sparkfun
START SOMETHING

# ESP8266 Thing Development Board Hookup Guide

CONTRIBUTORS: *JIMBLOM*

♡ FAVORITE 14   [twitter] [facebook] [pinterest] ⌑ SHARE

Introduction

The ESP8266 is a cost-effective, and very capable WiFi-enabled microcontroller. Like any microcontroller, it can be programmed to blink LEDs, trigger relays, monitor sensors, or automate coffee makers, and with an integrated WiFi controller, the ESP8266 is a one-stop shop for almost any Internet-connected project. To top it all off, the ESP8266 is incredibly easy-to-use: firmware can be developed in Arduino and uploaded over a simple, serial interface.

To take advantage of all of those benefits, we've created the ESP8266 Thing Development Board -- an ESP8266 development board, with an integrated FTDI USB-to-Serial chip.
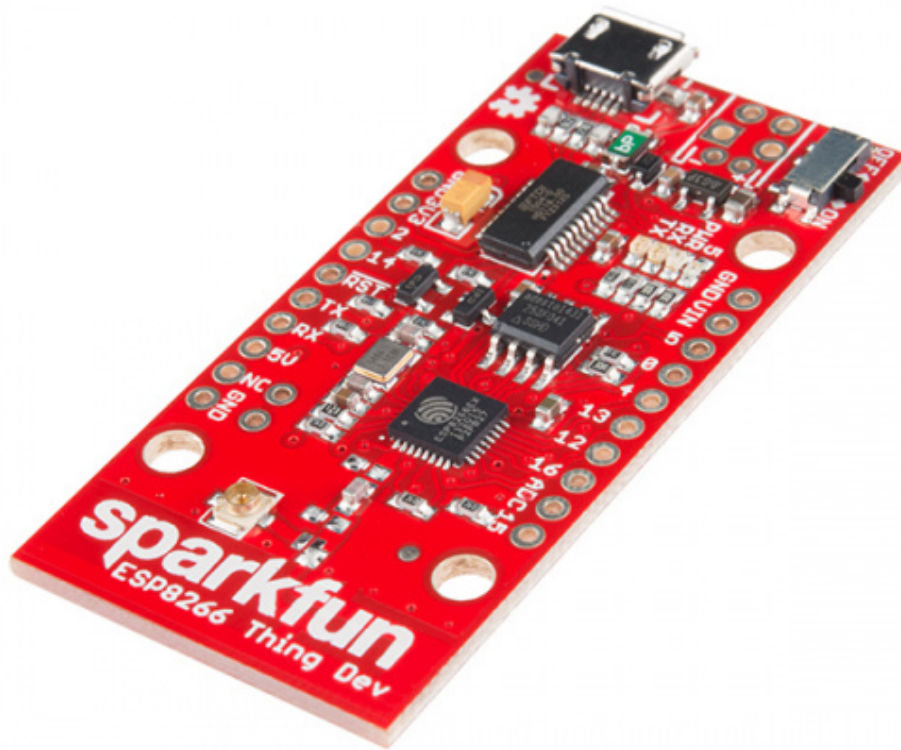
SparkFun ESP8266 Thing - Dev Board
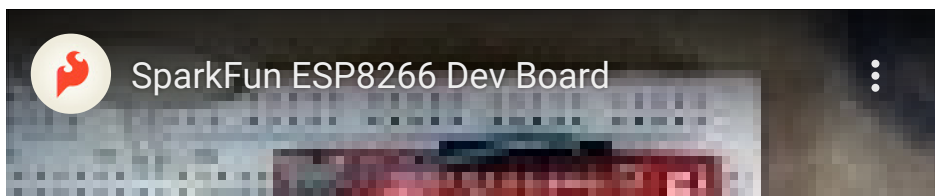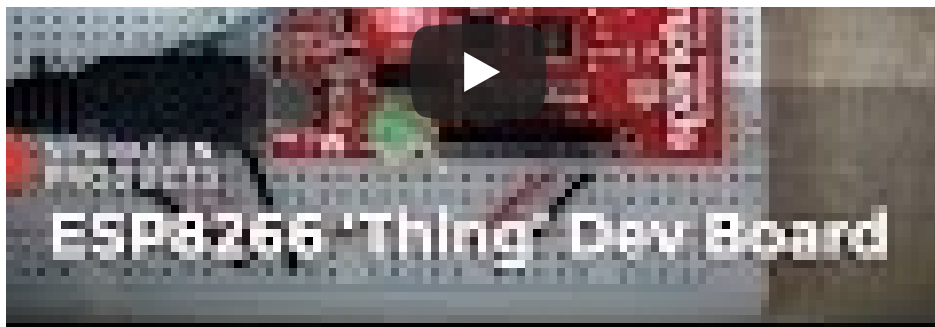
⊙ WRL-13711

$16.95

★★★★☆ 70

SparkFun ESP8266 Thing Dev Starter Kit

KIT-15259

$24.95

SparkFun ESP8266 Dev Board

Having trouble viewing the videos? Try resizing the video by clicking on the shortcut to view the product showcase in full screen.

The ESP8266 Thing Development Board breaks out all of the module's pins, and the USB-to-serial converter means you don't need any peripheral components to program the chip. Just plug in a USB cable, download the Arduino board definitions, and start IoT-ing.

Covered in this Tutorial

This tutorial will help you get your ESP8266 Thing Development Board from zero to Internet-controlled blinking. It's split into the following sections:

- Hardware Overview -- A quick rundown of the Thing Development Board's components and pinout.
- Hardware Setup -- Tips and recommendations on what to solder to the Thing Development Board's I/O pins.
- Setting Up Arduino -- What truly makes the ESP8266 so powerful is its potential for Arduino-compatibility.
- Example Sketch: Web Server -- Run an HTTP server on the Thing. Use it to serve web pages, print status messages, and control LEDs!
- Example Sketch: Blink with Blynk -- One of our favorite new toys is the Blynk phone app, which allows you to toggle LEDs and monitor inputs with a simple multi-platform phone app.
- Using the ESP8266 in Arduino -- A few gotchya's to look out for when programming the ESP8266 in Arduino.

Required Materials

If you are ordering the ESP8266 Thing Development individually, you will need a few components to get started. Beyond the ESP8266 Thing Development Board itself, all you should need to get started is a micro-B USB Cable, which will deliver power the board and set up our USB programming interface.

Depending on how you want to use the board, you may also want to add male headers, female headers, or hedge your bets with 10-pin stackable headers.

Break Away Headers - Straight

○ PRT-00116

$1.50

★★★★⯪ 20

Female Headers
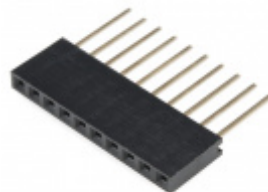
◉ PRT-00115

$1.50

★★★★☆ 7

USB micro-B Cable - 6 Foot

◉ CAB-10215

$4.95

★★★★⯪ 13

Arduino Stackable Header - 10 Pin

○ PRT-11376

$0.50

★★★★★ 2

Heads up! If you are looking for the development board with headers soldered, check out WRL-13804 with male headers soldered to the board as another option.
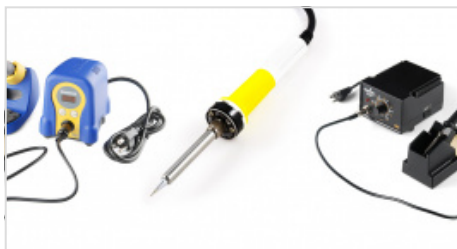
SparkFun ESP8266 Thing - Dev Board (with Headers)

⊖ WRL-13804

$17.95

★★★★⯪ 8

Suggested Reading

Before continuing on with this tutorial, you may want to familiarize yourself with some of these topics if they're unfamiliar to you:

How to Solder: Through-Hole Soldering

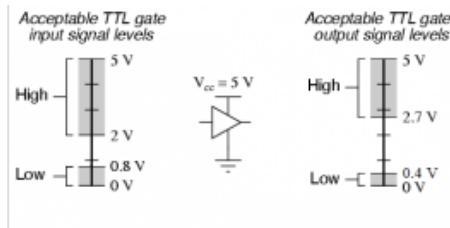This tutorial covers everything you need to know about through-hole soldering.

Serial Communication

Asynchronous serial communication concepts: packets, signal levels, baud rates, UARTs and more!

How to Power a Project

A tutorial to help figure out the power requirements of your project.

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

Hardware Overview

The ESP8266 Thing Development Board is a relatively simple board. The pins are broken out to two parallel, breadboard-compatible rows. The USB connector sits next to an optional power supply input, and an ON/OFF switch -- controlling power to the ESP8266 -- sits next to that. And LEDs towards the inside of the board indicate power, charge, and status of the IC.
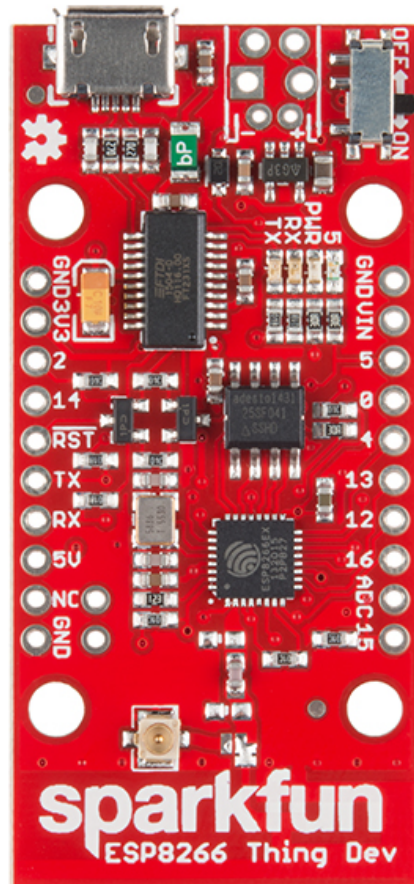
This section provides a quick overview of the Thing Dev Board's main components.

Serial and I^{2}C Header

The header on the left provides an interface for serial, I^{2}C, and power:

Pin Label	ESP8266 I/O Function(s)	Notes

GND		Ground (0V).
3V3		3.3V
2	GPIO2, SDA	Can either be used as ESP8266 GPIO2 or I^{2}C serial data (SDA).
14	GPIO14, SCL, SCLK	Can either be used as ESP8266 GPIO14 or I^{2}C serial clock (SCL). Also used as the SPI clock (SCLK).
RST		The ESP8266's **active-low** reset input. The board includes a 10kΩ pull-up resistor on this pin.
TX	GPIO7, TX1	ESP8266 UART1 data output.
RX	GPIO8, RX1	ESP8266 UART1 data input.
5V		USB supply output. If USB is connected, this pin will supply about 4.8V.
NC		Not connected to anything.
GND		Ground (0V).

The top portion of this header breaks out the ESP8266's I^{2}C interface, a popular interface for a variety of sensors including motion sensor, light sensor, digital-to-analog converter, or OLED display, I^{2}C is often the protocol of choice.

If you need the extra I/O, instead of I^{2}C, the SDA and SCL pins can be used as GPIO 2 and 14 respectively. The SCL pin also serves as the clock (SCLK) for the ESP8266's SPI interface.

The lower part of the header breaks out one of the ESP8266's **serial UARTs**. This serial port is **used to program the thing**, so be careful using it for other tasks.

General I/O Header

The rest of the power, control, and I/O pins are broken out on the other side of the board. They are:

Pin Label	ESP8266 I/O Function	Notes
GND		Ground (0V).
VIN		USB connected: ~5V output Can alternatively be used as a voltage supply input to the 3.3V regulator.
5	GPIO5	This pin is also tied to the on-board LED.
0	GPIO0	
4	GPIO4	
13	GPIO13, MOSI	Hardware SPI MOSI
12	GPIO12, MISO	Hardware SPI MISO
16	GPIO16, XPD	Can be connected to reset to wake the ESP8266 from deep sleep mode.
ADC	A0	A 10-bit ADC with a **maximum voltage of 1V**.

15	GPIO15	

External Power Supply

If your project requires a power source other than USB, the Thing Dev Board includes footprints for a 2-pin JST, 2-pin 3.5mm screw terminal, or a simple 0.1"-pitch 2-pin header.
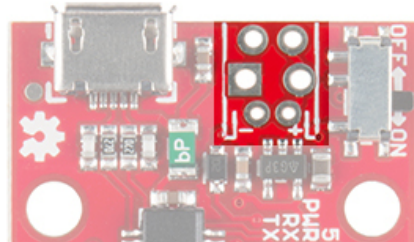
You can supply anywhere between **3.3V and 6V** into these inputs to power the board.

> **Note:** Unlike the original ESP8266 Thing, the ESP8266 Thing Development Board **does not have a built-in LiPo charger**. A LiPo battery can be connected into a populated JST connector, but you'll need to add some extra circuitry to charge it.

Power-Saving Jumpers

A pair of jumpers on the back of the board can be used to help reduce the Thing's power consumption.
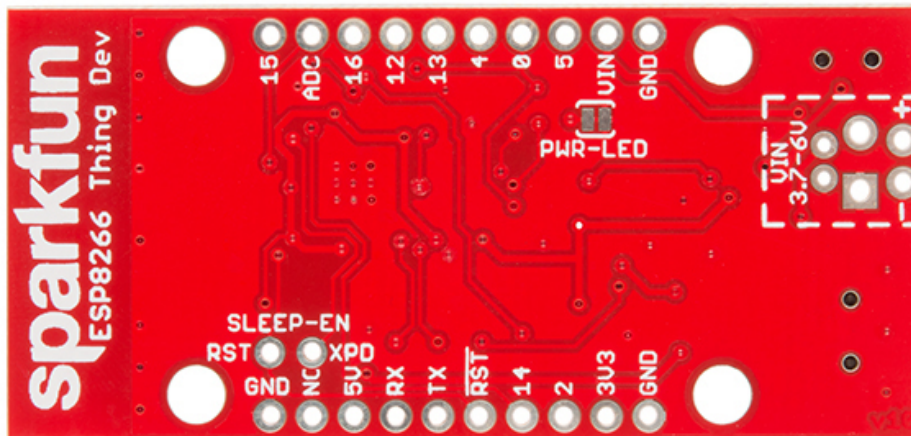
Jumper Label	Default Setting	Notes
SLEEP-EN	Open	Connects GPIO16 (XPD) to the ESP8266's RST pin.
PWR-LED	Closed	Completes the power LED indicator circuit.

The *SLEEP-EN* jumper connects GPIO16 (which has the XPD functionality) to the ESP8266's reset input. This connection is required if you want the ESP8266 to automatically wake itself from deep sleep.
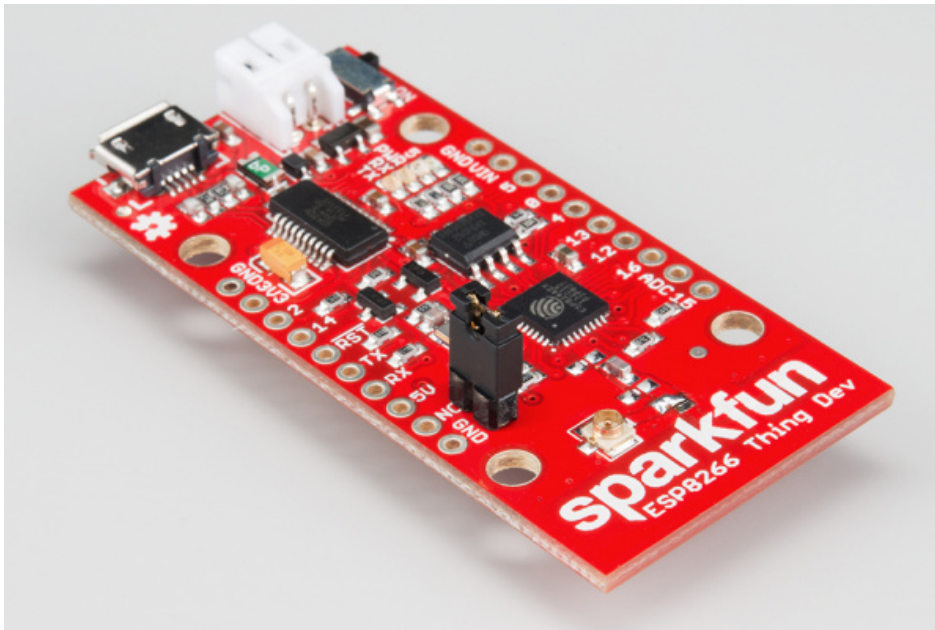
The SLEEP_EN jumper set: enables using and waking up from deep sleep mode, but disables programming.

To use the jumper solder a 2-pin male header and slide a 2-pin jumper on and off. This is a through-hole jumper only, because **you'll need to remove the jumper to program** the ESP2866.

The *PWR-LED* jumper allows you to disable the power LED. The LED will normally pull about 7mA, which is a ton compared to the ESP8266's 10's of μA consumption in sleep mode.

To disable the power LED, slice the interconnecting trace with your handy hobby knife.

Selecting the Antenna

The Thing Dev Board's default WiFi antenna is a PCB trace antenna based on this TI app note. It's cost-effective and actually works really well!

If you need to connect a more sensitive antenna, or need to route outside an enclosure, a U.FL connector is also available on the board, but isn't connected by default to the ESP8266's antenna pin. To connect this antenna to the chip, you'll need to swap the jumper by removing the solder blob and pushing it over to the other side.

*Antenna-select jumper set to U.FL, and an external attached.*

Then attach a U.FL WiFi antenna of your choice. Our adhesive antenna or a U.FL-to-RP-SMA adapter/2.4GHz Duck Antenna combo are good options.

Hardware Setup

To use any of the Thing Dev Board's GPIO pins, you'll need to solder *something* to the board. If you've never soldered before, this is a great time to start! These solder points are easy, through-hole pins, check out our How to Solder - Through-hole Soldering for help getting started.

What, exactly, you solder to the board depends on how you'll use it in your project. There are a variety of header options, including stackable headers, straight male headers, or straight female headers.

**Stackable Headers** make it convenient to both breadboard the Thing Dev Board and jumper wire out of it.
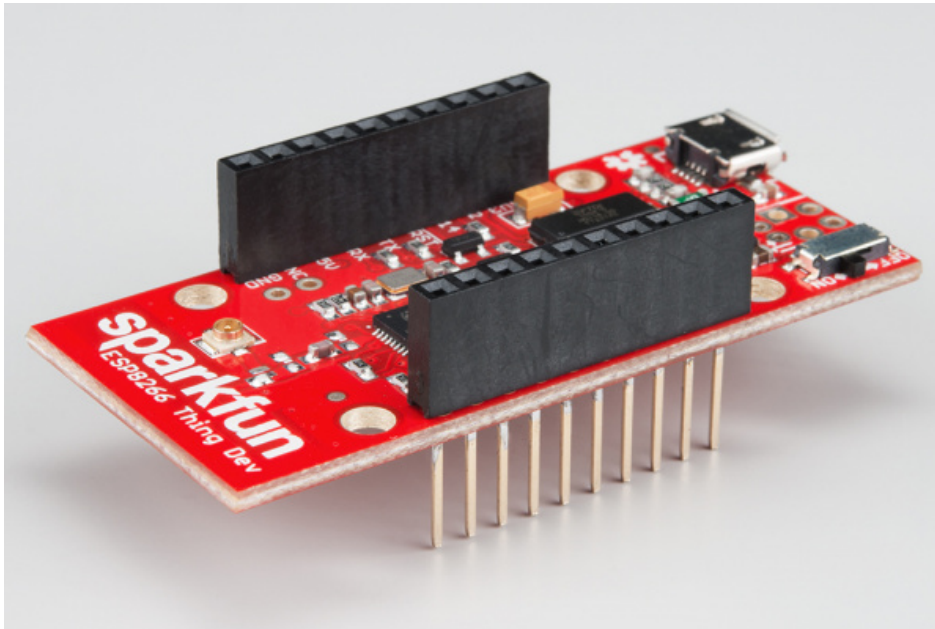
And, of course, wire can be soldered to any of the pins that have a long way to connect to something.

Powering the Thing Development Board

The easiest way to power the Thing Dev Board is by connecting a USB cable to the micro-B USB jack. The other end of the USB cable can be connected to your computer or a USB wall wart. After powering the board, make sure the ON/OFF switch is slid into the "ON" position, and you should see the "PWR" LED illuminate.
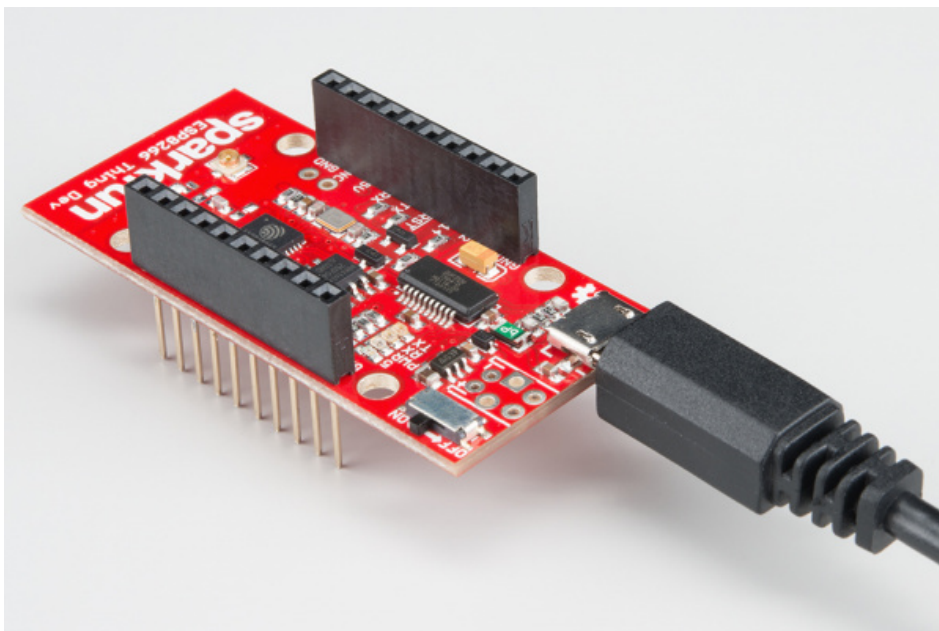
Alternatively, you can solder a variety of connectors into the VIN position to run the board on some other power supply. For example, you could solder a 2-pin JST connector and mate the board with a 2xAA Battery Holder to power your project.
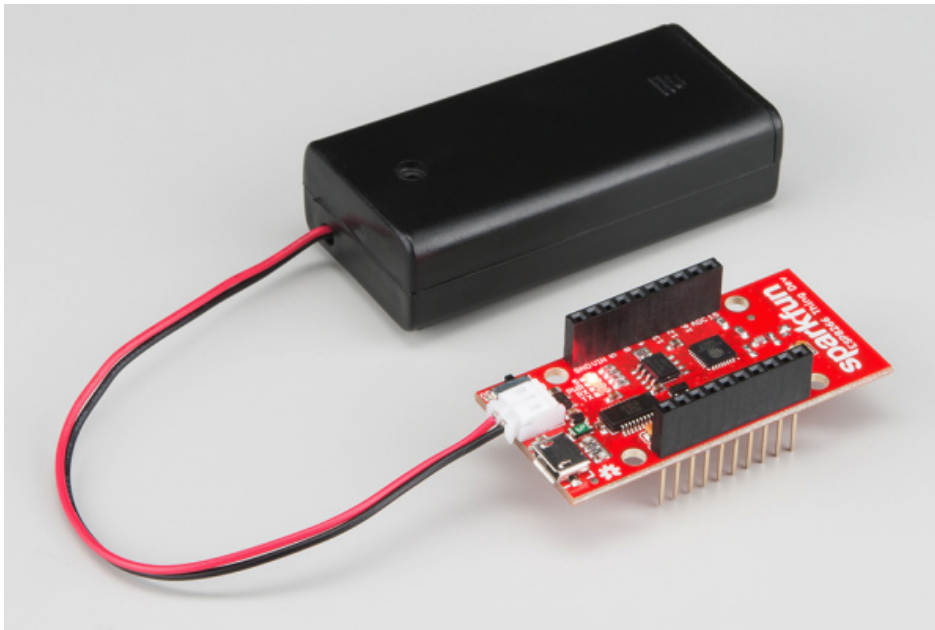
(A pair of AA batteries may slightly underpower the 3.3V regulator, but the board should still provide more than the 1.8V required for the ESP8266.)

Driver Install: If you've never used an FTDI device, you may need to install drivers on your computer before you can program the ESP8266. For help with that, follow along with our How to Install FTDI Drivers tutorial.

We have installation directions documented for all major operating systems:

WINDOWS   **MAC**   **LINUX**

Setting Up Arduino

There are a variety of development environments that can be equipped to program the ESP8266. You can go with a simple Notepad/gcc setup, fine-

tune an Eclipse environment, or use a virtual machine provided by Espressif. If you're just getting started, though, we recommend the comfy confines of the **Arduino IDE**.

> The amazing ESP8266 community has cooperatively created an ESP8266 addon for the IDE, which is what we'll focus on using throughout this tutorial. This ESP8266 addon for Arduino is based on the amazing work by Ivan Grokhotkov and the rest of the ESP8266 community. Check out the ESP8266 Arduino GitHub repository for more information.

## Installing the Addon With the Arduino Boards Manager

With the release of Arduino 1.6.4, adding third party boards to the Arduino IDE is easily achieved through the board manager. If you're running an older version of Arduino (1.6.3 or earlier), we recommend upgrading now. As always, you can download the latest version of Arduino from arduino.cc.

To begin, you'll need to point the Arduino IDE board manager to a custom URL. Open up Arduino, then go to the Preferences (**File** > **Preferences**). Then, towards the bottom of the window, paste this URL into the "Additional Board Manager URLs" text box:
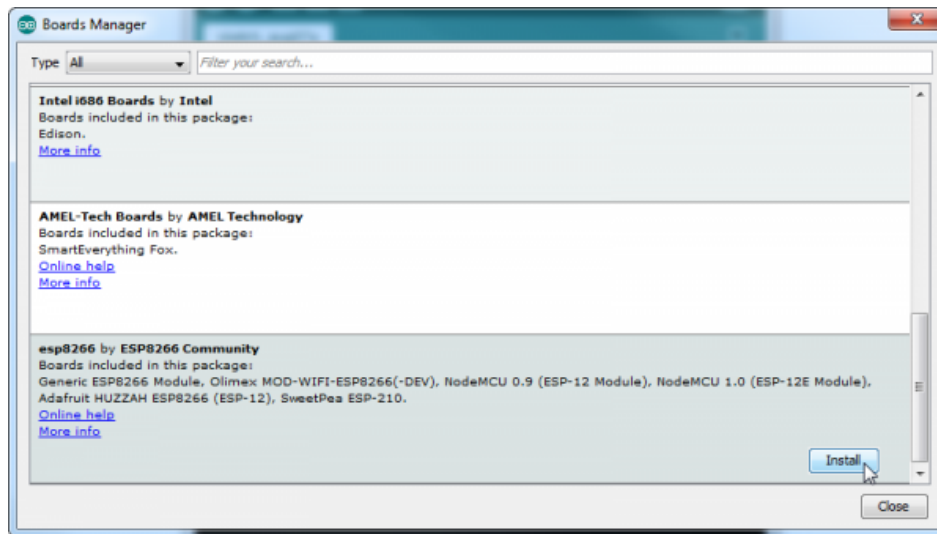
```
http://arduino.esp8266.com/stable/package_esp
8266com_index.json
```
COPY CODE

You can add multiple URLs by clicking the window icon, and pasting in one URL per line.
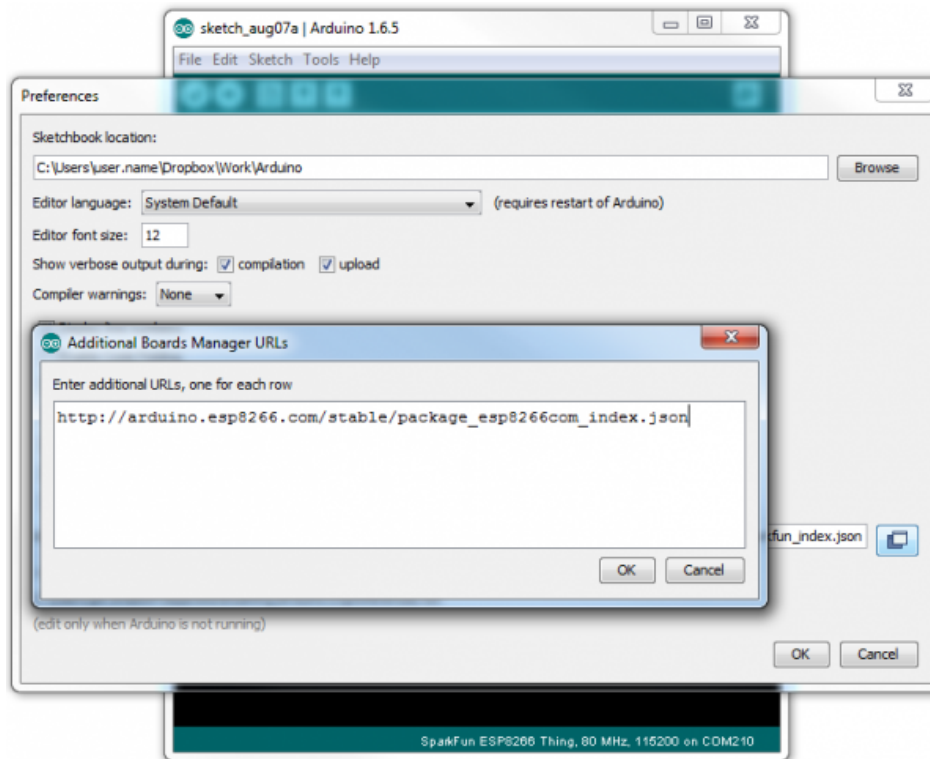
Hit OK. Then navigate to the Board Manager by going to **Tools** > **Boards** > **Boards Manager**. Look for **esp8266**. Click on that entry, then select **Install**.

The board definitions and tools for the ESP8266 include a whole new set of gcc, g++, and other reasonably large, compiled binaries, so it may take a few minutes to download and install (the archived file is ~110MB). Once the installation has completed, an Arduino-blue "INSTALLED" will appear next to the entry.

Selecting the ESP8266 Thing Board

With the Board addon installed, all that's left to do is select "**SparkFun ESP8266 Thing Dev**" from the **Tools** > **Boards** menu.
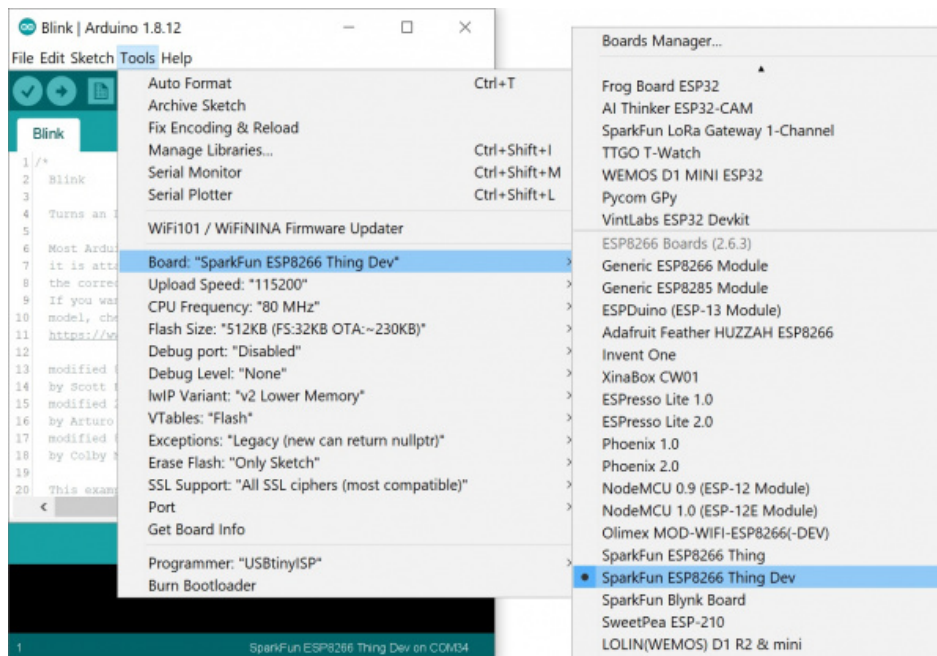
Then select your FTDI's port number under the **Tools** > **Port** menu.

Example Sketch: Blink

Example 1: Blink

To verify that everything works, try uploading the old standard: Blink. But instead of blinking pin 13, **toggle pin 5**, which is attached to the onboard LED.

```
#define ESP8266_LED 5

void setup()
{
  pinMode(ESP8266_LED, OUTPUT);
}
```
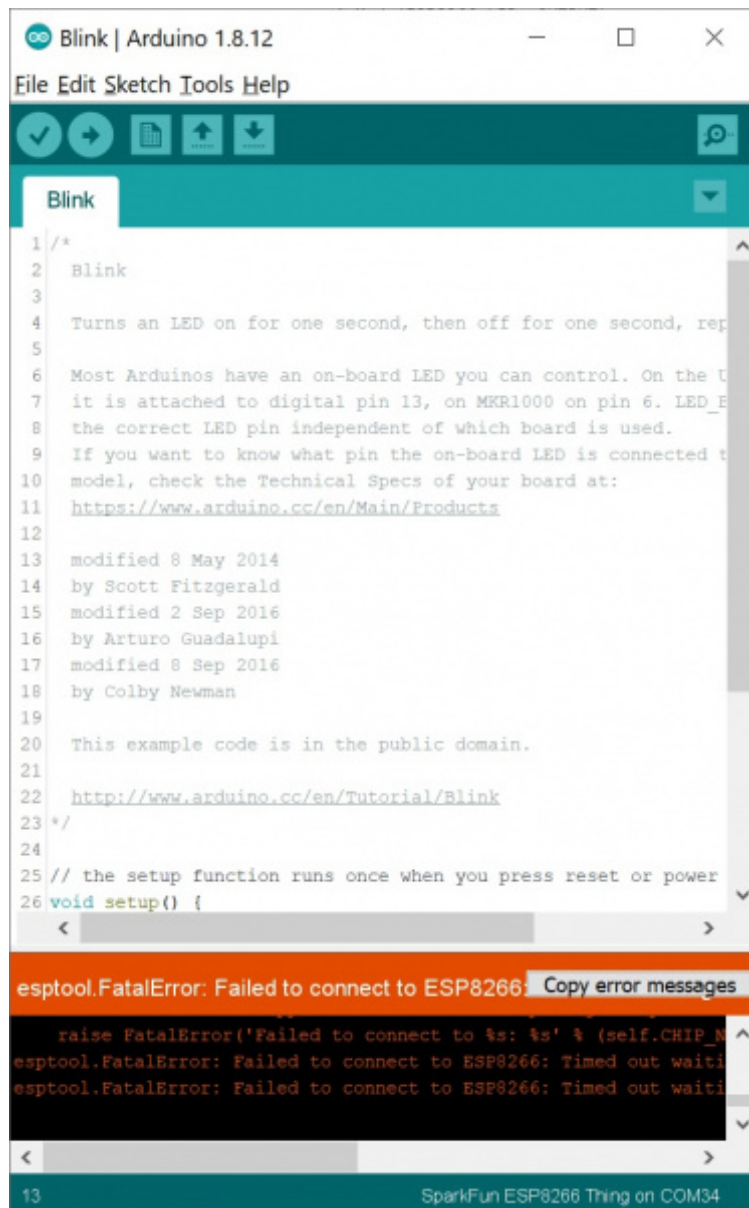
COPY CODE

```
void loop()
{
  digitalWrite(ESP8266_LED, HIGH); // LED off
  delay(500);
  digitalWrite(ESP8266_LED, LOW); // LED on
  delay(500);
}
```

Upload Fails – Troubleshooting

If you get the following error ending with something like

`esptool.FatalError: Failed to connect to ESP8266: Timed out waiting for packet header` :

Make sure that you select the **SparkFun ESP8266 Thing Dev** as the board. Selecting just the "*SparkFun ESP8266 Thing*" without the word

"**Dev**" can cause problems uploading when using the ESP8266 Community Board Add-On.
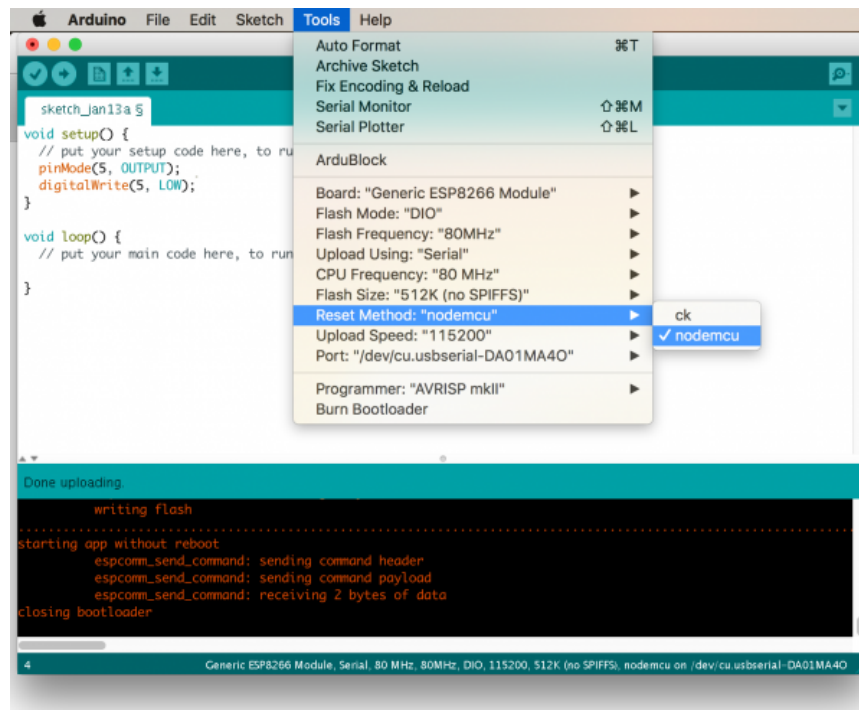
If every upload attempt results in an error ending with something like `error: espcomm_open failed` :
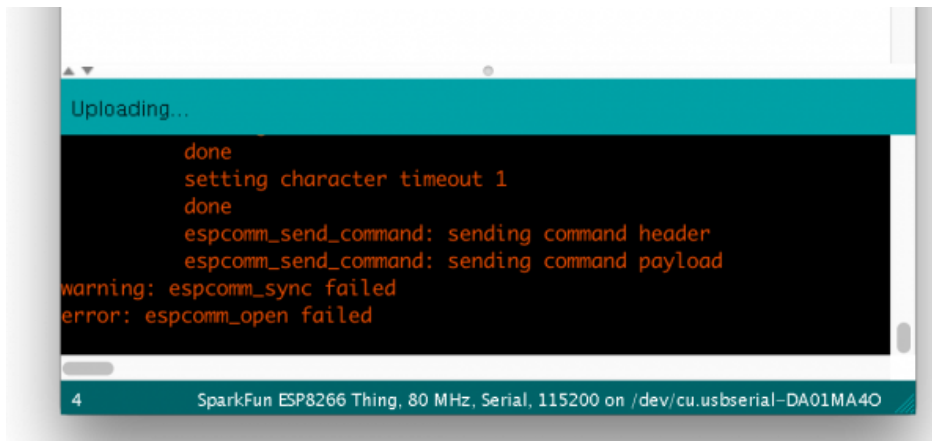

Try changing the board to **Generic ESP8266 Module**, and in the sub menus, make sure the following sub-menu's are also set:

- Flash Mode: DIO
- Flash Frequency: 80MHz
- Upload Using: Serial
- CPU Frequency: 80MHz
- Flash Size: **512K (no SPIFFS)**
- Reset Method: **nodemcu**

Faster Uploads! The serial upload speed defaults to **115200 bps**, which is reliable, but can feel a bit slow. You can increase the upload speed by a factor of about 8 by selecting **921600** under the **Tools** > **Upload Speed** menu.
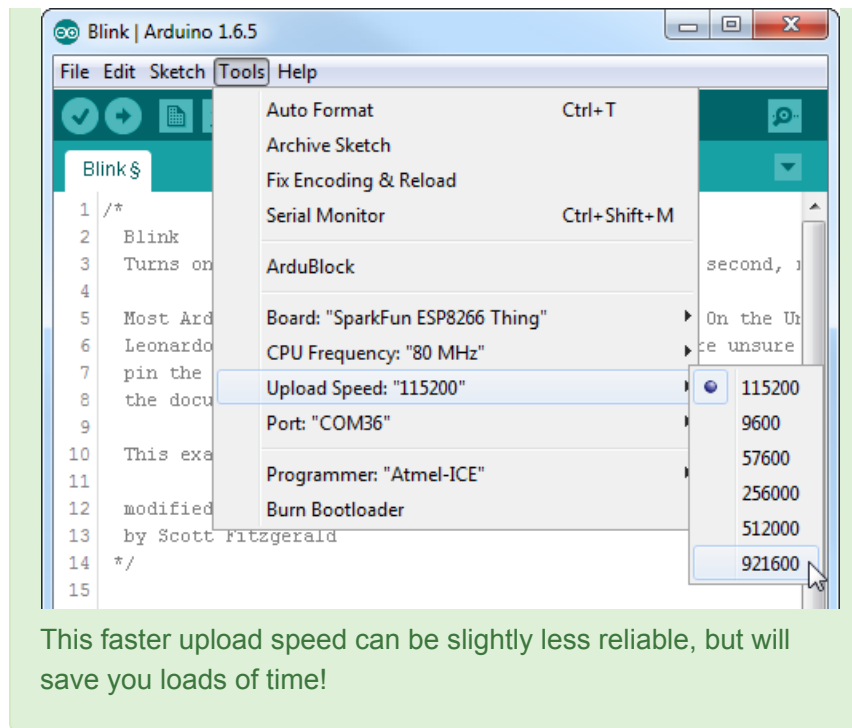
This faster upload speed can be slightly less reliable, but will save you loads of time!

There are still some bugs to be fleshed out of the esptool, sometimes it may take a couple tries to successfully upload a sketch. If you're still not having any luck uploading, try turning the board on then off, or unplug then replug the USB cable. If you still have trouble, get in touch with our amazing tech support team.

Example Sketch: Web Server

If we use the ESP8266 as a web *server*, you can use a web browser to control the ESP8266's LED, or read the status of its GPIO's.

This section provides two web server examples. The first example sets the ESP8266 up as a WiFi station, connecting to a WiFi router like the previous example. The other example sets the ESP8266 up as a WiFi **access point**,

so you can use a WiFi-enabled device to connect directly to the little WiFi chip.

Example 2: Station (and mDNS) Web Server

To begin, let's connect the Thing Dev Board back up to the WiFi access point you used in the previous example. Here's some example code setting up a web server and pointing a local mDNS domain of "thing.local" to your ESP8266. Make sure you **adjust the values for** `WiFiSSID` **and** `WiFiPSK` .

```
#include <ESP8266WiFi.h>                           COPY CODE
#include <ESP8266mDNS.h>


/////////////////////
// WiFi Definitions //
/////////////////////
const char WiFiSSID[] = "WiFiSSID";
const char WiFiPSK[] = "WiFiPSK";


/////////////////////
// Pin Definitions //
/////////////////////
const int LED_PIN = 5; // Thing's onboard, green LED
const int ANALOG_PIN = A0; // The only analog pin on the Thing
const int DIGITAL_PIN = 12; // Digital pin to be read

WiFiServer server(80);
```
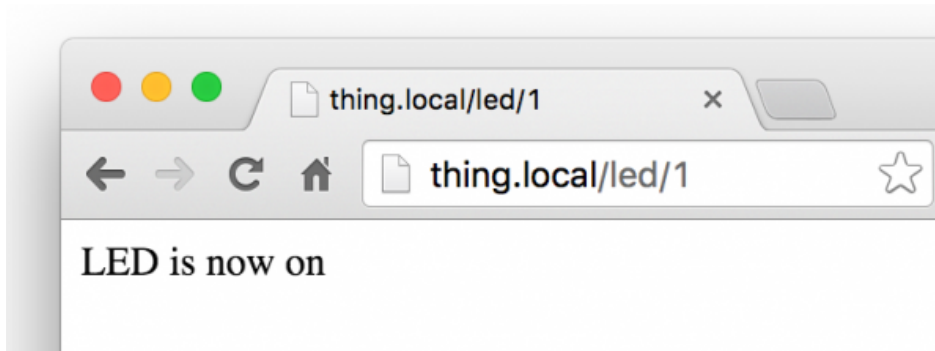
After uploading, find a device on the same WiFi network, and point it to one of these locations (the links below will only work if your device is on the same network):

- thing.local/read -- Read the status of the ADC and the digital status of pin 12.
- thing.local/led/0 -- Turn the LED on pin 5 off.
- thing.local/led/1 -- Turn the LED on pin 5 on.

The ESP8266 should server a web page, even if it's as simple as ensuring its LED is on.

> **Troubleshooting Tip:** If you get a web page error message relating to "*can't find sever or IP address*", this may be related to your web browser or a delay in your network's mDNS update. This can be bypassed by getting the IP address of the board using a serial terminal program and using that in place of "*thing.local*" in the web address. For example, the IP address of the ESP8266 Thing was **10.5.254.164**. Therefore, the address should be http://10.8.254.164/led/1 to turn the LED on.

Example 3: Access Point (AP) Web Server

Not only can the ESP8266 connect to a WiFi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This example demonstrates how to turn the ESP8266 into an **access point** (AP), and serve up web pages to any connected client.

Copy and paste the code from below in the Arduino IDE. Then hit the upload button.

```
#include <ESP8266WiFi.h>

/////////////////////
// WiFi Definitions //
/////////////////////
const char WiFiAPPSK[] = "sparkfun";

/////////////////////
// Pin Definitions //
/////////////////////
const int LED_PIN = 5; // Thing's onboard, green LED
const int ANALOG_PIN = A0; // The only analog pin on the Thing
const int DIGITAL_PIN = 12; // Digital pin to be read

WiFiServer server(80);

void setup()
```
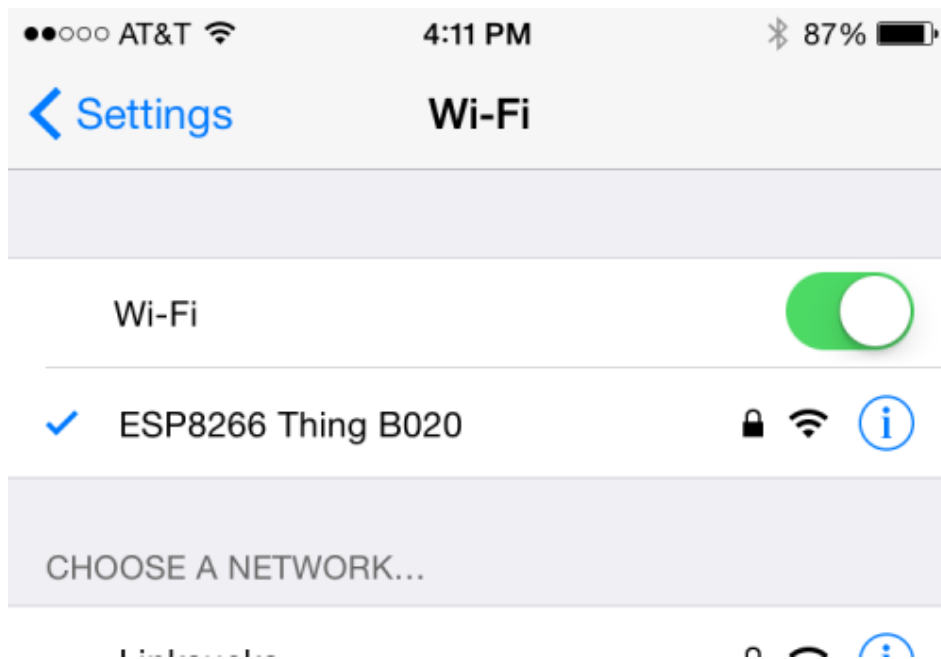
COPY CODE

After uploading this sketch, find another device that you can connect to a WiFi network -- phone, laptop, etc. Look for a network called "ThingDev-XXXX", where XXXX is the last 2 bytes of the Thing Development Board's MAC address.

The sketch sets the network's password to "sparkfun".

After connecting to your ESP8266's AP network, load up a browser and point it to `192.168.4.1/read` (unfortunately, mDNS doesn't work in AP mode). The Thing Dev Board should serve up a web page showing you its ADC and digital pin 12 readings:
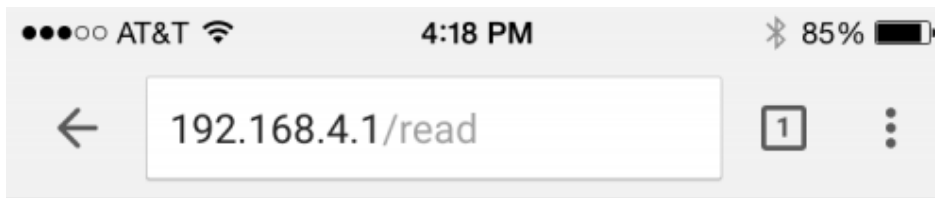
After that, give `192.168.4.1/led/0` and `192.168.4.1/led/1` a try, and keep an eye on the Thing's green LED while you do.

Auto Refresh

Tired of hitting the refresh button to update the I/O pin readings? Try uncommenting the line of code where it says: `s += "<meta http-equiv='refresh' content='1'/>\r\n";`. Then reupload the code to your ESP8266. This line of code will auto refresh the page every second.

```
  /*Note: Uncomment the line below to refresh
automatically
  *      for every 1 second. This is not ideal for large
pages
  *      but for a simple read out, it is useful for mon
itoring
  *      your sensors and I/O pins. To adjust the fresh
rate,
  *      adjust the value for content. For 30 seconds, s
imply
```

COPY CODE

```
    *          change the value to 30.*/
  //s += "<meta http-equiv='refresh' content='1'/>\r
\n";//auto refresh page
```

As always, check through the code comments to get a line-by-line breakdown of what's going on.

Example Sketch: Blink with Blynk

Example 4: Blinky with the Blynk App

The previous example is great for showing the nuts-and-bolts behind using HTTP to toggle outputs or read inputs. But if you're looking for a much simpler solution, we recommend checking out Blynk.

Blynk is a smartphone application that allows you to easily create "apps" that interact with Internet-connected hardware. It works with a wide variety of hardware platforms, including the Photon, Raspberry Pi, Arduino/Ethernet Shield, and, of course, the ESP8266. Here's a quick how-to on Blynk:

Get the App and Arduino Library

The Blynk app is available for both iOS and Android devices. Click one of the buttons below to get started downloading the app:

BLYNK FOR IOS   BLYNK FOR ANDROID

After downloading the app, create an account and log in. Welcome to Blynk!
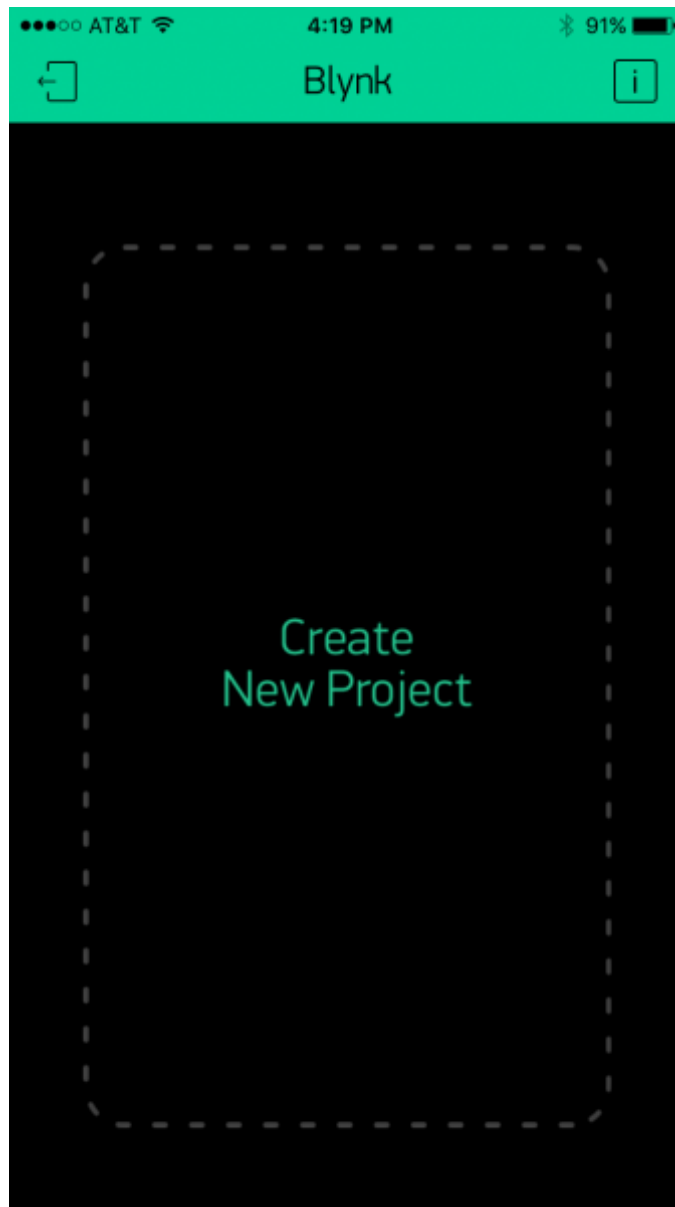
You'll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from

Blynk's GitHub repo, and follow along with the directions there to install the required libraries.

DOWNLOAD AND INSTALL THE BLYNK ARDUINO LIBRARY

Create a Blynk Project

Next, click the "Create New Project" in the app to create a new Blynk app. Give it any name you please, just make sure the "Hardware Model" is set to **ESP8266**.
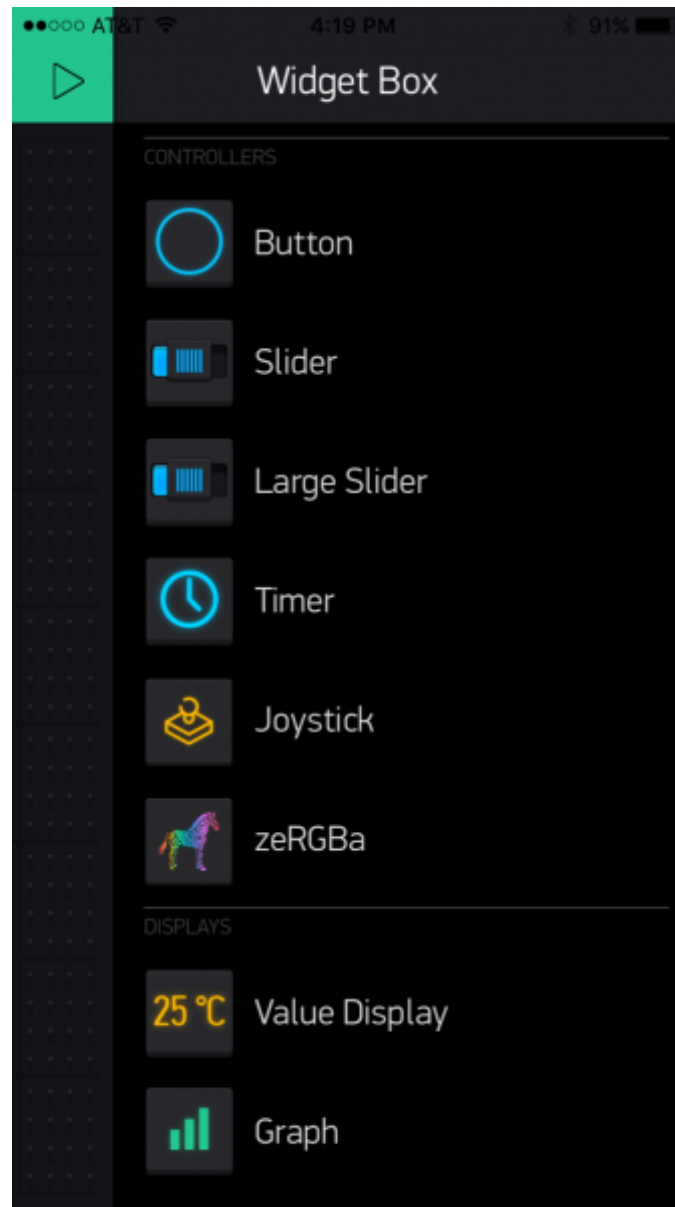
The **Auth Token** is very important -- you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it to yourself.
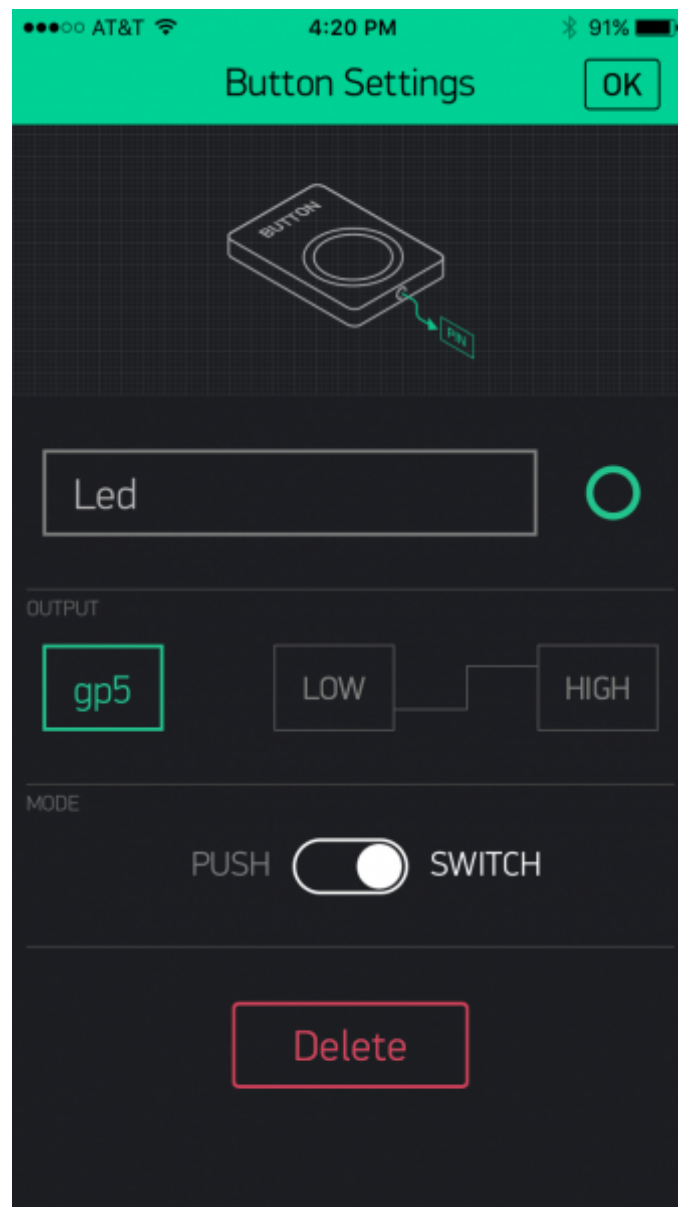
Add Widgets to the Project

Then you'll be presented with a blank new project. To open the widget box,
click in the project window to open.

Add a **Button**, then click on it to change its settings. Buttons can toggle outputs on the ESP8266. Set the button's output to **gp5**, which is tied to an

LED on the Thing Dev Board. You may also want to change the action to "Switch."

Upload the Blynk Firmware

Now that your Blynk project is set up, open Arduino and navigate to the **ESP8266_Standalone** example in the **File** > **Examples** > **Blynk** > **BoardsAndShields** menu.
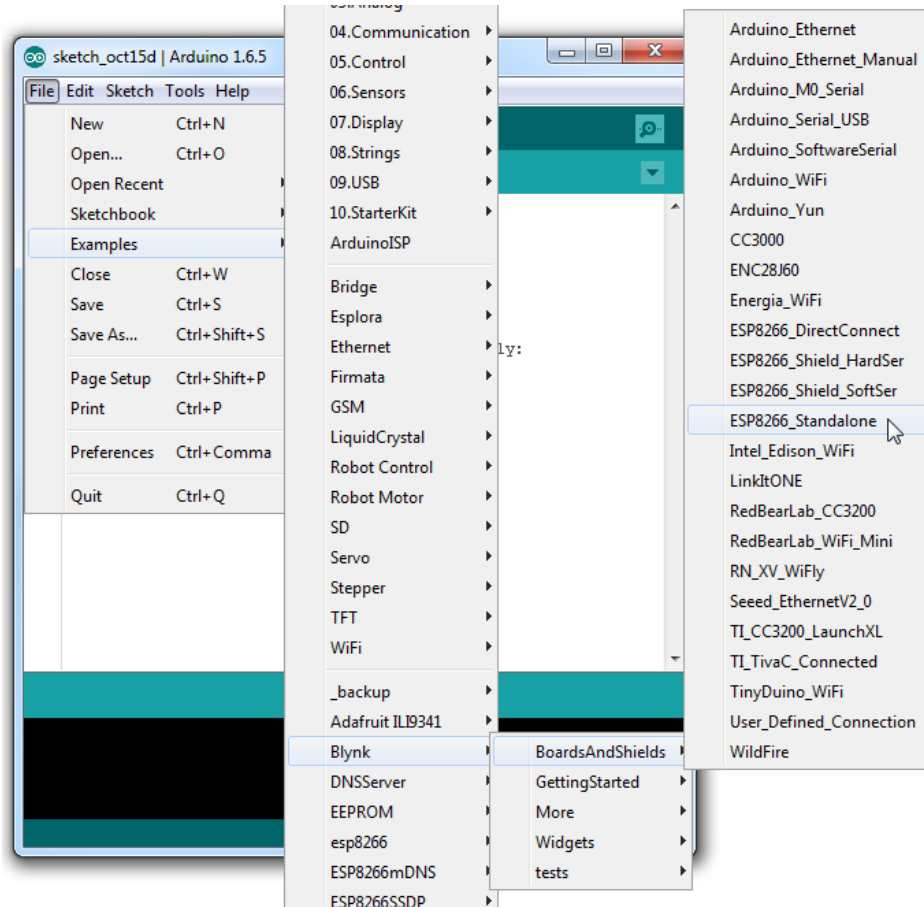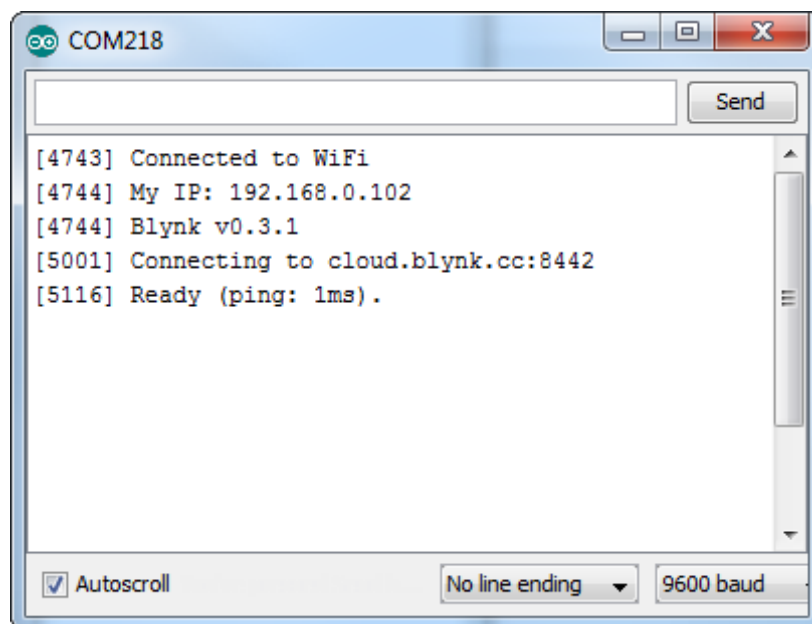

Before uploading, make sure to paste your **authoriazation token** into the `auth[]` variable. Also make sure to **load your WiFi network settings into the** `Blynk.begin(auth, "ssid", "pass")` function.

Then upload!

Run the Project

After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the "Ready (ping: xms)." message.

```
COM218                                          ─  ▣  ✕

                                              [ Send ]

[4743] Connected to WiFi
[4744] My IP: 192.168.0.102
[4744] Blynk v0.3.1
[5001] Connecting to cloud.blynk.cc:8442
[5116] Ready (ping: 1ms).




☑ Autoscroll          No line ending  ▾    9600 baud
```

Then click the "Run" button in the top right corner of the Blynk app. Press the button and watch the LED!

Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.
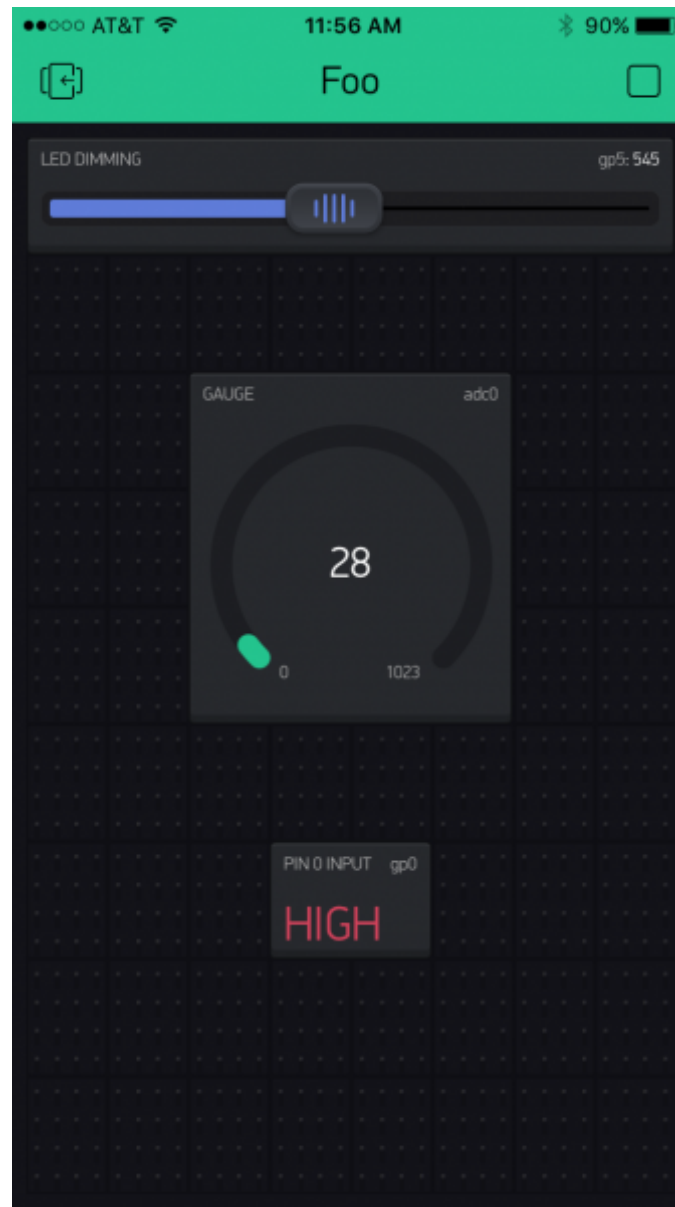
You can add analog output sliders, digital input monitors, and analog input gauges.

Using the ESP8266 in Arduino

If you've used Arduino in the past, there will be some new programming schemes to get used to in ESP8266 land. Here are a few of the most common gotchyas. For a more comprehensive reference, check out the ESP8266 Arduino Reference page.

READTHEDOCS: ESP8266 ARDUINO CORE REFERENCE

Pin Mappings

As with any other Arduino, the pin mappings printed on the board match the pin you read or write to. The TX and RX pins can be referenced as 7 and 8 respectively.
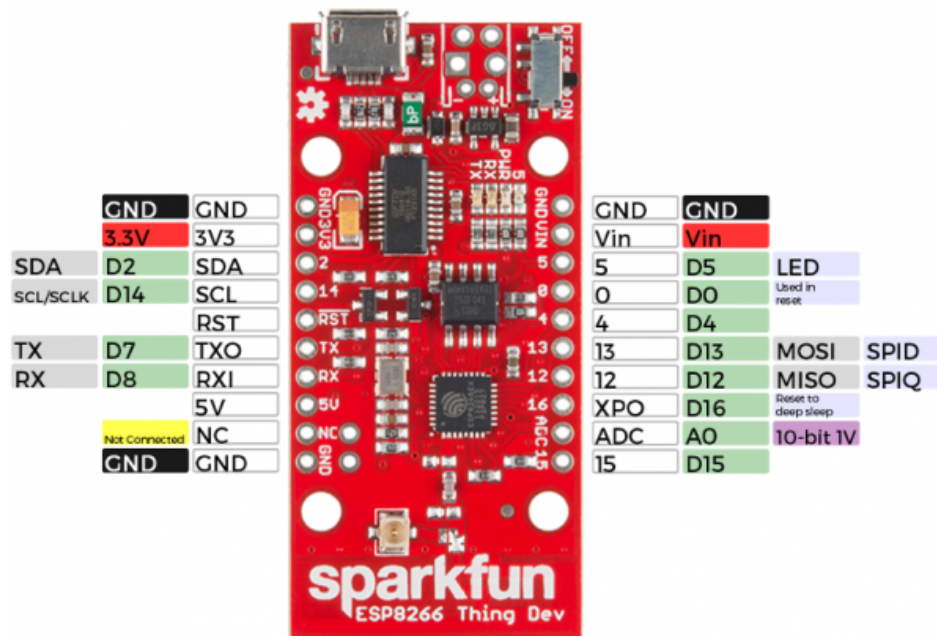
There's only one analog input pin, labeled *ADC*. To read the *ADC* pin, make a function call to `analogRead(A0)`. Remember that this pin has a weird

maximum voltage of 1V -- you'll get a 10-bit value (0-1023) proportional to a voltage between 0 and 1V.

All digital pins are also capable of PWM "analog" output. Use `analogWrite([pin], [value])` with a value between 0 and 1023 to dim LEDs with a 1kHz PWM signal.

Yielding

This is one of the most critical differences between the ESP8266 and a classic Arduino microcontroller. The ESP8266 runs a lot of utility functions in the background -- keeping WiFi connected, managing the TCP/IP stack, and performing other duties -- blocking these functions from running can cause the ESP8266 to crash and reset itself. To avoid these mysterious resets, **avoid long, blocking loops in your sketch**.

If you have a long loop in your sketch, you can **add a** `delay([milliseconds])` call within, to allow the critical background functions to execute. The ESP8266's `delay()` funciton, while of course delaying for a set number of milliseconds, also makes a quick call to the background functions.

The ESP8266 Arduino libraries also implement a `yield()` function, which calls on the background functions to allow them to do their thing. As an example, if your sketch is waiting for someone to press a button attached to pin 12, creating a loop like this will keep the ESP8266 from crashing:

```
pinMode(12, INPUT_PULLUP); // Set pin 12 as an input w/ pull-up
while (digitalRead(12) == HIGH) // While pin 12 is HIGH (not activated)
    yield(); // Do (almost) nothing -- yield to allow ESP8266 background functions
Serial.println("Button is pressed!"); // Print button pressed message.
```

COPY CODE

ESP8266WiFi Class

This is the ESP8266, so the WiFi class will probably be included in just about every sketch there is. If you've used the Arduino WiFi library before, the ESP8266 WiFi library will be very similar, there's just a few key differences:

- To **include the ESP8266 WiFi library** call `#include <ESP8266WiFi.h>` *not* `<WiFi.h>` .
- To **connect** to a network, like the normal WiFi library, call `WiFi.begin(NetworkSSID, NetworkPassword)` . You can also set the ESP8266 up as a WiFi access point by calling `WiFi.softAP(AP_SSID, AP_Password)` .
- To set the ESP8266's **mode**, which can be access point (AP), station (STA), or combo (the ESP8266 can do both at the same time!), call `WiFi.setMode([mode])` with either `WIFI_AP` , `WIFI_STA` , or `WIFI_STA_AP` as the parameter.

The examples earlier in this tutorial should have demonstrated all of these differences.

Libraries Available/Not Available and the Differences

A lot of the core Arduino libraries have been re-written to work for the ESP8266, including:

- **Wire** -- The ESP8266 should work with any I^{2}C sensor you can throw at it -- just use the same Wire API calls you're used to. There are a few differences:
  - Pin definition: The ESP2866 doesn't actually have any hardware I^{2}C pins -- those labeled on the Thing are the default, but you can actually use any two pins as SDA and SCL. Calling `Wire.begin()` will assume pins 2 and 14 are

SDA and SCL, but you can manually set them to any other pin by calling `Wire.begin([SDA], [SCL])`.
- **SPI** -- The ESP8266 Thing can control an SPI bus using function calls made standard by the Arduino SPI library.
  - An additional function to set the frequency -- `SPI.setFrequency([frequency])` -- is added. You may need to call that in your setup to slow the clock down from its default value. For example, `SPI.setFrequency(1000000)` will set the SPI clock to 1MHz.
  - The MISO, MOSI, and SCLK SPI pins are hard-coded and can't be moved, they are:

Pin Number	SPI Function
12	MISO
13	MOSI
14 (SCL)	SCLK
15	CS

Deep Sleep

The ESP8266 has a pretty decent low-power sleep mode -- operating around 70µA. To put the ESP8266 to sleep, use the `ESP.deepSleep(<microseconds>)` function.

```
ESP.deepSleep(30000000); // Sleep 30 seconds
```
COPY CODE

For it to wake itself back up, the ESP8266 requires an external connection between pin 16 and its RST pin. Use the handy "Sleep-EN" jumper to set this connection up.

Resources and Going Further

Now that you've successfully got your ESP8266 Thing Development Board up and running, it's time to incorporate it into your own project! For more information, check out the resources below:

- Schematic
- Eagle Files
- Graphical Datasheet (PDF)
- GitHub Repo (Design Files) - The ESP8266 Thing is open source hardware! If you need, or just want to look at, the PCB design files, you can find them in our hardware repo.
- SFE Product Showcase
    - SparkFun ESP8266 Dev Board
    - Product Showcase: SparkFun ESP8266 Thing & Thing Dev Starter Kits

An astoundingly awesome community has grown around the ESP8266. We owe them big time for the amazing Arduino addon they've cooperatively built. For all of your ESP8266 needs, we recommend checking out the esp8266.com Community Forum. In addition to that, here are a few ESP8266-related resources we've found incredibly helpful:

- ESP8266 GitHub User Repos -- Tons of incredible tools can be found here. From Crosstool (to compile your own Xtensa GCC, G++, etc.) to the ESP8266 Arduino GitHub Repo
- ESP8266 Community Wiki -- Related to the community forum, there's a good amount of information available in this wiki.
- NodeMCU Firmware and the NodeMCU Flasher -- NodeMCU is a popular firmware for the ESP8266. It implements a LUA-based interpreter on the ESP8266 MCU.
- Espressif Board Forums -- Espressif, the manufacturers of the ESP8266, have a forum of their own. You can sometimes find updated software development kit downloads, or other helpful links here.

- **Espressif GitHub Repos** -- Espressif is also somewhat active on GitHub. They host a couple versions of the SDK here.

Need a little project inspiration, now that you've got your ESP8266 Thing up-and-running? With its deep sleep ability, the Thing is a great foundation for a WiFi-based weather station, a friendly, huggable, interactive plushy. Maybe some of these related SparkFun tutorials will help spur some ideas.

Weather Station Wirelessly Connected to Wunderground

Build your own open-source, official Wunderground weather station that connects over WiFi via an Electric Imp.

Are You Okay? Widget

Use an Electric Imp and accelerometer to create an "Are You OK" widget. A cozy piece of technology your friend or loved one can nudge to let you know they're OK from half-a-world away.

LED Cloud-Connected Cloud

Make an RGB colored cloud light! You can also control it from your phone, or hook up to the weather!
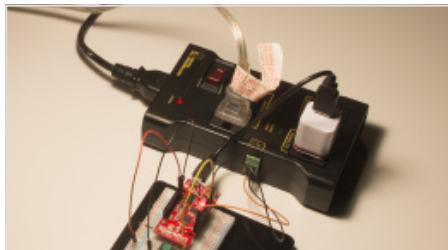
### Internet of Things Experiment Guide

The SparkFun ESP8266 Thing Dev Board is a powerful development platform that lets you connect your

hardware projects to the Internet. In this guide, we show you how to combine some simple components to remotely log temperature data, send yourself texts and control lights from afar.

Or check out SparkFun's project spotlight on the SparkFun Rogue Router for a Solar Powered File Server with the ESP8266:

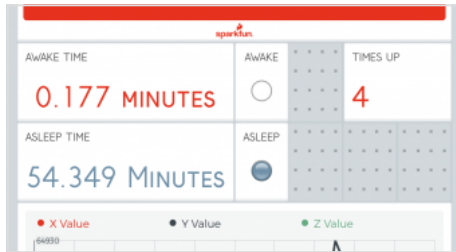Or check out some of these blog posts for ideas:
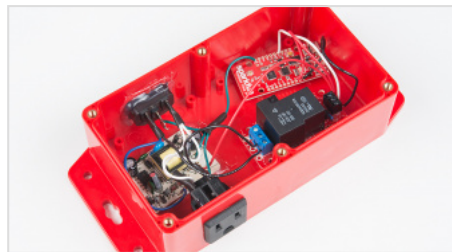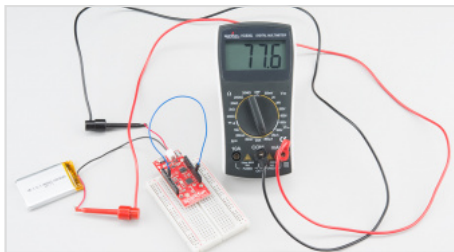
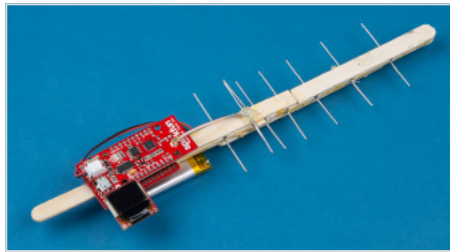### Enginursday: Deep Sleep Adventures

MAY 28, 2015

Enginursday: Light Up Your Home Life

JUNE 9, 2016

Baby Blynk Monitor Thing

SEPTEMBER 14, 2016

IoTuesday: WiFi Maximizer with DIY Yagi-Uda Antenna
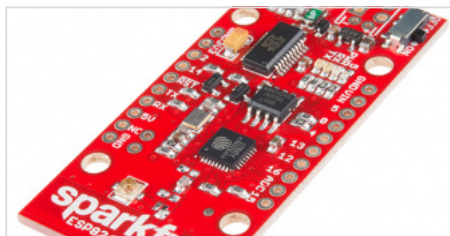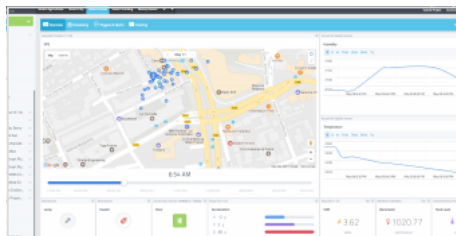
MARCH 28, 2017
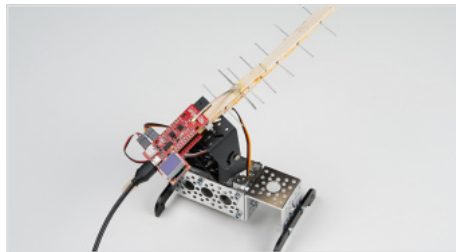
IoTuesday: Maximize Your WiFi With This One Weird Trick!

APRIL 18, 2017

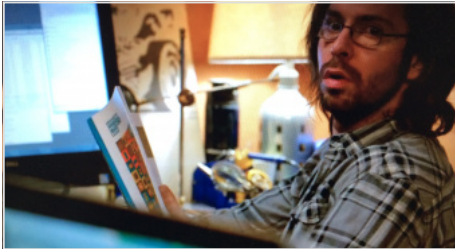### Enginursday: ComSphere

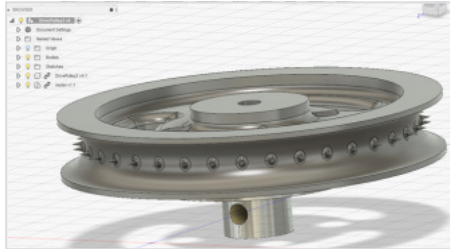MAY 18, 2017

Three IoT Platforms for Makers
JUNE 20, 2017

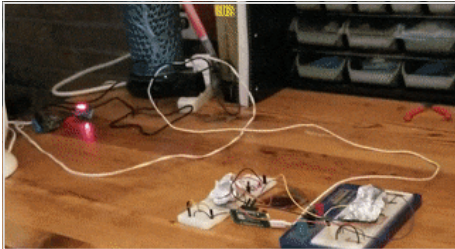### ESP8266 + Cayenne = <3
JULY 18, 2017

Enginursday: Gilfoyle's Bitcoin Price Alert
APRIL 12, 2018

Look, up in the... ceiling!
AUGUST 7, 2018

Automatic Light Switch on the Internet of Things
MARCH 21, 2019

SHARE

STAY IN TOUCH WITH US

Email address   **SUBMIT**

Weekly product releases, special offers, and more.

ABOUT SPARKFUN

Read Our Story

Press & Media

Success with SparkFun

SparkFun Education

Job Openings

PARTNER WITH US

See Our Partners

Become a Distributor/Reseller

Receive Volume Discounts

Explore Services

Apply for a Hardware Donation

SUPPORT

Customer Support

Purchase Orders & Payment Terms

Technical Assistance

FAQs

SITE INFORMATION

Terms of Service

Privacy Policy

Compliance

Site Map

Contact Us

SparkFun Electronics ® / **6333 Dry Creek Parkway, Niwot, Colorado 80503**