adafruit learning system

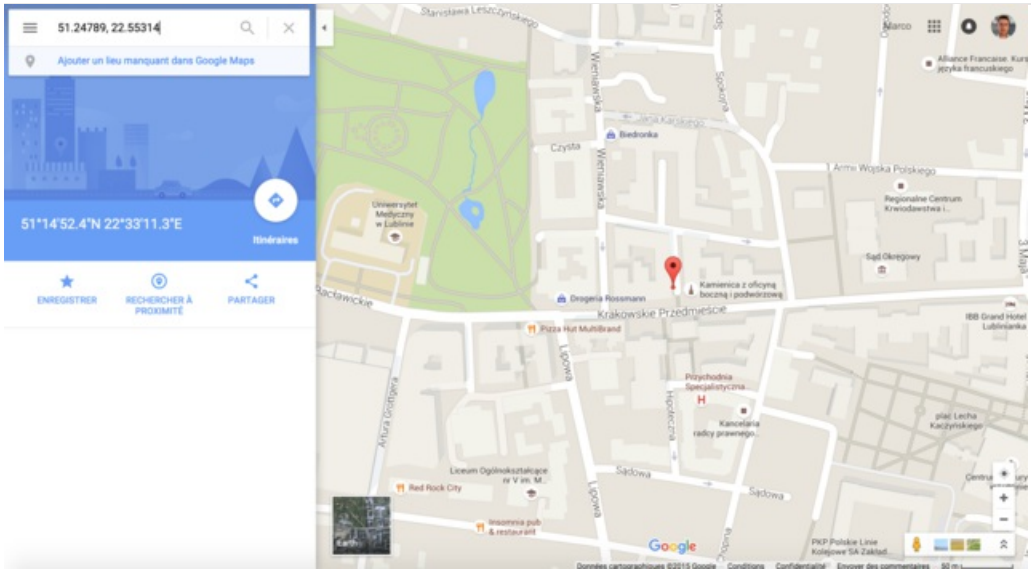
Geofencing with the FONA 808 & Adafruit IO

Created by Marc-Olivier Schwartz

Last updated on 2018-08-22 03:50:28 PM UTC

Guide Contents

Introduction



The Adafruit FONA 808 (http://adafru.it/2542) is an amazing little breakout board that integrates not only a GSM/GPRS chip, but also a GPS on the same board.

In this project, we are going to use the FONA 808 breakout board (http://adafru.it/2542) along with Arduino to make a cool geofencing project. This project can for example be used on objects that you don't want to go after a given distance. For example, you could place this project in a bag, and immediately know if it has been stolen.

Basically, we'll be able to tell where the project is, and calculate the distance between it's current & initial positions. We are actually going to build two variations of the project.

First, we will use a simple on-board alarm system, composed of a LED and a Piezo buzzer, to tell when the fence has been breached. Then, we will use Adafruit IO to track the location of the project in real-time, and immediately know if the fence has been breached from the Adafruit IO dashboard. Let's start!

Hardware Configuration

We are now going to build the project. You will need an Arduino board (I used an Uno here), a FONA 808 breakout board, and a breadboard & jumper wires.

For the FONA board, you will need a set of additional components: a GSM antenna, and also a GPS antenna. You will also need a working GPRS SIM card, with some data credit on it, that needs to be placed inside the FONA 808 breakout board. You will also need a 3.7V LiPo battery to power the breakout board.

For the 'alarm' part, you will need a simple red LED, a 330 Ohm resistor, and a piezo buzzer.

Let's now assemble the project. If you're using the FONA 808 shield, just plug it in. For the breakout, a little wiring is required. First, connect the power supply to the breadboard:

- Connect the **5V pin** from the Arduino board to the red power line on the breadboard
- The **GND** pin to the blue power line.

Then, place the FONA 808 breakout on the breadboard.

- Connect the **VIO** pin to the red power line
- The **GND** & **Key** pins to the blue power line.

After that,

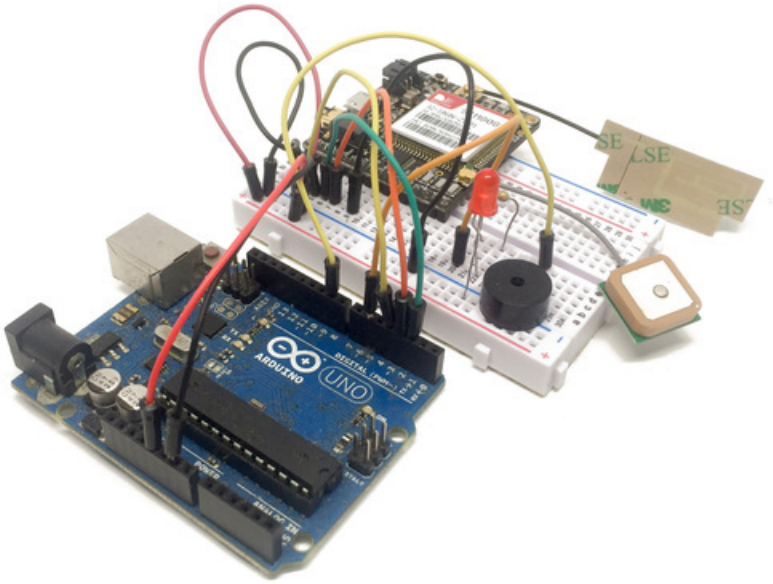
- Connect the FONA **RST** pin to Arduino pin **4**,
- FONA **TX** to Arduino pin **3**
- **RX** to Arduino pin **2**

Also connect the 3.7V LiPo battery, the GPS antenna, and the GSM antenna to the FONA 808 shield or breakout.

Also place the LED on the breadboard, in series with the 330-1K Ohm resistor. Connect it to pin number **6** of the Arduino board, and connect the other end to the ground.

Finally, place the buzzer on the breadboard, and connect it to pin **9** of the Arduino board. Connect the other pin of the buzzer to the ground.

This is how the completely assembled project looks like:



You will also need the latest version of the Arduino IDE for this project, along with the following libraries:

- Adafruit FONA library (https://adafru.it/dDC)
- Adafruit MQTT library (https://adafru.it/fp6)
- Adafruit SleepyDog library (https://adafru.it/fp8)

Testing the GPS

Before we can dive into the core of the project, we are going to test the most important element: the GPS. While testing the project, I found out that sometimes it can be difficult to get a fix on the GPS, even while working near a window. So we are first going to make sure that everything is wired correctly & that we can get the GPS coordinates of our project.

The sketch starts by importing the required libraries:

```
#include <Adafruit_SleepyDog.h>
#include <SoftwareSerial.h>
#include "Adafruit_FONA.h"
```

Then, we declare the variables that are going to store the GPS location:

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

We also need to set the pins on which we connect the FONA 808 breakout board:

```
#define FONA_RX             2   // FONA serial RX pin (pin 2 for shield).
#define FONA_TX             3   // FONA serial TX pin (pin 3 for shield).
#define FONA_RST            4   // FONA reset pin (pin 4 for shield)
```

After that, we create the instance of the FONA 808:

```
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);     // FONA software serial connection.
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);                 // FONA library connection.
```

Then, inside the setup() function of the sketch, we initialise the FONA module:

```
Serial.begin(115200);
Serial.println(F("Geofencing with Adafruit IO & FONA808"));

// Initialize the FONA module
Serial.println(F("Initializing FONA....(may take 10 seconds)"));
fonaSS.begin(4800);

if (!fona.begin(fonaSS)) {
  halt(F("Couldn't find FONA"));
}

fonaSS.println("AT+CMEE=2");
Serial.println(F("FONA is OK"));
```

We also enable the watchdog, that will reset the Arduino board if it gets stuck:

```
Watchdog.enable(8000);
Watchdog.reset();
```

We also need to enable the GPS:

```
fona.enableGPS(true);
```

After that, in the loop() function of the sketch, we get the location from the GPS module:

```
bool gpsFix = fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude);
```

Finally, we print this location on the Serial port:

```
Serial.print("Latitude: ");
printFloat(latitude, 5);
Serial.println("");

Serial.print("Longitude: ");
printFloat(longitude, 5);
Serial.println("");
```

Note that I used a special function here (included in the sketch) to print the location data. This is because the default print function from the Arduino IDE is not precise enough to print the float variables containing the location.

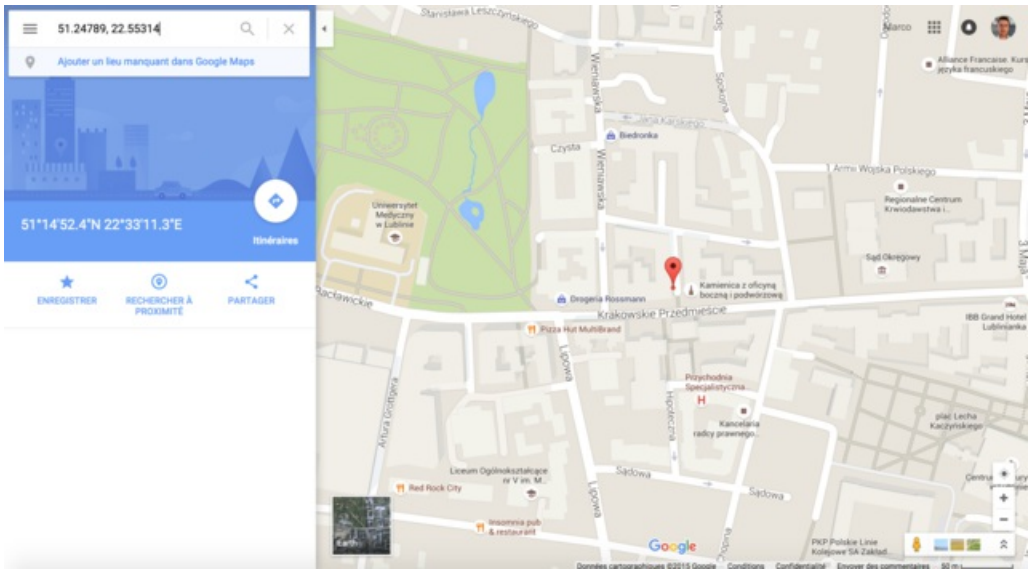
Now, grab the whole sketch from the GitHub repository of the sketch:

https://github.com/openhomeautomation/arduino-geofencing (https://adafru.it/jds)

It's now time to test the GPS! Upload the code to the board, and open the Serial monitor.

You should see that the FONA board is being initialised, and after that you should the GPS location (latitude and longitude) inside the Serial monitor.

To check that this data is correct, you can go to Google Maps and simply enter thoose coordinates:



If it doesn't work, and you get a message saying that you don't have a GPS fix, try to put the project close to a window, or even outside, and wait a moment. You should quickly get a good GPS fix if you have no obstacle on top of your

antenna.

Geofencing Alerts

We are now going to use the GPS to build a first geofencing project. In this first project, we will continuously check the location of the project, and see if we exceeded a given distance that we set in the sketch as well.

If that's the case, we will make the LED blink, and also make the buzzer emit a sound. As this sketch shares a lot with the previous one, I will only talk about what I added here, and you will find the complete sketch inside the GitHub repository of the project.

We first declare some constants & variables for the alarm:

```
// LED & Buzzer pins
const int ledPin = 6;
const int buzzerPin = 9;

// Alarm
int counter = 0;
bool alarm = false;
```

Then, we set the maximum distance that the project can go without raising the alarm. Note that the precision of a civilian GPS like this one is around 10 meters, so I really suggest setting a value which is superior to 20 meters or so:

```
const float maxDistance = 100;
```

We also declare two variables that will contain the initial location of the project:

```
float initialLatitude;
float initialLongitude;
```

By default, we set the alarm to be false:

```
alarm = false;
```

In the setup() function of the sketch, we get a GPS fix to set the initial location of the project:

```
bool gpsFix = fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude);
initialLatitude = latitude;
initialLongitude = longitude;
```

In the loop() function of the sketch, we constantly get the current GPS location, and then calculate the difference between this & the initial location:

```
float distance = distanceCoordinates(latitude, longitude, initialLatitude, initialLongitude);
```

This function, that I won't detail here, is using the Haversine formula:

http://www.movable-type.co.uk/scripts/latlong.html (https://adafru.it/jdt)

This is a well-known formula used to calculate the distance between two GPS coordinates. You can check the details

of the implementation inside the sketch.

After that, we also print this distance inside the Serial monitor:

```
Serial.print("Distance: ");
printFloat(distance, 5);
Serial.println("");
```

If the measured distance exceeded the maximum distance we allowed, we also set the alarm on:

```
if (distance > maxDistance) {
  alarm = true;
}
```

After that, we check if we are in alarm mode or not, and act on the LED & Piezo buzzer accordingly:

```
if (alarm == false) {

  if (millis() - counter > 5000) {
    digitalWrite(ledPin, HIGH);
  }

  if (millis() - counter > 5100) {
    digitalWrite(ledPin, LOW);
    counter = millis();
  }

  noTone(buzzerPin);

}
else {

  if (millis() - counter > 100) {
    digitalWrite(ledPin, HIGH);
  }

  if (millis() - counter > 200) {
    digitalWrite(ledPin, LOW);
    counter = millis();
  }

  tone(buzzerPin, 1000);
}
```

You can find all the code for this part inside the GitHub repository of the project:

https://github.com/openhomeautomation/arduino-geofencing (https://adafru.it/jds)

It's now time to test the project! Upload the code to the Arduino board, and open the Serial monitor. You should see that initially, the distance is equal to zero, or at a small value.

Indeed, note that the precision of the GPS is around 10 meters, so between two measurements the position will change slightly. You can also take the project for a short walk, and see if it raises the alarm!

Integration with Adafruit IO

We are now going to take the same hardware & integrate it with Adafruit IO. We are going to use Adafruit IO to display the location of the project in real time on a map, and also display an alert in case the fence has been breached.

Here as well, the sketch is quite similar to the GPS test sketch, so I will only detail the most important parts that were added.

You will need the following libraries:

```
#include <Adafruit_SleepyDog.h>
#include <SoftwareSerial.h>
#include "Adafruit_FONA.h"
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_FONA.h"
```

If needed, you will also have to enter the GPRS parameters for your SIM card:

```
#define FONA_APN             "internet"
#define FONA_USERNAME        ""
#define FONA_PASSWORD        ""
```

This depends on your operator, so contact them directly if you are not sure about those parameters.

After that, you need to enter your Adafruit IO username & key:

```
#define AIO_SERVER           "io.adafruit.com"
#define AIO_SERVERPORT       1883
#define AIO_USERNAME         "your-adafruit-io-name"
#define AIO_KEY              "your-adafruit-io-key"
```

We also set a name for the Adafruit IO feed that will contain the location data:

```
#define LOCATION_FEED_NAME       "location"
```

We declare an instance of the MQTT library, that we will use to connect to Adafruit IO:

```
Adafruit_MQTT_FONA mqtt(&fona, MQTT_SERVER, AIO_SERVERPORT, MQTT_USERNAME, MQTT_PASSWORD);
```

After that, we declare different feeds that we will use to store data about our project: the location, the current charge of the battery, and the alerts:

```
const char LOCATION_FEED[] PROGMEM = AIO_USERNAME "/feeds/" LOCATION_FEED_NAME "/csv";
Adafruit_MQTT_Publish location_feed = Adafruit_MQTT_Publish(&mqtt, LOCATION_FEED);

const char BATTERY_FEED[] PROGMEM = AIO_USERNAME "/feeds/battery";
Adafruit_MQTT_Publish battery_feed = Adafruit_MQTT_Publish(&mqtt, BATTERY_FEED);

const char ALERTS_FEED[] PROGMEM = AIO_USERNAME "/feeds/alerts";
Adafruit_MQTT_Publish alerts_feed = Adafruit_MQTT_Publish(&mqtt, ALERTS_FEED);
```

In the setup() function of the sketch, we initialise the GPRS connection of the FONA:

```
fona.setGPRSNetworkSettings(F(FONA_APN), F(FONA_USERNAME), F(FONA_PASSWORD));
delay(2000);
Watchdog.reset();
Serial.println(F("Disabling GPRS"));
fona.enableGPRS(false);
delay(2000);
Watchdog.reset();
Serial.println(F("Enabling GPRS"));
if (!fona.enableGPRS(true)) {
  halt(F("Failed to turn GPRS on, resetting..."));
}
Serial.println(F("Connected to Cellular!"));
```

We also connect the project to Adafruit IO via MQTT:

```
int8_t ret = mqtt.connect();
if (ret != 0) {
  Serial.println(mqtt.connectErrorString(ret));
  halt(F("MQTT connection failed, resetting..."));
}
Serial.println(F("MQTT Connected!"));
```

Finally, as we as starting the sketch, we set the alerts feed to 0:

```
logAlert(0, alerts_feed);
```

In the loop() function, instead of setting an alarm, if the distance is breached we set the alerts feed to 1:

```
if (distance > maxDistance) {
  logAlert(1, alerts_feed);
}
```

We also get the current charge of the battery from the FONA breakout board:

```
uint16_t vbat;
fona.getBattPercent(&vbat);
```

Finally, we log the location & the battery charge on Adafruit IO:

```
logLocation(latitude, longitude, altitude, location_feed);
logBatteryPercent(vbat, battery_feed);
delay(5000);
```

You will find all the code inside the GitHub repository of the project:

https://github.com/openhomeautomation/arduino-geofencing (https://adafru.it/jds)

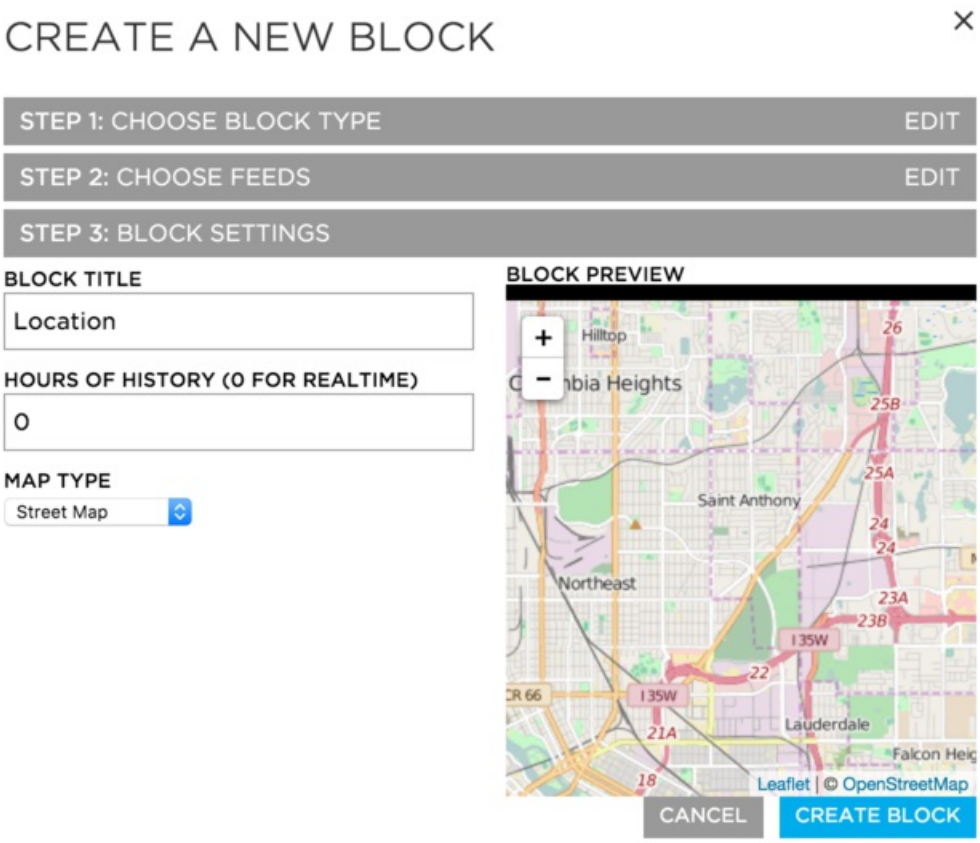
It's now time to test the project! Make sure that you entered your Adafruit IO credentials into the sketch, as well as your GPRS APN settings.

Then, upload the code to the board, and log to Adafruit IO.

You should see that a new 'location' feed has been created, which contains the project GPS location:

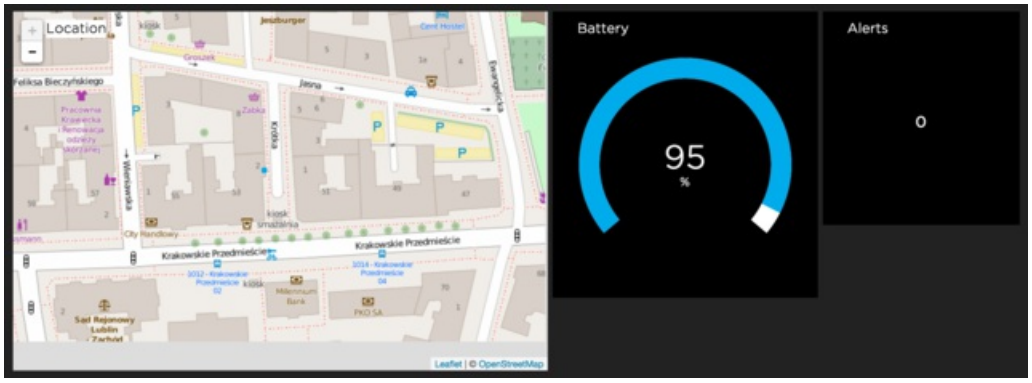
VALUE	CREATED AT	LONGITUDE	LATITUDE	ELEVATION	COMPLETED AT
0	less than 5 seconds ago	22.5531	51.248	0	
0	less than 5 seconds ago	22.5531	51.248	0	
0	less than 10 seconds ago	22.5531	51.248	0	
0	less than 10 seconds ago	22.5531	51.248	0	
0	less than 10 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	
0	less than 20 seconds ago	22.5531	51.248	0	

That's good, but it would be much better with an actual map. Create a new dashboard, and then a map widget linked to the location feed:



You should immediately see the current location of the project inside the dashboard. Finally, add a gauge widget link to the battery, and a text widget linked to the alerts feed.

You should now have a complete dashboard to monitor your geofencing project:



How to Go Further

In this project, we created a geofencing project using Arduino and the Adafruit FONA 808 breakout board. We saw how to set an alarm if the project was going beyond a given distance, and we also connected the project to Adafruit IO.

You can of course go further & mix the two projects together, to have both the alarm and the tracking via Adafruit IO. You could also think for example about forbiden zones on a map that would also raise an alert. Finally, you could also connect the project to IFTTT, to automatically send you notifications in case the fence is breached.