DOCUMENTATION: CODE NOTES

By: Lauren Gaber

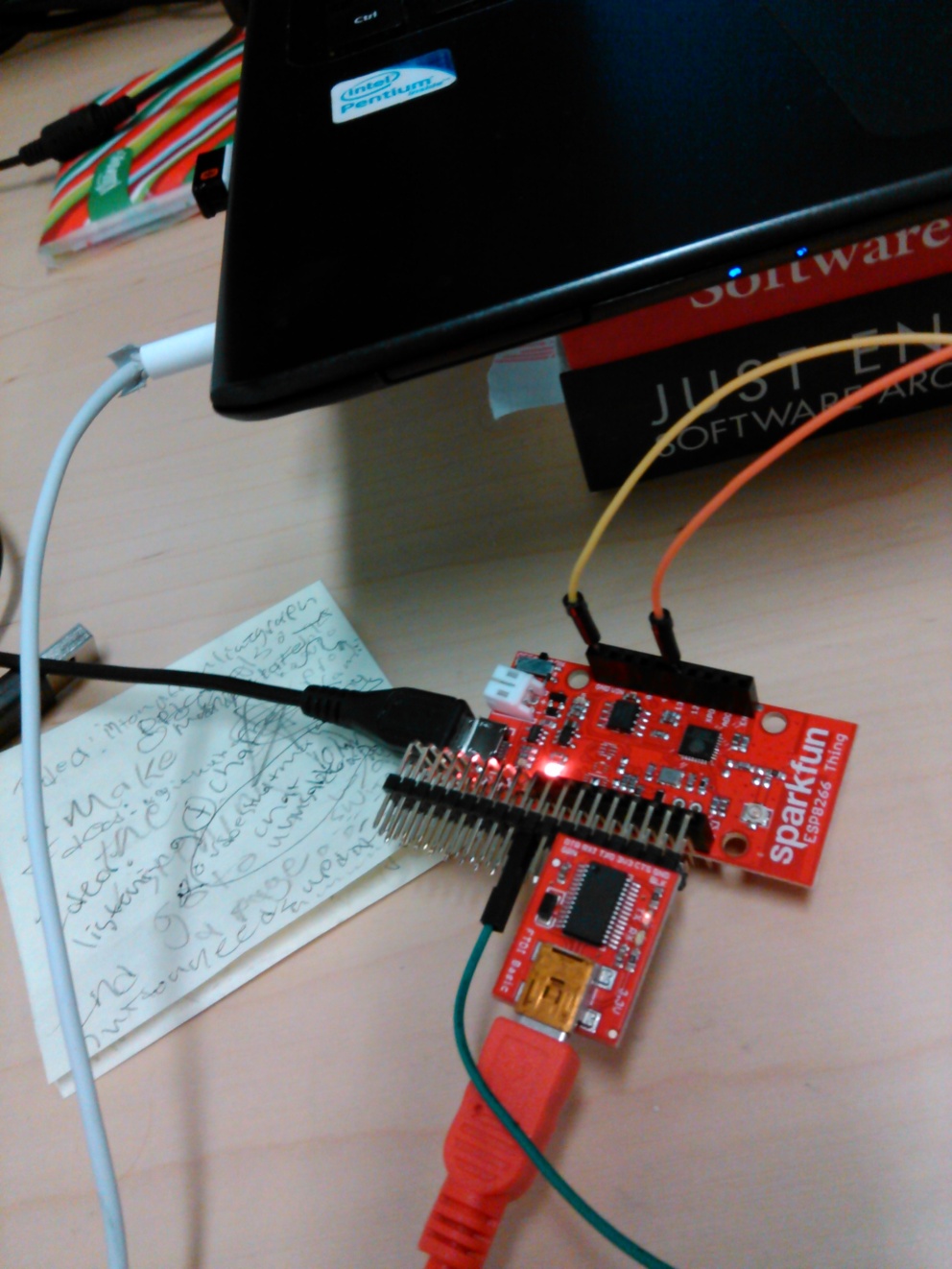
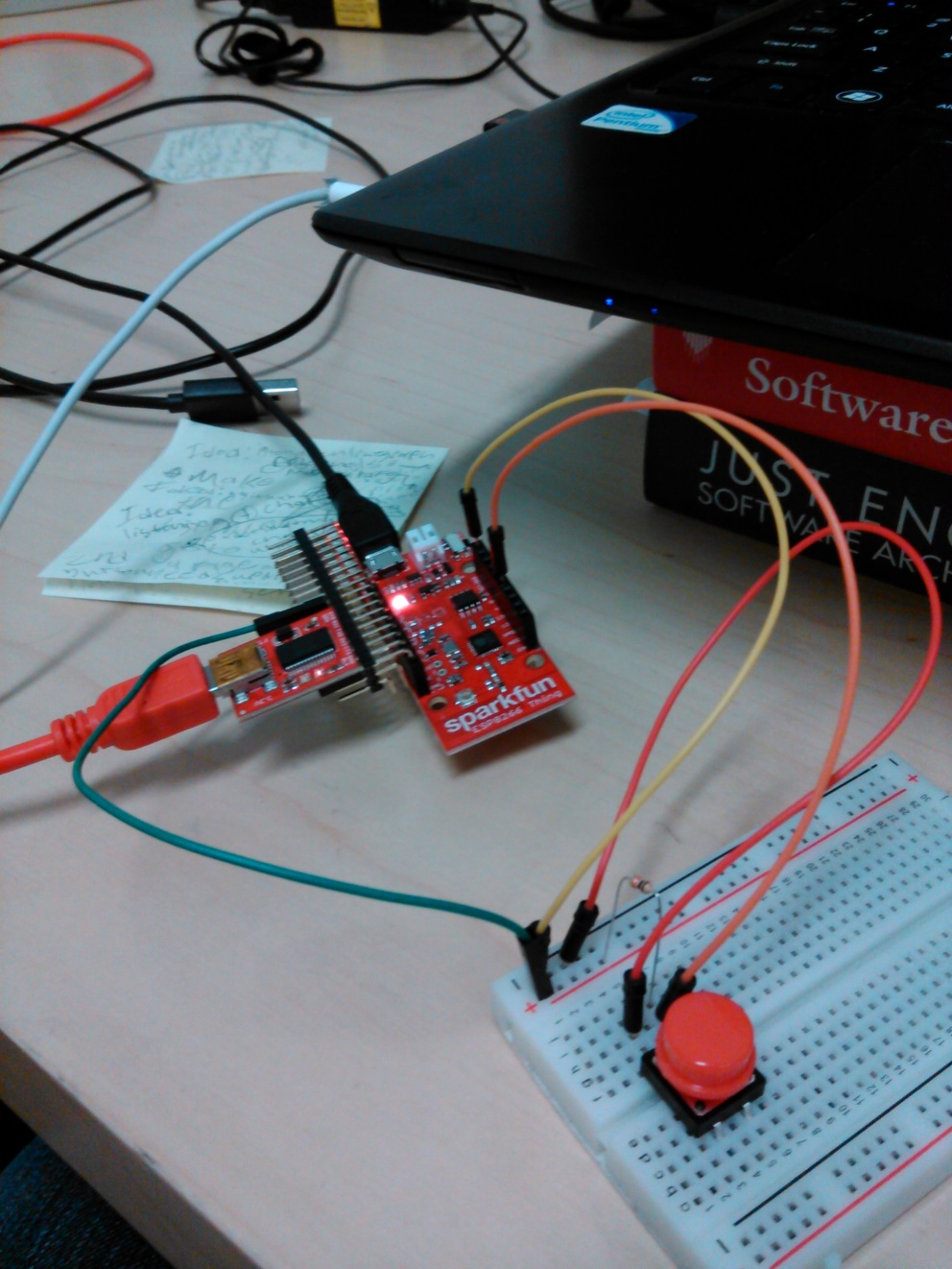
(this documentation should be included on the usb drive’s file for this project)

Notes about C# Xamarin forms TTSTest2 project code:

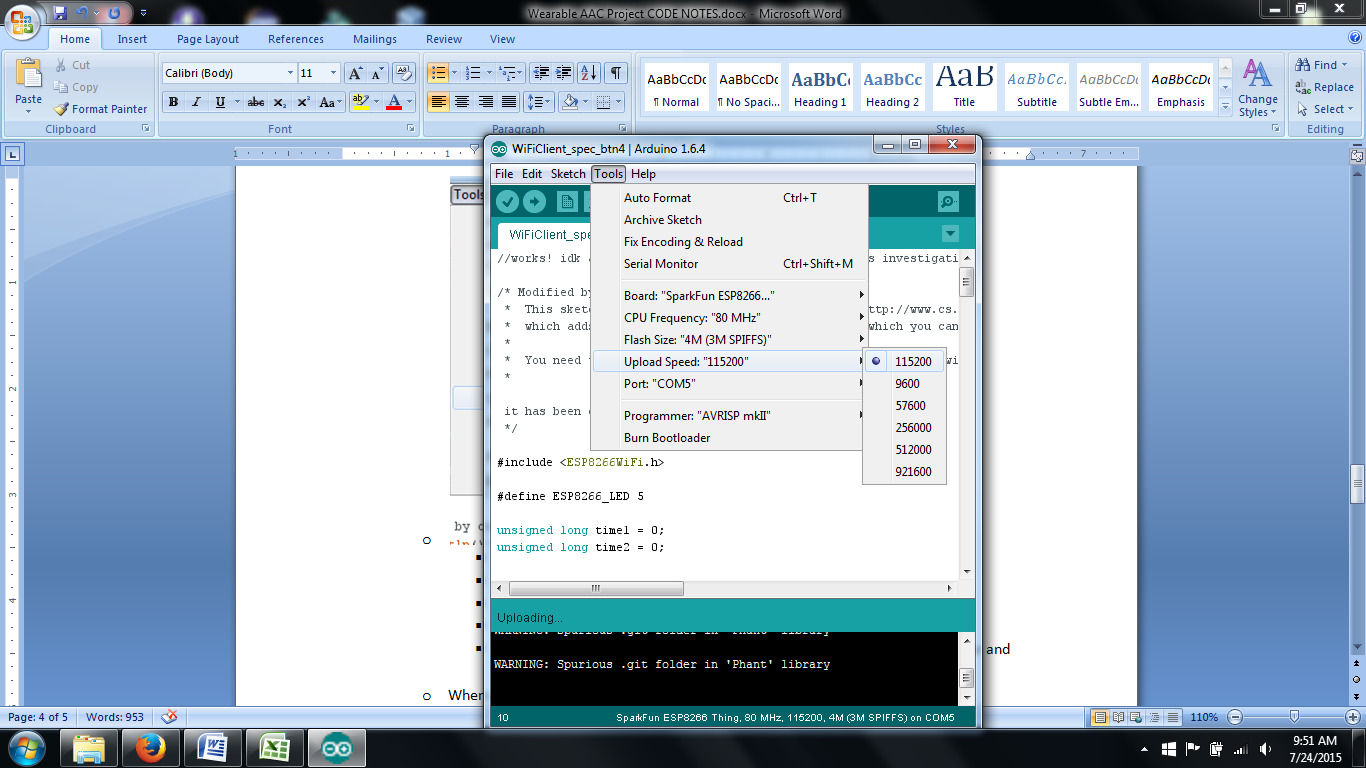
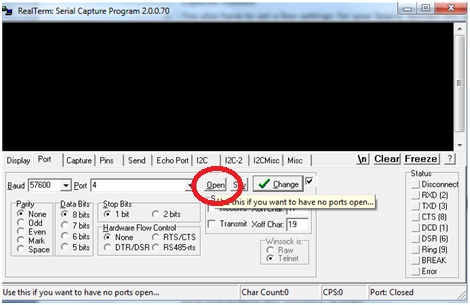
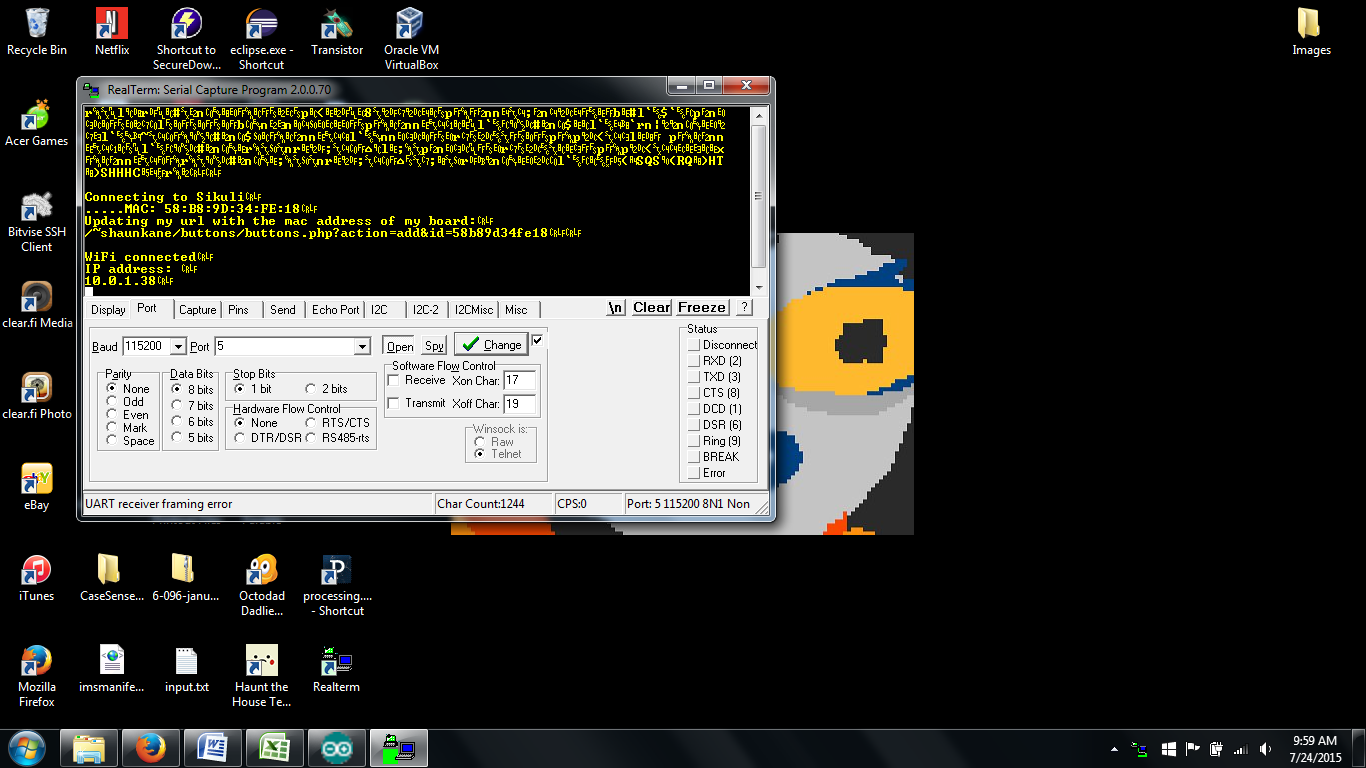
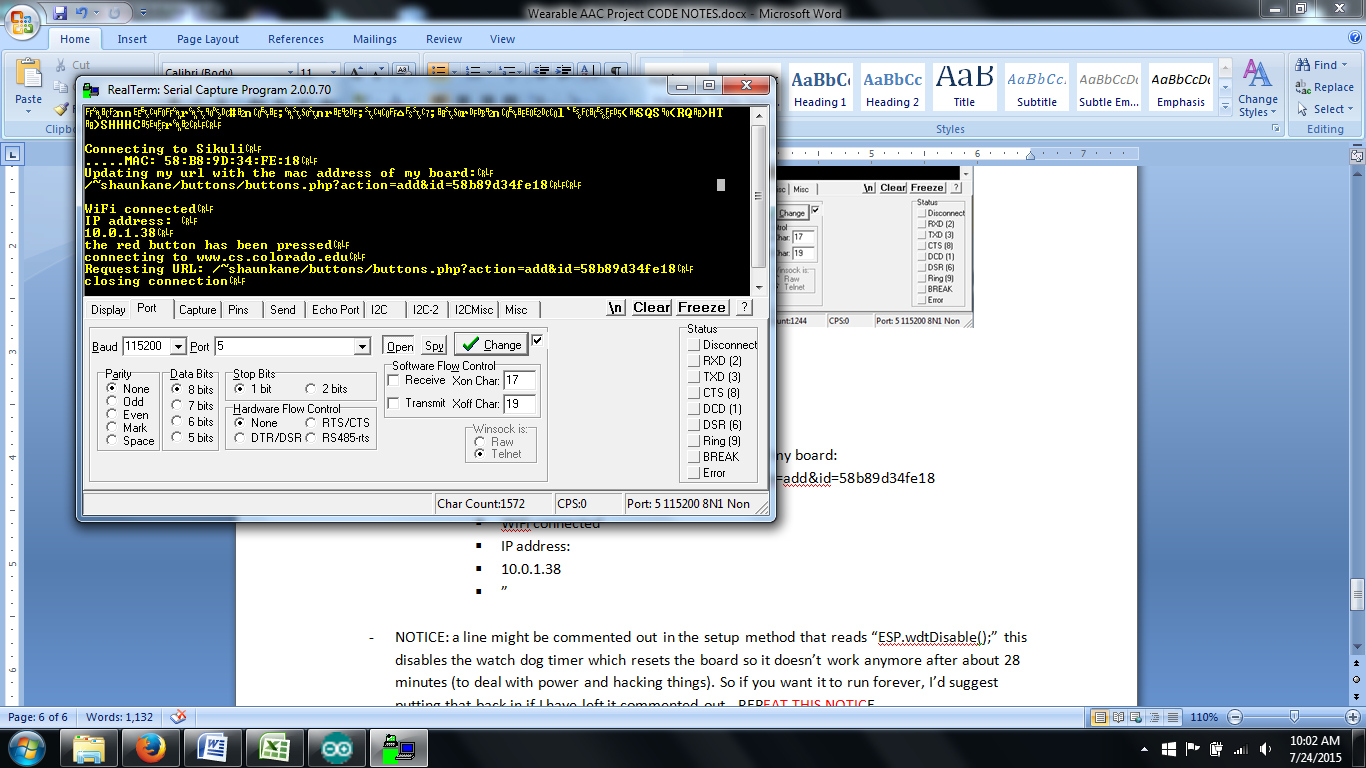
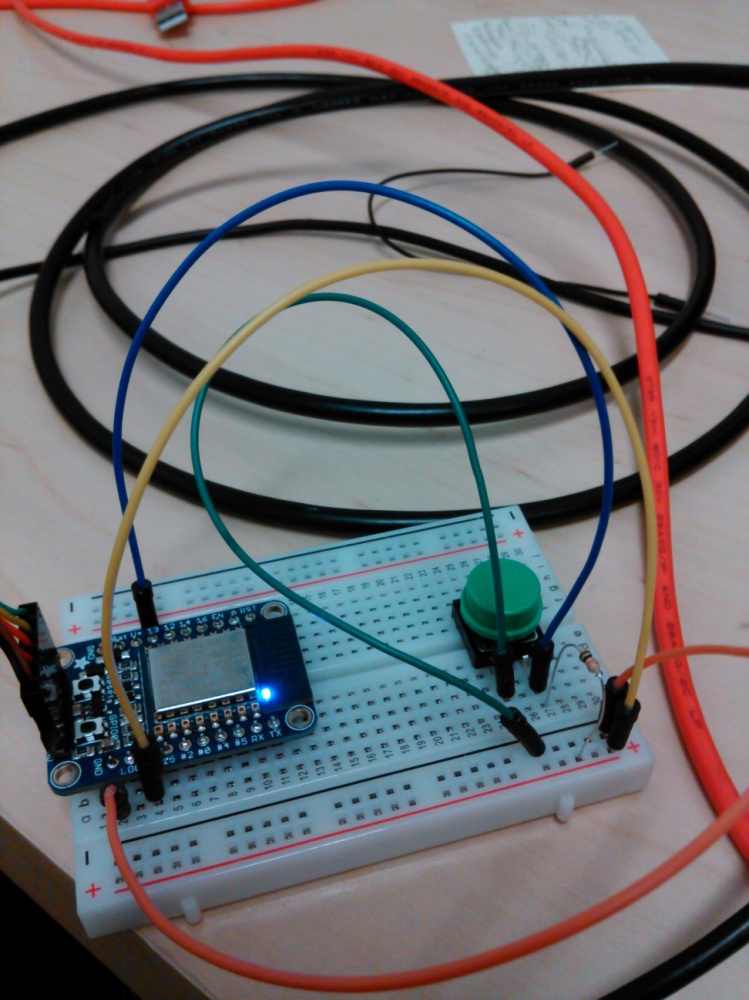
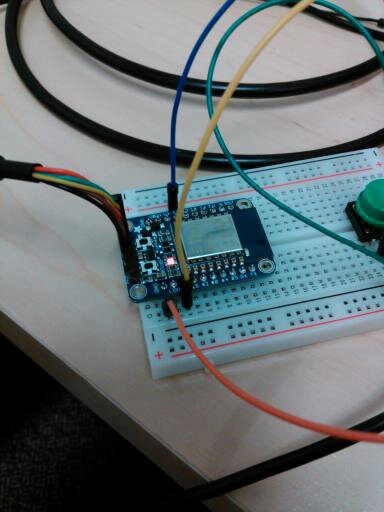
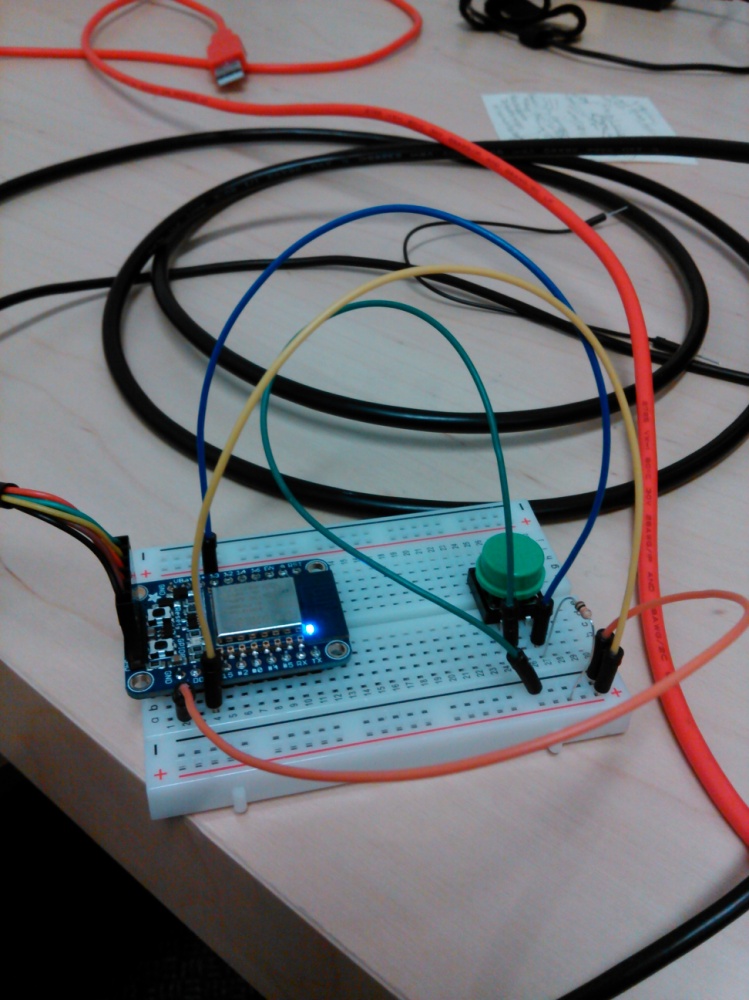
* Programmed in Microsoft Visual studio 2012 (not the version 4 update if you’ve heard of that (it probably works fine but I didn’t want to risk it so I didn’t update in the middle of coding the project))
* The first time tts is played since the app is initially opened; it takes a bit longer than all the other times. Also, it should be noted that the speed and pitch values are not applied until the second play usually.
* There are descriptions and warnings at the top of all the class files which you should pay close attention to.
* I did not test any iOS platform specific code because that requires a mac.
* I tested the code using my personal phone, an android zte N800.
* You need to add a few libraries and permissions, by doing several things:
  + Add using statements to parts of the code
  + Going to the solution explorer window and right-clicking TTSTest2 or TTSTest2.droid or TTSTest2.iOS and going to add reference or manage NuGet packages and searching for and installing or checking the checkbox for a library you need. I don’t remember all the libraries I needed to do this for, but some that come to mind are System.Net (pretty sure this one was done through NuGet), System.Runtime.Serialization (was through add reference), etc. etc. You should try and add the libraries for all the project types.
  + Make sure Build > Confuguration Manager has all the deploy check boxes available checked off. That can cause an error.
  + Make sure next to the play button the name of your devie shows up, indicating that is the device (or maybe an emulator? But they are horribly slow, at least for android. I was keeping track and it takes like 5-15 minutes to start it up each time and it is really laggy).
  + The android manifest needs some additional permissions. Do this by highlighting TTSTest2.droid in the solutions manager, then going to project > TTSTest2.droid properties, then got to the android manifest tab and scroll down to required permissions and check off RECORD\_AUDIO and save all files.

Notes about Arduino code:

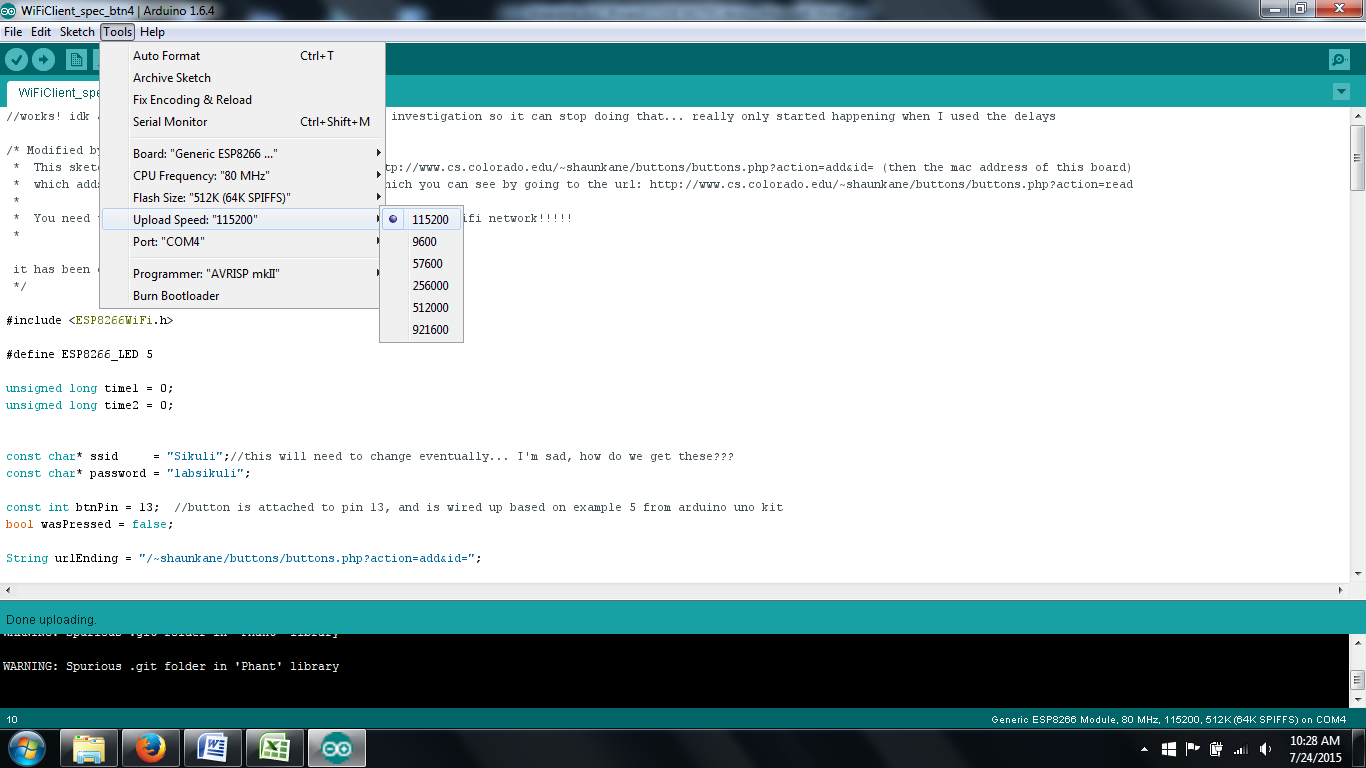
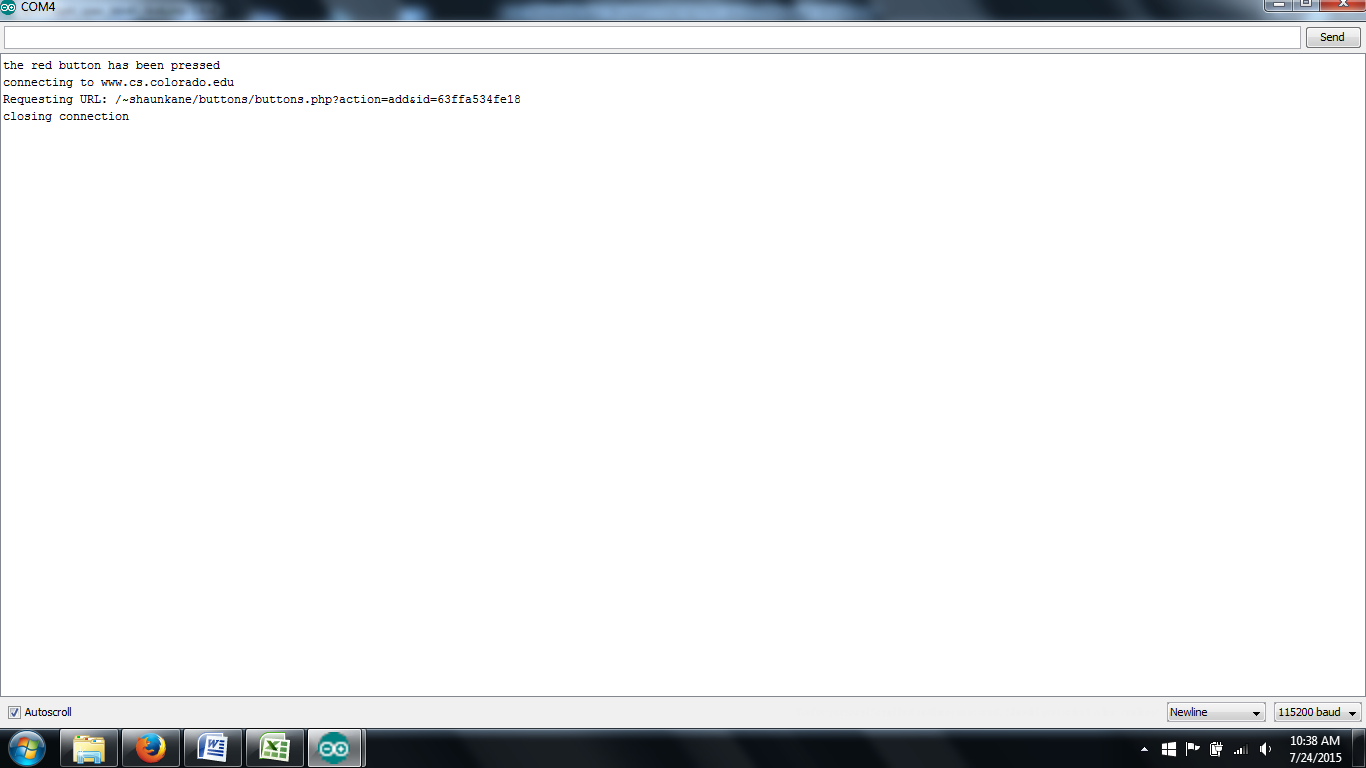
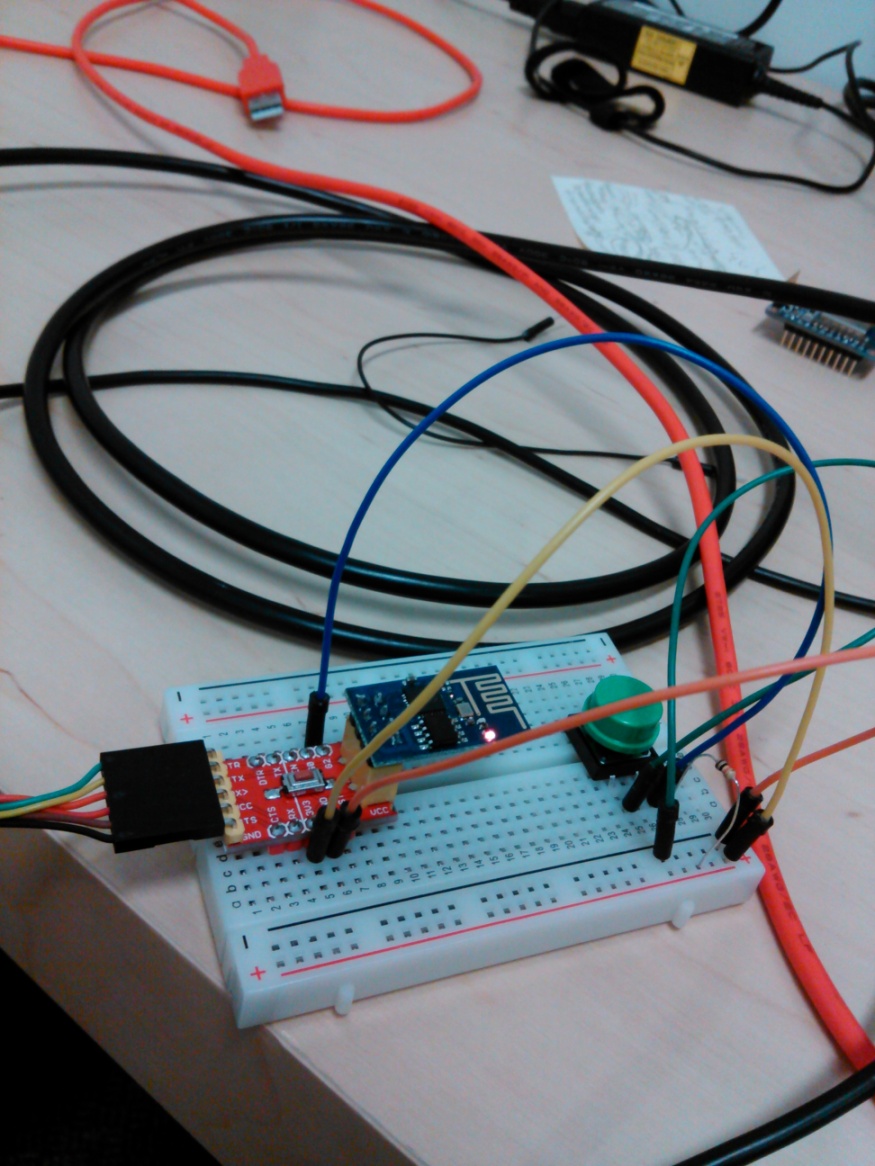
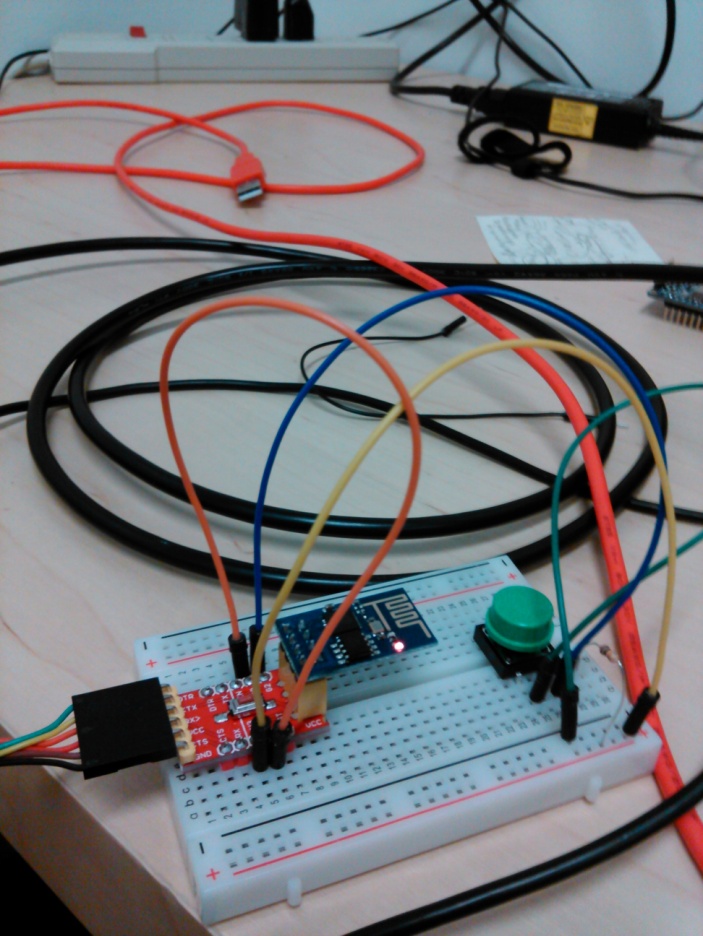
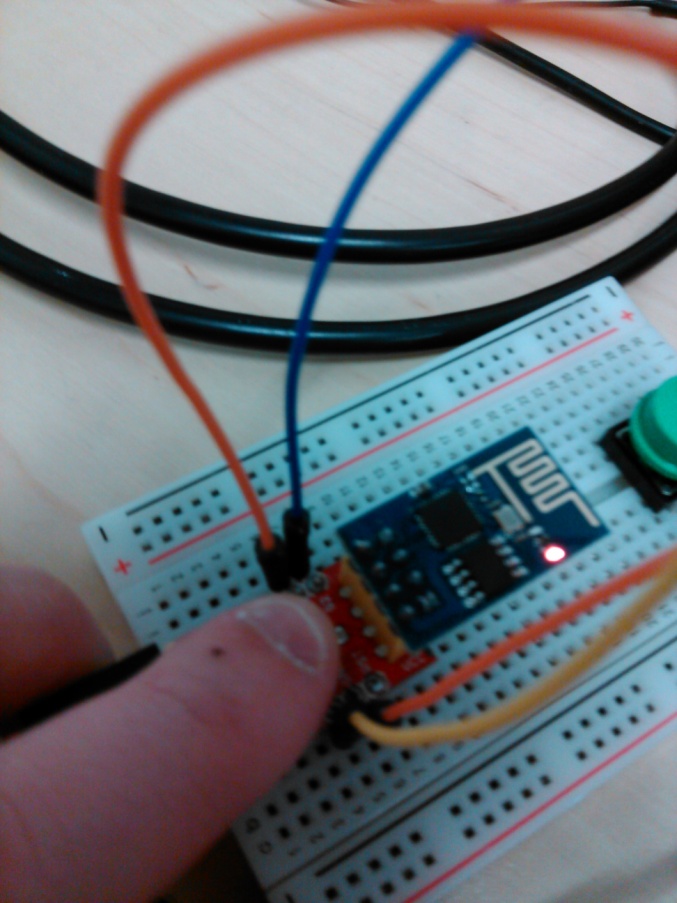
* **WiFiClient\_spec\_btn4.ino** is for Sparkfun ESP8266 Thing and Ada Fruit ESP8266 huzzah boards.
* **How to load code onto the Sparkfun ESP8266 Thing board (setup pictured below):**



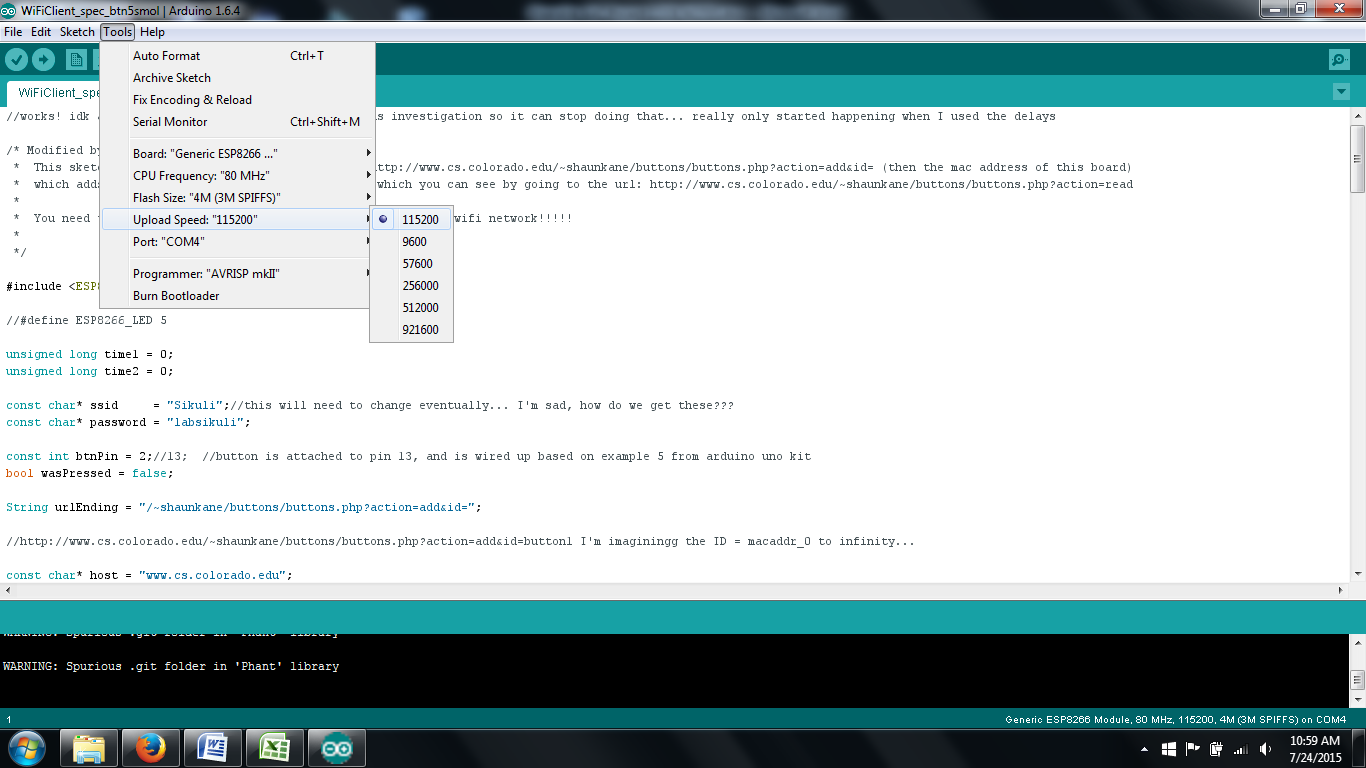
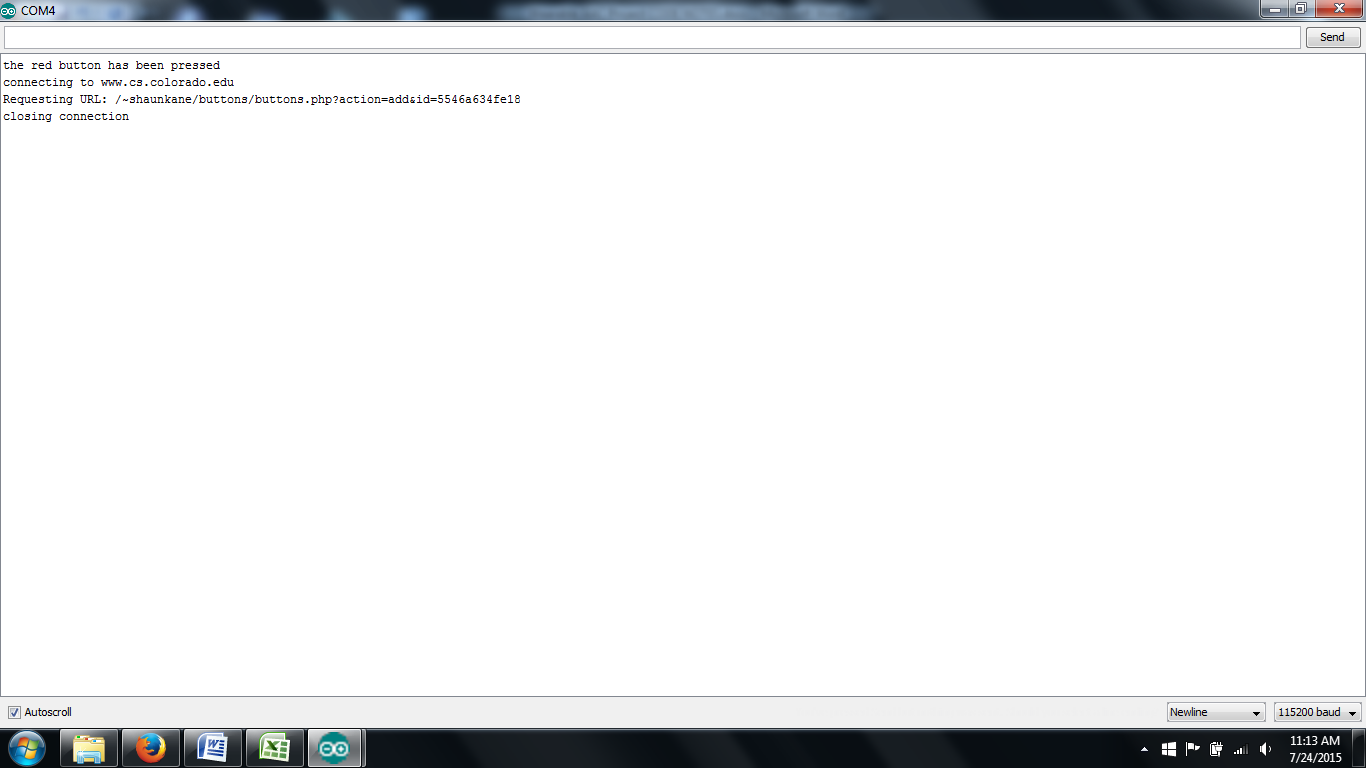
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **How to assemble program and button testing setup for Sparkfun ESP8266 Thing** | | | | |
| **Connect** | **To** | **Using** | **And To** | **Using** |
| FTDI Basic 3.3V (https://www.sparkfun.com/products/9873) | DTR, TXO, RXI, 3V3, NC, GND pins on board | 3X40 breakaway right angle headers (https://www.sparkfun.com/products/10095) |  |  |
| Breadboard + | 3V3 pin on board | A male- male wire if there are no right angle headers in the way, otherwise use a male - female connecting wire with the headers |  |  |
| laptop | board | USB Mini-B 6' cable (https://www.sparkfun.com/products/11301) |  |  |
| power supply (an outlet or your laptop) | board | USB Micro-B cable (https://www.sparkfun.com/products/10215) for laptop or that and a connector thata takes a usb and connects the usb to a power outlet. |  |  |
| Breadboard - | Other GND pin on board | a male-male wire |  |  |
| One button metal leg tab | Pin 13 on board | a male-male wire | Breadboard + | 10K resistor (brown stripe on button side) |
| Other button metal leg tab | Breadboard - | a male-male wire |  |  |

* + When you open up the Arduino file WiFiClient\_spec\_btn4.ino to be edited in the arduino code editor, you have to import the addon (<https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/all#installing-the-esp8266-arduino-addon> - also see/download the one on github, that might be the one that actually worked: <https://github.com/esp8266/Arduino> “Enter **http://arduino.esp8266.com/stable/package\_esp8266com\_index.json** into Additional Board Manager URLs field”). This library is the one where you will select all your board types from, for all 3 of these. This board’s board name is listed in the board manager drop down as Sparkfun esp8266 Thing, and the other two are listed as a General Esp8266 module. There should be four board types in the list section you are choosing from.
  + You also have to set a few settings for your board, besides this addon that lets you choose the right board in the code. Hey are listed under tools, and should be set as follows:
  + 
    - Board: “Sparkfun ESP8266 Thing”
    - CPU Frequency: “80 MHz”
    - Flash Size: “4M (3M SPIFFS)”
    - Upload Speed: “115200”
    - Port: “COM 5” (or some other number, it depends on your computer and the cable)
  + When you try to upload your code, it might fail and give you the error
    - “warning: espcomm\_sync failed error: espcomm\_open failed”
    - This means you need to usually do one or more of these things:
      * Respecify the board and settings
      * Make sure you have the right cable plugged in
      * Make sure the board’s power is on
      * unplug and replug the cable into your lap top
      * restart the Arduino programmer
      * Realterm is open and you need to close it because I’m pretty sure you can’t upload code while it’s open.
  + If you have uploaded successfully, your output tab should read “Done uploading.” And the compiling output message box at the bottom should read something like this:
    - Sketch uses 237,980 bytes (45%) of program storage space. Maximum is 524,288 bytes.
    - Uploading 33168 bytes from C:\Users\Lauren\AppData\Local\Temp\build3874327142437250341.tmp/WiFiClient\_spec\_btn4.cpp\_00000.bin to flash at 0x00000000
    - .................................
    - Uploading 204856 bytes from C:\Users\Lauren\AppData\Local\Temp\build3874327142437250341.tmp/WiFiClient\_spec\_btn4.cpp\_10000.bin to flash at 0x00010000
    - .........................................................................................................................................................................................................
  + The serial Monitor built in to the Arduino programmer will not work for this board, though it does work for the other boards. To view the serial monitor output for this board, download Realterm (<http://realterm.sourceforge.net/> ). After you upload your code to the board, open real term and perform this sequence of tasks:
    - Go to the Port tab and unselect open.
    - 
    - Then go to the Pins tab and Clear the DTR pin.
    - Return to the Port tab, set the Baud to 115200, choose a port that matches the number of the COM you selected in your arduino code, and open the Port again. If a bunch of garbage is spit out with no real words showing up, your board is probably not working right. What you shoud see is something like this:
    - 
    - Aparagraph of garbage, then:
    - “¨)SHHHC…äÿr‚
    - Connecting to Sikuli
    - .....MAC: 58:B8:9D:34:FE:18
    - Updating my url with the mac address of my board:
    - /~shaunkane/buttons/buttons.php?action=add&id=58b89d34fe18
    - WiFi connected
    - IP address:
    - 10.0.1.38
    - ”
  + And when you press the button on your board, it should then read/print:
    - 
    - “the red button has been pressed
    - connecting to www.cs.colorado.edu
    - Requesting URL: /~shaunkane/buttons/buttons.php?action=add&id=58b89d34fe18
    - closing connection
    - ”
    - And lastly, if you refresh your database found at <http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read> , you should see a new entry added at the top with the same mac address:
    - (EX: [{"rowid":"16","buttonID":"58b89d34fe18","unixTime":"1437732104","datetime":"2015-07-24 10:01:44"}, …..)
* NOTICE: a line might be commented out in the setup method that reads “ESP.wdtDisable();” this disables the watch dog timer which resets the board so it doesn’t work anymore after about 28 minutes (to deal with power and hacking things). So if you want it to run forever, I’d suggest putting that back in if I have left it commented out.
* **How to load code onto the Ada Fruit Huzzah ESP8266 Breakout board (setup pictured below):**
  + ****
  + ****
  + **(last picture: the ‘dim’ red led indicating ready to load code mode)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **How to assemble program and button testing setup for Ada Fruit ESP8266 Huzzah Breakout Board** | | | | |
| **Connect** | **To** | **Using** | **And To** | **Using** |
| laptop | Huzzah | FTDI Cable 5V VCC-3.3V I/O (https://www.sparkfun.com/products/9717) |  |  |
| Breadboard + | 3V pin on board | a male-male wire |  |  |
| Breadboard - | Other GND pin on board | a male-male wire |  |  |
| One button metal leg tab | Pin 13 on board | a male-male wire | Breadboard + | 10K resistor (brown stripe on button side) |
| Other button metal leg tab | Breadboard - | a male-male wire |  |  |

* + To program this board in Arduino, you have to download the addon/github addon that was mentioned in the set up for the Sparkfun ESP8266 Thing board. Downloading the Huzzah’s suggested material wouldn’t hurt, and it is what I did before I started testing it (see <https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide> or <https://learn.adafruit.com/downloads/pdf/adafruit-huzzah-esp8266-breakout.pdf> ). This board’s board name is listed in the board manager drop down as a General ESP8266 Module (from the same group that lists the spark fun thing board). There should be four board types in the list section.
  + You also have to set a few settings for your board like before. They are listed under tools, and should be set as follows:
    - 
    - Board: “Generic ESP8266 Module”
    - CPU Frequency: “80 MHz”
    - Flash Size: “4M (3M SPIFFS)”
    - Upload Speed: “115200”
    - Port: “COM 4” (or some other number, it depends on your computer and the cable)
  + BEFORE YOU UPLOAD CODE: you need to physically put the board in bootloader mode, or the mode where it can upload a program. When it is in bootloader mode, the red button will be lit dimly as seen in the 3rd picture of this board. To put it in boot loader mode, you have to press down the reset button, and while that is happening you also have to hold down the gpio0 button; it is kind of best done by rolling your thumb across the two buttons, first the reset, then the reset and gpio0, then just gpio0, then neither of them. It can take a few tries, definitely.
  + Once that’s done, upload your code! The success message and fail message (and the fail message solutions) are pretty much the same as how they were for the Sparkfun Thing board.
  + When it’s done uploading, you can open the serial monitor by going to tools > serial monitor. Once again you will see the kind of information Realterm printed out for you last time:
    - 
    - “the red button has been pressed
    - connecting to www.cs.colorado.edu
    - Requesting URL: /~shaunkane/buttons/buttons.php?action=add&id=63ffa534fe18
    - closing connection”
    - And lastly, if you refresh your database found at <http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read> , you should see a new entry added at the top with the same mac address:
    - (EX: [{"rowid":"17","buttonID":"63ffa534fe18","unixTime":"1437734302","datetime":"2015-07-24 10:38:22"}, …)
* NOTICE: a line might be commented out in the setup method that reads “ESP.wdtDisable();” this disables the watch dog timer which resets the board so it doesn’t work anymore after about 28 minutes (to deal with power and hacking things). So if you want it to run forever, I’d suggest putting that back in if I have left it commented out.
* **WiFiClient\_spec\_btn5smol.ino** is for the original smaller blue ESP8266 boards. (specify how to reset/put in programming mode)
* **How to load code onto the original ESP8266 WiFi Module board (setup pictured below):**
  + ****
  + ****
  + ****
  + **(last 2 pictures: parts of the set up task for putting the module into ready to load code mode)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **How to assemble program and button testing setup for Original ESP8266 Board** | | | | |
| **Connect** | **To** | **Using** | **And To** | **Using** |
| ESP8266 (Original) WiFi Module (https://www.sparkfun.com/products/13678) | ESP8266 FTDI and breadboard adapter with 3.3V reg. (https://www.tindie.com/products/FemtoCow/esp8266-ftdi-  and-breadboard-adapter-with-33v-reg/) | (NOTE: ADAPTER WILL BE REFERRED TO AS THE MAIN BOARD, EVEN THOUGH THE ACTUAL BOARD IS THE WIFI MODULE |  |  |
| laptop | ESP8266 FTDI and breadboard adapter with 3.3V reg. | FTDI Cable 5V VCC-3.3V I/O (https://www.sparkfun.com/products/9717) |  |  |
| Breadboard + | 3V3 pin on board | a male-male wire |  |  |
| Breadboard - | Other GND pin on board | a male-male wire |  |  |
| One button metal leg tab | Pin G2 on board | a male-male wire | Breadboard + | 10K resistor (brown stripe button side) |
| Other button metal leg tab | Breadboard - | a male-male wire |  |  |

* + To program this board in Arduino, you have to download the addon/github addon that was mentioned in the set up for the Sparkfun ESP8266 Thing board. This board’s board name is listed in the board manager drop down as a General ESP8266 Module (from the same group that lists the spark fun thing board). There should be four board types in the list section.
  + You also have to set a few settings for your board like before. They are listed under tools, and should be set as follows:
    - 
    - Board: “Generic ESP8266 Module”
    - CPU Frequency: “80 MHz”
    - Flash Size: “4M (3M SPIFFS)”
    - Upload Speed: “115200”
    - Port: “COM 4” (or some other number, it depends on your computer and the cable)
  + BEFORE YOU UPLOAD CODE: you need to physically put the board in bootloader mode, or the mode where it can upload a program. When it is set in bootloader mode, you should see a little blue light flash next to the already on red light, and then the red light will continue to be on. To put it in boot loader mode, you have to take the wire plugged into GND (Ground) and Breadboard -, unplug it from Breadboard -, plug it into G0 on the board, press the reset button (the big red button in the middle of the board, and see the small blue flash), then remove it from G0 and return it to Breadboard -.
  + Once that’s done, upload your code! The success message and fail message (and the fail message solutions) are pretty much the same as how they were for the Sparkfun Thing and Ada Fruit Huzzah board.
  + When it’s done uploading, you can open the serial monitor by going to tools > serial monitor. Once again you will see the kind of information it printed out for you with the Huzzah:
    - 
    - “the red button has been pressed
    - connecting to www.cs.colorado.edu
    - Requesting URL: /~shaunkane/buttons/buttons.php?action=add&id=5546a634fe18
    - closing connection”
    - And lastly, if you refresh your database found at <http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read> , you should see a new entry added at the top with the same mac address:
    - (EX: [{"rowid":"21","buttonID":"5546a634fe18","unixTime":"1437736295","datetime":"2015-07-24 11:11:35"}, …)
* NOTICE: a line might be commented out in the setup method that reads “ESP.wdtDisable();” this disables the watch dog timer which resets the board so it doesn’t work anymore after about 28 minutes (to deal with power and hacking things). So if you want it to run forever, I’d suggest putting that back in if I have left it commented out.

**PHP Code** for website (Shaun wrote it) (file name is buttons.php):

--------------------------------------------------------------------------------------------------------------------------------

<?php

// show errors: from http://stackoverflow.com/questions/1053424/how-do-i-get-php-errors-to-display

ini\_set('display\_errors',1);

ini\_set('display\_startup\_errors',1);

error\_reporting(-1);

// set the local timezone: http://stackoverflow.com/questions/470617/get-current-date-and-time-in-php

date\_default\_timezone\_set('America/Denver');

$instructions = "this code keeps track of all of our button presses

to add a button press, call the script with the action parameter

set to 'add' and the id parameter set to the button id. this

will store an entry in an sqlite database with the id and the

time of the button press

<br/><br/>

example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=add&id=button1

<br/><br/>

to read the data, call the script with the action parameter

set to 'read'. this will return all of the button presses

in json format.

<br/><br/>

example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read

<br/><br/>

you may specify a date using the since= parameter, including

a https://en.wikipedia.org/wiki/Unix\_time . This will

return only events after the specified time

<br/><br/>

example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read&since=1436172703

<br/><br/>

finally, you can set action=clear to clear out the database

<br/><br/>

example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=clear

<br/><br/>

if no action parameter is set, the script will print out

these usage instructions";

if (!isset($\_GET["action"])) {

echo $instructions;

}

else {

if ($\_GET["action"] == "read") {

// read our data: http://stackoverflow.com/questions/16622700/php-json-array-from-sqlite3

$db = 'sqlite:buttons.sqlite';

$dbConnect = new PDO($db) or die("cannot connect to database! boo!");

$query = "select rowid, buttonID, strftime('%s',time) as 'unixTime', datetime(time) as 'datetime' from buttonPresses";

// if we have a "since" parameter set, add that to our query

if (isset($\_GET["since"])) {

$query .= " where time > datetime(" . $\_GET["since"] . ", 'unixepoch')";

}

$query .= " order by rowid desc";

//echo $query;

$result = $dbConnect->query($query);

$results = array();

$result->setFetchMode(PDO::FETCH\_ASSOC);

while ($data = $result->fetch()) {

array\_push($results, $data);

}

echo json\_encode($results);

} elseif ($\_GET["action"] == "add" && isset($\_GET["id"])) {

// write

$db = 'sqlite:buttons.sqlite';

$dbConnect = new PDO($db) or die("cannot connect to database! boo!");

$query = "insert into buttonPresses(buttonID, time) values('" . $\_GET["id"] . "', datetime('now', 'localtime'))";

echo $query;

$dbConnect->exec($query);

echo "added successfully";

} elseif ($\_GET["action"] == "clear") {

$db = 'sqlite:buttons.sqlite';

$dbConnect = new PDO($db) or die("cannot connect to database! boo!");

$query = "delete from buttonPresses";

$dbConnect->exec($query);

echo "cleared out table!";

// write

} else {

echo "invalid syntax";

}

}

?>

-----------------------------------------------------------------------------------------------------------

EXPLANATION FOUND AT: <http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php>

* This code keeps track of all of our button presses to add a button press, call the script with the action parameter set to 'add' and the id parameter set to the button id. this will store an entry in an sqlite database with the id and the time of the button press   
    
  example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=add&id=button1   
    
  to read the data, call the script with the action parameter set to 'read'. this will return all of the button presses in json format.   
    
  Example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read   
    
  you may specify a date using the since= parameter, including a https://en.wikipedia.org/wiki/Unix\_time. This will return only events after the specified time   
    
  example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=read&since=1436172703   
    
  finally, you can set action=clear to clear out the database   
    
  example: http://www.cs.colorado.edu/~shaunkane/buttons/buttons.php?action=clear   
    
  if no action parameter is set, the script will print out these usage instructions