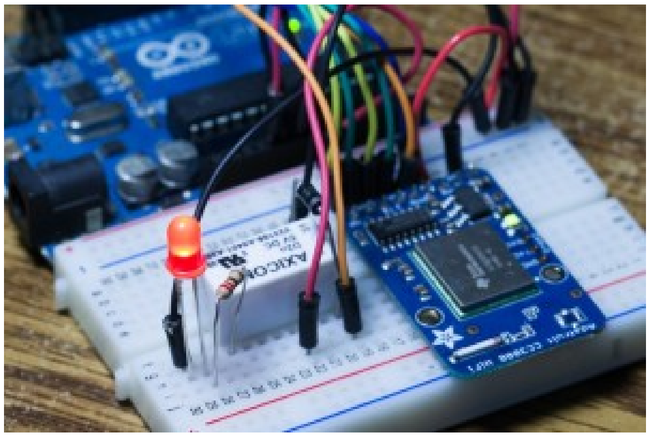
Open Home Automation

Blog

Wireless Relay Control with Arduino & the CC3000 WiFi chip

September 4, 2013 / 35 Comments / in Arduino, Tutorials, WiFi / by Marco Schwartz


[http://openhomeautomation.net/wp-content/uploads/2013/09/IMG_7517.jpg]

Controlling relays is a central piece in any home automation system. With them, you can easily switch on and off any electrical device in your home, like lights. This project is the second part of a series showing you how to use Arduino & the CC3000 WiFi chip for home automation purposes. The first part was dedicated to how to read data wirelessly from a temperature

& humidity sensor [http://openhomeautomation.net/arduino-wifi-cc3000/] .

In this project, we are going to see how to wirelessly control a relay from your computer or your smartphone, to build a wireless-controlled light switch for example. To make things more efficient, we are going to create a small web server running on the Arduino board, that will receive the commands coming from your computer. As usual, the whole code for this project can be found inside our GitHub repository [https://github.com/openhomeautomation/arduino-relay-wifi-server] . Let's dive in !

Hardware requirements

The whole project is based on the Arduino platform, so of course you will need an Arduino board. I really recommend using the Arduino Uno [http://www.amazon.com/gp/product/B006GX8IAY/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B006GX8IAY&linkCode=as2&ta 20] board for this project, as it is the only board that is currently compatible with the CC3000 library at the time this article was written.

Then, you need the CC3000 chip. As in the previous article, I recommend using the Adafruit CC3000 breakout board [http://www.amazon.in/exec/obidos/ASIN/B00H232MUE/petewill0e8-21] , which is the only one I tested that worked without problem. It is nice and compact, has voltage regulators onboard, as well as an onboard antenna. You can also make your own breakout board, as there are many PCB layout available online.

You also need a relay and some components around it. I used a 5V relay [http://www.amazon.com/gp/product/B009T2M012/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B009T2M012&linkCode=as2&ta 20] for this project, so it can directly be connected to the Arduino Uno board. You also need a standard rectifier diode [http://www.amazon.com/gp/product/B00ATNGEQW/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B00ATNGEQW&linkCode=as2& 20] to protect the relay, and one LED [http://www.amazon.com/gp/product/B004UMA60S/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B004UMA60S&linkCode=as2&t 20] & a 220 Ohm resistor [http://www.amazon.com/gp/product/B00B5LNEF6/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B00B5LNEF6&linkCode=as2&ta 20] to test if the relay is working correctly.

Finally, you need a breadboard and some jumper wires [http://www.amazon.com/gp/product/B008A9UORM/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B008A9UORM&linkCode=as2& 20] to make the connections between the different parts.

Software requirements

For this project, you just need the usual Arduino IDE [http://arduino.cc/en/main/software] , the Adafruit's CC3000 library [https://github.com/adafruit/Adafruit_CC3000_Library] , and the CC3000 MDNS library [https://github.com/adafruit/CC3000_MDNS] . We are also going to use the aREST library [https://github.com/marcoschwartz/aREST] to send commands to the relay via WiFi. To install a library, just download the required folder, and put it into your /Arduino/libraries/ folder.

You will also need a web server running on your computer. I recommend using MAMP [http://www.mamp.info/en/index.html] for OS X, EasyPHP [http://www.easyphp.org/] for Windows, and visiting this page [https://help.ubuntu.com/community/ApacheMySQLPHP] for Linux.

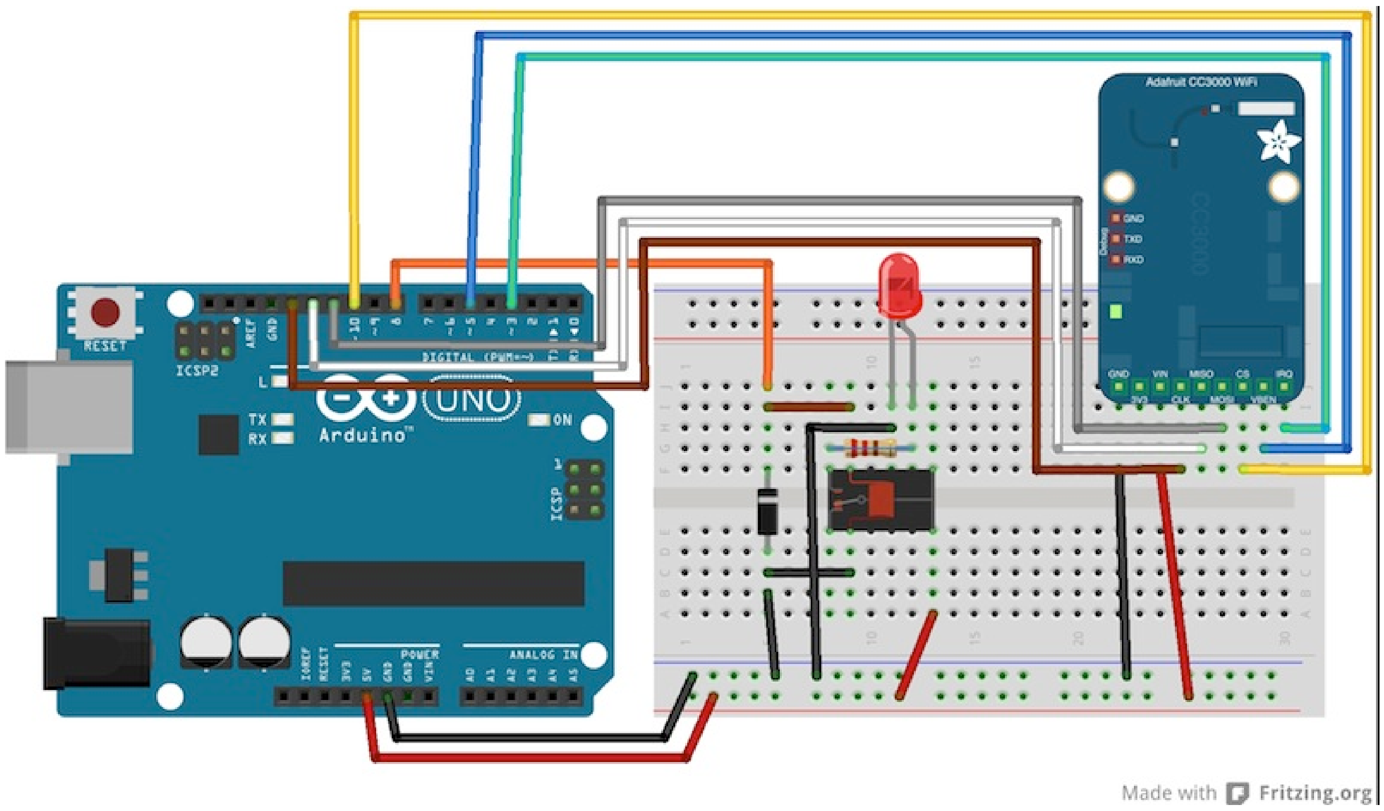
Hardware configuration

The hardware configuration for this project is actually not that complicated, thanks to the good informations that you will find on the CC3000 breakout board. Connect the IRQ pin of the CC3000 board to pin number 3 of the Arduino board, VBAT to pin 5, and CS to pin 10. Then, you need to connect the SPI pins to the Arduino board: MOSI, MISO, and CLK go to pins 11,12, and 13, respectively. Finally, take care of the power supply: Vin goes to the Arduino 5V, and GND to GND.

Let's now connect the relay. After placing the relay on the breadboard, you can start identifying the two important parts on your relay: the coil part which command the relay, and the switch part where we will attach the LED. First, connect pin number 8 of the Arduino board to one pin of the coil, and the other pin to the ground of the Arduino board. You also have to place the rectifier diode (anode connected to the ground pin) over the pins of the coil to protect your circuit when the relay is switching.

Then, connect the +5V of the Arduino board to the common pin of the relay's switch. Finally, connect one of the other pin of the switch (usually, the one which is not connected when the relay is off) to the LED in series with the 220 Ohm resistor, and connect the other side of the LED to the ground of the Arduino board.

The following picture summarizes the hardware connections:



[http://openhomeautomation.net/wp-content/uploads/2013/09/cc3000_relay_bb.jpg]

Testing individual components

I detailed the configuration of the CC3000 breakout board in a previous article [http://openhomeautomation.net/arduino-wifi-cc3000/] , so we are simply going to test the relay in this part. But don't worry, we are going to see the CC3000 & WiFi configuration again when testing the whole system.

You can test the relay with the following sketch:

```
const int relay_pin = 8; // Relay pin

void setup() {
  Serial.begin(9600);
  pinMode(relay_pin,OUTPUT);
}

void loop() {

  // Activate relay
  digitalWrite(relay_pin, HIGH);

  // Wait for 1 second
  delay(1000);
```

```

   // Deactivate relay
  digitalWrite(relay_pin, LOW);

  // Wait for 1 second
  delay(1000);
}
```

The code is pretty self-explanatory, you can just upload it to the board and the relay should switch states every seconds, and the LED should switch on and off accordingly.

Putting it all together

Let’s now control the relay wirelessly using the CC3000 WiFi chip. The software for this project is based on the TCP protocol, just like the WiFi weather station project I mentioned before. However, for this project, the Arduino board will be running a small web server, so we can “listen” for commands coming from the computer. We will first take care of the Arduino sketch, and then we will see how to write the server-side code and create a nice interface.

First, the Arduino sketch. The goal here is to connect to your WiFi network, create a web server, check if there are incoming TCP connections, and then change the state of the relay accordingly. I will only put the important parts of the code in this article, you can find the whole code here [https://github.com/openhomeautomation/arduino-relay-wifi-server] . You need to import the right libraries:

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include <CC3000_MDNS.h>
#include <Ethernet.h>
#include <aREST.h>
```

Then, you need to define inside the code what is specific to your configuration: WiFi name & password, and the port for the TCP communications (I used 80 here):

```
// WiFi network (change with your settings !)
#define WLAN_SSID       "yourNetwork"       // cannot be longer than 32 characters!
```

```
#define WLAN_PASS       "yourPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2 // This can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_S

// The port to listen for incoming TCP connections
#define LISTEN_PORT 80
```

We can then create the CC3000 instance, server and aREST instance:

```
// Server instance
Adafruit_CC3000_Server restServer(LISTEN_PORT);

// DNS responder instance
MDNSResponder mdns;

// Create aREST instance
aREST rest = aREST();
```

In the setup() part of the sketch, we can now connect the CC3000 chip to the network:

```
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
```

Now you can ask, how will the computer know where to send the data ? One way would be to run the sketch once, then get the IP address of the CC3000 board, and modify the server code again. But we can do better, and that’s where the CC3000 MDNS library comes into play. We will basically assign a fixed name to our CC3000 board with this library, so we can write down this name directly into the server code. This is done with this piece of code:

```
if (!mdns.begin("arduino", cc3000)) {
    while(1);
}
```

We also need to listen for incoming connections:

```
restServer.begin();
```

Now, we will code the loop() function of the sketch that will be continuously executed. We first have to update the mDNS server:

```
mdns.update();
```

The server running on the Arduino board will basically wait for incoming connections & handle the requests:

```
Adafruit_CC3000_ClientRef client = restServer.available();
rest.handle(client);
```

Then, the server side. The main part will be an HTML file, called interface.html, which will host two buttons to command the relay:

```
<div class="relayBlock"><span class="relayTitle">Relay</span>
    <button class="btn btn-block btn-lg btn-primary" type="button" id="1" onClick="relay
    <button class="btn btn-block btn-lg btn-danger" type="button" id="2" onClick="relayCl
</div>
```

The buttons are linked to some JavaScript code, that will call a PHP file that will communicate with the Arduino board. I won’t get into the details of JavaScript, but if you want to learn more about this language, I recommend following the JavaScript for Beginners course [http://click.linksynergy.com/link?id=3rziCtUCdbo&offerid=323058.8324&type=2&murl=https%3A%2F%2Fwww.udemy javascript%2F] on Udemy. This is the code for the JavaScript file, which is basically the function called when a button is clicked:

```
function relayClick(clicked_id){

    if (clicked_id == "1"){
        $.get( "update_relay.php", {
        command: "/digital/8/1"} );
    }

    if (clicked_id == "2"){
        $.get( "update_relay.php", {
        ommand: "/digital/8/0"} );
    }
}
```

This PHP code is stored in the curl.php file, and we get the desired command from the interface with:

```
$command = $_GET['command'];
```

Then, the code establishes the connection with the Arduino board with:

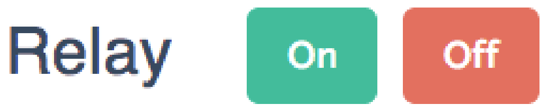
```
$service_url = 'http://arduino.local'. $command;
$curl = curl_init($service_url);
```

The PHP code then sends the request to the server:

```
curl_setopt($curl, CURLOPT_IPRESOLVE, CURL_IPRESOLVE_V4 );
$curl_response = curl_exec($curl);
curl_close($curl);
```

I also added some CSS to make things look a bit better. The complete code for this project can be found on our GitHub repository [https://github.com/openhomeautomation/arduino-relay-wifi-server] .

Finally, it's time to test everything. Make sure your web server is running on your computer. At this point, you can also upload the Arduino sketch to the board and open the serial monitor. Then, simply open the file interface.html on your local server (make sure you are accessing the folder from localhost) and press any button you want. On the Arduino side, you should see that the board is connecting to your WiFi network, getting an IP, and then waiting for incoming connections. Try now to press a button on the web interface: it should change the state of the relay nearly instantly. This is what you should see inside your browser:



[http://www.openhomeautomation.net/wp-content/uploads/2013/09/Screen-Shot-2014-05-04-at-09.45.30.png]

In case this doesn't work, it could be that your computer doesn't support the mDNS server (that allow the access to the Arduino board using the arduino.local name). But no worries: the sketch will actually print out the IP address of the CC3000 chip every time it starts. If you are having problems with the project, simply copy this IP address and replace the arduino.local with this IP address inside the PHP file.

Finally, you can also use your smartphone or tablet to use your WiFi-controlled relay. Simply enter the local IP address of your computer (where the web server is running) on your mobile device browser (and don't forget the port, for example 192.168.0.4:80), and navigate in your project's folder. You should see the exact same interface, and be able to control the relay from your device! If you managed to get it working, bravo, you just built a WiFi-controlled light switch! Of course, you can control much more than lights with this project. Just make sure your relay supports the power required for the device you want to control, and you're good to go!

Update 04/05/14: The project was updated and is now using the aREST library. It greatly simplified the code and made the project works more reliably. Let me know if you encounter any issues with the new code!

This is the list of the components that were used in this project:

- Arduino Uno [http://www.amazon.com/gp/product/B006GX8IAY/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B006GX8IAY&linkCode=as2& 20]
- Adafruit CC3000 breakout board [http://www.amazon.in/exec/obidos/ASIN/B00H232MUE/petewill0e8-21]
- 5V relay [http://www.amazon.com/gp/product/B009T2M012/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B009T2M012&linkCode=as2 20]
- Rectifier diode [http://www.amazon.com/gp/product/B00ATNGEQW/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B00ATNGEQW&linkCode=as 20]
- LED [http://www.amazon.com/gp/product/B004UMA60S/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B004UMA60S&linkCode=as2 20]
- 220 Ohm [http://www.amazon.com/gp/product/B00B5LNEF6/ref=as_li_qf_sp_asin_il_tl?

ie=UTF8&camp=1789&creative=9325&creativeASIN=B00B5LNEF6&linkCode=as2
20]

- Breadboard and some jumper wires [http://www.amazon.com/gp/product/B008A9UORM/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B008A9UORM&linkCode=as
20]

Receive the latest news about home automation

Over 100 home automation enthusiasts already joined our list !

Do you want to receive the latest news about open-source hardware and home automation ? Then join our mailing list today by entering your name and email below. It is 100% free and you can unsubscribe at any time if you're not satisfied.

First Name

Enter your name

Email

Enter your email

SUBSCRIBE !

Share this entry

35

REPLIES

ben

September 19, 2013 at 7:28 pm

If I understand correctly, this runs a local web server on your computer that the arduino/cc3000 connects to and pushes data onto it?

If so, what challenges would there be to put the webserver itself on the arduino/cc3000 itself? Could tools like espruino and node.js help?

Esprinuo I believe is an opensource javascript interpreter.. which if the webserver is on a remote computer, could serve updating javascript pages for the arduino to process.

In the case of node.js, a webserver could be implemented in javascript, but perhaps requires more processing power.

Reply

Marco Schwartz
September 24, 2013 at 10:40 am

Hey Ben,

Exactly! I guess the main challenge comes from the CC3000 library, from what I heard the Adafruit library does not yet support the web server capabilities of the CC3000 chip. But the hardware would allow it, and it works for example on the official Arduino WiFi shield, so there is nothing that would prevent that from working in the future.

Reply

Guilherme
October 18, 2013 at 2:10 am

Doesn't it Arduino Yun attend these necessities? In fact I just ordered one to do some experiments running ASW EC2 instances as my remote web servers.

Reply

Marco Schwartz
November 5, 2013 at 9:16 am

Hello, the Arduino Yun would also work as well. As it's all HTTP based communication, most of the code would probably work as well.

Reply

Yazin

December 11, 2013 at 9:18 am

Thanks for the awesome tutorial! I signed up to the mailing list so I don't miss any more awesomeness.

By the by, I found this script here that runs a UDP server on the Arduino: http://www.adafruit.com/forums/viewtopic.php?f=19&t=44652#p226199

Joe

September 25, 2013 at 2:20 am

Hi Marco,

Thanks for the tutorial. I learned A LOT.

After loading everything up on the arduino, running the webserver, and getting to your web interface I have run into a problem. When I click on the On or Off button I end up on a 404 Page not found. It seems to be attempting to load simple_relay.php which I cannot find in the Github directory. Am I missing a file?

Reply

Marco Schwartz

September 25, 2013 at 12:14 pm

Hello Joe,

Indeed, there was some problems with filenames, thanks for the comment! It is now fixed on the GitHub repository.

Reply

rommel acosta
December 7, 2013 at 3:36 am

how can i connect 8 relay in one arduino wifi ac power controller i need it in my thesis i have no idea for that can you give me complete instruction and materials to buy.
thnks dude….
hope u reply me on that pls..

Reply

Marco Schwartz
December 14, 2013 at 7:18 pm

Hello,

An Arduino can handle 8 relays without much problems, you will just have to modify the code to use 8 relays. For complete instructions and materials to buy, please send me an email directly.

Reply

David
December 26, 2013 at 10:51 am

In this example we have predefined SSID and Password. can't this assigned dynamically while the device switched on .
How can we tackle this problem? How can i make user configurable ssid and password.

Reply

Marco Schwartz
January 2, 2014 at 11:22 am

Hi David,

It's definitely possible to set the SSID/password while the device is connected, this feature of the CC3000 chip is called smartconfig. You can see more details here:

http://learn.adafruit.com/adafruit-cc3000-wifi/smartconfigtest

Reply

afterthought
January 8, 2014 at 8:38 pm

Hi Marco,

All parts are seemingly working-but not together.

1)I placed the server.php into the "sites" folder. I ran it in the terminal. I get a "socket created,Socket bin OK waiting for data" on the terminal

2) Apache,PHP,MySQL,Web sharing are on

3)Arduino with your relay_upd sketch running and the Serial Monitor "Awaiting response…Time to connect,send and read 15008"

4) In order to get to the relay.php I put into the browser the number from SystemPreference->Sharing->WebSharing
address WHICH IS DIFFERENT FROM THE CONNECTION NUMBER ON THE Serial Monitor in its last 2 digits.(1.2 vs
0.1) I click on relay.php and I get the ON and OF Buttons and 3 warning messages.

When I click on the On or Off buttons-nothing happens. The serial monitor keeps sending the above described messages.

Please, help

Thanks

Reply

Marco Schwartz

January 27, 2014 at 11:27 am

Hello,

I never encountered that error, but I plan to modify the tutorial to change the communication mode to TCP so it will be much simpler to use.

Reply

afterthought

January 11, 2014 at 10:22 pm

Hi Marco,

I am still struggling to make the relay work.

I put in a few Serial.println in the middle of your Sketch:

Except for the first printline "rA" nothing returns from the other two.

This is the Serial Monitor output:

```
Started AP/SSID scan

Connecting to NETGEAR96…Waiting to connect…Connected to WiFi network!
Requesting address from DHCP server…OK
rÁ=

Attempting connection…
Connect to 192.168.1.2:9999
connected!
Issuing request…
Awaiting response…Time to connect, send & read:
15010
rÁ=
Attempting connection…
Connect to 192.168.1.2:9999
connected!
Issuing request…
Awaiting response…Time to connect, send & read:
15011
rÁ=
```

This is the part of your Sketch which was modified by placing in 3 Serial.println .

```
void loop() {

// Send request to get relay state
relayBuffer[0] = ‘r’;
Serial.println(relayBuffer); //I put it in for checking, line 75
send_request_udp();

// Change relay state accordingly
if (relayBuffer[0] == ’0’) {
digitalWrite(relay_pin,LOW);
Serial.println(relay_pin); //I put it in for checking, line 81
}

if (relayBuffer[0] == ’1’) {
digitalWrite(relay_pin,HIGH);
Serial.println(relay_pin); //I put this in for checking, line 87
}
}
```

The “server_php” is puting out

```
Waiting for data …
192.168.1.4 : 1643 — rWaiting for data …
192.168.1.4 : 1644 — rWaiting for data …
192.168.1.4 : 1645 — rWaiting for data …
```

```
192.168.1.4 : 1709 — rWaiting for data …

192.168.1.4 : 1710 — rWaiting for data …

192.168.1.4 : 1711 — rWaiting for data …

192.168.1.4 : 1712 — rWaiting for data …

192.168.1.4 : 1713 — rWaiting for data …

192.168.1.4 : 1714 — rWaiting for data …

192.168.1.4 : 1715 — rWaiting for data …

192.168.1.4 : 1716 — rWaiting for data …

192.168.1.4 : 1717 — rWaiting for data …

192.168.1.4 : 2211 — Waiting for data …

192.168.1.4 : 2212 — Waiting for data …

192.168.1.4 : 2213 — Waiting for data …

192.168.1.4 : 2214 — Waiting for data …
```

There is nothing in the “relay_state.txt” file.

Where do I have the problem??? I would appreciate your help

Many thanks

Reply

Luke

January 24, 2014 at 11:47 am

Hi Marco,

Your site, blog and example code is very valuable and interesting, thanks for sharing your experiences 😉

I am interested in developing a system which uses the CC3000 for bi-directional communication between the Arduino and server – for example we exchange packets with the server to send a command to the Arduino, eg switch relay state, and we send packets back to the server – for example logging temperature from sensors, both at the same time, together.

This sort of resembles your CC3000 weather station demonstration and your CC3000 relay control demonstration combined together into a single system with both relay and sensor.

I was wondering if you might have some example code or software for this sort of application, or any suggestions or comments about what I would need to look out for in order to modify your code to handle this situation in an elegant way.

Is it worth keeping TCP for the sensor data and UDP for the relay data, or perhaps moving both to UDP for both?

Reply

Marco Schwartz
January 27, 2014 at 11:17 am

Hello Luke,

Thanks for your nice message ! I've developed what you mentioned indeed, I manage to have bi-directional WiFi transmission by running a server on the Arduino board with the CC3000 chip. However the code is not stable enough yet to publish a tutorial about it. I would suggest looking at the CC3000 server examples in the CC3000 library. I would also suggest to drop UDP and only use TCP, it is much easier to code and fast enough in the latest version of the WiFi library.

Reply

Jiose
February 6, 2014 at 3:59 pm

Hi Marco!

I was just wondering how to get this work in TCP mode like the previous tutorial? It seems I am having problems with UDP connection. I'm new to this stuff so I was hoping I can figure how can I get this to work. Thanks! More power!

Reply

Marco Schwartz
February 7, 2014 at 7:55 am

Hello,

There will be soon another post on the same tutorial with TCP, stay tuned !

Reply

Brett
February 8, 2014 at 1:57 am

Marco,

Marco,

Do you have any experience with putting the CC3000 into “shut down” mode to save power? I’ve read a couple articles by TI that show systems using the CC3000 running off a battery and coming out of a shutdown,where they claim the CC3000 will transition back into RX mode in less 700ms. I don’t think the Adafruit CC3000 library supports this functionality, but was wondering if you had any suggestions? Great tutorial!

Thanks!
Brett

Reply

Marco Schwartz
February 13, 2014 at 8:47 am

Hello Brett, thanks! No, I never tried out this functionality with the CC3000, but you’re right it would be great to have indeed. The only suggestion I would have is to message the Adafruit people directly to request this feature!

Reply

John

February 20, 2014 at 7:43 pm

Hi Marco! Great tutorial!

When I connect to a WLAN where I use “WLAN_SEC_WPA2” all works right, but when I need to use “WLAN_SEC_WEP” I can’t connect to a WLAN. I need to do something else?

Thanks!

Reply

Marco Schwartz

February 24, 2014 at 9:56 am

Hello John,

I don’t have much experience with WEP networks and the CC3000 chip, but I would really recommend switching to WPA2 WiFi network for more security.

Reply

Guri

February 23, 2014 at 10:31 pm

Hello there!

First of all thanks for your code. I am relatively new to Arduino, so please do forgive me if I am asking some very basic questions. So far I have managed to install a server (Apache 2.2.2) and PHP. The Wifi Shield CC3000 connects to my network and then connects to DHPC Servers and even sends the data every i get the following message
”

Connect to xxx.xxx.xxx.xx:80

Connected & Data sent

Closing connection

However, when I go to my Localhost/Relay.PHP

Nothing happens at all. I am not sure what am i doing wrong. Any advise/assistance will be greatly appreciated.

Best!

Guri

Reply

Marco Schwartz

February 24, 2014 at 9:58 am

Hello Guri,

Thanks for your comment! By going to the relay.php file, you should at least see the two buttons of the interface. Are they not doing anything ? However, I will soon change the tutorial for a much more efficient implementation using a web server running on the Arduino board, so stay tuned !

Reply

Guri

February 25, 2014 at 6:30 pm

Thanks for the reply Marco!

I see the two buttons – Yes and No, however nothing happens. I am wondering if it is how I have placed my files? Should I declare the complete path for the repository?

I will be waiting for the update!

thanks for sharing this, its great!!

Best,

Guri

Reply

Marco Schwartz
February 26, 2014 at 9:52 am

Hello Guri,

I just updated the tutorial with some new code using a web server running on the Arduino board, please try it out!

Reply

Guri
February 26, 2014 at 11:59 pm

Thanks Marco!

I am able to get to a point where I get "Listening for connections…" and then I press the "On" / "Off" button using the relay.html file (placed on my local server> C:\Apache 2.2.2\htdocs I have placed the update_state.php" in the same folder but nothing happens.

I know I might have stretched your generosity a bit too far, but I am new to this and I am very keen on creating this wifi relay!

Any assistance is greatly appreciated.

Thanks in anticipation!

Best,

Guri

Guri
March 3, 2014 at 7:36 pm

Marco,

Please could you help me with this error message

"GET http://localhost/update_state.php?relayState=Off 500 (Internal Server Error) "

Thanks!

Guri

Sam
March 21, 2014 at 7:12 pm

This is awesome! I would rather put the page on a remote server that is always on. Would something like Xively work for this or maybe serve the page directly on the cc3300?

Reply

Marco Schwartz
March 26, 2014 at 7:50 am

I would serve the page directly on the cc3000, but it might require some intensive coding!

Reply

Jorge Cunha
April 1, 2014 at 11:33 am

Hi, this example works on localhost? it's possible to control a relay in any network i'm conected?

Reply

Marco Schwartz

April 2, 2014 at 7:24 am

Hi Jorge, indeed this project is for localhost only. If you want to control a device from anywhere, I suggest having a look at our tutorials about the Arduino Yun !

Reply

Matt

April 17, 2014 at 11:58 pm

Marco,

I have a Mac OS-X running MAMP and a Adafruit CC3000 shield trying to run your sketch for the Wi-Fi relay on an Uno. I make the connection through my AP, however the connection between my Uno and the computer server does not seem to be stable. I get this output from your program on the serial interface;

Started AP/SSID scan
Connecting to GreenOak(my wi-fi network)-waiting to connect–connected to wi-fi network!
Requesting address from DHCP server…OK
Listening for connections……..
(then I go to my Mac and bring up the localhost:8080/relay.html on my browser and it displays the two buttons,
I push the "ON" button, and the following output occurs:)
HEAD / HTTP/1.1
Content-Type:text/html
On
(then everything stops on the serial monitor and the Uno has to be reset to do anything

else, and the relay never gets energized!).

I have all the support files (html,php,jscript,jquery,) all in the MAMP/ htdocs file on my Mac and can not get this to work any further than this point.

Can you give me some suggestions on how to resolve this problem, so I can make the relay turn on?

Any help would be vastly appreciated!!!!!

Thanks,

Matt

Reply

Marco Schwartz

April 23, 2014 at 8:07 am

Hello Matt,

That's a typical stability problem with the CC3000 chip & Arduino, unfortunately. I tried to standardise the communications with the CC3000 chip using a REST library. You can check that here: https://github.com/marcoschwartz/aREST. I'll be updating this article soon using this library.

Reply

Trackbacks & Pingbacks

. Internet of things to make – Wireless Relay Control with Arduino & the Adafruit CC3000 « adafruit industries blog

September 17, 2013 at 7:27 pm

[…] Open Home Automation | Wireless Relay Control with Arduino & the CC3000 WiFi chip – Open …. […]

Reply