Web Definitions for API use:

API - ***A****pplication* ***p****rogram* ***i****nterface* (**API**) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.

<http://www.webopedia.com/TERM/A/API.html>

REST (RESTful API) - REpresentational State Transfer is an architectural style, and an approach to communications that is often used in the development of Web services. The use of REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet. The SOAP approach requires writing or using a provided server program (to serve data) and a client program (to request data).

<http://searchsoa.techtarget.com/definition/REST>

SOAP - Simple Object Access Protocol is a messaging protocol that allows programs that run on disparate operating systems (such as Windows and Linux) to communicate using Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML).

<http://searchsoa.techtarget.com/definition/SOAP>

OAuth - an open standard for authorization, commonly used as a way for Internet users to log in to third party websites using their Google, Facebook, Microsoft, Twitter, One Network, etc. accounts without exposing their password. Generally, OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server.

<https://en.wikipedia.org/wiki/OAuth>

JSON - **J**ava**S**cript **O**bject **N**otation.

JSON is a syntax for storing and exchanging data.

JSON is an easier-to-use alternative to XML.

Evaluates to JavaScript Objects

The JSON format is syntactically identical to the code for creating JavaScript objects.

Because of this similarity, instead of using a parser (like XML does), a JavaScript program can use standard JavaScript functions to convert JSON data into native JavaScript objects.

\*\*\* See examples of both JSON and XML on this page \*\*\*

JSON Tutorial:

<http://www.w3schools.com/json/>

<https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/>

XML - XML stands for EXtensible Markup Language

XML is a markup language much like HTML

XML was designed to store and transport data

XML was designed to be self-descriptive

XML is a W3C Recommendation

Full tutorial here: <http://www.w3schools.com/xml/xml_whatis.asp>

POST vs GET -

## **Two HTTP Request Methods: GET and POST**

Two commonly used methods for a request-response between a client and server are: GET and POST.

* **GET** - Requests data from a specified resource
* **POST** - Submits data to be processed to a specified resource

<http://www.w3schools.com/tags/ref_httpmethods.asp>

<https://en.wikipedia.org/wiki/POST_(HTTP)>

<http://blog.teamtreehouse.com/the-definitive-guide-to-get-vs-post>

Idempotence - **Idempotence** is sometimes a confusing concept, at least from the academic definition. From a RESTful service standpoint, for an operation (or service call) to be **idempotent**, clients can make that same call repeatedly while producing the same result.

<http://www.restapitutorial.com/lessons/idempotency.html>

What does this mean to me?

In short an API lets you easily make a app or program, and lets 2 machines work together.

In some of our demos we have used a APIs to communicate with the arduino. And in some of our demos we have used one web service to talk to another web service and communicate with the arduino. Generally this was done using at least 2 different APIs.

REST (RESTful APIs) are generally easier to use, and well documented.

OAuth for the most part has been superseeded by OAuth2 - OAuth2 uses token exchanges, and the token change with each exchange. OAuth was much easier for the arduinos to use.

For the arduino a proxy is generally need for OAuth2.

Json vs XML - Json is much easier to work with, there are Json Libraries for the Arduino

<https://github.com/bblanchon/ArduinoJson> (I’ve not tried this yet, but will soon)

XML arduino library: <https://github.com/adamvr/arduino-sketches/tree/master/libraries/TinyXML> (I’ve not tried this, it was updated 6 years ago, so ???)

Generally if the API you want to use will send you Json formated data you are probably going to be much happier.

GET vs POST - Get is easy to use, just a HTTP URL, POST generally need more information but is used for data that should be secured, and shouldn’t be sent more than once.

GET is a default. POST is usually used for web-forms. The tutorials I posted do pretty good at explaining them.

Because the Arduinos don’t have a lot of power or memory, using a proxy for information exchange is common place. See my other posting about IoT websites.

<https://docs.google.com/document/d/1YxChoV18wzKrkpKDC3RNt7KJYDYBB3yY6sV6OfJv2Dw/edit?usp=sharing>

The demo I plan on doing on Thrs Aug 25, 2016 will be for the Teleduino API.

<https://www.teleduino.org/>

In short you hook up what ever you like to your Arduino UNO or MEGA, you run the “Client Proxy” example code, putting your exchange token(key) in the sketch. And you can control everything about the arduino using a RESTful API, the API returns Json feedback.

It appears that it can do quite a bit, the only down side that I see at this time is it is limited to the ethernet shield. There are 2 different version of the teleduino library one for the UNO and one for the MEGA - this tells me each version works slightly different. There are also android apps written to use teleduino.

One more example of using a API - this is something I made many years ago, it was written in PHP and was made to run from the command line (CLI). What it does is read a location that was sent from a radio to the APRS (Ham Radio stuff) system, the APRS system gates this information into the internet. The code, read the APRS.FI API, gets a latitude and longitude, uses the Yahoo, and Foursquare APIs to get information about businesses your locations you maybe near. IT then formats that information into something human readable, and posts it to twitter and foursquare (updating the foursquare status). It also looks at Yahoo API to get information about open WIFI you maybe near, and sends that to a SMS message using yet another API (a internet to SMS gateway API), it also looks up repeaters you maybe near if you are out of your normal location by 25 miles or so. (That can be set in the script, so you could say within 50 miles, don’t look up repeaters, out of 50 miles look them up) this is done using a repeater database API. It will also SMS those to you. So by using 6 or 7 different services (and APIs) I was able to build one app to do something that none of those companies did. And was able to do it relatively easily and quickly using just some examples from the various website.

It was done using PHP, but given time and energy could have been done using C/C++, Python, Javascript, or anything that can talk HTTP, Json, XML.

<https://bitbucket.org/kd8bxp/aprs2twitter>