Open Source GPS Tracking Collar

Written by Scott Ziv and Samuel Anderson
Advised by Meredith Keeley and Nicole Abaid, PhD.

Developed at Virginia Tech

Table of Contents

Overview	3
Ethical discussion	3
License	3
Contact us	3
Bill of Materials	4
Software Setup	5
Construction of Base Station	5
Construction of Nodes	
Software Setup	
Usage	
Commands	
Retrieving data	
Appendices	
Sample data	
Wiring diagram	

OVERVIEW:

This is a GPS tracking collar, originally designed for the purpose of tracking lemurs in Madagascar, but designed to be hackable, editable, and easily usable by other researchers for their various needs.

Much of the design is built upon Adafruit Industries feather boards and code, which means that parts are easy to source, instead of having to get a custom board made.

ETHICAL DISCUSSION AND DISCLAIMER:

This device was designed for a pressing need in conservation research: tracking small mammals, while still being adaptable, so that the scientist doing the research can add or remove additional sensors as necessary.

As with any invention, there are those that will misuse it. This system should not be used on humans without their express consent to be tracked. Respect the privacy of those around you, and while this system can be adapted to be used for nefarious purposes, it's up to the operator to use their own best judgment.

On that note, when attaching this to an animal, the operator must do right by that animal. A few design decisions for this system (mainly those concerning the battery), were put in place to prevent undue risk to the animal involved. While modification, adaptation, and sharing your adaptations to this system is allowed and encouraged, make sure the decisions that you make do not put the animal at risk, including but not limited to disabling safety scripts, over discharging protections, and battery shorts. If you are unsure if a modification will be safe, consult an engineer or err on the side of caution.

As a general disclaimer, the developers of this system will not be held responsible for injury or death or injury of any wildlife, and all liability resulting from the use of the equipment lies on the individuals who decide to use it.

LICENSE

CONTACT US

Questions or additional feature suggestions can be sent to one of us:

Scott Ziv
Undergraduate Researcher
scott.ziv@vt.edu

Samuel Anderson
Undergraduate Researcher
swand45@vt.edu

Meredith Semel
Graduate Advisor
merak91@vt.edu

Nicole Abaid, PhD
Doctorate Advisor
nabaid@vt.edu

BILL OF MATERIALS

For the nodes:

Part	Cost (USD)	Description
M0 Adalogger	22	Processor + SD + Battery Charging Circuit.
GPS Featherwing	40	GPS
2 * 2000mAH LiPo Battery	25	Largest we can get at our size
Accelerometer Adafruit MMA8451 breakout	8	Gets data about acceleration.
RFM69HCW Radio Featherwings	10	Packet Radio
Solar Panels	2	5V, 30mA.
Sandisk SD Card	20	4Gb is good enough. Sandisk SDs sleep better than other types.
Solar Panel Battery Charger	17.5	Manages Power from the Solar Panel and battery
Pololu S7V8A Voltage Regulator	6	Regulates the voltage from the solar panels

For the Base Station:

Part	Cost	Description
Side launch SMA connector	2.5	Connecting Antenna to Radio
RFM69HCW Radio Featherwing	10	Packet Radio for Base
SD Card	20	for storing a lot of data.
M0 Adalogger	22	Processor + SD + Battery Charging Circuit.
Feather protoshield	5	for interfacing to the radio
GPS Featherwing	40	For getting the same time as the lemurs
Solar Powered Power Bank	21	Basically a huge Power bank battery with a solar panel on it. Never the advertised size, but a 2W solar panel should be self sustaining.
SMA to UFL Connector	4	Connects GPS to Antenna
GPS antenna	10	Amplifies GPS Signal for faster acquisition
Directional 915Mhz Radio Antenna	24	Amplifies Radio Signal. Yagi antennas work best.
1200mAH LiPo Battery	10	Backup power for the Base + its RTC

Other Components

Part	Cost	Comments
PETG filament	15	PETG is Cheap, Strong, Water and UV resistant. When making the case, this is a good plastic
XL pin header	3	These 20mm long pins are long enough to go all the way through all three PCBs

Flex Seal	20	One can will last for the entire production run. Cover the device with it to waterproof it.
Solid Wire	5	some 22 gauge solid wire for modding and using as whip antennas

SOFTWARE SETUP

In order to upload the code, there are a couple steps to follow.

1. Download and install Arduino IDE from Arduino.org.
2. Open Arduino IDE, then go to file > preferences
3. In the space that says "additional board manager URLs", paste this URL:

https://adafruit.github.io/arduino-board-index/package_adafruit_index.json

4. Click OK
5. Restart Arduino.
6. Install the dependencies listed below
7. Load up the Sketch you want to upload BaseCode.ino or NodeCode.ino.
8. Go to tools > board:XXXX > Adafruit Feather M0.
9. Connect the board via USB
10. Find the COM port under tools > port
11. Press the arrow button to upload.

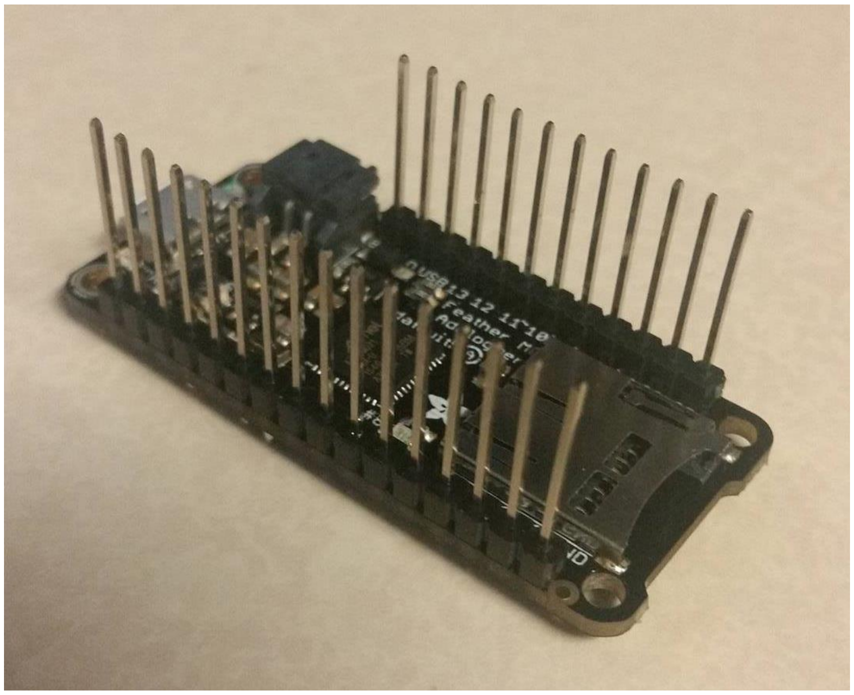
DEPENDENCIES

- Adafruit_MMA8451
- Adafruit_Sensor
- SDFat
- RadioHead
- RTCZero
- Adafruit_GPS

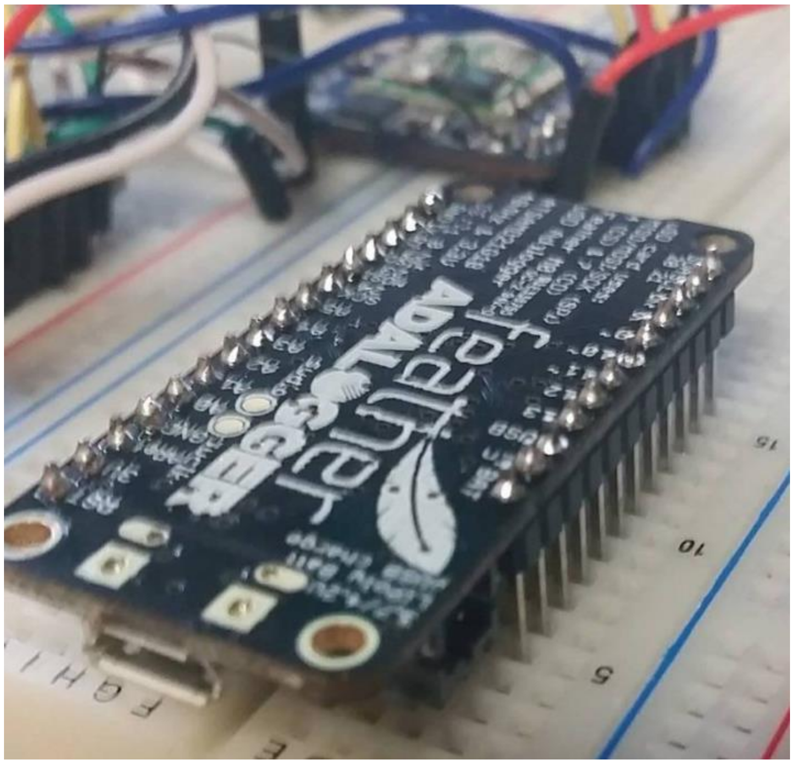
ASSEMBLY OF BASE STATION

Step 1: prepare the main board

Solder the pin headers to the Adalogger M0 control board, facing upwards. This serves as the main control board, and other components are stacked on top of this one.

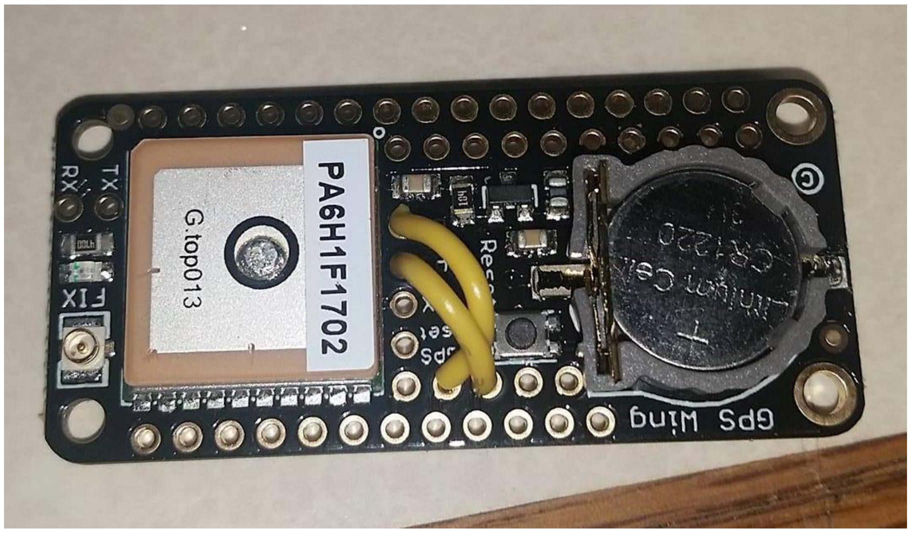


It may be easier to put it in a breadboard to solder straight, like so:



Step 2: prepare the GPS board

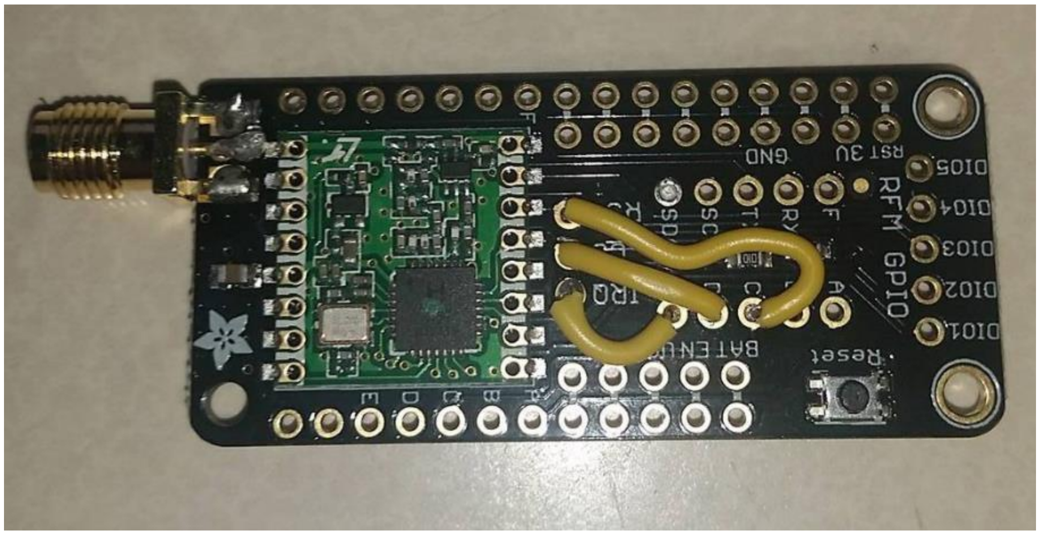
Connect the pins for Enable and PPS as shown below



The battery is optional, but suggested.

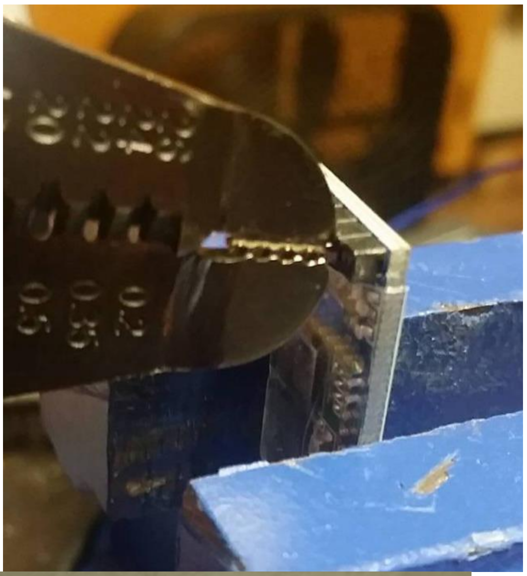
Step 3: prepare the radio board

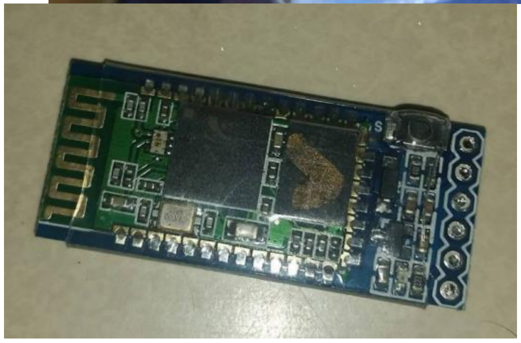
Connect and solder on the SMA connector, and connect the wires such that RST -> C, CS -> D, and IRQ -> E, as shown.



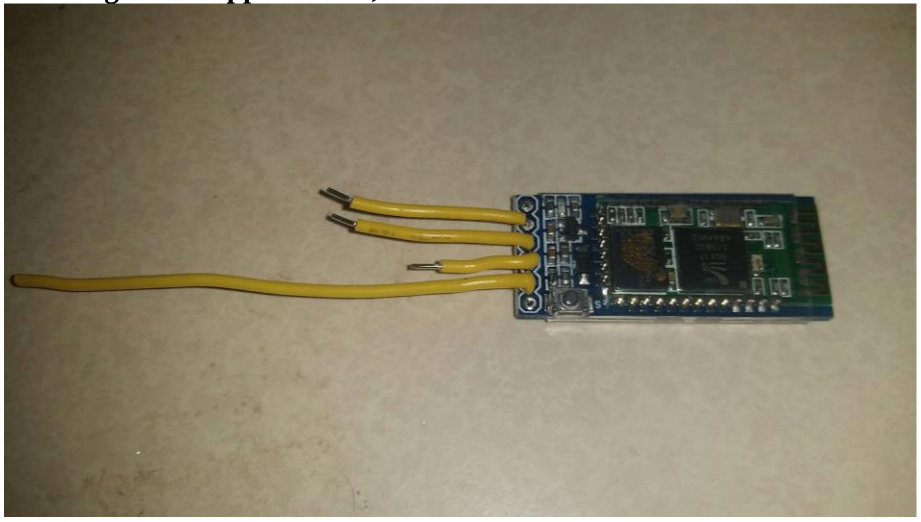
Step 4: Prepare the Bluetooth chip

Using pliers, straighten the pins on the Bluetooth HC-05 chip. Use a soldering iron and wick to remove solder and remove the pins. Be careful not to destroy the pads around the pins, as we will need to solder to them.





After the pins are removed, solder a set of wires to the middle four connectors, as shown below. Wire lengths are approximate, but too much wire is better than too little.

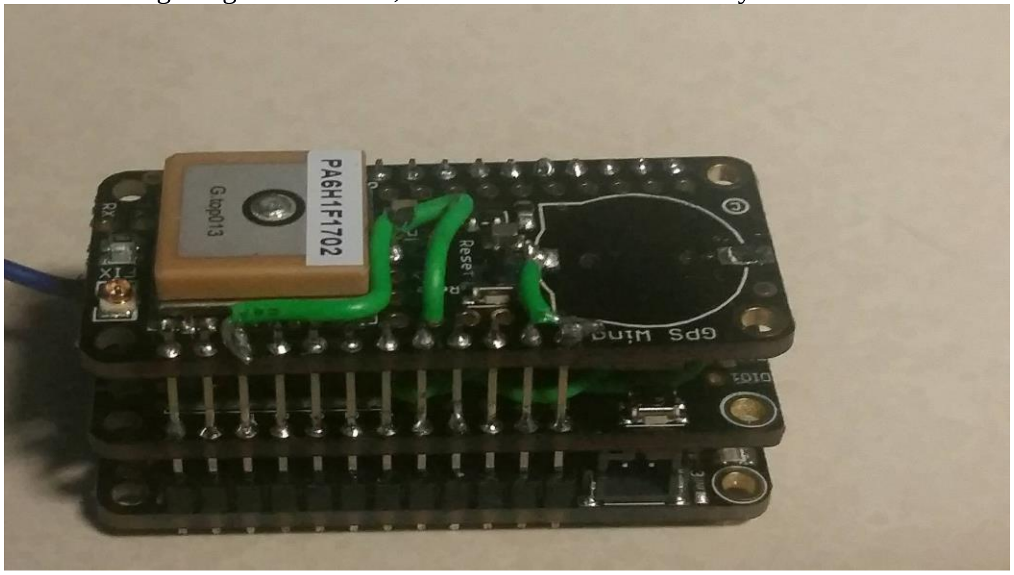


Step 5: Solder them all together

Double check all your connections before doing this, because once the wires are boards are connected, they cannot be unsoldered!

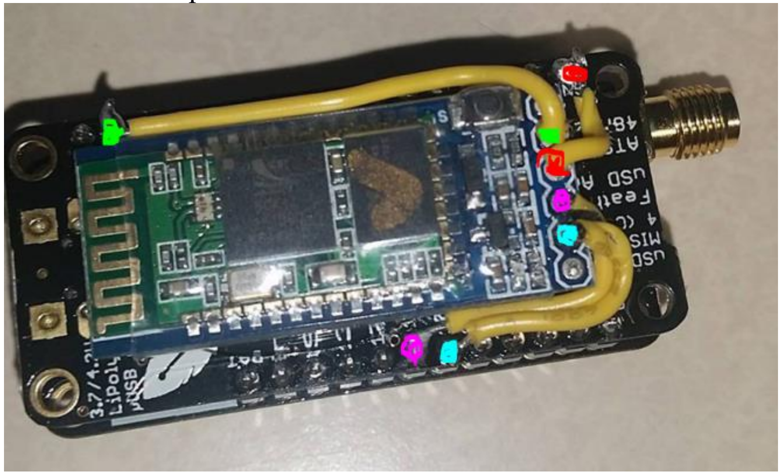
Starting with the base board, add the radio board, pushing it down until it is flush with the top of the battery connector. Start by soldering the four corners of the radio board, then fill in the rest. Once all the connections are soldered, attach the GPS board to the top of the stack, tack down the four corners, and fill in the rest.

The following image is of a node, but the base should look very similar.



Step 6: Solder the Bluetooth chip

Flip over the whole package, so you are looking at the bottom of the main board and solder the Bluetooth chip's wires as shown:

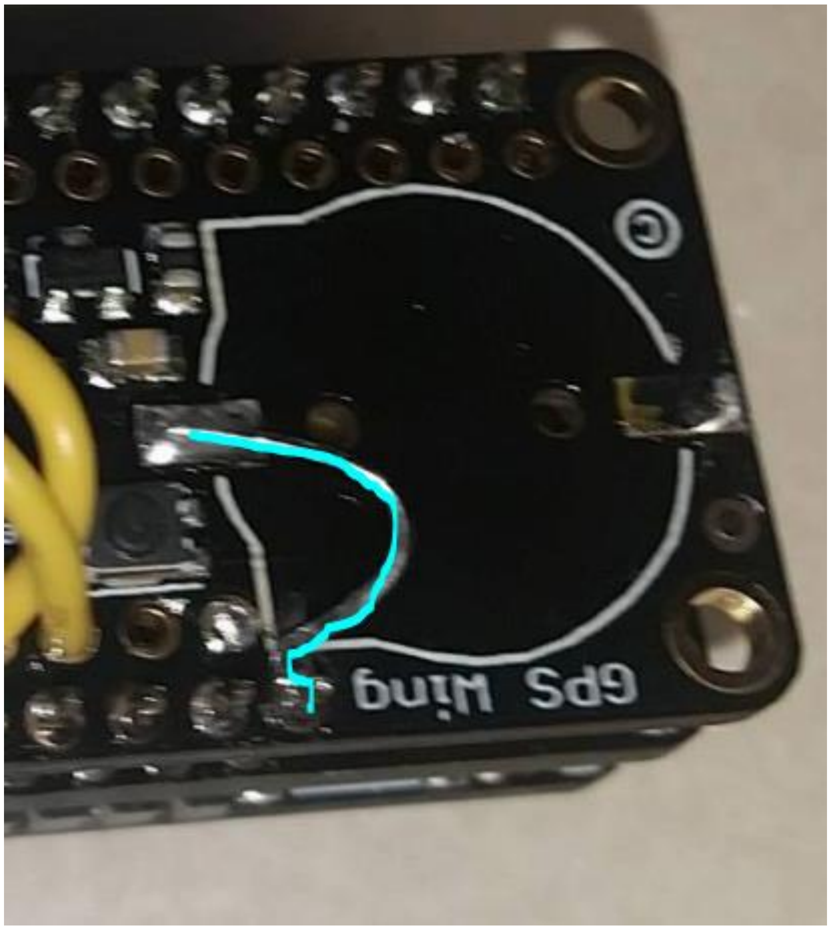


Once the connections are soldered, use a glob of hot glue to hold the chip down.

Step 7: Final touches

The coin cell keeps the almanac on board the GPS chip active. However, to keep the RTC correct, there is also a need to keep the normal battery attached. The coin cell will keep the GPS up to date, so it can do a hot start, and will keep the battery alive longer. Alternitivly, the button cell socket can be removed, and the positive lead can be wired directly to the LiPo battery connection, which will keep both the RTC and the Almanac updated, but will require the LiPo to be attached at all times.

In short, if you intend to turn the base station off and on multiple times, keep the button cell in place. If you lave a large enough battery with a charger, and intend to keep the base station powered, remove the button cell socket be desoldering both of its pads, and soldering a jumper wire from the pin nearest the battery connector on the main board, to the innermost pad around the socket, like so:



After that, the base station is assembled. Upload the code BaseCode.ino, and attach an SD card.

Step 8: Connecting with Bluetooth.

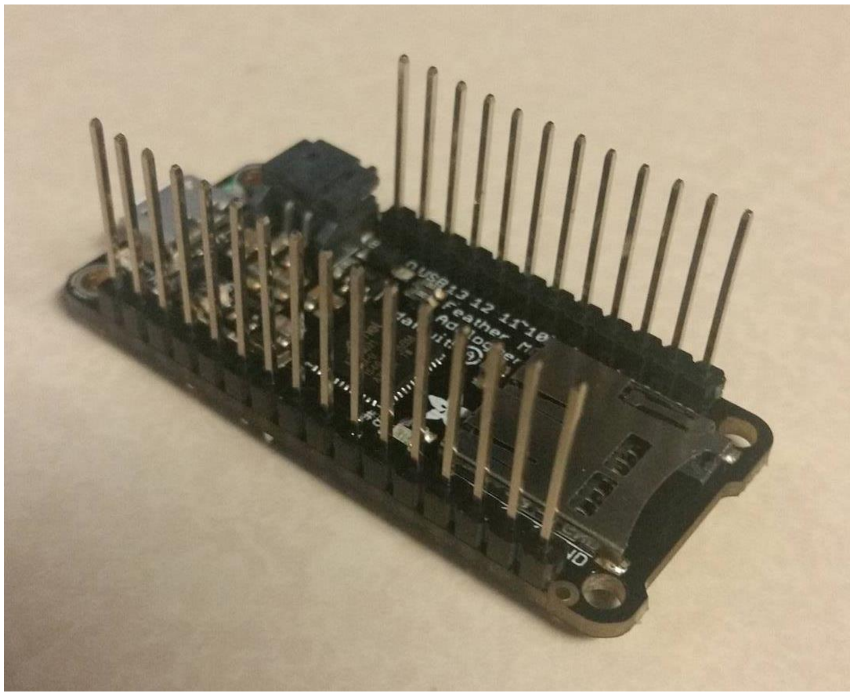
In order to communicate with the base station, you need a Bluetooth capable device that can communicate with HC-05 chips. Due to proprietary infrastructure and incompatible protocols, Apple products are not recommended, and if possible, an android mobile phone works well. There are many free Bluetooth terminal apps, but we have had good luck with this one: https://play.google.com/store/apps/details?id=project.bluetoothterminal&hl=en

Power on the base station by attaching the battery or a USB cable. Once that's done, search for and pair with the HC-05 chip within the android interface, typically a button along the top menu bar. Depending on the manufacturer, It may ask for a 4-digit pin, which is most often "1234". Once the pairing is successful, boot up the app, connect to the HC-05, and type "gettime". If everything is done correctly, the base should respond with the current time, which, since the RTC is not updated, should be around 0:00:00. Sending “getgps”, should make the base station update its clock.

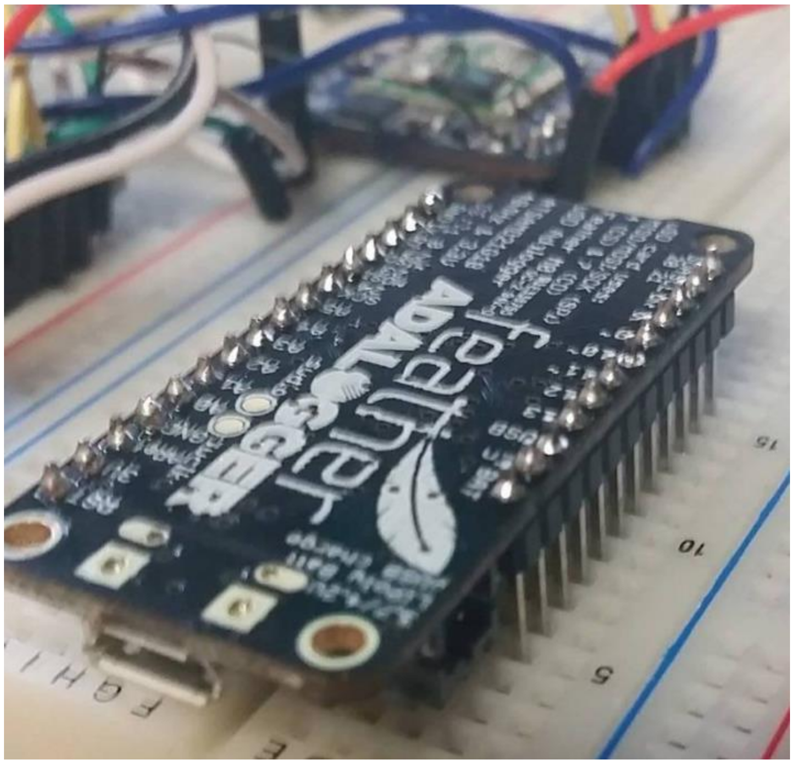
ASSEMBLY OF NODES

Step 1: prepare the main board

Solder the pin headers to the Adalogger M0 control board, facing upwards. This serves as the main control board, and other components are stacked on top of this one.

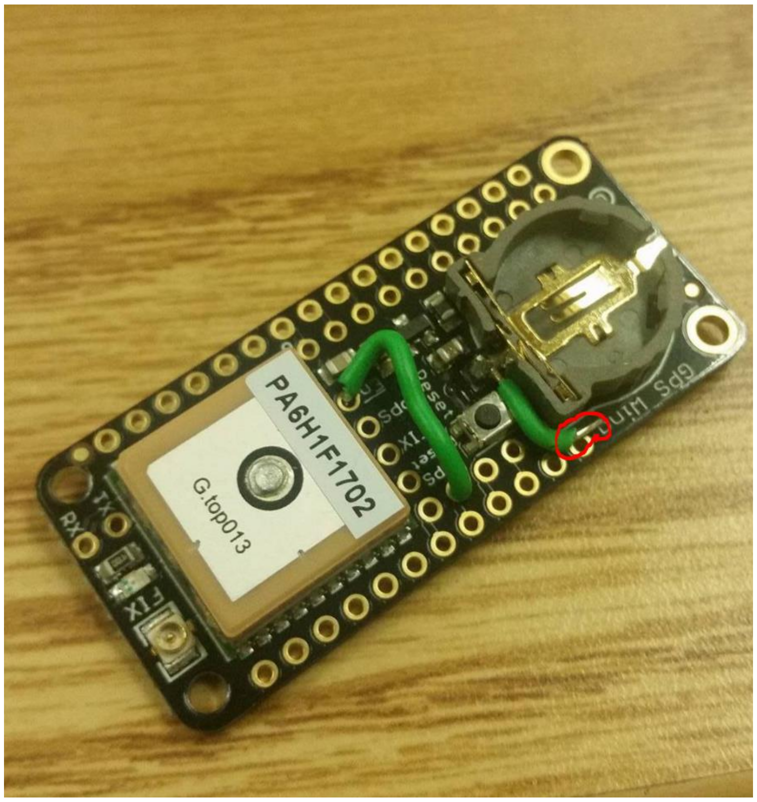


It may be easier to put it in a breadboard to solder straight, like so:



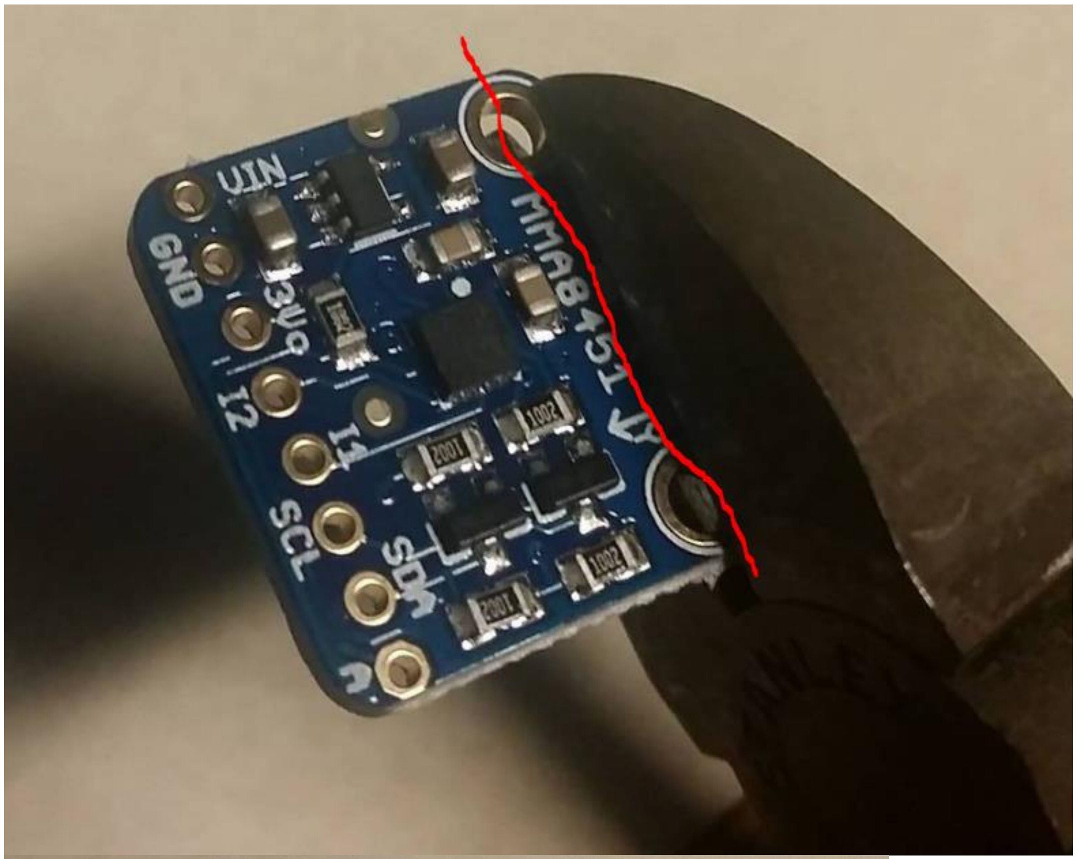
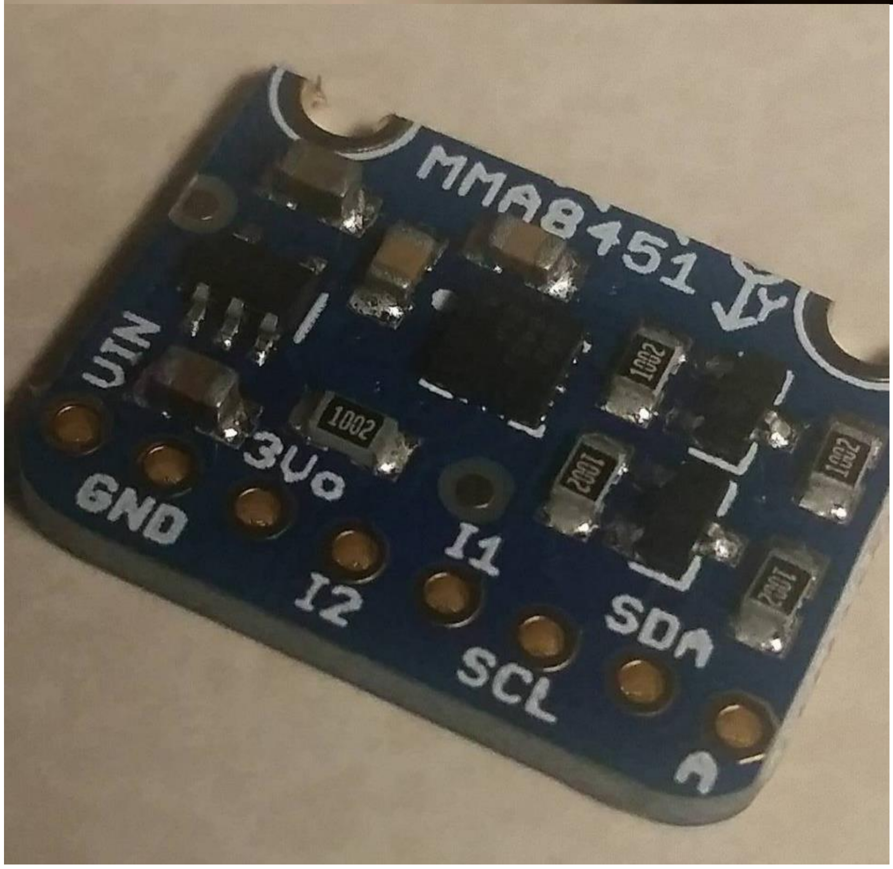
Step 2: prepare the GPS board

Solder the enable pin as shown. The red circle shows a hanging wire for the coin cell power pad. Later versions removed this socket entirely, but this wire must exist. It is suggested that you remove the coin cell socket entirely, and solder this wire to the pad. Do NOT connect the other end of the wire to anything yet.

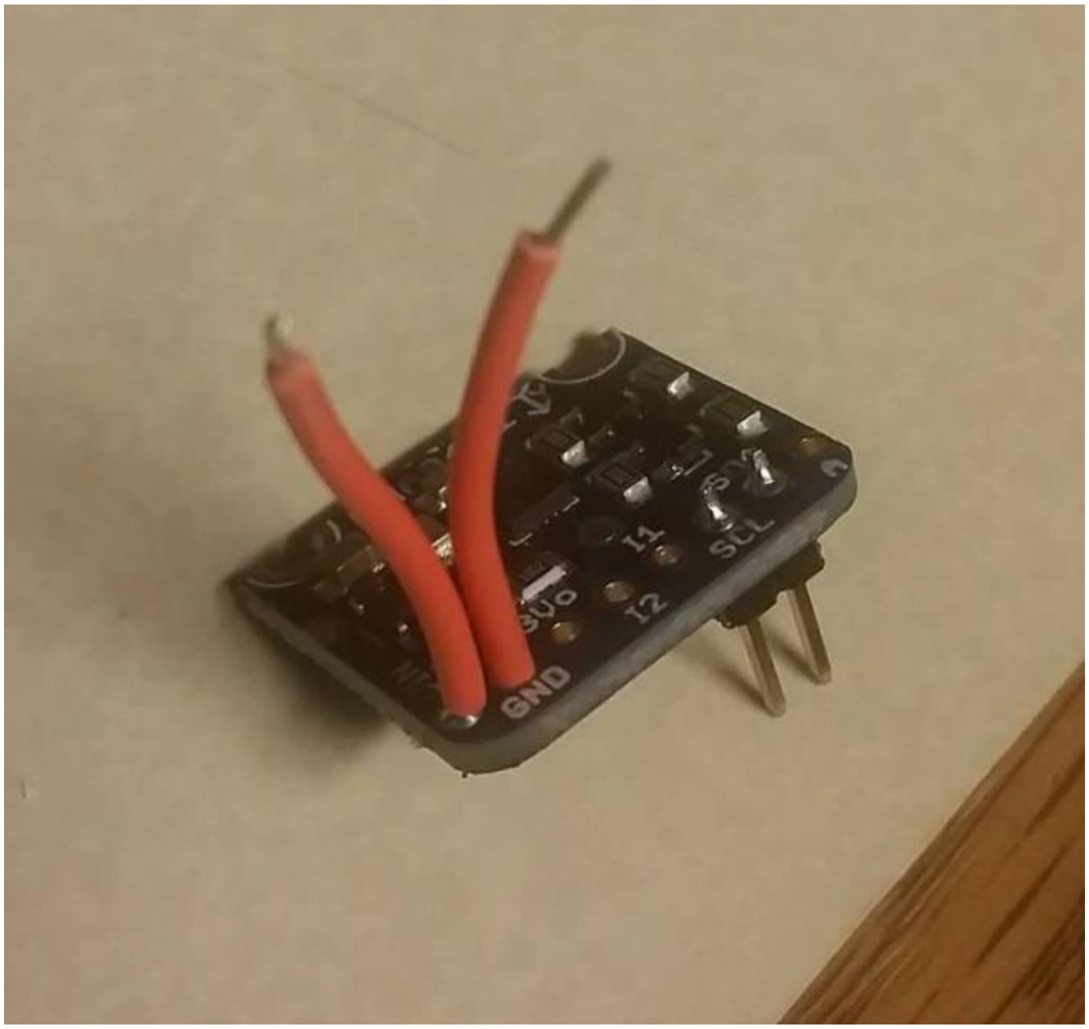


Step 3: prepare the radio/accelerometer board

Because the accelerometer board is just barely too long, we need to modify it. Using a pair of flat wire snippers, snip through the PCB, in a straight line between the center points of the two mounting holes. There are no electrical traces in his space, but be sure not to damage any components.

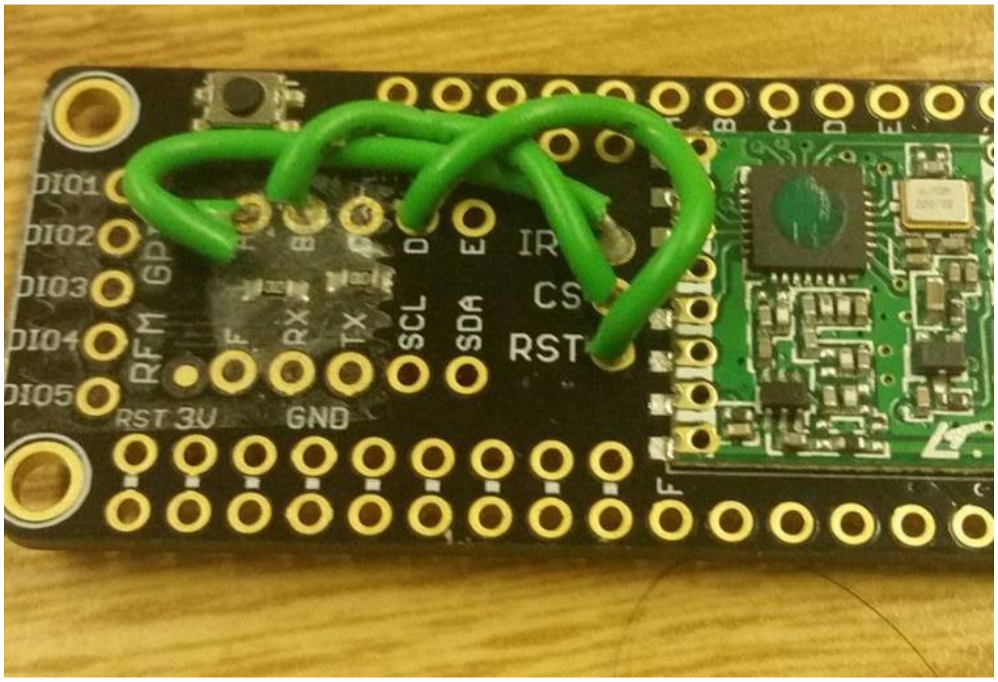

VIN
GND
3Vo
I2
I1
SCL
SDA
MMA8451
1002
2001

MMA8451
VIN
GND
3Vo
I2
I1
SCL
SDA
1002

Attach two pins of header to the SDA and SCL pins, facing down, and two short (1 in), wires to GND and VIN.



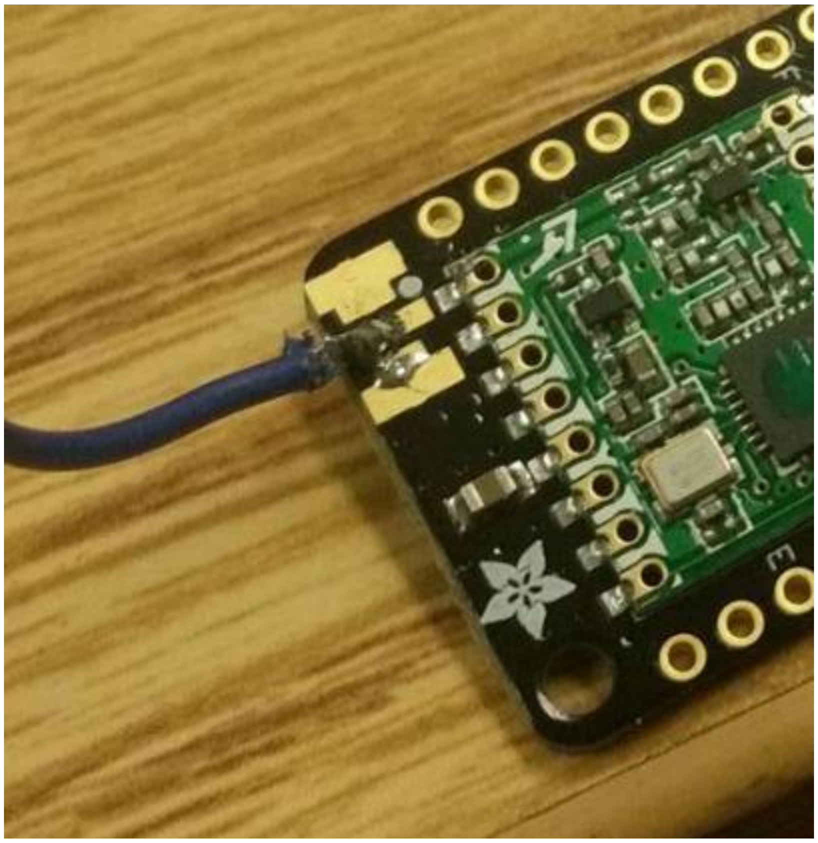
Set the Accelerometer board aside.

Solder the radio board so that IRQ -> A, CS -> B, and RST -> D, and, using a piece of tape (I used masking tape for visibility, but electrical tape is suggested, cover the F, RX, and TX pins in the middle of the board, while leaving SCL and SDA open, like so:

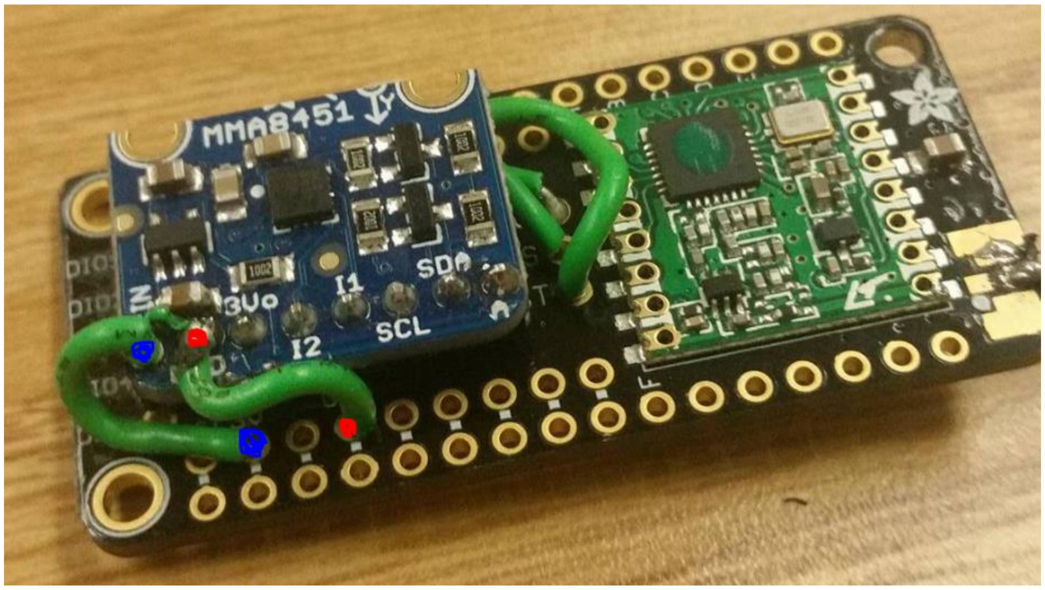


Make sure the wires can be pressed down.

Cut a wire 3 inches long, stripped at one end, but not at the other. This is a whip antenna. Not the best in terms of gain, but because it’s just a wire, it is very durable. Solder this wire to the ANT pin, which is on the far side of the radio chip, between two ground pads. Be careful not to jump the solder between the ground pads.



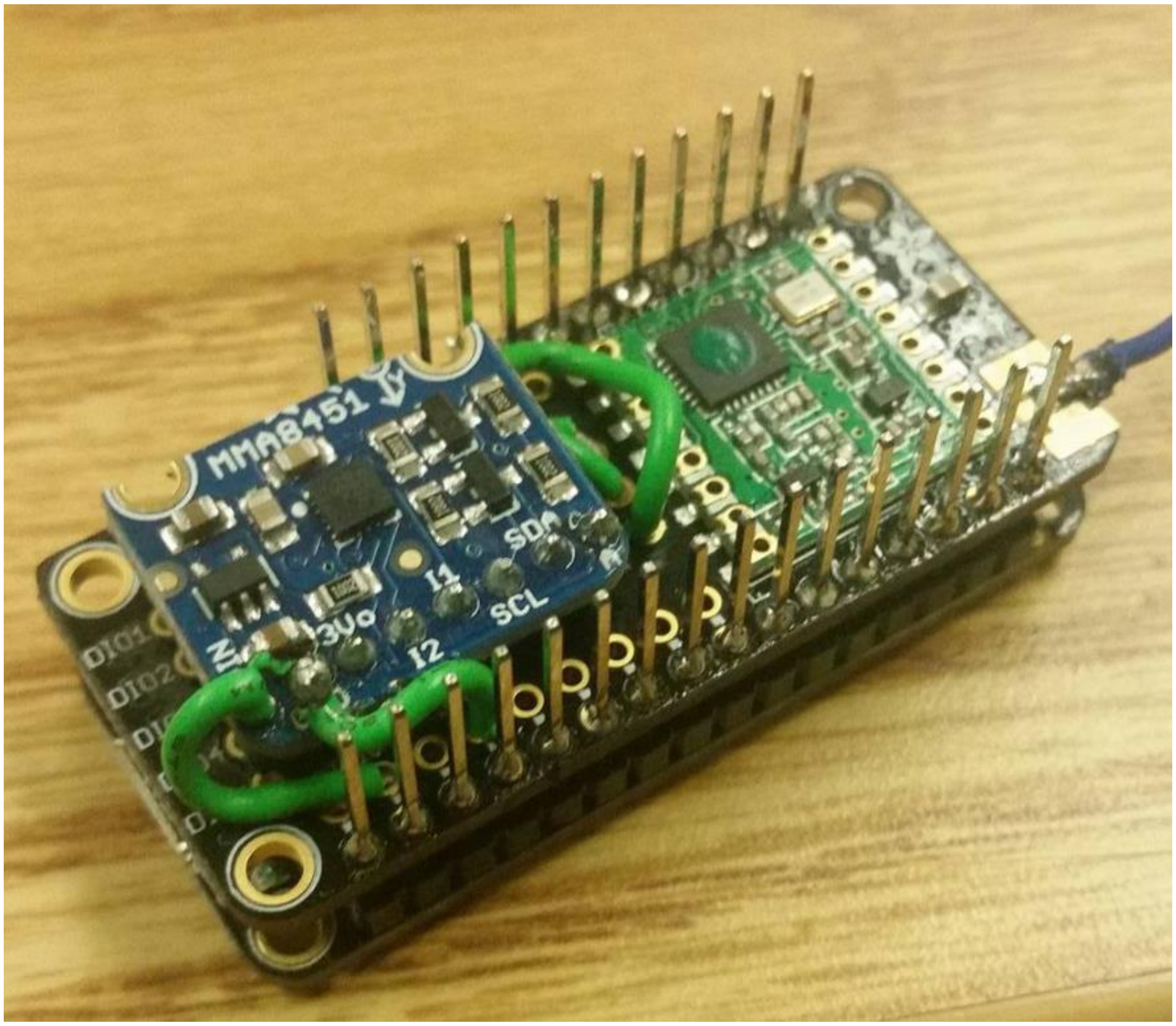
Go find the accelerometer board. As you may have guessed, the accelerometer board gets inserted such that the two SCL/SDA pins on the accelerometer line up with the SDA/SCL pins on the radio board. Insert and solder them, then cut off any excess header on the bottom. Connect and solder the wires for GND and VIN like so:



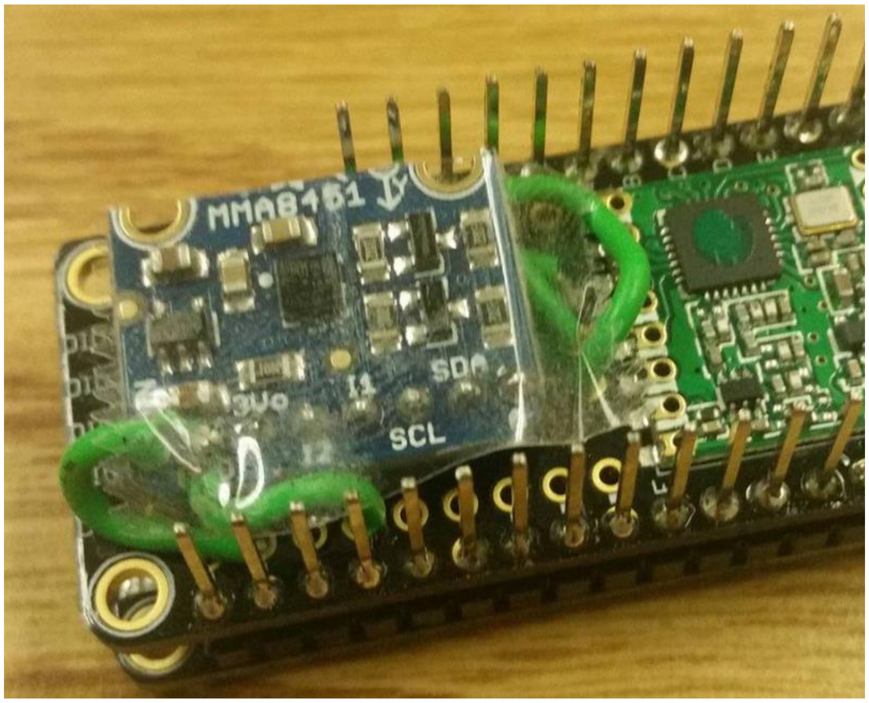
Step 4: Solder the radio board

Double check all connections thus far. Once soldered together, these boards will never come apart!

Insert the radio board onto the main board. If the accelerometer board doesn't clear the pins, or if anything is touching the exposed screw hole pad, remove the radio board assembly, reheat the solder on the SDA/SCL pins, and bend it out of the way. It is a tight fit, but it is doable. Once you are happy with the location, push it flush against the battery connector on the main board, double check to make sure nothing is shorting to the main board, solder the corners, then solder the rest of the connections.

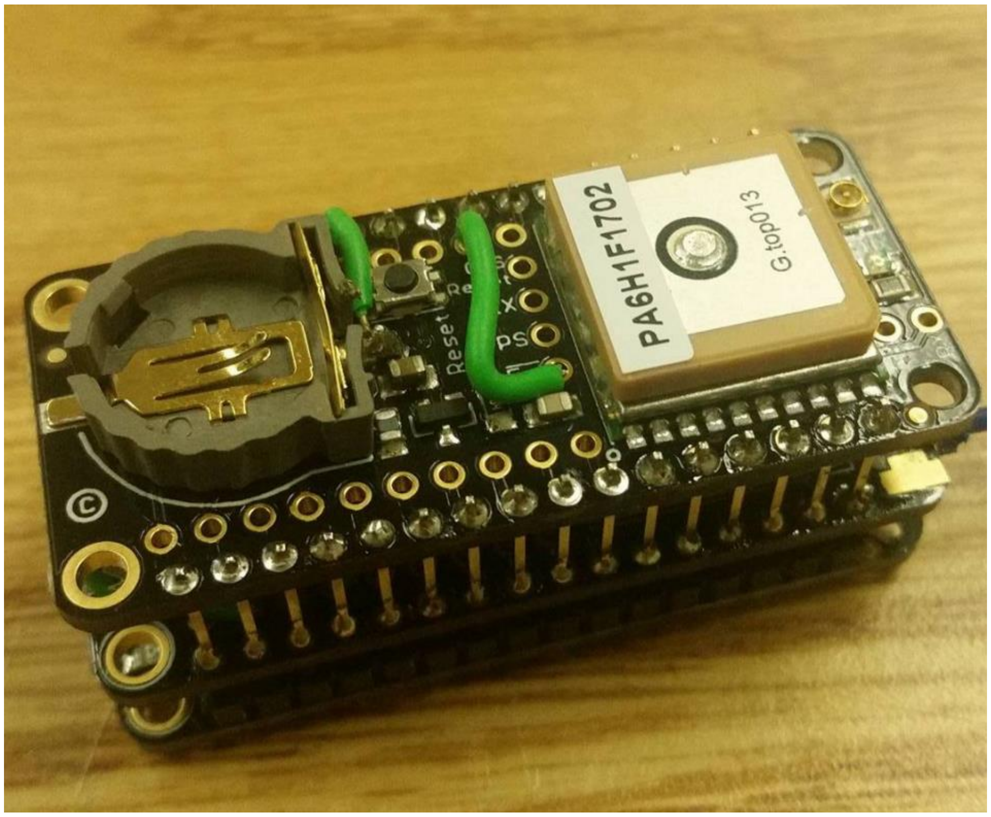


Add a piece of tape on top of the accelerometer. I used masking tape, but electrical tape is suggested.

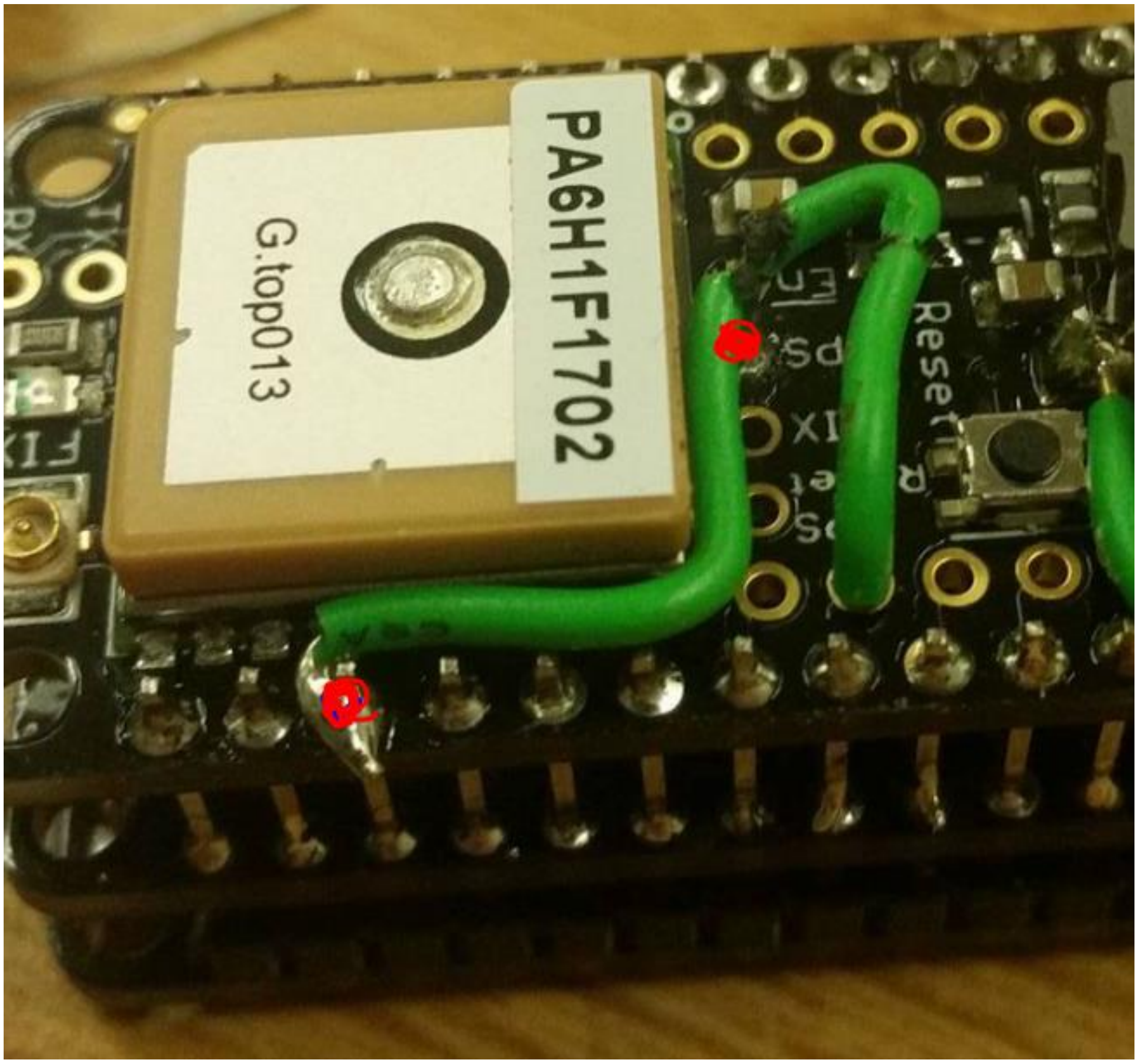


Step 5: Solder the GPS Board

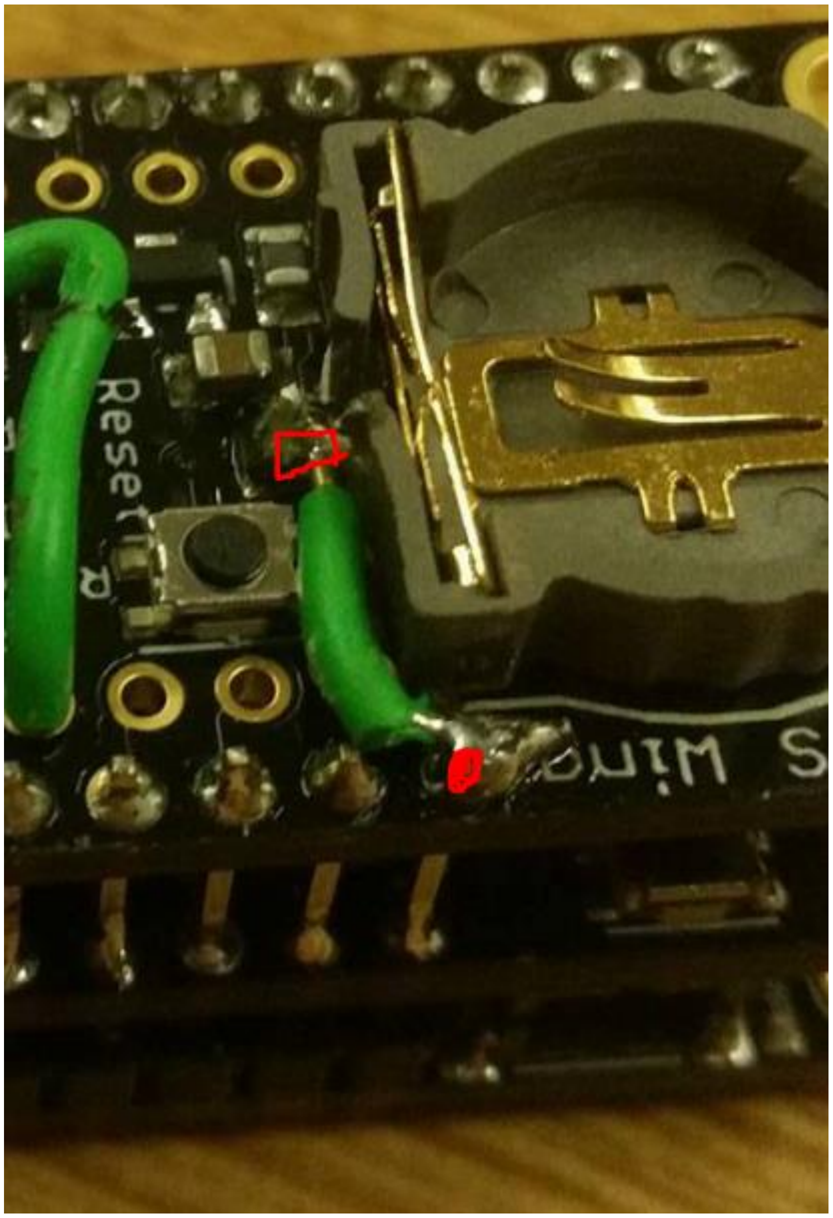
Insert the GPS board, solder the corners, and solder all the connections.



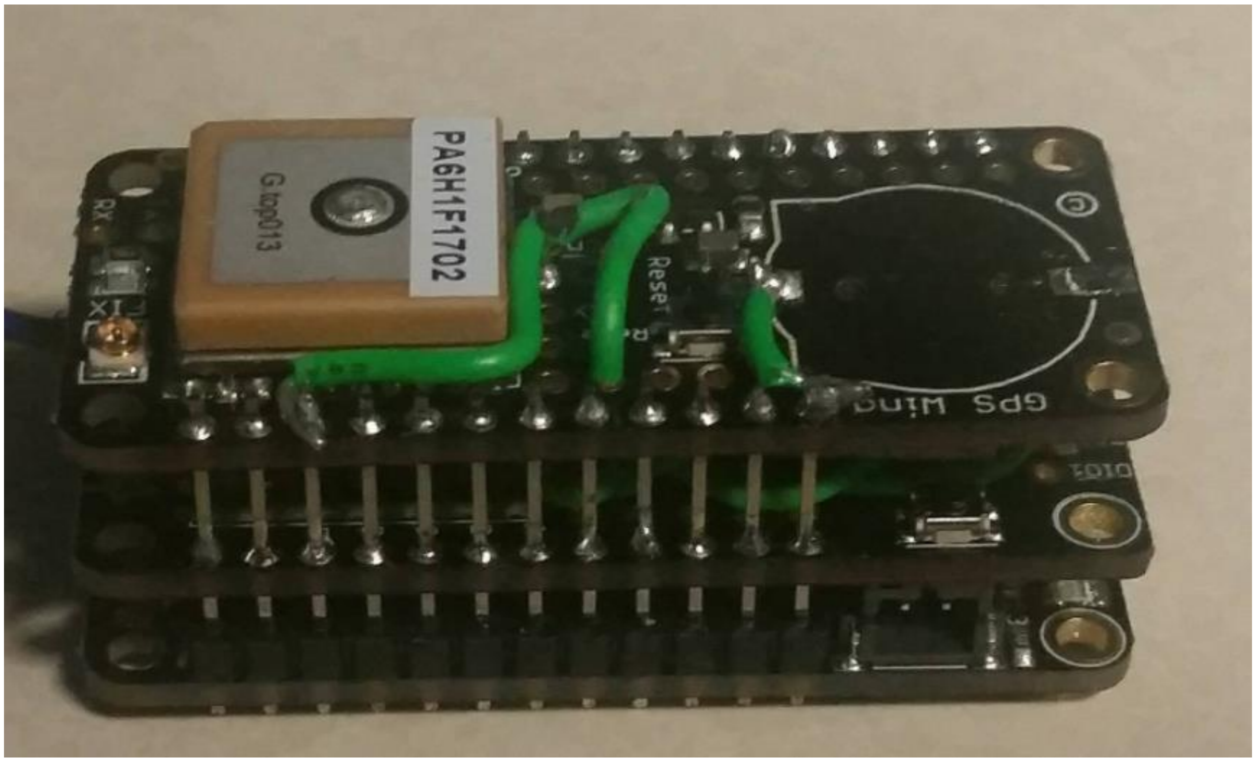
Cut another wire, and run it from the PPS pin on the GPS to the third pin from the end, like so:

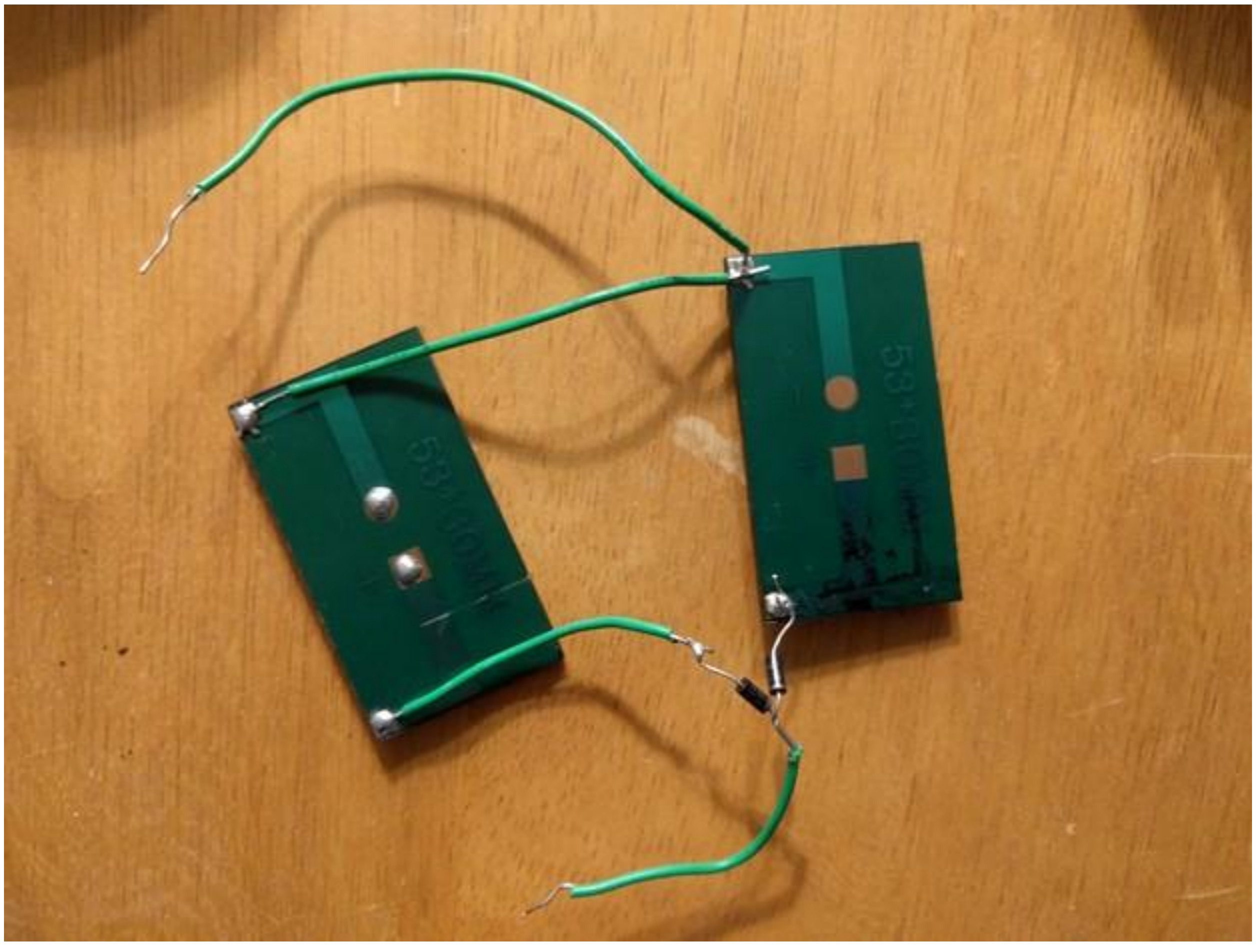


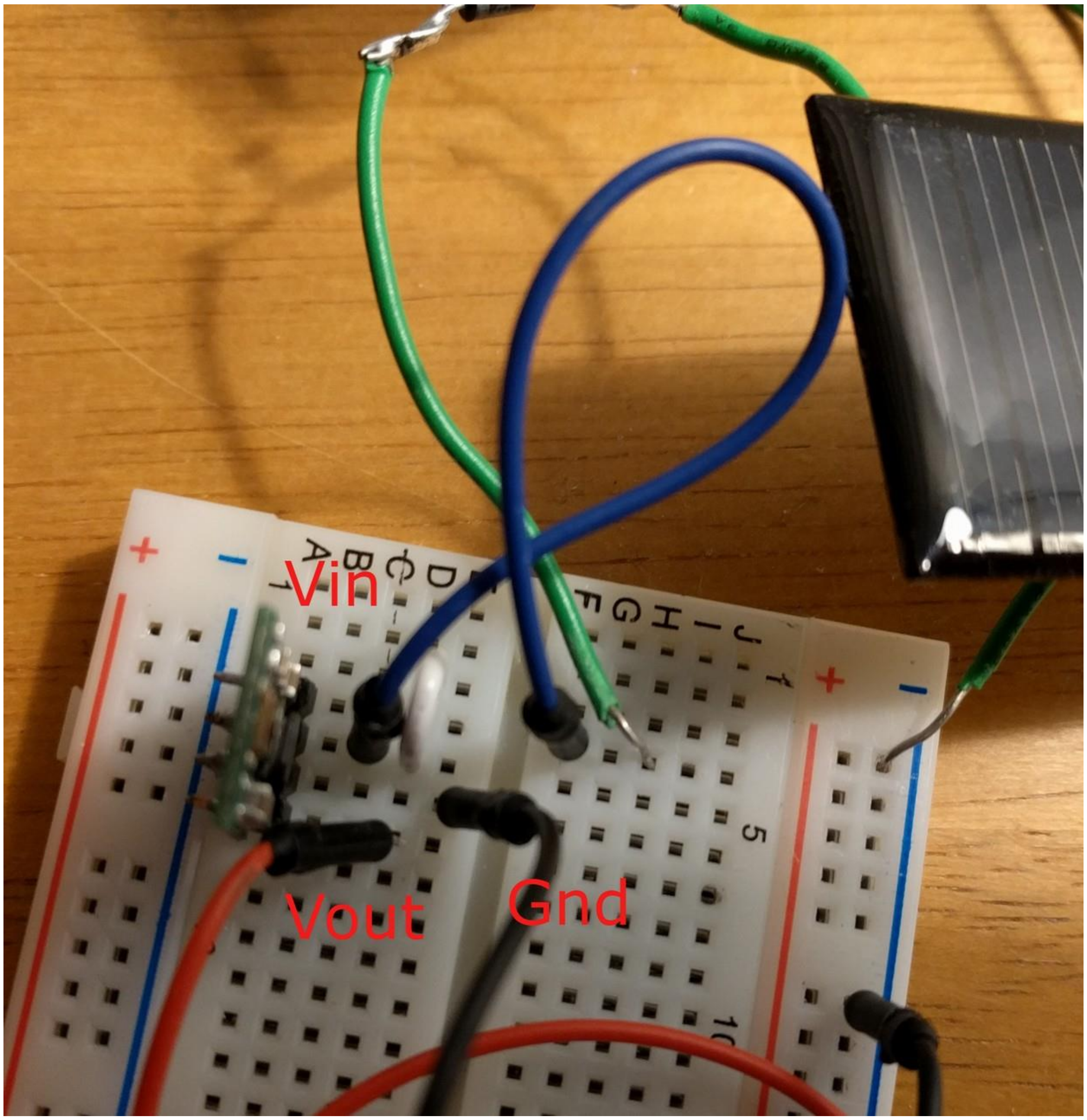
Remember that wire we left hanging? Connect it now:



If you want to save space, you can remove the battery connector entirely by heating the solder on one end, bending it up, and desoldering the other side while pulling the connector with tweezers.

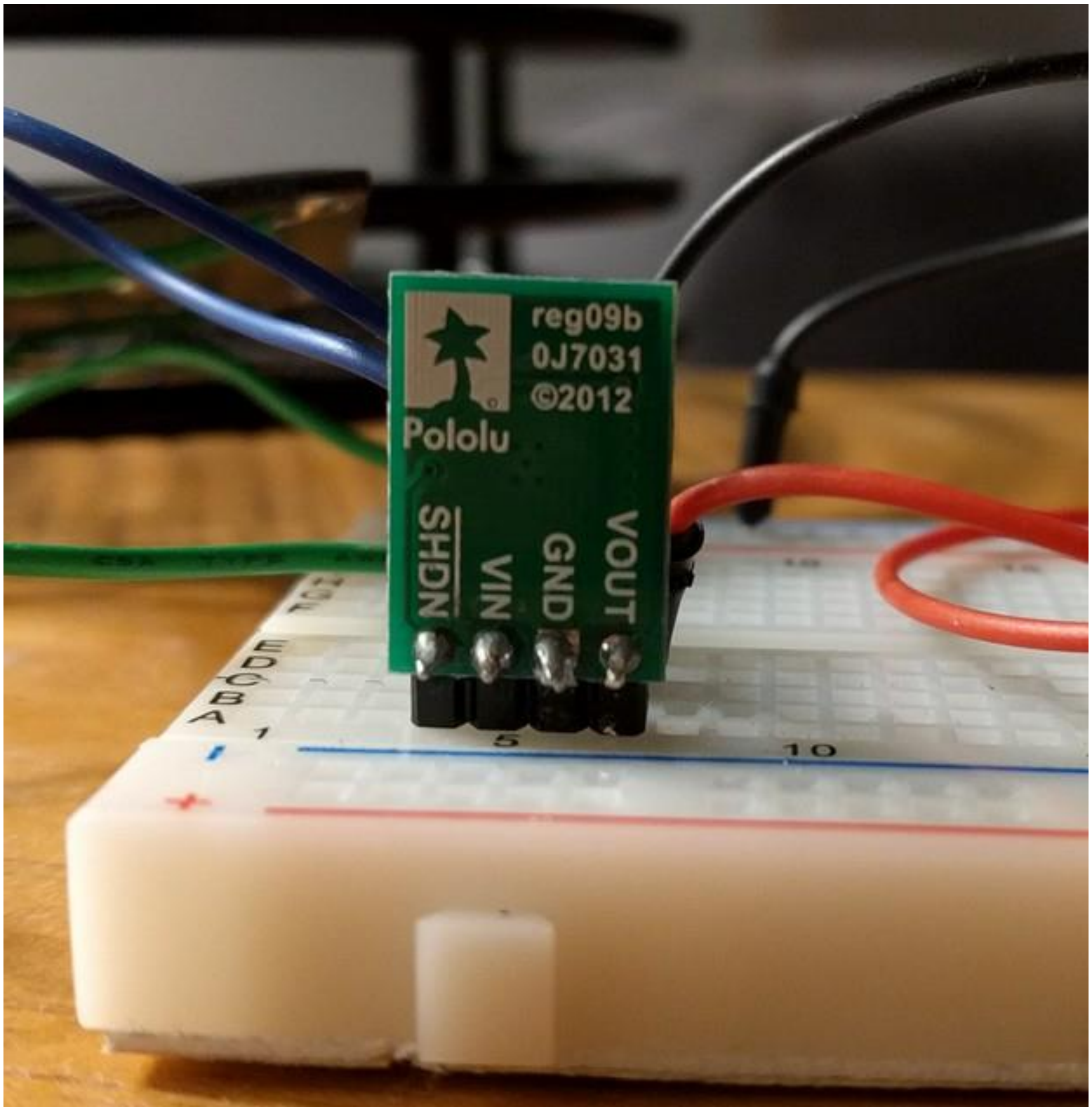

PA6H1F1702
Gtop013
Reset
GPS Wing
RX
TX


53*30MM
53*30MM



Solar:

Our setup uses two solar panels in parallel. Regardless of the number of solar panels, their output must be ran through a voltage regulator to normalize the voltage. Connect the Vout (+) end of the solar panels to the Vin and SHDN of the regulator. Connect the Ground (-) of the panels to the GND of the regulator. Finally, connect the Vout and GND of the regulator to the BAT and GND pin of the Arduino respectively.



Step 6: Software Setup

Upload NodeCode.ino to the board. See the “Software setup” section for more info.

Open and empty the SD card on a computer and add a file to it using right click > New > Text Document. Name the file “node”, so the full file name is “node.txt”

Open this file. This file holds only a single character, 1-9, no spaces, that represents which node this is. If you are only tracking one animal, make it “1”. If you are tracking two, make the first node “1”, and the second node “2”, and so on. Once that number is in place, save the file, close it, eject the SD, and put it into the main board.

POWER

The battery floats between 4.2V fully charged, and 3V when fully dead. However, any node with less than 3.3V in the battery will automatically shut off to prevent overdischarge. During this time, the solar panel will charge the battery until the voltage reaches 3.7V.

MEMORY AND COLLECTION

Default settings are for one line of text every minute. That means approximately 10000 lines of data/week. Current limits on data draw means that data can be pulled at a max speed of 4 lines/sec, putting a week worth of data at about 45 minutes of draw time. Work is being done to optimize this process, but until then, data should be drawn every 3 days if possible.

ENCRYPTION

By default, data is NOT encrypted over transmission. The Encryption key must be 16 bytes, and the same on the base and all the nodes. This can be changed the “Global variables for adapting behavior section” at the start of the code.

COMMANDS

The nodes will all be listening for a start command 59 seconds every hour, the entire 59th minute. They will not respond to any other command until this command is called. Once the start command is called, communications will be opened, and all effected nodes will listen for commands until the stop command is called to close communications. Data will be recorded every minute from the accelerometer and RTC. Every 5 minutes a check for nearby animals will be “pinged”. Every 15 minutes, the GPS will turn on, get a fix, and record the location on the SD card. If the GPS cannot get a fix within 30 seconds, it will skip it, and try again at the next 15 minute interval. Data is recorded every minute, and data that is not up to date will not be included.

The base can send a series of commands:

Internal Commands:

These commands are internal to the base station and do not affect the nodes.

Command	Name	Explanation	Arguments	Example
gettime	Print Current Time	This command is great to check if the base station is listening to the controller. It simply reads the time on the base station's RTC and prints it to the controller. This is the time that will be stamped on notes, so it is suggested that on bootup, getgps is called to update the time.	None	Example "gettime": prints the current time to the Android controller
settime	Manually set time	Manually sets time on the RTC. Not as accurate as getGPS, but it's faster, and works regardless of GPS signal strength. Not suggested unless GPS signal cannot be acquired.	MMDDYYHHMMSS: Month, day, year, hour, minute, second	"settime051417221330" Sets the base station RTC to May 14th, 2018, 22:13 and 30 seconds.
getgps	Update time from GPS	Updates time from GPS. Will spend up to 60 seconds attempting to get a signal. If a signal is acquired, it updates the time on the base RTC, to ensure notes are timestamped correctly. Also saves a gps coordinate, but will not update it until getgps is called again. Saved times will be erased and reset to 0 on a restart.	none	"getgps" updates RTC with current GPS time.

N/A	Note	If a statement is inputted that does not start with any of the other internal or external commands, it is timestamped, location stamped (accurate as of the last getgps), and saved in “notes.csv”. this is useful to ensure timestamped fieldnotes.	N/A	“Hello World” prints “Hello World!” to the SD card with a timestamp.

External Commands:

These commands are sent from the base to the nodes.

Command	Name	Explanation	Arguments	Example
strt	Start	This command should be the first call sent. The first 45 seconds of the 59th minute of every hour (12:59, 1:59, 2:59…) the nodes will all listen for the base station. If they receive “start” during this time, they will stop all their regular actions, and listen for other commands. Once they receive a “stop” command, they will continue regular actions. If they don’t receive a command for 5 minutes, they will automatically time out. This command also wakes nodes from extended shutdown, refer to below	None	“strt” will open communications with any node in the area
fndr	Finder	This command makes all the nodes transmit their node number for 30 seconds. The base station will listen and graph the signal strength. Using a Yagi or other directional antenna, it is possible to use this graph to	None	

		find out where the nodes are located		
dld	Download Data	Downloads data from a node. When the base station calls the DLD command, the node begins dumping data, line by line, and the base station grabs it and saves it to a file “nn.csv” where nn is the node being downloaded.	_nn: node number. This is the node you are trying to pull data from.	
wpd	Wipe data	To preserve data in case anything happens during data transmission, data will not be erased until explicitly told to be erased. After calling a download data, the user should check the data to see if it is correct. If something is missing from the data due to dropped packets, dld_nn can be called again to redownload data. If the data is fine, calling wpd_nn will delete the data to make it easier to download the next set of data.	_nn: node number. This is the node you are trying to erase data from.	
extshut	Extended Shutdown	The sleep command shuts down as much as possible on the collar and puts it into hibernation. No data whatsoever will be recorded. After the message is sent, all nodes in the area will immediately go to sleep It will wake up and listen for a start command at 12:00:00PM GMT, and if it doesn’t hear one by 12:00:59, it will update its RTC with GPS and go back to sleep.	_d: the day to stop sleeping, in standard form MMDDYY. The node turns back on at 12:00:01 AM GMT.	"extshut042318" will cause all lemurs the radio can touch to shut down immediately until April 23th, 2018 at 12:00:01 GMT.
flsh	Flash	Flashes an LED on the n’th node for 10 seconds	_nn: The nodenumber of the node to flash	“flsh_nn” flashes Node nn for 10 seconds
stp	Exit communications	Closes communications with base station. After this is called, the nodes will not respond to other commands unless a start command is	None	“stp" exits communications with all nodes within range.

		called during their 59 second listening periods.		

UNDERSTANDING DATA

The data output is displayed as a CSV file, which opens natively in Microsoft Excel. The Data is timestamped, and columns are the following, in order: Date, Time (in GMT), Latitude, North or South, Longitude, East or West, Altitude, Acceleration during the first second of the minute, Nearby nodes, and battery voltage. Altitude is included, but its precision depends heavily on which GPS satellites are in view, and may not be reliable.

Below is a bit of sample data, taken over 30 minutes:

6/17/2017	4:31:50	7933.375488	W	3738.831543	N	376.9	0.88		4.05
6/17/2017	4:32:00					0	0.57		4.05
6/17/2017	4:33:00					0	0.59		4.05
6/17/2017	4:34:00					0	1.48		4.04
6/17/2017	4:35:00					0	1.41		4.05
6/17/2017	4:36:06					0	1.61	1:-33	4.04
6/17/2017	4:37:00					0	1.11		4.05
6/17/2017	4:38:00					0	1.13		4.05
6/17/2017	4:39:00					0	1.03		4.05
6/17/2017	4:40:00					0	1		4.05
6/17/2017	4:41:06					0	1.78		4.05
6/17/2017	4:42:00					0	1.84		4.05
6/17/2017	4:43:00					0	1.43		4.05
6/17/2017	4:44:00					0	0.53		4.05
6/17/2017	4:45:00					0	0.92		4.04
6/17/2017	4:46:10	7933.375488	W	3738.831543	N	376.9	0.68	1:-32	4.05
6/17/2017	4:47:00					0	1.44		4.05
6/17/2017	4:48:00					0	0.57		4.05
6/17/2017	4:49:00					0	0		4.05
6/17/2017	4:50:00					0	0		4.05
6/17/2017	4:51:09					0	0		4.05
6/17/2017	4:52:00					0	0		4.05
6/17/2017	4:53:00					0	0		4.04
6/17/2017	4:54:00					0	0		4.04
6/17/2017	4:55:00					0	0		4.04
6/17/2017	4:56:09					0	0	1:-32	4.05
6/17/2017	4:57:00					0	0		4.04
6/17/2017	4:58:00					0	0		4.05
6/17/2017	4:59:00					0	0		4.05
6/17/2017	5:00:45					0	0		4.04

The date is June 17, 2017, at around 5am GMT. The Location numbers, 7933.375488 W, 3738.831543 N, are in degrees, minutes, and fractional minutes (seconds/60), so 7933.375488 is 79°33'22.5". However, Google maps can understand these coordinates if you add a space before the minutes, so 7933.375488 W, 3738.831543 N becomes 79 33.375488 W, 37 38.831543 N, putting the location somewhere along Interstate 81 in Virginia. The altitude is 376.9 meters above sea level, the battery voltage is between 4.05 and 4.04 volts, and because this data was taken in a car, the accelerometer is bouncing between 0 and 1.78 m/s^2, until the car stopped at 4:49 GMT.

The nearby nodes column was added in later to demonstrate. At 4:36 GMT, the node picked up another node in the area. The node it located was node number 1, and there was a -33 dB drop between them. dB drop is a very primitive, low power method of estimated rangefinding of other nodes, at the cost of precision. Every 5 minutes, the nodes drop the transmitting power of their radios, "ping" each other and measure the decibel drop between them. Assuming free space, clean, dry air, and line of sight, this dB drop can be approximately mapped to a distance. For example, a dB drop of -27 dB means the two nodes are within about 3 feet of each other, a dB drop of -30 is around 6 feet, and a drop of -91 means the signal is barely receivable, and this seems to be at >50 ft, but reliability is low at those ranges. In most practical applications, where line of sight is unlikely, the dB drop value will always have a greater magnitude, never a lesser. If a tree or rock is in the way, two nodes 6 feet from each other may have a drop of -40dB or more. So this range finding system can be interpreted, in more simple terms, as: if the magnitude of the drop magnitude is X dB or less, we *know* the other node is Y feet away or less. If the drop magnitude is X dB or more, it may be Y feet away but may be less distance with more interference. This function was added with socialization research in mind, but may have other uses.

PIN DEFINITIONS

For Base Station:

Pin Number	Use
6	Radio Enable pin
5	Radio Inturrupt
9	Radio Reset
13	GPS enable pin
4	SD Enable
10	Bluetooth TX
11	Bluetooth RX

For Nodes:

#Pin Number	Use
10	Radio Enable pin
11	Radio Inturrupt
6	Radio Reset
12	GPS enable pin
5	GPS Pulse-per-second
4	SD Enable

WIRING DIAGRAM

Coming soon…

CAD

Coming soon…