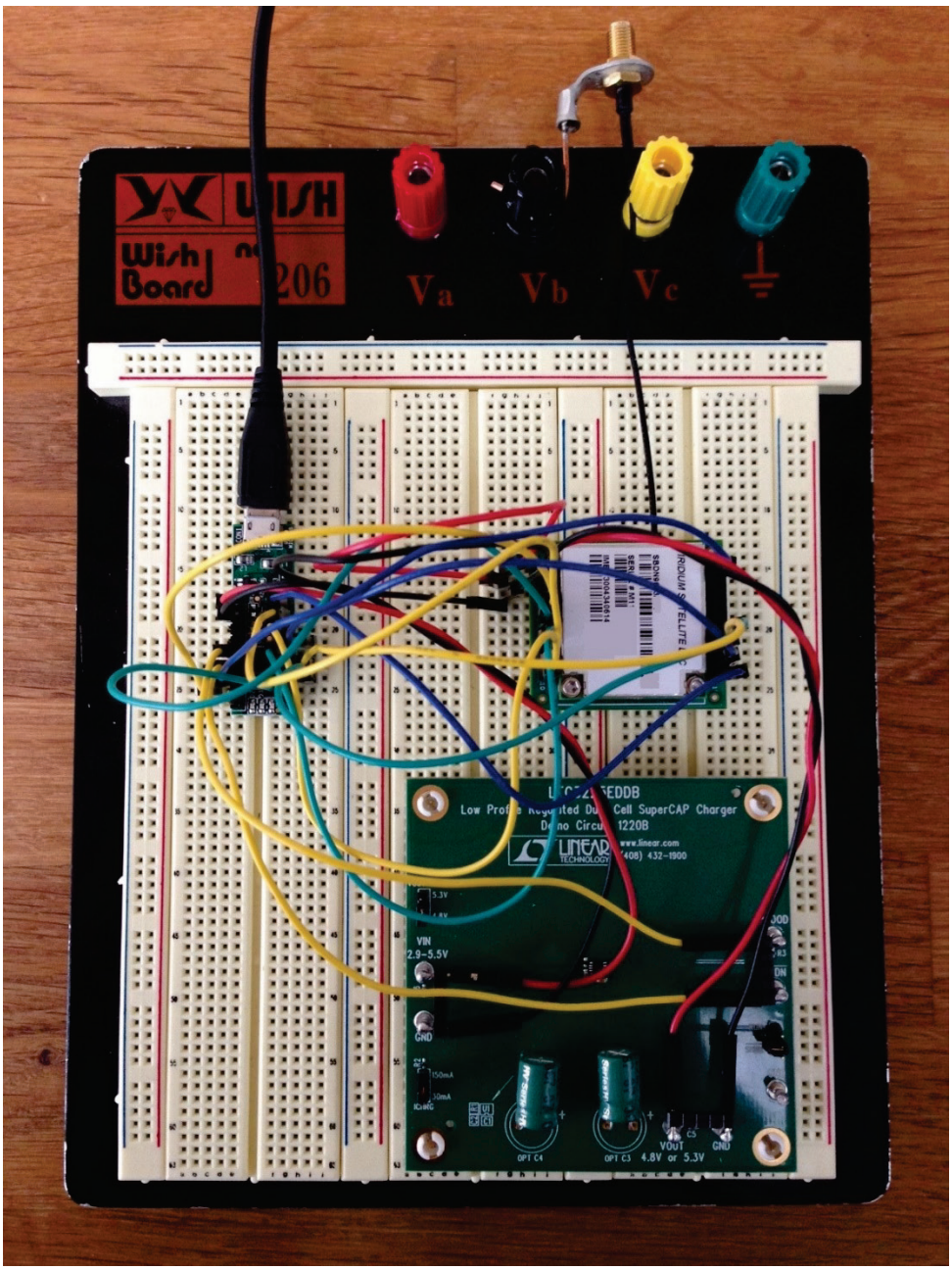
Iridium 9603 Lite USB Prototype



It ain't pretty. But it works!

Background:

Following on from the work I did on the Iridium 9603 Beacon,

https://github.com/PaulZC/Iridium_9603_Beacon

which got some nice coverage on Hackaday,

http://hackaday.com/2016/12/19/a-beacon-suitable-for-tracking-santas-sleigh/

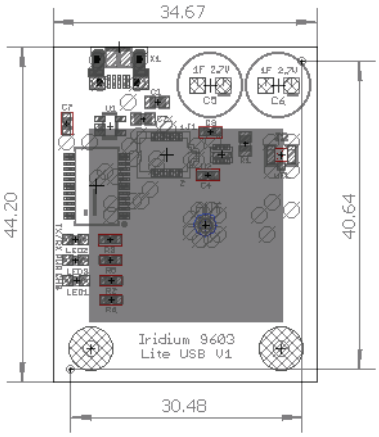
I decided to set myself the challenge of designing a really small, lightweight unit that could be used to send messages via the Iridium satellite network from high altitude balloons and other remote locations. You can already buy the fantastic RockBLOCK Mk2 from Rock7 Mobile , which is also available from Sparkfun, but what if you wanted something even smaller and lighter…?

Inspiration came in the form of the new SparkFun USB UART Serial Breakout which features the Cypress CY7C65213 USB to UART Bridge. This chip makes all eight serial port lines available (TXD, RXD, RTS, CTS, DSR, DTR, DCD and RI) plus eight general purpose input/output (GPIO) pins that can be used to control other functions. In the Iridium 9603 Lite USB, I use it to: provide a full USB to UART interface for the Iridium 9603 (including the Ring Indicator which can let you know a “mobile terminated” message is waiting to be collected); provide power for the 9603 via an LTC3225EDDB supercapacitor charger direct from the USB port; and use the GPIO lines for housekeeping.

I like to use Python where I can and was delighted to find that Taisuke Yamada had written a library called python-ucdev with which I was able to control the CY7C65213 GPIO pins (in manufacturing mode).

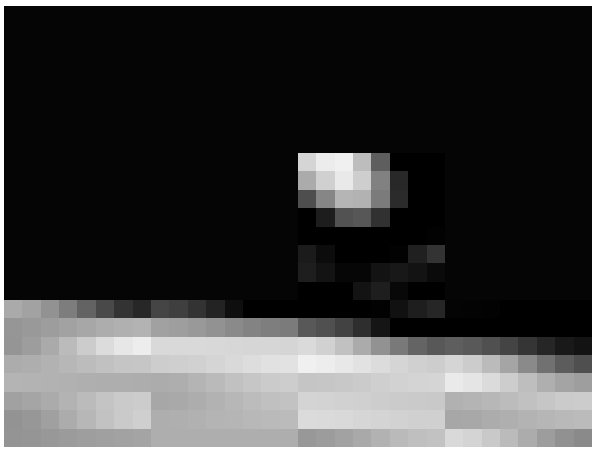
The current status is: the prototype works! It has been tested with Python 2.7 code running on Windows 10 (64-Bit) and on Debian Linux (Raspberry Pi). Have a look at Iridium_9603_Lite_USB.py elsewhere in this repo.

The next stage is to condense everything onto a small PCB. I have an Eagle design that is almost ready for manufacture, but it does need to be tested to see if a two layer board works or whether four layers are required (to provide a continuous ground plane under the Taoglas Iridium patch antenna). Either way, the final assembly will be approximately 35mm x 44mm x 18mm and weigh close to 33g (including the antenna).



After that, the plan is to: test it with Raspberry Pi; and – if time permits – write an Android app so you could send and receive short messages on your phone in areas with no mobile signal. Or, it could be connected to a Raspberry Pi Zero with Pi Camera, run from a battery, and used to send *very* compressed images from the edge of space (340 bytes is the limit for a *single* "mobile originated" message). The possibilities are endless!

From this….



To this…. (335 bytes!)

Image Credit: NASA

The Design:

The key components of the Iridium 9603 Lite USB Prototype are:

- Cypress CY7C65213
  - For breadboard prototyping, you can use:
    - https://www.sparkfun.com/products/13830
  - Or
    - http://www.cypress.com/documentation/development-kitsboards/cyusbs232-usb-uart-lp-reference-design-kit
    - Available from Mouser (Part# 727-CYUSBS232)
- Iridium 9603 Module
  - Available (in the UK) from e.g.:
  - http://www.ast-systems.co.uk/Product-Pages/Iridium-9603-SBD—Satellite-Tracking-Transceiver.aspx
  - http://www.rock7mobile.com/products-iridium-sbd
  - For breadboard prototyping, you can mount the module on a Satelligent 9603-DIP: http://satelligent.ca/products/accessories/9603-dip/
- Linear Technology LTC3225EDDB SuperCapacitor Charger
  - http://www.linear.com/product/LTC3225
  - For breadboard prototyping, you can use Linear's demo circuit (1220B):
    - http://www.linear.com/solutions/2187
  - Available as a bare chip from e.g. Farnell / Element14 (1715231)
  - Charges two e.g. Bussmann HV0810-2R7105-R 1F 2.7V capacitors (Farnell / Element14 2148482)
- The final Iridium 9603 Lite USB will use a:
  - Taoglas IP.1621.25.4.A.021 Iridium Patch Antenna
    - Available from e.g. Mouser (960-IP1621254A02)
- But for prototyping, you can connect any Iridium antenna using a readily available uFL to SMA adaptor (e.g. part number 7942894 from RS Components).
- For fixed or vehicle-mounted applications, you can use a larger dual-band Iridium+GPS antenna. E.g.:
  - https://www.beamcommunications.com/products/70-iridium-beam-whip-dual-mode-antenna

OK. I want to build one. What do I do first?

The first step is to configure the CY7C65213. Start by reading Sparkfun's hookup guide:

- https://learn.sparkfun.com/tutorials/sparkfun-usb-uart-breakout-cy7c65213-hookup-guide

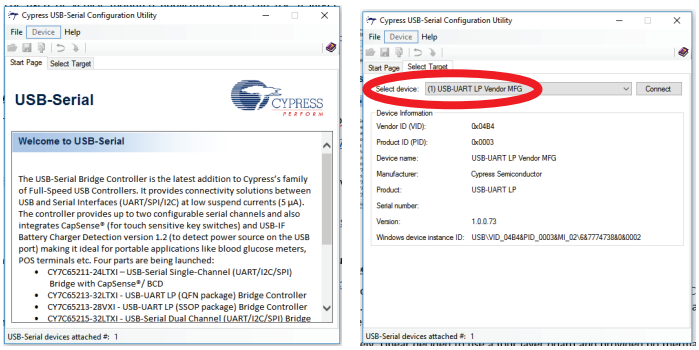
You will also want to download and install Cypress' USB-Serial Software Development Kit:

- http://www.cypress.com/documentation/software-and-drivers/usb-serial-software-development-kit

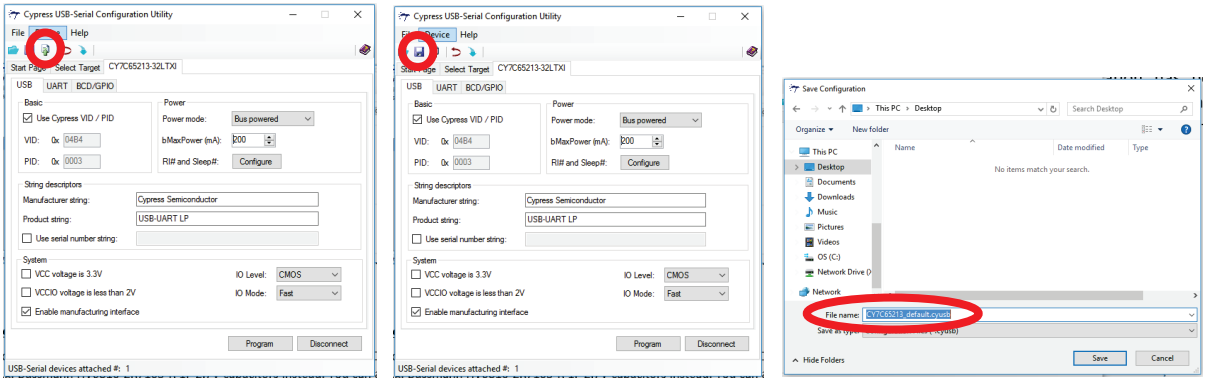
You might find Cypress' CY7C65213 Reference Design documentation useful too:

- http://www.cypress.com/documentation/development-kitsboards/cyusbs232-usb-uart-lp-reference-design-kit

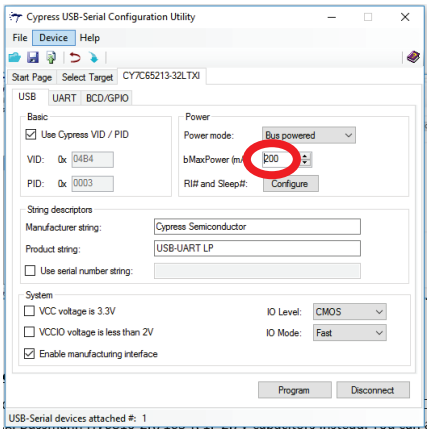
Plug in your board, run the USB Serial Configuration Utility and check that it detects the CY7C65213:



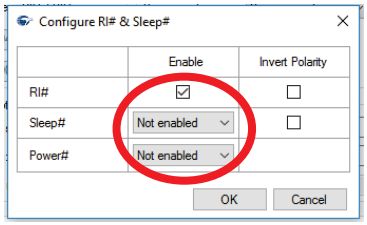
Select the "USB-UART LP Vendor MFG" and connect to it. Click the "Open Configuration from Device" icon to ensure the default device configuration has been read successfully. Click the Save icon and save the default configuration using a filename like "CY7C65213_Default.cyusb". That way, you can restore the default configuration if you need to.



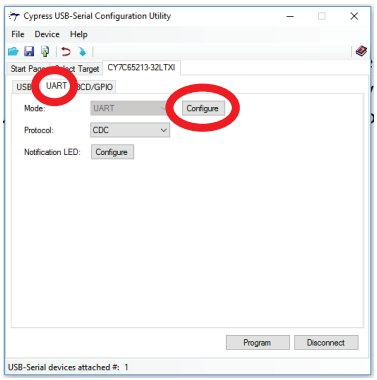
Set “bMaxPower (mA) to 200”:



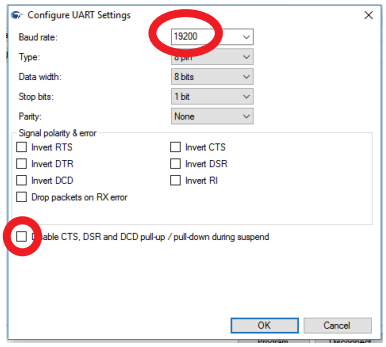
Click the Configure button next to “RI# and Sleep#”. Leave RI enabled. Set “Sleep#” and “Power#” to “Not enabled”, then click OK. (You could leave Sleep# set to GPIO 04 if you want the USB host to be able to put the supercapacitor charger to sleep automatically, but I haven’t tested this as yet.)



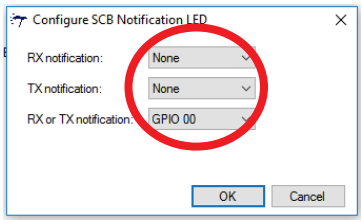
Select the UART tab and then click the Configure button next to “Mode”:



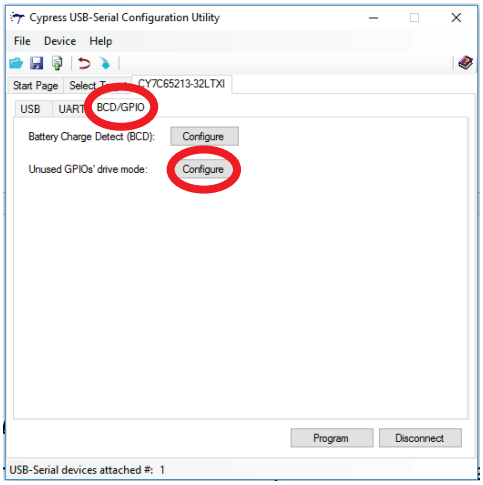
Set the Baud rate to 19200, untick the “Disable CTS…” box and then click OK.



Click the Configure button next to “Notification LED”. Set “RX notification” and “TX notification” to “None”. Set “RX or TX notification” to “GPIO 00”. Then click OK.

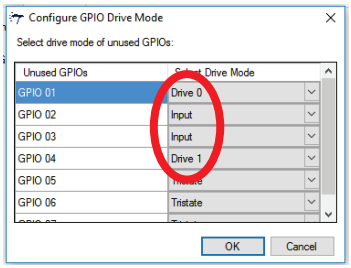


Select the BCD/GPIO tab and then click the Configure button next to “Unused GPIOs’ drive mode”:

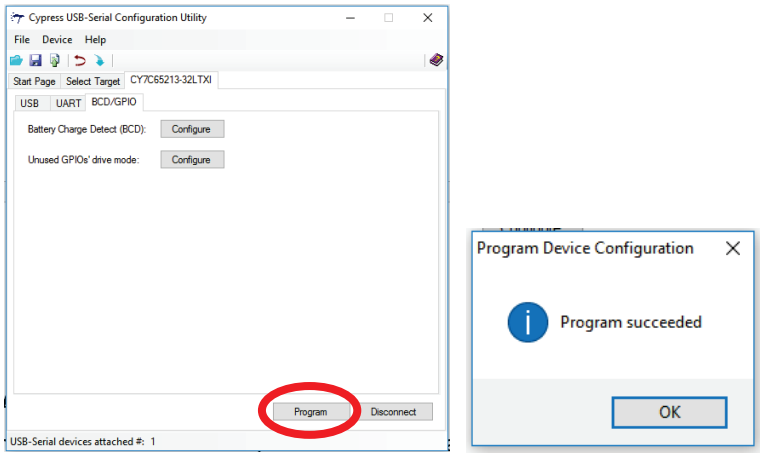


Select the following drive modes then click OK:

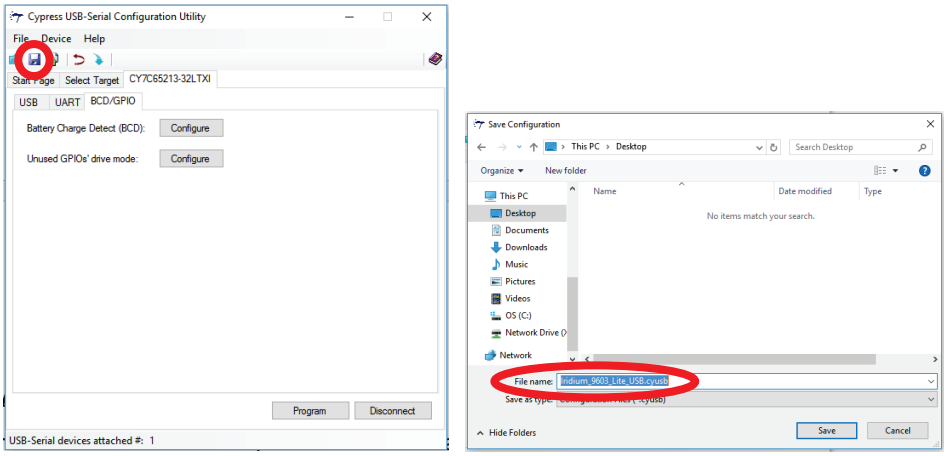
- GPIO 01: “Drive 0” (this will turn the 9603 off by default)
- GPIO 02: “Input” (for the supercapacitor charger PGOOD signal)
- GPIO 03: “Input” (for the Iridium 9603 Network Available signal)
- GPIO 04: “Drive 1” (this will power up the supercapacitor charger by default)
    - (GPIO 04 will be missing from the list if it is allocated to Sleep#)



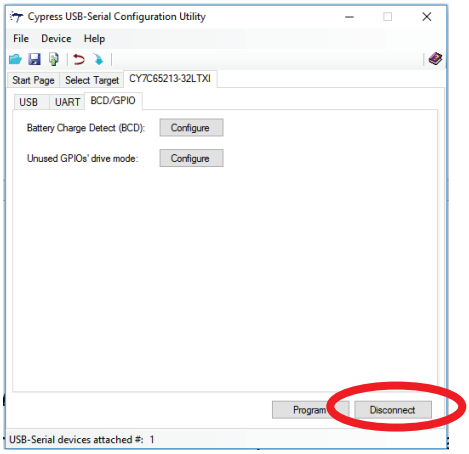
Click “Program” to program these settings into the CY7C65213. Check that the programming is successful.



Save the configuration into a new .cyusb file called something like Iridium_9603_Lite_USB.cyusb.



Finally, click “Disconnect” and unplug the CY7C65213. It will use the new configuration the next time it is powered up.



Do I need to modify the LTC3225EDDB Demo Circuit?

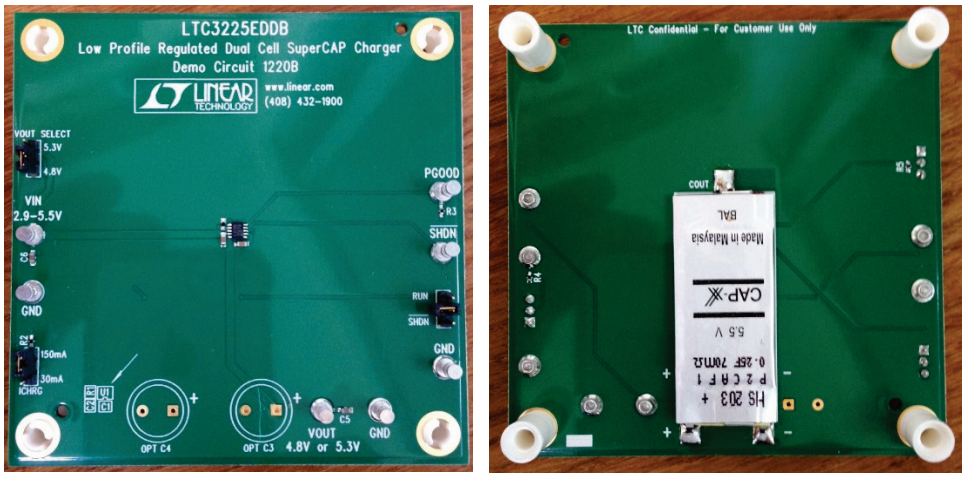
Yes. You'll need to remove the slim 0.25F capacitor from the back of the PCB and add two e.g. Bussmann HV0810-2R7105-R 1F 2.7V capacitors instead. You can also solder pin headers onto the terminals to make it easy to use hook-up wires.

Bizarrely, Linear decided to use a four layer board and provided no thermal relief on the C4 –ve pin. This makes it almost impossible to solder. I had to leave the capacitor leg long, bend it over and solder it to the –ve pad for the original slim capacitor.

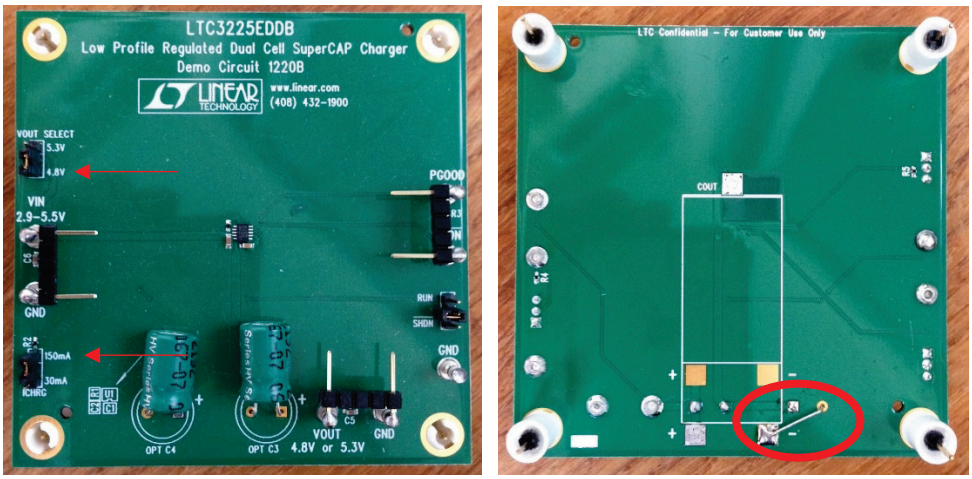
Make sure that:

- The ICHRG jumper link is set to 150mA
- The VOUT_SELECT jumper link is set to 4.8V
- The RUN/SHDN jumper link is removed

From this:

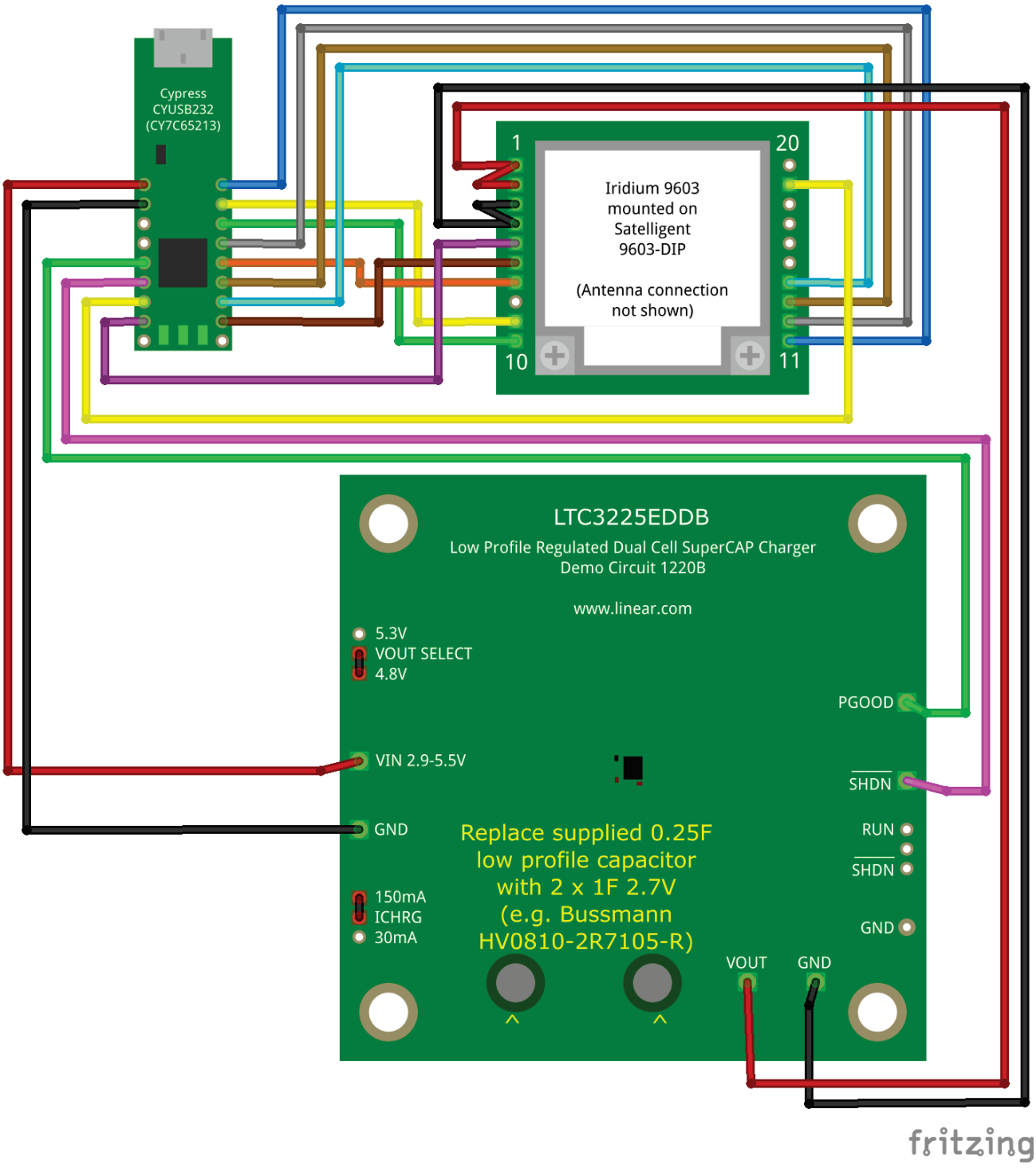


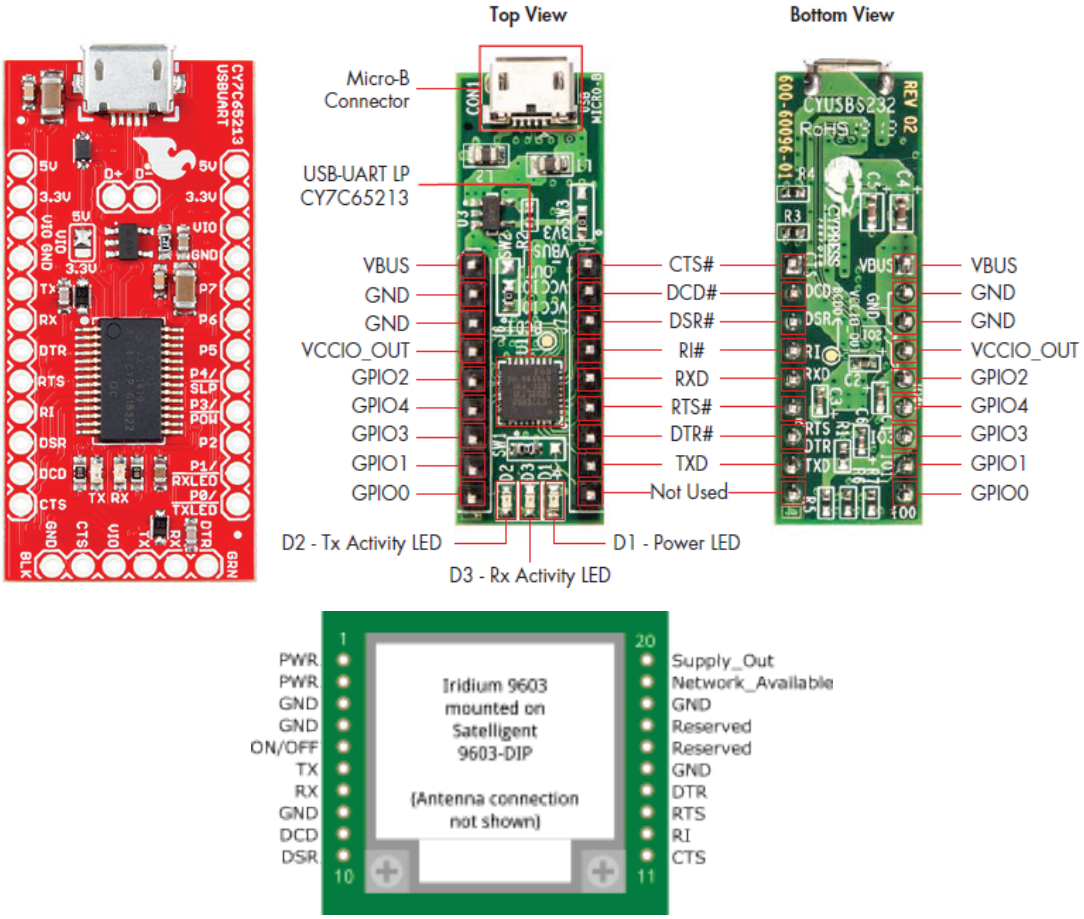
To this:



How do I connect everything?

Here's a Fritzing diagram to help:





Connect:

- CY7C65213 5V/VBUS to LTC3225EDDB VIN
- CY7C65213 GND to LTC3225EDDB GND
- CY7C65213 P2/GPIO2 to LTC3225EDDB PGOOD
- CY7C65213 P4/GPIO4 to LTC3225EDDB SHDN
- CY7C65213 P3/GPIO3 to Iridium 9603 Network Available (Pin 19)
- CY7C65213 P1/GPIO1 to Iridium 9603 On/Off (Pin 5)
- CY7C65213 TX/TXD to Iridium 9603 TX (Pin 6)
- CY7C65213 DTR to Iridium 9603 DTR (Pin 14)
- CY7C65213 RTS to Iridium 9603 RTS (Pin 13)
- CY7C65213 RX/RXD to Iridium 9603 RX (Pin 7)
- CY7C65213 RI to Iridium 9603 RI (Pin 12)
- CY7C65213 DSR to Iridium 9603 DSR (Pin 10)
- CY7C65213 DCD to Iridium 9603 DCD (Pin 9)
- CY7C65213 CTS to Iridium 9603 CTS (Pin 11)
- LTC3225EDDB VOUT to Iridium 9603 PWR (Pins 1&2)
- LTC3225EDDB GND to Iridium 9603 GND (Pins 3&4)

OK. I've got it connected up. Where's the code?

You will find Iridium_9603_Lite_USB.py in this repo.

You will also need to clone or download:

- https://github.com/PaulZC/python-ucdev
- Run "python setup.py install" to install the library, or make sure the cy7c65213 sub-directory is copied into the same directory as Iridium_9603_Lite_USB.py

If you haven't installed it before, you will also need CFFI:

- https://pypi.python.org/pypi/cffi?
- http://cffi.readthedocs.io/en/latest/

The code won't work without Cypress' cyusbserial.dll (on Windows machines):

- The dll gets installed as part of the SDK (see above)
- You will usually find the 64-Bit Windows version in:
  - C:\Program Files (x86)\Cypress\USB-Serial SDK\library\cyusbserial\x64
- The 32-Bit Windows version is usually found in:
  - C:\Program Files (x86)\Cypress\USB-Serial SDK\library\cyusbserial\x86

There are instructions in the python-ucdev repo to show you how to get the code up and running under Linux (Debian on Raspberry Pi).

How are the GPIO pins used?

- GPIO0 will flash the "TX" LED whenever data is transmitted or received over the USB port
- GPIO4 is used to enable or disable the LTC3225EDDB supercapacitor charger
  - Hold GPIO4 high to enable the LTC3225EDDB
  - Low puts the LTC3225EDDB into Shutdown
- GPIO2 is connected to the LTC3225EDDB PGOOD signal
  - High indicates that the power is good
  - Low indicates that the capacitors are still charging
- GPIO1 is used to turn the Iridium 9603 on (high) or off (low)
  - At power-up, hold GPIO1 low until GPIO2 (PGOOD) goes high
- GPIO3 is connected to the Iridium 9603 Network Available signal
  - This pin can be monitored to check whether the 9603 can see an available satellite network
  - High indicates that the network is available
  - Monitoring this pin is equivalent to issuing an AT+CSQ command

Are there any tricks I need to know?

The main one is that the Iridium 9603 will refuse to communicate unless both the RTS and DTR pins are pulled low. Using pyserial, you need to set both pins to "1" to produce a low output:

- set_RTS(1)
- set_DTR(1)

What other resources do you recommend?

Here are a few useful links:

- http://www.makersnake.com/rockblock/
- https://cdn.sparkfun.com/datasheets/Wireless/General/IRDM_ISU_ATCommandReferenceMAN0009_Rev2.0_ATCOMM_Oct2012.pdf
- https://www.networkinv.com/wp-content/uploads/2012/08/Iridium-9603-Developers-Guide-v2-1.pdf

Will the Iridium 9603 Lite USB really weigh 33g?

Yes, I think so. Here is how the mass estimate breaks down:

• PCB (estimate):	8.3g
• Iridium 9603 Module:	11.6g
• Iridium Antenna:	9.3g
• Module Fixings:	1.7g
• Super Capacitors:	1.7g
• uFL Cable:	0.4g
• Total:	33g

Why do you need the Super Capacitors?

The Iridium 9603 module draws a peak current of 1.3A when transmitting its short data bursts. That's too much for a standard USB port. The LTC3225 super capacitor charger draws 150mA from the USB port to charge two 1F 2.7V capacitors, connected in series, to 4.8V. The capacitors then deliver the 1.3A to the module when it sends the data burst.

What data can I send and receive?

"Mobile Originated" messages – messages sent from the 9603 – can be up to 340 bytes and sent as text or binary data.

"Mobile Terminated" messages – messages sent to the 9603 – can be up to 270 bytes.

You can receive MO data as an email attachment from the Iridium system. The email itself contains extra useful information:

- Message sequence numbers (so you can identify if any messages have been missed)
- The time and date the message session was processed by the Iridium system
- The status of the message session (was it successful or was the data corrupt)
- The size of the message in bytes
- The approximate latitude and longitude the message was sent from
- The approximate error radius of the transmitter's location

E.g.:

From: sbdservice@sbd.iridium.com
Sent: 20 August 2016 16:25
To:
*Subject: SBD Msg From Unit: 30043406174*****
*Attachments: 30043406174****_000029.sbd*

MOMSN: 29
MTMSN: 0
Time of Session (UTC): Sat Aug 20 15:24:57 2016 Session Status: 00 - Transfer OK Message Size (bytes): 61

Unit Location: Lat = 55.87465 Long = -2.37135 CEPradius = 4

Thanks!

This project wouldn't have been possible without the open source designs and code kindly provided by:

- **Sparkfun**:
  - The USB UART Serial Breakout CY7C65213 (Product BOB-13830)
  - https://www.sparkfun.com/products/13830
- Taisuke Yamada (Tai)
  - Python library for Cypress CY7C65211/CY7C65215 USB-Serial bridge
  - https://github.com/tai/python-ucdev

The small print