CC3000 Wifi Shield

http://www.electronicaembajadores.com/Productos/Detalle/26/LCA1SO2/modulo-arduino-cc3000-wifi-shield-sd

Problema que deja de funcionar:

https://forum.arduino.cc/index.php?topic=293165.0

Nos bajamos la biblioteca (que no librería) https://github.com/adafruit/Adafruit_CC3000_Library

adafruit_cc3000_library-master.zip

Servidor web con IP Estática

La ip fija se asigna con la función setStaticIPAddress que hay que poner antes que conectar a la red wifi. He puesto un delay de 1000 después de asignar la ip estática porque si no fallaba.

```
/*************************************************
  Adafruit CC3000 Breakout/Shield Simple HTTP Server
  This is a simple implementation of a bare bones
  HTTP server that can respond to very simple requests.
  Note that this server is not meant to handle high
  load, concurrent connections, SSL, etc.  A 16mhz Arduino
  with 2K of memory can only handle so much complexity!
  This server example is best for very simple status messages
  or REST APIs.

  See the CC3000 tutorial on Adafruit's learning system
  for more information on setting up and using the
  CC3000:
    http://learn.adafruit.com/adafruit-cc3000-wifi
  Requirements:
  This sketch requires the Adafruit CC3000 library.  You can
  download the library from:
    https://github.com/adafruit/Adafruit_CC3000_Library
  For information on installing libraries in the Arduino IDE
  see this page:
    http://arduino.cc/en/Guide/Libraries
  Usage:
  Update the SSID and, if necessary, the CC3000 hardware pin
  information below, then run the sketch and check the
  output of the serial port.  After connecting to the
  wireless network successfully the sketch will output
  the IP address of the server and start listening for
  connections.  Once listening for connections, connect
  to the server IP from a web browser.  For example if your
```

```
  server is listening on IP 192.168.1.130 you would access
  http://192.168.1.130/ from your web browser.
  Created by Tony DiCola and adapted from HTTP server code created by Eric
Friedrich.
  This code was adapted from Adafruit CC3000 library example
  code which has the following license:
  Designed specifically to work with the Adafruit WiFi products:
  ----> https://www.adafruit.com/products/1469

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ****************************************************/
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "utility/debug.h"
#include "utility/socket.h"

// These are the interrupt and control pins
#define ADAFRUIT_CC3000_IRQ   3  // MUST be an interrupt pin!
// These can be any two pins
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
// Use hardware SPI for the remaining pins
// On an UNO, SCK = 13, MISO = 12, and MOSI = 11

Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
                                         SPI_CLOCK_DIVIDER); // you can
change this clock speed

#define WLAN_SSID       "Pitufina"   // cannot be longer than 32 characters!
#define WLAN_PASS       "**************"
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or
WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2

#define LISTEN_PORT           80      // What TCP port to listen on for
connections.
                                      // The HTTP protocol uses port 80 by
default.

#define MAX_ACTION            10      // Maximum length of the HTTP action
that can be parsed.

#define MAX_PATH              64      // Maximum length of the HTTP request
path that can be parsed.
                                      // There isn't much memory available
```

```
so keep this short!

#define BUFFER_SIZE           MAX_ACTION + MAX_PATH + 20  // Size of buffer
for incoming request data.
                                                          // Since only the
first line is parsed this
                                                          // needs to be as
large as the maximum action
                                                          // and path plus a
little for whitespace and
                                                          // HTTP version.

#define TIMEOUT_MS            500    // Amount of time in milliseconds to
wait for
                                     // an incoming request to finish.
Don't set this
                                     // too high or your server could be
slow to respond.

Adafruit_CC3000_Server httpServer(LISTEN_PORT);
uint8_t buffer[BUFFER_SIZE+1];
int bufindex = 0;
char action[MAX_ACTION+1];
char path[MAX_PATH+1];

void setup(void)
{
  Serial.begin(115200);
  Serial.println(F("Hello, CC3000!\n"));

  Serial.print("Free RAM: "); Serial.println(getFreeRam(), DEC);
  // Initialise the module
  Serial.println(F("\nInitializing..."));
  if (!cc3000.begin())
  {
    Serial.println(F("Couldn't begin()! Check your wiring?"));
    while(1);
  }
  uint32_t ipAddress = cc3000.IP2U32(192, 168, 1, 200);
  uint32_t netMask = cc3000.IP2U32(255, 255, 255, 0);
  uint32_t defaultGateway = cc3000.IP2U32(192, 168, 1, 1);
  uint32_t dns = cc3000.IP2U32(8, 8, 4, 4);
  if (!cc3000.setStaticIPAddress(ipAddress, netMask, defaultGateway, dns)) {
    Serial.println(F("Failed to set static IP!"));
    while(1);
  }
  delay(1000);

  Serial.print(F("\nAttempting to connect to ")); Serial.println(WLAN_SSID);
  if (!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
    Serial.println(F("Failed!"));
```

```
    while(1);
  }
  Serial.println(F("Connected!"));
  // *****************************************************
  // You can safely remove this to save some flash memory!
  // *****************************************************
  Serial.println(F("\r\nNOTE: This sketch may cause problems with other
sketches"));
  Serial.println(F("since the .disconnect() function is never called, so
the"));
  Serial.println(F("AP may refuse connection requests from the CC3000 until
a"));
  Serial.println(F("timeout period passes.  This is normal behaviour
since"));
  Serial.println(F("there isn't an obvious moment to disconnect with a
server.\r\n"));
  // Start listening for connections
  httpServer.begin();
  Serial.println(F("Listening for connections..."));
}

void loop(void)
{
  // Try to get a client which is connected.
  Adafruit_CC3000_ClientRef client = httpServer.available();
  if (client) {
    Serial.println(F("Client connected."));
    // Process this request until it completes or times out.
    // Note that this is explicitly limited to handling one request at a
time!

    // Clear the incoming data buffer and point to the beginning of it.
    bufindex = 0;
    memset(&buffer, 0, sizeof(buffer));
    // Clear action and path strings.
    memset(&action, 0, sizeof(action));
    memset(&path,   0, sizeof(path));

    // Set a timeout for reading all the incoming data.
    unsigned long endtime = millis() + TIMEOUT_MS;
    // Read all the incoming data until it can be parsed or the timeout
expires.
    bool parsed = false;
    while (!parsed && (millis() < endtime) && (bufindex < BUFFER_SIZE)) {
      if (client.available()) {
        buffer[bufindex++] = client.read();
      }
      parsed = parseRequest(buffer, bufindex, action, path);
    }

    // Handle the request if it was parsed.
```

```
    if (parsed) {
      Serial.println(F("Processing request"));
      Serial.print(F("Action: ")); Serial.println(action);
      Serial.print(F("Path: ")); Serial.println(path);
      // Check the action to see if it was a GET request.
      if (strcmp(action, "GET") == 0) {
        // Respond with the path that was accessed.
        // First send the success response code.
        client.fastrprintln(F("HTTP/1.1 200 OK"));
        // Then send a few headers to identify the type of data returned and
that
        // the connection will not be held open.
        client.fastrprintln(F("Content-Type: text/plain"));
        client.fastrprintln(F("Connection: close"));
        client.fastrprintln(F("Server: Adafruit CC3000"));
        // Send an empty line to signal start of body.
        client.fastrprintln(F(""));
        // Now send the response data.
        client.fastrprintln(F("Hello world!"));
        client.fastrprint(F("You accessed path: "));
client.fastrprintln(path);
      }
      else {
        // Unsupported action, respond with an HTTP 405 method not allowed
error.
        client.fastrprintln(F("HTTP/1.1 405 Method Not Allowed"));
        client.fastrprintln(F(""));
      }
    }

    // Wait a short period to make sure the response had time to send before
    // the connection is closed (the CC3000 sends data asyncronously).
    delay(100);

    // Close the connection when done.
    Serial.println(F("Client disconnected"));
    client.close();
  }
}

// Return true if the buffer contains an HTTP request.  Also returns the
request
// path and action strings if the request was parsed.  This does not attempt
to
// parse any HTTP headers because there really isn't enough memory to
process
// them all.
// HTTP request looks like:
//  [method] [path] [version] \r\n
//  Header_key_1: Header_value_1 \r\n
//  ...
```

```
//  Header_key_n: Header_value_n \r\n
//  \r\n
bool parseRequest(uint8_t* buf, int bufSize, char* action, char* path) {
  // Check if the request ends with \r\n to signal end of first line.
  if (bufSize < 2)
    return false;
  if (buf[bufSize-2] == '\r' && buf[bufSize-1] == '\n') {
    parseFirstLine((char*)buf, action, path);
    return true;
  }
  return false;
}

// Parse the action and path from the first line of an HTTP request.
void parseFirstLine(char* line, char* action, char* path) {
  // Parse first word up to whitespace as action.
  char* lineaction = strtok(line, " ");
  if (lineaction != NULL)
    strncpy(action, lineaction, MAX_ACTION);
  // Parse second word up to whitespace as path.
  char* linepath = strtok(NULL, " ");
  if (linepath != NULL)
    strncpy(path, linepath, MAX_PATH);
}

// Tries to read the IP address and other connection details
bool displayConnectionDetails(void)
{
  uint32_t ipAddress, netMask, defaultGateway, dns;
    Serial.print(F("\nIP Addr: ")); cc3000.printIPdotsRev(ipAddress);
    Serial.print(F("\nNetmask: ")); cc3000.printIPdotsRev(netMask);
    Serial.print(F("\nGateway: ")); cc3000.printIPdotsRev(defaultGateway);
    Serial.print(F("\nDNS Srv: ")); cc3000.printIPdotsRev(dns);
    Serial.println();
    return true;
}
```