Using the AudioMoth USB Microphone Firmware and Hardware

theteam@openacousticdevices.info

23rd October 2023

The standard AudioMoth firmware (AudioMoth-Firmware-Basic) writes recordings to SD card and uses the USB connection for configuring the recording settings and schedule. By contrast, the new AudioMoth USB Microphone firmware (AudioMoth-USB-Microphone) allows all existing AudioMoth hardware versions to be used as configurable USB microphones with a sample rate of up to 384kHz. The received audio is identical to that recorded by the standard AudioMoth firmware and the same filter options are available. It can also be used with external microphones and hydrophones with the AudioMoth 1.2 and AudioMoth Dev variants. The AudioMoth USB microphone can be used with desktop and laptop computers, with Android smartphones, and with single-board computers, such as the Raspberry Pi.

1 Firmware and Desktop Configuration App

The AudioMoth USB Microphone firmware can be installed on any AudioMoth device (including uMoth and AudioMoth Dev variants) by downloading it from the AudioMoth website[1], and using the 'Use Local File' option within the AudioMoth Flash App[2]to update the device firmware.

When the AudioMoth is now switched to USB/OFF it will communicate with the AudioMoth USB Microphone App (see Figure 1). This allows the sample rate, gain and various filter options to be set. These settings are persistent and will be retained by the AudioMoth when powered down.

When the AudioMoth is now switched to DEFAULT or CUSTOM, it will enumerate as a USB microphone and appear alongside other audio input devices on the computer to which it is connected. The device name is augmented with the configured sample rate. When in CUSTOM mode, all the configured settings are applied, and the red LED flashes. When in DEFAULT mode, only the configured sample rate and gain are applied, and the green LED flashes. It is possible to switch between DEFAULT and CUSTOM without the USB microphone disconnecting (by moving the switch directly from one position to the other), allowing the effect of different settings to be compared within a single recording. Similarly, external microphones may be plugged in and out, without the USB microphone disconnecting.

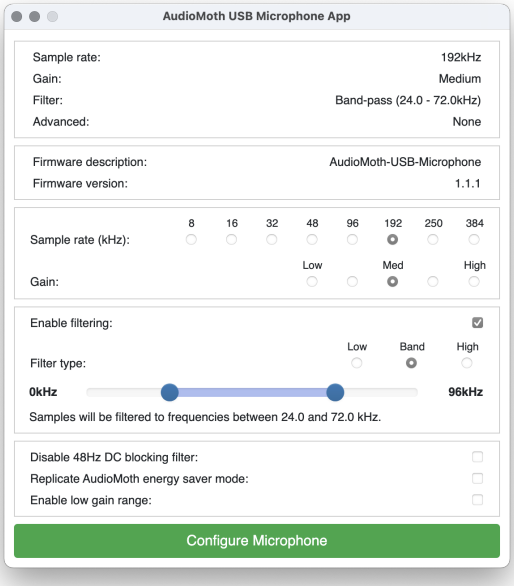

Figure 1: *AudioMoth USB Microphone App showing a configured microphone.*

The AudioMoth USB Microphone can then be used directly with any audio application. For example, Figure 2 shows an AudioMoth USB Microphone configured with a sample rate of 192kHz being used to view live sonograms of bat calls using SignalScope X in macOS.

Similarly, the AudioMoth USB Microphone can be used with many Android smartphones that support either USB On-The-Go (OTG) or USB-C. A wide range of applications can then be used to make and view recordings, including *Bat Recorder* that allows the live generation of sonograms (see Figure 3).[3]

[1] https://www.openacousticdevices.info/usb-microphone
[2] https://www.openacousticdevices.info/applications
[3] https://play.google.com/store/apps/dev?id=8274851164581402087

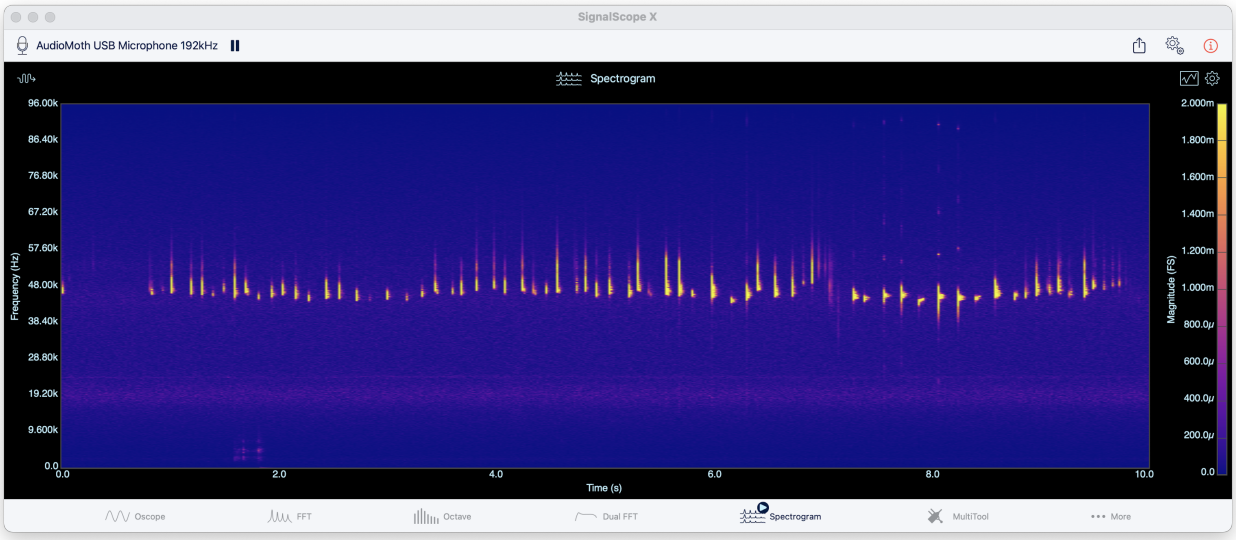

Figure 2: *Live sonogram generated using SignalScope X in macOS whilst recording bat calls on an AudioMoth USB Microphone.*

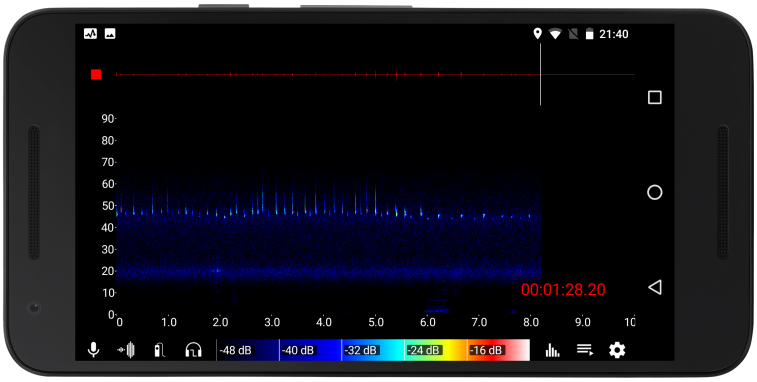

Figure 3: *Live sonogram generated using Bat Recorder on a smartphone whilst recording bat calls on an AudioMoth USB Microphone.*

Figure 4: *AudioMoth USB Microphone custom hardware.*

2 Custom Hardware

When used as a USB microphone, many of the standard AudioMoth components (such as the battery holder, external 256KB SRAM, SD card, and 32768 Hz real-time clock crystal) are no longer needed. The AudioMoth USB Microphone hardware is an AudioMoth variant with these components removed. It is designed to specifically run the AudioMoth-USB-Microphone firmware (see Figure 4) and uses an identical audio front-end to the AudioMoth 1.2 and AudioMoth Dev variants. It has a permanently mounted 3.5mm jack to support the use of external microphones and hydrophones, and measures just 40mm x 35mm.

3 Recording With Raspberry Pi

The AudioMoth USB Microphone can be used with single-board computers, such as the Raspberry Pi, to build sophisticated autonomous recorders capable of off-loading recordings to the internet, or analysing recordings on the device itself. On a Raspberry Pi, or similar Linux-based computer, recording from the AudioMoth USB Microphone can be performed using the `arecord` command line tool, or in Python using the `pyaudio` library.

3.1 Recording from the command line

The `arecord` command line tool provides a simple way to make recordings without installing any additional software. To use `arecord`, connect the AudioMoth USB Microphone to the Raspberry Pi via USB, and put the switch in either the DEFAULT or CUSTOM position. Then run the command below to list the available audio devices.

```
> arecord -L
```

One of the entries will be direct hardware access, without conversions, to the AudioMoth USB Microphone.

```
hw:CARD=A192kHz,DEV=0
    AudioMoth USB Microphone 192kHz, USB Audio
    Direct hardware device without conversions
```

This device can then be used to make recording at the configured sample rate of the microphone (in this case 192kHz). Using the 'Direct hardware device without conversions' option ensures that the recording parameters are identical to those of the equivalent standard AudioMoth recording. The example below records for sixty seconds at 192kHz, using the standard AudioMoth 16-bit resolution, and writes the resulting audio to a file called `test.wav`.

```
> arecord -D hw:CARD=A192kHz,DEV=0
  -f S16_LE -r 192000 -d 60 test.wav
```

The command line will report that a recording is in progress.

```
Recording WAVE 'test.wav' : Signed 16 bit little
Endian, Rate 192000 Hz, Mono
```

If the computer has audio output, the resulting WAV file can be played back immediately.

```
> aplay test.wav
```

These command line tools are quite flexibly and can automatically split long recordings into multiple file and automatically rename files. More information is available through the Linux manual pages.[4][5].

3.2 Recording using Python

When more control of recording process is required, or the application is intending to perform signal processing on the samples directly, it is also possible to use Python to immediately access data from the AudioMoth USB Microphone. To do so, we use the `pyaudio` library, and must first install it.

```
> sudo apt-get install python3-pyaudio
```

The example code (`record.py`) shown in Figure 5 finds the connected AudioMoth USB Microphone, records for sixty seconds at 192kHz and writes the resulting audio to a file called `test.wav`. The code can be run from the command line as below.

```
> python3 record.py 2>/dev/null
```

Note that this instruction redirects `stderr` to `dev/null` to remove some of the unnecessary error reporting that `pyaudio` generates as it attempts to open audio devices. The code captures the full recording in memory before transferring to the file. Other approaches could process the incoming audio in real-time, writing processed results to file, or off-loading the recordings to the internet.

[4] `https://manpages.org/arecord`
[5] `https://manpages.org/aplay`

```
import pyaudio
import wave

audio_format = pyaudio.paInt16
number_of_channels = 1
sample_rate = 192000
chunk_size = 4096
duration = 60

filename = 'test.wav'

# Create pyaudio instance and search for the AudioMoth

device_index = None

audio = pyaudio.PyAudio()

for i in range(audio.get_device_count()):
    if 'AudioMoth' in audio.get_device_info_by_index(i).get('name'):
        device_index = i
        break

if device_index == None:
    print('No AudioMoth found')
    exit()

# Create pyaudio stream

stream = audio.open(format=audio_format, rate=sample_rate, channels=number_of_channels, \
                    input_device_index=device_index, input=True, \
                    frames_per_buffer=chunk_size)

# Append audio chunks to array until enough samples have been acquired

print('Start recording')

data = []

total_samples = sample_rate * duration

while total_samples > 0:
    samples = min(total_samples, chunk_size)
    data.append(stream.read(samples))
    total_samples -= samples

print('Finished recording')

# Stop the stream, close it, and terminate the pyaudio instance

stream.stop_stream()
stream.close()
audio.terminate()

# Save the audio data as a WAV file

wavefile = wave.open(filename, 'wb')
wavefile.setnchannels(number_of_channels)
wavefile.setsampwidth(audio.get_sample_size(audio_format))
wavefile.setframerate(sample_rate)
wavefile.writeframes(b''.join(data))
wavefile.close()
```

Figure 5: *Python code, `record.py`, to record from the AudioMoth USB Microphone.*