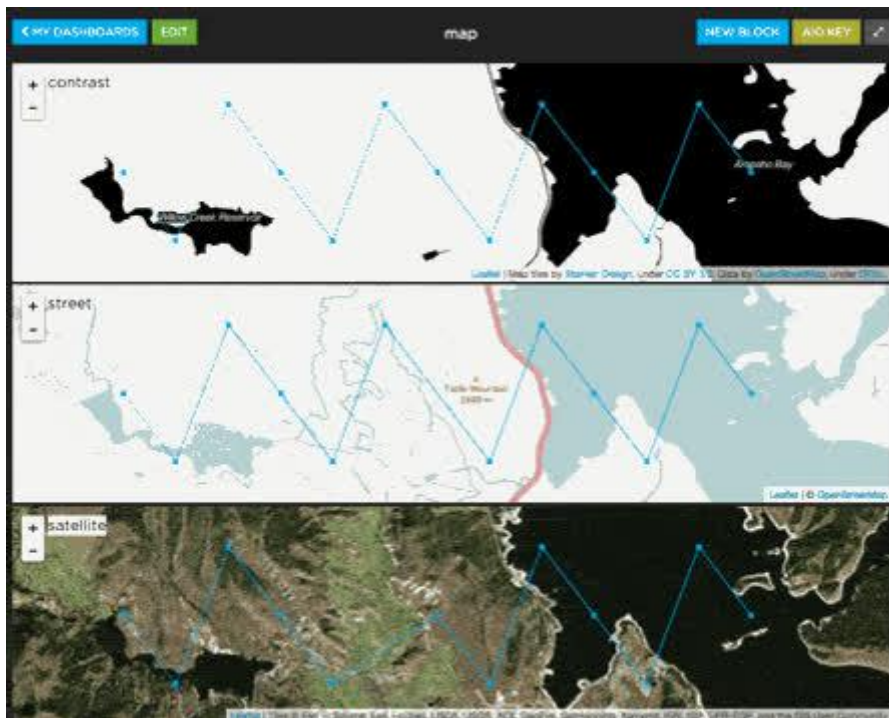




Adafruit IO Basics: GPS

Created by Todd Treece



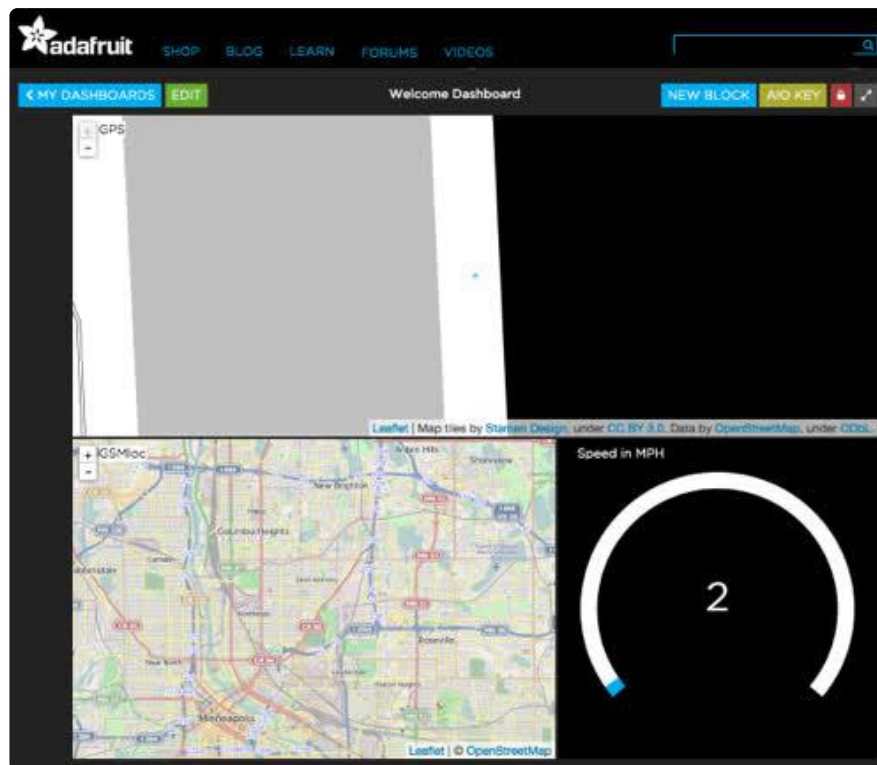
<https://learn.adafruit.com/adafruit-io-basics-gps>

Last updated on 2023-08-29 02:58:03 PM EDT

Table of Contents

Overview	3
Adafruit IO Setup	3
• Creating the Feeds	
• Adding the Blocks to a Dashboard	
Wiring	6
Arduino Code	7
• Arduino Sketch	
• MQTT CSV Location Format	

Overview



This guide is part of a series of guides that cover the basics of using Adafruit IO. It will show you how to map live GPS location information on Adafruit IO using a FONA 808 Shield and Arduino Uno.

If you haven't worked your way through the Adafruit IO feed and dashboard basics guides, you should do that before continuing with this guide so you have a basic understanding of Adafruit IO.

- [Adafruit IO Basics: Feeds \(\)](#)
- [Adafruit IO Basics: Dashboards \(\)](#)

You should also go through the [FONA 808 Shield Setup Guide \(\)](#) before continuing with this guide. If you have went through all of the prerequisites, you are now ready to move on to the Adafruit IO setup. Let's get started!

Adafruit IO Setup

The first thing you will need to do is to login to your [Adafruit IO account \(\)](#) and get your Adafruit IO Key if you haven't already. Click the AIO KEY button on the right hand side of the window to retrieve your key.

YOUR AIO KEY

Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your AIO key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new AIO key, all of your existing programs and scripts will need to be manually changed to the new key.

YOUR AIO KEY

e5bb0c8f44a8e5128f56faf15d87503c580ea751

REGENERATE AIO KEY

A window will pop up with your Adafruit IO key. Keep a copy of this in a safe place. We'll need it later.

Creating the Feeds

You will now need to create a feed called "GPS". If you need help getting started with creating feeds on Adafruit IO, check out the [Adafruit IO Feed Basics guide](#) ().

CREATE A NEW FEED

NAME

GPS

DESCRIPTION

CANCEL

CREATE FEED

Once you have created the GPS feed, you will need to create a feed called "GSMloc".

CREATE A NEW FEED

NAME







GSMloc

DESCRIPTION

CANCEL

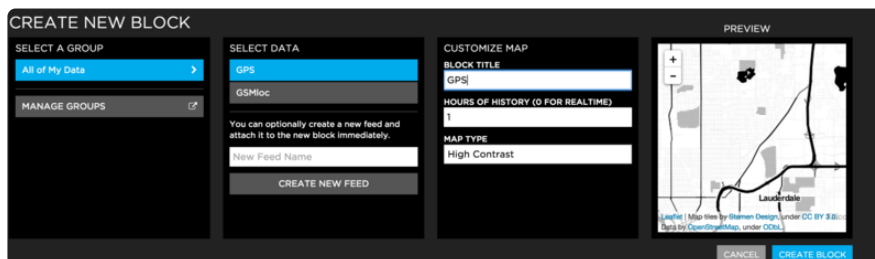
CREATE FEED

After you have finished, your feed list should look like the image shown below.

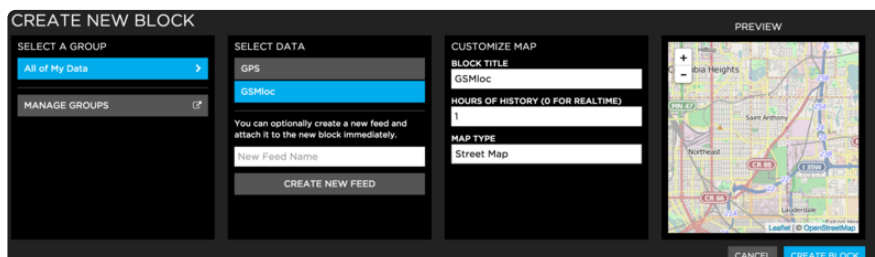
Your Feeds						
SEARCH			CREATE FEED			
ID	NAME	KEY	LAST VALUE	RECORDED	GROUP	ACTIONS
699	GSMloc	gsmloc				  
698	GPS	gps				  

Adding the Blocks to a Dashboard

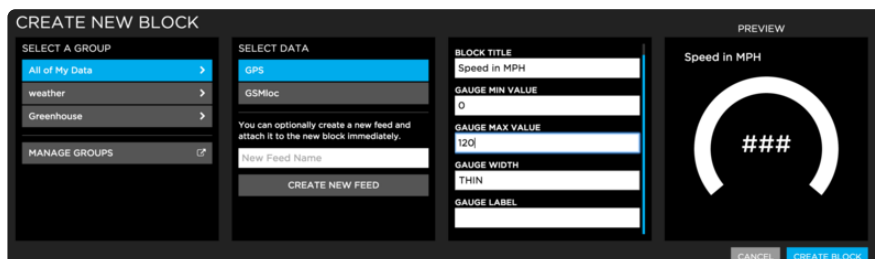
Next, add two Map blocks and one Gauge block to a new or existing dashboard. If you need help getting started with Dashboards on Adafruit IO, check out the [Adafruit IO Dashboard Basics guide](#) ().



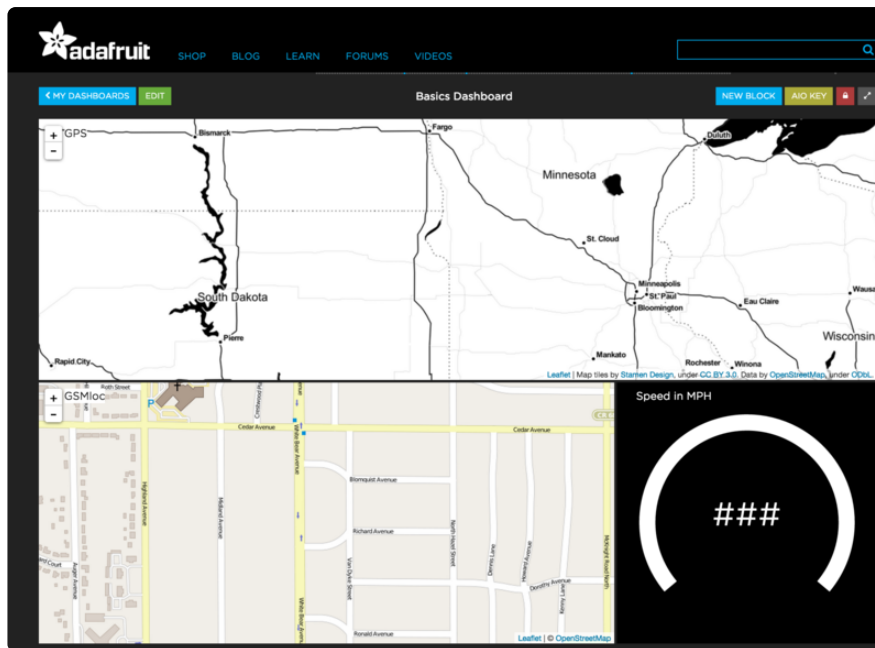
Select the GPS feed for the first map block, and click Create Block to add the new block to the dashboard.



Select the GSMloc feed for the second map block, change the map type to Street Map, and click Create Block to add the new block to the dashboard.



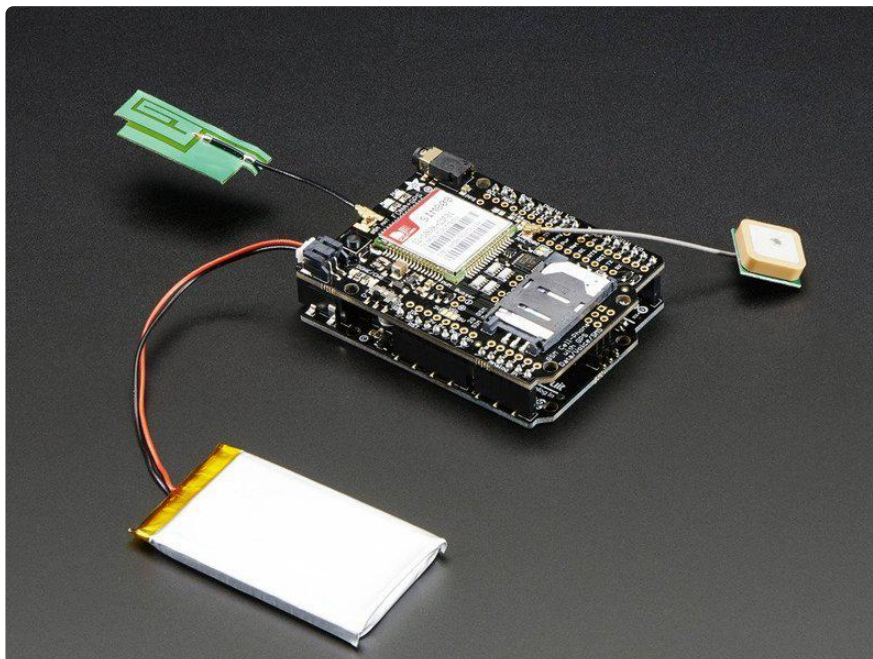
Use the GPS feed as the data source for the gauge block, and set the max value to 120. Click Create Block to add the new block to the dashboard. Arrange the blocks on your dashboard like the image shown below.



Next, we will look at how to connect the FONA 808 Shield the Arduino Uno.

Wiring

This example uses the Adafruit FONA 808 Shield and Arduino Uno to send speed values and GPS info to Adafruit IO. Wiring this project together is fairly simple because we will be using the FONA 808 shield. Assemble the shield as shown in the [FONA 808 Shield Guide \(\)](#), and attach it to the Uno.



Attach the lithium ion battery, passive GPS antenna, and GSM antenna to the FONA 808 shield as shown above. Once that is complete, we are ready to upload the Arduino demo sketch.

Arduino Code

You will need the following Arduino libraries installed to compile the example sketch:

- [Adafruit FONA \(\)](#)
- [Adafruit SleepyDog Library \(\)](#)
- [Adafruit MQTT Library \(\)](#)

The easiest way to install them is by using the [Arduino IDE v1.6.4+ Library Manager \(\)](#).

Arduino Sketch

The Arduino sketch for this project is fairly straight forward. If you haven't downloaded or cloned the [Adafruit IO Basics GitHub repo \(\)](#), you should do that first. We will be using the gps sketch located in the Arduino FONA folder.

Adafruit IO Basics Sketches

You will need to replace the Adafruit IO username and key placeholders in the sketch with your Adafruit IO username and the key that you retrieved in the Adafruit IO Setup section of this guide.

```
#define AIO_USERNAME    "...your AIO username..."
#define AIO_KEY         "...your AIO key..."
```

You will then need to check that the names of your GPS & GSMloc feeds match the feed names defined in the sketch. We will be using the CSV input MQTT topics to send location info along with our data.

```
// Notice MQTT CSV location topics for Adafruit IO
// follow the form: <username>/feeds/<feedname>/csv
const char GSMLOC_FEED[] PROGMEM = AIO_USERNAME "/feeds/gsmloc/csv";
Adafruit_MQTT_Publish gsmloc = Adafruit_MQTT_Publish(&mqtt, GSMLOC_FEED);
const char GPSLOC_FEED[] PROGMEM = AIO_USERNAME "/feeds/gps/csv";
Adafruit_MQTT_Publish gpsloc = Adafruit_MQTT_Publish(&mqtt, GPSLOC_FEED);
```

MQTT CSV Location Format

The Adafruit IO MQTT CSV feed topic expects data published in the following format:

```
sensor_value,latitude,longitude,elevation
```

So if we were sending speed in MPH as our sensor value, the data would look like this:

```
23,39.283277, -76.611818,10
```

Latitude and longitude are expressed in decimal degrees, and elevation is recorded in meters above sea level.

Most of the work of converting GPS and GSM location info in to the proper format is handled by the FONA Arduino library. However, we will need to convert the floating point data returned by the FONA library to the CSV format expected by Adafruit IO.

```
float latitude, longitude, speed_kph, heading, speed_mph, altitude;

// if you ask for an altitude reading, getGPS will return false if there isn't a
3D fix
boolean gps_success = fona.getGPS(&latitude, &longitude, &speed_kph,
&heading, &altitude);

if (gps_success) {
    Serial.print("GPS lat:");
    Serial.println(latitude);
    Serial.print("GPS long:");
    Serial.println(longitude);
    Serial.print("GPS speed KPH:");
    Serial.println(speed_kph);
    Serial.print("GPS speed MPH:");
    speed_mph = speed_kph * 0.621371192;
    Serial.println(speed_mph);
    Serial.print("GPS heading:");
    Serial.println(heading);
    Serial.print("GPS altitude:");
    Serial.println(altitude);

    // snprintf(sendbuffer, 120, "%d,%f,%f,0", x, latitude, longitude);
    // but that doesnt work in arduino
    char *p = sendbuffer;
    // add speed value
    dtostrf(speed_mph, 2, 6, p);
    p += strlen(p);
    p[0] = ','; p++;

    // concat latitude
    dtostrf(latitude, 2, 6, p);
    p += strlen(p);
    p[0] = ','; p++;

    // concat longitude
    dtostrf(longitude, 3, 6, p);
    p += strlen(p);
    p[0] = ','; p++;

    // concat altitude
    dtostrf(altitude, 2, 6, p);
    p += strlen(p);

    // null terminate
    p[0] = 0;

    Serial.print("Sending: "); Serial.println(sendbuffer);
}
```



```

    if (! gpsloc.publish(sendbuffer)) {
        Serial.println(F("Failed"));
        txfailures++;
    } else {
        Serial.println(F("OK!"));
        txfailures = 0;
    }

    Watchdog.reset();
}

```

The FONA 808 also can provide GSM location data. We will be sending this data to the second feed so that we can compare the accuracy of the GPS data to the GSM location data.

```

boolean gsmloc_success = fona.getGSMLoc(&latitude, &longitude);
if (gsmloc_success) {

    Serial.print("GSMLoc lat:");
    Serial.println(latitude);
    Serial.print("GSMLoc long:");
    Serial.println(longitude);

    // snprintf(sendbuffer, 120, "%d,%f,%f,0", x, latitude, longitude);
    // but that doesnt work in arduino
    char *p;

    // paste in 'value' first, just an incrementer for GSMLoc
    itoa(x, sendbuffer, 10);
    p = sendbuffer+strlen(sendbuffer);
    p[0] = ','; p++;

    // concat latitude
    dtostrf(latitude, 2, 6, p);
    p += strlen(p);
    p[0] = ','; p++;

    // concat longitude
    dtostrf(longitude, 3, 6, p);
    p += strlen(p);
    p[0] = ','; p++;
    p[0] = '0'; p++; // 0 altitude
    // null terminate
    p[0] = 0;

    Serial.print("Sending: "); Serial.println(sendbuffer);
    if (! gsmloc.publish(sendbuffer)) {
        Serial.println(F("Failed"));
        txfailures++;
    } else {
        Serial.println(F("OK!"));
        txfailures = 0;
    }
}

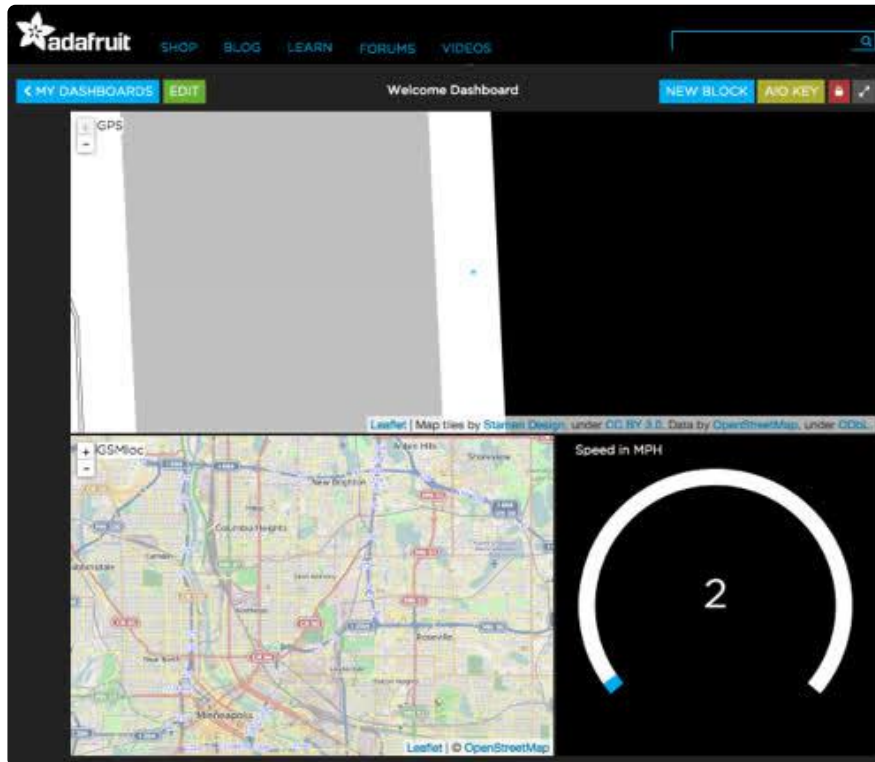
x++;

// wait a couple seconds before starting over
Watchdog.reset();
delay(2000);
Watchdog.reset();

```

```
}
```

When you are finished reviewing the sketch and have finished making the necessary config changes, upload the sketch to your Uno using the Arduino IDE. You should also open up your Adafruit IO dashboard so you can monitor the maps and speedometer.



If everything goes as expected, you will see the location update in the dashboard. You should make sure to open the Arduino IDE's serial monitor if you are having issues with the sketch. It will provide valuable debugging info.

You will need to make sure the FONA's GPS antenna is in clear view of the sky for this demo to work. Flip the FONA shield's switch from Charge to Run, and you should now have a remote location logging device!