SPRITE CHIPSAT USER'S MANUAL

V. HUNTER ADAMS AND MASON PECK
CORNELL UNIVERSITY
SPACE SYSTEMS DESIGN STUDIO

June 21, 2016

CONTENTS

THANK YOU

Thank you for purchasing a Sprite development kit. Unlike other development boards with which you may have worked, the Sprite is the subject of extremely active research. By purchasing this board, you have become a part of that research. Please share your software so that other members of the community can benefit from your work, and please report any bugs/issues that you may discover. Ultimately, we would like to build a corpus of software that will be available for future flight missions. Please also be patient if our customer service is not as responsive as that of a company with the resources for such things. We are still a research lab, but we will do everything we can to help. Thanks again, and happy hacking.

1 GETTING STARTED

The Sprite is a 3.5 by 3.5 centimeter single-board spacecraft, though its suite of sensors make it useful for additional applications. These spacecraft have flight heritage from the KickSat mission launched April 18, 2014 and will also be flown in an upcoming launch with the NASA CubeSat Launch Initiative. The Sprite is capable of wireless programming and powering, and active attitude determination and control. The four-layer board includes an embedded loop of wire for inductive powering and to act as a torque coil. The boards also feature two ambient light sensors, a gyroscope, magnetometer, accelerometer, radio, microcontroller, and four solar cells. The purpose of this document is to get a new user comfortable programming and working with the Sprite.

The board has been designed to make the barrier to entry for new users as low as possible. To the greatest extent possible, we have used sensors with existing Arduino libraries from places like SparkFun and Adafruit. We've also wired the board in such a way that software that has been written for other development boards that use the same microcontroller can be used directly on the Sprite. In short, we've attempted to make the board as hackable as possible.

1.1 SPECIFICATIONS

- Dimensions: 3.5 x 3.5 cm
- MCU: CC430F5137 (MSP430 core + CC11xx radio SOC)
- Flash: 32KB
- RAM: 4KM
- Info Memory: 512 bytes
- Onboard LED, 9-axis IMU, ambient light sensors (2)
- Onboard solar cells (4), forward/reverse motor driver, full bridge rectifier
- Rx current: 18 mA max
- Tx current: 36 mA max
- Sleep current: 1-2 uA
- Maximum Tx Power: +12 dBm
- RF bands: 433, 868, 905, 915, 918 MHz
- Programmable via SBW, serial BSL and wirelesly (SWAP Python library)

1.2 Pin Definitions

From a schematic perspective, the Sprite's CC430 is wired identically to that of the open-sourced *PanStamp NRG 2*. In doing so, we are able to immediately take advantage of the excellent software that has been written for these boards. The following pin definitions are with respect to the Arduino programming environment. When programming the Sprite, use the pin numbers below to specify the pin associated with a particular sensor or device.

Sprite CC430 Pin Mapping		
Pin Number	Pin Name	Sensor, Device, or Purpose
23	P3.7	Onboard LED
8	P2.0	Front light sensor
13	P2.5	Back light sensor
3	P1.3	I2C SDA (to LSM9DS1)
4	P1.4	I2C SCL (to LSM9DS1)
15	P2.7	Motor driver - Enable
16	P3.O	Motor driver - In 1
17	P3.1	Motor driver - In2

1.3 A Tour of the Board

1. Solar cells: The Sprite features four *TriSolX Solar Wings* - two on the front and two on the back. The power from these cells feed the electronics directly without any regulation or power storage.

2. Embedded coils: The inner two layers of the board feature two coils of trace which serve the dual purpose of turning the Sprite into an electromagnet and generating power in the presence of an oscillating magnetic field.

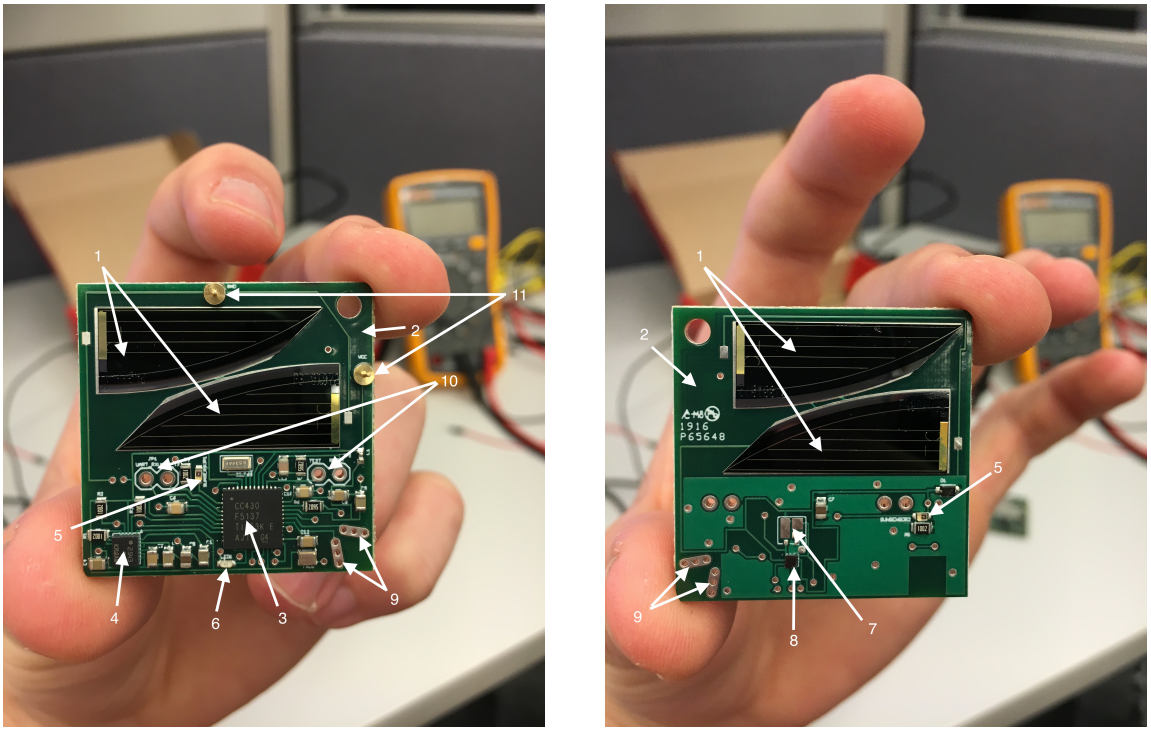
3. CC430F5137 Microcontroller/Radio: The *CC430F5137* is a convenient microcontroller for the Sprite because it has a built-in radio. This microcontroller is programmed from the Arduino IDE via a USB-UART converter, as described in a later section.

4. LSM9DS1 9-axis Inertial Measurement Unit: The Sprite features the *LSM9DS1* 9-axis IMU. This chip includes a magnetometer, accelerometer, and gyroscope, and it interfaces with the CC430 on an I2C bus. In addition to packing so many sensors into such a small footprint, this chip has ready-made libraries from SparkFun which make it extremely simple to use.

5. Ambient Light Phototransistor: The Sprite features two *phototransistors*, one on the front and one on the back. These transistors can be used in combination with the gyroscope and magnetometer to estimate the orientation of the Sprite relative to the Earth. The phototransistors interface with two analog in pins of the CC430.

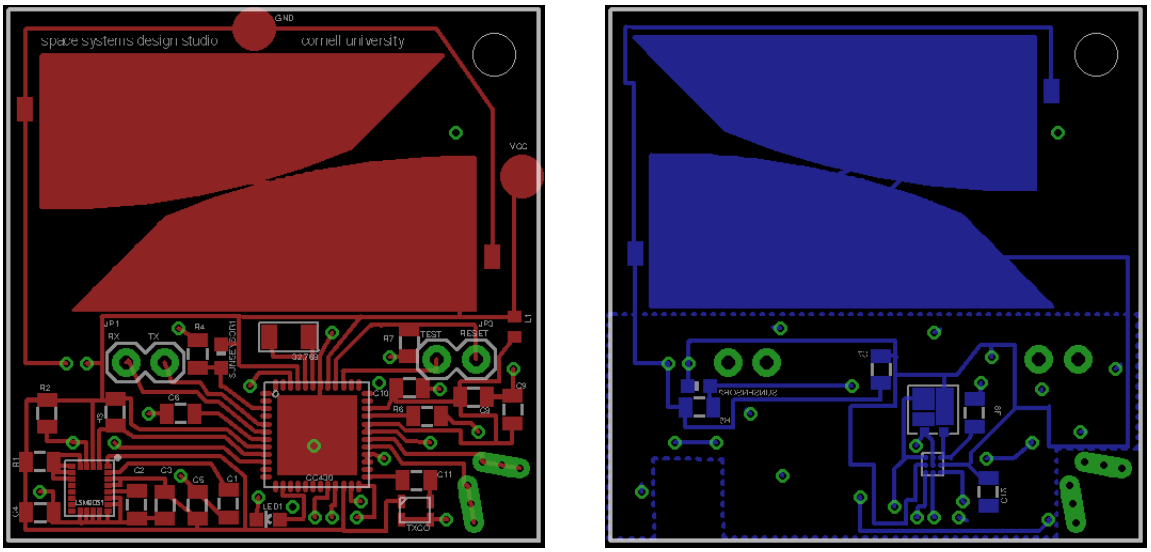
6. LED: The LED is included to provide feedback to the user during programming/operation. It interfaces with an output pin of the CC430.

7. Full bridge rectifier integrated circuit: The Sprite features a *rectifier integrated circuit* that enables wireless inductive powering of the boards.



(a) Sprite front (b) Sprite back

Figure 1.1: Sprite, with labeled components



(a) Sprite front (b) Sprite back

Figure 1.2: Sprite Eagle layout

8. H-bridge motor driver: The Sprite's *forward/reverse motor driver* is used to control the direction of current through the embedded torque coils. By changing direction of current, one can change the polarity of the electromagnet into which the Sprite can turn itself.

9. Antennae: The Sprite features a dipole antenna used for RF communication.

10. Serial interface: The TX, RX, TEST, and RESET pins are used to program the CC430, as shown in a later section.

11. VCC and Ground: These pins power the Sprite during programming, and double as a mechanical interface to the Sprite mothership while in orbit

The non-labeled components on the board are all of the components required to make the CC430 and sensors operate (capacitors, resistors, inductors, oscillators, etc.). These are all shown the schemtic shown in Fig. 1.3. Fig 1.4 shows the CAD of the interior traces that form the torque/inductive coils of the Sprite.

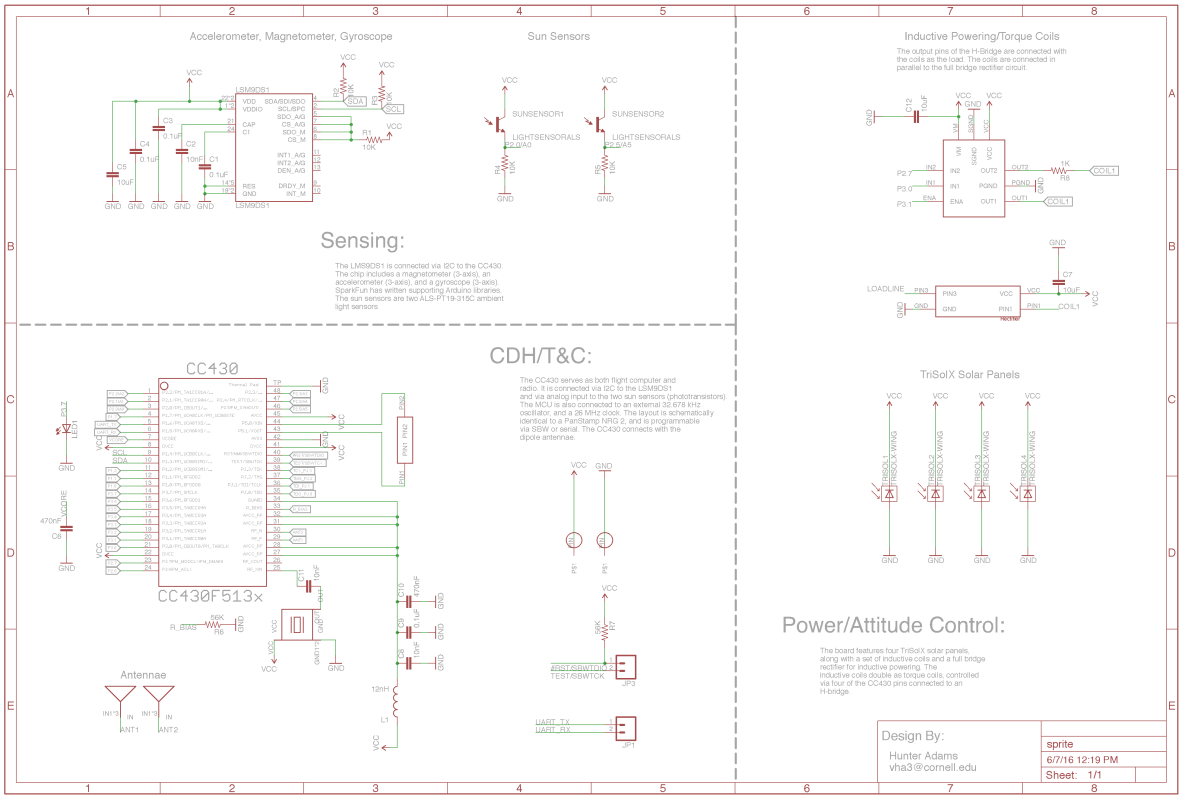


Figure 1.3: Sprite schematic

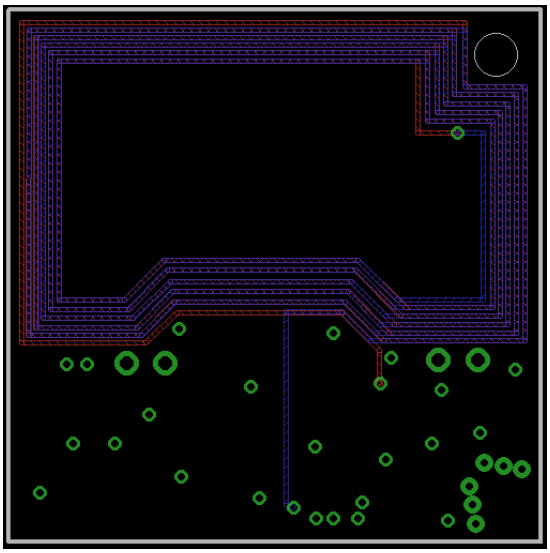


Figure 1.4: Sprite interior traces forming torque/inductive coils

1.4 Software Installation

As previously mentioned, the Sprite is designed to take advantage of the software written for the open-sourced board featuring the same microcontroller - the *PanStamp NRG 2*. The Sprite and PanStamp NRG 2 are identical from the Arduino IDE's perspective. For that reason, the software installation process is the same for both boards. That process can be found at *this website*, and also below.

1.4.1 Installing Board Cores

- Navigate to the *downloads page of the Arduino website*. The recommended version of Arduino to download is 1.6.5.
- Open Arduino v.1.6.5, and navigate to *Arduino→Preferences*. That will open the window shown below.

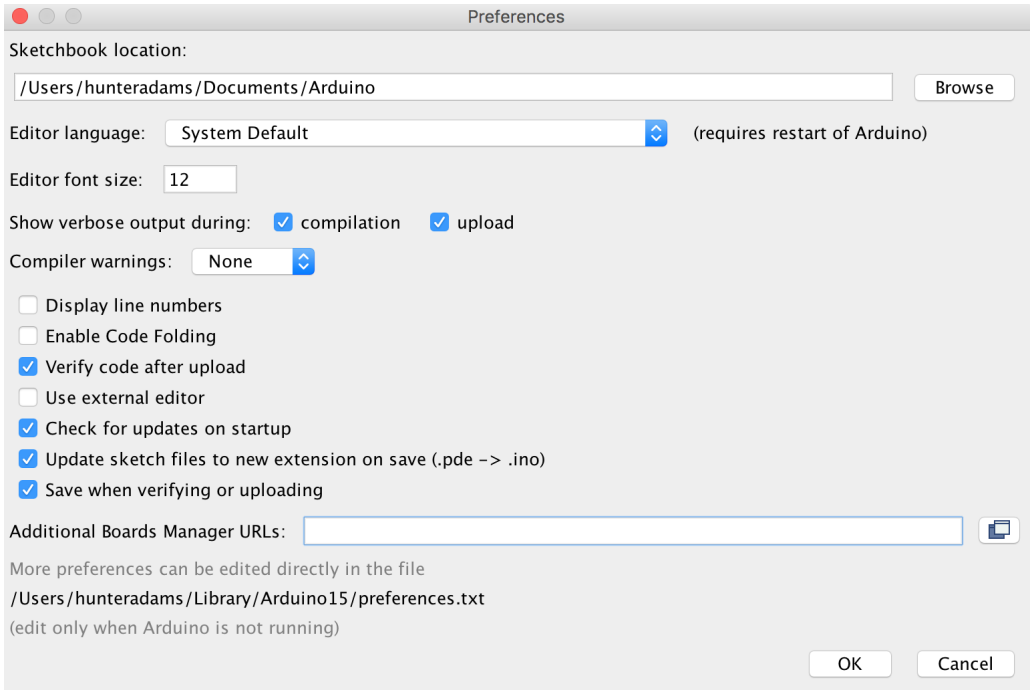


Figure 1.5: Preferences window

- As shown in the image below, enter the following text in the field labeled "Additional Board Manager URL's": http://panstamp.org/arduino/package_panstamp_index.json

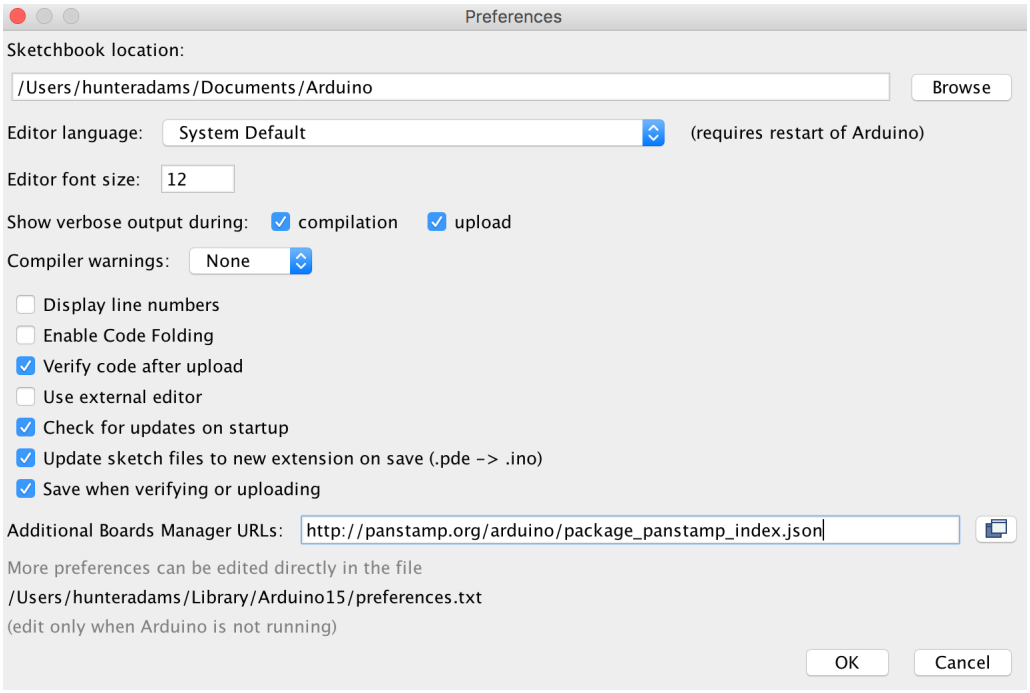


Figure 1.6: Preferences window, with additional board URL

- Save changes by pressing "OK", which will close the window
- Navigate to the *Tools* → *Board* menu and select "Boards Manager" from the dropdown, as shown below

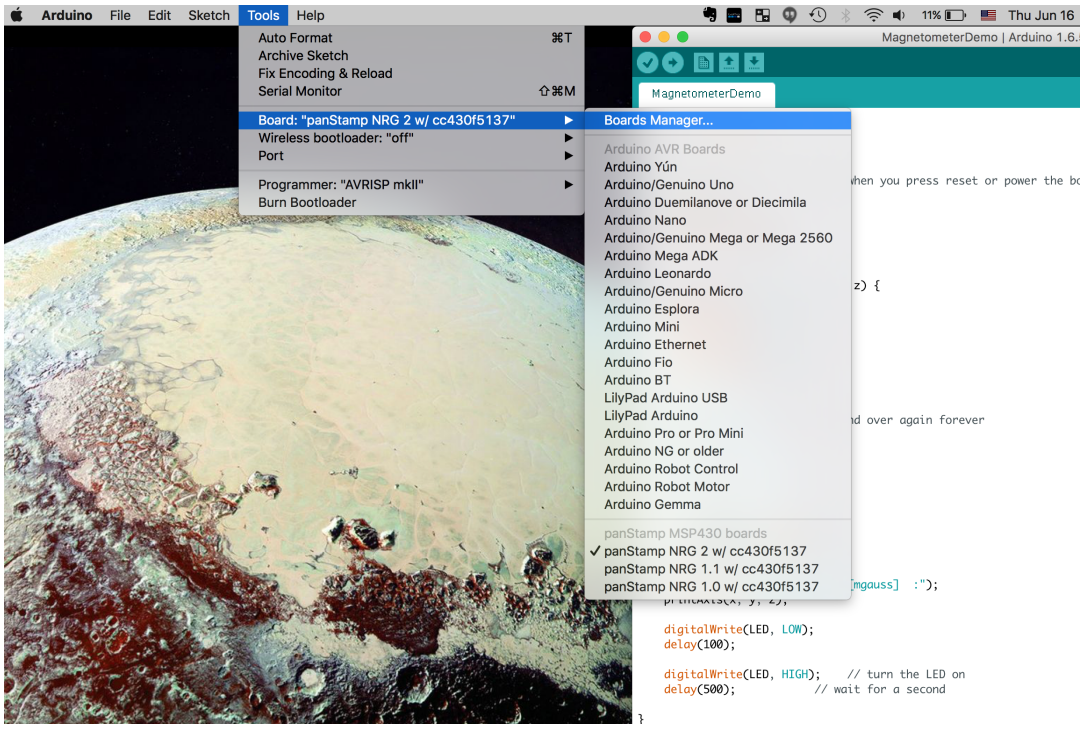


Figure 1.7: Selecting Boards Manager

- In the window that opens, scroll down until you find "PanStamp NRG w/ CC430f5137." Click "Install"
- Navigate to *Tools→Board*. In the dropdown menu, you will now find a board called "panStamp NRG 2 w/cc430f5137." This is the board that will be selected for programming the Sprite

1.4.2 Installing Arduino Libraries

Arduino libraries are installed a bit differently. There are a few libraries that will be required in order for some of the sample sketches to work.

- First, navigate to *https://github.com/vha3/sme-lsm9ds1-library*. This is a fork of a library written by SmartEverything for the LSM9DS1 IMU. Click *Clone or Download→Download zip*, as shown fig. 1.5.

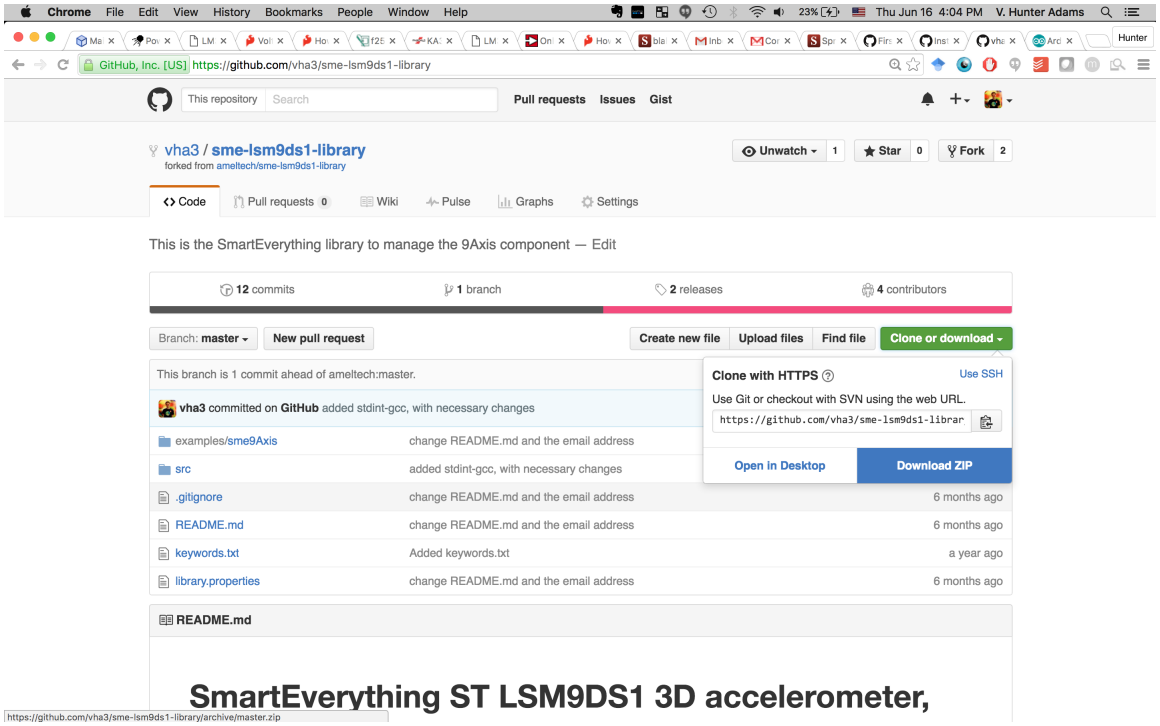


Figure 1.8: Download zip of IMU library

- In the Arduino IDE, navigate to *Sketch→Include Library* and select *ADD .ZIP LIBRARY* from the dropdown menu
- In the window that opens, navigate to the .zip file that you just downloaded and open it
- If you return to the *Sketches→Import Library* menu, you will now see the new library available for use. The zip folder has been expanded in the *libraries* folder of your Arduino sketches directory

Some other libraries that may need to be installed in exactly the same way for sample sketches to work include the following:

1. *SWAP*
2. *SPI SRAM*

1.4.3 Downloading Sample Sketches

Some sample sketches are available from *this repository*. The repository contains Sprite-specific example sketches, and it links to the panStamp example sketches which will also work on the Sprite. As you develop software of your own, please make pull requests on this repository so that it can be used by others.

2 Programming the Sprite

NOTE: READ THROUGH THE WIRING SECTION IN ITS ENTIRETY BEFORE MAKING ANY CONNECTIONS TO AVOID DAMAGE TO THE SOLAR PANELS

2.1 Wiring

All necessary prototyping materials are included in the Sprite development kit. For those that bought or made a Sprite independently, *here* is a DigiKey cart will all necessary supplies.

In order to prevent reverse-biasing the solar panels, the Sprites are programmed with VCC at 2.2 volts. Unfortunately, the SparkFun USB/UART converter sets VCC to 5 volts. The LM317 adjustable voltage regulator is used to regulate the voltage to 2.2 volts. Note that any USB/UART converter will work, and any voltage regulator that regulates to 2.2V will work. The schematic in Fig. 2.1 assumes the SparkFun converter and an LM317. If the solar panels glow red when VCC/GND are attached to the board, the voltage is too high. This is not catastrophic, but continued reverse biasing can lead to damage. It is good practice to keep the voltage low enough to be safe.

As an aside, the wiring involved with programming the Sprite may beg the question of why there is not onboard voltage regulation that would preclude such precautions. The reason is that the Sprite is designed to be as reliable as possible on orbit, at the expense of some convenience when programming. By omitting onboard voltage regulation, we could simplify the board and reduce the number of components, making more operationally reliable product.

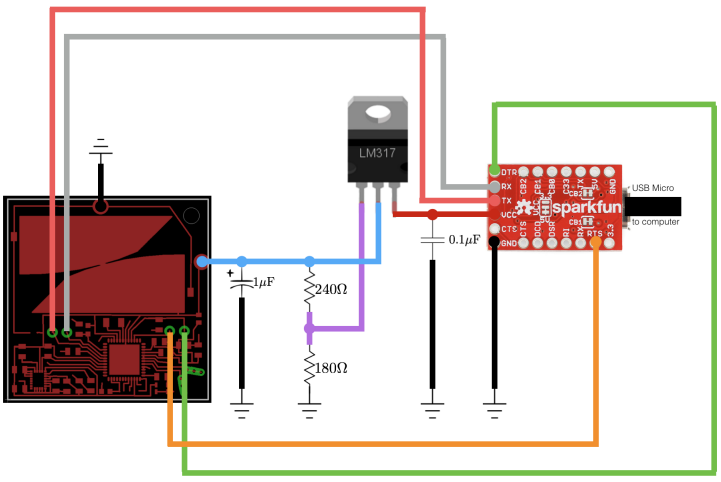


Figure 2.1: Wiring schematic for programming Sprite

2.2 Loading Programs

With the Sprite wired as shown in Fig. 2.1, plug the USB Micro into the computer and open the Arduino IDE. Then execute the following steps:

1. Navigate to *Tools→Board* and select "panStamp NRG 2 w/ cc430f5137".
2. Navigate to *Tools→Port* and select the serial port into which the Sprite is plugged.
3. Navigate to *Tools→Wireless bootloader* and select "Off."
4. Load one of the sample sketches into the development window (simpletest.ino is recommended).
5. Click the right-facing arrow in the upper-left corner of the Arduino window to load the software onto the Sprite.

After completing the above steps, the onboard LED will be blinking on the Sprite. If you unplug the Sprite and carry it to a sunny spot, the solar panels will power the board and the LED will continue to blink.

3 TROUBLESHOOTING

We've listed the solutions to some common issues below. Please let us know if you're met with problems not listed here and they will be included in this list.

1. Why are the solar cells glowing red?: A glowing solar cell means that it has been reverse-biased, and current is flowing in the wrong direction through them (they're actually behaving like an LED when glowing). This doesn't cause catastrophic damage, but repeated reverse biasing can deteriorate the cell. Lower VCC to 2.2 volts when programming to avoid this issue.
2. Why doesn't the Sprite appear under the available ports in the Arduino IDE?: This could be the result of one of a number of things. First, make sure that you have the proper board selected from the *Tools→Board* menu ("panStamp NRG 2 w/ cc430f5137"). If the Sprite still doesn't show up after selecting the proper board, it's possible that the serial port is busy. There are elegant ways to reset a serial port on different operating systems, but rebooting the computer should do the trick for every machine. Try rebooting, re-plugging in the Sprite, and then check the serial port menu again.
3. The program loaded properly according the Arduino IDE, but it isn't running on the Sprite?: Try unplugging the RESET pin from the Sprite and then plugging it back in.
4. The Arduino IDE won't load the program onto the Sprite?: First, turn on verbose mode in the Arduino compiler (*Arduino→Preferences*, then check both boxes for "Show verbose output during: "). If the Sprite is always responding with zeros (which can be seen in the verbose ouput), it's possible that the TX and RX lines are swapped. Also check the *Tools→Port* menu to make sure that the Sprite still appears in the serial port. If not, you may need to switch serial ports, reset the serial port, or reboot the computer.

4 RESOURCES FOR ADVANCED TOPICS

A lot can be learned by playing with the various example sketches. As the Sprite matures, more information will be added to this document. Those looking for information on more advanced topics, wireless programming, and more technical details should refer to the resources listed below:

- *panStamp NRG 2 Wiki*: This wiki contains details on memory organization, API's, bootloaders, and other details of the panStamp NRG 2 on which the Sprite is modeled.
- *panStamp Forum*: This is the equivalent of Stack Overflow for the panStamp boards