Detailed Data Acquisition and Calibration Methodology for MSNSound

May 30, 2024

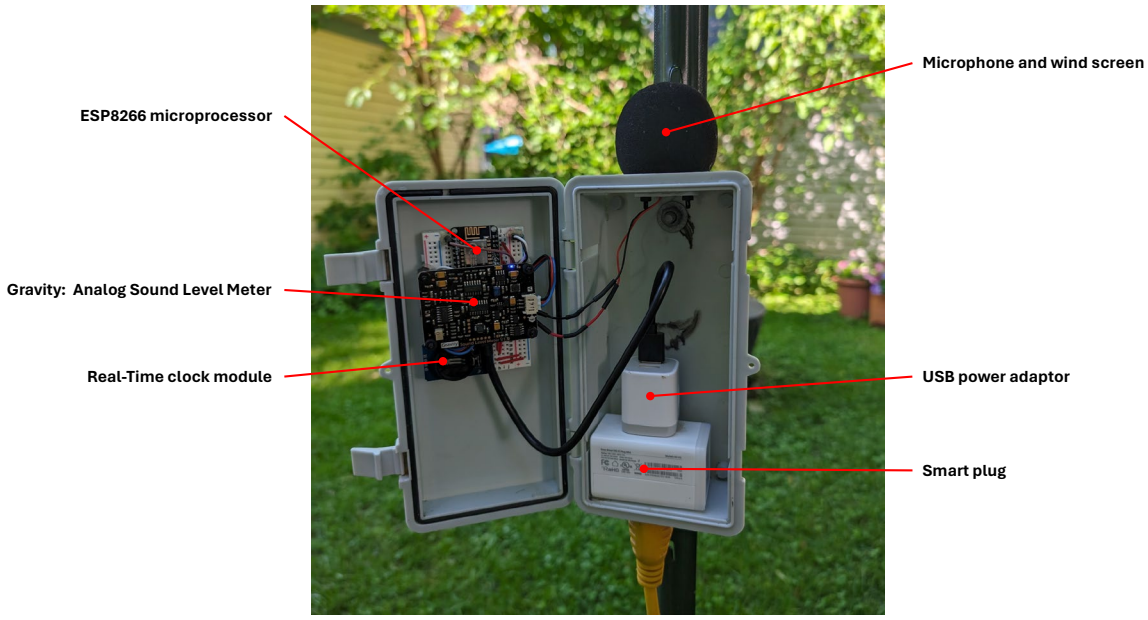
This document describes the details of how sound level and aircraft data are acquired and stored for the MSNSound network of noise meters.

Sound Monitoring

Hardware

The key hardware components for each monitoring station are shown in Figure 1. A sound level meter captures decibel levels at the meter location and converts them to a DC voltage signal that is read by the microprocessor and transmitted to a central server. A real-time clock (RTC) provides time stamps for the data. A USB adaptor and smart plug provide power for the system and allow the unit to be remotely power-cycled if needed.

Figure 1. Key components of an MSNSound monitoring station.



Sound Level Meters

Two types of sound level meters are used for the fundamental sensing of decibel levels at the monitoring locations:

- Gain Express SLM 24
- Gravity: Analog Sound Level Meter, manufactured by DFRobot

The manufacturer's specifications for both meters list a measurement range of 30 to 130 dBA with an accuracy of ±1.5 dB, and each type of meter provides an analog DC voltage output that is proportional to A-weighted decibels. In each case, the output voltage output from the meter is updated every 125 milliseconds (corresponding to a standard 1/8th second fast-response reading) and held constant between updates.

The Gain Express meter (Figure 2) is intended as a hand-held meter but can be powered indefinitely via a USB power input at the bottom of the meter. The meter can also be configured to read C-weighted decibels, but defaults to A-weighting when initially powered.

Figure 2. Gain Express SLM24 Sound Level Meter.

The DFRobot unit is a bare-board device (Figure 3). For the MSNSound outdoor application, the microphone is desoldered from the circuit board and relocated to the end of a 3D-printed, upward-facing microphone extension on top of the field enclosure (Figure 4).

Figure 3. Gravity: Analog Sound Level Meter.

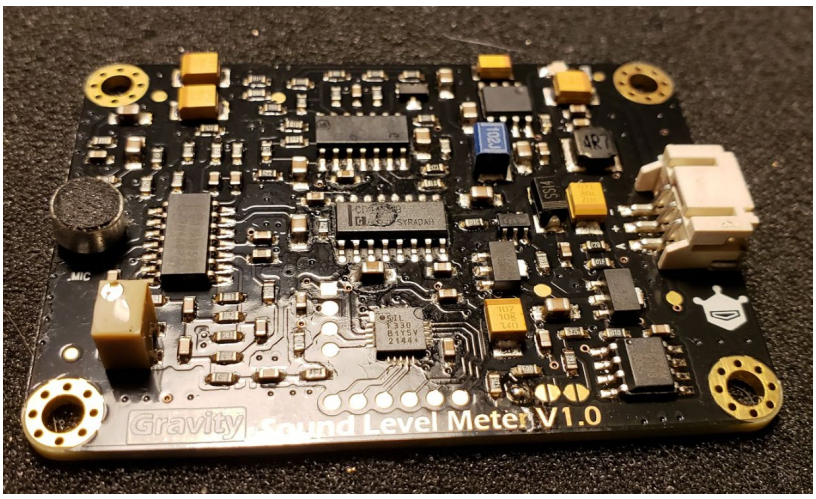
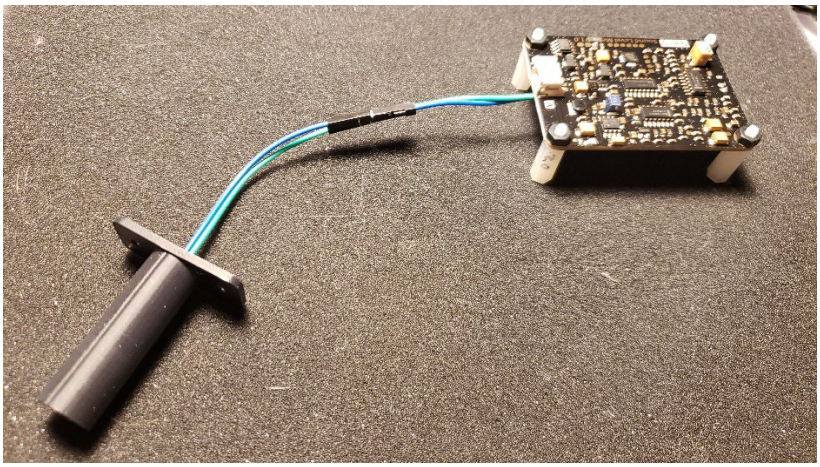


Figure 4. Gravity: Analog Sound Level Meter with microphone relocated for mounting externally to monitoring enclosure.



Microprocessor

The microprocessor used is an Espressif ESP8266-based development board manufactured by KeeYees (ESP8266 ESP-12E NodeMCU), and flashed with micropython.

Real-Time Clock

The real-time clock used to maintain time accuracy is a DS3231-based AT24C32 IIC RTC Module Clock Timer, with a listed accuracy of ±2ppm for temperatures from 0°C to +40°C and ±3.5ppm for temperatures from -40°C to +85°C. The RTC is interfaced with the microprocessor via I2C protocol.

Smart Plug

Power to both the sound level meter and microprocessor is delivered through a USB adaptor plugged into a Kasa HS103 smart plug. This facilitates remote manual and automated rebooting of both devices.

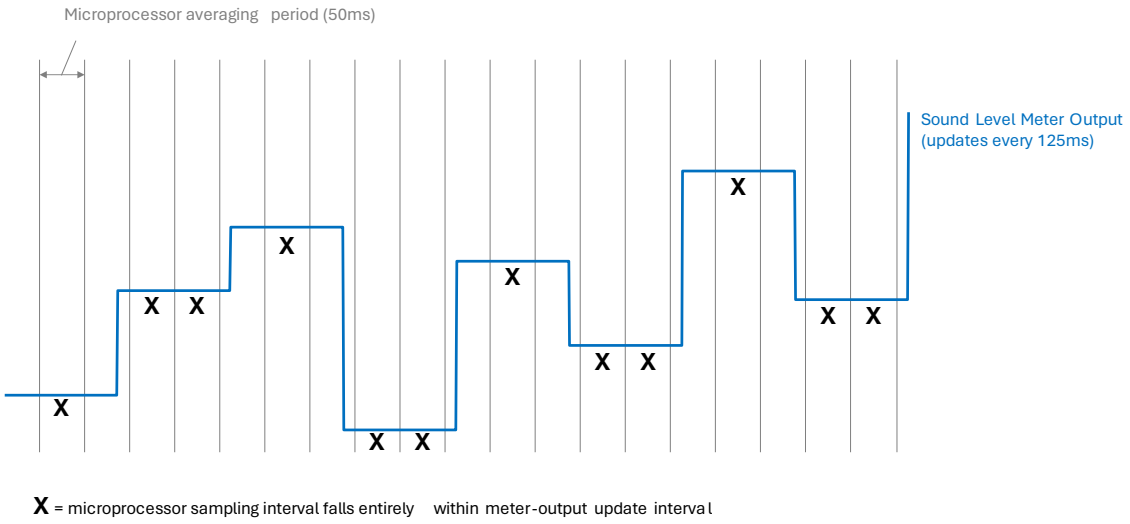
Data Acquisition and Processing

The analog output signal from the sound level meter is connected to the 10-bit analog-to-digital converter (ADC) input of the ES8266, which converts 0 to 3.3V readings from the sound level meter to an integer value between 0 and 1,024. The ESP8266 is programmed to repeatedly scan the ADC pin and calculate various decibel statistics at one-second and one-minute aggregation periods, as follows:

1. The voltage output of the meter is scanned via the ADC pin of the microprocessor at a rate of approximately 1,000 samples per second, with raw ADC integer output values counted and summed into 50-ms and 1-second accumulation bins.
2. Every 50 milliseconds (by the ESP8266 internal clock):
    a. The 50-ms average ADC value is calculated and converted to the equivalent decibel level using individually calibrated slope and offset factors (see *Calibration*)
    b. The most recent 50-ms decibel value is checked against prior minimum and maximum values within the last second, and the 1-second min and max values are updated as needed.
    c. The 50-ms decibel value is converted to a sound pressure level proportional value ($10^{(dBA/10)}$) and added to a 1-second SPL-weighted decibel accumulator.
    d. The DS3231 RTC is polled to see if the second of the minute or the minute of the hour has changed.
3. Once per second (per the 50-ms polls of the RTC), the following decibel values (rounded to the nearest 0.1dB) are calculated and transmitted to the remote data- acquisition server to be stored in a permanent database (see Communications):
    a. 1-second average unweighted decibel level, based on the converted 1-second sum and count of raw ADC readings.
    b. 1-second average SPL-weighted decibel level, based on the SPL-weighted average of the 50-ms decibel levels
    c. Minimum and maximum 50-ms decibel levels within the second
4. Once per minute, the unweighted average, minimum and maximum 1-second values for the minute and number of seconds of data in the minute are also transmitted to the data-acquisition server for storage.

The meters natively update the DC voltage output 8 times per second, but a 50-ms averaging interval is needed with the microprocessor to ensure that there is at least one 50-ms period that falls fully within every 125-ms meter update interval (Figure 5). Implemented in this way, the minimum and maximum 50-ms averaging periods will appropriately reflect the minimum and maximum $1/8^{th}$ second voltage outputs from the meter.

Figure 5. Illustration of 50-ms sampling to ensure complete coverage of 125-ms sound-level-meter update intervals.



In addition to the 1-second and 1-minute data acquisition and transmission, the code on the microprocessor synchronizes the RTC to network time every six hours.

Communications and Data Transfer

The microprocessor and smart plug are connected to the meter host's WiFi network. Data are transmitted via MQTT protocol to a remote MQTT broker running on a Raspberry Pi 4 single-board computer. Table 1 and Table 2 show the data payloads for the 1-second and 1-minute data, respectively. A Python script on the Raspberry Pi polls the MQTT broker, ingests incoming data, applies a secondary calibration adjustment (see Calibration) and inserts the data into an InfluxDB 1.8 database instance also running on the Raspberry Pi.

Table 1. Description and example of 1-second MQTT data-transmission payload.

401716974521052305180529 0523			
A	*B*	*C*	*D* *E* *F*
Item	**Start position**	**End Position**	**Description**
A	1	2	Microprocessor code version number
B	3	12	Unix Epoch timestamp
C	13	16	Unweighted mean 1-second dBA (x10)
D	17	20	Minimum 50-ms dBA (x10)
E	21	24	Maximum 50-ms dBA (x10)
F	25	28	SPL-weighted mean 1-second dBA (x10)

Table 2. Description and example of 1-minute MQTT data-transmission payload.

401716974879060053605200596 A B C D E F			
Item	**Start position**	**End Position**	**Description**
A	1	2	Microprocessor code version number
B	3	12	Unix Epoch timestamp
C	13	15	Number of seconds of data
D	16	19	Unweighted mean 1-minute dBA (x10)
E	20	23	Minimum 1-second unweighted dBA (x10)
F	24	27	Maximum 1-second unweighted dBA (x10)

Additional administrative MQTT messages are transmitted upon microprocessor boot up, clock re-synchronization and (once per day) to transmit the local WiFi RSSI signal strength.

The smart plug allows for remote manual (via phone app) or automated (via IFTTT) power cycling of the monitoring station. Automated power cycling occurs when a Python script on the main MSNSound Raspberry Pi detects that no data has been received from a meter for one minute.

Calibration

Each monitoring station (sound level meter + microprocessor) is individually calibrated using a combination of a 94/114 dB calibrator (REED R8090) and a Type 1 sound level meter (Sper Scientific Model 840015), each with a NIST-traceable calibration certificate.

The calibration steps are as follows:

1. A custom 3D printed coupler with attached earbud is fitted to the microphone extension of the field sound level meter and an audio file of a 1kHz tone at 6 approximately evenly spaced decibel levels between 60 dB and 120 dB is played while the raw ADC output of the microprocessor is logged for about 10 seconds at each level.

2. The coupler is attached to the Type 1 sound meter and the audio file is replayed at the same volume setting, with the dBA levels for the Type 1 meter manually recorded.

3. The dedicated calibrator is attached to the field sound level meter, and the raw ADC output of the meter is logged for 10 to 30 seconds at the fixed 94- and 114-dB output levels of the calibrator.

4. The calibrator is attached to the Type 1 meter to check for agreement at 94 and 114 dB.

These steps yield two sets of data: (1) average ADC output at six decibel levels spanning 60 to 120 dB from the audio file and custom coupler; and, (2) average ADC output at two decibel levels (94 and 114

dB) from the calibrator. The ADC output for the audio-file data is highly linear with recorded decibel levels from the Type 1 meter, but because the custom coupler does not have the same acoustic properties of a dedicated calibrator (which are designed to be Helmholtz resonators), the overall calibration fit from the audio-file data often deviates somewhat from the two-point calibration fit obtained from the calibrator. However, the audio-file data provides data across a wide range of decibel levels, while the calibrator yields only two calibration points that are at high decibel levels.

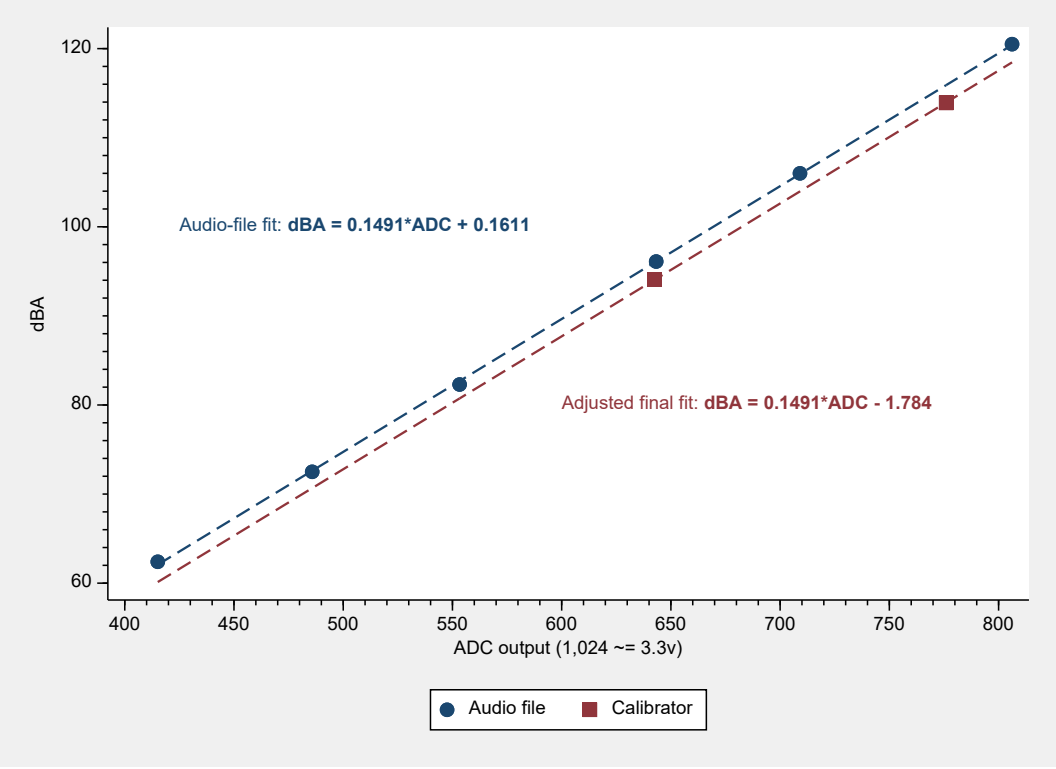
To make the best use of the available data, the following process is used to obtain a final calibration fit for the field meter:

1. Decibels are regressed against raw ADC output for the six-step audio-file data set. The slope term from this regression establishes the basic decibels per volt (or more precisely, per integer ADC output) response slope for the field unit.

2. The offset term (constant) from the above regression is then adjusted to minimize the error at the 94- and 114-dB levels recorded from the calibrator, which has greater absolute measurement accuracy.

Figure 6 shows a typical example of a calibration fit.

Figure 6. Example of a meter calibration fit.



The initial calibration fit procedure yields ADC-to-dBA slope and offset terms that are hard-coded into the configuration file for each meter and applied at the time of measurement and data transmission. Subsequent recalibration of meters in the field uses the same fitting process, but secondary slope and

offset terms are applied to the field-transmitted dBA data and not to the raw ADC values. This secondary adjustment takes place when the data are ingested at the remote server.

Aircraft Tracking

Hardware

Aircraft tracking is accomplished using a Raspberry Pi 3B+ single-board computer with a FlightAware Pro Stick Plus and external 1090 MHz antenna to receive ADS-B Out signals from aircraft within line of site of the antenna. The receiver and roof-mounted antenna are located 2.6 statute miles south and 0.49 miles west of the threshold for Runway 36 at an approximate altitude of 880 feet MSL.

Data Acquisition and Processing

The system is set up as a feeder for the ADS-B Exchange flight-tracking service, with the Raspberry Pi running software from a downloaded software image for that purpose. A key software subcomponent of that system is a variant of DUMP1090, which is an ADS-B decoder for reading received ADS-B out radio signals from the Flight Aware Pro Plus device. Key data ADS-B data elements for the purpose here include a hex code identifying individual aircraft, latitude, longitude, altitude, speed and track heading.

DUMP1090 continually updates received aircraft information and makes the information available in a time-stamped JSON-format file at a specific port on the Raspberry Pi 3B+ (Figure 7). A Python script on the main Raspberry Pi 4 computer for MSNSound polls the ADS-B Exchange feeder once per second and stores the received data in an InfluxDB 1.8 database.

Figure 7. Example of a JSON data element for an ADS-B signal received from an aircraft taxiing at the DCRA.


```
{
      "hex": "a4773f",
      "squawk": "5673",
      "flight": "N387W   ",
      "lat": 43.134727,
      "lon": -89.331527,
      "nucp": 7,
      "seen_pos": 27.9,
      "altitude": "ground",
      "track": 2,
      "speed": 11,
      "category": "A1",
      "mlat": [],
      "tisb": [],
      "messages": 4,
      "seen": 27.9,
      "rssi": -27.7
    }
```