Build Instructions and User Guide for the GK-RadMon Kit

Hardware version v1.1 / Software version v1.3 / Doc Rev. 1.3

Getting Started

Congratulations! This kit is a great way to get your Geiger readings on the web with a dedicated monitoring station driven by the ESP8266. I hope you enjoy building it as well as using it.
Since this is the first public release of these build instructions, I would appreciate any suggestions to improve 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:

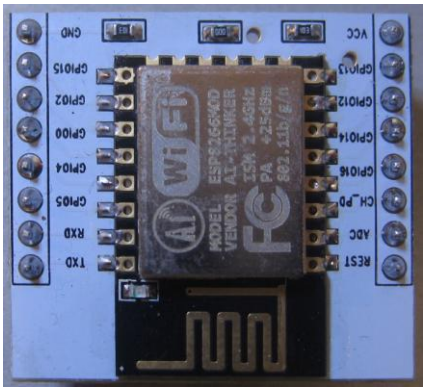
Build the Carrier Board

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.

The carrier board will mount on the main board as shown below.

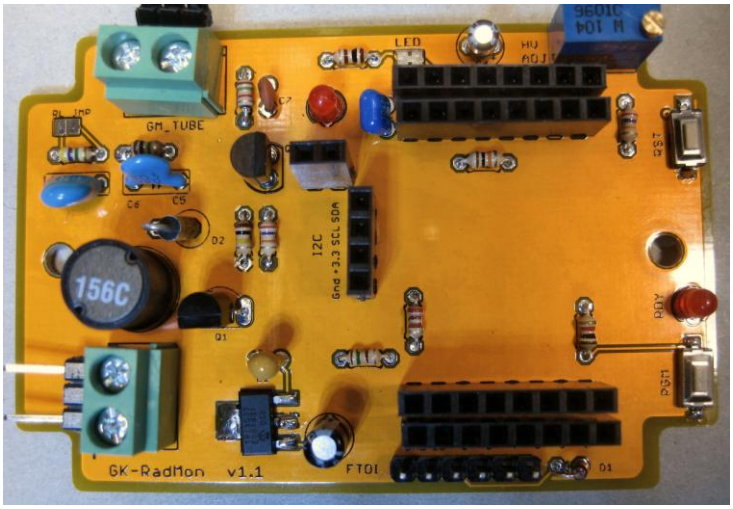
Build the Main Board

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.

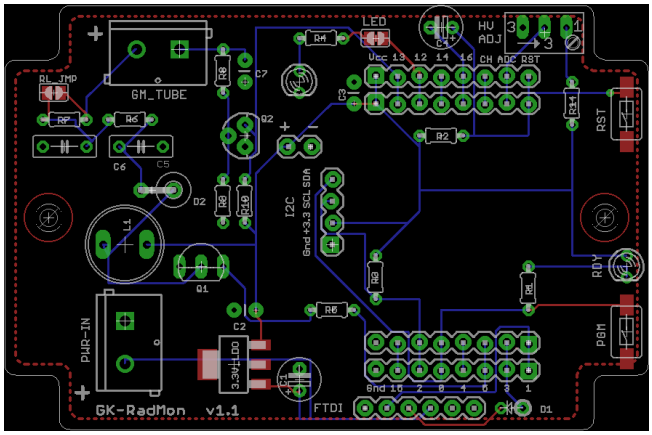
Build Sequence and Parts List					
Ref #	**Qty**	**Value**	**Description**	**Notes** **polarized? ->**	**Y N**
PCB	1		2.95" x 1.95" (~7.5 x 5 cm)	**Orientation:** GM_TUBE terminal is at **top** of board.	--
R1	1	1kΩ	BN,BK, RD	Be sure last band is red – not orange	N
R2	1	10kΩ	BN,BK,OR	Be sure last band is orange – not red	N
R3	1	2.2kΩ	RD,RD,RD		N
R4	1	100Ω	BN,BK, BN		N
R5	1	150Ω	BN,GN, BN		N
R6	1	1MΩ	BN,BK,GN		N
R7	1	4.7MΩ	YL,VT,GN		N
R8	1	1.5kΩ	BN,GN, RD		N
R9	1	100kΩ	BN,BK,YL		N
R10, R14	1	27kΩ	RD,VT,OR		N
3.3V REG	1	MCP1825	3.3V 500mA LDO Iq=120uA	**Tip** - Put a bit of solder on the big pad, position, and solder that pad first.	Y
push button	2	SMD	Reset & Program sw	**Tip** - Put a bit of solder on one pad, position, and solder that pad first.	
C7	1	330pF	#331 ceramic capacitor		N
LED	2	RED	RDY and COUNT	**polarity:** Small flat on side, or shorter lead, goes down on both.	Y
D1	1	1N4148		**polarity:** bend over the lead on the **banded** side (cathode). The body will go in the hole on the right. **See pictures below.**	
D2	1	UF4007	1000V 1A Ultra Fast diode	**polarity:** Bend over the lead on the **banded** side (cathode) The anode (body) goes into larger silkscreen circle, and the cathode goes into the hole on the right. **See pictures below.**	Y
C5, C6	2	.01uF	#103 HV ceramic cap		Y
Q1	1	STX0560	NPN HV transistor (straight leads)	**polarity:** flat side towards bottom Spread leads a bit – don't try to push in all the way to PCB.	Y
Q2	1	2N4401	NPN BJT transistor (straight leads)	**polarity:** flat side right Bend the center lead back – don't try to push in all the way to PCB.	Y
C2	1	1uF	#105 Tantalum Cap	**polarity:** "+ bar" or longer lead, goes right	Y
C4	1	2.2uF	50V electrolytic capacitor	**polarity:** "-" stripe towards left.	Y
C1	1	4.7uF	50V electrolytic capacitor	**polarity:** "-" stripe towards top.	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 or tack in. Remove the carrier board for now.	N
headers	2	8 pin female	I/O breakout headers	Solder to the pads outside of the carrier board headers with a small space in between.	N
header	1	4 pin fem.	I2C connector	If a long pin header was included, save and use the short pin header.	N
C3	1	.1uF	#104 ceramic capacitor	close fit at left end of the top two 8 pin headers.	N
header	1	2 pin fem.	AUX power		N
header	1	6 pin male	FTDI connector	FTDI board plugs in vertically with GND on right.	N
screw term	2	2 pin	5 mm pitch	**Suggestion:** If you trim the leads, don't use your precision cutters for this.	Y
L1	1	15mH	15mH 80mA 31.0 ohms 13R156C	Orientation shouldn't matter but install as pictured	N
HV Pot	1	100K	blue 25 turn HV pot	**Note:** pot is preset for ~400V (~2kΩ ADC to ground)	Y
solder jumper	1	LED disable	small pads on top center	Cutting this disables the COUNT LED.	--
solder jumper	1	RL_JMP	small pads on left	Short this ***only*** if anode resistor will be at the GM tube.	--

Assembly Images:

Completed board . . .



PCB layout . . .



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

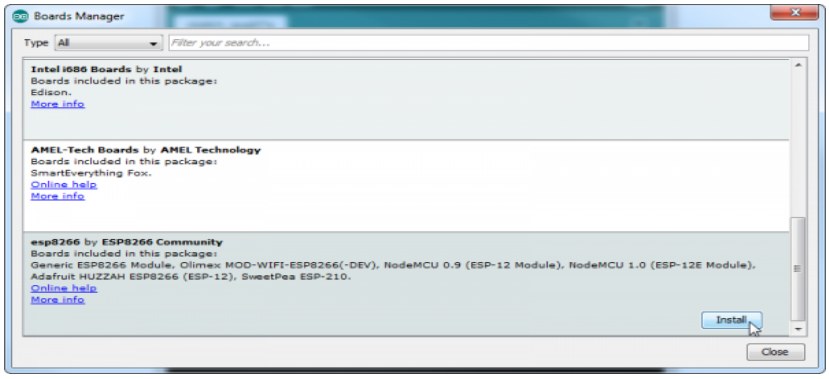
The GK-RadMon kit uses the ESP8266 to connect to WiFi and it is also the processor. Programming is done using the Arduino IDE, but In order to program this chip, you must install an add-on package to the IDE.

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 and this one from Sparkfun. 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'm currently using 1.6.5 - available here. *As of this writing the ESP8266 core is not ready for IDE version 1.6.6.* ***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.

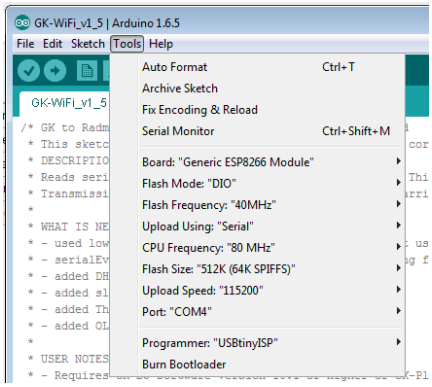
2. Open the IDE and go to File > Preferences. Add this line in the "Additional Board Manager URLs:" field;
   `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   Press OK.

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:



Generally there is no need to change these other settings - except for your **Port:**.

(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-RadMon board. More on that later.

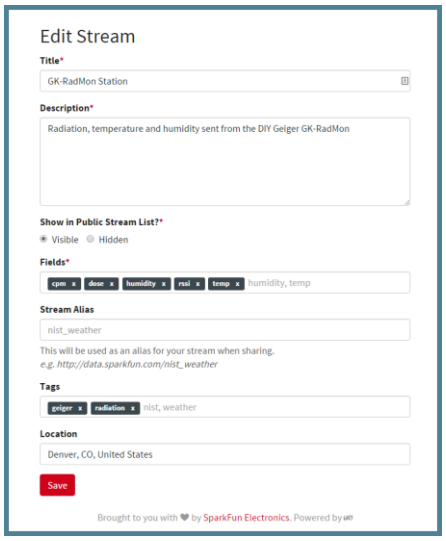
Step 3 – Create Accounts:

The software supports connecting to 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: (no relation)

- 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:

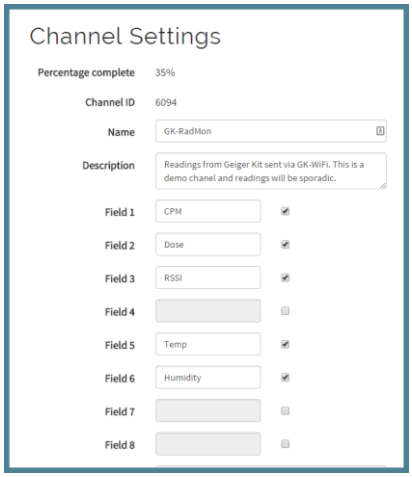


- Create a free datastream - **here.** During this process you will list the **Field Types** that you want. In order to use the sketch "as is" you must use the same Field Types that the sketch is providing data for. These are:
  `cpm, dose, rssi` Fields must be lowercase. *(see image left)*

- It's best to start with just those 3 fields. If you add the HDC1008 or 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. Unlike Sparkfun, 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` = RSSI, `field4` is not used. *(see image below)*
- Again, it's best to start with just those 3 fields. If you add the HDC1008 or 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-RadMon:

In order to use the GK-RadMon 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. You will also need the library package from **here**. 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. (Do not put the libraries in the "libraries" folder under your IDE install.)

- 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 will 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 - antenna towards switches. You will probably not be able to power the GK-RadMon with only the USB to Serial adaptor. Use with a separate supply - 6V max. The ESP8266 uses lots of power at startup and USB ports may not handle it. Now plug in your USB to Serial adaptor into the pins on the bottom. The GND pin on the USB to Serial adaptor goes to the right.

- Press and hold the PGM button on the GK-RadMon you should see the RDY led light up brightly. While holding that button, press and release the RST button, now 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 the Tx & Rx leds flashing, 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 III - Software Problems**.

Step 5 - Run it:

Finally! But take a quick look at **Appendix I** and **Appendix II** to get an idea about adjusting the HV and powering the kit.

Now you can add your GM tube. 1/4" Fuse clips are provided to connect to the most common GM tubes. A heavy wire is not necessary for the connection. Observe polarity for the tube. The + on the screw terminals goes to the anode (+).

Power it up and you should see the Count LED at the top of the board flashing. If you have an OLED you will see updates on CPM and uSv. (By default, the CPM to uSv is #defined at 175.43 which is the common for the SMB-20 and STS-5.)

After one minute, it will send to the IOT sites that are configured in the Credentials tab. The default is ThingSpeak and Sparkfun. For those sites, the kit will send; CPM, uSv, and RSSI. RSSI is a measure of the signal strength of your WiFi connection. It is a negative number from -1 to -100. The closer to zero the better the signal. Generally anything above -60 is good. If you have an OLED you will see the values being sent and to whom they are sent.

After sending, the GK-RadMon will sleep for 4 minutes. Then it will reboot and begin the process all over again.

You can go to the IOT sites you signed up for and check your results there. If you don't see any data, take a look at the serial output. Debug is on by default and you will see the response from the site. Generally the most common problem is a mismatch between what is sent and what the site expects.

If all went well - congratulations! If not, there is the a troubleshooting section in **Appendix III**.

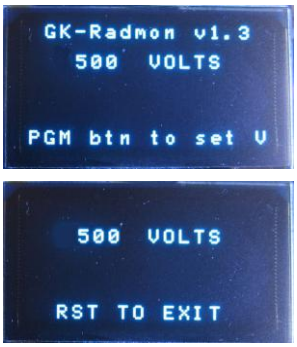
I hope you enjoy your kit!
John

Appendix I - HV Adjust:

The HV Adj pot comes pre-set for a GM tube voltage of about 400V. So if you're using an SBM-20 or the like you should be fine right out of the box. But hey, you got to tweak it, right?

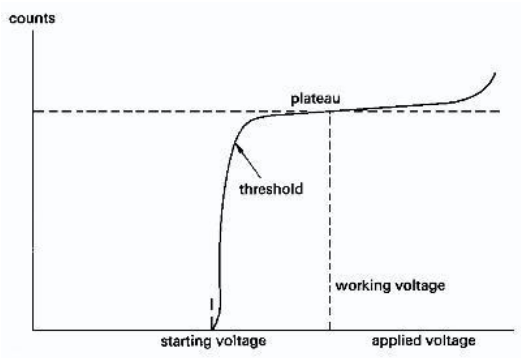
Rather than torture you with helpful background information, let's go right into the procedure.

1. Remove the GM tube and power the kit with a good 5V source.
2. Press the RST button. If you have an OLED you will see the screen on the right. If not, attach your serial adaptor and open the Monitor on the IDE at 115200 baud. You will see "`400 VOLTS`" (or your current HV), "`Hit PGM button NOW to set Voltage`", and then "`Counting...`".
3. Now that you know what to expect, hit the RST button and try again. This time, press the PGM button before you see the "`Counting...`" line. You have about 5 seconds to do this. If the PGM button press isn't recognized, hit RST and try again. *(It's a bit tricky to read this particular button in code.)*
4. When you get into the voltage set mode you will see the screen on the right if you have an OLED. If not, you will see "`Hit RST to exit when done`". After that, you will see a stream of voltage readings.
5. Now you can adjust the HV pot to your desired voltage - clockwise to increase and counter-clockwise to decrease. The range is about 350-1000V but, Caution! Going over 800V may cause arcing on the board. This won't harm the components but continued arcing will carbonize that area on the PCB and you will have to remove it. *(Future PCB revisions may have more isolation but there is no point in cranking it up unless you have an unusual tube.)*
6. Some things to note:
    a. Voltage is re-checked once a second in the adjustment mode.
    b. There is play in the pot when reversing the direction.
    c. You may get old trying to make adjustments down to the volt - you're not tuning a violin.
    d. Displayed voltage is pretty accurate - a few % in my tests.
7. When you're done, press the RST button.



Helpful Background:

GM tubes typically have a large operating range. You can get a better idea of this by looking at its “plateau” as shown to the right. Within this plateau the tube will have about the same sensitivity regardless of the voltage. **Put another way, once the tube is in its operating range, the HV you run at is not critical, and has very little effect on accuracy**.



One thing to keep in mind though, is that the HV will tend to sag during counts in the thousands of CPM. Though this is unlikely for a background monitor, it is recommended that the voltage be set near the high end of the tubes range. Doing this will keep the tube running in its operating range during high counts.

If you want to measure the HV directly keep in mind it's a bit tricky. The GM tube needs a lot of voltage but only a tiny amount of current. The HV circuit only needs to provide a very tiny current, and that's what it does. This is good because the battery will last longer - and it won’t kill you. However, it makes measuring the high voltage a bit more complicated.

When measuring voltage, a typical DVM will put a load on the circuit it's measuring of about 10MΩ. This load is far too much for the tiny amount of current available, and the DVM will read much lower than the actual voltage. A meter with a 10MΩ input impedance will read about 133V when the voltage is at 400V. You need about a gig-ohm (1000MΩ) of input impedance to get accurate values of the HV for Geiger circuits.

One way to increase the input impedance of your DVM is to put large resistors in series with the probe and multiply the reading you get. Adding 9 10MΩ resistors in series adds 90MΩ. If you want a full gig-ohm of input impedance, it's best to just buy a single 1GΩ resistor (example). Once you have the resistors added in series with the meter, you have to multiply the reading by some factor. The formula for this is:

$$(R_{probe} + R_{meter} / R_{meter}) \times V_{reading} = V_{actual\ voltage}$$

So for example, if you built a 90MΩ “probe” for a typical 10MΩ meter, you’d have **90 + 10 / 10 = 10** so you’d multiply your reading by 10. If you used a 1 GΩ resistor (1000MΩ) with the same meter it would be **1000 + 10 / 10 = 101** so you’d multiply your reading by 101.

The HV is best measured between ground and lead coming from the top (band side) of D2, but using the + GM_Tube terminal for the positive is about as good. (Do not use the other GM_Tube terminal for ground.)

Appendix II - Power:

Initially you will want to power the GK-RadMon with a 5V power adaptor (6V maximum). The ESP8266 can use a lot of power - up to 180mA when transmitting and over 70mA when running. A power adaptor with at least 500mA is recommended.

However, now that you're wireless it's likely you will want to run this sucker on batteries - maybe solar powered. At this writing I have just begun to use solar power. See **Appendix VI** for a description, but first, here are some considerations:

- In the beginning of the main tab there are two #defines - `COUNT_PERIOD` and `SLEEP_PERIOD`. By default they are set to a 1 minute counting period and a 4 minute sleep period. This means that every 5 minutes the GK-RadMon will update the servers. Also note that a full 1 minute count provides the best accuracy.

  Given the solar power budget below, I doubt if sleep period can be shorter. The fact is, the ESP8266 is just to power hungry to be run constantly. I also feel that a 5 minute update is fast enough to catch most trends in background radiation.

- The HV circuit powers down in sleep mode and there should be no LEDs lit during sleep. You can save a tiny bit of power by opening up the LED solder jumper on the board to disable the count LED, but only during the counting period. Similarly, you can remove the LED from the ESP8266 module which flashes with I2C activity.

- An OLED display is a nice addition, but it's probably not needed when it's sitting in your backyard. You can unplug the OLED and the kit will still operate.

- If you find other ways to squeeze blood out of this turnip, let me know.

Power budget:

Lets discuss the power budget needed to run the GK-RadMon kit on solar power to see what is feasible.

Based on my tests (normal background, no OLED) I have measured the following:

Counting: 78mA
Transmitting: 107mA
Sleeping: 108uA

Let's disregard transmitting for now since it only lasts a few seconds. Using the default setting of 1 minute count and 4 minutes sleep . . .

- The average power consumption is **15.68mA** (1 x 78mA + 4 x .108mA / 5 minutes)
- Now assume a 2700mAH LiPO. It would last **172 hours** (2700 / 15.68) or **~7.2 days** without a charge.
- For this power budget we'll use this Adafruit solar cell and this Adafruit charger [NOTE]. This 2W cell will theoretically provide 330mA in perfect sun. The charger has inefficiencies. let's say the whole system is 75% efficient.
- So **247.5 mA** (330 x .75) to the battery during full sun.
- Given a 2700mAH LiPO, a flat battery would fully charge in about **11 hours**. (2700 / 247.5)
- So *in theory* the system has 7.2 days to catch 11 hours of solar charging.

The actual performance will not be quite that good, but sustaining the system on solar power with that gear appears to be feasible. To see how this actually worked out, see **Appendix VI.**

If you want to power the GK-RadMon with solar power, I recommend that you do not skimp on the solar panel and on the solar charger. If a smaller panel is used, an option would be to increase the sleep time. For example, using the above with a sleep time of 9 minutes the system would need 14.2 days instead of 7.2 days to catch 11 hours of solar charging.

[NOTE] Adafruit has a great article on their charger and solar charging in general - here.

Appendix III - 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

- The best thing to do first, is to go through the build table again and make sure you have the right part in the right place with the right orientation. Also check that you haven't forgotten any solder joints.

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

- If you are not getting counts during the counting phase, check your HV. With a DVM between ground ("-" on the 2 pin header) and the + of the tube you should see ~130V when the HV is set for 400V. Also check connection and polarity of the tube.

- Use the schematic in **Appendix V** to run continuity and voltage checks. Look for 3.3V at the points expected. On the ESP8266 you should see 3.3V on Vcc, CH_PD, Reset, GPIO0, and GPIO2. You should see a very small voltage on the ADC pin - like 0.044V. This the HV pot adjustment. If you have a scope, you should see a nice 2.5kHz PWM signal on GPIO5 during the counting phase.

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.

- **Can't upload sketch** - `error: espcomm_open failed` (or the like)
    - Make sure you have good power the GK-RadMon board The power supply should be able to supply at least 500mA . If you have other peripherals (sensor, OLED 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.
    - A 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 uploading 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. This can help a lot.
    - 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.)

Connection problems

- If you're not seeing "WiFi Connected" at the end of the count period 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 is the best way to diagnose connection problems. It provides serial output of what is being sent to each site and the response from that site. The response can be very informative. For example, it 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** - If you have loading other sketches they may have messed with the watchdog timer. If you see "`wdt reset`" in the serial output constantly, 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).

Appendix IV - Adding Peripherals:

Adding an OLED Screen

At the center of the board you'll find a 4 pin I2C header. You can use it to add I2C sensors, but it's also setup to accept a little OLED display. The display will show the progress of the counts, the final data sent, and who it was sent to. It's pretty cool!

There are two sizes of I2C OLED that I've tried that work - .96" and the 1.3" (both monochrome). You can search eBay for these. Since the 1.3" and the white .96" have become more expensive, I use the blue .96" like these. Please note that they must be the I2C type rather than the SPI type. (I have heard that some have different pinouts, so check this.)

Finally note that the I2C header is the same height as the carrier board headers so if you want to just plug the OLED in to see it, you must extend the pins on the display. A 4 pin female header with long pins may be included for that purpose. Other I2C devices can be attached to the I/O header - GPIO0 is SDA, and GPIO2 is SCL.

Adding Temperature / Humidity Sensor

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 GPIO14. It's also best to add a 10k pullup to the output pin. You can add it right at the sensor.

Once you set either #define the sketch will send temperature and humidity to Sparkfun and 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 there are also very cheap modals with this chip. Just search eBay for "PCF8591".

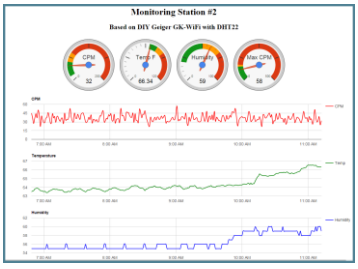
A test sketch to show you how to use this chip is here. It makes it easy to use this chip, then it's simply a matter of sending it's values to the sites you'd like.

Casing the Board

The GK-RadMon board was designed to fit in this case. This was done knowing that the case had no room for a battery or GM tube. (It was just an option better than no option!) A larger case like this or this will hold everything. You can mount the GK-RadMon on a substrate that fits your particular case.

Note that in the voltage set OLED screen also shows the current RSSI. This may be handy when positioning the GK-RadMon outside.

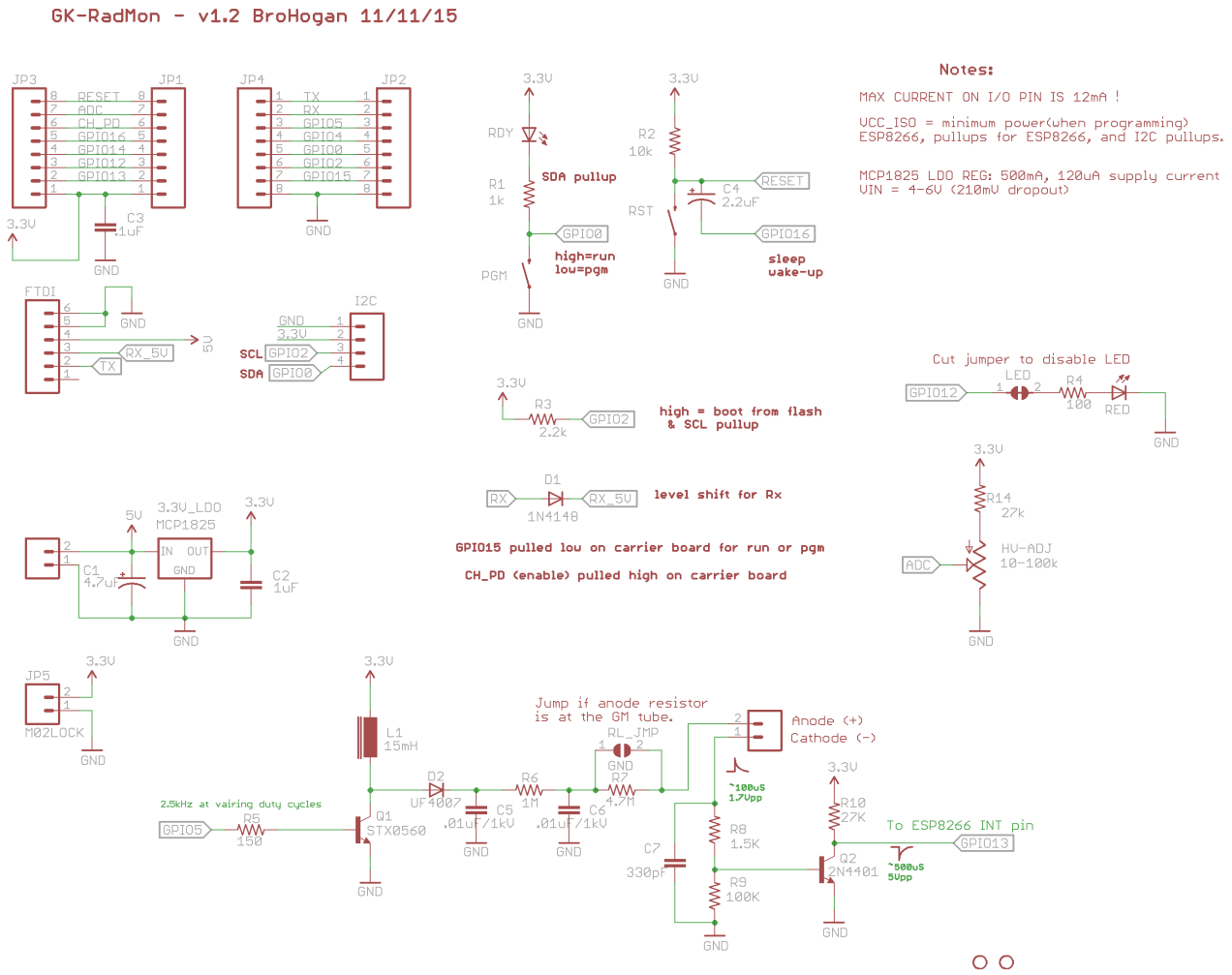
Using Google Charts to Display Your Data



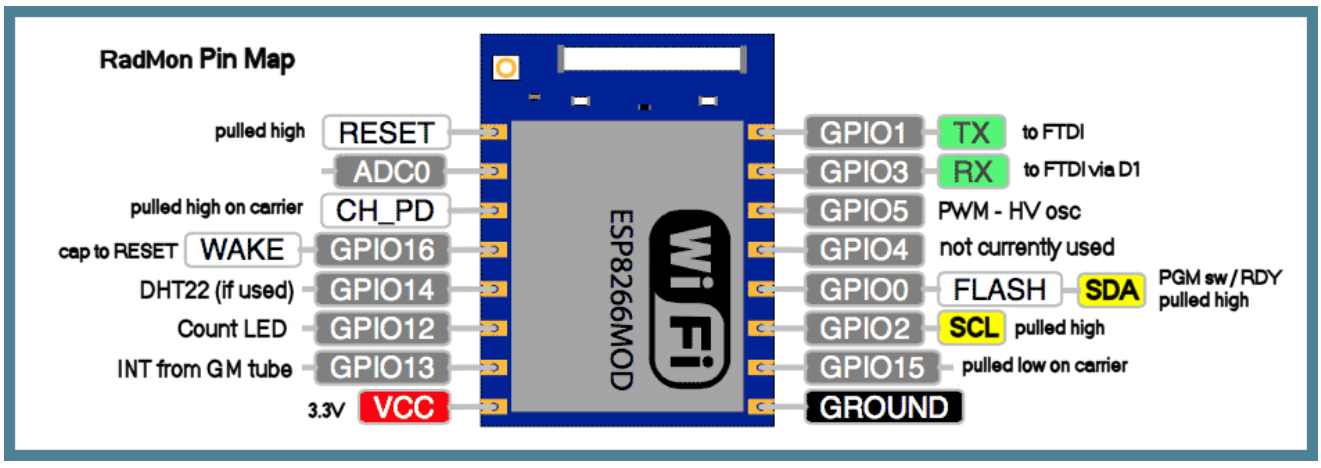
I've experimented a bit with some HTML to use Google Charts. (I'm not an HTML guy though!) The idea is to read the data from the Sparkfun IOT site and display it with Charts on a local page living on Dropbox or your own site.

If you're interested in doing this, the HTML is **here**, and the comments in it should be all you need.

Appendix V - 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. Instead it restarts.

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

Appendix VI - Example using Solar Power:

It is hoped that documenting this project will help you with your own.

Notes on the build:

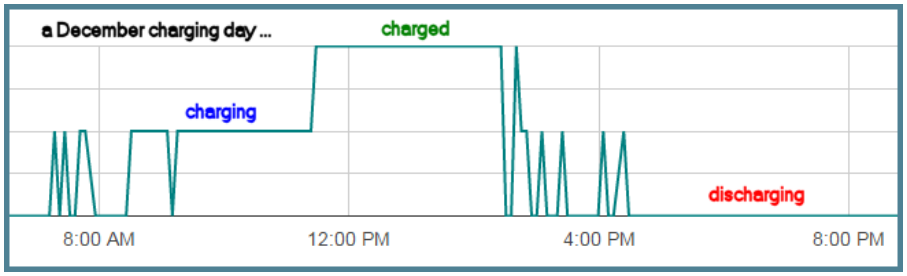
- This project used this Adafruit solar cell . This is a 6V 2W cell that is said to provide 330mA in perfect sun. As a cheaper alternative you could try 2 or 3 of these wired in parallel or perhaps one of these.
- It also used this Adafruit charger . One of the nice features of this charger is that it provides "status" pins to indicate when it is charging, and when the battery is fully charged. There is a #define `CHG_STATUS` in the code to read these status pins and send this information to the IOT sites. This gives a good indication of what's going power-wise. Without a battery status it will take awhile before you know that your system is self sustaining.
- An (old!) 2700mAH LiPO was used. Several 18650 LiPO batteries in parallel may be a better idea.
- This weather-proof enclosure was used to provide plenty of room for kit, SBM-20 GM tube, charger and battery. The components were mounted on a base that sits inside. The GK-RadMon and GM tube were mounted with standoffs in case of condensation. The solar panel is attached to the enclosure with strong magnets.
- An I2C cable was run outside the enclosure to connect to the HDC1008 temp / humidity sensor. The cable also has a header on it that allows the OLED be plugged in without opening the case. Gotta love that I2C.
- A power switch on the box comes in handy.
- If you have trouble with WiFi signal strength you could try the ESP-07 variant which supports an external antenna. For best results, you should remove the 0Ω resistor next to the connector to disconnect the chip antenna. Another alternative may be to use a WiFi range extender. (example only)

Conditions:

- The unit was located outside ~30 feet (10M) from the house in the sunniest spot. The WiFi router is in the center of the house.
- The average RSSI at this location with the ESP-12E is -84. With an ESP-07 and external antenna it was -75.
- Tests were done in Denver in December (!). *The initial charge on the LiPO was unknown*.

Results:

- Day 1: The "charging" status was on for 7 hours. The "charged" status never came on.
- Day 2: Again, "charging" for 7 hours, but at the very end the "charged" status came on. Eureka!
- Day 3+: After ~3 hours of charging, the battery status shows fully charged and remains so for another 3 hours until the sun was lost. The system looks to be self sustaining at this point. See below.



Build Pictures: