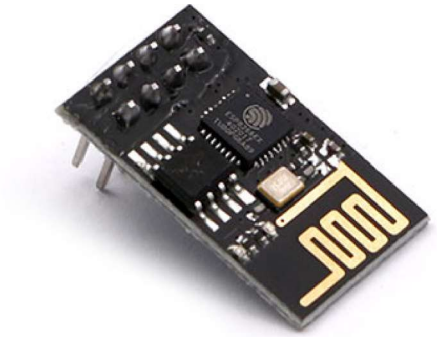
الحمـــادي للألكترونيـــات
ALHAMMADI FOR ELECTRONICS

Arduino WIFI
Model:ESP01-8266
User Manual



Overview:

The ESP8266 based wifi breakout boards are becoming more popular with Makers due to a low cost and a powerful, programmable microcontroller on-board.
The cost is a magnitude lower than solutions previously used including Arduino+Wifi Shield or an Arduino Yun.
To quote Make publisher Brian Jepson:
“This is inexpensive enough to be very much in the territory of ‘thousands of sensors-launched-out-of-acannon’-cheap.”

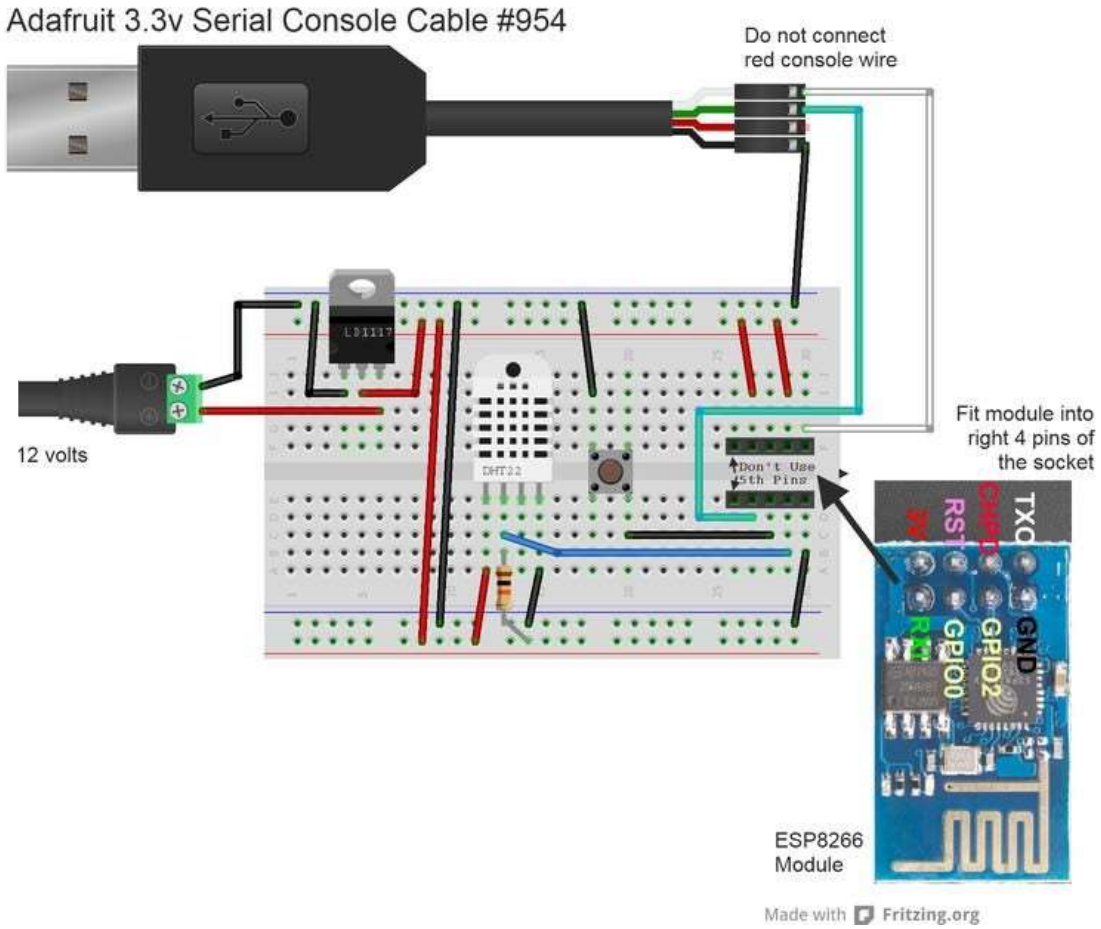
Couple the ESP8266 with one of the inexpensive DHT series digital temperature and humidity sensors and we have a project that may literally be deployed anywhere to broadcast sensor data.
The broadcasting used in this tutorial is using the ESP8266 web server code and respond to web requests (like in a browser or a web client) to return temperature and humidity data (in a REST type format). This project demonstrates the programming of the ESP8266 ESP-01 module with the Arduino IDE and interfacing with a DHT temperature/humidity sensor. Using other sensors and their corresponding libraries, other electronics may be interfaced with the ESP8266 and monitored via wifi.
Wiring

This project is presented on a breadboard as it best demonstrates the connections and how the digital pins might connect to other types of sensors. A final project may only have power, a regulator, the ESP-01, and a sensor, which could fit in a very small container.
Parts List

• 412 ARDUINO WIFI ESP8266
• Voltage Regulator LM1117
• 412 ARDUINO SENSOR HUMIDITY DHT11
• Female / Male Jumper Wires
• jumper wire

The ESP8266 is a 3.3 volt device only. It can draw over 300 milliamps at some peak operations. To give it a safe margin, the LM1117-3.3 regulator is safe, able to supply 800 milliamps when it needs to, cool at 500 milliamps. So you can connect a 3.7 volt LiPo battery, 5 volt "Cell Phone Recharger" battery, or 9 volt battery. You may also use a 5 volt "wall wart" power supply. You might also consider a power switch. If you have noisy power, like a poor quality cell charger, etc. place a capacitor between power and ground on the input and output. 0.1 microfarad would be typical, 10 microfarad electrolytic capacitor (which have + and - leads) for a more noisy supply.

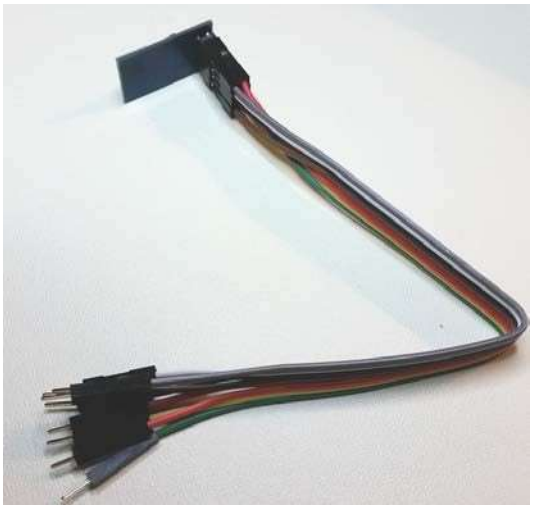
Many web tutorials show the ESP8266 being connected via generic FTDI USB to Serial device. 5 volt signal levels can harm the ESP8266. This may be mitigated via level shifting or a voltage divider on the ESP8266 receive (RXI) pin for such devices. The FTDI Friend is safe as the default transmit and receive signal level is 3.3 volts if left on the default on the back.

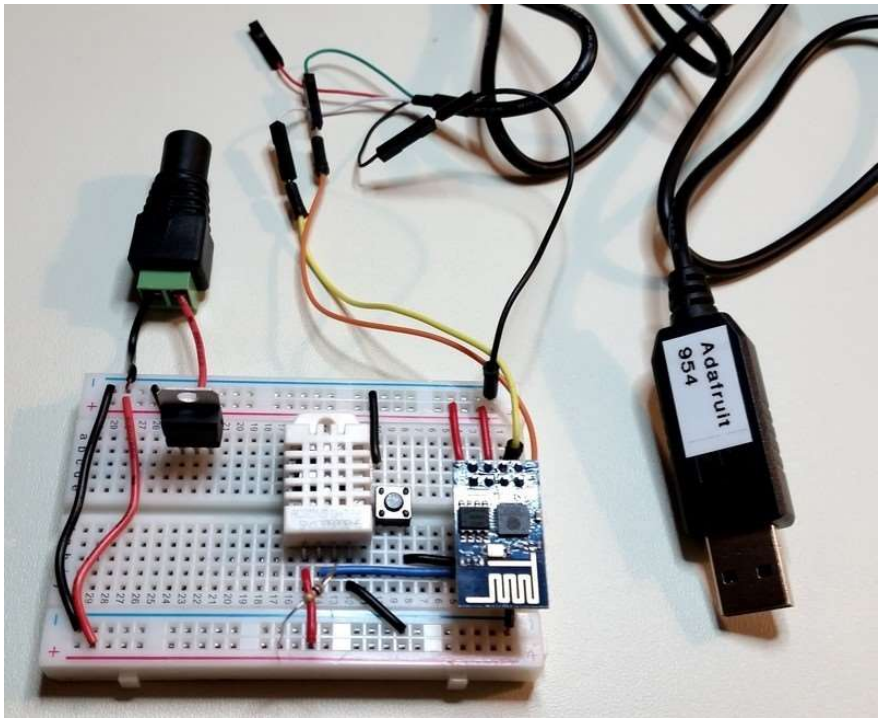


The serial-TTL converter runs at 3.3 volt signal levels so it's safe (safe for the sensitive Raspberry Pi also). The only thing is do not connect the red 5 volt wire, it is not needed.

If you do not have a 4x2 or 5x2 socket, you can use female to male rainbow connector wires like the one below.

Here is the complete project on the breadboard (using the breadboard socket for the ESP8266).





This project uses the ESP8266 board add-in for the Arduino 1.6.x development environment.
The best way to set this up is to follow guide on adding support for boards like the ESP8266 for the Arduino IDE. This would be a the add-in for the Arduino 1.6.4+ IDE Boards menu. The ESP8266-arduino project online does have a prepackaged IDE with ESP8266 built-in. ***Required Libraries:***

When you install the ESP8266 support to the Arduino 1.6.4+ IDE, there should several ESP8266 examples installed. One, ESP8266Webserver, has much of what is needed. The libraries ESP8266WiFi, WiFiClient, and ESP8266WebServer libraries may be pre-installed.
The other library needed is the DHT library.
Begin by downloading the DHT library from software.. Rename the uncompressed folder DHT and make sure that it contains the dht.cpp file and others. Then drag the DHT folder into thearduinosketchfolder/libraries/ folder. You may have to create that libraries sub-folder if it doesn't exist. You must restart the Arduino IDE programming program for the library to be available.

Program:

Copy the program below and save as DHTServer.ino
Copy Code

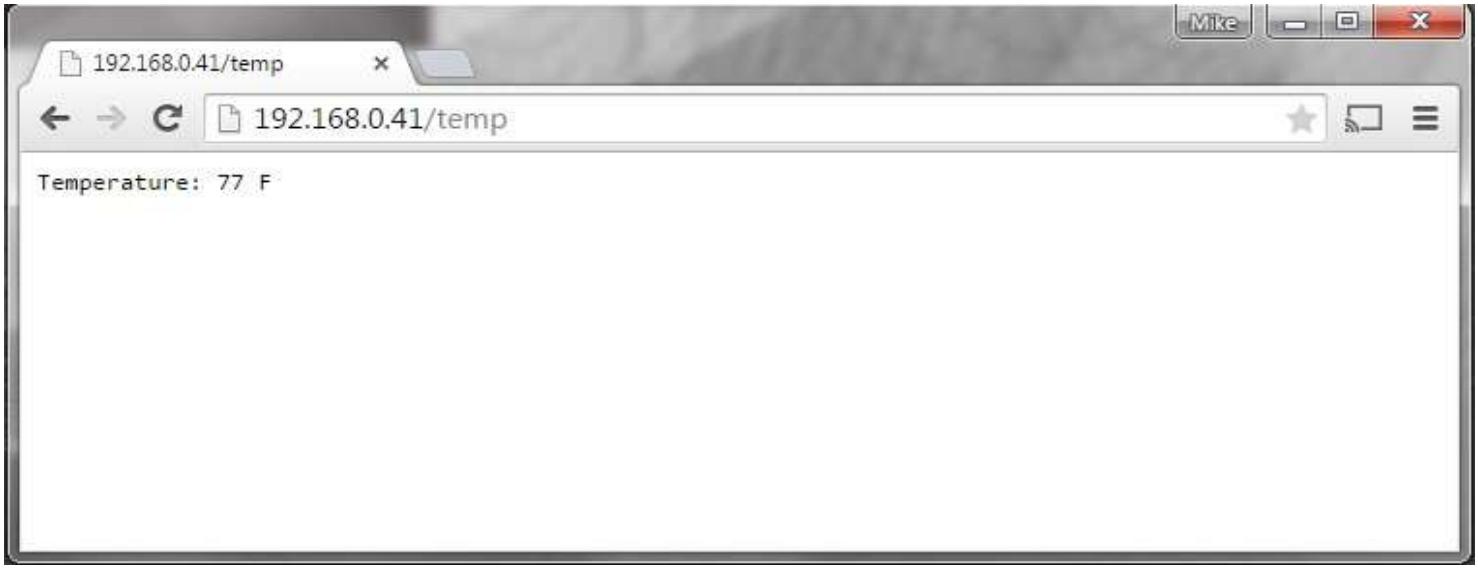
```
1.  /* DHTServer - ESP8266 Webserver with a DHT sensor as an input
```

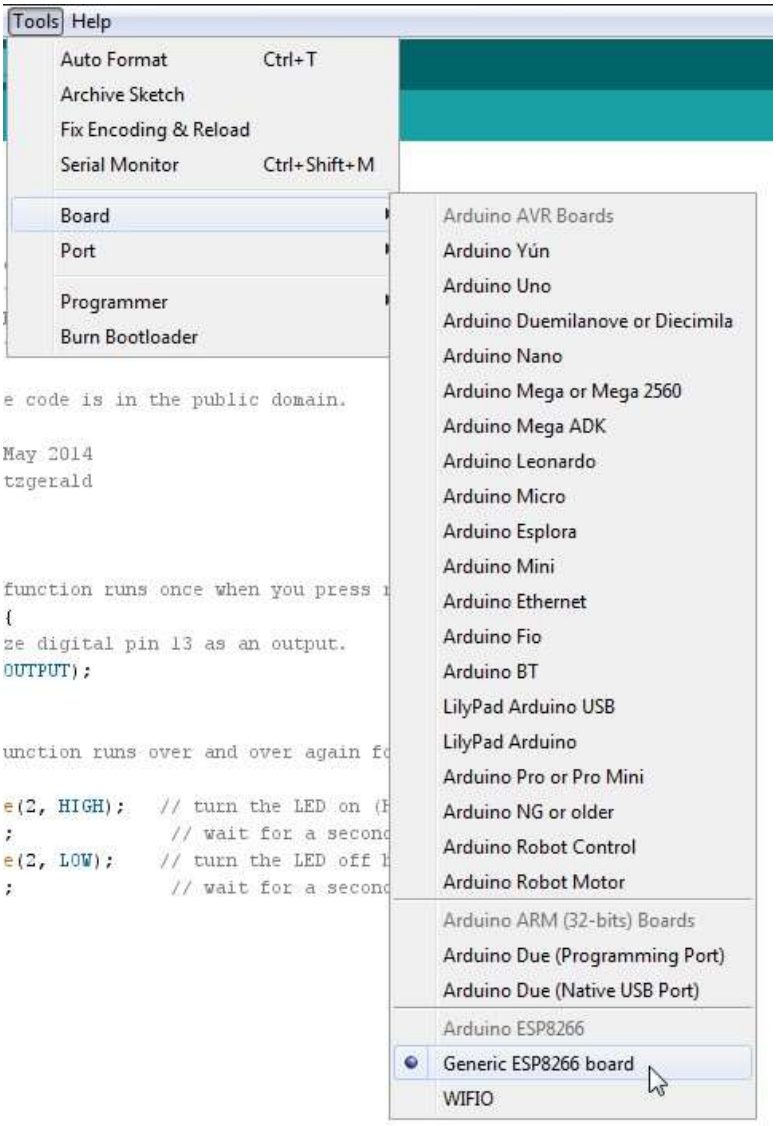
```
2.
3.      Based on ESP8266Webserver, DHTexample, and BlinkWithoutDelay (thank you)
4.
5.
6.  */
7.  #include <ESP8266WiFi.h>
8.  #include <WiFiClient.h>
9.  #include <ESP8266WebServer.h>
10. #include <DHT.h>
11. #define DHTTYPE DHT22 12. #define DHTPIN  2 13.
14. const char* ssid     = "YourRouterID";
15. const char* password = "YourRouterPassword";16.
17. ESP8266WebServer server(80);
18.
19. // Initialize DHT sensor
20. // NOTE: For working with a faster than ATmega328p 16 MHz Arduino chip, like an ESP8266,
21. // you need to increase the threshold for cycle counts considered a 1 or 0.
22. // You can do this by passing a 3rd parameter for this threshold.  It's a bit
23. // of fiddling to find the right value, but in general the faster the CPU the24. // higher the value.  The
    default for a 16mhz AVR is a value of 6.  For an
25. // Arduino Due that runs at 84mhz a value of 30 works.
26. // This is for the ESP8266 processor on ESP-01
27. DHT dht(DHTPIN, DHTTYPE, 11); // 11 works fine for ESP8266 28.
29. float humidity, temp_f;  // Values read from sensor
30. String webString="";     // String to display
31. // Generally, you should use "unsigned long" for variables that hold time
32. unsigned long previousMillis = 0;        // will store last temp was read33. const long interval = 2000;
    // interval at which to read sensor
34.
35.  void handle_root() {
36.  server.send(200, "text/plain", "Hello from the weather esp8266, read from /temp or /humidity");
37.  delay(100);38. } 39.
40.  void setup(void)
41.  {
42.  // You can open the Arduino IDE Serial Monitor window to see what the code is doing
43.  Serial.begin(115200);  // Serial connection from ESP-01 via 3.3v console cable
44.  dht.begin();           // initialize temperature sensor
45.
46.  // Connect to WiFi network
47.  WiFi.begin(ssid, password);
48.  Serial.print("\n\r \n\rWorking to connect");
49.
50.   // Wait for connection
51.   while (WiFi.status() != WL_CONNECTED) {
52.   delay(500);
53.   Serial.print(".");
54.   }
55.   Serial.println("");
56.   Serial.println("DHT Weather Reading Server");
57.   Serial.print("Connected to ");
```

```
58.    Serial.println(ssid);
59.    Serial.print("IP address: "); 60. Serial.println(WiFi.localIP());
61.
62. server.on("/", handle_root);
63.
64.    server.on("/temp", [](){  // if you add this subdirectory to your webserver call, you get text below :)
65.    gettemperature();       // read sensor
66.    webString="Temperature: "+String((int)temp_f)+" F";   // Arduino has a hard time with float to string
67.    server.send(200, "text/plain", webString);          // send to someones browser when asked
68.    }); 69.
70.    server.on("/humidity", [](){  // if you add this subdirectory to your webserver call, you get text below :)
71.    gettemperature();           // read sensor
72.    webString="Humidity: "+String((int)humidity)+"%";
73.    server.send(200, "text/plain", webString);            // send to someones browser when asked
74.    }); 75.
76.   server.begin();
77.   Serial.println("HTTP server started"); 78. } 79.
80. void loop(void)
81. {
82. server.handleClient(); 83. } 84.
85.   void gettemperature() {
86.   // Wait at least 2 seconds seconds between measurements.
87.   // if the difference between the current time and last time you read
88.   // the sensor is bigger than the interval you set, read the sensor
89.   // Works better than delay for things happening elsewhere also
90.   unsigned long currentMillis = millis();
91.
92.   if(currentMillis - previousMillis >= interval) {
93.   // save the last time you read the sensor 94.       previousMillis =
      currentMillis;
95.
96.    // Reading temperature for humidity takes about 250 milliseconds!
97.    // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
98.    humidity = dht.readHumidity();          // Read humidity (percent)
99.    temp_f = dht.readTemperature(true);     // Read temperature as Fahrenheit 100.            // Check if any
       reads failed and exit early (to try again).
101.              if (isnan(humidity) || isnan(temp_f)) {
102.              Serial.println("Failed to read from DHT sensor!"); 103.           return;
104.              }
105.              }
106.              }
```

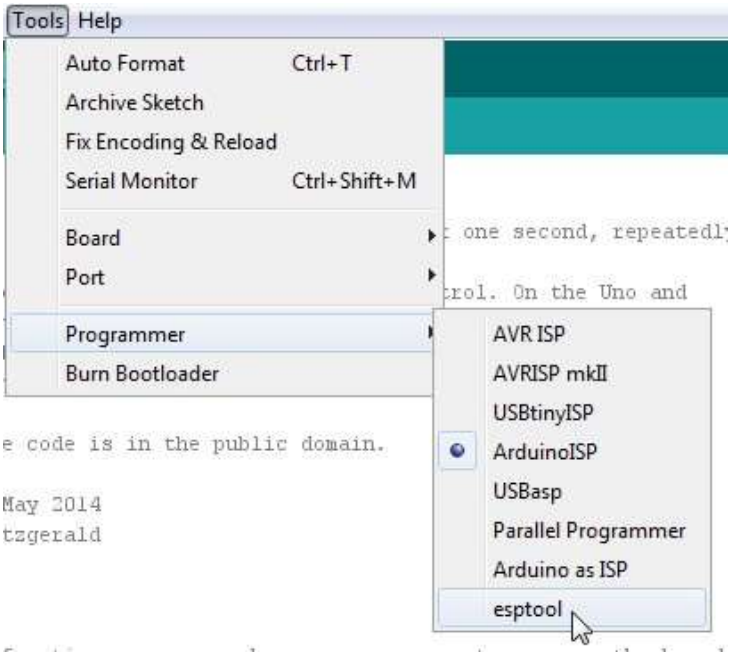
IDE Setup:

You'll then want to select the menu items for the ESP. Thanks to Hackaday, here are the steps:
When you have the ESP8266 support installed you can then select the Generic ESP8266 as a board.

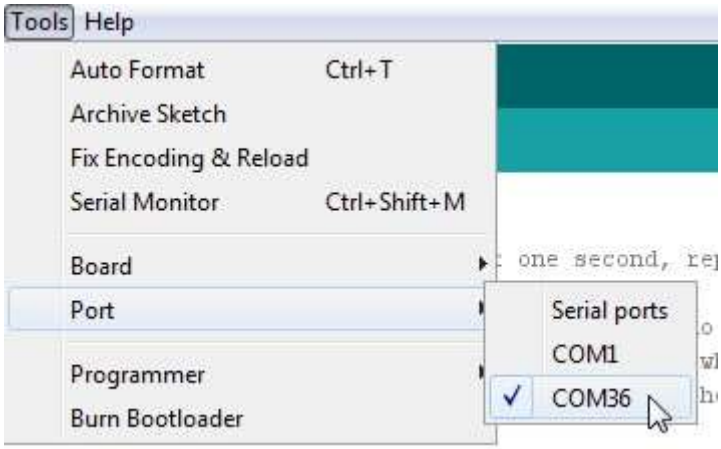




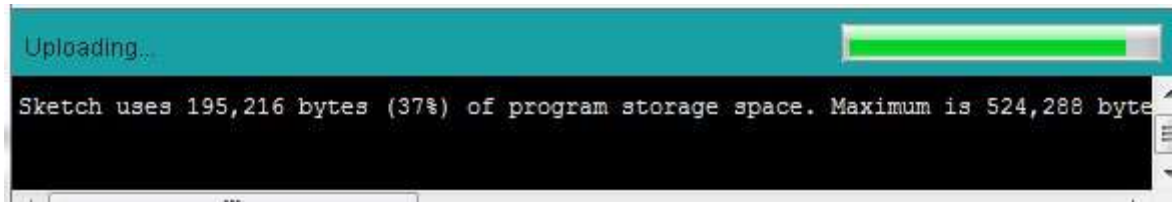
From the Tools -> Programmer menu item, select esptool



With your project powered and the USB to serial cable plugged in, you will need to select the serial port which the cable shows up as. On Windows, use the "Devices and Printers" Control Panel view to see the cable.



Now you should be able to compile the program loaded on the last page. Click the round Check icon to make sure the program compiles.
If you have errors which mention DHT then your DHT library was not recognized (go back and follow the install steps again).
If everything is set, the program will compile ok:



Uploading to the ESP8266

To upload your compiled program to the ESP8266:

1. Diconnect power to the ESP.
2. Bring GPIO0 to ground (press AND HOLD DOWN the pushbutton).
3. Power up the ESP.
4. Release the pushbutton
5. Click the round right arrow icon to upload the program

If you have errors in communications, check your connections and that you have the serial port selected appropriately. Hopefully the right color wires from the serial cable are connected to the ESP-01 module. When it works, you should see the tiny blue light on the ESP-01 module flashing and the following display on the Arduino IDE:

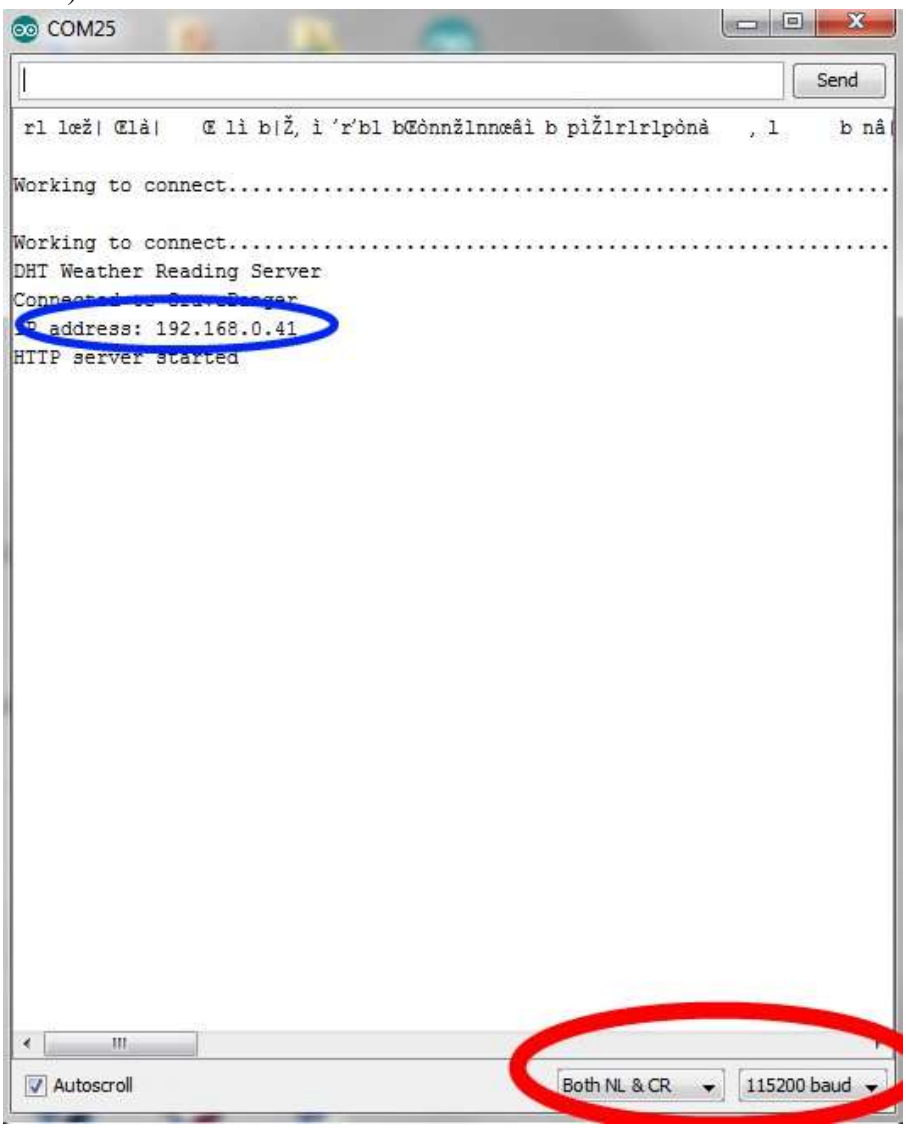
```
Sketch uses 174,082 bytes (33%) of program storage space. Maximum is 524,288 bytes.
Uploading 29072 bytes from C:\Users\hwiguna\AppData\Local\Temp\build491626908110780874
.............................

Uploading 145048 bytes from C:\Users\hwiguna\AppData\Local\Temp\build491626908110780874

.....................................................................................
```

Once the upload stops (second set of periods stops updating), your program is loaded and running. If it does not appear to run, you can unplug the power and replug in the power. ***Server Running:***

Open the Arduino IDE serial terminal by using the menu Tools -> Serial Monitor. Set the baud rate in the bottom left (red circle below) to 115,200 baud (default for current firmware, older firmware was 9,600 baud). You should see a screen like that below:

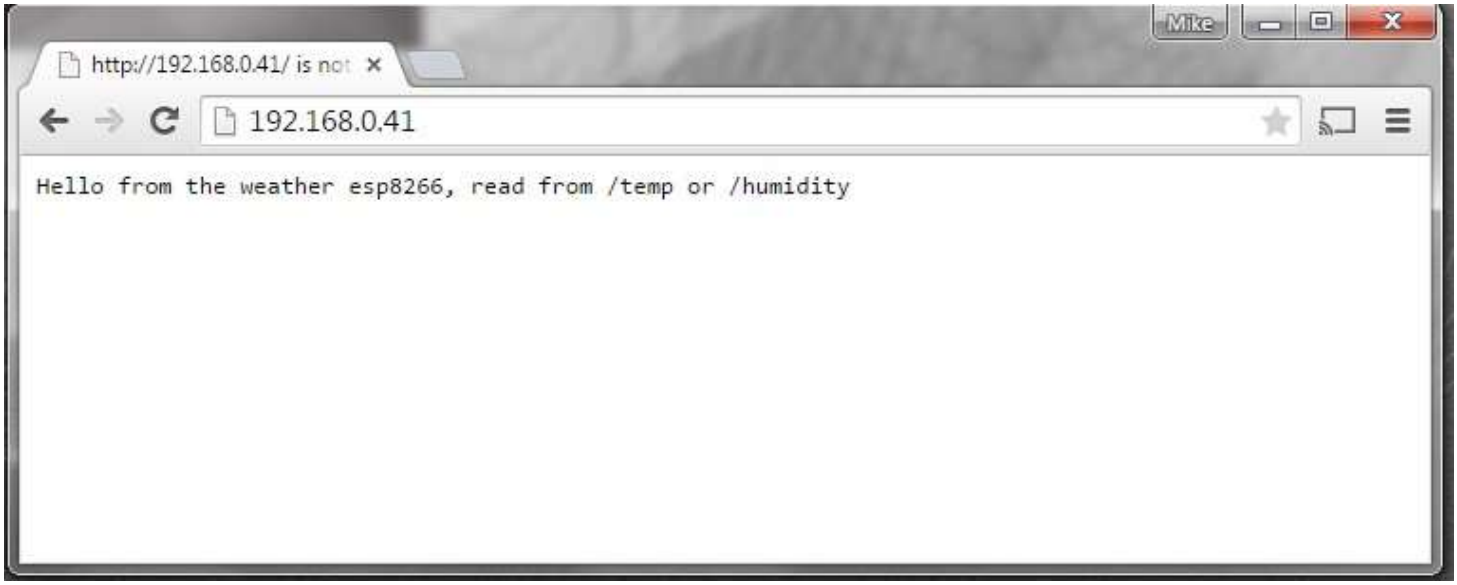


When the project powers up you'll get some random characters then some periods as the ESP8266 connects to the router where you specified the SSID and password in your Arduino sketch (you did change that, yes? If not, change the program and upload the new program).

When the ESP8266 connects to your router, It will say "DHT Weather Reading Server". Note the IP Address (in blue circle) for the next step.

Using the Webserver:

Open your favorite web browser on any device on your router's network. Type in the address of your



ESP8266 project noted in the previous step into the address bar:

Success, you are connected to the project! Now to read the temperature, type in the web address (on my network 192.168.0.41) with "/temp" after the address:
Cool! If you want a more traditional REST response (just the value and not the text telling you it is the temperature), you can edit the appropriate line of the program to remove it.
Now to ask for the humidity by appending "/humidity" to your ESP8266 project's IP address:

That's it. You have a ESP8266 module running your own custom code reading a sensor! No Arduino or

other microcontroller needed.