Lab Three - Interfacing with APIs

An important concept behind many innovations and advancements is to build upon or improve what already exists. In other words, it is in your best interest to use existing systems that accomplish what you plan to build upon or use as a foundation for your own work. This not only saves you time because you can spend it elsewhere working on your own key contributions rather than reinventing something that has already been done, but it can also make your own products or innovations easier to use or interface with since they build upon existing technologies. This is especially true for IoT systems and devices; one of the primary aims for IoT products is not only to allow for a network of communication between physical devices and systems, but to also do so in convenient manner by utilizing existing and flexible infrastructures.

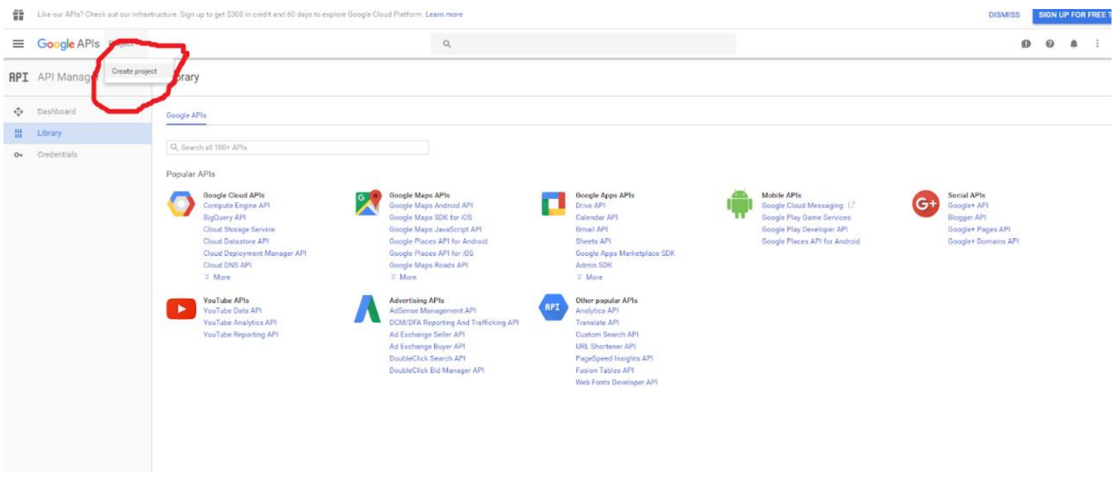
In this lab, we will use Google APIs to add some nifty features to our smartwatch application. An API (application programming interface) makes the primary functions and commands available to the user or developer, so that they can utilize the existing technology or functionalities that the API provides. Just as a computer screen shows the user what is available or the functionalities on that computer, an API shows the developer what features of the technology are available to use. While a user can communicate actions and commands to a computer through peripherals, like a mouse and keyboard, a developer can communicate with many APIs through HTTP, an application layer communication protocol, or through the RESTful API, which uses HTTP as its basis.

For our smartwatch application, we will be using the Google Geolocation API and the OpenWeatherMap API to track the weather in our area and the Twitter API to send tweets with our smartwatch application. To incorporate these APIs, we must do the following on our ESP8266 embedded system.

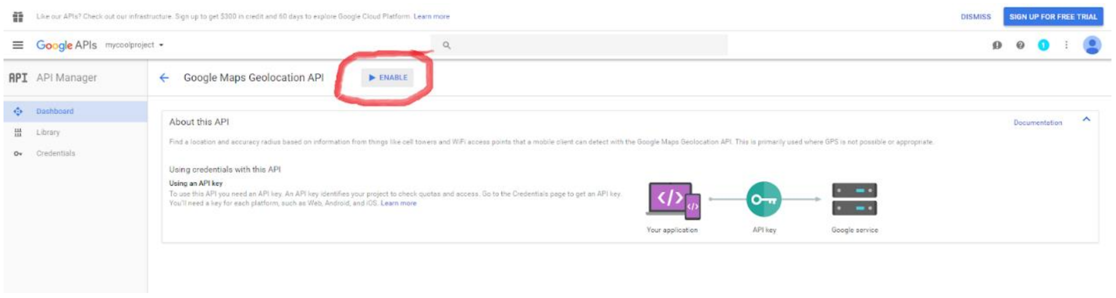
Part One: Geolocation and Weather

Useful Links:
https://developers.google.com/maps/documentation/geolocation/intro
http://openweathermap.org/current
http://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/network_basics.html
http://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/network_tcp.html
http://www.json.org/JSONRequest.html
https://developer.atlassian.com/display/CROWDDEV/JSON+Requests+and+Responses
http://silex.sensiolabs.org/doc/cookbook/json_request_body.html

1. Use an existing or create a Google account and login to the Google API developer's console at https://console.developers.google.com/. Enable the Geolocation API and create an API key. Now that you have the API key, you can interface with the API through the ESP8266 through HTTP POST commands to the host address, https://www.googleapis.com, and by formatting your commands in JSON. This can be accomplished on the ESP8266 through the use of sockets. More information on how to setup the Google APIs are shown below.


Like our APIs? Check out our infrastructure. Sign up to get $300 in credit and 60 days to explore Google Cloud Platform. Learn more

After logging into the developers console, create a new project for your smartwatch with whatever name that comes to mind, as shown above. Once you have created your project search for the Google Geolocation API in the search bar. Once you have found the API, click the "Enable" button to enable the API, as shown below.


Like our APIs? Check out our infrastructure. Sign up to get $300 in credit and 60 days to explore Google Cloud Platform. Learn more

About this API

Find a location and accuracy radius based on information from things like cell towers and WiFi access points that a mobile client can detect with the Google Maps Geolocation API. This is primarily used where GPS is not possible or appropriate.

Using credentials with this API

Using an API key

To use this API you need an API key. An API key identifies your project to check quotas and access. Go to the Credentials page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. Learn more

After enabling the API, go to "Credentials" on the left side of the screen and create an "API key". Now that you have created an API key for the Geolocation API, you can now use Google Geolocationing in your own projects. More information on how to interface with the Geolocation API can be found here: https://developers.google.com/maps/documentation/geolocation/intro. You can find documentation about any of Google's other APIs by looking them up on a search engine.

2. Now that the Google Geolocationing API has been set up on the developer's console, display the coordinates you receive on the OLED display.
3. Feed the coordinates from the geolocation API into the OpenWeatherMap API to receive weather data for your location. Display the current weather on the OLED display. You should at least display the temperature and description (e.g. mostly cloudy). More information about the OpenWeather API can be found at http://openweathermap.org/api and http://openweathermap.org/current.

Part Two: Twitter

Useful Links:
https://apps.twitter.com/
https://thingspeak.com/apps
https://ifttt.com/twitter

Interface with the Twitter API to send tweets from your ESP8266. Make sure to register for a Twitter account if you don't have one and go to https://apps.twitter.com/ to get started. As a side note: there are many other APIs that allow you to send twitter messages, other than the Twitter API; you are welcome to use any of them.   The tweets you send do not have to be typed out and can be preset. There are links to a few other APIs listed above (Thingspeak and IFTTT) in the "Useful Links" section, that also support Twitter commands.

EECS 4764 Lab Three Checkpoints:

1. Interface with the Google Geolocation API and display your coordinates on the OLED display.

2. Feed your coordinates into the OpenWeatherMap API to receive information about the weather in your area; display the results on your OLED display.

3. Create a system to send tweets using the ESP8266.