Atmospheric Science Payload Electronics Manual for the NASA GSFC Balloon Program

Joseph M Galante – joseph.m.galante@nasa.gov

This manual details a simple balloon payload that uses an Arduino board to gather temperature, barometric pressure, relative humidity, and location data from a variety of sensors. The sensor data is handed off to a memory card device for later analysis. A rechargeable LiPo battery powers the entire electronic system. The complete electronic system should look like the following diagram, but don't worry! This manual will take you through the assembly step by step.

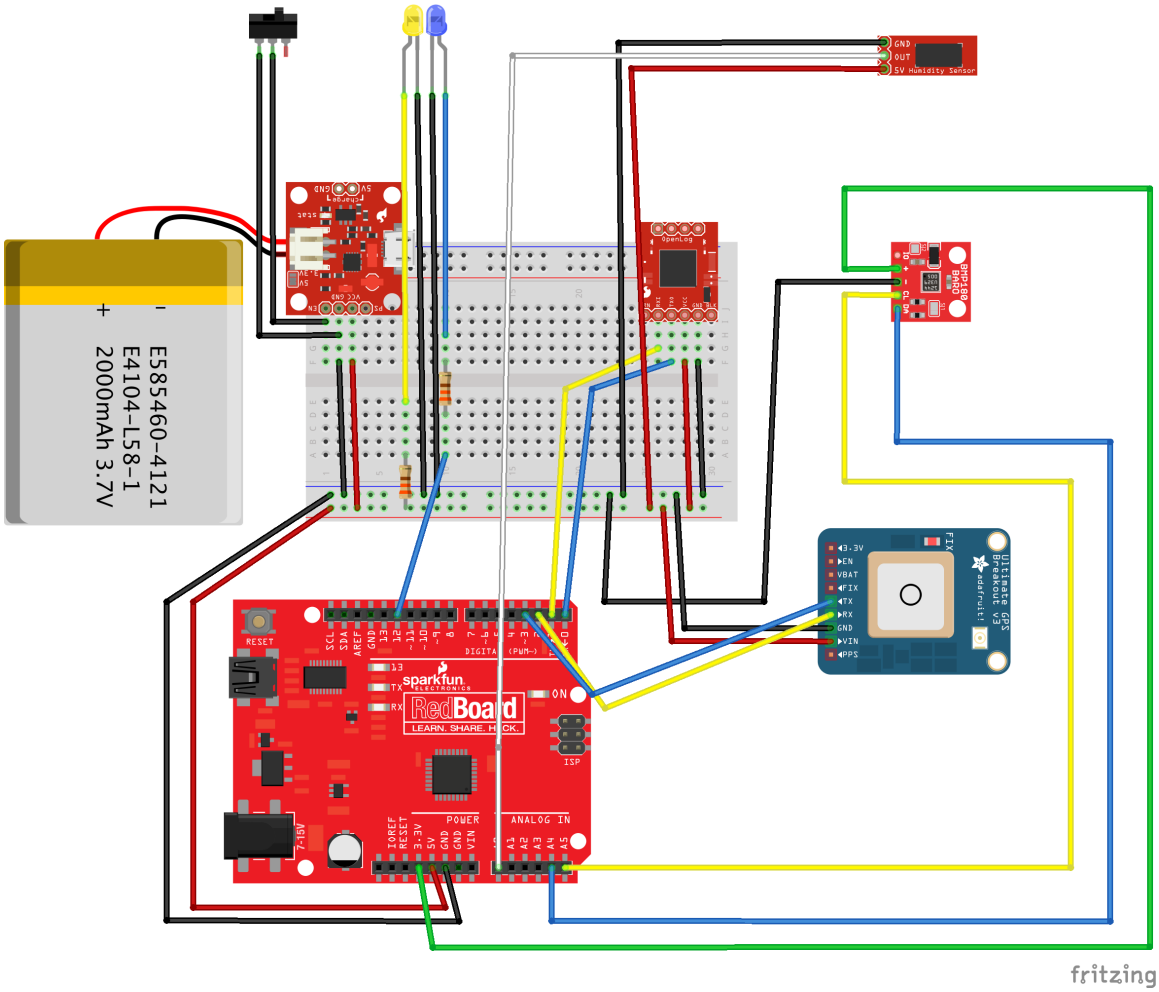


Table of Contents

Parts List

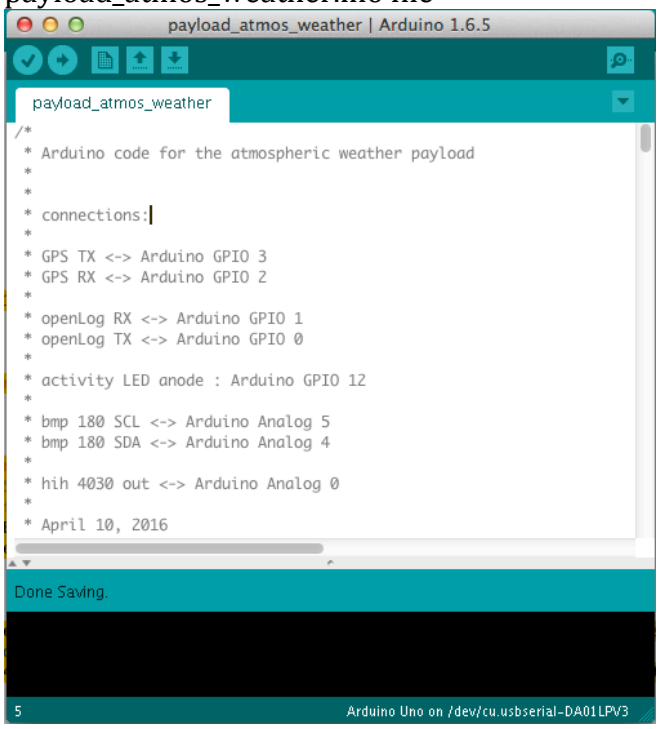
Breadboard
Arduino Uno (specifically, a Sparkfun RedBoard)
2000 mAh LIPO battery
powercell circuit
openLog SD card logger
SD card
Ultimate GPS sensor
BMP-180 barometric pressure sensor
HIH-4030 relative humidity sensor
Blue LED
Yellow LED
Switch
330 ohm resistors (orange orange brown gold color marking)
wires

Load Arduino Code onto Arduino Board

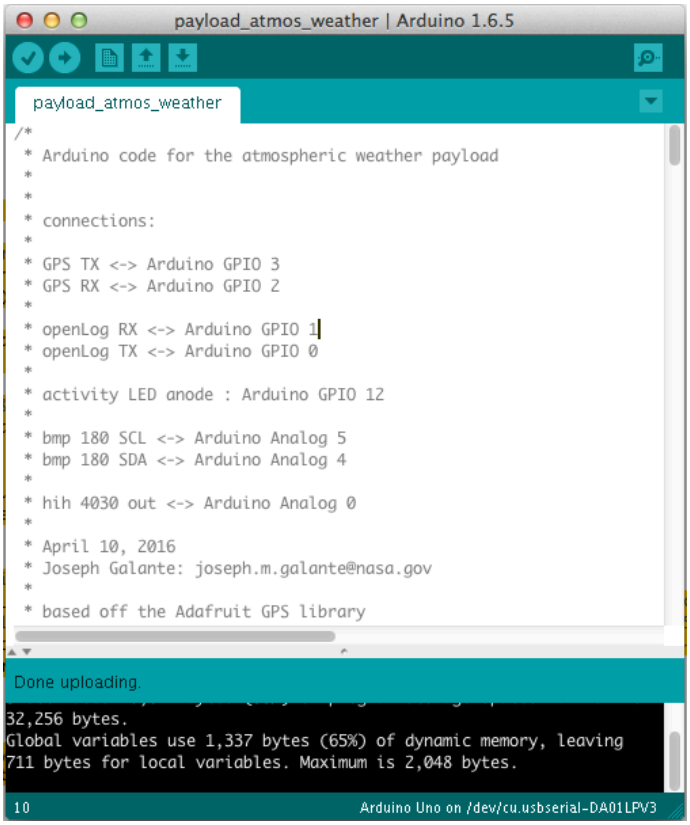
Download payload_atmos_weather.ino from BlackBoard or
https://github.com/xkcdFan1011011101111/balloons/tree/master/arduino_code

Connect your Arduino Board to your computer via USB cable – the ON LED should be illuminated green

Open the Arduino Software Application on your computer, open the payload_atmos_weather.ino file



Upload payload_atmos_weather.ino to your Arduino Board – the RX and TX LEDs should flash during upload; once upload is complete, the RX and TX LEDs should be off and the Arduino Software Application should state “Done Uploading”

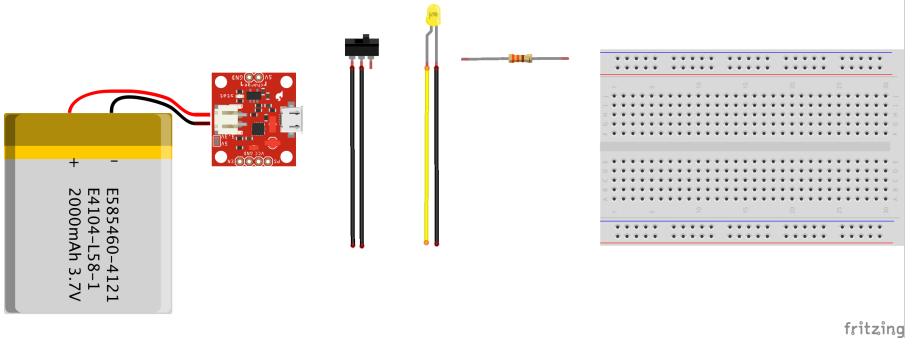


Disconnect your Arduino Board from your computer

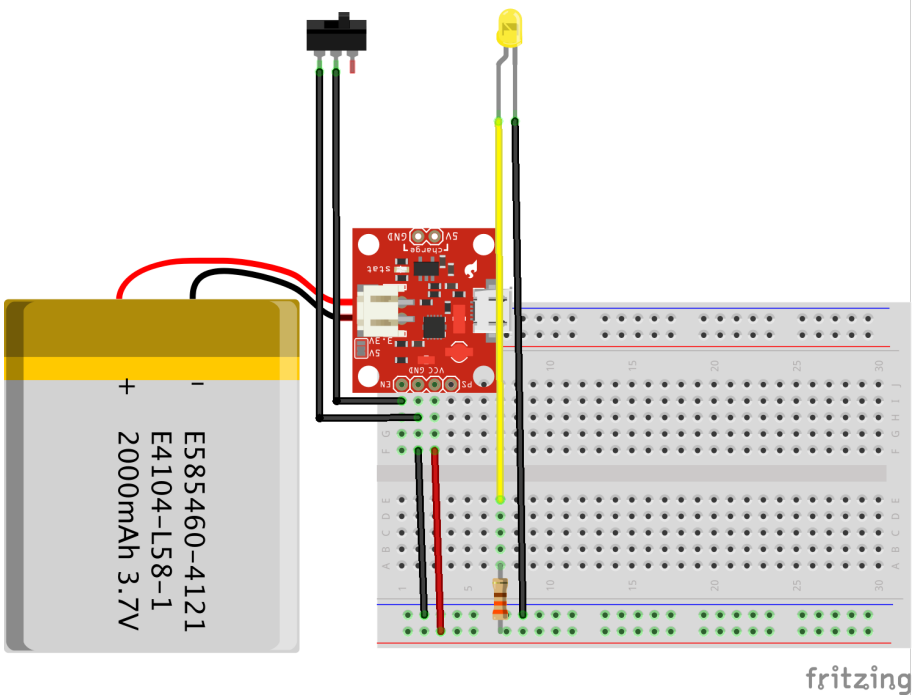
Set aside your Arduino Board for now, we'll get back to it later

Build Power Subsystem

Parts needed for this step, from left to right: 2000 mAh battery, powercell, switch, yellow LED, 330 ohm resistor (orange orange brown gold), breadboard



Assemble the circuit as shown below:



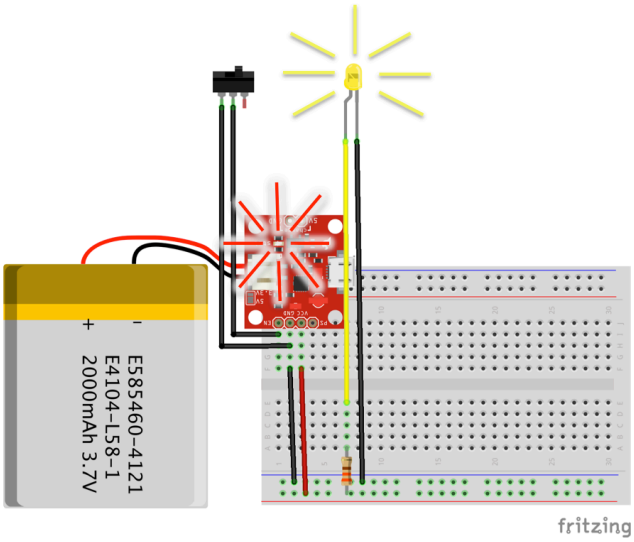
Charge the Battery

Now charge your battery using a microUSB cable. You'll need to plug one end of the cable to your laptop computer or wall outlet adapter; plug the other end into the powercell.

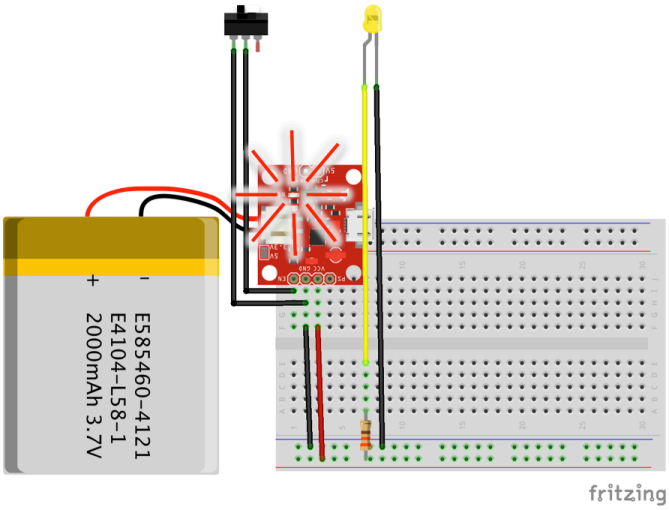
The red LED on the powercell (labeled stat) will illuminate when the powercell receives external power.

The yellow LED is an indicator LED to inform us when our circuit is energized. We can choose to either power our system (not yet connected except for the yellow LED) using the battery or to charge the battery using the switch.

Switch knob in right position, yellow indicator LED illuminated, battery will not charge!



Switch knob in left position, yellow indicator LED not illuminated, battery will charge (assuming powercell is connected via USB cable to a power source)



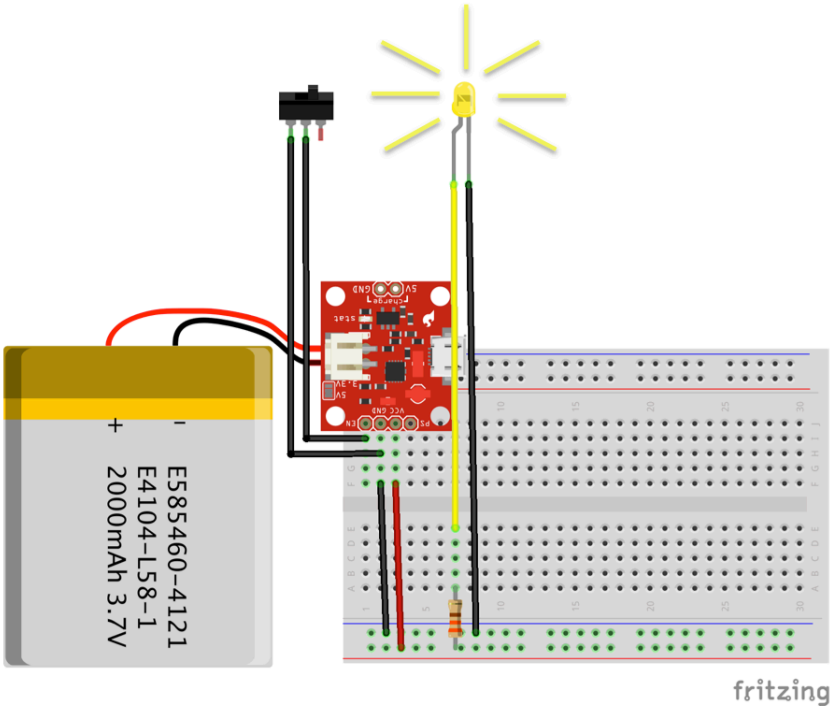
The red LED turns off once the battery is fully charged. If you don't want to wait, you can partially charge the battery (charge for at least 10 minutes).

Test the System

Now that you have a completely (or partially) charged battery, it is time to test the system!

Disconnect the USB cable from the powercell. The red LED on the powercell, labeled stat, should definitely not be illuminated.

Flip the switch so the knob is over the unconnected switch wire. The yellow LED should light up. The system should look like this:



If this is working, hooray! Move along to the next step. If this isn't working, go back to the earlier steps and check your work carefully. Are all the wires connect in the right places? Is the battery sufficiently charged?

Prepare microSD Card

The openLog SD card logger will make data recording easy for us. In order to use it, we need to configure the openLog by leaving a configuration file on the SD card. openLog knows to look for a file with a special name. We'll set up the configuration file so the openLog listens at the correct data rate and in the correct mode to store our data.

Load the microSD card into your computer.

Make sure there is a plain text file on the microSD card named CONFIG.TXT. If the file is missing, create one or download a copy from BlackBoard or **https://github.com/xkcdFan1011011101111/balloons** . The name is case sensitive; config.txt will not work! Also, the file must have a .txt suffix, neither CONFIG.rtf nor CONFIG.doc will work.

The contents of the first line of the file should be:

```
115200,26,3,0,1,1,0
```

openLog will ignore anything else in the file. I often write out the meaning of each number in the line below for reference:

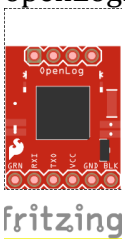
```
115200,26,3,0,1,1,0
baud,escape,esc#,mode,verb,echo,ignoreRX
```

For more information about the openLog, check out
https://github.com/sparkfun/OpenLog/wiki

Delete any existing files on the microSD card to make room for the data we hope to acquire (except for the config file!). Now eject the microSD card. We are ready to install the memory card in our system.

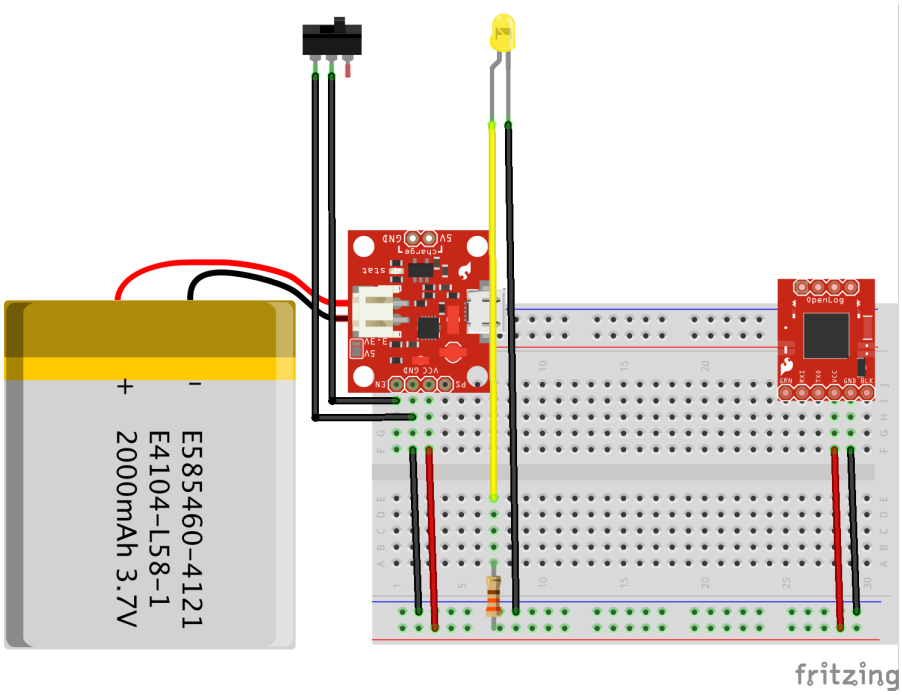
Build Memory Subsystem

Let's give our system some long term memory. Insert the microSD card into the openLog. The openLog is the part that looks like this:



Turn your system off by using the switch. Make sure the yellow LED is not illuminated whenever you add parts to the system.

Now plug the openLog into the breadboard and wire it up to the system power as shown below.



Test System

We just added a new part to our system, so we should test to make sure it is wired and behaving correctly.

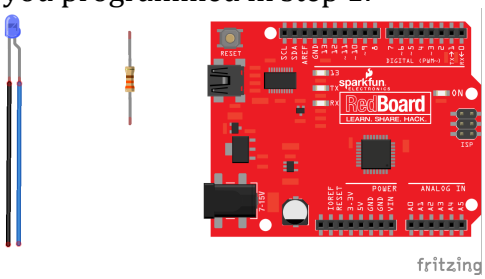
Your system should be off if you just finished the previous step.

Turn your system on by flipping the switch. As before, the yellow LED should illuminate indicating that the battery is powering your system. The openLog's green LED should briefly flash, followed by a longer flash from the openLog's blue LED, then the openLog should remain with all LEDs off. Feel free to turn the system off and back on again to repeat this step.

If your system passes those checks, turn the system off (make sure the yellow LED is off) and proceed.

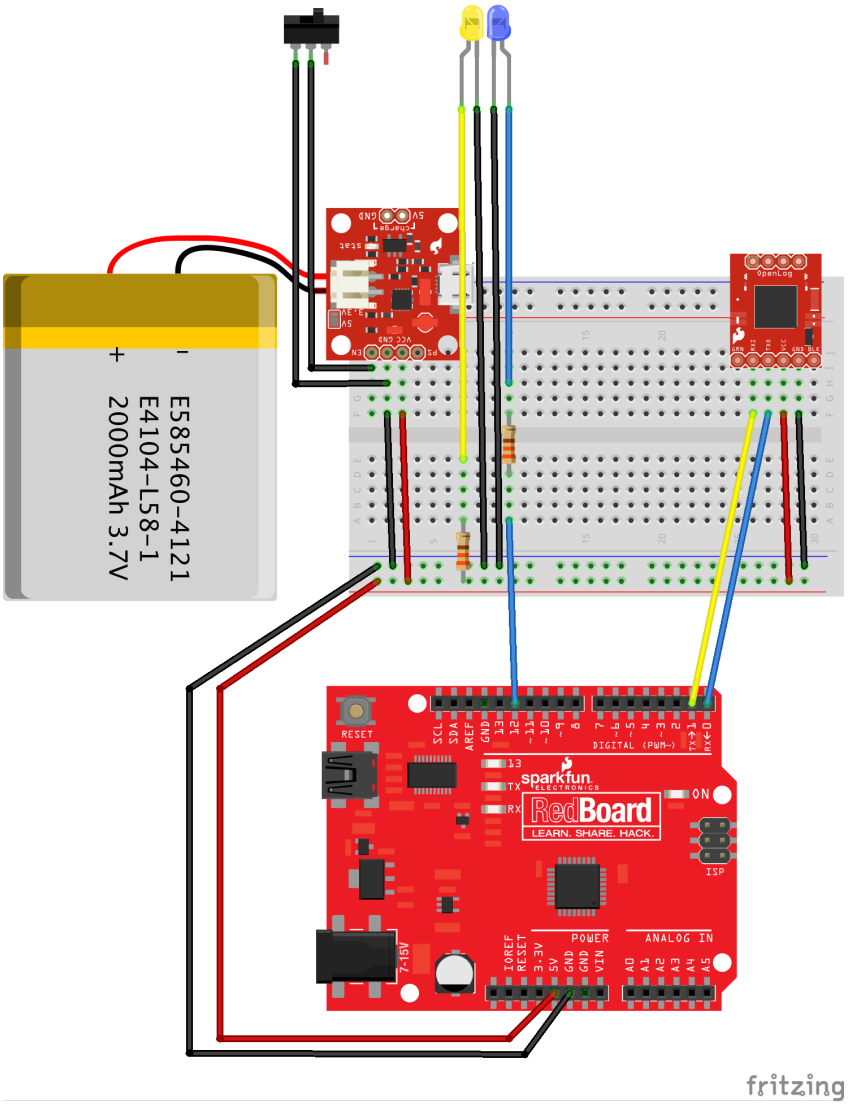
Connect the Arduino Board

Now we'll give the system some brains. For this step we'll need the blue LED, another 330 ohm resistor (orange orange brown gold), and the Arduino board that you programmed in step 1:



Turn your system off. Make sure the yellow LED is not illuminated whenever you add parts to the system.

Connect the Arduino board, blue LED, and resistor as shown:



Test System

We just added a major new part to the system, so let's test it out!

Flip on the system using the switch. The yellow LED should illuminate to indicate that the battery is powering your system. The green LED on the Arduino Board labeled on should illuminate and stay that way. The Arduino Board will also briefly flash a few of its other LEDs, but they should remain off after a few seconds.

The openLog should again briefly illuminate its green and blue LEDs as in step 7. Now, however, the Arduino Board is attached to the system. After a few seconds needed to initialize, the Arduino Board should speak to the openLog every 2 seconds. Whenever this happens, the blue LED on the openLog should briefly flash.

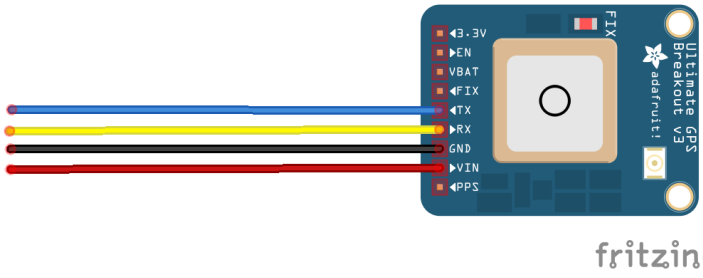
What is the Arduino Board saying to the openLog? Once our circuit is complete, the Arduino Board will report the GPS measurements to the openLog every two seconds. When the GPS fails to report measurements (for example, when the GPS unit isn't attached) the Arduino Board will instead send a bunch of zeros to the openLog. The openLog will dutifully record those zeros down. A new row of zeros is being added to a log file every time the openLog's blue LED flashes.

The large blue LED that you connected will be illuminated when the Arduino Board believes it is receiving legitimate GPS data. The GPS unit isn't connected yet, so the Arduino Board should think there is a problem. The Arduino should signal that it doesn't have legitimate GPS data by not illuminating the blue LED.

If your system passes those checks, turn the system off (make sure the yellow LED is off) and proceed.

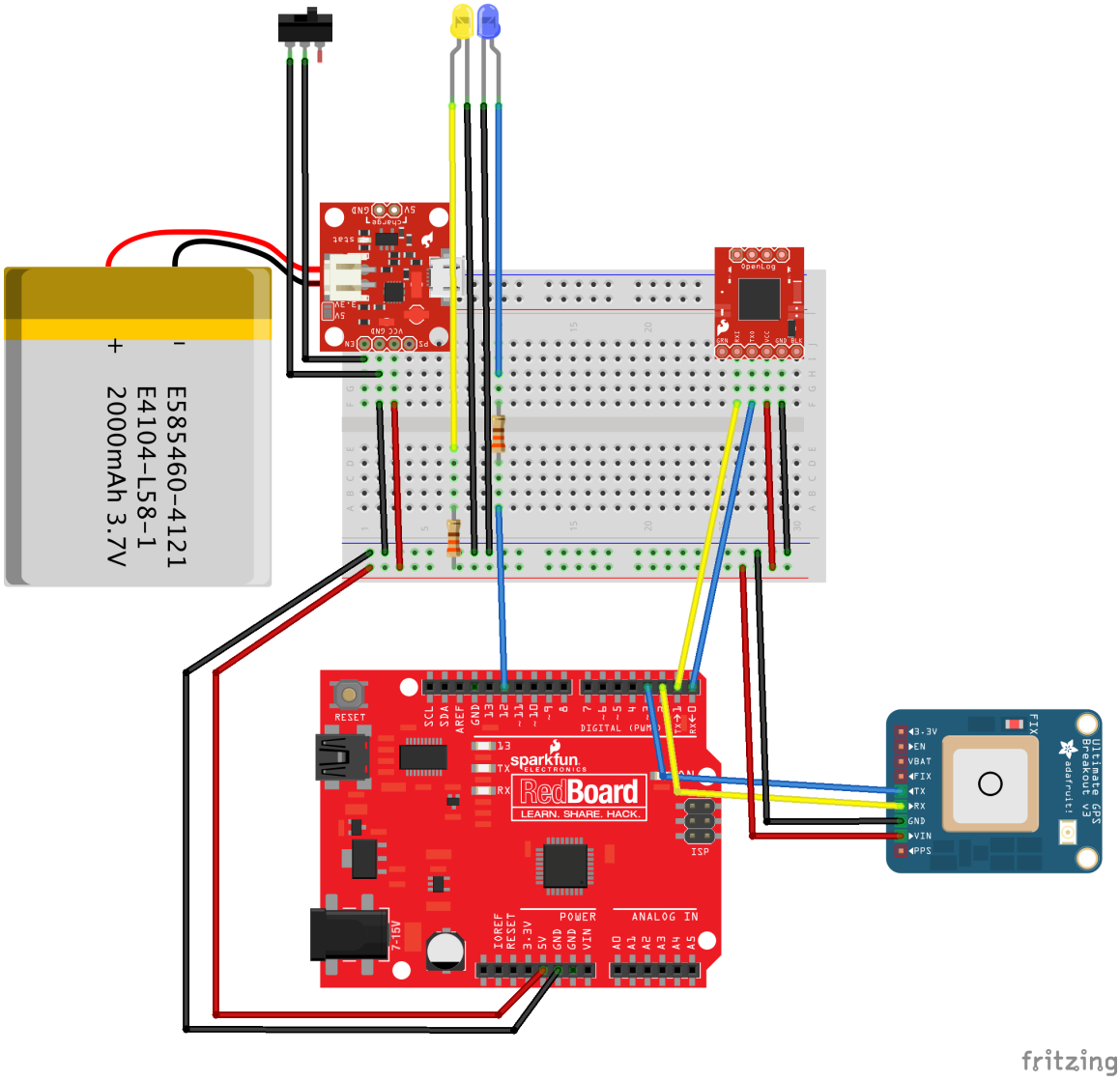
Connect GPS

Next, let's add the GPS unit to the system. The GPS unit should look like this:



Turn your system off. Make sure the yellow LED is not illuminated whenever you add parts to the system.

Connect the GPS as shown:



We're using an Ultimate GPS unit from Adafruit Industries. This GPS is very small and has a lot of features for just $40. For more information, check out the purchasing webpage: http://www.adafruit.com/products/746 and tutorial: https://learn.adafruit.com/adafruit-ultimate-gps .

Test System

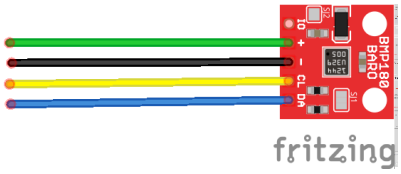
We just added a major new part to the system, so guess what time it is? Test time!

Flip on the system using the switch. The yellow LED should illuminate to indicate that the battery is powering your system. The green LED on the Arduino Board labeled on should illuminate and stay that way. The Arduino Board will also briefly flash a few of its other LEDs, but they should remain off after a few seconds.

The GPS unit has an LED labeled "FIX" that should blink red once per second. This indicator LED is telling us that the GPS doesn't have a fix, but that's okay for now so long as it is powered up correctly.

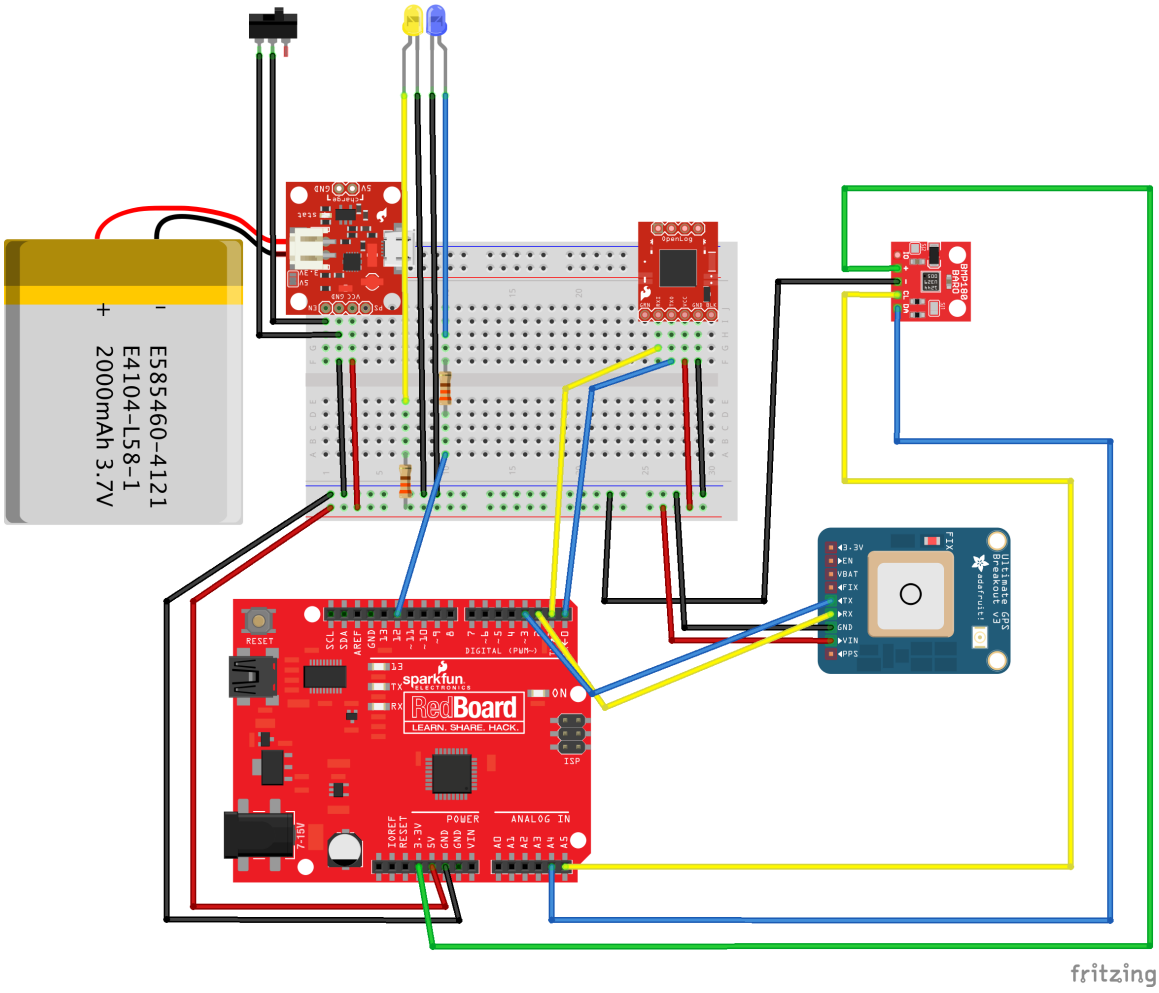
Connect Barometric Pressure Sensor

Now we'll add the barometric pressure sensor, the BMP-180, to the system. It should look like this:



Turn your system off. Make sure the yellow LED is not illuminated whenever you add parts to the system.

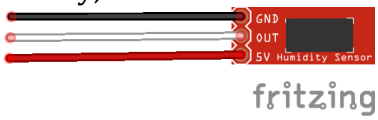
The pressure sensor should be connected like so. Note carefully that the **BMP-180's power wire must be plugged into the arduino's 3.3v supply**, not the 5v supply like all the other devices. The BMP-180's power wire is green to avoid confusion.



Connect Relative Humidity Sensor

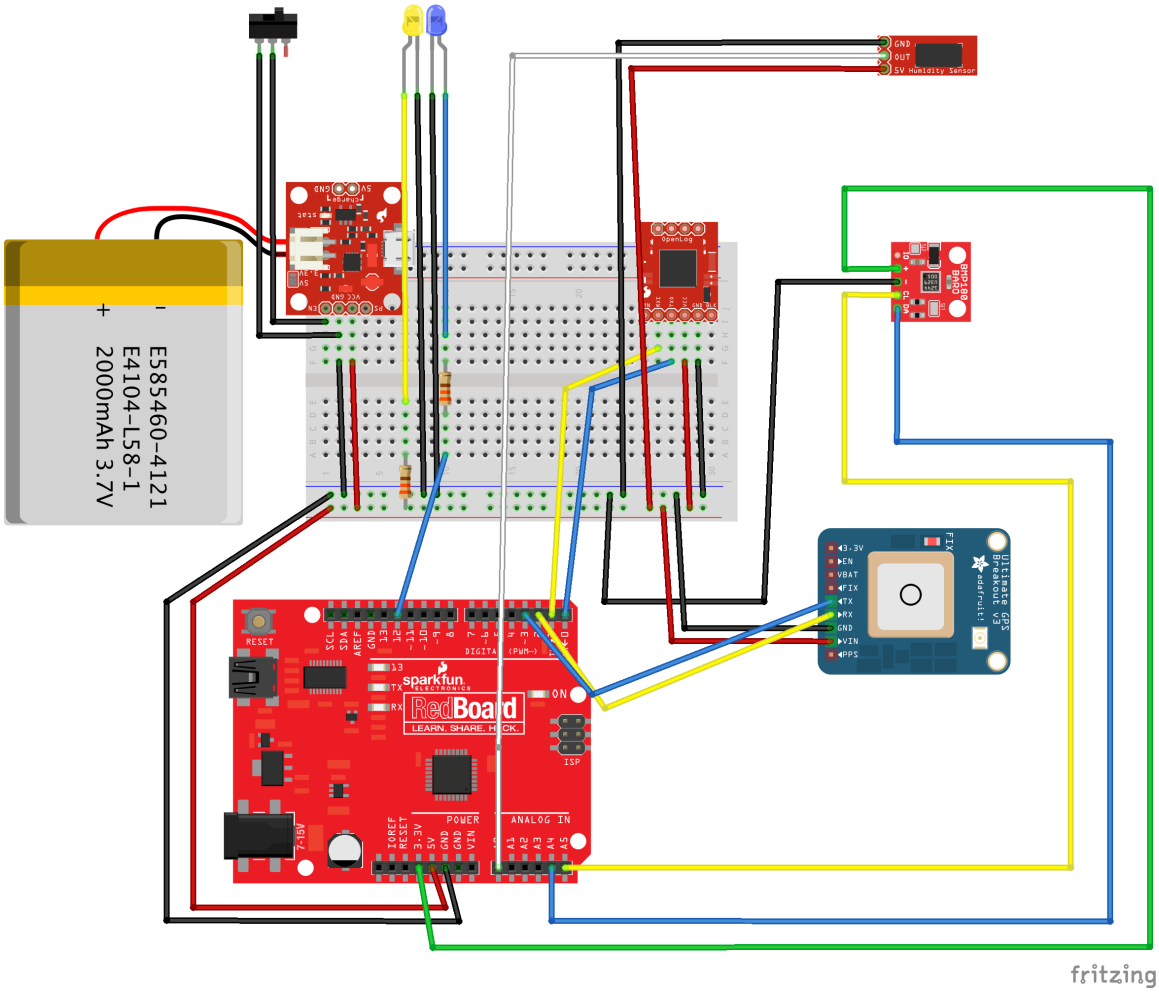
You might be expecting a step to test the system before moving on, but neither the BMP-180 nor HIH-4030 have LEDs to indicate status. Instead, we'll forge ahead with the last sensor.

Finally, let's add the relative humidity sensor, the HIH-4030. It should look like this:



Turn your system off. Make sure the yellow LED is not illuminated whenever you add parts to the system.

Connecting this sensor should be straightforward, just attach as indicated:



Test System

All right, the payload electronics system is complete! We're not done yet though, we first need to test to make sure everything is correctly connected.

Turn the system on.

The yellow LED should illuminate to indicate the battery is powering the system. The green LED on the Arduino Board should illuminate to indicate the Arduino Board is executing its software. The openLog's blue LED should flash every second to indicate that it is receiving data.

The blue LED still shouldn't illuminate, indicating the Arduino doesn't have any legitimate GPS data. But the GPS unit is connected now! What gives?

The GPS unit's red LED labeled FIX should blink every second. This indicates that the GPS unit is either still trying to get its initial fix, or it can't even see any satellites. The GPS unit needs to be outdoors with a direct view of the sky to receive signals from the GPS satellite constellation.

I recommend you turn off your system, go outside, and test again.

Every time you turn the system on, the openLog creates a new log file. The log file with the largest number in its name is the most recent log file.

Note that the first time the GPS unit tries to get a fix can take a looooooooong time. In the best of conditions, this GPS unit can get an initial fix in under 45 seconds; however, depending on your location, tall buildings nearby, local tree coverage, atmospheric conditions, and even solar flares can make this process take longer than 30 minutes! Joseph has typically seen initial fix times take 3-5 minutes with this particular GPS unit.

After the GPS unit achieves its first fix, it will provide location measurements at up to 10 Hz. Our system only logs once every 2 seconds, so we ignore most of the measurements. We don't expect our balloons to move very fast, so logging once every 2 seconds allows us to use cheaper electronics and keeps our log files smaller. If we put our GPS unit on a car or a UAV, we might want to make a faster logging system to utilize all the GPS measurements.

The good news is that our GPS units have batteries attached. Once the GPS unit has achieved its first fix, the information is stored in the GPS unit. This helps the GPS unit get an initial fix the next time that we power the system on.

Once the GPS unit has attained an initial fix, it should keep tracking. The GPS unit's red FIX LED should only flash once every 15 seconds. The blue LED connected to the Arduino Board's GPIO pin 12 should now remain illuminated.

Analyze Data

TODO: Example data and analysis to go here