Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

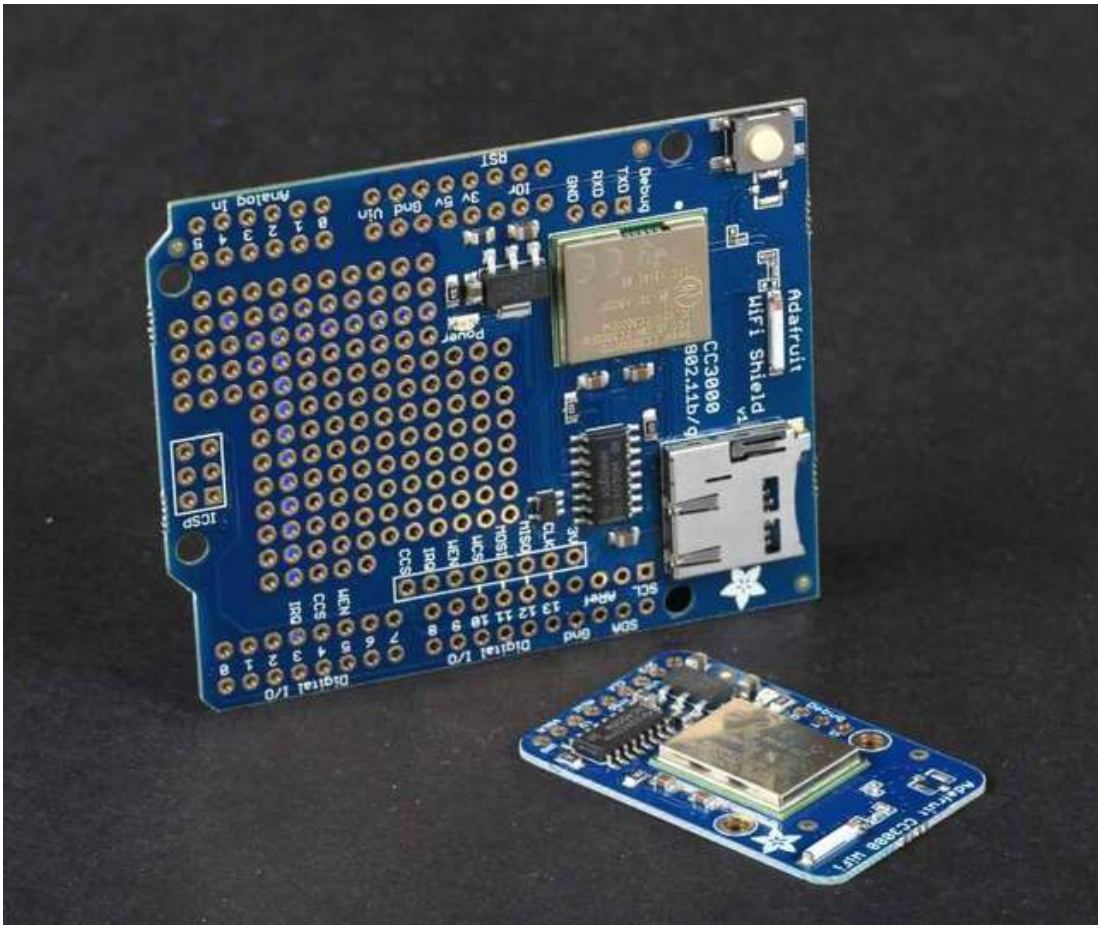
Adafruit CC3000 WiFi

Created by Rick Lesniak

Last updated on 2016-09-21 01:58:06 AM UTC

Guide Contents

Overview



The CC3000 WiFi module from Texas Instruments is a small silver package which finally brings easy-to-use, affordable WiFi functionality to your Arduino projects.

It uses SPI for communication (not UART!) so you can push data as fast as you want or as slow as you want. It has a proper interrupt system with IRQ pin so you can have asynchronous connections. It supports 802.11b/g, open/WEP/WPA/WPA2 security, TKIP & AES. A built in TCP/IP stack with a "BSD socket" interface supports TCP and UDP in both client and server mode, with up to 4 concurrent socket connections. The CC3000 does not support "AP" mode, it can connect to an access point but it cannot be an access point.

The CC3000 is available from Adafruit As a Breakout Board, and as an Arduino Shield.

Both the shield and the breakout board have an onboard 3.3V regulator that can handle the 350mA peak current, and a level shifter to allow 3 or 5V logic level. The antenna layout is identical to TI's suggested layout and we're using the same components, trace arrangement, and antenna so the board maintains its FCC emitter compliance (you'll still need to perform FCC validation for a finished product, but the WiFi part is taken care of). Even though it's got an onboard antenna we were pretty surprised at the range, as good as a smartphone's.

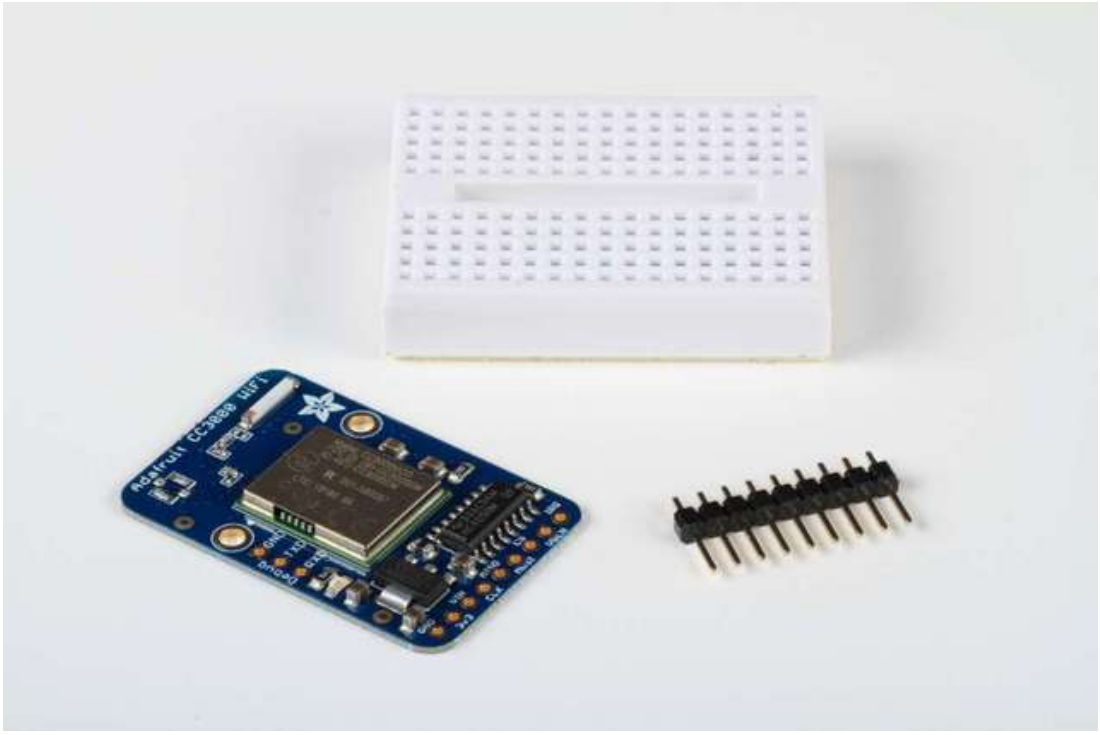
The shield also features a MicroSD socket, and a reset button.

AND, the shield supports the Arduino SPI passthrough header pins, so it's compatible with the Mega, Leonardo, and Due, right out of the box - no rewiring necessary! Just solder closed the MISO, SCK, and MOSI jumpers on the back of the shield.

Assembly and Wiring

Check out the next couple of pages for detailed instructions for setting up your CC3000 shield or breakout board!

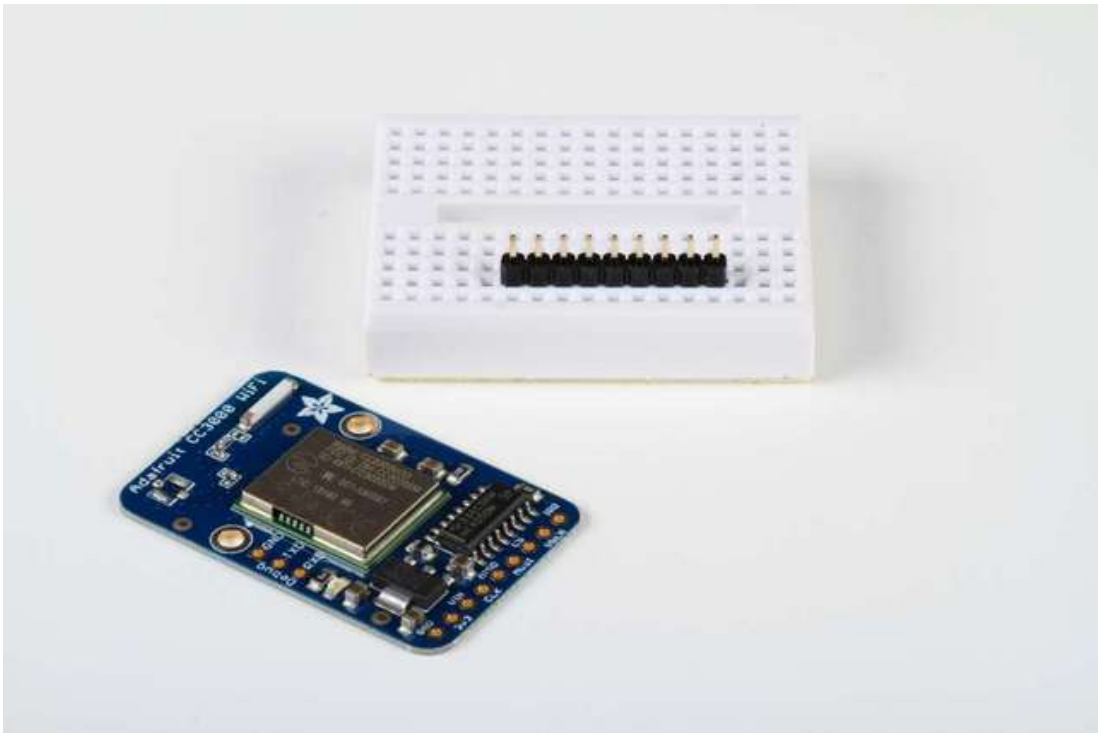
CC3000 Breakout



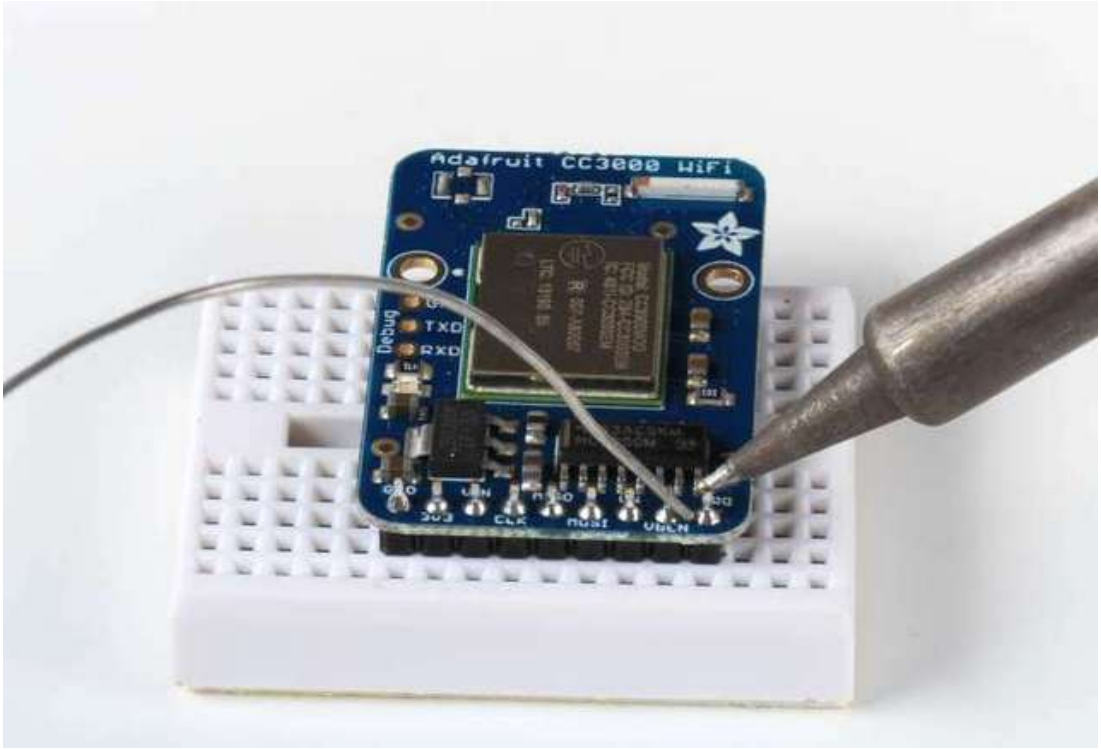
Assembly

The CC3000 breakout board ships with a strip of header pins. Snip off a 9-pin section and solder it to the 9 holes on the side of the board.

The easiest way to do this is to first insert the header pins into a breadboard, to hold them securely while you solder.



Set the breakout board over the pins, and carefully solder each pin (see our Guide To Excellent Soldering (http://adafru.it/aTk) for instructions and tips on getting the best results).

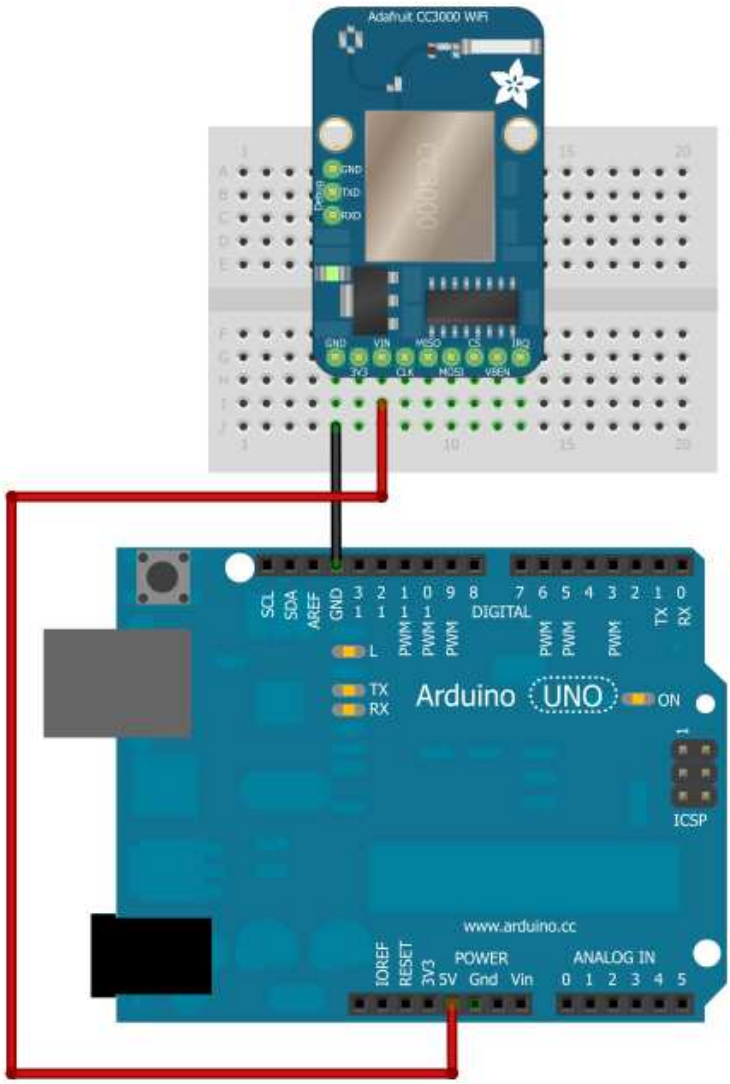


Wiring

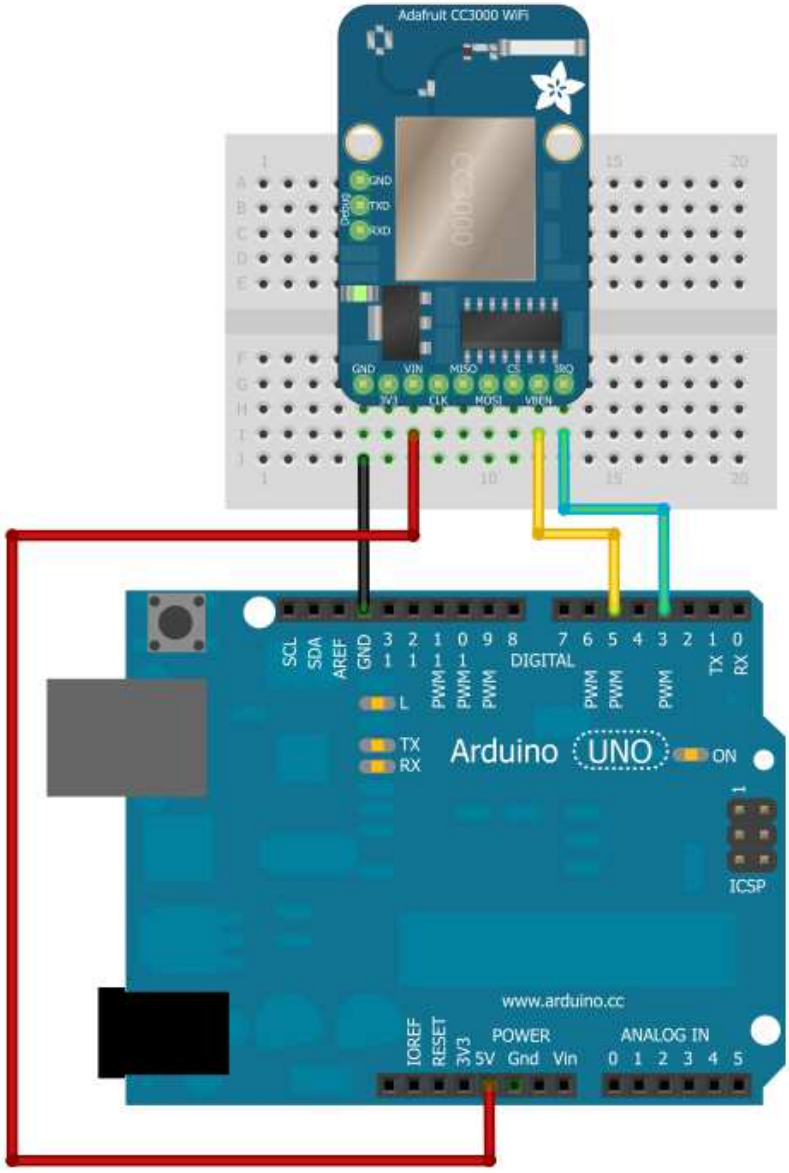
Use jumper wires to attach the CC3000 breakout board to your arduino:

Connect GND to one of the Arduino GND pins:
Connect Vin to Arduino +5V

NOTE: If using an Arduino Due, which is not tolerant of 5V on its input pins, you must instead connect the CC3000 3V3 pin to the Due's 3.3V power pin. Don't connect Vin to the Due's +5V!



Next, connect the enable and interrupt lines:
VBEN to Digital 5
IRQ to Digital 3



Now, connect SPI:

- CLK to Digital 13
- MISO to Digital 12
- MOSI to Digital 11
- CS to Digital 10

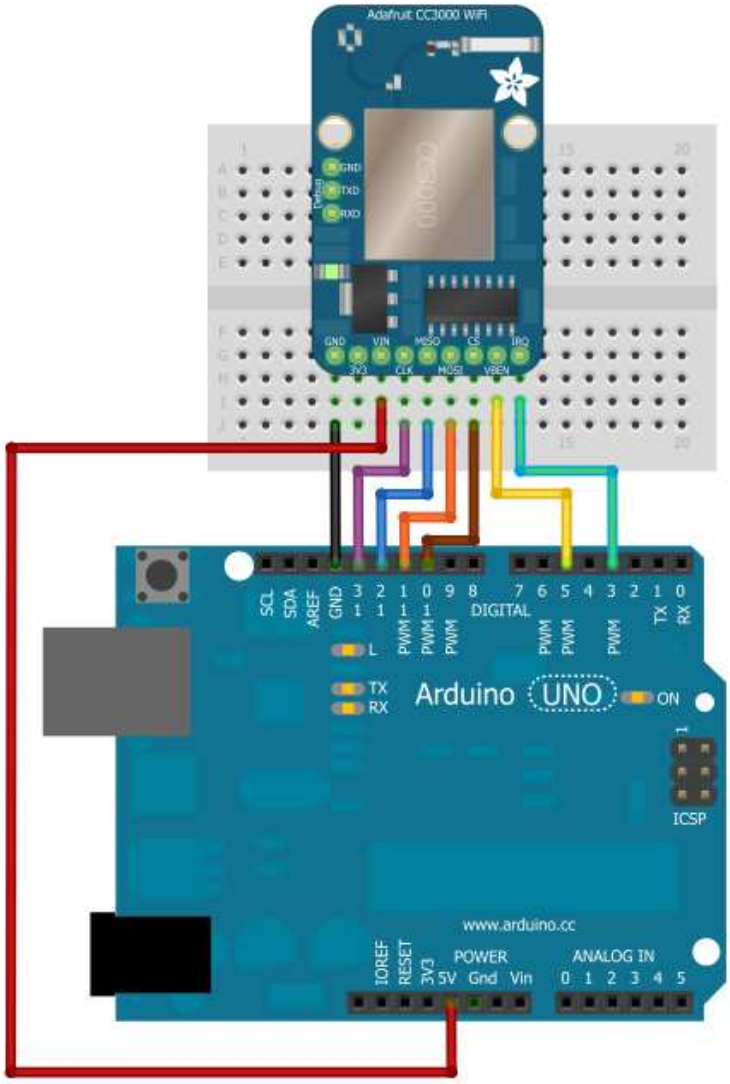
If you're using a Mega, you'll need to connect to the hardware SPI pins:

- CLK to Digital 52
- MISO to Digital 50
- MOSI to Digital 51
- CS to Digital 10

If you're using an Arduino Due, you'll need to connect to the hardware SPI pins. See the[excellent diagram in this forum post](http://adafru.it/dVz) if you aren't sure where the hardware SPI pins are located on the Due.

You want to connect to the SCK, MISO, and MOSI pins on the small 6 pin male header next to the Due's SAM3X8E processor:

- CLK to SPI SCK
- MISO to SPI MISO
- MOSI to SPI MOSI
- CS to Digital 10



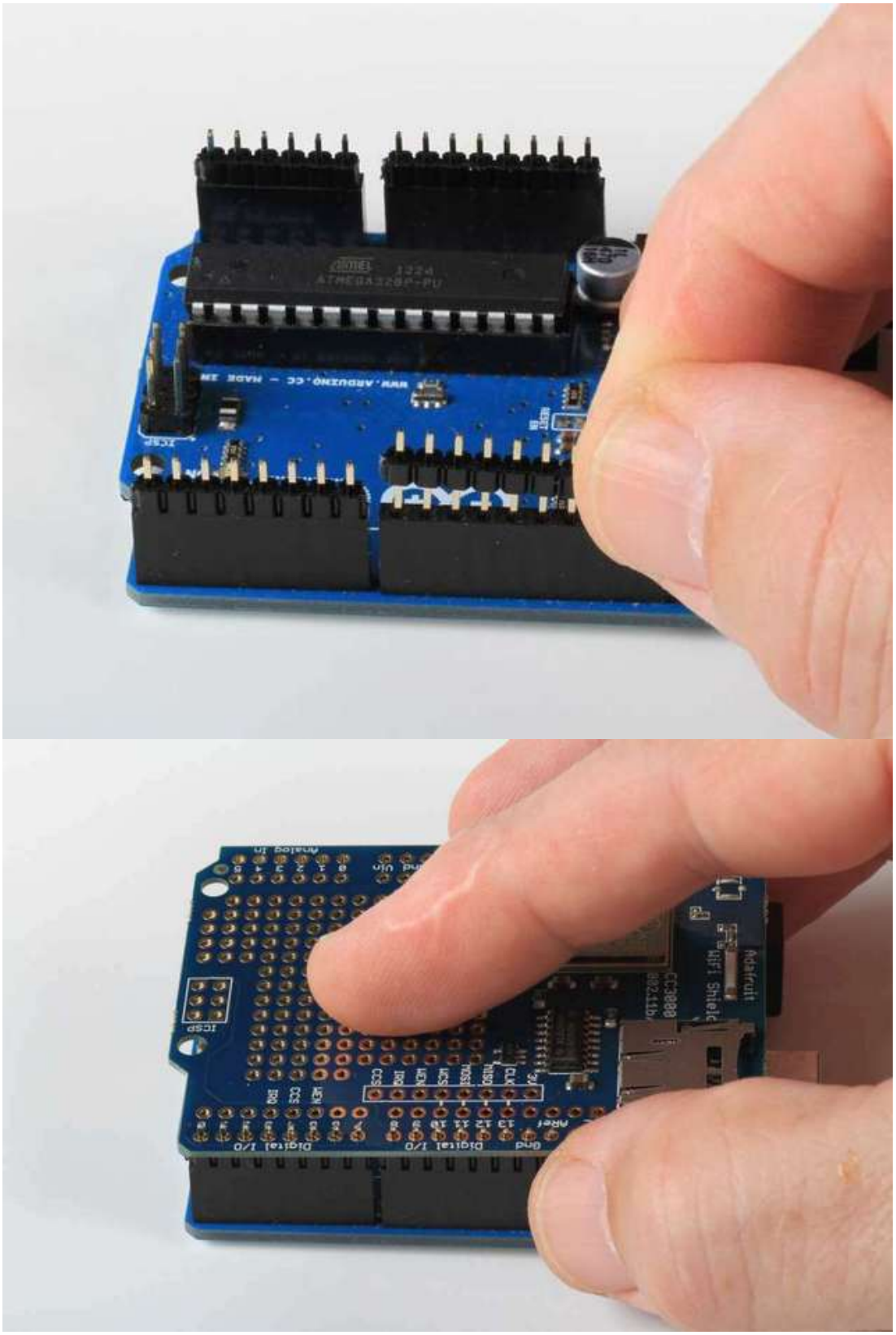
3.3v is an output from the breakout board's voltage regulator - we won't be connecting anything to it in this tutorial.

CC3000 Shield

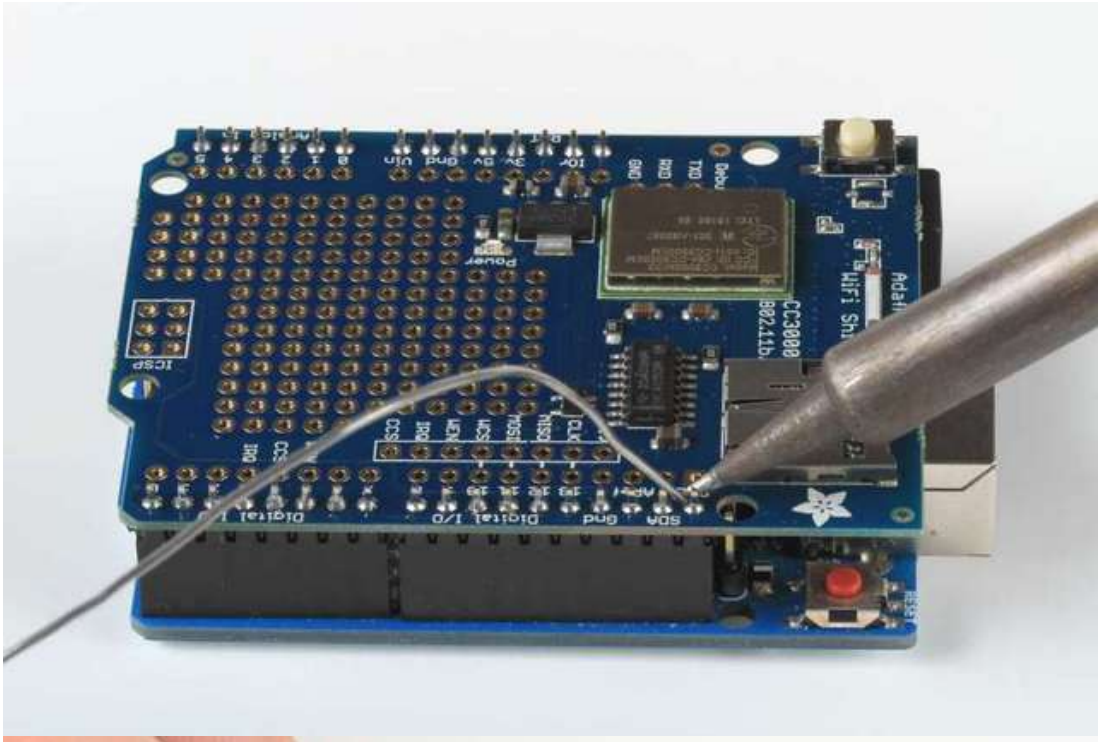
Assembly

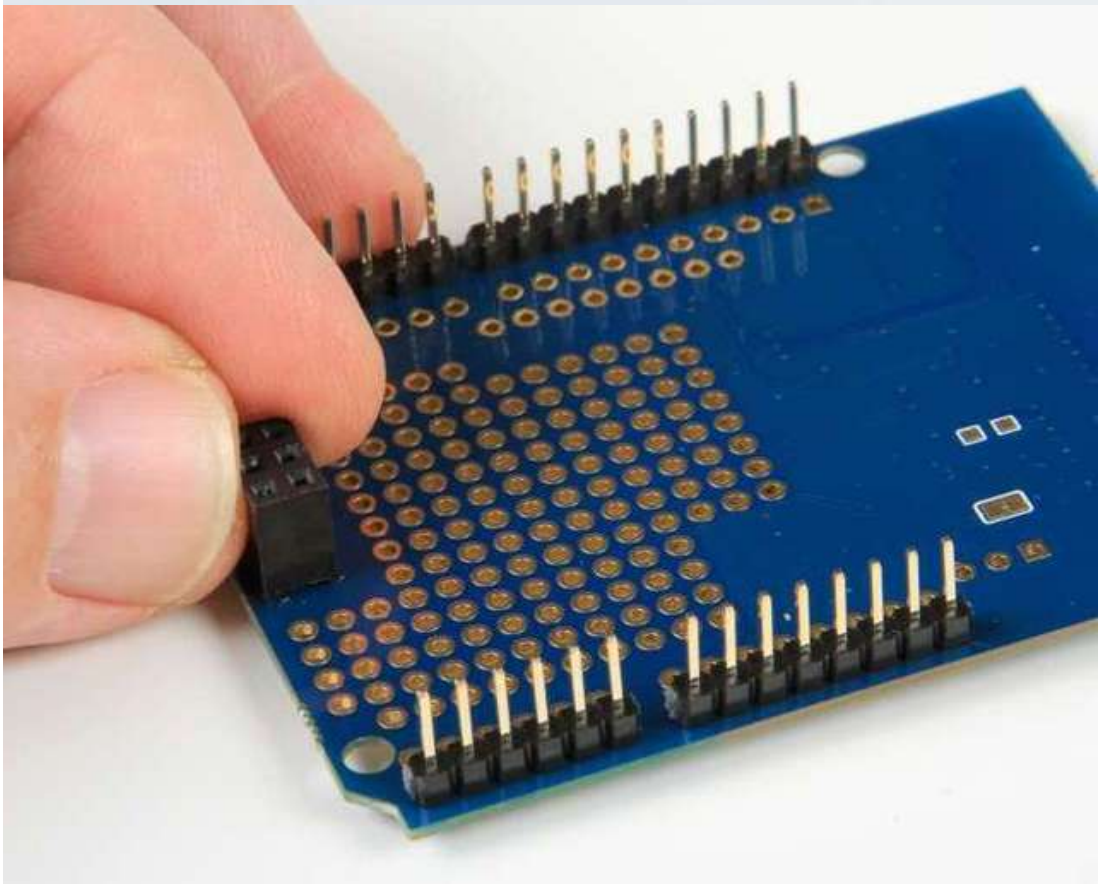
The CC3000 Shield ships with a strip of header pins.

Break off 6, 8, or 10-pin sections and insert them into the header sockets of your Arduino



Place the shield over the pins, and carefully solder each one in place (see our Guide To Excellent Soldering (http://adafru.it/aTk) for instructions and tips on getting the best results).





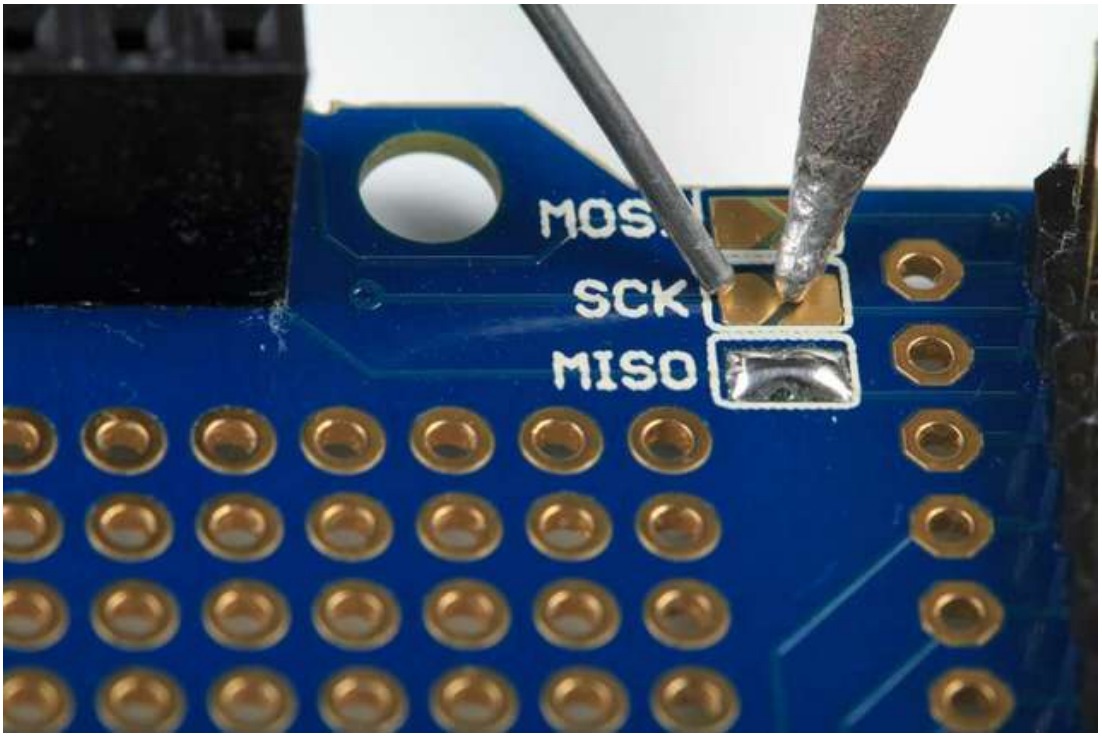
The shield also comes with a 2X3 pin female header socket. This will plug into the 2X3 ICSP pin header on your Arduino, to bring SPI up to the shield. This allows you to use the shield with an Arduino Mega, Leonardo, or Due without having to cut traces or solder jumper wires for SPI.

Set the socket header into the holes on the shield, then flip the shield over and solder the pins. The height of the header block matches the height of the rest of the shield header pins, so the block should be perfectly positioned for soldering!

By default, SPI through the ICSP header is not connected. To enable the ICSP header connections, you'll have to connect three solder jumpers on the bottom of the CC3000 shield.

Soldering these jumpers is required if you are using the shield with a Mega, Leonardo, or Due, but is not required for use with an UNO!

Simply melt a blob of solder, connecting the pads, on each of the three solder jumpers (keep your solder inside the white boxes - don't let the solder cross between boxes!)



And that's it! Your CC3000 Shield is ready to use. Move on to the next tutorial page to get started!


Adafruit
CC3000
802.11b/

Connections

Pinouts

The CC3000 is (electrically) fairly simple to use. The module requires an SPI connection, including a clock (CLK), data in from a microcontroller (MOSI) and data out to the microcontroller (MISO). It also uses a chip-select line (CS) for SPI to indicate when a data transfer as started

Along with the SPI interface, there is a power-enable type pin called**VBAT_EN** which we use to start the module properly and also an **IRQ** pin, which is the interrupt from the CC3000. The IRQ pin is required to communicate and must be tied to an interrupt-in pin on the Arduino. On the Mega/UNO, we suggest #2 or #3

On the CC3000 shield, we use the following pin connections

- SCK - #13
- MISO #12
- MOSI #11
- CS for CC3000 #10
- VBAT_EN #5
- CS for SD Card #4
- IRQ #3

On the breakout, be aware that the MISO (data out from module) pin does not go 'high impedance' when CS is driven high. Check the shield for how we use a 74AHC125 to manually tri-state this pin when it's shared with an SD card.

Optional Antenna

If you have a shield or breakout with a uFL connector (instead of an on-board ceramic antenna) you can use a uFL to RP-SMA (http://adafru.it/852) or uFL to SMA (http://adafru.it/851) (less common) adapter and then connect to any 2.4 GHz antenna (http://adafru.it/945). This is handy when you want to place the module in a box but have the antenna on the outside, or when you need a signal boost

Please note that when using an external antenna, the module is no longer FCC-compliant, so if you want to sell the product with FCC certification, it must be retested.

Using the CC3000

Make sure your Arduino is powered by a 1 amp or higher rated external power supply when using with the CC3000! Powering an Arduino + CC3000 from a computer/laptop USB port will lead to unstable behavior and lockups because the USB port can't supply enough power!
Use the Arduino 1.6.4 version or newer with the CC3000, EXCEPT for the firmware update sketches in the library examples folder. Use 1.0.6 for those!

Download the Library

We will start by downloading the Adafruit CC3000 Library, available from our GitHub repository (http://adafru.it/cFn).

You can download the latest ZIP file by clicking the button below.
Download the latest Adafruit_CC3000 library
http://adafru.it/cHe
Rename the uncompressed folder **Adafruit_CC3000**. Check that the **Adafruit_CC3000** folder contains **Adafruit_CC3000.cpp** and **Adafruit_CC3000.h;** also **ccspi.cpp, ccspi.h,** an **examples** folder, and a **utility** folder

Place the **Adafruit_CC3000** library folder your ***sketchbookfolder*/libraries/** folder. You may need to create the libraries subfolder if its your first library. Restart the IDE. You can figure out your ***sketchbookfolder*** by opening up the Preferences tab in the Arduino IDE.

If you're not familiar with installing Arduino libraries, please visit our tutorial: All About Arduino Libraries (http://adafru.it/aYM)!

Sample Sketches

The Adafruit CC3000 Library contains several example sketches, demonstrating different capabilities of the CC3000 along with some useful programming techniques.

To run the sample sketches, you'll have to edit them to include the SSID and password of your access point.

```
#define WLAN_SSID       "myNetwork"       // cannot be longer than 32 characters!
#define WLAN_PASS       "myPassword"
```

Also, make sure that the right wireless security scheme is selected (unsecured, WEP, WPA, or WPA2)

```
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

WEP with HEX Passphrases

If you are using WEP security, and your passphrase is a series of HEX digits, you can't simply enter it as a literal string. Instead you have to define is as an actual binary sequence.
For example, if your passphrase is 8899aabbccdd, you would define it as follows:

```
//  #define WLAN_PASS       "8899aabbccdd"  //don't do it this way!
//do it this way:
const char WLAN_PASS[] = {0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0x00};
```

Remember to append 0x00 to the declaration, after the passphrase, as shown in the example!
Be aware the library does not currently support WEP passphrases with 0x00 null characters! See this bug for more details: https://github.com/adafruit/Adafruit_CC3000_Library/issues/97

buildtest

buildtest

The **buildtest** sketch does a full test of core WiFi connectivity:

- Initialization
- SSID Scan
- Access Point connection
- DHCP address assignment
- DNS lookup of www.adafruit.com (http://adafru.it/aK0)
- Ping www.adafruit.com (http://adafru.it/aK0)
- Disconnect

It's a good idea to run this sketch when first setting up the module. It will let you know that everything is working correctly.

Before you run the sketch, edit it to replace the dummy SSID and password with your own:

```
#define WLAN_SSID       "yourNetwork"        // cannot be longer than 32 characters!
#define WLAN_PASS       "yourPassword"
```

If you're using WEP, the password should look like this:

const char WLAN_PASS[] = {0x1A, 0x2B, 0x3C, 0x4D, 0x5E, 0x00};

Since it's a collection of bytes not 'passphrase' style key
Also, make sure that the right wireless security scheme is selected (unsecured, WEP, WPA, or WPA2)

```
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Here's a sample of the Serial Monitor output of buildtest. You should see something similar:

```
Hello, CC3000!

RX Buffer : 131 bytes
TX Buffer : 131 bytes
Free RAM: 1237

Initialising the CC3000 ...
Firmware V. : 1.19
MAC Address : 0x08 0x00 0x28 0x01 0xA8 0x8A
Started AP/SSID scan


Networks found: 3
================================================
SSID Name    : Extreme
RSSI         : 58
Security Mode: 3

SSID Name    : Express
```

```
RSSI        : 59
Security Mode: 3

SSID Name    : fios63
RSSI        : 57
Security Mode: 3

================================================

Deleting old connection profiles

Attempting to connect to fios63
Started AP/SSID scan



Connecting to fios63...Waiting to connect...Connected!
Request DHCP

IP Addr: 192.168.1.23
Netmask: 255.255.255.0
Gateway: 192.168.1.1
DHCPsrv: 192.168.1.1
DNSserv: 192.168.1.1
www.adafruit.com -> 207.58.139.247

Pinging 207.58.139.247...5 replies
Ping successful!


Closing the connection
```

Make sure you can see and recognize all of the access points around, connect to the access point, get a good connection with DHCP, can do a DNS lookup on www.adafruit.com (http://adafru.it/aK0) and ping it successfully. If all this works, then your hardware is known good!

WebClient

WebClient

The WebClient sketch does a test of the TCP client capability:

- Initialization
- Optional SSID Scan (uncomment code section to enable)
- Access Point connection
- DHCP address assignment
- DNS lookup of www.adafruit.com (http://adafru.it/aK0)
- Optional Ping of www.adafruit.com (http://adafru.it/aK0) (uncomment code section to enable)
- Connect to website and print out webpage contents
- Disconnect

The sketch connects to www.adafruit.com (http://adafru.it/aK0) and opens a special webpage (http://adafru.it/cFo) we have prepared for this example. It reads the contents of the page and prints that out to the Serial Monitor.

Before you run the sketch, edit it to replace the dummy SSID and password with your own:

```
#define WLAN_SSID       "yourNetwork"        // cannot be longer than 32 characters!
#define WLAN_PASS       "yourPassword"
```

Also, make sure that the right wireless security scheme is selected (unsecured, WEP, WPA, or WPA2)

```
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Here's a sample of the Serial Monitor output of WebClient. You should see something similar:

```
Hello, CC3000!

Free RAM: 1157

Initializing...
Started AP/SSID scan



Connecting to fios63...Waiting to connect...Connected!
Request DHCP

IP Addr: 192.168.1.23
Netmask: 255.255.255.0
Gateway: 192.168.1.1
DHCPsrv: 192.168.1.1
DNSserv: 192.168.1.1
www.adafruit.com -> 207.58.139.247

Connect to 207.58.139.247:80
-------------------------------------
HTTP/1.1 200 OK
```

```
Date: Thu, 12 Sep 2013 11:04:02 GMT
Server: Apache
Access-Control-Allow-Origin: http://learn.adafruit.com
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Accept-Encoding, Authorization, Referer, User-Agent
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 1728000
Last-Modified: Thu, 27 Jun 2013 14:13:27 GMT
Accept-Ranges: bytes
Content-Length: 74
Connection: close
Content-Type: text/html

This is a test of the CC3000 module!
If you can read this, its working :)
-------------------------------------

Disconnecting
```

Once you get this working, you can change the webpage you want to access to any kind of webpage on the Internet

ntpTest

ntpTest

The ntpTest sketch does a test of the library's SNTP (Simple Network Time Protocol) client:

- Initialization
- SSID Scan
- Access Point connection
- DHCP address assignment
- SNTP time synchronization
- Extract and print current time and date

The sntp client performs a time synchronization with servers from us.pool.ntp.org and pool.ntp.org. You can also optionally provide it the addresses of one or two of your own time servers. The ntpTest sketch tries time.nist.gov first, before falling back to one of the pool servers.

The client also breaks out the synchronized network time into a structure containing current date and time fields. The sketch formats and prints this information to the Serial Monitor.
To avoid unnecessary loading of NTP servers, please perform the time synchronization as infrequently as possible. Once per day or longer should be plenty to maintain reasonably accurate time.
Before you run the sketch, edit it to replace the dummy SSID and password with your own:

```
#define WLAN_SSID       "yourNetwork"        // cannot be longer than 32 characters!
#define WLAN_PASS       "yourPassword"
```

Also, make sure that the right wireless security scheme is selected (unsecured, WEP, WPA, or WPA2)

```
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Here's a sample of the Serial Monitor output of ntpTest. You should see something similar:

```
    Hello, CC3000!

Free RAM: 843

Initialising the CC3000 ...
Firmware V. : 1.19

Deleting old connection profiles

Attempting to connect to fios63
Started AP/SSID scan



Connecting to fios63...Waiting to connect...Connected!
Request DHCP
UpdateNTPTime
Current local time is:
7:18:52.65445
Thursday, September 12, 2013
```