Using the AudioMoth GPS Board and Hat

theteam@openacousticdevices.info

23rd June 2024

Many AudioMoth applications require recordings to exhibit accurate timings in order to synchronise acoustic events with other devices (such as video cameras) or to localise the source of a sound across recordings from multiple AudioMoth devices. From firmware version 1.7.0 all AudioMoth devices greater than 1.0.0, and all MicroMoth and AudioMoth Dev devices, are compatible with external GPS receivers, and from firmware version 1.10.0 the GPS location will be stored in each WAV file as part of the GUANO metadata.

1 Time Setting from GPS

Accurate time is kept on AudioMoth devices using two crystal oscillators: a 48 MHz high-frequency crystal oscillator (HFXO) that is used whilst AudioMoth is actually making recordings, and a 32768 Hz low-frequency crystal oscillator (LFXO) that is used by the real-time clock (RTC) in order to start recordings at the correct time. These oscillators have quoted accuracy of ±10 and ±20 ppm respectively. However, due to manufacturing variance, and the lack of temperature compensation, their frequencies will deviate slightly when in use. This drift is most apparent in the LFXO and can lead to the start time of an AudioMoth recording drifting by up to 2 seconds per day.

In many applications, this drift is of little consequence, and adequate time synchronisation can be ensured by simply updating the AudioMoth time prior to deployment (either when the device is configured or in the field using the smartphone app). When greater accuracy is required, it is now possible to use an external GPS receiver to automatically update the AudioMoth time prior to each scheduled recording period. The

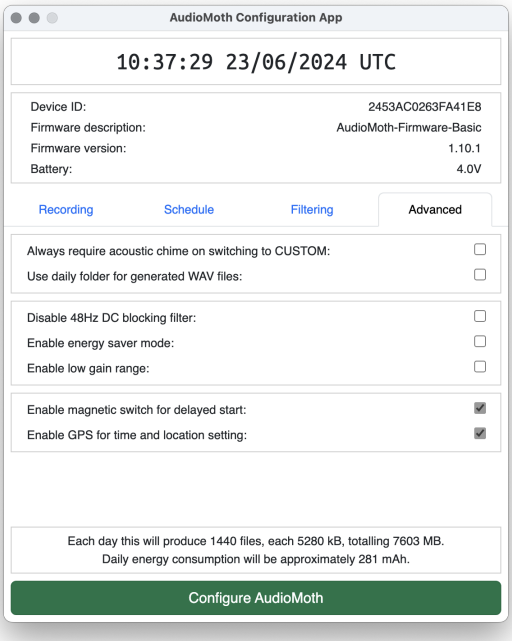

Figure 1: *AudioMoth Configuration App showing the GPS options under the 'Advanced' pane.*

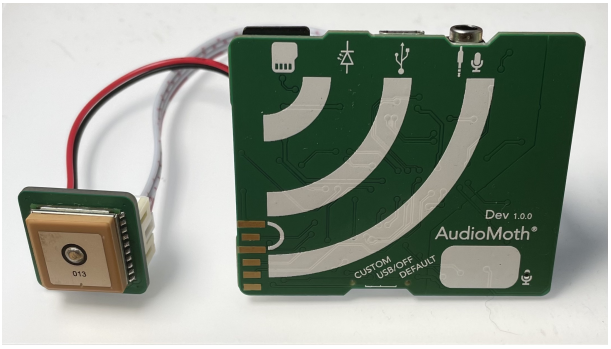

Figure 2: *AudioMoth GPS Board connected to an AudioMoth Dev board with standard JST PH jumper cables.*

Figure 3: *AudioMoth GPS Hat soldered to an AudioMoth 1.2.0 (in this case, also with a 3.5mm socket).*

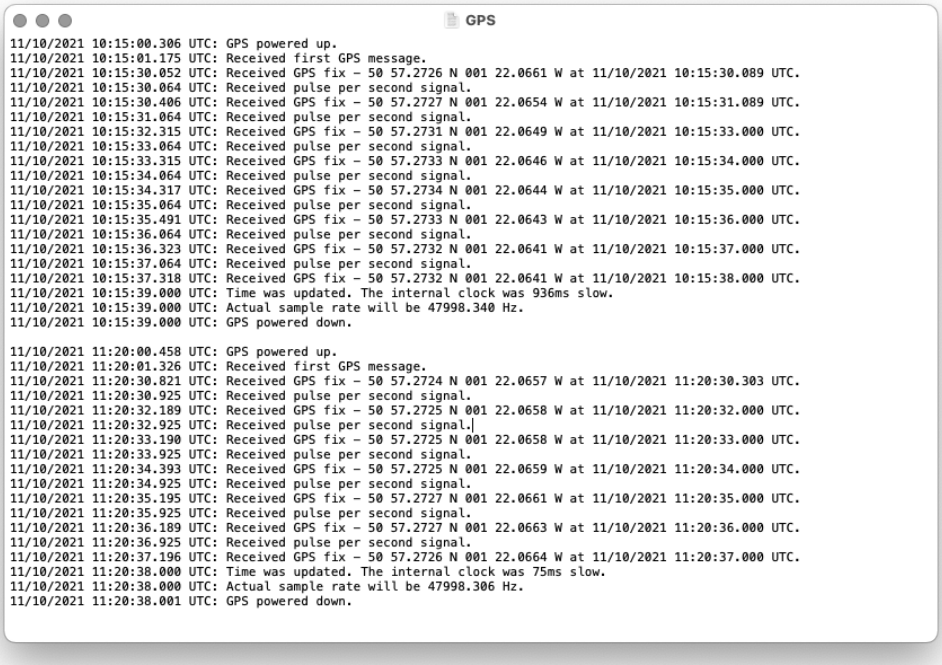

Figure 4: `GPS.TXT` file written to the SD card by AudioMoth A to record the results of the time setting process.

standard firmware supports this by automatically powering on the GPS five minutes before each recording period, then decoding the 9600 baud serial navigation messages (specifically, NMEA-0183 RMC messages), and detecting the pulse per second (PPS) output, from a connected GPS receiver to set the time to a precision of one millisecond. Enabling this behaviour is done by simply checking the *Enable GPS for time setting* option in the AudioMoth Configuration App (see Figure 1).

The results of the time setting process are recorded in a `GPS.TXT` file on the SD card (see Figure 4). This file is updated each time the GPS is powered up. It reports the navigation messages and pulse per second signals received before the time is set, the offset of the real-time clock, and the actual sample rate that will be achieved due to any HFXO frequency offset. The location recorded by the GPS module is also written to each WAV file as part of the GUANO metadata.

2 AudioMoth GPS Receivers

There are two external GPS add-on receivers available for AudioMoth products: (i) AudioMoth GPS Board used with AudioMoth Dev, and (ii) AudioMoth GPS Hat used with AudioMoth 1.1.0, 1.2.0 or MicroMoth. These receivers are compatible with a wide range of GPS modules, such as the PA1616S[1] module that uses just GPS satellites to calculate position fixes, and the PA1616D[2] module that by default uses both GPS and GLONASS satellites.

Note that neither of the variants is compatible with the AudioMoth IPX7 Waterproof Case.

2.1 AudioMoth GPS Board

The AudioMoth GPS Board (see Figure 2) plugs directly into the power-out and GPIO sockets of the AudioMoth Dev board using JST-PH jumper cables[3][4] (see Figure 5). The full schematic of the AudioMoth GPS Board is shown in Figure 12.

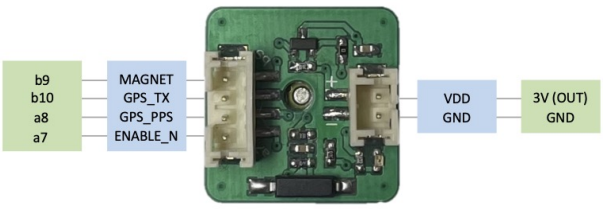

Figure 5: *AudioMoth GPS Board connections showing GPS (blue) and AudioMoth (green) connection labels.*

[1] `https://www.cdtop-tech.com/products/pa1616s`
[2] `https://www.cdtop-tech.com/products/pa1616d`
[3] `https://www.adafruit.com/product/4714`
[4] `https://www.adafruit.com/product/3568`

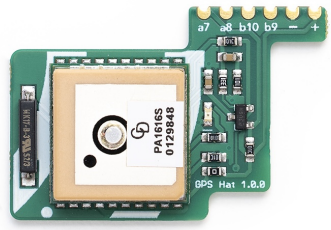

Figure 6: *AudioMoth GPS Hat.*

2.2 AudioMoth GPS Hat

The AudioMoth GPS Hat (see Figure 6) is electronically identical to the AudioMoth GPS Board except it is in a different form factor. Instead of making connection by JST-PH connectors, it is soldered directly to the exposed GPIO pads on AudioMoth 1.1.0 or 1.2.0 (see Figure 3). The board may also be soldered directly to the GPIO pads on the MicroMoth board.

3 Delayed Start

The AudioMoth GPS receivers also contain a magnetic switch which is used by the standard firmware to support delayed start. This can be enabled by checking the *Enable magnetic switch for delayed start* option in the AudioMoth Configuration App (see Figure 1).

When enabled, switching to CUSTOM will put the AudioMoth into a delayed start mode, flashing the green LED once every four seconds, rather than the normal two seconds when in CUSTOM mode waiting to start a recording. In this state, the AudioMoth uses very little power and will wait until activated by a magnet held near the magnetic switch before switching into CUSTOM mode fully. This allows an AudioMoth to be placed inside a sealed housing, with its batteries connected and the switch set to CUSTOM, prior to transport to the deployment site, where it can then be fully switched into CUSTOM mode from outside the housing with a magnet.

This feature can be used with or without GPS time setting. In the former case, the AudioMoth will go to delayed start immediately and on activation by the magnet it will power up the GPS receiver to set the time and also to determine the location. In the latter case, the behaviour of the device depends on whether or not the time has already been set on the AudioMoth (as per normal AudioMoth operation). If the time has been set, the AudioMoth will immediately enter delayed start mode. If the time has not been set, the AudioMoth will enter acoustic configuration mode to allow the smartphone app to be used to set the time. On setting the time successfully, the AudioMoth will then switch to delayed start mode.

The magnetic switch can also be used to stop recordings and to return to delayed start mode at any time. The AudioMoth will indicate that it is stopping a recording by rapidly flashing the red LED ten times, and will report this in the WAV file header comments.

4 Source Localisation

The GPS-synchronised recordings from multiple devices can be used to localise the source of an acoustic event. To evaluate this process we deployed four AudioMoth Dev boards (A, B, C and D), each connected to an AudioMoth GPS Board, at the corners of a football (soccer) pitch (see Figure 8). At the centre spot of the pitch, we simulated the noise of an impulsive gunshot using a mallet and a dustbin lid (see Figure 9). We set the AudioMoth with two scheduled recording periods, one from 10:20 to 11:20 UTC and one from 11:25 to 11:30 UTC. We simulated gunshots at minute intervals from approximately 10:20:11 to 11:19:11 UTC.

Figure 7 shows an example event recorded at approximately 10:30:11 UTC by AudioMoth A. This event is approximately 611 seconds from the start of the recording. Using an amplitude threshold we can locate the start of this event to within a few tenths of a millisecond. For the example shown in Figure 7, we estimate the time of arrival (TOA) of the acoustic signal to be 611.3618 seconds from the start of the recording.

4.1 Time of Arrival Correction

We can improve these time of arrival measurements by correcting for the frequency offsets of the HFXO and the LFXO. To correct the former we note that, as per Figure 4, AudioMoth A set its time from the GPS immediately before making the recording at 10:15:39 UTC and after making the recording (triggered by the scheduled recording period at 11:25 UTC) at 11:20:38 UTC. In addition to setting the time, the AudioMoth used the accurate PPS output from the GPS to measure the frequency of the 48 MHz oscillator and reported the actual sample rate that will be achieved. In this case, 47998.340 Hz before the recording and 47998.306 Hz after the recording. Taking the average of these as 47998.323 Hz, we can correct the time of arrival such that $48000 / 47998.323 \times 611.3618 = 611.3832$ seconds from the start of the recording.

To correct for the frequency offset of the LFXO we note, again from Figure 4, that the internal real-time clock of AudioMoth A was found to be 75ms slow when the time was set after the recording. Thus we can calculate a correction based on the drift of the internal clock from the time that it was set at 10:15:39 UTC to the time that the recording started at 10:20:00 UTC. In this case, the correction is $261 / 3899 \times 75 = 5.0$ ms, where 261 is the interval in seconds between the first time setting and the start of the recording, and 3899 is the interval in seconds between the first and second time settings.

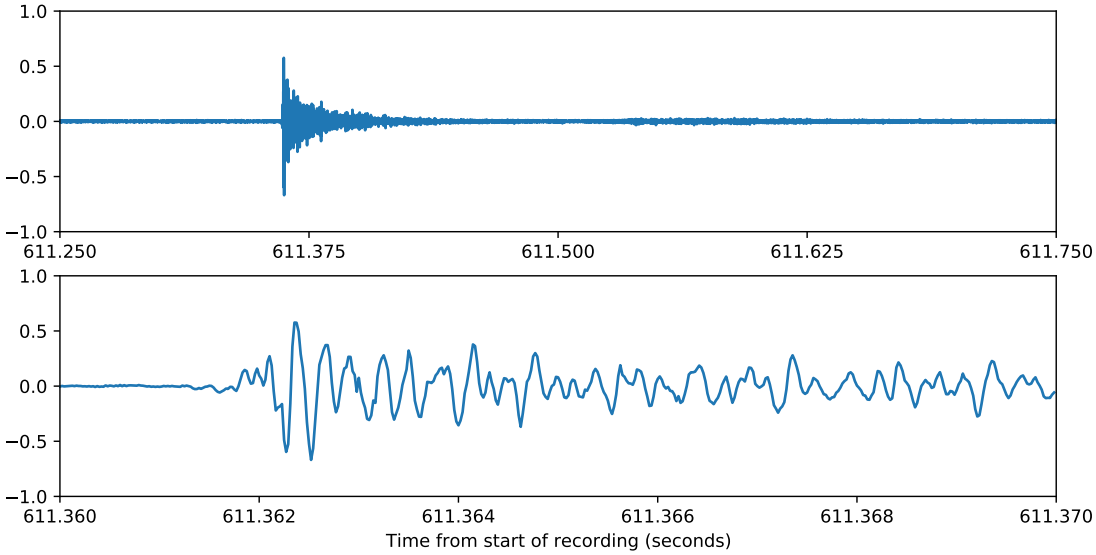

Figure 7: *Example acoustic event recorded by AudioMoth A.*

Correction Type	Offset (ms)	Time (s)
Uncorrected	-	611.3618
Corrected HFXO	+ 21.4	611.3832
Corrected HFXO and LFXO	+ 26.4	611.3882

Table 1: *Example time of arrival corrections.*

	Position Error (m)			
Correction Type	Min	Mean	Median	Max
Uncorrected	1.6	12.9	13.5	22.2
Corrected HFXO	1.3	1.7	1.7	1.9
Corrected HFXO and LFXO	0.1	0.7	0.8	1.0

Table 2: *Source localisation results over all sixty simulated gunshots.*

The corrections for this example are shown in Table 1. Repeating this for all 60 acoustic events, for each of the four AudioMoths, gives a set of time of arrival measurements that can be used to localise the source.

4.2 Source Position Calculation

Given the locations of each AudioMoth,[5] and the time of arrival of an acoustic event at each AudioMoth, it is possible to calculate the position of the source. The time difference of arrival (TDOA) between any two AudioMoth devices defines a hyperbolic curve on which the source is located. In this case, we have four receivers and a 2-dimensional surface on which the source is located. Thus the problem is over-determined (there are more equations than unknowns) and we can use a least squares optimisation approach to find the most likely source location given the uncertainty in the time of arrival of the acoustic events. Code to perform this calculation is shown in Figure 11.[6]

The results of this process are shown in Figure 10. Figure 10a shows positions calculated with uncorrected time of arrival measurements, while Figure 10b corrects for the frequency offset of the HFXO, and Figure 10c corrects for the frequency offsets of both the HFXO and LFXO. Figure 10a shows the steady drift in position over the one-hour recording period. This is removed by correcting for the frequency offset of the HFXO in each AudioMoth to leave a small fixed offset which is then removed by correcting for the frequency offsets of both the HFXO and LFXO in each AudioMoth. The final accuracy achieved is shown in Table 2 and indi-

[5]These locations, relative to the centre sport of the pitch, were surveyed using two Reach RS+ RTK GNSS receivers to provide centimeter level accuracy.

[6]The code is based on the approach described in `https://github.com/jurasofish/multilateration` and makes use of a standard Python optimisation library.

cates a mean error in source localisation over the 60 measurements of better than one meter.

Figure 8: *AudioMoth and GPS board mounted on poles at each corner of the football (soccer) pitch.*

Figure 9: *Dustbin lid and mallet used to simulate impulsive gunshot sounds at the centre spot of the football (soccer) pitch.*

5 Continuous GPS Operation

The AudioMoth firmware described here powers up the GPS receiver for approximately 20 to 60 seconds before each scheduled recording period. This minimises the energy consumption of the device but requires an estimation of the HFXO and LFXO corrections in order to localise the source. For continuous GPS operation an alternative firmware is available, called AudioMoth GPS Sync. This is a more accurate approach that powers the GPS whilst the recording is being made, tracking and recording the continuous arrival of pulse-per-second signals. This approach allows sample-level correlation of recordings across multiple devices. AudioMoth GPS Sync can be found on the Alternative Firmware section of the website[7] with more details found in the 'Using AudioMoth GPS Sync to Synchronised Recordings' application note.[8]

[7] https://www.openacousticdevices.info/gps-sync

[8] https://github.com/OpenAcousticDevices/Application-Notes/blob/master/Using_AudioMoth_GPS_Sync_to_Make_Synchronised_Recordings/Using_AudioMoth_GPS_Sync_to_Make_Synchronised_Recordings.pdf

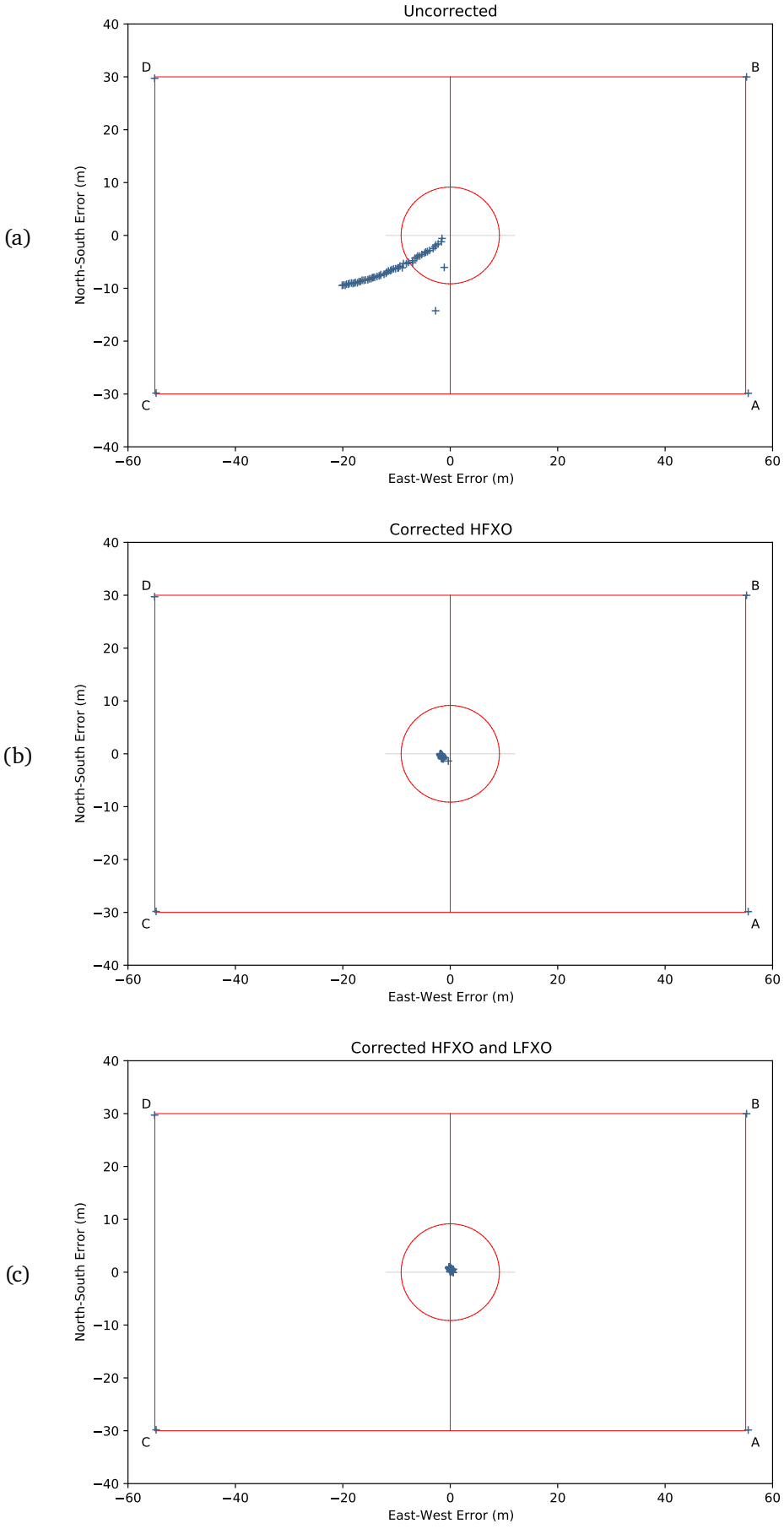

Figure 10: *Calculated source location shown against the centre spot and 10-yard (9.15 m) centre circle of the football (soccer) pitch.*

```
import numpy as np

from scipy.optimize import least_squares

velocity_of_sound = 343.0

audiomoth_positions = np.array([[ 55.47, -29.86],
                                [ 55.16,  29.98],
                                [-54.76, -29.82],
                                [-55.04,  29.72]])

time_of_arrival = np.array([[ 11.3823, 11.3797, 11.3808, 11.3804],
                            [ 71.1305, 71.1281, 71.1294, 71.1303]])

assert np.shape(audiomoth_positions)[0] == np.shape(time_of_arrival)[1]

number_of_events = np.shape(time_of_arrival)[0]

number_of_dimensions = np.shape(audiomoth_positions)[1]

initial_guess = np.zeros(number_of_dimensions)

position_estimates = np.zeros((number_of_events, number_of_dimensions))

for i in range(number_of_events):

    c = np.argmin(time_of_arrival[i])

    t_c = time_of_arrival[i][c]
    p_c = np.expand_dims(audiomoth_positions[c], axis=0)

    all_t_i = np.delete(time_of_arrival[i], c, axis=0)
    all_p_i = np.delete(audiomoth_positions, c, axis=0)

    def eval_solution(x):
        return np.linalg.norm(x - p_c, axis=1) \
                - np.linalg.norm(x - all_p_i, axis=1) \
                + velocity_of_sound * (all_t_i - t_c)

    result = least_squares(eval_solution, initial_guess, method='lm')

    position_estimates[i] = result.x

print(position_estimates)
```

Figure 11: *Python code to calculate source location from time of arrival measurements via least squares optimisation.*

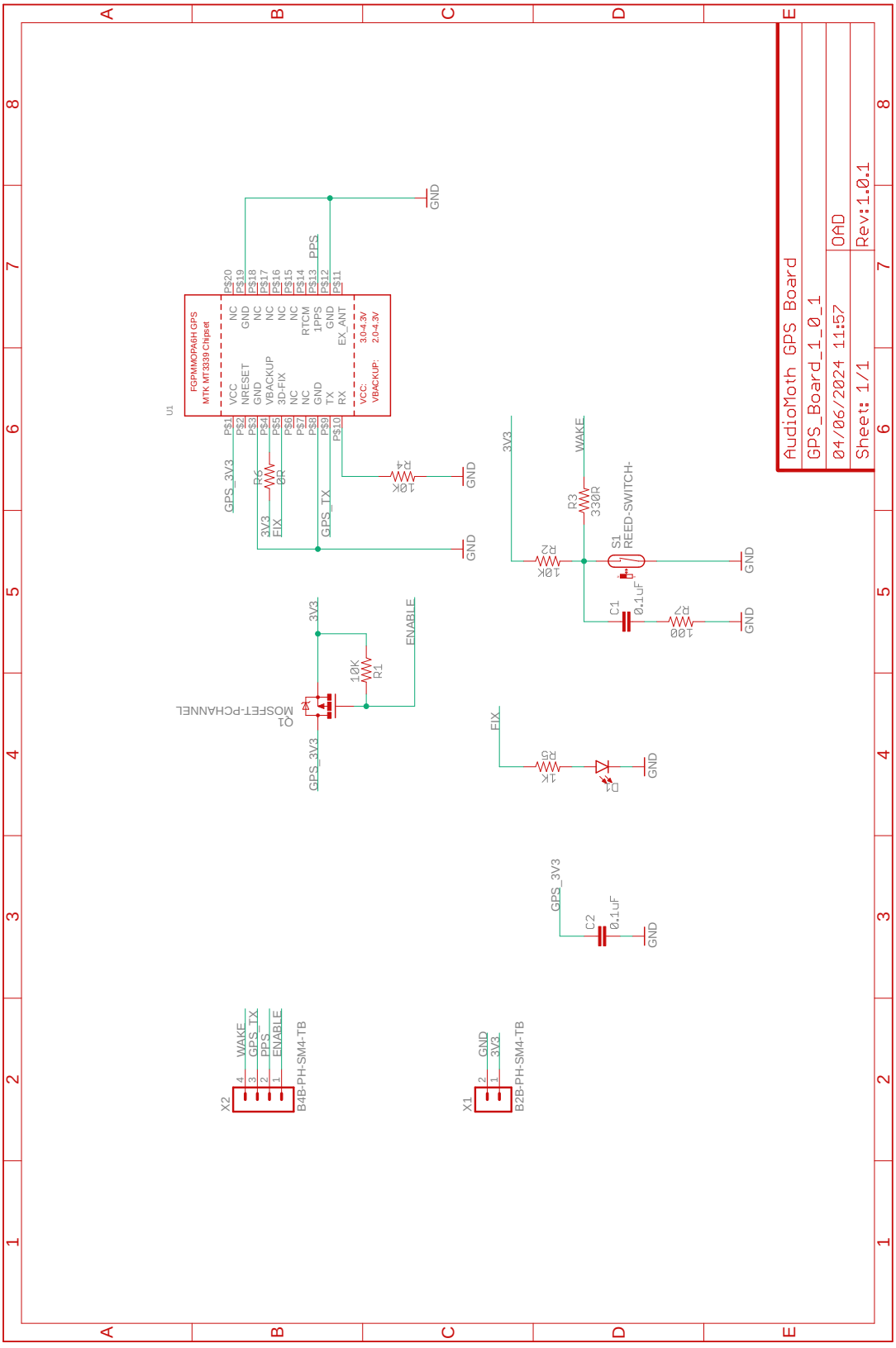

Figure 12: *Schematic of the AudioMoth GPS Board.*