ESP8266 WIFI Arduino ThingSpeak Demo

A really good source of information on these boards can be found here:
https://nurdspace.nl/ESP8266 and here: http://www.electrodragon.com/w/Wi07c
another (more or less) resource is: http://www.xess.com/blog/esp8266-resources/
Xess does have some information about the device, but it's mostly links to more information, the writter does point out the different versions (I talk about below) and some of his frustration with it

Both of the above sites have more or less the same information, the Wi07c site has some better pin-out information, while the ESP8266 site has some good general information.

There are apparently 3 version of this board, from the Wi07c site I gather the 3rd revision is physically different. But it looks like version 1 and 2 look identical, but work a little different.

So if you end up with a version one board it looks like you only need to hook up 4 wires, while the version 2 board hooks up 5 wires.
Version 1: 3v Power, Ground, TX and RX
Version 2: 3v Power, Ground, TX and RX and a 2nd 3v Power (This turns it on)
Version 3: ?

With that being said, there must be some variations even better versions. For instance the ESP8266 site says the baud rate is 57600, while a site that I came across claims the baud rate to 115200. But Mine (both of them) only work at 9600 baud - mine also dont seem to respond to the reset command the same as what ESP8266 claims.

One of the sites says that you can put custom firmware on these, and it is possible that they are running custom firmware - which is what I suspect, since one of the 'errors' I did produced a Chinese website (that is probably the manufacture of this chip)

So to start with I found this sketch over at seeedstudio.com
http://www.seeedstudio.com/wiki/WiFi_Serial_Transceiver_Module
Below is the un-edited sketch

```
#include <SoftwareSerial.h>
  #define SSID "xxxxxxxx"
  #define PASS "xxxxxxxx"
  #define DST_IP "220.181.111.85" //baidu.com
  SoftwareSerial dbgSerial(10, 11); // RX, TX
  void setup()
  {
    // Open serial communications and wait for port to open:
    Serial.begin(57600);
    Serial.setTimeout(5000);
    dbgSerial.begin(9600); //can't be faster than 19200 for softserial
  dbgSerial.println("ESP8266 Demo");
  //test if the module is ready
  Serial.println("AT+RST");
  delay(1000);
  if(Serial.find("ready"))
  {
    dbgSerial.println("Module is ready");
  }
  else
  {
    dbgSerial.println("Module have no response.");
    while(1);
  }
  delay(1000);
  //connect to the wifi
  boolean connected=false;
  for(int i=0;i<5;i++)
  {
    if(connectWiFi())
    {
      connected = true;
      break;
    }
  }
  if (!connected){while(1);}
  delay(5000);
  //print the ip addr
  /*Serial.println("AT+CIFSR");
  dbgSerial.println("ip address:");
  while (Serial.available())
  dbgSerial.write(Serial.read());*/
  //set the single connection mode
  Serial.println("AT+CIPMUX=0");
}
void loop()
{
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += DST_IP;
  cmd += "\",80";
  Serial.println(cmd);
  dbgSerial.println(cmd);
  if(Serial.find("Error")) return;
  cmd = "GET / HTTP/1.0\r\n\r\n";
  Serial.print("AT+CIPSEND=");
Serial.println(cmd.length());
if(Serial.find(">"))
{
  dbgSerial.print(">");
  }else
  {
    Serial.println("AT+CIPCLOSE");
    dbgSerial.println("connect timeout");
    delay(1000);
    return;
  }
  Serial.print(cmd);
  delay(2000);
  //Serial.find("+IPD");
  while (Serial.available())
  {
    char c = Serial.read();
    dbgSerial.write(c);
    if(c=='\r') dbgSerial.print('\n');
  }
  dbgSerial.println("====");
  delay(1000);
}
boolean connectWiFi()
{
  Serial.println("AT+CWMODE=1");
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  dbgSerial.println(cmd);
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("OK"))
  {
    dbgSerial.println("OK, Connected to WiFi.");
    return true;
    }else
    {
      dbgSerial.println("Can not connect to the WiFi.");
      return false;
    }
  }
```

What this does, is connects the ESP8266 to a wifi network, and quarries a website (in this case baidu.com which is a Chinise search engine) What you will note thou is it uses the websites IP address in this case: 220.181.111.85
This doesn't use DNS which might be wise since its small, and DNS would take some resources it doesn't have or need.
So if you only know the URL of the website you will need to convert it to an IP, there are a number of websites that do this, just google it - but here is the one I used:
http://tejji.com/ip/url-to-ip-address.aspx or
http://whatismyipaddress.com/hostname-ip
Just put the URL in the form, and click "convert to IP"
Here are a few popular sites:

Google.com (Has multiple servers, and IP address):

74.125.228.36

74.125.228.35

74.125.228.38

74.125.228.41

74.125.228.37

74.125.228.40

74.125.228.46

74.125.228.34

74.125.228.33

74.125.228.32

74.125.228.39

eBay.com (again multiple servers):

66.211.160.86

66.211.160.87

66.135.216.190

This works sort of, but I haven't been able to get it to read just a text file or json or xml generally I get an error, but the error doesn't make much since.

I found this in a hackaday instruction: (it looks like it was written for a different version of the device, but I was able to get it to work - the “ascii” website (retro style) is down and gone now thou) :-(
http://hackaday.io/project/3072/instructions
and http://dunarbin.com/esp8266/retroBrowser.ino
The above sketch is very different then what I used before, while it was written for a different version of the device, I took and copied the IP address over to my “working” sketch and found that the site was gone.

The hackaday does point out this wasn’t meant to be a web-browser, but with a lot of work and time it can work as a retro style browser.
They also point out why you have to hook it up to the hardware serial port on an Arduino - their reason was the device is setup for 115200bps and the software serial can’t handle this speed. (So again it was written for a different version, and firmware)
The latest firmware is at 9600bps

So here are a couple of other sites that attempt to “browse” a website
http://zeflo.com/2014/esp8266-weather-display/ (I couldn’t get this to work at all, the Json library it uses I couldn’t find, the one I did find wasn’t the correct one) But this looked promising if the right library could be found.

http://www.seeedstudio.com/wiki/WiFi_Serial_Transceiver_Module
This is what I used to finally get things to work (I think I put this link up above too)

So if it doesn’t browse the internet, what good is it?

Well How about sending temperature (or any sensor data) to a IoT site?
Using a DHT11 sensor and Thingspeak it was easy to send updates with this

http://ruten-proteus.blogspot.tw/2014/11/internet-of-thing-arduino-esp8266.html
The above site is almost completely in Chinese and didn’t translate very well, but there was enough to figure out what they did and get it to work. And it works quite nicely.
The code is loosely based the seeedstudio code and this code found at instructables.com
http://www.instructables.com/id/ESP8266-Wifi-Temperature-Logger/?ALLSTEPS
The instructables code is in English and is explained pretty good, but it doesn’t work, there is some type of problem with reading the DS18B20 that a few people (including myself) are having with it.
in the Comments section of the write up the Chinese man points out a few of the problems and directs people to his code with the DHT11 which works better.

As I said this is loosely based off the seeestudio code - since the seeestudio sends a url, and url can be sent so updating a simple API is very simple and easy to do.

The possibilities are endless for using this are endless, from simple sending data to a IoT website, to updating/sending SMS messages to your cell phone.