Build Instructions for the GK-WiFi Kit v1.4

Rev. 1.2

Getting Started

Congratulations! This is a fun kit. It's not only a great way to get you Geiger readings on the web, but it's a good development board for having fun with the ESP8266. I hope you enjoy building it as well as using it.
Please Note: This kit requires that you be prepared to program the ESP8266 with the Arduino IDE. Good instructions for this are provided here, but if you are not up for this you can return the unassembled kit for a refund.

General tips:

- "***Sometimes just a few hours of trial and error debugging can save minutes of reading instructions.***" Even if you're experienced, you run the risk of wishing you had considered something beforehand.
- Use the **Build Sequence** (below). It describes the part orientation and options as you go.
- Use the **Assembly Images** and schematic (below) to help you.
- Missing parts / extra parts – You are more likely to get an extra part, but if something is missing, let me know.
- Take your time! It usually takes a few hours to build this kit. Solder the right part, the right way, the first time. Parts are hard to unsolder.

Soldering:

Solder the joint so that you have a nice round dot. Do not use too much solder, and add enough heat for a good flow. The holes are plated through, so don't worry about getting solder up to the top of the board.

A "3rd hand" with a piece of solder in one of the alligator clips can be handy when tacking in headers, etc.

Careful with flux pens! Many will leave a residue that is slightly conductive. External fluxes can cause wacky problems. Simply use rosin core solder. Never use a flux *paste*.

I do not recommend using lead free solder for the kit. In my experience, it makes parts even harder to unsolder, and more heat is needed which may damage the pads. I will not do any board repair if lead free solder was used.

Step 1 - Building the Kit:

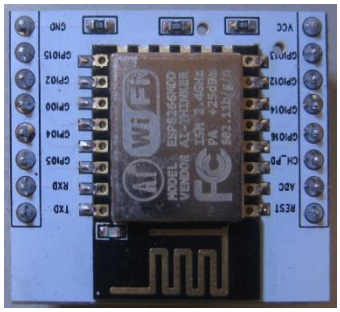
Build the Carrier Board

Here you will solder the ESP8266 to the little carrier board. Be sure the ESP8266 is facing the right way (see pic). The technique I use is as follows.

- Center the ESP8266 on the pads, hold it down with your finger and tack in a few pads. There is usually a bit of solder already on the pads, but put a some on the iron's tip if it helps. Check that everything aligns.

- Now solder all the pads. Place the iron on the pad away from the half hole ("castellated hole"). Add a bit of solder and slide the iron until it touches the half hole. Heat until the solder flows up the hole. A flux pen may come in handy for this, but it's usually not necessary.

- After all pins are soldered, clean the top of the board with alcohol if needed - especially if a flux pen was used.



Finally, install the two 8 pin male headers. It's a good idea to the put the headers in a breadboard first. Then place the carrier board on top, and solder. This ensures the headers will be square to the board.

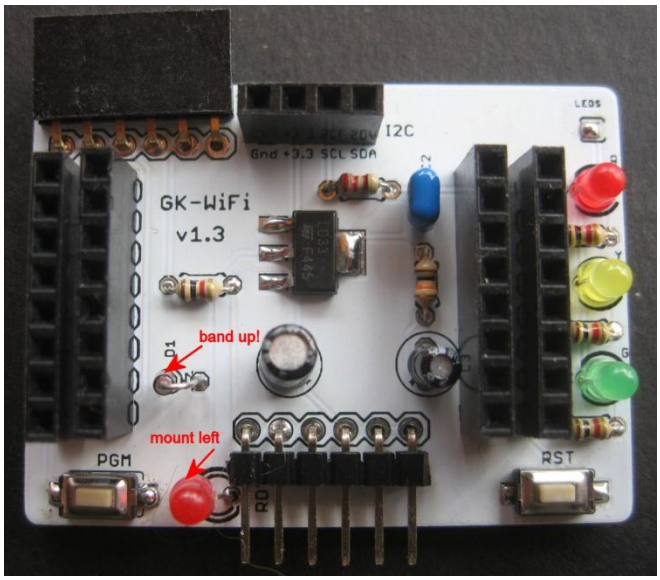
Build the Main Board

Use the table below as your guide to building the kit. Its approach is to build the board by height – starting with the shortest components. It's easier to work on a board that lays flat and holds the parts in place when you flip it over to solder. While working, refer to images to double check orientations of parts, etc.

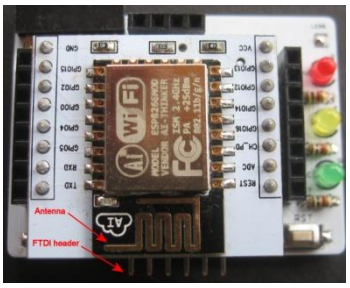
Build Sequence and Parts List					
Ref #	**Qty**	**Value**	**Description**	**Notes** **polarized? ->**	**Y N**
PCB	1	v1.4	1.9" x 1.4" (~4.83 x 3.55 cm)	**Orientation:** I2C header is at **top** of board.	--
R1,R4,R5,R6	4	1kΩ	BN,BK, RD	Be sure last band is red – not orange	N
R2	1	10kΩ	BN,BK,OR	Be sure last band is orange – not red	N
R3	1	2.2kΩ	RD,RD,RD		N
REG	1	MCP1825	3.3V / 500mA LDO	**Tip** - Put a bit of solder on the big pad, position, and solder that pad first.	Y
header	1	6 pin 90° **female**	To-GK serial to kit	Clip excess leads on bottom after soldering.	N
header	1	6 pin 90° **male**	FTDI connector	Clip excess leads on bottom after soldering	N
push button	2	SMD	Reset & Program sw	**Tip** - Put a bit of solder on one pad, position, and solder that pad first.	
C2	1	.1uF	#104 ceramic capacitor		N
LEDs	3	RED, YEL, GRN	Status LEDs	**"R", "Y", and "G" are marked on board.** **polarity:** Small flat on side, or shorter lead, goes **left**.	Y
LED	1	RED	Program Ready LED	**polarity:** Small flat on side, or shorter lead, goes **right**. On v1.3, this LED mounts too close to the FTDI connector. Mount higher and move to the left as much as possible. **See picture** Corrected on v1.3b.	Y
D1	1	1N4148		**polarity:** bend over the lead on the **banded** side (cathode). The body will go in the hole with the white circle. **See pictures below.**	
C3	1	2.2 or 3.3uF	50V electrolytic capacitor	**polarity:** "-" stripe towards top.	Y
C1	1	4.7uF	50V electrolytic capacitor	**polarity:** "-" stripe towards left.	Y
headers	2	8 pin female	carrier board headers	Place both headers on the male pins of the assembled carrier board. Then place on the two inside rows of the main board. Flip over and solder. Remove the carrier board for now.	N
headers	2	8 pin female	I/O breakout headers	Solder to the pads just outside of the carrier board headers..	N
header	1	4 pin fem.	I2C connector	Use header with standard pins. Save 4 Pin female header with long pins to test OLED display.	N
solder jumper	1	---	small pads on bottom above LEDs	Shorting this allows the 3 LEDs to work. If you want to use these I/O pins for other reasons, you can leave this jumper open. To short, solder across both pads.	--

Assembly Images: **(These images are for the v1.3 PCB but the v1.4 PCB is virtually the same.)**

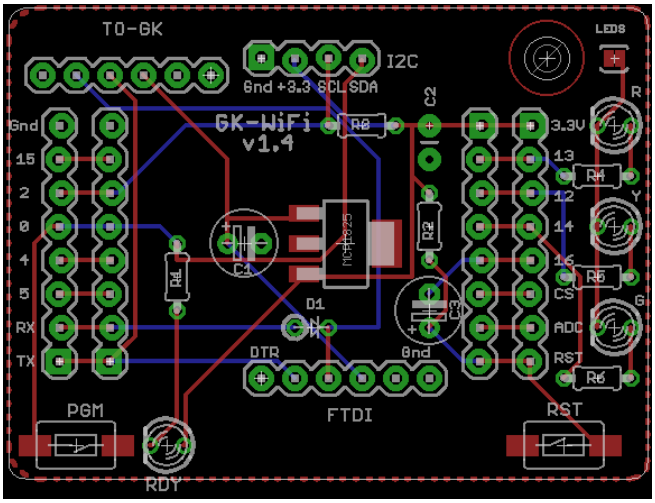
Completed board



PCB layout . . .



The carrier board with the ESP8266 plugs into the inside headers with the antenna over the FTDI header.



Step 2 - Installing the Arduino IDE and the ESP8266 Add-on:

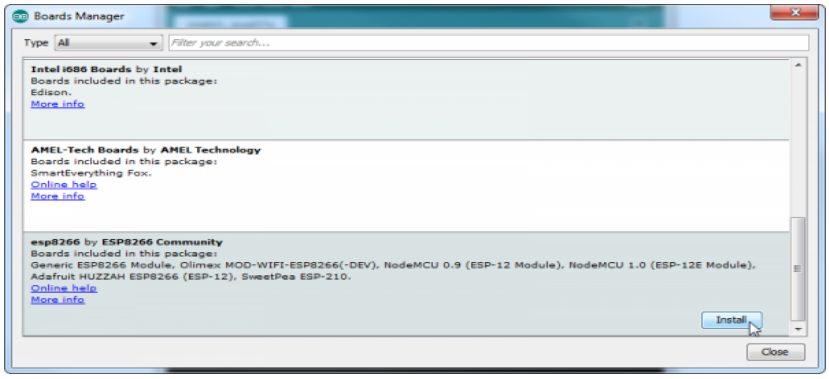
The GK-WiFi kit uses the ESP8266 to connect to WiFi and it is also the processor. Programming is done using the Arduino IDE, but In order to program this chip, you must install an add-on package to the IDE.

It may seem a little daunting at first, but if you follow the steps below you will get through it alive. There are also many guides online for this - for example, this one from Adafruit. These can provide additional details. Please note that I can provide only limited help with this setup. ... OK, let's take this one step at a time.

1. Get the IDE. I am currently using version 1.6.12 of the IDE - available here. Newer versions are constantly available but in the past some versions of the IDE had issues with the ESP8266 core, or other issues. Therefore. I can only vouch for v1.6.12.
   Tip: *If you're installing on Windows and you have older versions of the IDE that you want to keep, choose the "Windows zip" file install. Otherwise your older IDE will be removed.* Follow the guidelines for installing the new IDE.

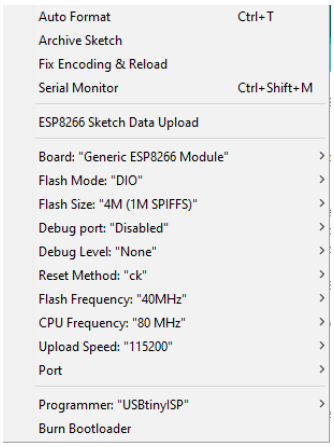
2. Open the IDE and go to File > Preferences. Add this line in the "Additional Board Manager URLs:" field;
   `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   Press OK.

3. Now go to Tools > Boards > Boards Manager. You should see a list of all the installed boards. At the bottom you will see and entry for **esp8266 ...** Click the **Install** button in that panel.



It will take a few minutes to download and install. Once it's complete, INSTALLED will appear next to the entry.

4. Now open Tools > Boards again. You should see **Generic ESP8266 Module**. Select it. Open Tools > Boards again and you should see settings similar to this:



Change the **Flash Size:** to 4M (1M SPIFFS) (for better memory usage info) and **Port** to your serial port.

Generally there is no need to change these other settings.

(Later you can try increasing the Upload Speed from 115200 to a faster speed but it's best to start with the default speed. Faster upload speeds can cause upload problems. Speed may change back to the default when you change board types.)

You're over the hump!

5. You can test out your setup by going to File > Examples and scroll down till you see **esp8266**, then pick **Blink**. Then click the Verify button on the toolbar (check mark). You should see the Blink sketch compile. (No need to load it.) Now you can use the Arduino to program the ESP8266 chip on the GK-WiFi board. More on that later. Note that you must disconnect the GK-WiFi board from the Geiger kit before you can upload any sketch.

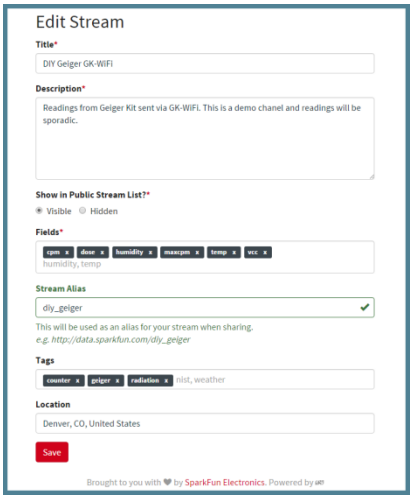
Step 3 – Create Accounts:

The software supports connecting to any or all of the three sites listed below. Use the steps below to create accounts for the sites you will be connecting to. Each will give you account credentials that are entered in the Credentials.h file of the sketch.

Radmon.org:

- Open an account with Radmon.org - **here.** Note the **Username** and **Password** you choose.
- You will use the username and password in setting up your sketch.
- You may also wish to download RadLog program for the PC. It's not required for directly connecting but it has some handy features. It's in the "Downloads" section of the Radmon.org site.

data.sparkfun.com: NOTE! As of October 2017 Sparkfun no longer supports this IOT site.

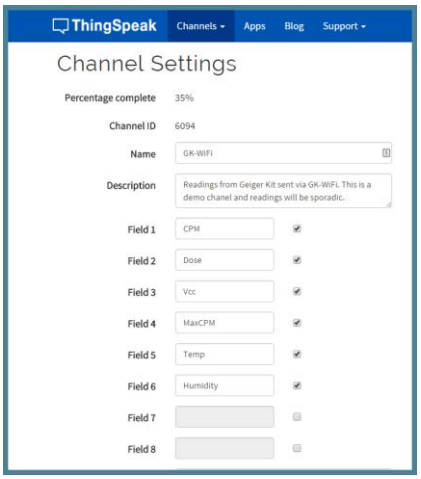


- Create a free datastream - **here.** During this process you will list the **Field Types** that you want. In order to use the sketch "as is" you must use the same Field Types that the sketch is providing data for. These are: `cpm, dose, vcc, maxcpm` Fields must be lowercase. *(see image left)*

- It's best to start with just those 4 fields. If you add the DHT22 for temperature and humidity later, you can add the additional fields which are: `temp`, and `humidity`.
- You will be given a set of keys - a public key, a private key, and a delete key. You will also be provided with the URL for your feed. (It contains the public key.) You have the option of emailing this info to yourself which is suggested.
- You will use the public key and private key in setting up your sketch. You can use your delete key to remove a datastream by going to this page.
- You may find the CSV export to be a handy feature.

For ThingSpeak:

- Create a ThingSpeak account - **here.** Click **New Channel** where you will list the fields that you want. The field names are predefined as field1, field2, field3, etc. So you can assign any descriptor to each field. However, in order to use the sketch "as is" the *order* of the fields is important. In other words, `field1` = CPM, `field2` = Dose, `field3` = Vcc, `field4` = MaxCPM. *(see image below)*
- Again, it's best to start with just those 4 fields. If you add the DHT22 for temperature and humidity later, you can add the additional fields which are: `field5` = temperature and `field6` = humidity.
- You will be given a Write API Key. You will use this key in setting up your sketch.
- You may find the MATLAB feature handy, but I haven't messed with it.



Your WiFi Credentials:

In addition to the credentials for the accounts selected, you will need your SSID and network password.

Step 4 – Program the GK-WiFi:

In order to use the GK-WiFi the software provided must be modified with your WiFi credentials (SSID and password) and the user name and password of all the IOT sites that you wish to connect to. There are also options (#defines), described later, that you can use to add or adjust some features.

- Get the latest sketch from **here**. Unzip it and put it in your sketchbook. Also get the library package from **here**. *You may be taken to dropbox.com to complete the download.(You do not need an account.)* Unzip it and put the contents in a "libraries" folder in your sketchbook. If you don't have a libraries folder there already, create one.

- Open the sketch in the Arduino IDE. On the main tab near the top, you will see a section called `Other User Settings` with #defines for various functions. They are described in the comments and below. However it's best to keep the defaults in the beginning.

- Open the Credentials.h tab. Here you will add your network credentials, use the #defines to select the accounts you wish to send to, and add the credentials for those accounts. The comments should give you the details you need. Save your sketch.

- Be sure your IDE is set for the **Generic ESP8266 Module** and run a Verify to make sure all is well. You may see compile red *warnings*, this is normal. In the end you should see a "Done compiling".

- Now, finally, it's time to upload the sketch. Plug the ESP8266 carrier board into the main board, but do not plug the GK-WiFi into the Geiger kit. (The ESP8266 uses lots of power at startup and some USB ports can barely handle it.) Plug in your FTDI into the pins on the bottom. You should see the red led on the ESP8266 light up.

- Press and hold the PGM button on the GK-WiFi you should see the RDY led light up brightly. While holding that switch, press and release the RST button, then release the PGM button. You should see the RDY led light up dimly. If you do, the ESP8266 is ready to accept a program.

- Click Upload on the IDE. Compile time is a lot slower for the ESP8266. Eventually you will see a row of red dots across the bottom screen and finally the message "Done uploading". Have a beer.

- If you get errors, disconnect your serial cable to the USB to Serial adaptor, reconnect it, go back into program mode, and try again. After any upload fail, make sure you put the board back in program mode as described above. If are still having problems uploading, see **Appendix V - Software Problems**.

Step 5 - Run it:

First, a little menu work on your Geiger kit . . .

- Go into the Setup menu and set the LOGGING PERIOD for the period that you want to send data at. For radmon.org no less than 30 seconds is recommended, but you're impatient right now, so set it at 15 seconds.
- On the **GK-B5**, or **GK-Mini** make sure you have the “USE RADLOGGER” option turned off.
  On the **GK-Plus**, set the SERIAL OUTPUT MODE to the “GK-B5” option.

Power off the Geiger kit, disconnect the FTDI, and plug the 6 pin female header (serial input) of the GK-WiFi into the FTDI connector of your Geiger kit. Now power on the Geiger kit. You should see . . .

- The red led go on - connecting to WiFi, and then off - connected!
- The yellow led go on - waiting for data from the Geiger kit, and then go off - got data!
- The green led will flash when sending data to each site - data sent!

Now you can go to the IOT sites you signed up for and check your results there.

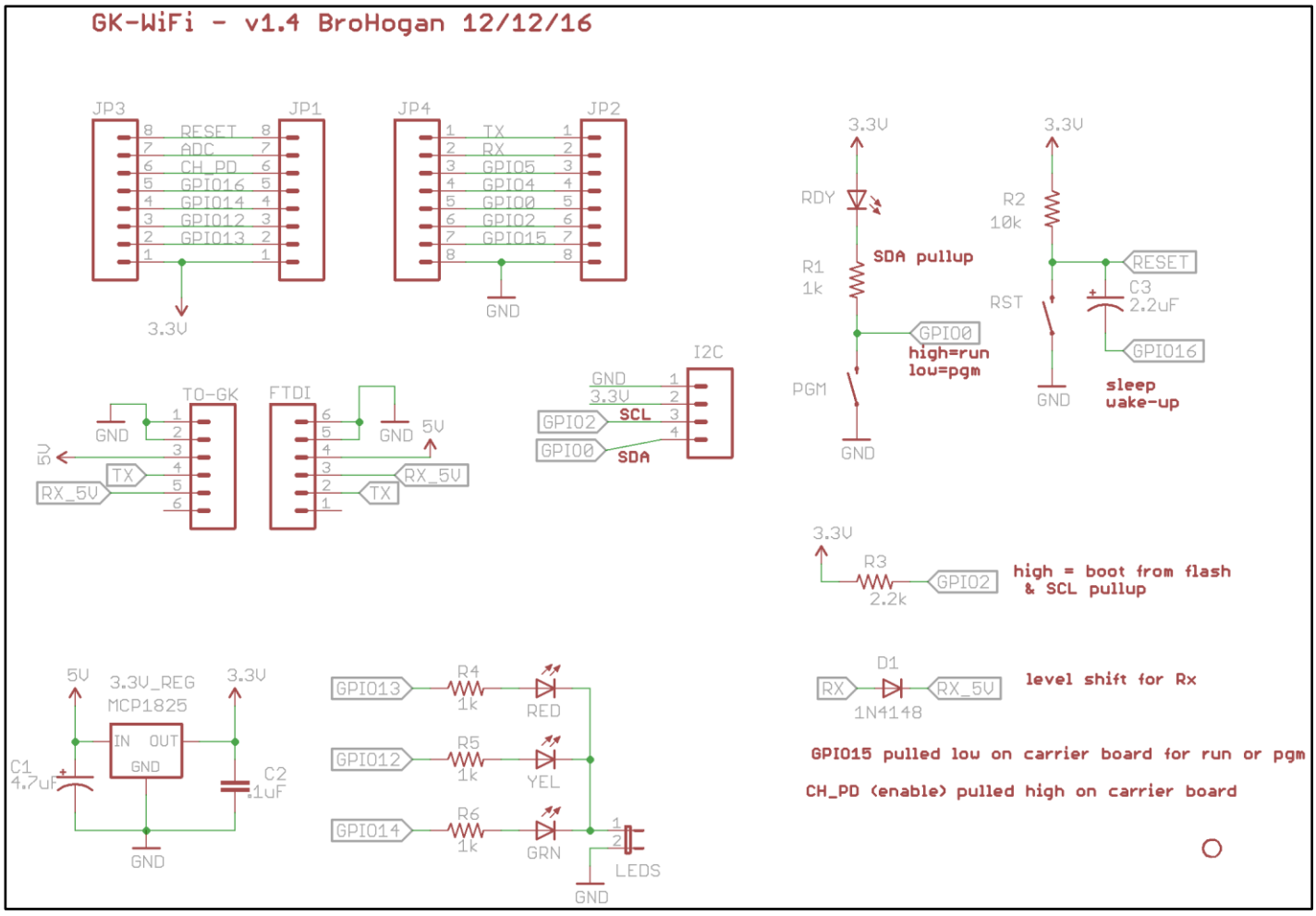
If all went well - congratulations!

If not, there is the beginnings of a troubleshooting section in **Appendix V**.

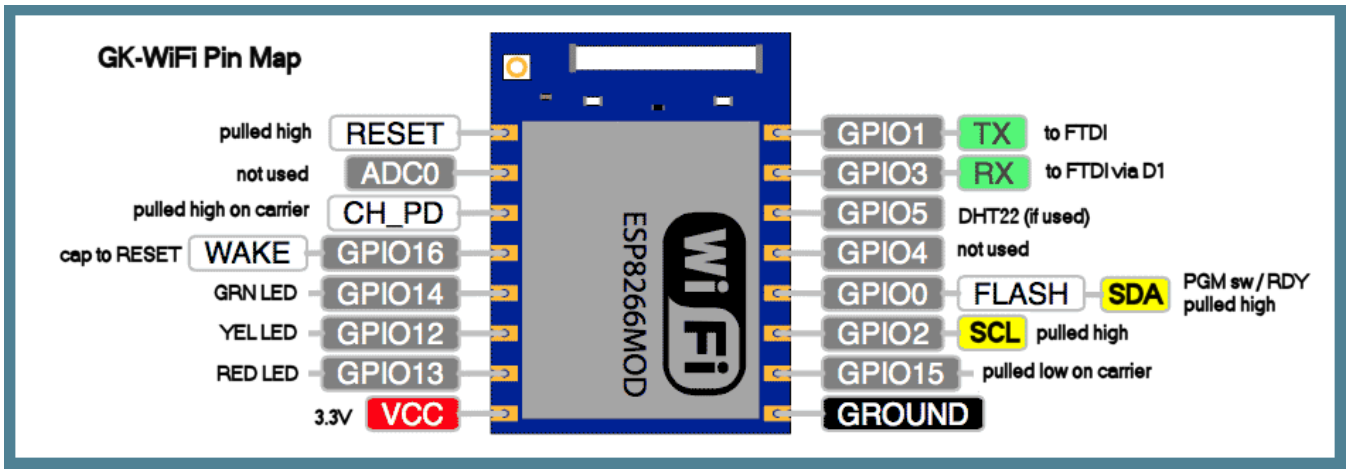
Be sure to check out **Appendix I - IV** below for other details.

Enjoy your kit!

Appendix I - Schematic:



Pin Map:



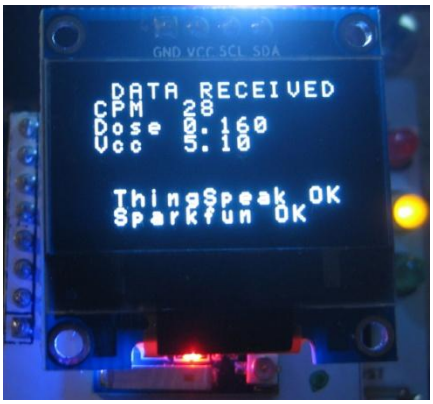
Hardware Notes:

The ESP8266 goes into programming mode differently than the ATmega microprocessors. It requires that GPIO 0 be set low while starting after Reset. Therefore two buttons are used - one to pull GPIO 0 low (PGM) and the other to pull Reset high. *Some ESP8266 development boards use DTR to accomplish this which is more convenient but can have issues with serial output .*

GPIO 16 is tied to RESET through a 2.2uF capacitor. This is used to wake up (restart) from sleep mode. Unlike the ATmega microprocessors, the ESP8266 will not pick up where it left off before sleep mode.

Appendix II - Add-ons and Casing:

Adding an OLED Screen



At the top of the board you'll find a 4 pin I2C header. You can use it (with programming) to add I2C devices. It can also be used to add a little OLED display. You need to set `#define USE_OLED` to 'true' and add the display. It will show the data received and who it was sent to as shown on the left.

You can find these .96" I2C OLEDs on eBay (example). Please note that some have the Gnd pin on the left. These will plug into the I2C header directly. Others have the Vcc pin on the left. These will need to be connected to the I2C header with wires that cross the Gnd and Vcc. Given the remarkable variability of Chinese products, I can't be responsible if you get the wrong type.

If you have an OLED that can plug in directly, you can use a 4 pin female header with long pins that may be included so that the display will clear the carrier board.

Adding Temperature / Humidity Sensor

Version 1.7 of the sketch is already setup to support two kinds of temperature and humidity sensors. The Adafruit HDC1008 sensor is preferred over the DHT22 sensor. Since it's I2C, it doesn't use extra I/O pins, and it is much more stable and more accurate than the DHT22. To use either sensor set the appropriate `#define` and set the `#define CELSIUS` as needed.

The HDC1008 can be connected to the I2C header or if you are already using that for the OLED, it can be connected to the I/O header - GPIO0 is SDA, and GPIO2 is SCL and 3.3V and ground from the I/O headers or the Aux power header.

If you are using the DHT22, connect it to Vcc and Gnd, with the output pin to GPIO5. It's also best to add a 10k pullup to the output pin.

Once you set either #define the sketch will send temperature and humidity to ThingSpeak, so you must add the field names mentioned in Step 3 to these accounts. (Generally, the data sent must match the field names exactly or no data will be received.)

Expanding the I/O

If you like to add more sensors - especially analog or if you need a DAC, I suggest adding my favorite chip - the PCF8591. Just hang this puppy on the I2C buss and you have 4 analog inputs and a "true" analog output (not PWM). The DIP version is available on eBay and I see there are also very cheap modals with this chip. Just search eBay for "PCF8591".

At some point I will link a test sketch to make it easy to use this chip, then it's simply a matter of sending it's values to the sites you'd like.

Relocating the GK-WiFi Board in a Case

Depending on which Geiger kit you have, the GK-WiFi will hang off the right, bottom, or left side of the kit. So when casing you should add a 6 pin 1:1 DuPont cable or make your own.

Appendix III - Power:

For long duration monitoring it's expected that the Geiger kit and GK-WiFi will be run with a 5V power adaptor. The ESP8266 can use a lot of power - up to 170mA when transmitting.

However, now that you're wireless it will tempting to run this sucker on batteries - maybe solar powered. My *hunch* is that there will be some challenges but it's doable. Here are a few tips:

- There is already a sleep mode built into the sketch. You need to set `const int sleepTimeS` to the seconds that you want to sleep for. There are some notes on sleep mode in the header of the sketch.
- You can control the power to the HV circuit in the Geiger kit by first separating it from the processor by opening the UC-PWR jumper on the Geiger kit. Then, HV is powered at the terminal block and the processor is powered by the user power header. The HV power can be sourced by an I/O pin connected to the terminal block. But the rest - matching sleep on the MCU and ESP8266, I/O for HV power, etc. - will be up to you.
- You will probably want to open the LED solder jumper on the GK-WiFi board and maybe remove the leds on the ESP8266.
- Let me know if you pull this off!

Appendix IV - Use as an ESP8266 Development Board:

The GK-WiFi board was also designed to be a development board for the ESP8266. The carrier board allows you to switch to different applications easily.

If you intend to explore the ESP8266, one of the best resources is the **ESP8266 Community Forum**.

Some things to keep in mind when using the GK-WiFi board as a development board are:

- The ESP8266 is a 3.3V device. This includes all the I/O pins. Do not directly feed them 5V signals - level shift.
- The maximum current on an I/O pin is 12mA.
- Open the LED solder jumper to use GPIO pins 12, 13, and 14. GPIO pins 4 and 5 are not used (although the sketch expects a DHT22 on pin 5).
- The ESP8266 is a power hungry device. Powering with an FTDI board may reduce WiFi range, etc.

I'd be interested in hearing about other uses you have found for this board.

Appendix V - Troubleshooting:

It's been a long road and there were many ducks to get in a row. Here you find the latest troubleshooting hints I've come up with based on my own experience and what I've learned from customers. It will be updated with experience gained, so check the latest revision of these Build Instructions.

If your still having problems contact me. Note however, that I tend to respond with the same level of detail as was given to me. So please provide a good description of the problem and tell me what you tried. If I respond with questions and you ignore them and then flood me with new details, it usually makes me angry ☺. I'm a step by step guy.

Hardware problems

- Double check your connections between the ESP8266 and the carrier board. Check continuity if unsure.

- **Check voltages** - Remove the ESP8266 carrier board and any other peripherals (DHT22, etc.) you may have added. Plug the GK-WiFi board into the Geiger kit and power it on. Attach your meter to Gnd. Go down the rows of I/O pins. You should see:
  - ~3.3V on Vcc, GPIO 2, and RST and about 2.4V on GPIO 0.
  - You may see some strange low or negative on other pins from an antenna affect. This is normal.

- Use the schematic in **Appendix I** to run more continuity and voltage checks.

Software problems

- **Compile Issues** - You shouldn't have these if you followed Steps 2 and 4 closely.
  - Make sure the library package is installed in the correct location.
  - If you have general Arduino questions, it's best to ask on the Arduino Forum.

- **Won't go into program mode** - Make sure you are using the proper button sequence described in Step 4. If so, it could be the connections between the ESP8266 and the carrier board, or a bad ESP8266

- **Can't upload sketch** - `error: espcomm_open failed` (or the like)
  - Make sure you disconnect the GK-WiFi board from the Geiger kit before uploading a sketch. If you have any other peripherals (DHT22, etc.) try disconnecting them also.
  - Make sure you are putting the ESP8266 into program mode as described in Step 4. The RDY LED should be lit dimly. Try holding the RST button a bit longer before you let it go.
  - Note from Sparkfun: *"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 continue to fail, try turning the ESP8266 on then off, or unplug then replug the FTDI in."*
  - So the hierarchy of what to try is: try to upload again, remove power, disconnect FTDI, and lastly, close the IDE and reopen it.
  - If you changed to a faster Upload Speed, set it back to the default - 115200.
  - Load the Blink sketch for the ESP8266 - File > Examples > esp8266 >Blink. Try to upload that.
  - Here is an alternate button pressing technique that some use to get to program mode: *Press and hold the RST button, and then press and hold the PGM button. Release the RST button, and while holding the PGM button pressed, click the Upload arrow in the Arduino IDE. The sketch should compile in a minute or so, and when it is complete, release the PGM button.* (I have not tried this to solve upload problems.)
  - If your USB to Serial adaptor can't supply enough power, you may have to power the WiFi board separately while uploading.

Connection problems

- If you're not seeing "WiFi Connected" make sure your SSID and password are correct in the Credentials tab. You might try to load a simple example that shows available WiFi networks by going to File > Examples and scroll down till you see **ESP8266WiFi**, then pick **WiFiScan**.

- Debug Mode - The easiest way to diagnose connection problems is to upload a debug version. Setting `#define DEBUG` to true generates test counts that increment with each post at the interval set by `#define TEST_POST_PERIOD`. It also provides serial output of what is being sent to each site and the response from that site.

  The response from each site can be very informative. For example, the response may tell you that field names sent do not match the field names set up on the site. This may be the most common connection problem, so check that the field names match. Debug mode (as well as the OLED) will also let you know that you are connected to your WiFi network.

- **Constant Restart** - This may be a sign that another sketch you loaded may have messed with the watchdog timer and the change has been written to the chip. To find out, open the Serial Monitor. After the initial garbage is displayed at 74880 baud, if you see "`wdt reset`" periodically, then that's the problem. If you Google "ESP8266 wdt reset" you will find articles to help you. They generally involve re-flashing the chip and Python scripts. I am looking into a easier solution - perhaps using this tool (Windows only).