GEAR has one Nova cellular modem and one GPS device. These devices can be used to determine the location of the device on the surface of the Earth. There are many applications in IoT where a determination of location would be invaluable. As part of an IoT application, it might be useful to display the location of the device on a map. Displaying a specific location, provided we have the coordinates, is an easy matter with Google Maps. This lesson will cover two ways to display a location.

With Google Maps there are a number of methods to specify a location. We will use latitude and longitude, values that are provided by a GPS device or the Nova cellular modem. This lesson does not cover how to retrieve latitude and longitude from the device. This lesson covers how to display the location once you have the coordinates.

Below is the HTML code for a web page. The page has a link labeled "Statue of Liberty." When the link is clicked, a Google Map will load into the browser window and you will have a view of the location of the Statue of Liberty. The Statue of Liberty is located at latitude 40.689246 and longitude -74.044530. Check the code carefully to see where and how these numbers are specified. In a text editor, enter the HTML code below and save the file with the extension .html. Then open the file with a web browser and click on the link. The Google Map should load.

```
<!doctype html>
<html>
<head>
<style>
p {
 font-family: verdana;
 font-size: 16px;
 margin-left: 80px;
 margin-right: 80px;
 }
</style>
<title>Statue of Liberty</title>
</head>
<body>
<p> </p>
<p>Demonstration of a link to Google Maps with specification to a
latitude and longitude.</p>
<p><a href="https://maps.google.com/maps?q=40.689246,-74.044530"
target="_blank">Statue of Liberty</a></p>
</body>
</html>
```

Now you can change the location of the map link. Open up Google Maps (https://maps.google.com/) . Navigate to your house on the map. Zoom in on the map so that you can see your house at very close view. Place the mouse cursor over your house and click with the right mouse button. In the menu that opens select What's here? The latitude and longitude of your house are displayed in a box. Copy these numbers and place them in the HTML code in the proper place. Save the file and open it in your web browser. When you click the link, does the map show your house? You may have to zoom the map in a little to be sure (there should be + and – buttons near the bottom-right of the map).

If you are using the Chromium browser on the Raspberry Pi, it will probably be set to lite mode. In that case, you can't determine the coordinates by the method provided above. However, if you carefully center your house in the center of the screen, you can get the coordinates in the address slot of the web browser.

Now let us try a second method for coding the map. In this case, the map loads directly into the web page (you don't need to click a link). Copy the code below and load into a web browser.

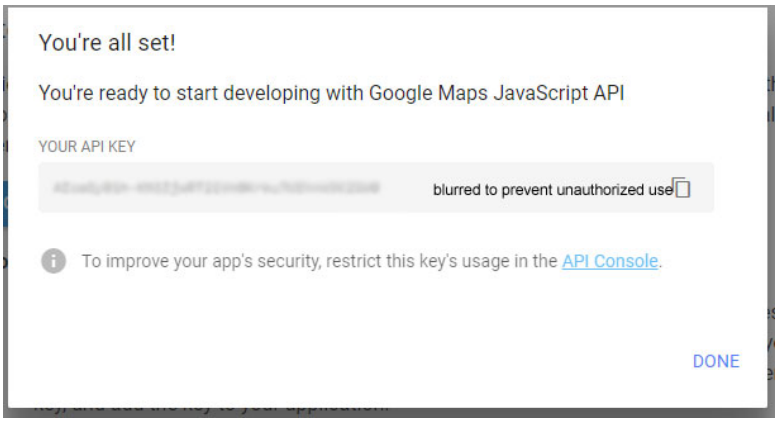
```
<html>
 <head>
 <style>
 #map {
 height: 400px;
 width: 100%;
 }
 </style>
 </head>
 <body>
 <h3>Statue of Liberty</h3>
 <div id="map"></div>
 <script>
 function initMap() {
 var uluru = {lat: 40.689246, lng: -74.044530};
 var map = new google.maps.Map(document.getElementById('map'), {
 zoom: 14, //a value of 19 is a close up view
 center: uluru,
 //mapTypeId: 'satellite'
 //remove comment for above line to display satellite view
 });
 var marker = new google.maps.Marker({
 position: uluru,
 map: map
 });
 }
 </script>
 <script async defer
 src="https://maps.googleapis.com/maps/api/js?key=[put your key here]&callback=initMap">
 </script>
 </body>
</html>
```

You have more control on how the map will be displayed with the above code. You can specify the height and width of the map. You can specify a view, such as roadmap or satellite view. Put the coordinates for your house in above code and try it out. You need a developer key from Google to make this work if the file is served by a web server. You can leave the key blank for testing the page locally on your computer. To get a key, you must have a Gmail account. Here are the steps to get a developer key:

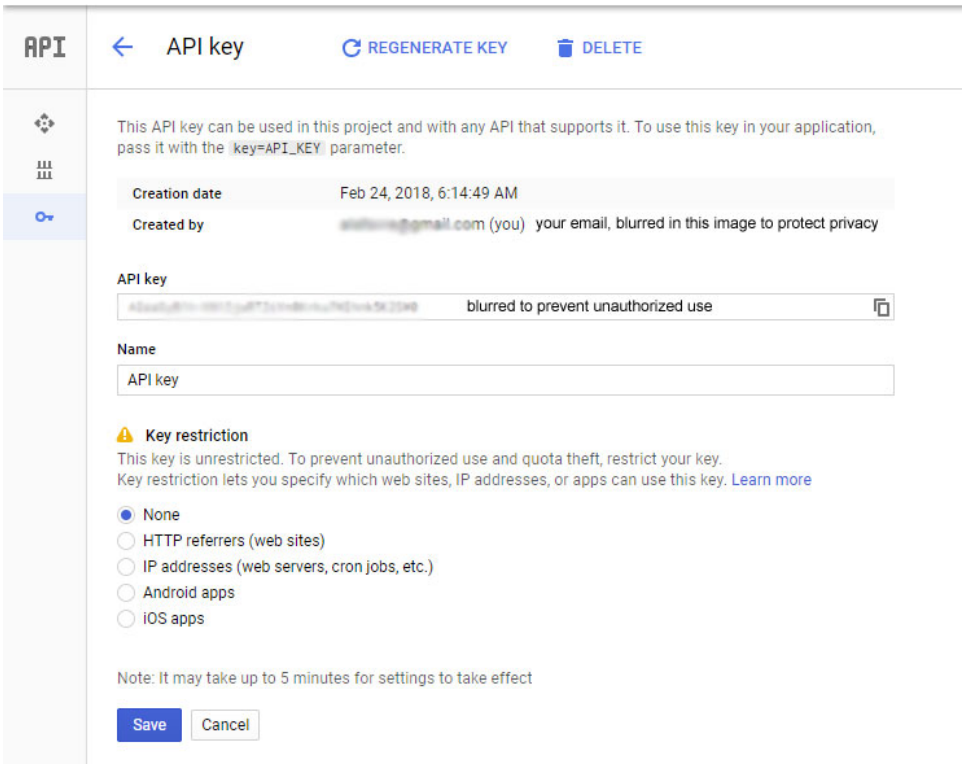
1. First make sure you are signed in to your Google account (open your Gmail)
2. Go to this address: https://developers.google.com/maps/documentation/javascript/get-api-key

3. Click the **Get a Key** button.
4. Click into the Yes radio button, then click on the Next button

5. Your API key will be displayed. Copy this and keep in a safe place. Copy this key into your code to make it work for a web page file stored on a web server. Now click on the API Console link in the box to take a look at the console.



6. Below is a screen copy of the console for my API account. My email is listed on the "Created by" line. In my case the key is unrestricted. If someone looks at the code on my page, they can get my key. You could restrict the web page to a specific IP address here (your computer) to protect the key. Another way is not to share the URL of the web page with anyone or post a link to it.



Visiting your console at a later time

Go to this address: https://console.developers.google.com

Best practices for securely using API keys

When you use API keys in your applications, take care to keep them secure. Publicly exposing your credentials can result in your account being compromised, which could lead to unexpected charges on your account. To keep your API keys secure, follow these best practices:

- **Do not embed API keys directly in code**: API keys that are embedded in code can be accidentally exposed to the public—for example, if you forget to remove the keys from code that you share. Instead of embedding your API keys in your applications, store them in environment variables or in files outside of your application's source tree.
- **Do not store API keys in files inside your application's source tree**: If you store API keys in files, keep the files outside your application's source tree to help ensure your keys do not end up in your source

code control system. This is particularly important if you use a public source code management system such as GitHub.

- **Restrict your API keys to be used by only the IP addresses, referrer URLs, and mobile apps that need them**: By restricting the IP addresses, referrer URLs, and mobile apps that can use each key, you can reduce the impact of a compromised API key. You can specify the hosts and apps that can use each key from the console by opening the Credentials page and then either creating a new API key with the settings you want, or editing the settings of an API key.
- **Delete unneeded API keys**: To minimize your exposure to attack, delete any API keys that you no longer need.
- **Regenerate your API keys periodically**: You can regenerate API keys from the API Console Credentials page by clicking **Regenerate key** for each key. Then, update your applications to use the newly-generated keys. Your old keys will continue to work for 24 hours after you generate replacement keys.
- **Review your code before publicly releasing it**: Ensure that your code does not contain API keys or any other private information before you make your code publicly available.