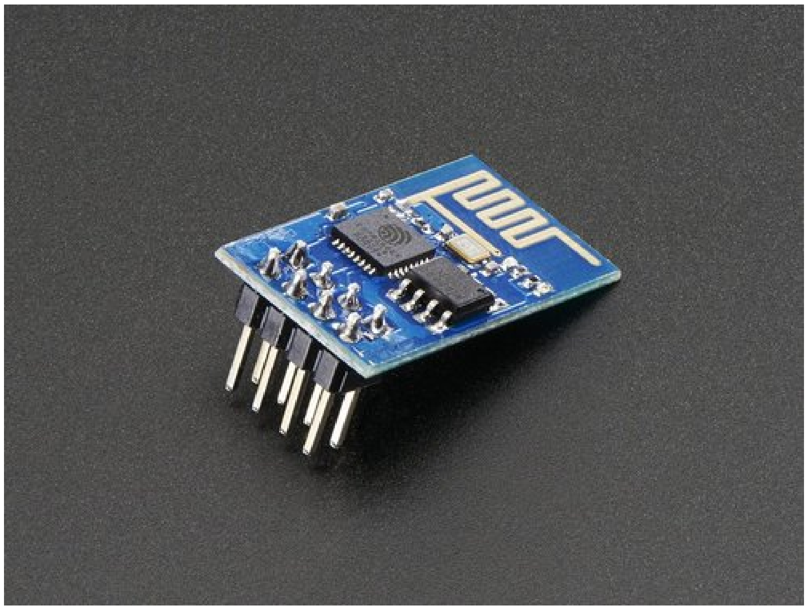
adafruit learning system

ESP8266 Temperature / Humidity Webserver

Created by Mike Barela



Last updated on 2018-08-22 03:48:03 PM UTC

Guide Contents

Guide Contents 2
Overview 3
Wiring 5
Code 9
- Required Libraries 9
- Program 9

IDE Setup 12
Uploading to the ESP8266 14
- Server Running 14

Using the Webserver 16
Going Further 18

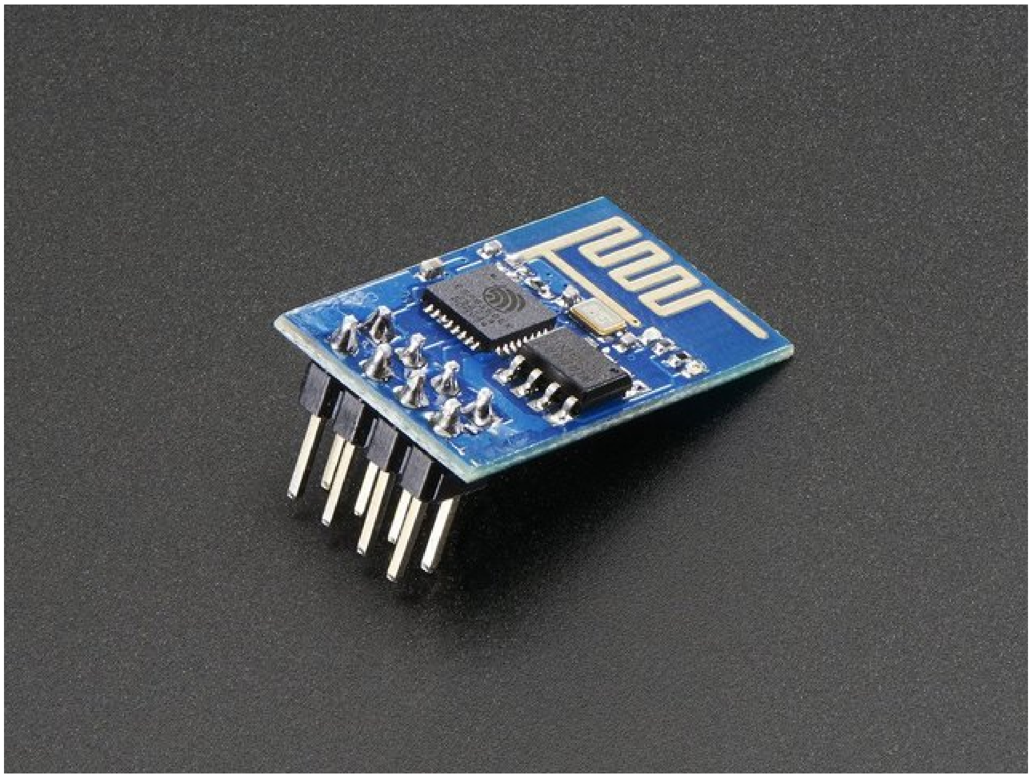
Overview

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 (https://adafru.it/f6L):

> "This is inexpensive enough to be very much in the territory of 'thousands of sensors-launched-out-of-a-cannon'-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 (http://adafru.it/2282) 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.

> This tutorial was written before we put in the HUZZAH ESP8266 breakout in the shop https://www.adafruit.com/product/2471 - we recommend going with that as it has a regulator, level shifter and is breadboard friendly!

Wiring

This tutorial was written before we put in the HUZZAH ESP8266 breakout in the shop https://www.adafruit.com/product/2471 - we recommend going with that as it has a regulator, level shifter and is breadboard friendly!

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.

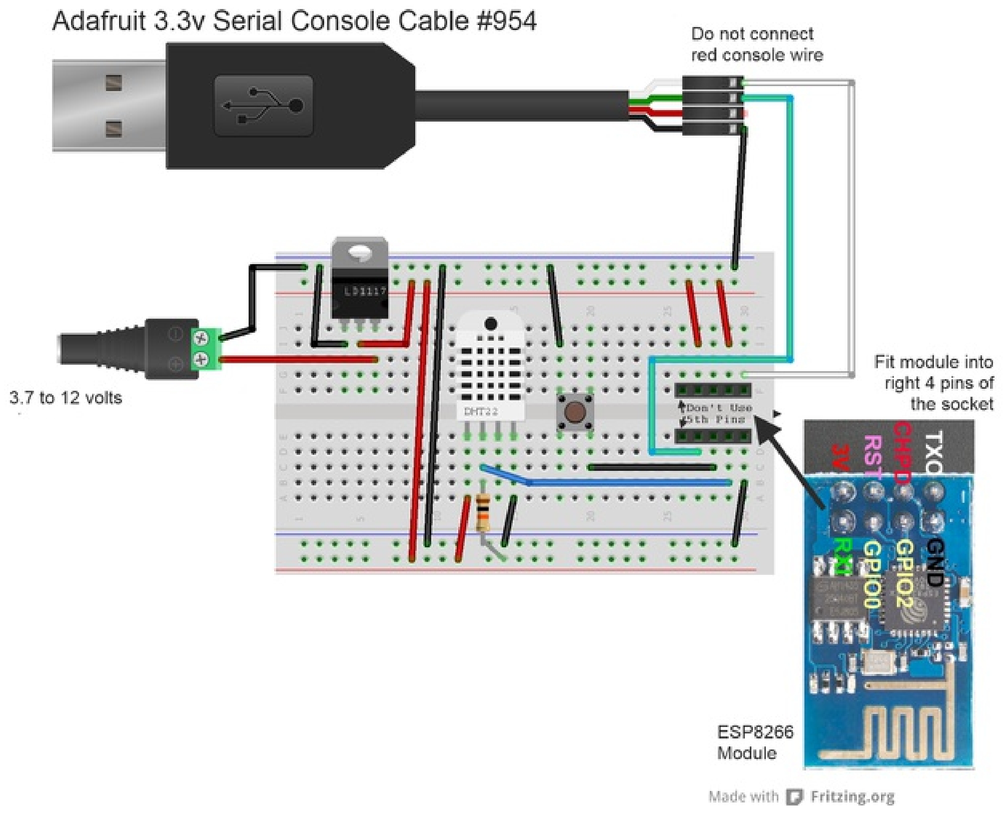
Parts List

- ESP8266 ESP-01 Wifi Module #2282 (http://adafru.it/2282)
- USB TTL Serial Console Cable #954 (http://adafru.it/954)
- 3.3 V 800mA Voltage Regulator LS1117-3.3 #2165 (https://adafru.it/f6M)
- DHT22 #385 (http://adafru.it/385) or DHT11 #386 (https://adafru.it/f6N) Sensor (comes with 10K resistor)
- IDC Breakout Helper 2x5 (pins to breadboard) #2102 (http://adafru.it/2102) or
- Female / Male 6" Jumper Wires #826 (http://adafru.it/826)
- Tactile Button Breadboard Friendly #367 (https://adafru.it/eww) (or jumper wire)

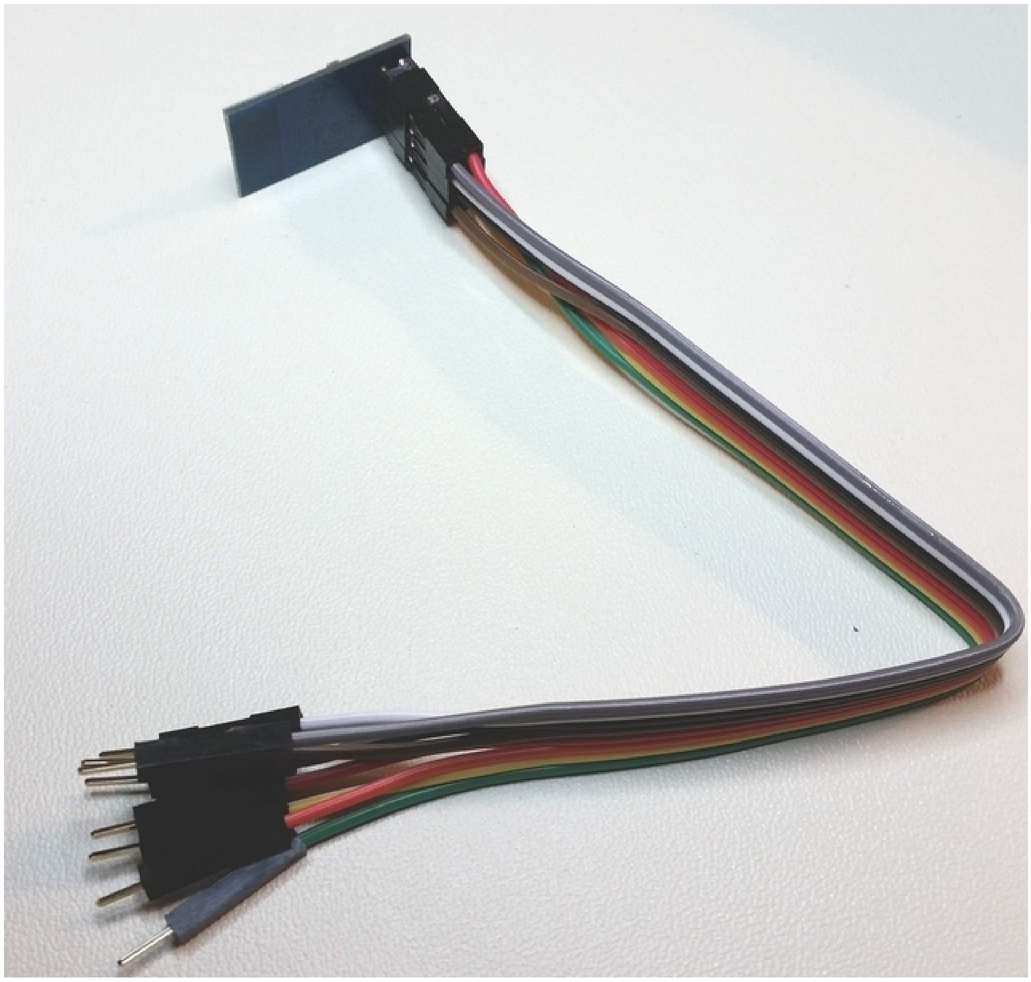
The ESP8266 is a 3.3 volt device only. It can draw over 300 milliamps at some peak operations. To give it a safe margin, the LD1117-3.3 regulator is safe, able to supply 800 milliamps when it needs to, cool at 500 milliamps. So you can connect a 3.7 volt LiPo battery (http://adafru.it/258), 5 volt "Cell Phone Recharger" battery (http://adafru.it/1959), or 9 volt battery (http://adafru.it/80). You may also use a 5 volt "wall wart" power supply (http://adafru.it/276). You might also consider a power switch (http://adafru.it/1125). 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 Adafruit FTDI Friend is safe as the default transmit and receive signal level is 3.3 volts if left on the default on the back.

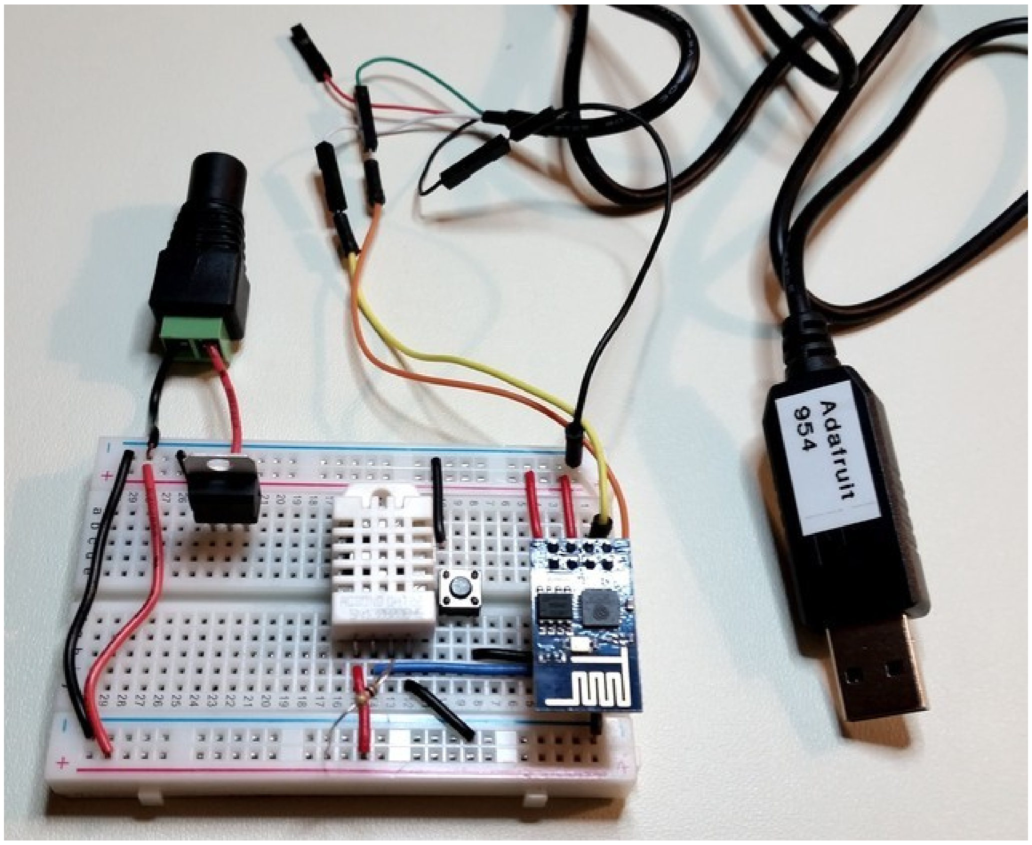
The Adafruit Console Cable #954 (https://adafru.it/dDd) runs at 3.3 volt signal levels so it's safe (safe for the sensitive Raspberry Pi also). The only thing is do not connect the red 5 volt wire, it is not needed.


Adafruit 3.3v Serial Console Cable #954
Do not connect red console wire
3.7 to 12 volts
LD1117
DHT22
Don't Use 5th Pins
Fit module into right 4 pins of the socket
3V
RST
CHPD
TXO
RXI
GPIO0
GPIO2
GND
ESP8266 Module
Made with Fritzing.org

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).


Adafruit
954

Code

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 Adafruit's guide on adding support for boards like the ESP8266 for the Arduino IDE (https://adafru.it/BY2). 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. Going forward, Adafruit recommends the 1.6.4+ add-in method of adding in support.

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 (if you have the new 1.6.4+ IDE and the libraries are not visible, see this repository (https://adafru.it/BY3) for the libraries with their corresponding examples).

The other library needed is the Adafruit DHT library. If you need help knowing about Arduino software libraries, see Adafruit's great tutorial on libraries! (https://adafru.it/dNR)

Begin by downloading the DHT library from the Adafruit github repository (https://adafru.it/aJX). To download, click the DOWNLOADS button in the top right corner. Rename the uncompressed folder DHT and make sure that it contains the dht.cpp file and others. Then drag the DHT folder into the *arduinosketchfolder*/libraries/ folder. You may have to create that libraries sub-folder if it doesnt 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

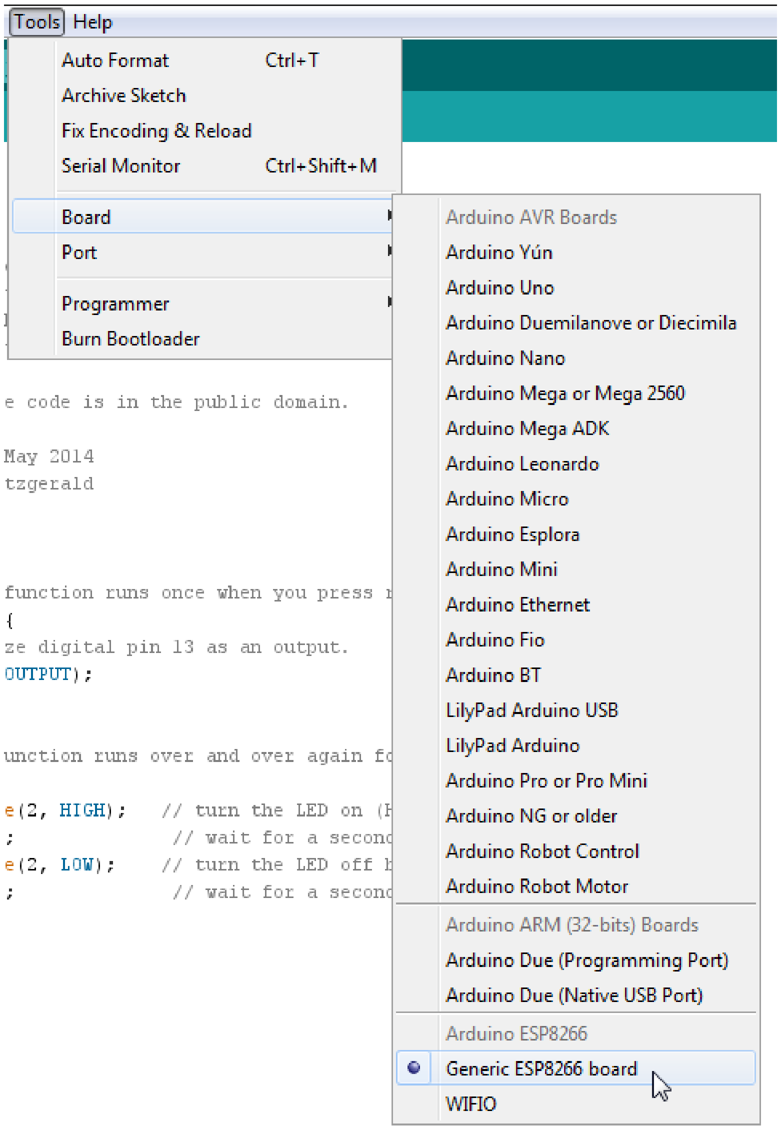
```
/* DHTServer - ESP8266 Webserver with a DHT sensor as an input

   Based on ESP8266Webserver, DHTexample, and BlinkWithoutDelay (thank you)

   Version 1.0  5/3/2014  Version 1.0   Mike Barela for Adafruit Industries
*/
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTTYPE DHT22
#define DHTPIN  2

const char* ssid     = "YourRouterID";
const char* password = "YourRouterPassword";

ESP8266WebServer server(80);

// Initialize DHT sensor
// NOTE: For working with a faster than ATmega328p 16 MHz Arduino chip, like an ESP8266,
// you need to increase the threshold for cycle counts considered a 1 or 0.
// You can do this by passing a 3rd parameter for this threshold.  It's a bit
// of fiddling to find the right value, but in general the faster the CPU the
// higher the value.  The default for a 16mhz AVR is a value of 6.  For an
// Arduino Due that runs at 84mhz a value of 30 works.
// This is for the ESP8266 processor on ESP-01
```

```
DHT dht(DHTPIN, DHTTYPE, 11); // 11 works fine for ESP8266

float humidity, temp_f;  // Values read from sensor
String webString="";     // String to display
// Generally, you should use "unsigned long" for variables that hold time
unsigned long previousMillis = 0;        // will store last temp was read
const long interval = 2000;              // interval at which to read sensor

void handle_root() {
  server.send(200, "text/plain", "Hello from the weather esp8266, read from /temp or /humidity");
  delay(100);
}

void setup(void)
{
  // You can open the Arduino IDE Serial Monitor window to see what the code is doing
  Serial.begin(115200);  // Serial connection from ESP-01 via 3.3v console cable
  dht.begin();           // initialize temperature sensor

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.print("\n\r \n\rWorking to connect");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("DHT Weather Reading Server");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", handle_root);

  server.on("/temp", [](){  // if you add this subdirectory to your webserver call, you get text below :)
    gettemperature();       // read sensor
    webString="Temperature: "+String((int)temp_f)+" F";   // Arduino has a hard time with float to string
    server.send(200, "text/plain", webString);            // send to someones browser when asked
  });

  server.on("/humidity", [](){  // if you add this subdirectory to your webserver call, you get text belo
    gettemperature();           // read sensor
    webString="Humidity: "+String((int)humidity)+"%";
    server.send(200, "text/plain", webString);               // send to someones browser when asked
  });

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void)
{
  server.handleClient();
}

void gettemperature() {
  // Wait at least 2 seconds seconds between measurements
  // wait at least 2 seconds seconds between measurements.
  // if the difference between the current time and last time you read
  // the sensor is bigger than the interval you set, read the sensor
  // Works better than delay for things happening elsewhere also
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= interval) {
    // save the last time you read the sensor
    previousMillis = currentMillis;

    // Reading temperature for humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
    humidity = dht.readHumidity();          // Read humidity (percent)
    temp_f = dht.readTemperature(true);     // Read temperature as Fahrenheit
    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temp_f)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }
  }
}
```

IDE Setup

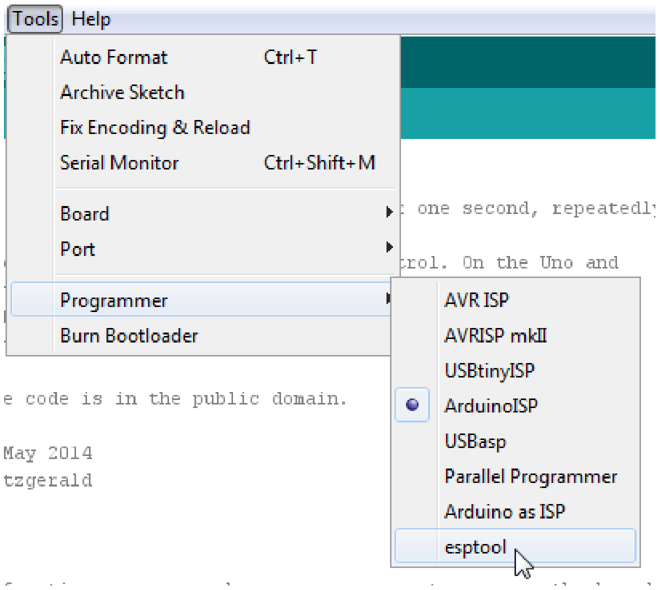
You'll then want to select the menu items for the ESP. Thanks to Hackaday (https://adafru.it/f6O), here are the steps:

When you have the ESP8266 support installed you can then select the Generic ESP8266 as a board.

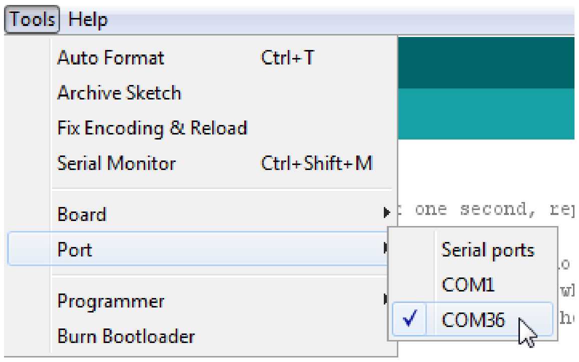


```
e code is in the public domain.

May 2014
tzgerald



function runs once when you press r
{
ze digital pin 13 as an output.
OUTPUT);


unction runs over and over again fo

e(2, HIGH);   // turn the LED on (H
;              // wait for a second
e(2, LOW);    // turn the LED off b
;              // wait for a second
```

From the Tools -> Programmer menu item, select esptool



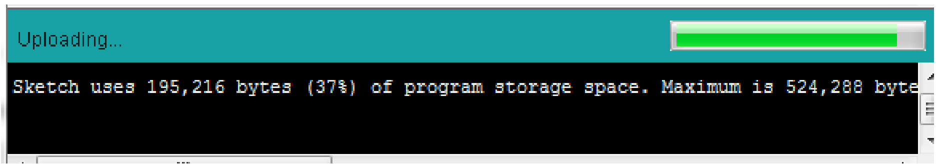
With your project powered and the USB to serial cable plugged in, you will need to select the serial port which the cable shows up as. On Windows, use the "Devices and Printers" Control Panel view to see the cable.



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 Adafruit DHT library was not recognized (go back and follow the install steps again).

If everything is set, the program will compile ok:

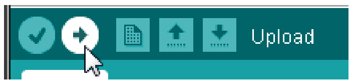


```
Sketch uses 195,216 bytes (37%) of program storage space. Maximum is 524,288 byte
```

Uploading to the ESP8266

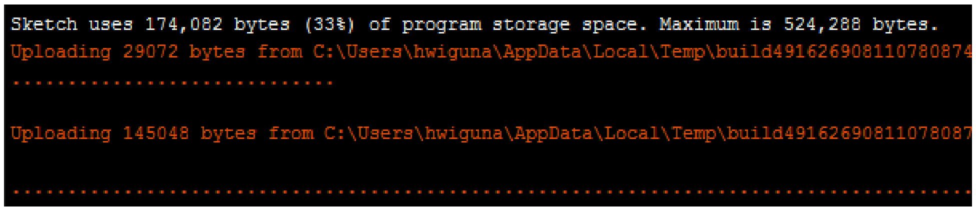
To upload your compiled program to the ESP8266:

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:

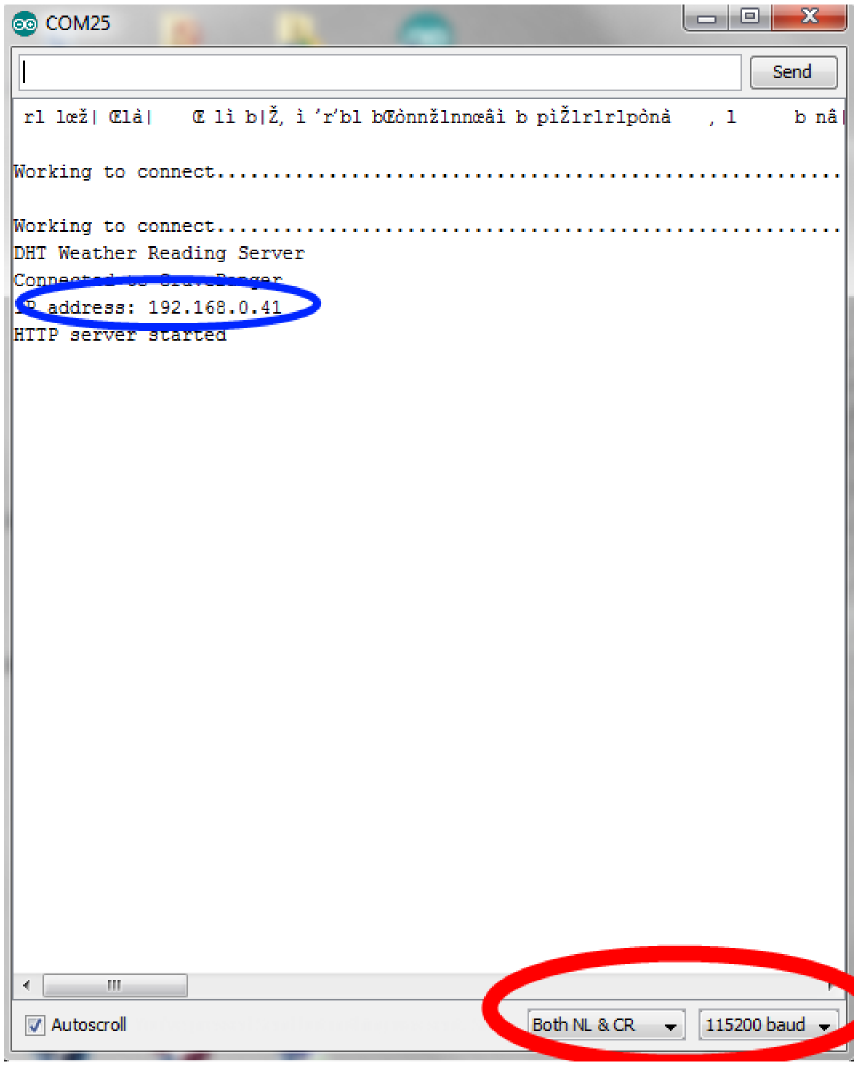


```
Sketch uses 174,082 bytes (33%) of program storage space. Maximum is 524,288 bytes.
Uploading 29072 bytes from C:\Users\hwiguna\AppData\Local\Temp\build491626908110780874
.............................

Uploading 145048 bytes from C:\Users\hwiguna\AppData\Local\Temp\build49162690811078087

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

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:



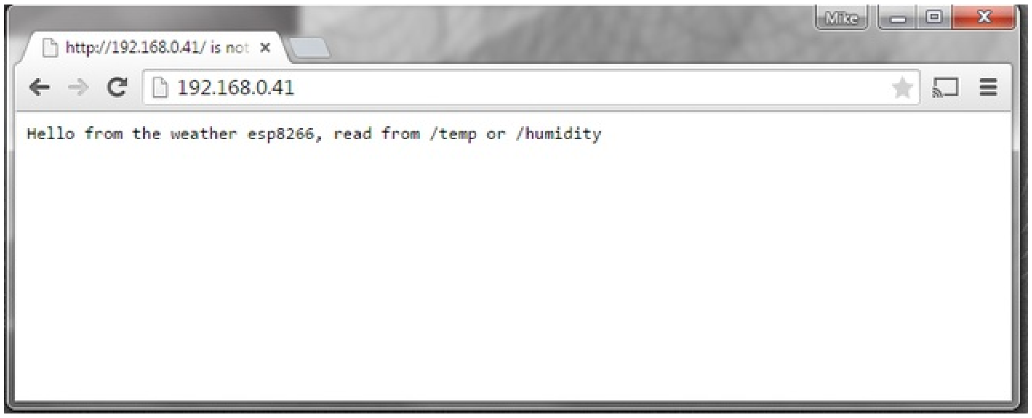
```
 rl lœž| Œlà|    Œ lì b|Ž, ì ’r’bl bŒònnžlnnœâì b piŽlrlrlpònà   , l     b nâ|

Working to connect.........................................................

Working to connect.........................................................
DHT Weather Reading Server
Connected to [illegible]
IP address: 192.168.0.41
HTTP server started
```

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.

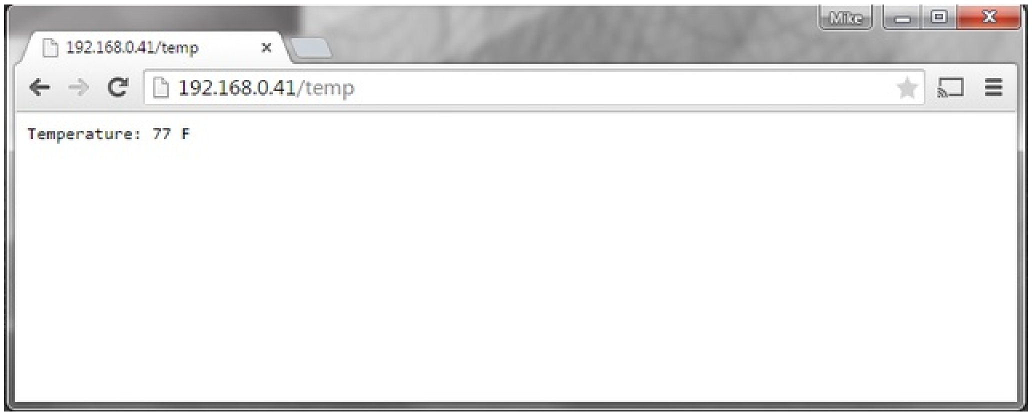
Using the Webserver

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:



```
Hello from the weather esp8266, read from /temp or /humidity
```

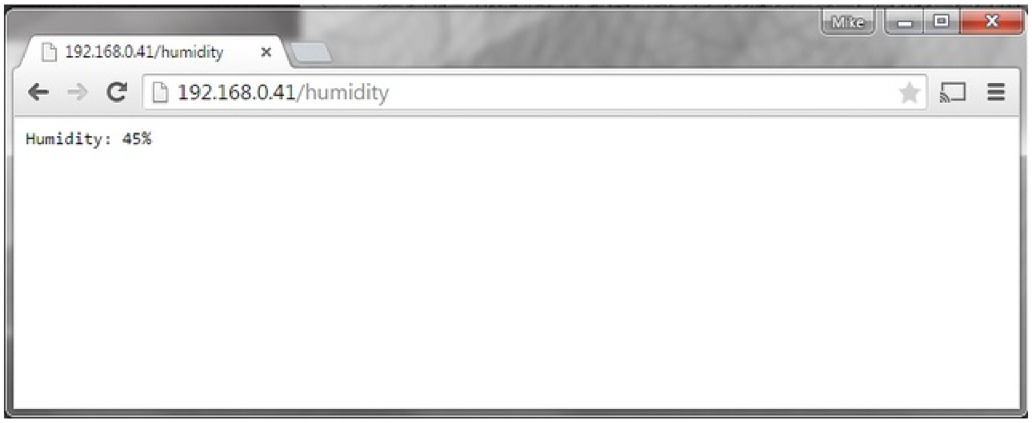
Success, you are connected to the project! Now to read the temperature, type in the web address (on my network 192.168.0.41) with "/temp" after the address:



```
Temperature: 77 F
```

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:



```
Humidity: 45%
```

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

Going Further

I chose the DHT11 / DHT22 sensor because it reads a couple values and uses digital logic. The small ESP-01 module only has two digital pins free (GPIO0 and GPIO2). No analog pin is available so using a TMP36 sensor is out.

So what other digital sensors would also work:

- Tilt sensor (http://adafru.it/173)
- Ultrasonic Sensor (https://adafru.it/f6P)
- Passive Infrared Sensor (http://adafru.it/189)
- Magnetic Contact Sensor (http://adafru.it/375)
- several others

Changing up the Internet interface, using the Twitter APIs to tweet (https://adafru.it/BY4) when a sensor is activated is possible (I have not tried if the ESP will do an SSL connection which can be hard). Sending readings to Adafruit.io (https://adafru.it/iRf) would be great also. It is all up to you.