- Make sure the HTTPACTION responds with a 200
- Check your Sparkfun Datastream to see if the data appears.
- I tried to make the code as legible as possible, there are lots of comments.

At this point you should have the GSM module autonomously getting location and posting it to Sparkfun every 5 minutes, which means we are successful. You can now power your arduino with a battery and take it for a walk and see if it works!

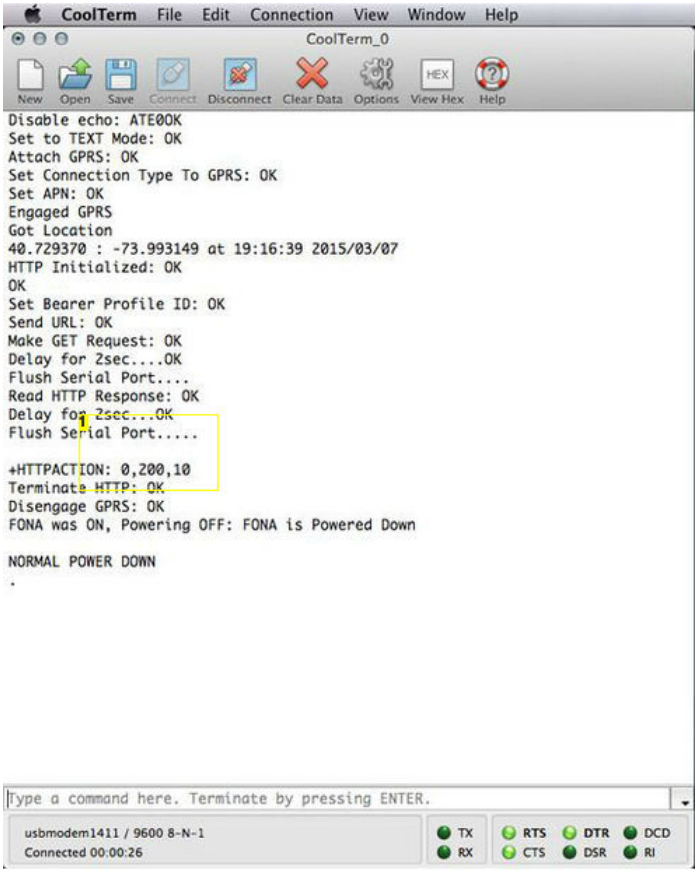


Image Notes
1. Make sure the HTTPACTION responds with a 200. Otherwise it isn't completing the request ok

File Downloads

FONA_Location.ino (13 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'FONA_Location.ino']

Step 6: Epilogue

The other aspect we haven't talked about here is power consumption. It would be safe to assume that if you are building a cellular tracker, you will want to put it on something that moves, which will mean it will need to stray from the power outlet/usb ports of our world. This means battery power, which means measuring power consumption and calculating battery life, etc.

Cellular stuff has a reputation for being pretty power hungry. Adafruit says this module can draw up to 2 Amps (!!) in short bursts. However, the way we are using it, it spends most of its time power off, consuming very little.

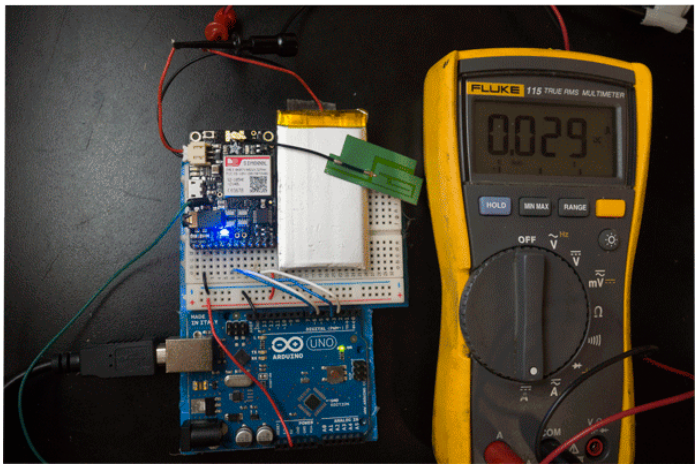
I measured the power consumption from the Battery during normal use.

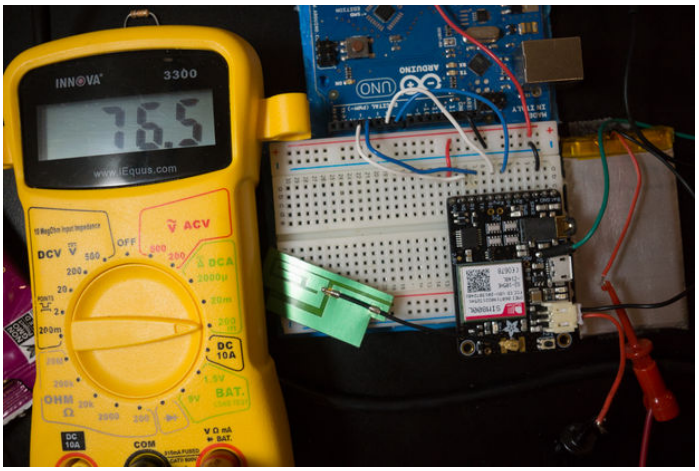
- Powered on: ~ 25mA
- Get Request: Peaks at 150-200mA for short durations
- One full cycle of requesting location and sending a GET request consumes ~ 7.47 joules
  - avg of 53.4 mA for 35 Seconds
- Powered off: ~150µA quiescent power consumption

If we were using a battery with a capacity of 1000mAH, we could get send locations for ~ 10 days at 15 minute intervals before the battery gave out. (in theory, and not considering the power consumption of the Arduino)

Note:

I did notice some strange behavior when the battery was connected to the FONA and the Arduino was unplugged. There was between 20-100mA flowing out of the battery for no reason. I'm not sure why this is, but I would be cautious about leaving batteries plugged in to the FONA without it being controlled by a microcontroller.





Related Instructables

SM5100B GPRS and General Notes by BenHarper

Intel Edison Location Logger by chipmc

Remote control via GPRS/GSM SMS(Arduino) by Elecrow

Mailbox Phone Alert by nikoala3

Tutorial EFCom - GRPS/GSM Shield Arduino by ElecFreaks

EFCom Pro GPRS/GSM Module User Guide by ElecFreaks

Comments