A case of 24 Sapphire infrasound nodal sensors ready for deployment. Each tray holds 12 Sapphires. The case is sized to meet most airline carry-on size limits.

SapphireMini Guide

Version 5.4.8 Project Sapphire [1]

2025.03.04

Chris Hayward

Dept Earth Sciences

Southern Methodist University

[1] Project Sapphire's name is a homage to the GEM recorder (Anderson, et al 2017)

*This guide includes both the original Sapphire Mini (**model 1, Serial numbers 1-120**) and the revised Sapphire Mini (**model 2 revision5, Serial numbers 200-650)** versions. Where there is a significant difference, the notes that apply only to the model 2 notes are highlighted in blue and notes that apply only to the model 1 are in green.*

Warnings

If it is necessary to remove the PCB board, use care near the PCB board facing the top cover as there is a slightly protruding 34 gauge needle. While the needle is blunt tipped it is small enough to pierce skin. Also touching the end can transfer enough skin oils that the needle may become plugged.

If the units are operating with lithium primary batteries, use appropriate shipping labels indicating lithium batteries.

Known Issues

1. A config.jsn file that is too long or incorrectly formatted will be ignored.
2. Not all config.jsn parameter combinations will work reliably. Be conservative in changing parameters and test prior to a critical field deployment.

Patch 2

1. ?BDF polarity is inverted
2. BDE is incorrect

Patch 3

1. BDF polarity corrected
2. BDE amplitude corrected

Patch 4

1. Added support for IIM42351 accelerometer
2. Sync IIM42351 to sample clock
3. Added check for corrupted BDF sensor
4. SOH period 0 suspends SOH code
5. GNSS period 0 suspends GNSS code
6. Watchdog check for RTC alarm suppression
7. Traceback diag history corrected
8. Adjusted stack sizes to increase available memory for buffers"

Patch 5

1. Added output error file
2. Added FRAM error traceback

Patch 6

1. Added revision, version, patch to the SOH file headers
2. Added station, network to configuration
3. Changed GNSS first lock to extend time after initial clock lock rather than before

4. Added GNSS high performance tracking for first lock
5. Set GNSS to low power for all but first lock
6. Added hAcc in meters to GPS log messages
7. Added additional delay on BMP388 reset
8. Clear debug history upon SD insert
9. Minor changes to error history, GNSS logging, SOH intervals, and startup LED sequences

Patch 7

1. Added option to delay firstLock logic for N GNSS cycles
2. Reformatted compile time in output messages
3. SD card config.jsn file modifies defaults rather than FRAM. FRAM is used only if config.jsn file is missing or invalid
4. GNSS is started in high performance (high power mode) for getting the initial time. Later sessions are done in low power mode (except for the survey session)

Patch 8

1. Added code to support options ADS1115 ADC
2. Added UBLOX command to save config and restore each power up
3. Made low voltage cutoff FATAL rather than something that causes reboot
4. Added BOD detection to boot loader (requires separate bootloader update)
5. Added GNSS switch to high power if low power won't get PPS in a reasonable time (defined in config)
6. Simplified LED sequencing during startup

Overview

The Sapphire recorder is a simple nodal infrasound recorder designed for short campaigns, especially for infrasound signals in the 0.2-5 Hz band and for stronger signals from 5 to 60 Hz. Data is typically recorded in miniseed with reading converted to physical units such that little post processing other than simple filtering is needed to begin data analysis.

Quick Start

WARNING: It is possible to permanently corrupt the infrasound sensor by resetting the Sapphire using the reset button. This is rare, but known to happen should the reset occur in the short time period that a sample is being taken. For this reason, we suggest resetting the Sapphire by power cycling or if it is necessary to use the reset button, then doing so immediately after power up (for example, in the case of loading new firmware).

WARNING: Pressure changes of more than 3 PSI over less than a few minutes will do permanent damage to the low pressure sensor by either permanently changing the calibration or rendering the sensor completely inoperative, or resulting in erratic readings. For this reason, systems should not be shipped in sealed containers or with caps tightly sealed. When units are shipped with hose caps, the rubber washer normally part of a hose cap should be removed.

Since a large temperature change to a sealed volume also results in a pressure change, this concern also extends to opening sealed containers that have experienced a large delta temperature or subjecting units to temperature swings of very large temperature changes. For systems that are being moved from the office to extreme field conditions (greater than 30℃ [50℉] temperature change), we suggest moving them in the carrying case to field conditions at least several hours before deployment.

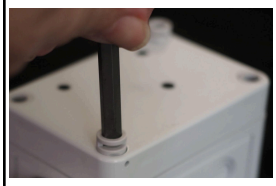
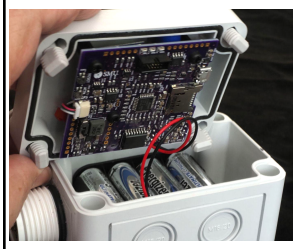
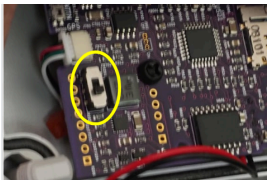
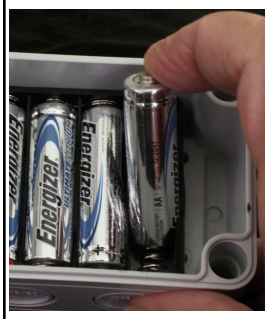
IMPORTANT: *Correct the shipping configuration if needed. Generally this will NOT be necessary except in exceptional circumstances. If the units are in a shipping configuration this should be noted with an attached label.*

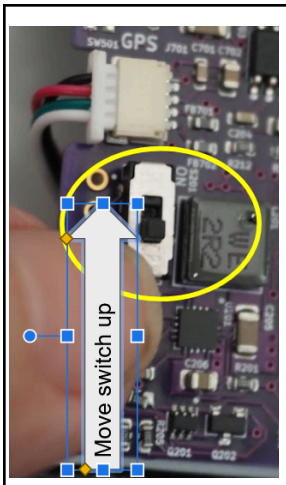
When there is a possibility that the Sapphire may encounter a pressure change of more than a 20kPa (3 PSI) over a few minutes, units may be placed into a shipping configuration by loosening the cap on the backing volume vial. The differential pressure sensor has a proof pressure rating of ~20kPa. Pressures in excess of this may permanently change the sensor calibration or break the sensor.

If the sensors were configured for shipping, there will be a label on the outside indicating that they are in '**Shipping Configuration**'. If there is no such label, you may skip the following paragraph.

Before the first use in the field, the backing vial cap must be carefully tightened. Open the Sapphire by loosening the 4 screws on the top. Remove the 3 screws inside on the PCB securing the PCB to the top so that you can access the back of the PCB. Be careful not to touch the protruding hypodermic needle. Carefully retighten the vial cap while holding the vial to prevent movement. The vial is secured to the PCB with double stick tape or adhesive and it is important to tighten the cap without otherwise moving the vial. Once this is done, the unit will be ready to be used in the field.

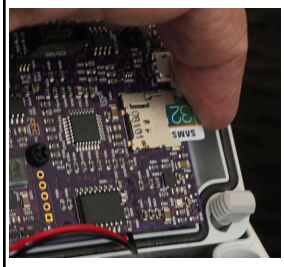
WARNING: The Sapphire may be damaged by excessive static electricity. For this reason, we recommend that users wear natural fabrics especially in dry conditions when working with the Sapphires. This is especially important for winter work where common coats are prone to building up large static charges. Using an antistatic mat on the lab table is also recommended.

	Allow the Sapphire to come to ambient temperature Before turning on power, it's important to allow the Sapphire to come to ambient temperature especially if condensate might form on the PCB. Condensate on the PCB can cause rapid corrosion when power is applied.
	**Loosen the Bolts** Using a flat blade screwdriver loosen the four plastic bolts. These are captive bolts and will stay connected to the lid.
	**Open the Lid** Carefully open the lid to avoid pulling on the two wires connected to the battery pack. The wires should be long enough to allow the top to sit upside down on the table. *Images here are for the M1 Sapphire. The M2 is similar, but components may be is slightly different locations*
	**Set the Switch to Off** Set the switch in the off position. The handle will be away from the nearby white connector. For the M2, there are tiny labels on the board indicating the off and on position.
	**Install 4 fresh AA batteries.** For deployments in cold temperatures or for maximum run time, use Eveready Lithium L91 batteries or rechargeable NiMH batteries. For normal deployments, use inexpensive alkaline AA batteries. We have had good experience Voniko batteries as they are more leak resistant than those commonly found in US stores. Double check the polarity of the batteries as they are installed and make sure they are fully seated (especially the battery at each end of the battery holder).



Turn the power switch on

Turn on the power by moving the slide switch towards the on label. On the M1 Sapphires this will be towards the white connector. On the M2 Sapphires this will be towards the SD card. There is a small white lettered OFF ON label on the PCB.

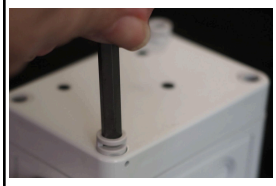


Monitor the LEDs

If you have time, monitor the LEDs for a few minutes. The LEDs will work through several patterns. Eventually, you should see the green LED blink twice per second, the center red LED indicating no GPS lock yet, and the end red LED blinking each time data is written to the SD card.

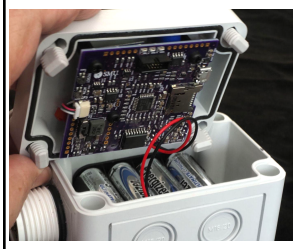
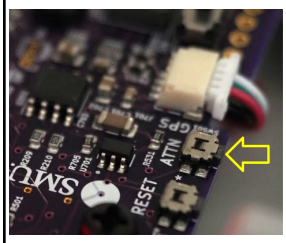
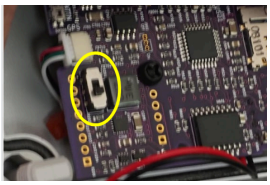
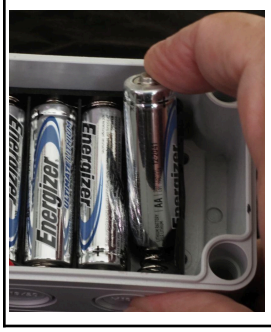
If the system stalls for several minutes with all LEDs on solid, there may have been a bus lock-up that will require a power cycle to clear. In such a case turn the power off, wait 5 minutes, then turn the power back on.

It's also possible that the Sapphire will encounter and error condition on startup. In that case there will be a rapid flash of all three LEDs followed by a code. This will repeat every few seconds as long as the power is on. Power cycling (with 5 minute off time) will reset the system if it can be cleared. If not, consult the Troubleshooting section.



Close the Sapphire

Carefully close the top, making sure not to pinch any protruding wires. The screws should be just firm enough to close the box and engage the gasket. If the top is not evenly closed, it's likely that a wire may have been pinched. Open the case and rearrange wires as needed.

	Deploy the Sapphire Place the unit in an appropriate wind shielded location. If rain is expected during the deployment, tilt hose connection slightly downward to avoid ponding water on the top as this will interfere with GPS signals.
	Retrieve the Sapphire Retrieve the unit when the experiment is complete. It's convenient to leave the unit closed until you get back to the office/lab.
	**In the Lab, allow the unit to come to ambient temperature perform opening the unit** Open the unit. To avoid dirt and water, this is best done in an office, lab, or inside a vehicle.
	**Depress and release the ATTN button.** This will force the current file to be closed correctly. Failure to do this step can result in a loss of the last data prior to a directory update, but early data should be ok. Directory update period is an adjustable parameter, but is typically set at 15 minutes.
	**Turn the power off.**
	**Remove the batteries** If the batteries are exhausted, low or won't be used in the next several weeks, remove and discard them prior to storing the unit. Batteries are inexpensive relative to the total cost of most experiments.

	Recover the Data There are two options to recover the data. One is to simply extract the SD card and read it using a USB micro SD card reader. The other is to plug a micro USB cable into a computer and allow the computer to mount the drive directly. For large datasets, using a USB card reader is faster. The SD card socket includes a retainer and pulling the SD card out (once the push to release is done), requires a bit of careful wiggling from side to side.

Field Considerations

Wind Noise

Wind noise is the dominant interfering noise on infrasound observations. By far the biggest infrasound blinding noise is direct wind on the sensor. While the Sapphire is adapted for porous hose wind reducers, and these are particularly effective below a few Hz, the small size of the Sapphire means that there are other easy (although perhaps less effective) methods to reduce noise on the sensor.

Dense vegetation is an effective reducer and can be used alone or with very short segments of porous hose. Units should be placed on the ground to avoid wind driven motion.

The small size allows the units to be placed in sprinkler valve boxes, typically tied wrapped or taped to the top to provide the least obstructed view of GPS satellites and minimize damage from flooding of the valve box.

Weather Resistance

The Sapphires are not weather sealed. When deployed the hose port is about ½" above the base and water less than ½" is not a concern. Water over 1¼" will enter the box, but should (if everything was sealed correctly) not flood the battery compartment. If used without a porous hose, the major concern is blowing rain or rain splash.

Important:
When connecting to a porous hose the Sapphire should be placed at the highest point of the hose, ideally even elevated on a brick or rock to avoid water running into the enclosure.

In difficult environments, in particular near salt water, units may be sealed in a ziplock bag which will be transparent to infrasound, although it can raise the noise levels if it is directly exposed to the wind.

Boards are not fully conformal coated and systems are also sensitive to condensing humidity. In these conditions, seal the units in zip lock bags with 5 gm packets of indicating silica gel drying agent to provide protection from condensation. The packets should be discarded or renewed at the end of the experiment.

Sapphire M2 units have no conformal coating and therefore should not be exposed to condensing conditions.

GPS Considerations

The GPS is the dominant power consumer in the Sapphire and the time required for a GPS to lock to satellites is dependent upon the GPS signal strength and the number of satellites visible at any one time. For the GPS, the ideal placement is in the open with no additional materials above the lid. This often isn't practical (finding a no wind area with a clear sky view is difficult). In general plastic containers have little effect on the signals. Thin films of water, wet ground above the lid, and heavy wet vegetation obscure the signals and can even result in a complete failure to lock. Other containers, including those of conductive black plastic should be tested.

A GPS antenna concern is ponded water on top of the box. This water can block GPS signals. Leaving the units on a slight incline will minimize the possibility of ponding water.

In general we've had good results for GPS signals even in heavy vegetation, and acceptable GPS locks even for areas with a narrow sky view (such as near tall buildings, inside buildings on window ledges, etc.

If there is a concern about a particular installation for GPS, a test unit can be loaded with the GPS_Serial firmware and satellite SNR can be examined using UBLOX's U-Center software. This does require a field laptop and USB cable, but may provide valuable information for new kinds of installations.

Power Considerations

The GPS can require 140mW while operating although it will be less once all required satellites are located and tracked. Time to achieve a clock synchronization is dominated by the time to acquire GPS signals and is typically a few minutes. Power may be reduced by extending the GPS cycle times or if clock drift can be tolerated (as may be the case for widely spaced deployments) free running without periodic GPS correction, relying on the 2 PPM internal timekeeping clock.

As an alternative to the 4 AA batteries, the Sapphire XL modification provides an external DC power connector. This will accept power from 4 to 13 volt external battery packs. There is also a larger enclosing case that provides room for a 4 D-Cell pack, an 8 D-Cell

pack, or a 6-volt lantern battery. These alternative alkaline battery sources will provide extended run times and shown in the table below. The calculated Run Time Multiplier is the comparison to Duracell Ultrapower AA alkaline batteries, for example 4 D cells should last 5.3 times as long as 4 AA cells. Alkaline battery capacity is reduced in cold temperatures (below 20℃) and is effectively zero below about -20℃ when the electrolyte freezes. Lithium L91 batteries are recommended if AA batteries are to be used. For extended run times with D-Cells (using the XL version), NiMH batteries may be used.

Battery Type	Approximate Capacity	Run Time multiplier
4 Duracell Ultrapower AA	13.6 WHr	1x
4 DuraCell Ultrapower D	72 WHr	5.3x
Eveready MN918 lantern battery	100.8 WHr	7.4x
8 DuraCell Ultrapower D	144 WHr	10.6x

Micro SD Card

After the GPS, the next most power hungry operation is writing to the SD card. SD cards vary in their power efficiency, with some cards requiring more than 3x the power of others. High speed and high capacity cards often require more power than their less expensive low speed varieties. Reducing the number of writes by reducing the sample rate, turning off unneeded channels, using power efficient formats, and avoiding frequent disk synchronization will all help reduce the total power needed.

MicroSDs writes are most efficient after a Sapphire card erase cycle since this will indicate to the internal SD card software that nothing on the card must be preserved. On a 32GB card, this efficiency will continue until the card has encountered 32GB of writes (deleting files does not change the write count significantly). At that point, the card power consumption will increase. Formatting cards on a PC or deleting files will not improve the efficiency, the card must be erased using the Sapphire erase card procedure described above.

We have had good experience with 32GB and 64GB Extreme SanDisk cards.

Data use for 30 day recording

	30 day data use in Gbytes			
	Sample Rate			
Channels	**16**	**32**	**64**	**128**
1	0.2	0.3	0.7	1.4

2	0.3	0.7	1.4	2.8
3	0.5	1.0	2.1	4.2
4	0.7	1.4	2.8	5.6
5	0.9	1.7	3.5	6.9

Total battery life depends on both the battery capacity (about 10,000mW-hrs) for the set of batteries and the peak power drains that have to be delivered. As batteries near exhaustion, it becomes impossible to support the GPS power drains without going to brown out conditions. For this reason, once battery power reaches a critical threshold, GPS correction is suppressed and the remaining battery power is devoted to extending the data collection as far as possible. The cutoff values for GPS suppression and for final shutdown are user selectable parameters.

Temperature

Some critical components in the Sapphire were specified for Industrial Temperature ranges, -40 °C to +85 °C**,** namely the temperature compensated oscillator and microcontroller. However, at the time of purchase, some components were only available in commercial ranges, 0 °C to +60 °C, and therefore operation in cold climates may need to be tested prior to deployments. For cold temperatures, lithium batteries, Eveready L91 should be used in place of alkaline.

The Sapphire, like all differentially based microbarographs, is sensitive to rapid temperature changes. Because the Sapphire differential gauge has a long period response, small changes in temperature due to direct sunlight may appear as long period noise. If long period signals are of concern, the units should not be placed in direct sunlight.

Limits for temperature changes
The low cut acoustic filter, determined by the backing volume leak, for the Sapphires with the standard 34 gauge 0.5 inch needle is over 120 seconds (and can be longer if the needle becomes partly clogged).. A 0.4℃ change in temperature over a few minutes of the backing volume results in 125 Pa change in pressure in a closed volume and will result in an over-range condition indicated by NaN's in the recorded data. The insulation and thermal mass of the enclosure provides some protection from rapid changes, but cannot isolate the differential pressure measurements from sudden changes as a result of for example a hot enclosure suddenly chilled with rain, or strong intermittent shadowning.

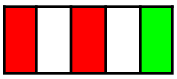
It is possible to decrease the long period response by changing the 34 gauge needle to a 33 or 32 gauge needle, but the instrument corner would need to be evaluated. Because small changes in the needle diameter (within the needle tolerance) will cause perceptible changes in the low cut corner, it's advisable to keep the nominal low cut corner at least one octave from the lowest frequency of interest.

In general, it will be necessary to high pass filter the resultant infrasound BDF channel to eliminate the long period noise.

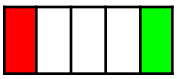
LED Status during normal operation

The three LEDs on the board edge, Green (G), Center Red(C), and Edge Red(E), provide indications as to the current system state. However, they have to be assessed in context. A green LED at the start of the program indicates something different than it would after acquisition starts. The details also may change depending upon the exact firmware version and patch level.

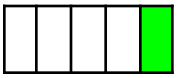
Table of sequence of LED codes while the system is starting up and then operating normally. This does not include the error codes (listed later in the document). LEDs are shown as position RRG. If the LED is on continuous it is shown as R,or G (depending upon color). If it is flashing a F is used. LEDs are shown in position with the green LED on the right.



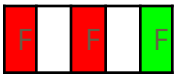
Initial startup. All LEDs on for ¼ second



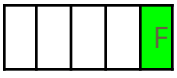
If USB connected, then this indicates READ/WRITE to SD card. Do not disconnect or turn off in this condition. This will be skipped if the USB port is not connected to a computer



If USB connected, then this indicates IDLE and safe to power off or to force a USB unmount by pressing the ATTN button or by issuing eject command from computer. This will be skipped if the USB port is not connected to a computer.

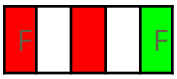


Alternate RED/RED flashing for 5 seconds. Waiting for ATTN button press to clear the configuration. No press keeps the current configuration. A press for 2 - 4 cycles will reset the configuration to defaults, but keep the current serial number. A press for 5 or more cycles will clear the configuration to factory default including setting the serial number to 000.

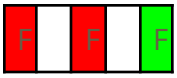


Begin boot sequence. Green will flash 5 times/second during this few minute period

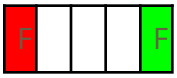
The following two LED patterns will occur during the rest of the acquisition until the system either stops due to an error, is reset (in which the patterns start from the top), or is turned off.



GNSS has not completed its full lock yet, green LED blinks about twice per cycle indicating sampling is happening. Middle red LED indicates GNSS is powered on, end red LED flashes when data is dumped to the SD card.



Middle red LED flashing about once/second indicates GNSS failed to lock on last session and will try again. Green LED blinks about twice per cycle indicating sampling is happening. End red LED flashes when data is dumped to the SD card.



Acquisition is occurring and there has been a successful GNSS lock during the last attempt.

There are additional LED indications if a fault should occur that are shown in the diagnostics tables.

Field Startup

When the weather is cooperative without precipitation or excessive humidity, the field start procedure in the quick start guide is sufficient (ie opening the units in the field, switching on power, and observing if GPS is acquired. Under difficult field conditions, blowing dust, rain, snow, and generally miserable work conditions it is preferable to start the units in a vehicle or lab, seal them, and place them while they continue to take data. However, the initial GPS lock is done within a few minutes of power up, so field placement should be within a few minutes if one will be relying on the recorded GPS locations.

Units subject to hard shocks may shake the batteries loose (it takes a substantial shock).

Maximizing GPS efficiency

On the initial start, the GPS will run for an extended time (determined by the relevant config.jsn parameter) after it acquires a location to provide a more accurate starting location. However, often one wishes to start the unit inside a vehicle to avoid dust and static in the field. Selecting an appropriate 'firstLock_sec' can provide sufficient time to place the unit in position. Even if the unit is moved, successive GPS locks (normally on the hour) will record the position although the location locks will be less accurate (typically around 20-30 meters)..

The GPS included with the Sapphires includes a battery backup that provides an accelerated startup under some conditions. When units have been off for an extended period (days) or are moved substantial distances (>100 km), the initial GPS lock can require more time than normal resulting in multiple failures to lock and increased power use upon the first locks. In addition, when the GPS is allowed to stay on for an extended time, it will track additional satellites and will reduce the time for subsequent uses.

Recording Format

The Sapphire dumps a file named S###_Rnnn.log each time it starts, where ### is the unit serial number and nnn is the run number. The run number is nominally the number of times the unit has booted since the sdcard has been cleared of recorded files. The run number is incremented each time a new log is added to a sdcard with prior *.log files. This file contains the current recording parameters in JSON format (see the section on Recording Parameters for more explanation). This file should be stored with the data in case there are questions about the settings for a particular experiment.

Sample Format

Miniseed records are typically written as 32 bit floats with the conversion to Pa, hPa, ℃, or g already done. A user may optionally record any channel in 32 bit integer, 16 bit integer, Steim1 or Steim2 formats. Formats other than float will result in power savings, but conversion to physical units will have to be done by the user. For the DLHR and accelerometer channel the conversion requires simple rescaling. For the BMP barometer channel, the conversion is more complex and requires calibration constants from each individual chip and scaling using a 12th order polynomial as described in Section 9 of the BMP388 datasheet. Calibration values are found in the LOG channel.

Each recorded channel may use a different recording format. In addition it's possible to record one sensor with multiple formats. This is usually done during testing, for example, to discover if a particular format is suitable for an experiment or to validate conversion factors that are applied externally to integer recordings.

Data Formats	
float	32 bit floats with instrument calibration factors applied to deliver results in physical units. This is most useful for the barometer sensor which requires a more elaborate non-linear instrument correction. The accelerometer and differential pressure gauge require only simple rescaling

int32	32 bit integers that will need to be converted with the listed calibration constants. This results in a small power savings, and produces a file that is the simplest to parse for those writing custom readers.
int24	24 bit integers that will represent the full dynamic range of any of the sensors. In particular both pressure sensors can use up to 24-bits.
int16	16 bit integers that will represent the full dynamic range of the accelerometer. Pressure sensors can clip for large signals.
steim1	Data compression for 32 bit integers. This is a fairly simple compression that results in good power and space savings
steim2	Data compression for 30 bit integers. This compression is recommended for maximum power and space savings.

Timing

Provided the GPS has locked, miniseed records have the start time of the first sample of each block and do not need further correction. When the GPS has never locked, and the internal clock does not have an approximate time,[2] the Sapphire will start with the time 2022-01-01 00:00:00. Therefore some experiments may have a segment of data starting with this time. Once a GPS lock is completed, sampling will be restarted with the correct time. If a later GPS cycle determines that the clock has drifted significantly, sampling will be again be stopped and reinitialized resulting in a small data gap.

The Sapphire M2 includes a battery backed local clock that maintains an approximate current time during power outages. This clock is set upon a GPS jamset. Provided the clock has a valid time, the Sapphire M2 will start with an approximate time until a valid GPS has occurred rather than using the default 2022-01-01. If the backup power is exhausted, the time will be set to a default 2022-01-01.

[2] Sapphire SN1-100 do not have a battery backed internal clock and will reset to 2022-01-01 each time they are switched off. Sapphires 200-600 have a battery backed internal clock that provides an approximate time unit the GPS is able to get its first lock.

For the LOG channel, the time in the miniSEED header represents the time that the particular block of log was dumped and should generally be ignored.[3] There is a timestamp inside the JSON text which indicates the time associated with the log message.

NaN and Flag samples

At times, due to conflicts with various threads running on the Sapphire, there may be a missed sample. In this case the Sapphire will insert a flag value of -128 to indicate a missing sample. This is done to retain the correct timing of succeeding samples. In processing, I replace -128 values with the median or mean of the surrounding samples. A large number of missing samples indicates that some recording parameters may need to be adjusted or that the sdcard is unable to maintain a sufficient write speed.

If the sensors record an out-of-range (clipped) condition, NaN will be inserted into the sample stream.[4]

Channel Names

The Sapphire records in block multiplexed miniSEED format with the possible channels in the following table. Up to 5 channels may be simultaneously recorded. The location code is set to 00 for each channel unless a channel is repeated with more than one format.

Sapphire channel names

Channel Name	Contents
BDF	Differential pressure gauge. This is the traditional infrasound channel. Data is recorded in single precision float format and scaled to Pa.
BDE	Barometer gauge. This is a microbarometer recorded in signal precision float format and scaled to hPa
BDK	This is the temperature of the differential pressure gauge in Centigrade. Data is in single precision float format.
BDL	This is the temperature recorded by the barometer in Centigrade. Data is in single precision float format.
BNZ	These are the accelerometer output

[3] Some miniseed software will complain about the inconsistent sample rates or timestamps in the LOG channel. One should ignore these complaints. The miniseed file is consistent with the miniseed definition.
[4] If out-of-range conditions on the BDF channel persist for the entire experiment period, it indicates a problem in the BDF sensor that may require sensor replacement.

BN1 BN2	recorded in mg in signal precision float format. Most Sapphires omit this channel (accelerometer chips were difficult to source in 2021). Sapphire M2 units all have accelerometers and these channels are available. One or more of these channels may be selected.
BAT	This is the current battery voltage. Note that it is also logged on the SOH channel.
PPS	This is a one pulse per second whose rising edge occurs on the second. This is a test signal used for validating timing.
SIN	This is an approximately 10 second period sine wave used to validate the various recording formats
FIB	This is a test signal whose differences will be a sequence incrementing by 1 (1,2,3...) used for validating various recording formats
LOG	This is a JSON formatted miniseed ascii block containing state-of-health information such as battery voltage and GPS location. It is not recorded at a fixed sample rate and generally has to be read outside of ObsPy. There are Perl scripts that may be used to extract the JSON data.
ADC	If the Sapphire is configured for an external ADC, this will be the ADC channel

File Naming Convention

MiniSEED files are named as S###_Rnnn_yymmdd_Nzzz.msd where ### will be replaced by the Sapphire serial number, and nnn by the run number The yymmdd is the year, month, and day of the clock when the file was opened and zzz will be the file sequence starting with 001 for the first recorded file and incrementing each time. MiniSEED files are never overwritten). The file time associated with the file is the Sapphire time when the file was closed. If the GPS hasn't locked, this time will be 2022-01-01. Otherwise it should reflect the first time block in the file.

Name	Date modified	Type	Size
config.jsn	3/16/2024 8:56 PM	JSN File	2 KB
S201_R001.log	12/31/2021 11:00 PM	Text Document	2 KB
S201_R001_220101_N001.msd	3/17/2024 2:06 AM	MSD File	6 KB
S201_R001_240317_N001.msd	3/17/2024 3:00 AM	MSD File	9,052 KB
S201_R001_240317_N002.msd	3/17/2024 4:00 AM	MSD File	10,133 KB
S201_R001_240317_N003.msd	3/17/2024 5:00 AM	MSD File	10,133 KB
S201_R001_240317_N004.msd	3/17/2024 6:00 AM	MSD File	10,131 KB
S201_R001_240317_N005.msd	3/17/2024 7:00 AM	MSD File	10,131 KB

Output data files. The S201_R001_220101 files are usually the start of each recording session prior to a good starting time from the GPS or internal clock*

Data Recovery Workflow

Since the recording is native miniSEED, there are a variety of tools that can read the raw data directly. Usually the LOG channel has to have some special handling, but other channels are easily adapted.

Tools for examining the miniseed data

System	Libraries and Commands
Python	ObsPy
Matlab	rdmseed[5]
Antelope	miniseed2days, miniseed2db
Geotool	Will read native file, but skips LOG records
PQL	Passcal Quicklook. This is currently an inexpensive commercial product

Python

Someone who knows ObsPy needs to add a short segment here on reading the Sapphire Data

Matlab

Use the community function rdmseed at https://github.com/IPGP/mseed-matlab.

Comment out the 3 lines as shown below regarding WordOrder swaps

```
        case 4
                % --- decoding format: IEEE floating point
                D.EncodingFormatName = {'FLOAT32'};
```

[5] A minor patch must be made to correctly handle floats.

```
                        dd = fread(fid,ceil((D.DataRecordSize -
D.OffsetBeginData)/4),'*float');
%             if xor(~WordOrder,le)
%                         dd = swapbytes(dd);
%             end
              D.d = dd(1:D.NumberSamples);
```

While in matlab, the command 'doc mseed' will list a detailed set of instructions for the function. Then a simple retrieval and plot would be

```
rdmseed('S098N04.msd','plot');
```

Antelope

In Antelope, for efficiency, it is convenient to demultiplex the input miniseed file and build miniseed day files for processing. All files from all Sapphires can be loaded into a single directory, typically called Sapphire_Raw. Then the sequence of commands for preprocessing and displaying the files is:

```
miniseed2days Sapphire_Raw
miniseed2db 2024 Sapphire_db
dbpick Sapphire_db
```

The above assumes that the year is 2024.

Geotool

Geotool is a general seismic trace display tool available to institutions with some connection to the CTBTO. If using geotool, open the file, specify miniseed format, and change the required extension from 'mseed' to 'msd'. The file should be able to be directly read into Geotool.

Working with the LOG channel

The log channel is recorded in miniSEED ascii format and is embedded as a sequence of JSON strings. Although most miniseed libraries can read the ascii file, they often get confused since the miniseed header times are not evenly sampled. Therefore some work has to be done to make sense of the files. The simple PERL script below will strip the ascii out of the miniseed to produce a file that can be examined, printed, or processed with standard JSON libraries.

```
#!/bin/perl -l
#extract LOG file block from miniseed and output in a readable format
#
#
#
$lineN = 0;
$workingtxt = '';
while ($ARGV = shift) {
    open(ARGV,$ARGV)||warn "Couldn't open $ARGV";
    while(read(ARGV,$buf,512)>0) {
      @log = unpack("x15 A3 x2 x10 S x24 a456",$buf);
      if ($log[0]=~/LOG/) {
```

```
          @text = split(//,$log[2]);
          $workingtxt = $workingtxt . join(//,@text[0..$log[1]-1]);
          @curlines = split(/\}\{/,$workingtxt);
          for $i (0..$#curlines-1) {
               $line = '{' . $curlines[$i] . '}';
               print $line;
          }
          $workingtxt = $curlines[-1];
          $line[$lineN] = join(//,@text[0..$log[1]-1]);
          $lineN++;
      }
      }
}
  #$lines = join(//,@line);
  ##$lines =~ s/\000//g;
  #$lines =~ s/}{/}\n\{/g;
  #$lines =~ s/}1{/}\n\{/g;
  #print "$lines";
```

To use the file, copy the above lines to a file named 'extractLog.pl', then execute it as

```
extractLog.pl SN098*.msd
```

While in the directory with the miniSEED files.

Log Variables

Samples of a log file are extracted below

Sapphire Log file data segments

{"gps":{"Time":1640995200,"Lat":0,"Lon":0,"Height":0,"Nfix":0,"OnTime":71,"Slew":0,"SlewRate":0,"Aging":0,"i":01},"i":00} {"power":{"Time":1640995201,"batVolt":5.70919,"current_mA":73,"power_mW":259.6,"busVoltage_V":3.696,"i":02},"i":00}	GPS initializing, no time fix yet
{"gps":{"Time":1640995212,"Lat":328097526,"Lon":-967228401,"Height":116880,"Nfix":1,"OnTime":10,"Slew":0,"SlewRate":0,"Aging":0,"hAcc":10.23,"i":011},"i":00}	GPS has time fix and is now slewing the local clock
{"power":{"Time":1660182135,"batVolt":5.70919,"current_mA":66,"power_mW":321.6,"busVoltage_V":3.612,"i":067},"i":00} {"gps":{"Time":1660182135,"Lat":328097510,"Lon":-967228102,"Height":1168901,"Nfix":4,"OnTime":82,"Slew":-1500,"SlewRate":66,"Aging":-3,"i":067},"i":00} {"power":{"Time":1660182136,"batVolt":5.70919,"current_mA":11.8,"power_mW":321.6,"busVoltage_V":3.612,"i":068},"i":00}	Local time is locked to GPS and GPS is being switched off

The above shows the JSON formatted log messages. Variables are explained in the following table. Variables in purple are not available on the Sapphire M2.

JSON variable names and use

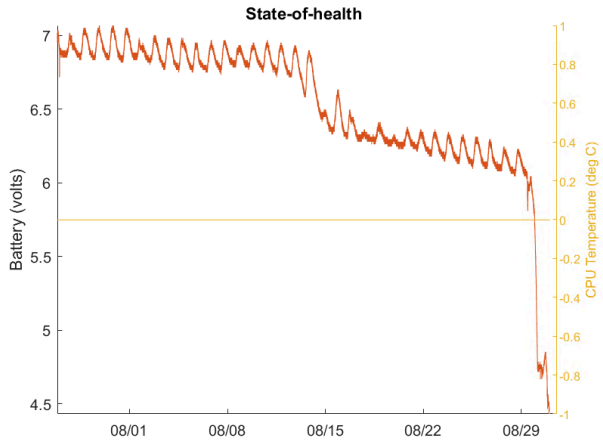
Variable Name	use
Time	POSIX time (seconds from 1970/1/1). This is the local time for this record.
batVolt	Battery voltage as measured by the microprocessor A2D. This is the value used when comparing battery voltage to cutoff levels
temp	Processor temperature in degrees C
current_mA[6]	Current in mA at the battery voltage level. Since the current is only measured when the processor is at high operation levels (not at idle), this only reflects peak power use.
busVoltage	Battery voltage measured at the power monitor chip which should be about the same as that at the microprocessor A2D
power_mW	Power as calculated by the power monitor chip
Lat	Latitude * 1e7
Lon	Longitude * 1e7
Height	Height in m * 1e3
Nfix	Number of valid clock pulses at this time. This number is periodically reset. In general we require Nfix > 4 before believing the time
OnTime	Number of seconds the GPS module has been powered on
Slew	Current calculated slew in nS between the local clock and the GPS time
SlewRate	Current slew rate in nS/S of the local oscillator.

[6] The current_mA, busVoltage, and power_mW have been removed from version 5.4 since they do they are not supported on the Sapphire M2

hAcc	Estimated horizontal accuracy in meters
Aging	A number from -128 to +128 that is used to trim the local clock frequency.
TTL	This is the measured time-to-lock, the number of seconds from power on to the first stable PPS. It will be zero except immediately after the PPS lock.
temp	DS3231 temperature in degrees C
i	Counter starting with 1 that increments for each of this type of message (one index for gps messages and another for power messages)

Monitoring Battery Voltage

Battery voltage extracted from log file



The LOG file records the battery voltage each time state-of-heath is recorded. The figure to the left shows a typical battery voltage for Lithium L91 AA batteries for a deployment with a single shot GPS, two channel recording at 128 SPS. The offset step around 8/14 is a characteristic of L91 batteries as they reach their 50% capacity. The diurnal ripples are the effect of temperature on the battery voltage. Note that this configuration resulted in nearly 30 days of recording.

Recording Parameters

The Sapphire uses a set of recording parameters that may (with caution) be adjusted by the user. When parameters are adjusted by means of a config.jsn file, they replace the default parameters held in non-volatile memory and will be used for all future acquisitions if the config.jsn file is missing or invalid. This guide assumes that all parameters are set to their defaults. If parameters are changed from their defaults they either should be set back to default at the end of the experiment, a note is made for the next user of the changes, **or**

the config.jsn file should remain on the SD card for the next user to confirm the configuration.[7]

There are two file names that control the configuration. The file 'config.jsn' is read first and typically would be used to set parameters common to all units. The file 'config_###.jsn' where ### is the serial number is read last and will override anything previously set. It is typically used to set individual parameters for units, for example extended GPS searches for units in difficult locations.

Almost all testing during firmware development was made with default parameters. There has been relatively little testing with parameters adjusted far from defaults. While we are interested in correcting any errors discovered for defaults that you regard as sensible, we recommend through testing before doing an experiment if you modify parameters. ***You have been warned!***

Examining the current configuration

The current configuration is recorded in the file S###_Rnnn.log where ### is the Sapphire serial number. It is a valid JSON structure. The file is constructed in the following manner:

1. The parameters are loaded from program constant defaults.
2. If there is a file named config.jsn on the SD card, and it has the correct JSON format, it is parsed and any redefined parameters are loaded into FRAM. If there is no config.jsn file, the last saved parameters are loaded from FRAM.
3. If there is a file named config_###.jsn on the SD card it is parsed and any redefined parameters are loaded into NVRAM
4. For a few parameters, the firmware probes the hardware to discover what sensors are connected, the size of the nvram, and the size of the SD card. These values are overwritten upon each execution.
5. The final parameters are written to the *.log file

An example CFG Log file is shown below


```
{
  "version": 5,
  "revision": 4,
  "patch": 0,
  "acq": {
        "sampleRate": [
        128,
        128,
        128,
        128,
        128
        ],
        "missedSample": true,
        "format": [
        "float",
        "float",
        "float",
```


[7] The recommendation is that the config.jsn file and any configXXX.jsn file be left on the SD card if the data is cleared out. If a unit does not have a config.jsn file on the sd card, one may recover the current configuration by operating it for a short time and examining the *.log file.

```
      "float",
      "float"
      ],
      "chan": [
      "BDF",
      "BDE",
      "BNZ",
      "PPS",
      "BDK"
      ],
      "reserveTime": [
      750,
      750,
      750,
      750,
      750
      ]
},
"sdcard": {
      "randomDump_min": [
      2,
      8
      ],
      "syncTime_min": 14,
      "filePeriod_min": 60
},
"gnss": {
      "mode": "cycled",
      "period_min": 15,
      "minOn_sec": 8,
      "searchPeriod_sec": 900,
      "requiredOn_sec": 4,
      "minVoltage": 4.400000095,
      "firstLock_sec": 120
},
"timing": {
      "sync": true,
      "jamsetThreshold_us": 500,
      "trimThreshold_us": 100,
      "requiredTrim_sec": 20,
      "requiredNDelta_sec": 10,
      "agingOffset": 0,
      "microOffset_us": 500,
      "latency_us": 175
},
"debug": {
      "diagInterval": 0,
      "usbOnTime_min": 10,
      "debugMask": 0,
      "minRunTime": 900,
      "maxFiles": 0,
      "incrSerial": 0,
      "maxBoots": 10,
      "factoryReset": false
},
"status": {
      "sohPeriod": 60,
      "serialNo": 201,
      "batteryCutoff_volts": 4.099999905
},
"compiled": {
      "compileTime": 1710622963,
      "optionsFingerprint": 1045373,
      "waitOnInput_sec": 20,
      "rtcPeriod_ms": 0,
      "rtcOffset_ms": 0,
      "sampleOffset_ms": 0
},
```

```
  "probed": {
        "addresses": [
        41,
        118,
        105,
        0,
        104,
        41
        ],
        "sizeFRAM": 256,
        "sizeSD": 64,
        "nboots": 2,
        "bootTime": 1710641150
  }
}
```


The configuration parameters are explained below

Parameter	**Valid Values**	**Use**
version	5	unused
revision	4 or greater	If the version in NVRAM is different than the default version, NVRAM is completely cleared and all parameters revert to their defaults. This may include a serial number reversion to 0000
patch	0-999	If the patch level in the program default is different than that in NVRAM, all of NVRAM is cleared and all parameters revert to their defaults. When NVRAM is cleared, the serial number will also be cleared!
acq		
expName	Unknown	String of 78 or fewer characters with the name of the experiment. This is written into the first SOH record of every miniseed file
station	5 characters	Defaults to SNnnn if set to “”. This will NEVER be preserved in FRAM so if it is not set in the config.jsn or confignnn.jsn it will revert to SNnnn where nnn is the Sapphire serial number
network	2 characters	Defaults to “ZZ”
sampleRate	1,2,4,8,16,32,64,**128**	This is an array, with one sample rate for each of the 5 possible channels. Reduced sample rates have improved self noise and will extend battery life
missedSample	**true**/false	Flag missed samples with -128 if true. If false, duplicate the prior sample to avoid introducing

<table>
<tr><td></td><td></td><td>spikes that have to be removed in processing</td></tr>
<tr><td>format</td><td>float,
int32,
int24,
int16,
steim1,
steim2</td><td>Format for each of the 5 possible recorded channels. Note that this list, like the sampleRate list should list all 5 channels even if they are not going to be used.</td></tr>
<tr><td>chan</td><td>OFF, BDF, BDE, BDK, BDL, BNZ, BN1, BN2, BAT, PPS, SIN, FIB</td><td>Selected sensors to be recorded in a list. The PPS, SIN, and FIB selections are test signals that may not be available with all versions of firmware. Channels may be repeated if it is desired to have multiple formats recorded.</td></tr>
<tr><td>reserveTime</td><td>757</td><td>Number of milliseconds to buffer data. This needs to be large enough that no samples are missed during SD card writes but small enough that there is room in memory for all channels. Use care if changing this and test thoroughly. The maximum value is memory limited and will depend upon the number of active channels and sample rate.
<table>
<tr><th>Sample Rate</th><th>Recommended Value</th></tr>
<tr><td>16</td><td>3000</td></tr>
<tr><td>32</td><td></td></tr>
<tr><td>64</td><td></td></tr>
<tr><td>128</td><td>750</td></tr>
</table></td></tr>
<tr><td>gnss</td><td></td><td></td></tr>
<tr><td>Gnss mode</td><td>oneshot,
cycled,
continuous</td><td>If oneshot, the time is set once at the beginning and then one time later to check for clock drift. If cycled, the GPS is checked every period. If continuous, the GPS is left on all the time. This last option increases the power to over 200mW but will result in a very closely locked time.</td></tr>
<tr><td>startDelay_min</td><td>0</td><td>This is the number of minutes (must be <255) to delay before starting to acquire the GPS signals for the first time. This allows one to start the systems in a lab, and carry them to the field for deployment, without having to open the cases in the field.</td></tr>
</table>

period_min	>3 **60**	Number of minutes between GPS attempts if in cycled mode. If this is set to 0, then the GPS code will be completely suppressed.
minOn_sec	>4 **8**	Minimum GPS on time to lock the local oscillator to the GPS time during the jamset and first stage slews. The second stage slew will double this value. This must be less than 63, but should be kept small for GPS efficiency.
searchPeriod_sec	>60 **600**	Maximum time for the GPS to stay on in one attempt. If there has been no lock in this many seconds, the GPS shuts down until the next cycle time. If an initial lock occurs near the end of the period, the slew and clock adjustment may be terminated early when the timer elapses.
requiredOn_sec	>1 **4**	Minimum time for the GPS date/time to be accepted.
minVoltage	>2 **4.4**	When the supply voltage is below the GNSS minVoltage cutoff, the GPS is quenched. Data will continue to be recorded down to the cutoff voltage. The default is suitable for standard alkaline batteries.
surveyWindow_min	15	Additional time in minutes added to the GPS timing to obtain a good location. This is used to improve the initial location and is used during one GPS session.
surveyCycle	0	If non-zero, this will delay the high precision location determination for N GNSS cycles enabling a user to start all the units in a convenient location and then deploy them allowing the location to be set several hours later
timing		Generally, parameters in the timing paragraph should be left at their defaults. The operation is sensitive to small changes in these and performance has not be tested for other values.
Timing sync	**true**/false	If true the local time is synced to GPS. If false, the local time is allowed to drift. This is sometimes used to rate the stability of the local oscillator. For field experiments it should be left as true.
jamsetThreshold_ms	>50 **100**	When a GPS lock occurs and the time offset is larger than the jamset threshold microseconds,

		sampling is stopped and the time is reset. For offsets under this threshold, the clock is slowly adjusted to the correct time.
agingOffset	-128 - 128 **0**	This is the adjustment for the oscillator crystal aging. This number is only used until a GPS time lock is acquired and can safely be left at zero.
sdcard		
sdcard syncTime	>2 **15**	This is the number of minutes between file directory updates. Effectively data is recorded in syncTime blocks. Hence resetting the system or turning off the system could lose this number of minutes of data. Lower numbers decrease battery life and can result in more missed samples.
filePeriod_min	>5 **240**	New files are created each filePeriod minutes and are aligned on the hour if an even number of hours are used in each file. Closing a file and opening a new one is a power expensive operation.
randomDump_min	0	Randomizes the SD card write interval to provide additional jitter in the internal clocking if tones are problematic. **Recommended to stay at zero!**
debug		
diagInterval	0-255	Number of minutes between thread monitor. This is used only during diagnostics
usbOnTime_min	**12**	This is the number of minutes the USB serial port is held active. If the system is plugged into a computer debug and status messages will be listed as the system runs. A negative number indicates that the unit should stay in high power mode indefinitely. This will quickly exhaust batteries, but is sometimes useful in debugging.
debugMask	0-15	Bit mask to suppress various features. 0x1 suppresses watchdog resets leaving the error code blinking on the LEDs. 0x2 suppresses hard fault resets, writing the error codes into memory 0x4 suppresses SOH thread 0x8 suppresses GNSS thread
maxBoots	0-255 **5**	Maximum number of watchdog timeout reboots allowed. Once this number is exceeded, the

		Sapphire will force an error conditions and blink the LEDs rather than attempting a restart
factoryReset	**false**/true	If set to true, will erase all parameters stored in NVRAM including the serial number. This is typically used to recover from a user blunder that puts the Sapphire in an error state.
minRunTime	0-32767 **900**	Number of seconds of continuous operation to be considered running and set number of reboots (nboots) back to zero
status		
Status sohPeriod	>0 **60**	Seconds between LOG messages. Practically, the log messages need to be frequent enough to capture battery. It will automatically be set to 1 second while the GPS is operating, If the sohPeriod is 0, then the SOH code will be suppressed and no LOG channel will be output.
serialNo	0-999	The serial number is special in that it is retained during any NVRAM update operations. If the current serial number is non-zero, this parameter will be ignored.
batteryCutoff_volts	>2 **4.0**	If during an SOH check the battery voltage drops below the cutoff voltage, the system does an immediate shutdown. This prevents disk errors during brown-out conditions
compiled		
compiled compileTime[8]	Program defined	This is the unix time of the firmware compilation and provide a second check on the firmware version in case someone forgot to change the patch level
optionsFingerprint	Program defined	This is a fingerprint of the compile time options that were used with the code. Combined with the compileTime, version, and revision, this gives a unique fingerprint as to the firmware code.
waitOnInput_sec	0-86400 **20**	This is the number of seconds that the Sapphire will wait for a micro USB cable to be connected to a software serial port. During the wait, the end red LED will flash rapidly. At the end of the wait, the program will continue even if no cable is

[8] Items in purple are defined by the software and are not able to be modified by the user. They are included in the *.LOG file for completeness

		connected.
rtcPeriod_ms	0	unused
rtcOffset_ms	0	unused
sampleOffset_ms	0	Unused
probed		These values are input by the firmware upon every execution. The values starting with addr are the bus address of the various sensors. If an address is zero, then the device does not exist or is not working on this Sapphire.
sizeSD	probed	This is the size of the current SD card in GB.
nboots	probed	This increments from 0 for every watchdog interrupted reboot when the system has been running for less than minRunTime seconds
bootTime	probed	The DS3231 time recorded for this boot. If this is the first boot after power on, the time will be set to the zero time (2022-01-01). Otherwise the time will be set during the last valid GPS jamset and then updated once/second.[9]

Changing the configuration

To change the configuration, allow the Sapphire to run for a short period so that it creates the S###_Rnnn.log file. Edit this file to reflect the needed changes. If you are not familiar with JSON format, you may wish to review the details at https://en.wikipedia.org/wiki/JSON. Only the items you wish to change need to be retained. **The items in purple above should be deleted from the JSON file**. The file should be saved on the micro SD card and named **config.jsn**. For the example below, we will change state-of-health log interval.

The file config.jsn is shown below

```
{
  "status": {
          "sohPeriod": 10
  }
}
```

[9] On the Sapphire M2, time keeping will be preserved for days after the power is removed.

Run the Sapphire a short time and then examine the output CFG###.log file to confirm the change. If the format of the config.jsn file was incorrect, it will be ignored and the CFG###.log file will show no change.

In addition to a file config.jsn, you may also use a file name of config_###.jsn where ### is the serial number of the unit. Typically, one would use the config.jsn file to set global parameters common to every node and the config_###.jsn file to set the parameters applicable only to a particular node.

IMPORTANT
If the config.jsn file format has any errors (misplaced commas, bad parameters, it will be completely ignored and the configuration will remain in the NVRAM state. In addition, a config.jsn file that is too long will be ignored. The sections labeled 'compiled' and 'probed' should be deleted to reduce the size of the file. Don't forget that the last remaining section should not have a following comma.

Resetting the configuration to firmware defaults

It may be useful to reset the configuration to the firmware defaults. This can be done in several ways.

Using the ATTN button

This is done by depressing the ATTN button during startup when the LEDs start their sequence of alternate RED/GREEN flashing. The button must be held down for at least two cycles and released after less than 5 cycles while the LED countdown is occuring. To completely clear all defaults and set back to initial conditions (including a serial number of 00), hold the button down for at least 5 cycles.

Confirm the configuration was set back to default by examining the *.log file. If the configuration was completely cleared, you will need to add a JSON configuration file to reset the serial number.

Using the factory reset option in the config.jsn file

Setting the factory reset to true in the config.jsn file on the SD card will set all parameters back to their default values before loading the rest of the config.jsn values. Typically this would be used if a user wants to avoid inheriting old configuration values from a previous user. It isn't necessary if all config.jsn values will be loaded from the sd card AND if the current serial number is correct

Using a patch level option in the config.jsn file that is different than the current patch level

Updating the firmware to a new patch level will also cause a factory reset when the system first boots and the previous patch level is older. The patch level in the config.jsn file MUST match the new patch level to be accepted.

Connecting to the micro USB

The micro USB connector provides 4 functions

1. It allows power to be provided from the USB port rather than batteries if the power switch is off. This allows certain functions to run without depleting the battery pack, or allows extended use should an external USB battery bank be used
2. It provides USB disk drive emulation during startup to allow configuration files to be quickly inspected or modified and allows small amounts of miniseed data to be downloaded to a computer without needing to extract the micro SD card.
3. It provides a serial connection during firmware startup that will provide diagnostic and state of health information to help isolate problems.
4. It provides access to the boot loader to allow firmware to be replaced or updated.

Power Options

A wide range of battery chemistries may be used. Ubiquitous alkaline primary batteries are convenient for use in temperatures above freezing. We use the Voniko AA batteries as they are more resistant to leaking than many common brands. For cold weather and the most reliable run duration, Eveready L91 lithium primary batteries are recommended. Rechargeable NiMH batteries are also an option for cold weather, but for these it is useful to validate the battery capacity with an advanced charger that will report the mA-Hr of each battery.

If the power switch is in the off position or the battery pack voltage is below 5 volts, power may be taken from the USB connector. Peak power spikes may be as high as 150mA which should be available from most USB connections without problems.

One can also run from an external USB power pack;, however, for some power packs the Sapphire power drain is so small that the pack may switch the port off. In addition, the Sapphire will have no way of knowing that a pack is near exhaustion and therefore may be switched off with little warning. More sophisticated power options are described later in the document, but external power USB may be of use in some conditions.

A few Sapphires (Serial numbers 601-625) have been fitted with an external power adapter that can be used for long duration field deployments. Depending upon the recording characteristics, an EN529 alkaline lantern battery is expected to last up to 200 days in the field. Longer durations could be obtained with larger batteries.

Disk Drive Emulation

If the Sapphire is off and is then plugged into a computer, it will begin USB disk drive emulation. Depending upon the amount of data on the micro SD card, it may take a few minutes to initialize. While in disk drive mode the green LED will remain on. The end red LED will be on when there is pending activity, such as the initial read, or if files are changed or erased.

Disk drive emulation ends when the drive is ejected, either with a command on the attached computer (preferred) or when the ATTN button on the Sapphire is depressed and released. Normal initialization for acquisition begins at that point.

While it is possible to transfer files using the USB connection, this is slower than extracting the SD card and using a SD card reader. The USB drive emulator is most convenient for editing a config.jsn file, for deleting old files, for examining the CFG log file, or for downloading one or two data files.

Serial Monitor Mode

When in normal acquisition, the USB port has a serial port definition and emits a number of program state, diagnostic, and state-of-health messages. These may be viewed with serial

monitor programs. The Arduino IDE is one such package that includes the necessary serial port drivers and monitor software.

Bootloader Mode

If the reset button is depressed twice with about ½ second between presses, the Sapphire will go to bootloader mode. The end red LED will slowly dim and brighten and the GACBOOT disk drive will be available to the connected computer.

Loading new firmware

Loading new firmware into the Sapphire requires a computer with a micro-USB cable and a firmware update in *.uf2 format (from the Sapphire github site). With the power switch in the off position, plug the usb cable into the Sapphire connector, then press the RESET switch twice with about ½ second between presses. Too long or too short of a time will cause the boot process to abort. If the timing is correct, the edge red LED will stay on in a slowly wavering glow. If you elect to abort the load, simply press the reset button once.

Once in boot mode, the Sapphire will appear as a USB disk drive named GACBOOT. There will be 3 files present.

INFO_UF2.TXT simplify identifies the current boot loader

INDEX.HTM points to this Sapphire reference page.

CURRENT.UF2 is a copy of the currently running firmware. You can copy this file to your local computer if you want to save a copy of what is currently running (often useful if you find out you need to revert to an older version).

To load new firmware, simply copy the desired *.uf2 file onto the GACBOOT disk, usually by dragging the file over to the disk.

Updating the boot loader

The boot loader rarely needs to be upgraded. When it is necessary, simply drag the update-bootloader*.uf2 file on the GACBOOT disk. Note that doing so may erase the acquisition firmware and require an acquisition firmware reload as well.

Updating Sapphire Acquisition firmware

Sapphire software release packages may be found at
https://github.com/ChrisHayward-dev/SapphireReleases.git

You should use the most current version UNLESS you've discovered errors and need to revert or you are deploying an array or replicating an experiment and wish to maintain a consistent firmware release.

Using Test Firmware

Included in the releases are various utilities and diagnostic programs that we've found useful from time to time. Notes accompanying them are brief and these will generally only be useful for advanced users that are diagnosing problems. They are included for completeness.

Test_DHLR

The DHLR sensor is the differential microbarograph sensor that measures the pressure between one port and a backing volume. In some cases the backing volume has an additional unplanned leak or the backinig volume needle has become clogged. This program simply dumps the differential readings to the serial port. We use it with the Arduino Serial Plot function to look at real time data as adjustments are made to the backing volume plumbing.

GPS_Serial

The GPS_Serial firmware is a two-way Serial bridge from the USB serial port to the GPS serial port. It allows use of the Ublox U-Center software to examine and operate the GPS receiver. This can be used if one suspects that there has been an antenna failure of the GPS.

Instrument Characteristics

Clock drift

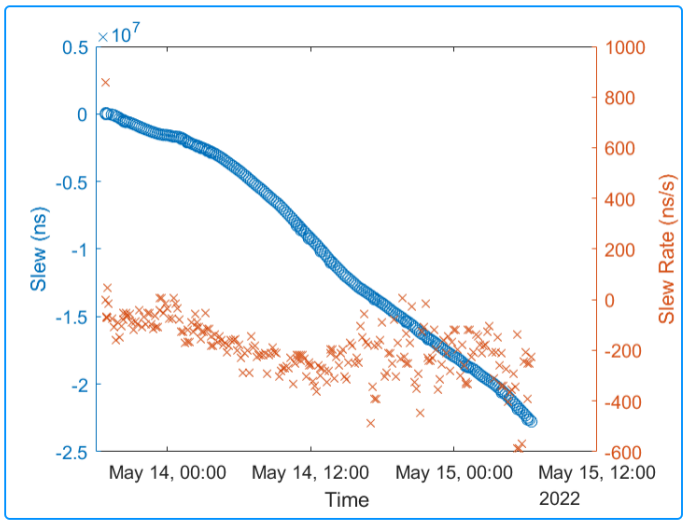
The local oscillator has a manufacturer rated clock drift of 2ppm (or 2 uSec for every second). Provided temperatures do not shift radically from one GPS fix to the next, we usually trim to better than 1ppm. Thus, the maximum drift between GPS fixes is given by the following table.

GPS drive estimates assuming 1ppm clock stability

Maximum Drift in Samples			
	GPS Period (min)		
Sample Rate	**60**	**120**	**240**
16	0.12	0.23	0.46
32	0.23	0.46	0.92
64	0.46	0.92	1.84
128	0.92	1.84	3.69

If the experiment is such that highly accurate sample time is not needed (for example a sparse network interested in microbarom signals), maximum run times will be attained

with the GPS run a single time, at startup, and then left off for the duration of the acquisition.



Observed clock drift on a sample system. The uncorrected clock drift resulted in about 15mS/day, about 0.2 ppm stability, 10x better than the nominal estimate. Actual use in the field will depend upon daily temperature variations. Clock stability can be improved by insulating the enclosure with simple materials.

Power consumption

Estimating Sapphire total run time until the power is exhausted is prone to large errors because much of the power budget depends upon how long it takes the GPS to acquire a lock and this depends not only on the specifics of the antenna on a particular unit (some units have better antennas than others), but also on the placement of the Sapphire and on weather conditions.

Estimated power and run time durations for various configurations for the Sapphire running version 5.2. Sapphire M2 run times for version 5.4 are expected to be similar although they have not been measured

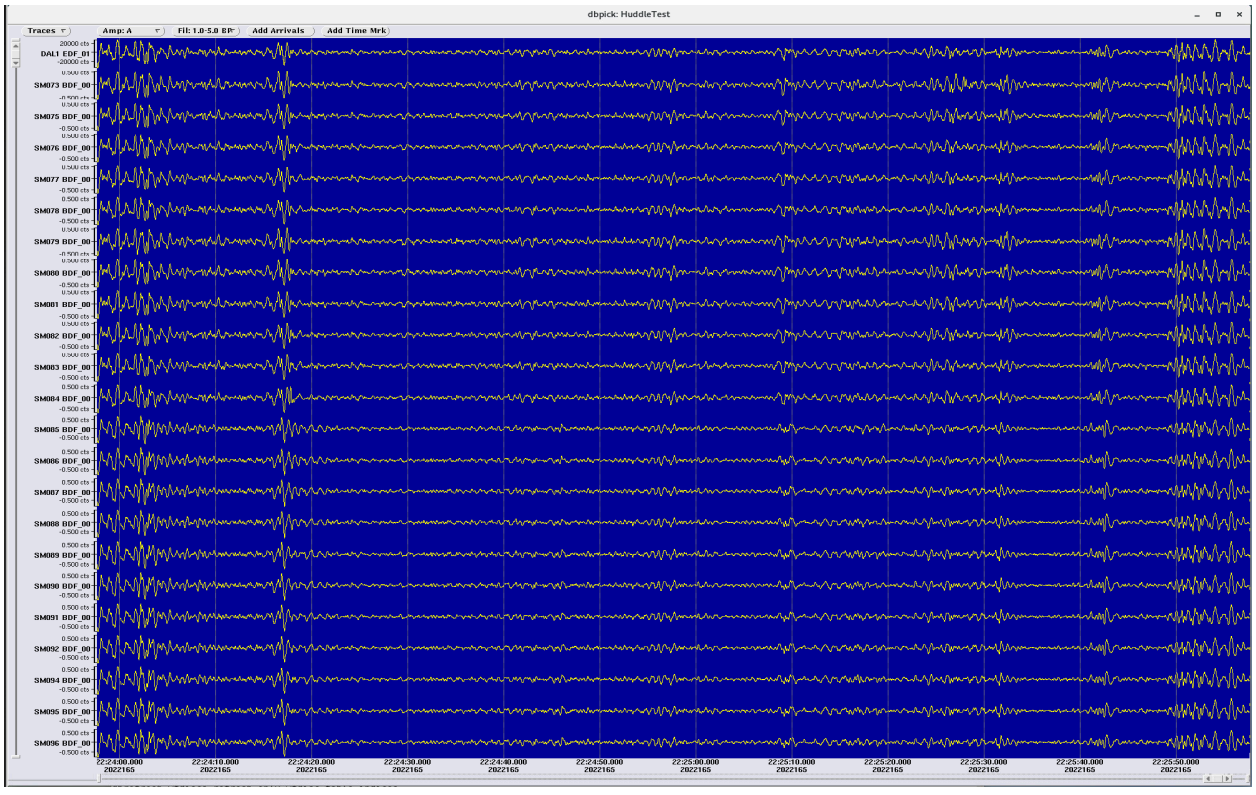
		Power for acquisition only (no GPS)		Days w/ GPS one-shot	Days at GPS cycle times		
Channels	Sample Rate	mA avg @5V	mW		30	60	120
int32 1	16	1.8	8.64	60.7	34.18	43.75	50.87
float 1	16	1.99	9.552	54.9	32.26	40.66	46.74
2	16	2.34	11.232	46.7	29.24	35.98	40.66
3	16	2.36	11.328	46.3	29.09	35.74	40.36
4	16	2.15	10.32	50.9	30.81	38.38	43.75
4	32	1.81	8.688	60.4	34.07	43.58	50.64
4	64	2.74	13.152	39.9	26.42	31.80	35.40
1	128	3.13	15.024	34.9	24.14	28.56	31.43

2	128	3.89	18.672	28.1	20.68	23.83	25.80
3	128	4.04	19.392	27.1	20.11	23.07	24.91

To a smaller extent, the power will depend upon the aggregate sample rate (number of channels times the channel sample rate). Some estimates are given below for an average GPS lock time of 7 minutes and a battery cutoff at 90% of the maximum capacity. Lowest power results from recording in int32 format at 16SPS with only the BDF channel and no GPS, resulting in an estimated run time of 58 days.

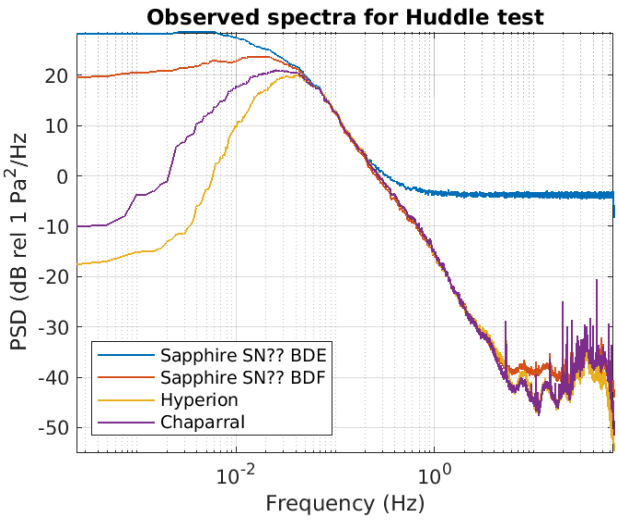
Instrument Characteristics

Vendor data sheets provide a limited description of the two primary sensors. The barometer is a BMP388 or BMP390 depending upon the Sapphire. The differential pressure sensor is a DLHR F50D gauge. Both sensors include internal factory calibration values. However, once the systems are integrated into the Sapphire, additional noise factors and characteristics may come into play. For this reason, we suggest huddle testing units before and after any critical data acquisition experiments.



The above figure shows a typical huddle test of the DLHR sensors for most of the units in one box. A reference trace from a Hyperion sensor is shown as the first trace (labeled DAL1). Traces have been bandpass filtered 1-5 Hz.

The nominal pressure range from the DLHR F50D is ±125Pa (0.5in water). Pressures beyond this range result in a NaN output. Pressures beyond ±25kPa may permanently shift the calibration of the unit and pressures beyond ±75kPa may burst the diaphragm.

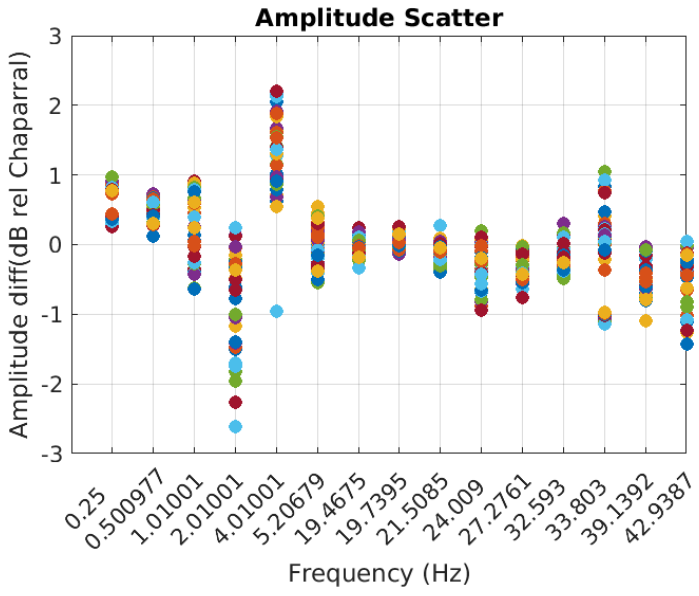


Sample spectra for one Sapphire compared to a Hyperion and Chaparral

For consistency among experiments, we typically declare one sensor (in this case a Hyperion) as a reference and compare all units against that in a typical operating environment (in this case a continually operating campus infrasound site). If such a site is not easily accessible, the internal BMP388 barometer can be compared to the DLHR sensor in the 0.01 to 0.1 Hz band.

The observed spectra for one example Sapphire is shown below. The blue curve is the spectra for the BDE, BMP388, barometer channel. The system self noise begins to dominate the spectra at frequencies above 0.1 Hz. Below 0.1 Hz to about 50 seconds, the BDE and BDF channels should match. Above 0.1 Hz the Hyperion and Chaparral spectra are identical and overlay the Sapphire up to about 3 Hz. At this site, beyond 3Hz Sapphire instrument noise level interferes with low amplitude areas, but the Sapphire does resolve most of the line spectra that are above the self noise of about -40dB rel 1 Pa2/Hz. DHLR self noise depends upon the sample rate and low rates reduce sensor self noise.

Amplitude scatter

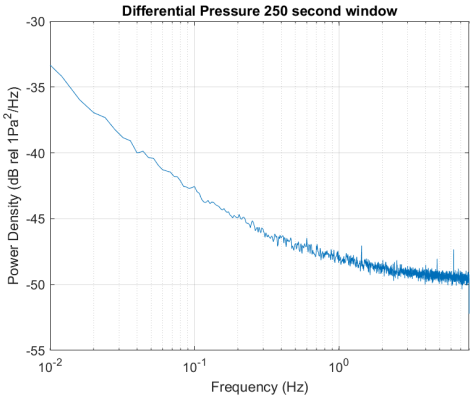


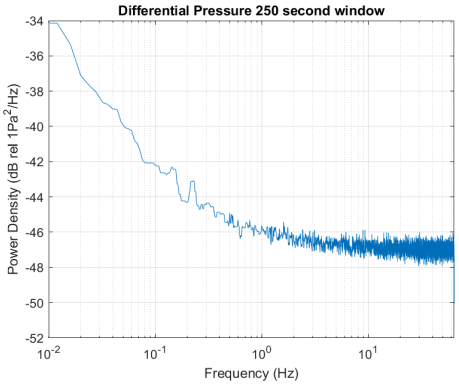
Thirty-eight Sapphire units were compared with the Chaparral at specific frequencies (the arbitrary 0.25 to 4.0 Hz and the 10 highest power peaks for line spectra in the 5-60Hz band. For good SNR peaks (those above 5 Hz), measured amplitudes are within +/- 1dB. For the arbitrary frequencies in low SNR conditions, differing noise causes spectral amplitudes to scatter more broadly, up to 3dB. Below 0.5 Hz, units are

within 1dB of each other and this is the expected range for any signals with good SNR.

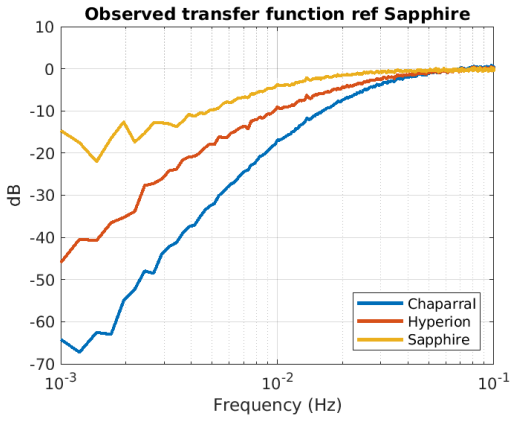
DLHR Response

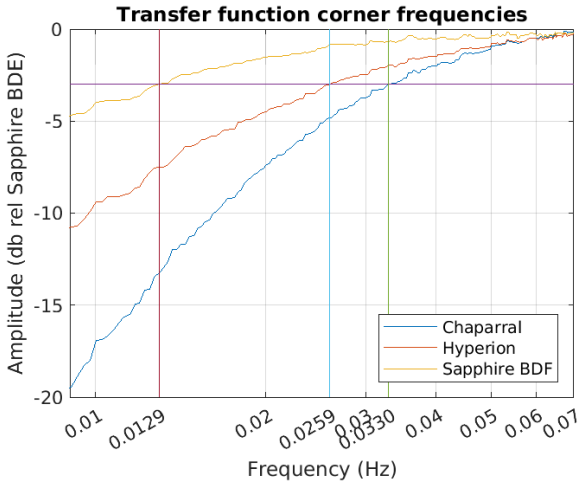
Noise levels for the DLHR sensor are dependent upon the sample rate, the ambient temperature stability, and the system power mode. The best noise performance requires the processor to run in low power mode. This only occurs when the USB connection is off..

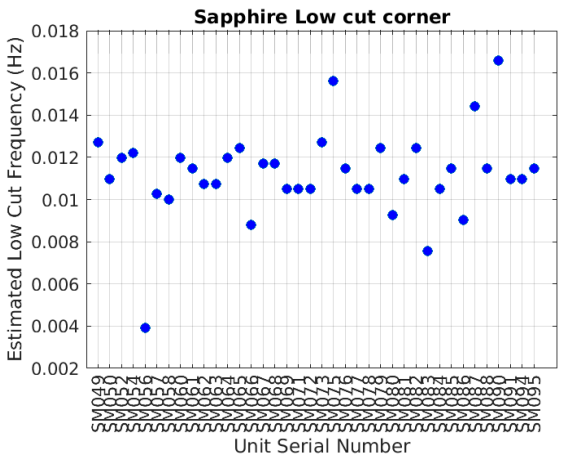




At 16SPS, the high frequency instrument noise is roughly -50 dB rel to 1 Pa^2/Hz. At 128 SPS the noise is slightly higher..



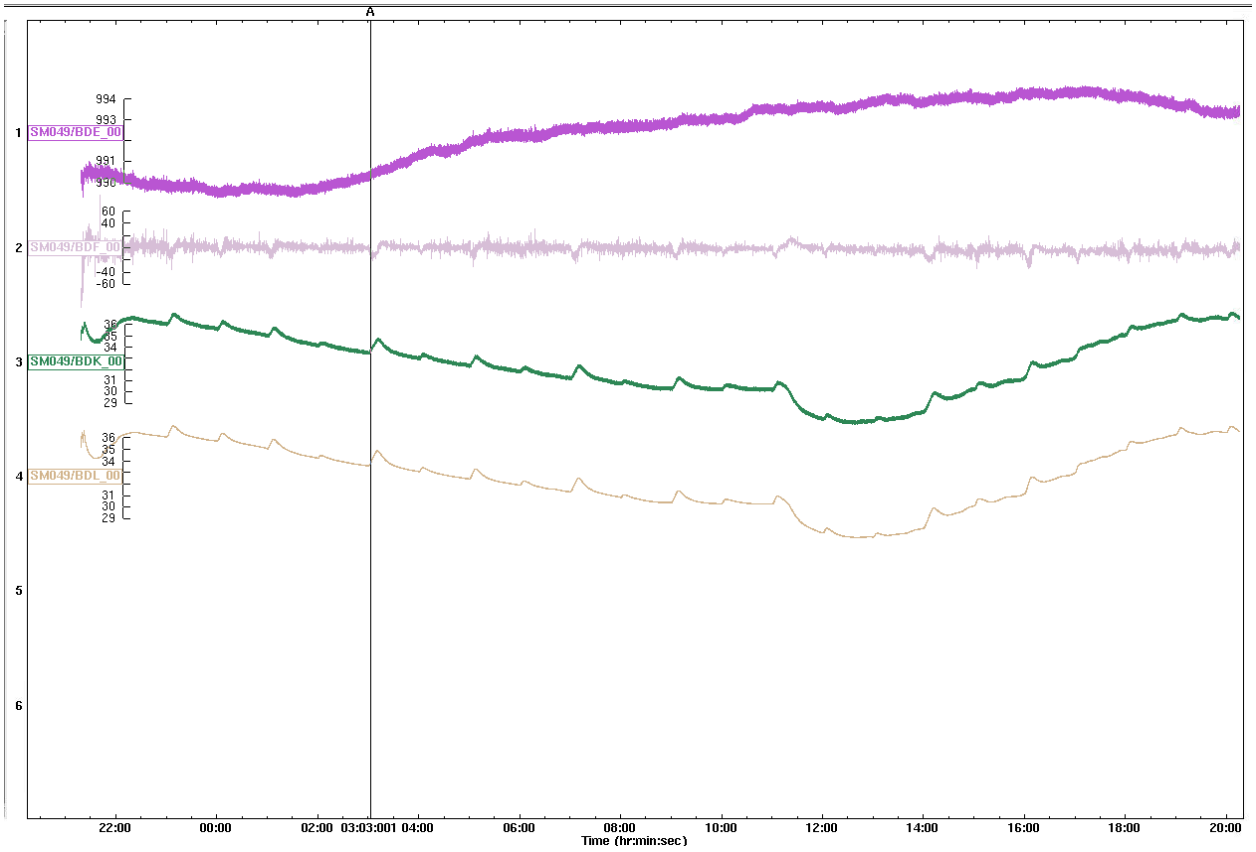




The frequency response of the DLHR differential sensor can be determined by comparing it with the BMP388 barometer for frequencies below 0.1 Hz. The figure shows the Chaparral Model 2.3 and Hyperion 3000 sensors for comparison. The first low cut corner is determined as the -3dB point. Because the low cut corner is dependent on details in the leak capillary and this results in some scatter at the corner.

The DLHR sensor and backing volume are sensitive to operation of the GPS module, a

temperature related effect. Fortunately most temperature changes are slow and outside the band of interest to infrasound research. This temperature effect appears as a long period pulse on the BDF channel. It can be recognized as a GPS induced noise by comparison to the BDF barometer channel which will not show the same feature. In addition, it will occur only during GPS activity as shown on the LOG channel and usually the temperature effect will be recorded on the BDK and BDL channels (if they are being recorded).



A 0.01 Hz low cut filter will eliminate this noise. One can use the BDE channel for observations below 0.1 Hz.

Temperature Sensors

Both the BMP388 and DLHR sensors include temperature sensors that are used to temperature compensate pressure readings. The BMP388 (BDL) channel seems to be a lower noise measurement.

The BDK channel (DLHR) temperature has a typical ±0.05 ℃ noise while the BDL channel (BMP388) is lower noise with a typical ±0.003 ℃ noise level.

Modifications

Firmware Supported Modifications

The Sapphire fully supports a few minor hardware modifications. These have been useful during testing and may have some use for advanced users who want to experiment with different deployments.

External Power

The Sapphire uses a buck regulated power supply capable of input voltages between 3.5 and 15 volts. A standard 12-volt battery may be connected in place of the AA battery pack to provide extended run times. This requires drilling a hole in the case for the power connector. In such use, if a hose connection is in use, the connector should be made airtight such that it does not introduce an unwanted acoustic connection into the Sapphire.

External power switch

The PCB includes a 3 pin connection at J502 that will support a second power switch. Shorting the left two pins of J502 will force the power supply off. Adding a magnetic reed switch would allow one to quickly turn off or on the Sapphire without opening the box. This may be of interest in difficult deployments.

This option is NOT available on the Sapphire M2

Plug in sensors

The Sapphire supports a plug in LIS3DH accelerometer on J504 to record vertical acceleration on a 5th channel. It will also support a plug in BMP388 or BMP390 sensor on J506. Both of these sensors are also supported on the STEMMA connector at J507.

The Sapphire M2 has an unpopulated STEMMA connector on the back side of the board that may be used for additional I2C expansion.

External GPS

The current Sapphire version uses a Beitian 220 GPS, a module designed for small drones. Other Ublox modules may be used and connected via J703. This will allow modules with PPS outputs to be easily integrated.

The Sapphire M2 uses a board mounted UBlox module as a GPS. There is an unpopulated connector (J805) on the back of the board that provides access to the GPS serial ports should one wish to remove the UBlox module and revert to a Beitian or other GPS.

Hardware Supported Modifications

The Sapphire includes several accessible hardware bus connections that are available for future firmware or modifications. Some may be of interest to advanced uses.

Plug in I2C modules

The STEMMA connectors provide access to the I2C bus. Since this bus is shared with the other sensors, care must be used to avoid interfering with normal sampling, but additional sensors have been successfully tested.

Analog Microphone A2D

The J504 connector may be reconfigured to support an analog microphone A2D. This has been prototyped, but not produced. It is designed to support higher frequency sampling at lower noise levels.

External high capacity FRAM

Connector J501 supports Adafruit high capacity FRAM on the SPI bus. This is for future expansion.

Expansion Connector for WINC1500

There is a 0.1" pin connector to support an Adafruit WINC1500 WiFi expansion board to provide WiFi access to the Sapphire M2.

Other Expansion and Diagnostic Connectors

Connector J802, a 5x2 unpopulated connector provides access to the chip debug pins and is used to do the initial flash of the boot loader. Connector J803 provides access to the SPI bus used for the SD card and access to a second unused SPI bus as well as a spare pin.

Future Modifications

There are several future firmware modifications being considered as a result of user field experiences. These are:

1. Scheduled window recordings. This would allow the system to go into deep sleep and awaken at specific times.

2. Real time stream. This would allow a miniseed reader to pull data from the Sapphire in real-time.

We welcome suggestions for changes or enhancements. Such user input is extremely valuable not only for further development of this device, but also for other similar community developed devices.

Troubleshooting

The Sapphire is a simple device, so most problems are either simply solved or impossible to fix in the field. For that reason, it's recommended that when an experiment is planned, a few systems are kept in reserve in case one or more fail to start. If everything works, these extra systems can be deployed in parallel at critical sites, or used for secondary sites.

Recovering from problems in the field

If the hardware is intact (not broken) and the correct firmware is loaded, then there are only a few things that cause unrecoverable and repeated errors. The batteries could be low; the GPS antenna could be blocked by something outside the box; the SD card could have an unrecoverable error; or the FRAM could have an invalid configuration. Loading fresh batteries will address the low power issues. If the SD card has an error, any data on it should be recovered on a PC and then the card may be reformatted on the computer or for lowest power use using the procedure outlined in the quick start section. If the FRAM has an invalid condition, it may be reset to default configuration or to factory zero condition using the procedures above. If this is done in the field, without adding a JSON configuration file, it may result in resetting the SN to 0000 and the acquisition parameters to a default condition. It's useful to label such a system so that it may be noted when the data is extracted.

If none of this fixes the problem, it may be necessary to do more advanced troubleshooting by examining the files on the micro SD card and/or connecting a computer to the microUSB connector and running a serial console to examine the error messages.

Troubleshooting table

Symptom	Cause
When power cycled no LEDs ever turn on	Check battery condition. The system should operate when the battery pack voltage is above 4.5 volts. Install fresh batteries, verify that each cell is firmly seated and that all polarities are correct. Also check the battery pack cable. Sometimes these have been crimped between the top and bottom cutting the conductors inside the insulation.

End red LED rapid flashing during startup	The system is awaiting a USB console connection. This is normal and should time out within a few minutes.
All three LEDs on continuously for over a minute	Bus lockup. Turn the unit off for at least 3 minutes before powering back on. If this error occurs repeatedly it may indicate a failed sensor.
Middle red LED on during acquisition	GPS hasn't acquired a lock. If it has been more than 15 minutes, the GPS has given up and will not try again for 1 hour (depending upon the setting in the JSON file). To encourage the lock moving the unit to a new location or restarting by pushing the reset button. If this has occurred after several hours of acquisition, but was ok at first, this indicates that at the last attempt, GPS was not found. The data is likely ok, but there may be some clock drift depending upon how long the GPS was unsuccessful. Usually the unit will recover upon the next GPS window.
All three lights flash 4 times, then a light pattern flashes. This repeats once every few seconds.	This is a runtime error. The system has shut down completely. The pattern is an octal code with the green LED the low order bit (green = 1, middle red=2, end red=4. A zero is 4 flashes of all the lights. Read the code, and consult the table in this document for more information on the error. For most errors, you may also get a description of the error by restarting the Sapphire (power cycling) and waiting 5 minutes. At that point, if the SD card is writable, the Sapphire will create a file PreviousErrors.txt that will have a one line description of the error. The most common non-firmware errors are a low battery, a SD card write error, or a bad configuration. Firmware errors should be passed along to hayward@smu.edu for investigation. If you

	have modified the recording parameters using a custom config file, you may want to revert back to default parameters and verify the error is not related to a bad choice of recording parameters.

Sample Error File

If a Sapphire encounters an error, it will save the error in internal memory. When the unit is restarted, if there is an error recorded in memory, it will be written to the SD card in a file named 'PreviousErrors.txt'. This file, if it exists, is appended upon each restart, so it is possible for multiple errors to be listed. The example output shown is an uncorrectable hardware error that requires a chip replacement.

Sample output of PreviousErrors.txt file

```
Error File Traceback
AA Mini Logger Version 5r4p5 Oct  3 2024 13:20:42
Most recent Error:
--------------------
Previous fatal error:  after time 2000-00-255T255:255:255
This unit has a corrupted low pressure sensor (BDF)
Fault Dump in sequential order
Diag History Buffer:
Line# Mask
284 4
30 0
38 9
39 9
40 9
43 9
52 9
53 9
70 9
723 7
724 7
768 7
50 10
203 10
222 10
227 255
====================
```

Problem Diagnostics

Testing the DLHR sensor

Testing the GPS module

LED Error Codes

Error codes reflect only the applicable firmware version. Tables are regenerated upon each firmware update and are found in the ./src directory in the relevant github project. Each table is dated and may be matched with the date of the software (printed during the initial startup on the USB serial port. If you have a different firmware version, codes may be slightly different as source code lines move around when new sections are inserted.

Error codes will blink repeatedly on the LEDs. You can force an error by starting the unit and letting it run for a few minutes, then extracting the SD card while the unit runs (this may corrupt the file system requiring that the card be reformatted). The unit will flash an error code about SD card errors as found in the following tables

Codes are octal numbers. A zero is represented by 4 quick flashes with all LEDs. Other digits are the LEDs as positions. For example a single green LED is 1, the middle red LED is 2, and the edge red LED is 4. Digits 5-7 are represented by multiple LEDs,

LED	DIGIT
rrg	0
--G	1
-R-	2
-RG	3
R--	4
R-G	5
RR-	6
RRG	7

Sample Table of Error Codes for a particular release

Octal	Line	Source File	Error
00074	0060	SM_miniseed2.ino	SD Init failed with 0x0000 0
00100	0064	SM_miniseed2.ino	Can't find valid FAT partition.
00111	0073	SM_miniseed2.ino	Unable to open file!
00152	0106	SM_sdcard.cpp	
00156	0110	SM_sdcard.cpp	DEBUG: Low Power Mode
00226	0150	Threads.ino	LOG overflow on DLHR header
00230	0152	Threads.ino	LOG overflow on BMP header
00232	0154	Threads.ino	LOG overflow on ACC header
00241	0161	Sensors.ino	IN219 bus hung
00250	0168	Sensors.ino	IN219 bus hung
00271	0185	Sensors.ino	SemaphoreWire blocked

```
00322  0210     Sensors.ino        DLHR Not detected!
00343  0227     Sensors.ino        This unit has a corrupted low pressure sensor (BDF)
00362  0242     Mini_Logger5.4.ino         Inconsistent patch level in SM_config defaults!!!
00365  0245     Mini_Logger5.4.ino         Unable to parse JSON version override
00470  0312     Threads.ino        Unable to get a jamset lock!
00531  0345     Sensors.ino        BMP decimation rate must be less than or equal to BMP_T rate
00556  0366     Sensors.ino        Error accessing BMP sensor at 0x00
00604  0388     SM_miniseed_lib.cpp        Unable to write header
00633  0411     SM_miniseed_lib.cpp        Write fail at sample 0 of  (0)
00674  0444     SM_ublox.cpp               Restart Message!
01156  0622     Sensors.ino        Error accessing LIS2DH12
01176  0638     Threads.ino        Low Voltage Cutoff
01225  0661     Sensors.ino        No acceleromter found
01240  0672     Sensors.ino        Unknown accelerometer
01324  0724     Sensors.ino        Unknown accelerometer: 0x0
01403  0771     Threads.ino        I2C LOCKUP!
01452  0810     Threads.ino        Unknown or unapplicable output format requested!
01507  0839     Threads.ino        RTC Timeout!: 0 ticks
01525  0853     Threads.ino        Overrun 0
01536  0862     Threads.ino        Less than 1 sample
01554  0876     Threads.ino        Sample Timeout
01645  0933     Threads.ino        Unknown format!
01660  0944     Threads.ino        Sample Overrun! (0 of 0) @ 0(RTCsampleCount)
01724  0980     Threads.ino        Diag Task is blocked!
02045  1061     Threads.ino        Diag task blocked
```

Diagnostic Traceback

In the event of a serious error, such as a watchdog timeout or processor hardfault, the system will attempt to provide a traceback of the most recent debug lines listed in oldest to youngest order (the final error should be at the bottom of the list). Interpreting this traceback may require access to the source code although the tables below will provide an indication of the actions leading up to the error. This traceback information is saved into program memory for the first fault since the firmware was loaded and will not be overwritten should later errors occur. To clear the error report, new firmware must be loaded into the system. While the error report will remain frozen in memory, a system restart will continue to collect data. A sample traceback is shown below.

```
I2C Status:Ok (SDA and SCL are high)
Adding: INFO
Adding: SETUP
Adding: ERROR
Previous Fault Dump
--------------------
Diag History Buffer:
Line# Mask
230 4
235 4
30 0
244 1
81 8
82 8
83 8
```

```
86 8
89 8
91 8
455 7
248 7
362 8
365 8
374 8
510 7
====================
```

The mask refers to the debug mask which is generally unique for each source file. The line number is the source line number. Debug masks are assigned during program initialization and can change if significant changes are made to the order in which routines are first called.

Mask	Source
0	Information only. May occur in any file
1	Setup
2	
3	Ublox code
4	SD card code
5	FRAM code
6	Configuration code
7	Multi-Thread code
8	Clock sync code

The sample list of trace statements is listed below for this particular firmware release. The full table, like the error table, is found in the ./src directory.

Sample Table of Debug Trace statements

Octal	Line	Source File	Error
00004	0004	SM_fram.cpp	
00010	0008	SM_ublox.cpp	
00013	0011	SM_miniseed_steim.cpp	Begin Steim
00014	0012	SM_config.cpp	
00015	0013	SM_config.cpp	Memory Free: 0
00022	0018	Sensors.ino	
00024	0020	SFFS.cpp	

```
                00024   0020        SM_sdcard.cpp
00025   0021        SFFS.cpp
00025   0021        SM_miniseed_steim.cpp   Allocated memory for Steim1 Frames
00025   0021        SM_sdcard.cpp
00027   0023        SFFS.cpp
00030   0024        SFFS.cpp
00031   0025        SM_fram.cpp             FRAM: Volume not initialized!
00031   0025        SM_miniseed2.ino
00032   0026        SM_clockSync.cpp
00033   0027        SM_fram.cpp             FRAM: Unable to create new FRAM volume
00036   0030        SFFS.cpp            *** num = 0x
00036   0030        SM_sdcard.cpp           SD file update time: 0/00/00T00:00:00
00037   0031        SFFS.cpp
00037   0031        SM_fram.cpp             FRAM volume: , available 0 of 0
00040   0032        Threads.ino
00041   0033        SM_fram.cpp             WARNING: FRAM VolumeName is , expected it to be
00046   0038        SM_clockSync.cpp        Begin clocksync
00047   0039        SM_clockSync.cpp        Using clock frequency of 0 Hz
00050   0040        SM_clockSync.cpp        Using 0.000 ns/tick
00053   0043        SM_clockSync.cpp        Wire.begin
00054   0044        SM_fram.cpp             FRAM: Unable to initialize FRAM!
00055   0045        SM_sdcard.cpp           Begin sdcard
00060   0048        SM_fram.cpp             FRAM: Unable to create new FRAM volume
00061   0049        SM_clockSync.cpp        No local clock preserved
00062   0050        SM_miniseed_steim.cpp   Clear Steim
00063   0051        Sensors.ino         Setup
```

Drawings and Schematics

These are available as separate PDF files

Enclosure Drawings

Circuit schematics

PCB Layout

PCB Copper