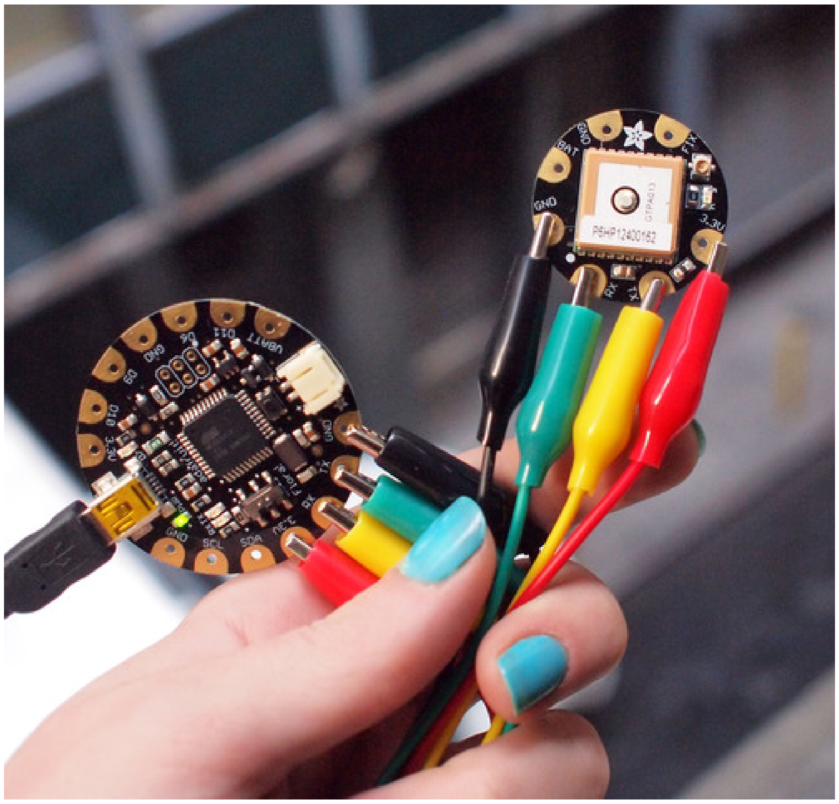
adafruit learning system

# Flora Wearable GPS

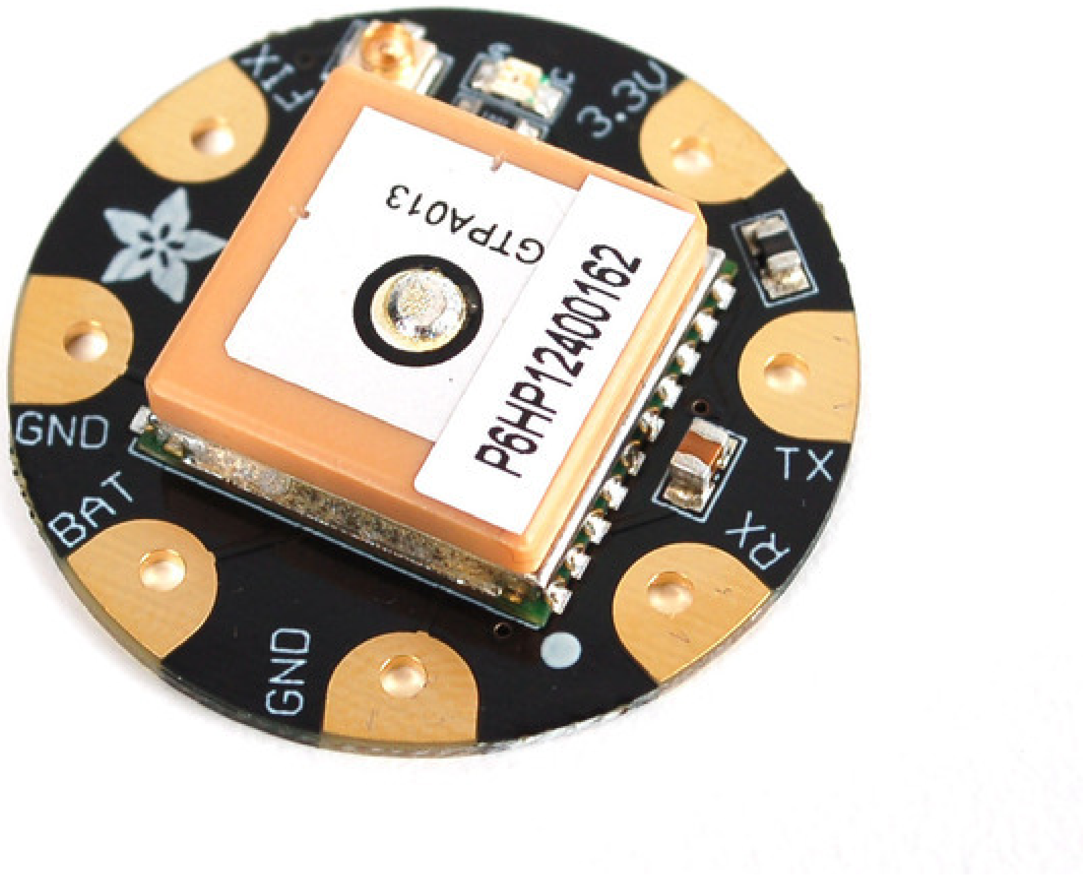
Created by Becky Stern



Last updated on 2015-01-15 10:30:30 PM EST

## Guide Contents

## Overview

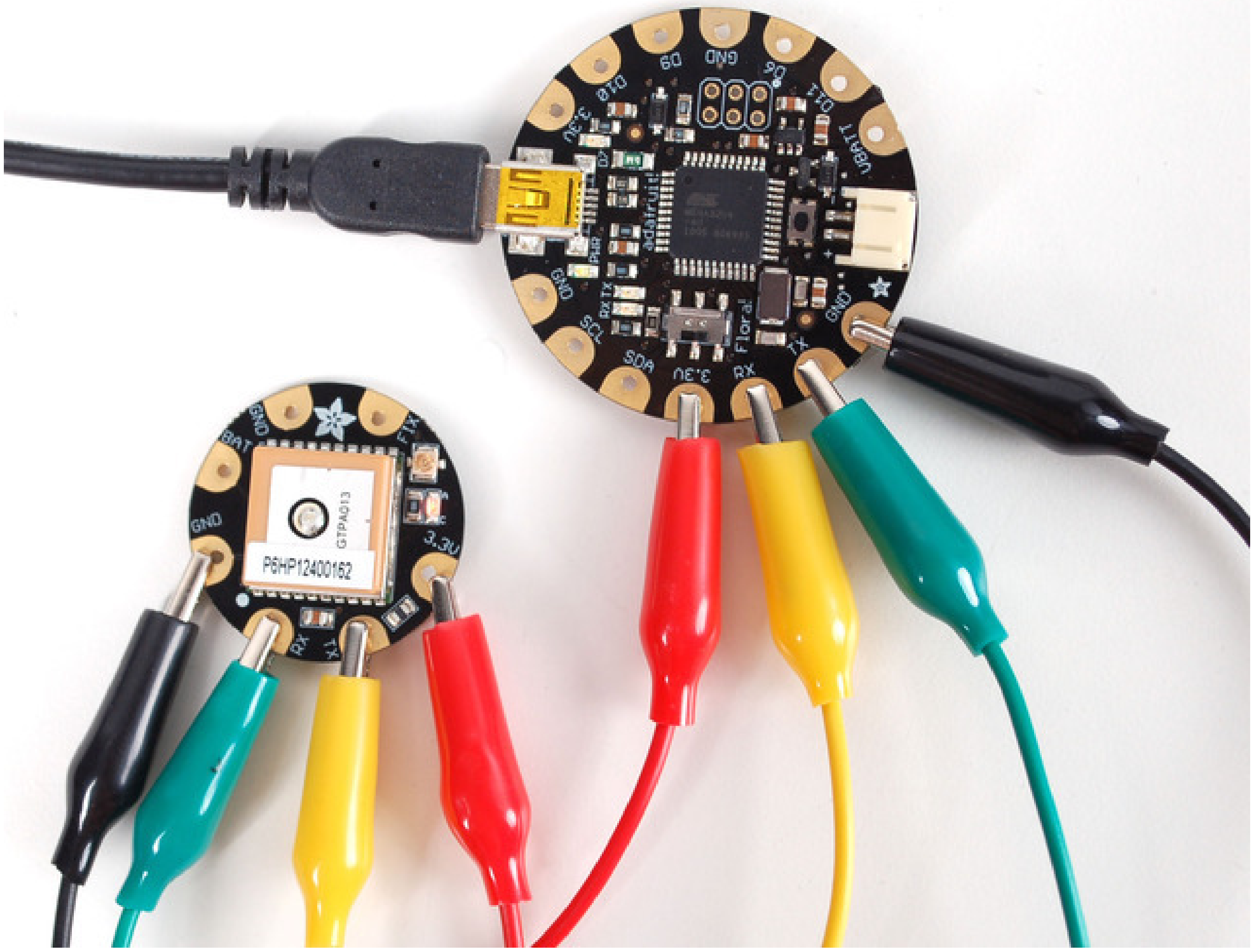


This module is the best way to add a GPS to your wearable project. It's part of the Adafruit Flora series of wearable electronics, designed specifically for use with the Flora motherboard. Installed on the PCB is the latest of our Ultimate GPS modules, a small, super-thin, low power GPS module with built in data-logging capability! This module's easy to use, but extremely powerful:

- -165 dBm sensitivity, 10 Hz updates, 66 channels
- Designed for wearable use with the Flora system
- Only 20mA current draw
- RTC battery-compatible - sew a battery on to create a atomic-precision real time clock
- Built-in datalogging
- >25Km altitude
- Internal patch antenna + u.FL connector for external active antenna
- Fix status LED

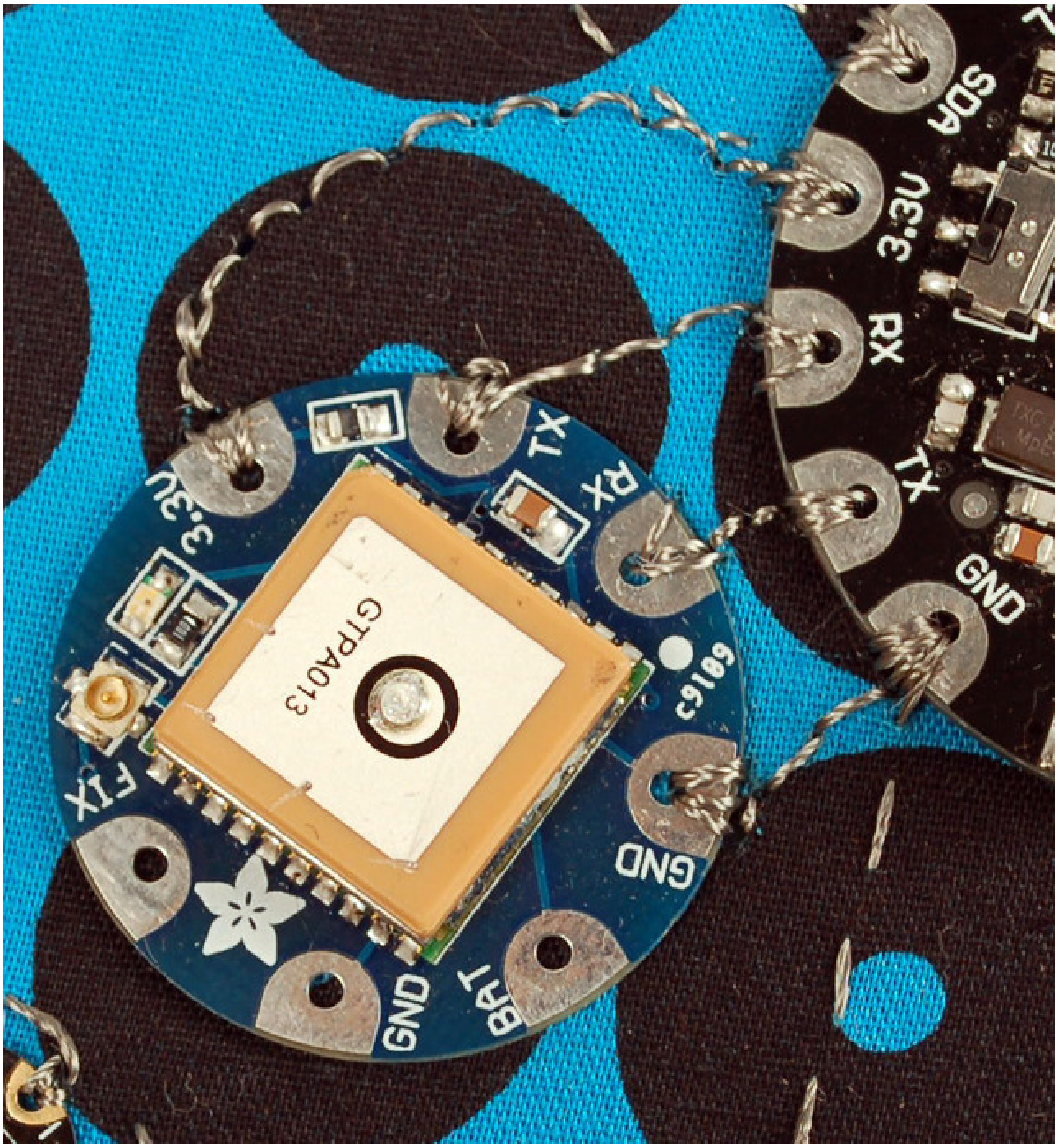
This guide will get you started with the Flora GPS.

## Hook up GPS



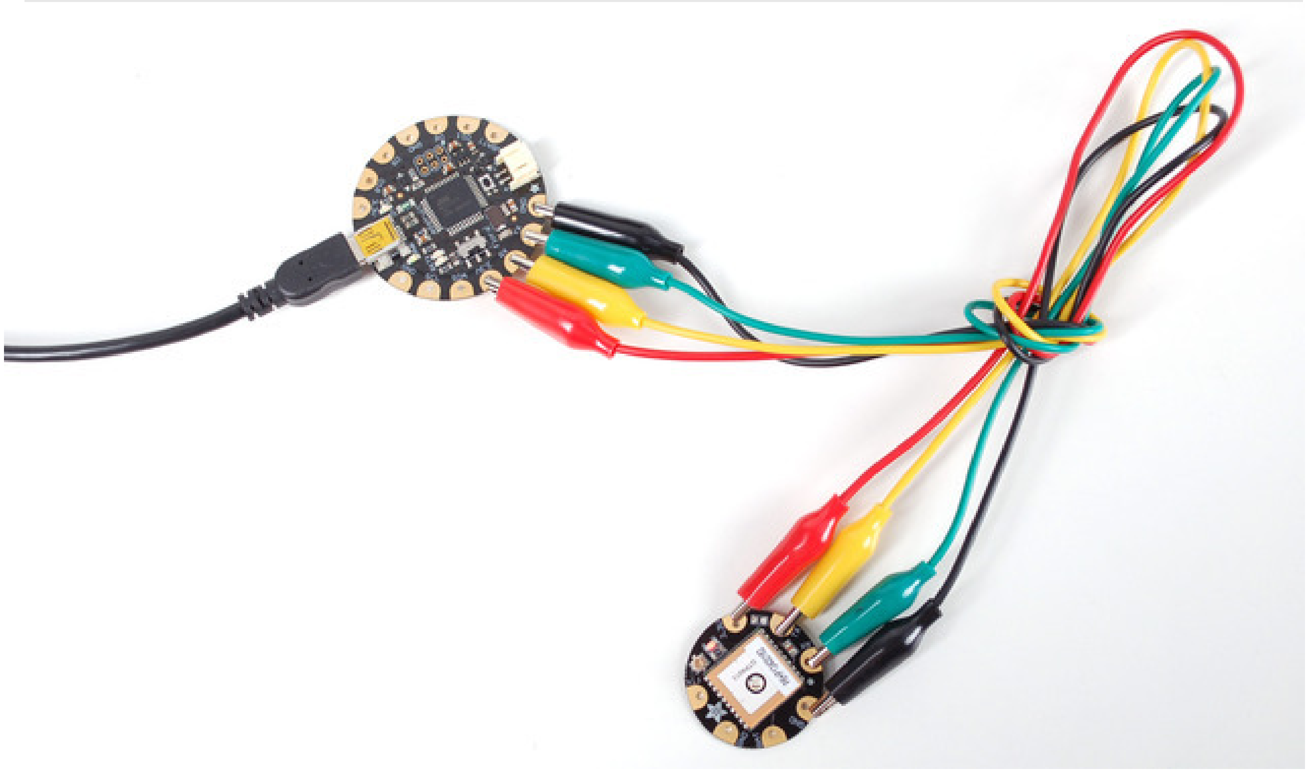
Use alligator clips to connect Flora's 3.3V pad to the 3.3V pad on the GPS. Likewise connect RX to TX and TX to RX, then finally GND to GND.

Unlike the GPS breakout and GPS shield we carry, the Flora GPS is for use with 3.3V power and logic only!



The four connectors are all in a row just to the left of Flora's JST battery connector for easy sewing. But test your project with alligator clips before sewing it in!

## Program FLORA



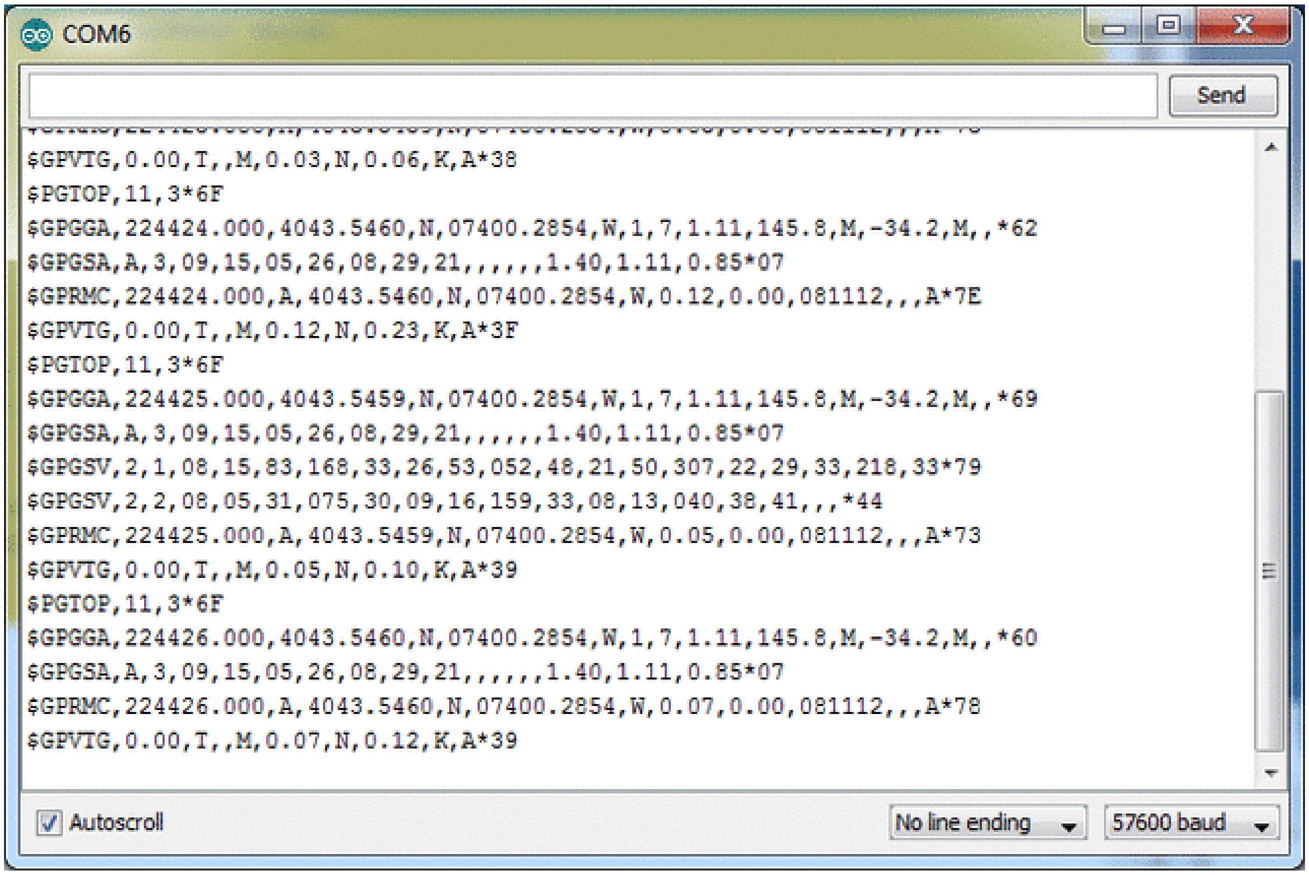
Make sure the USB cable is connecting your computer and Flora.

## Basic Echo Test

We'll start with the most basic test, where we listen to the raw GPS data, to make sure it shows up! Copy and paste this code into a new sketch window and upload it to your Flora

```
// test a passthru between USB and hardware serial

void setup() {
  while (!Serial);
  Serial.begin(9600);
  Serial1.begin(9600);
}



void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    Serial1.write(c);
  }
  if (Serial1.available()) {
    char c = Serial1.read();
    Serial.write(c);
  }
}
```

You should see something like the following from the serial monitor. You may not have as many numbers, but there should be sentences that start with **$GPRMC** and **$GPGGA**, etc. If you see text like that it means your GPS and connection are working fine.



```
$GPVTG,0.00,T,,M,0.03,N,0.06,K,A*38
$PGTOP,11,3*6F
$GPGGA,224424.000,4043.5460,N,07400.2854,W,1,7,1.11,145.8,M,-34.2,M,,*62
$GPGSA,A,3,09,15,05,26,08,29,21,,,,,,1.40,1.11,0.85*07
$GPRMC,224424.000,A,4043.5460,N,07400.2854,W,0.12,0.00,081112,,,A*7E
$GPVTG,0.00,T,,M,0.12,N,0.23,K,A*3F
$PGTOP,11,3*6F
$GPGGA,224425.000,4043.5459,N,07400.2854,W,1,7,1.11,145.8,M,-34.2,M,,*69
$GPGSA,A,3,09,15,05,26,08,29,21,,,,,,1.40,1.11,0.85*07
$GPGSV,2,1,08,15,83,168,33,26,53,052,48,21,50,307,22,29,33,218,33*79
$GPGSV,2,2,08,05,31,075,30,09,16,159,33,08,13,040,38,41,,,*44
$GPRMC,224425.000,A,4043.5459,N,07400.2854,W,0.05,0.00,081112,,,A*73
$GPVTG,0.00,T,,M,0.05,N,0.10,K,A*39
$PGTOP,11,3*6F
$GPGGA,224426.000,4043.5460,N,07400.2854,W,1,7,1.11,145.8,M,-34.2,M,,*60
$GPGSA,A,3,09,15,05,26,08,29,21,,,,,,1.40,1.11,0.85*07
$GPRMC,224426.000,A,4043.5460,N,07400.2854,W,0.07,0.00,081112,,,A*78
$GPVTG,0.00,T,,M,0.07,N,0.12,K,A*39
```

# Getting location data

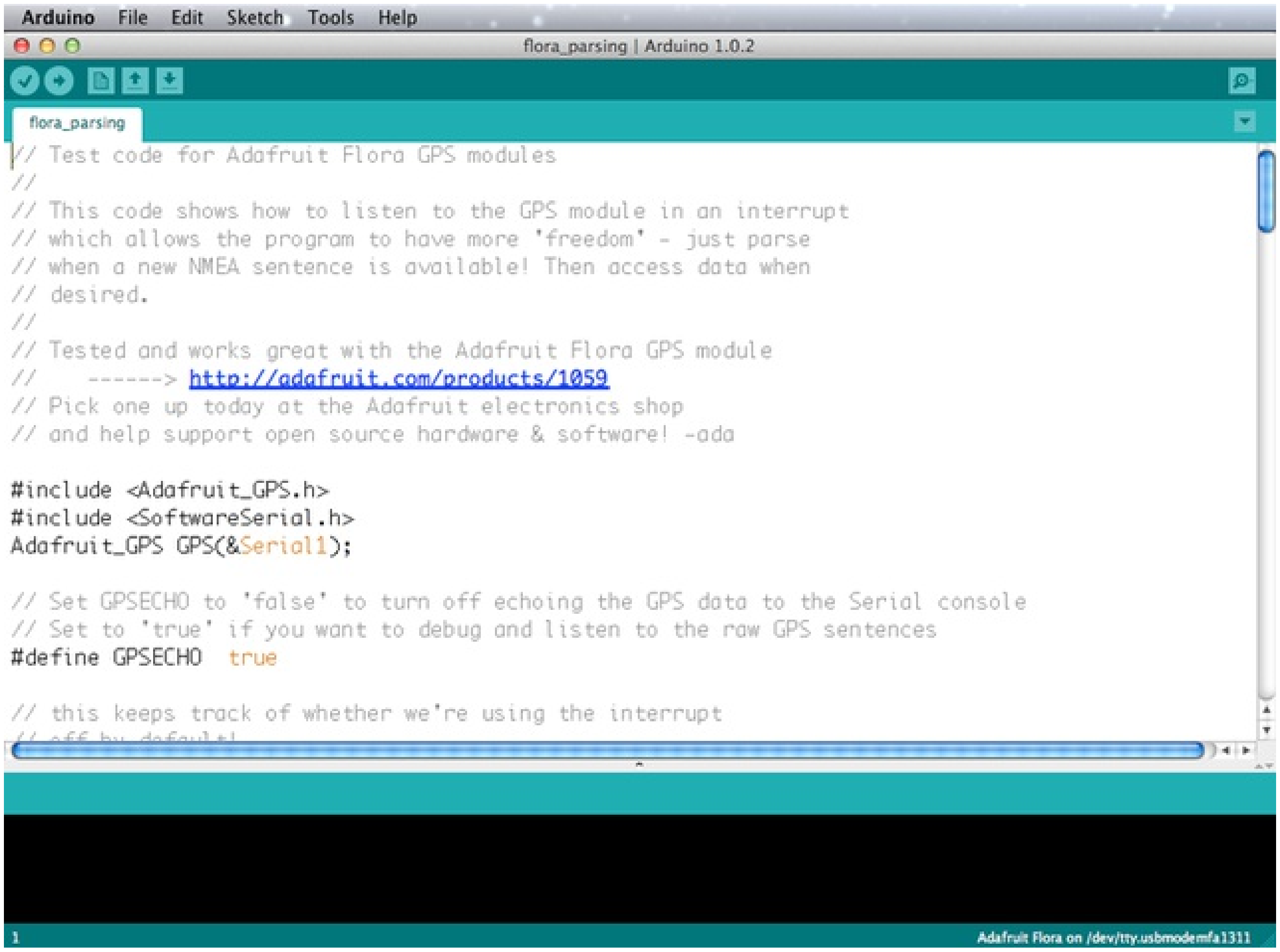
## Detailed GPS Test

Now that we now it basically works, we'll try to get 'fix data' from the GPS. For this, you will have to have the GPS outside. It cannot be inside a building, even if its right at the window. The silver antenna must be pointing up with a clear view of the sky!

First install the Adafruit GPS Library (http://adafru.it/aMm).

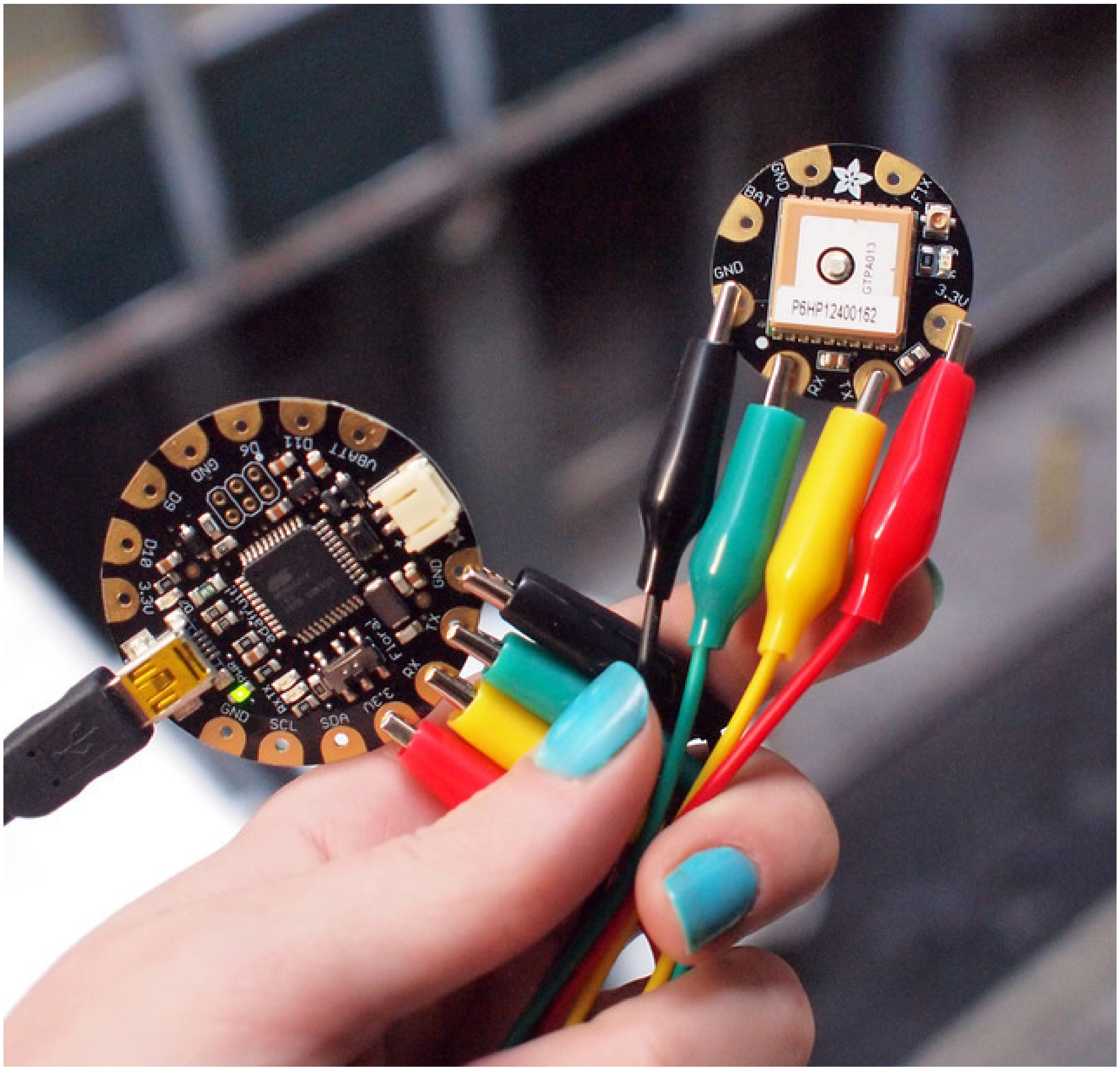
Then upload the following test code to the Flora

```
// Test code for Adafruit Flora GPS modules
//
// This code shows how to listen to the GPS module in an interrupt
// which allows the program to have more 'freedom' - just parse
// when a new NMEA sentence is available! Then access data when
// desired.
//
// Tested and works great with the Adafruit Flora GPS module
//    ------> http://adafruit.com/products/1059
// Pick one up today at the Adafruit electronics shop
// and help support open source hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
Adafruit_GPS GPS(&Serial1);

// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences
#define GPSECHO false

// this keeps track of whether we're using the interrupt
// off by default!
boolean usingInterrupt = false;

void setup()
{
  // connect at 115200 so we can read the GPS fast enough and echo without dropping chars
  // also spit it out
  Serial.begin(115200);
  Serial.println("Adafruit GPS library basic test!");

  // 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
  GPS.begin(9600);
  // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitu
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  // uncomment this line to turn on only the "minimum recommended" data
  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  // For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
  // the parser doesn't care about other sentences at this time

  // Set the update rate
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz update rate
  // For the parsing code to work nicely and have time to sort thru the data, and
  // print it out we don't suggest using anything higher than 1 Hz

  delay(1000);
  // Ask for firmware version
  Serial1.println(PMTK_Q_RELEASE);
}



uint32_t timer = millis();
void loop()                     // run over and over again
{
  // read data from the GPS in the 'main loop'
  char c = GPS.read();
  // if you want to debug, this is a good time to do it!
  if (GPSECHO)
      if (c) Serial.print(c);

  // if a sentence is received, we can check the checksum, parse it...
  if (GPS.newNMEAreceived()) {
    // a tricky thing here is if we print the NMEA sentence, or data
    // we end up not listening and catching other sentences!
    // so be very wary if using OUTPUT_ALLDATA and trytng to print out data
    Serial.println(GPS.lastNMEA());   // this also sets the newNMEAreceived() flag to false

    if (!GPS.parse(GPS.lastNMEA()))   // this also sets the newNMEAreceived() flag to false
      return;  // we can fail to parse a sentence in which case we should just wait for another
  }

  // if millis() or timer wraps around, we'll just reset it
  if (timer > millis())  timer = millis();

  // approximately every 2 seconds or so, print out the current stats
  if (millis() - timer > 2000) {
    timer = millis(); // reset the timer
    Serial.print("\nTime: ");
    Serial.print(GPS.hour, DEC); Serial.print(':');
    Serial.print(GPS.minute, DEC); Serial.print(':');
    Serial.print(GPS.seconds, DEC); Serial.print('.');
    Serial.println(GPS.milliseconds);
    Serial.print("Date: ");
    Serial.print(GPS.day, DEC); Serial.print('/');
    Serial.print(GPS.month, DEC); Serial.print("/20");
    Serial.println(GPS.year, DEC);
    Serial.print("Fix: "); Serial.print((int)GPS.fix);
    Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
    if (GPS.fix) {
      Serial.print("Location: ");
      Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);
      Serial.print(", ");
      Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);

      Serial.print("Speed (knots): "); Serial.println(GPS.speed);
      Serial.print("Angle: "); Serial.println(GPS.angle);
      Serial.print("Altitude: "); Serial.println(GPS.altitude);
      Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
    }
  }
}
```


Arduino File Edit Sketch Tools Help
flora_parsing | Arduino 1.0.2
flora_parsing
Adafruit Flora on /dev/tty.usbmodemfa1311

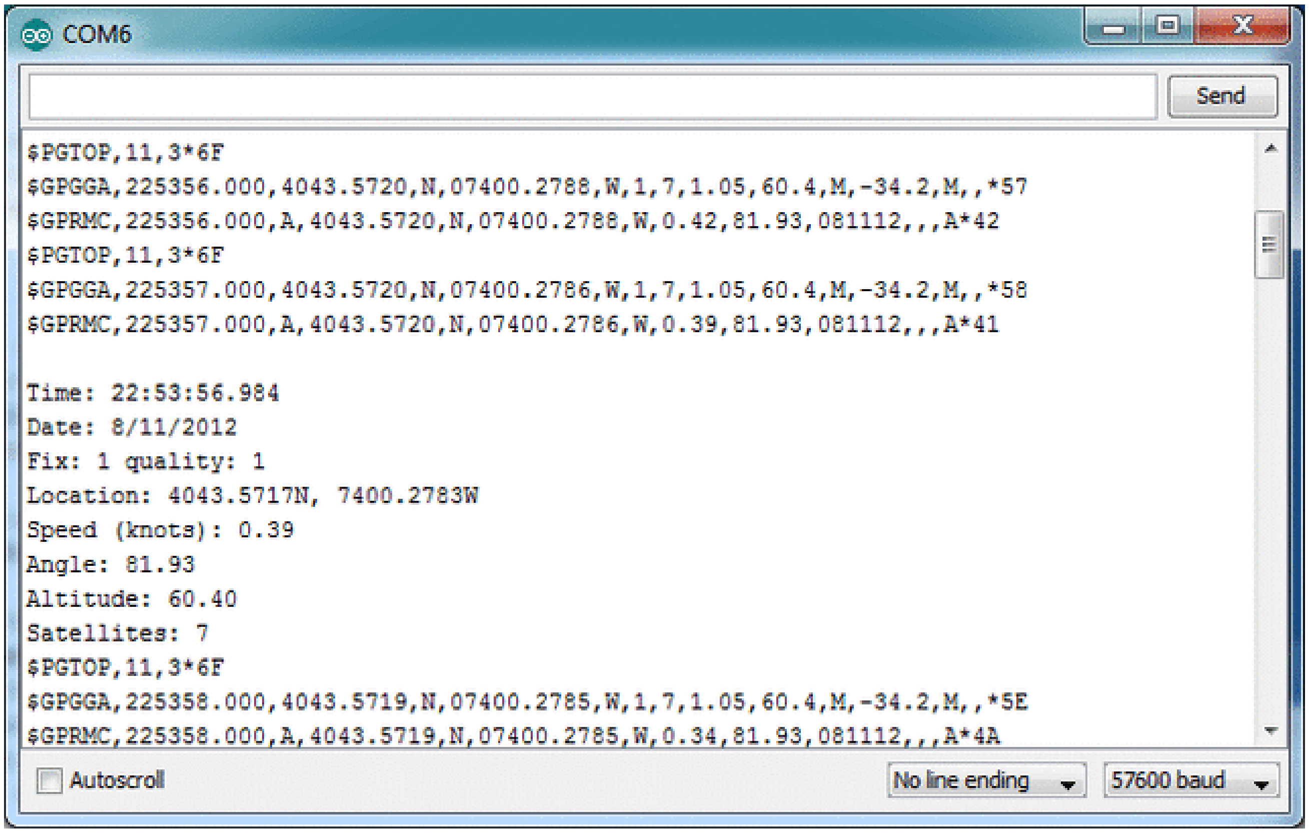
```
// Test code for Adafruit Flora GPS modules
//
// This code shows how to listen to the GPS module in an interrupt
// which allows the program to have more 'freedom' - just parse
// when a new NMEA sentence is available! Then access data when
// desired.
//
// Tested and works great with the Adafruit Flora GPS module
//    ------> http://adafruit.com/products/1059
// Pick one up today at the Adafruit electronics shop
// and help support open source hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
Adafruit_GPS GPS(&Serial1);

// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences
#define GPSECHO  true

// this keeps track of whether we're using the interrupt
```

Check your board and serial port settings (http://adafru.it/aRS) and upload this sketch to your Flora using the Upload button in the IDE.



Place the GPS module (still connected to the Flora)  outside. Once the GPS has located the satellite data, the red LED on the GPS will stop blinking. If you see the LED blinking once a second, it does not yet have a fix! It can take many minutes to get a fix if it doesn't see any satellites immediately. Once it has a fix, you can check the serial monitor for the GPS data, which includes the current date and time in UTC. It will also give you your latitude, longitude and approximate altitude.

GPS modules will always send data EVEN IF THEY DO NOT HAVE A FIX! In order to get 'valid' (not-blank) data you must have the GPS module directly outside, with the square ceramic antenna pointing up with a clear sky view. In ideal conditions, the module can get a fix in under 45 seconds. however depending on your location, satellite configuration, solar flares, tall buildings nearby, RF noise, etc it may take up to half an hour (or more) to get a fix! This does not mean your GPS module is broken, the GPS module will always work as fast as it can to get a fix.



```
$PGTOP,11,3*6F
$GPGGA,225356.000,4043.5720,N,07400.2788,W,1,7,1.05,60.4,M,-34.2,M,,*57
$GPRMC,225356.000,A,4043.5720,N,07400.2788,W,0.42,81.93,081112,,,A*42
$PGTOP,11,3*6F
$GPGGA,225357.000,4043.5720,N,07400.2786,W,1,7,1.05,60.4,M,-34.2,M,,*58
$GPRMC,225357.000,A,4043.5720,N,07400.2786,W,0.39,81.93,081112,,,A*41

Time: 22:53:56.984
Date: 8/11/2012
Fix: 1 quality: 1
Location: 4043.5717N, 7400.2783W
Speed (knots): 0.39
Angle: 81.93
Altitude: 60.40
Satellites: 7
$PGTOP,11,3*6F
$GPGGA,225358.000,4043.5719,N,07400.2785,W,1,7,1.05,60.4,M,-34.2,M,,*5E
$GPRMC,225358.000,A,4043.5719,N,07400.2785,W,0.34,81.93,081112,,,A*4A
```

Now we know where we are. According to the GPS, my location is **4043.5715 N** (Latitude 40 degrees, 43.5815 minutes North) & **07400.2783 W**. (Longitude 74 degrees, 0.2783 minutes West) To look at this location in Google maps, type **+40° 43.5715', -74° 00.2783'** into the google maps search box (http://adafru.it/aMl) . Unfortunately gmaps requires you to use +/- instead of NSWE notation. N and E are positive, S and W are negative.

People often get confused because the GPS is working but is "5 miles off" - this is because they are not parsing the lat/long data correctly. Despite appearances, the geolocation data is NOT in decimal degrees. It is in degrees and minutes in the following format: Latitude: DDMM.MMMM (The first two characters are the degrees.) Longitude: DDDMM.MMMM (The first three characters are the degrees.)